

**КУРСОВОЙ ПРОЕКТ**

**по МДК 01.01** Разработка программных модулей

**специальности** 09.02.07 Информационные системы и программирование

**на тему:** Разработка программного модуля API для обработки данных роботами

Выполнил студент

группы 4ИС3010

Федотов В.В

(Фамилия, инициалы)

Проверил преподаватель

Тарасов.А.Д

(Фамилия, инициалы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Москва, 2022

Содержание

Содержание 2

Введение 3

1 Теоретическая часть 5

1.1 Описание предметной области 5

1.2 Постановка задачи 7

2 Практическая часть 10

2.1 Описание сценариев использования 10

2.2 Проектирование модели данных 10

2.3 Проектирование интерфейса 10

2.4 Разработка программы 10

2.5 Тестирование программы 10

Заключение 11

Список использованных источников 12

Приложения 13

Введение

С течением времени каждый вид деятельности человека стал приобретать черты автоматизации процесса производства, что привело к появлению механизмов, которые научились обрабатывать информацию и на основе этой информации делать определенные действия.

Такие механизмы могут обладать малой вычислительной мощностью, поэтому не могут быть обработаны все возможные форматы информации, которые потребуются программисту для реализации того или иного алгоритма действий или предоставлять только необработанную информацию, что сильно замедляет процесс разработки программного продукта.

Данная тема актуальна, так как разработка программного модуля API позволит решить эту проблему.

API – это специальный интерфейс для разработчиков, позволяющий общаться со сторонней программой в своей программе.

Практическая значимость заключается в использовании программного модуля другими программистами для разработки собственного программного обеспечения, которое позволит им управлять данными механизмами.

Целесообразность данной работы заключается в том, что программный модуль API позволит реализовать свои программные продукты на его основе.

Цель данной работы – это разработка программного модуля API для обработки данных роботами.

Задачи данной работы:

* выяснить, каким образом механизмы отправляют информацию и что нужно для её обработки;
* создать алгоритмы обработки этих данных и определить в каком формате они должны быть реализованы;
* Определить данные алгоритмы в составе API.

Гипотеза исследования – если API будет реализовано, то оно поможет ускорить разработку программного обеспечения, основанного на нём.

Объект исследования является механическая площадка с роботизированными устройствами.

Методом исследования является анализ данных, поступающих с роботов.

1. Теоретическая часть
   1. Описание предметной области

Робот – это автоматическое устройство, предназначенное для осуществления различного рода механических или математических операций, которое действует по заранее заложенным в него действиям. Каждый из роботов может выполнять несколько задач сразу. Обычно он получает информацию о состоянии окружающего пространства посредством датчиков, установленных отдельно или встроенных в него, а также может получать команды от человека через специальное программное обеспечение. На рисунке 1 представлен робот-манипулятор с вакуумным захватом.

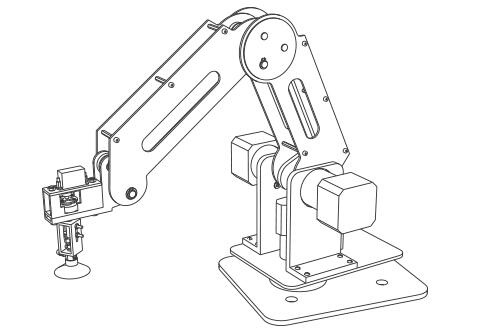


Рисунок 1 – Робот-манипулятор с вакуумным захватом

Датчики могут собирать различную информацию, передающейся в текстовом виде и обрабатывающейся специальной комбинацией инструкций, такая комбинация инструкций, выполняемых последовательно, именуется программой.

Каждый робот требует своей собственной уникальной программой или полноценным программным обеспечением для совершения действий, в которое может входить визуальный интерфейс или возможность внешнего управления с помощью механических внешних устройств.

Роботы так же могут соединяться между собой в одну сеть, такой сетью в современном мире может выступать конвейерная лента, площадка для взаимодействия механизмов.

В качестве объекта будет выступать площадка, включающая в себя две четырёхцветные сигнальные лампы, робот-политайзер и робот-манипулятор, а также удаленный терминал для управления площадкой.

Четырёхцветная сигнальная лампа – это устройство для отображения фазы выполнения роботом поставленной задачи. Лампа содержит в себе цвета: зеленый, синий, оранжевый и красный и подключается к площадке различными способами по сети. Возможными способами может быть WIFI сеть или проводное соединение. Данная лампа может только принимать четыре разных значения, смена который определит включен или выключен определенный цвет лампы. Цвета лампы могут отображать не только фазу выполнения задачи роботом, но и могут быть произвольно включены или перепрограммированы.

Удаленный терминал – это устройство для управления и отправки данных на роботов. Он содержит четыре лампы для отображения состояния фазы выполнения роботом поставленной задачи, дисплей для отображения IP адресов, нужных для подключения к нему, и произвольной информации, которую может задать программист или оператор платформы, три кнопки: зеленую, красную и желтую, для использования их в непосредственном удаленном управлении любым элементом на площадке, а так же рычаг, выполняющий функции “зажатой кнопки” и джойстик для непосредственного управления вращением робота при прямом подключении терминала к нему. Данное устройство может рассылать данные в сеть, из которых можно получить IP адрес устройства, состояние рычага и количество нажатий на красную, зеленую или желтую кнопку за время работы устройства. Для управления лампами на терминале может быть принята текстовая информация. Дисплей и джойстик не обладают принимаемой или отправляемой информацией.

Робот-политайзер – это устройство для совершения механических действий. Данный тип робота обладает манипулятором с предметами – вакуумной присоской, но может иметь и другие виды инструментов, к примеру карандаш, и может вращаться на 360 градусов. Он может рассылать данные о нагреве своих 6 двигателей, расположенных на подвижных местах робота, состоянии вакуумной присоски, а также может принимать данные для управления по декартовой системе координат и включения или выключения вакуумной присоски.

Робот-манипулятор – это устройство для совершения механических действий. Этот тип роботов обладает манипулятором с предметами, который захватывает предмет и может его перемещать. Обладает пятью двигателями на местах смещения роботов и постоянно рассылает данные о их состоянии, в которых так же входит температура, управляется по декартовой системе координат.

Для взаимодействия с роботами выше используется протокол сети - User Datagram Protocol (UDP). Данный протокол уникален тем, что данные приходят быстрее, чем в остальных случаях за счет отсутствия проверки на целостность данных. Такой протокол может быть использован для реализации систем, не требующих получения данных в последовательном порядке, а перезаписывающий их. Каждое из перечисленных устройств использует этот протокол и отправляет данные в сеть раз в несколько миллисекунд, что обеспечивает наиболее быстрое соединение.

Чтобы обрабатывать данные приходящие по этому протоколу программисты разрабатывают специальные интерфейсы для межпрограммного обмена, которые называются API.

API – интерфейс для упрощения создания программ, который может позволяет программистам уменьшать время разработки программы за счет предоставления разработчикам программы специальных, подготовленных заранее, способов общения с программой на более низком уровне. Помимо использования для ускорения времени разработки оно так же может быть использовано для доступа к каким-то данных напрямую.

Чтобы работать с API программист должен использовать его методы. Метод — это набор инструкций (действий программы), созданный для повторного его использования. Каждый метод API может совершать разные действия и принимать разные данные от программиста для их обработки. Методы могут возвращать какое-то значение, получать или изменять его. API создают именно для использования подобных методов на постоянной основе, не изменяя базовую программу.

Зачастую для разработки API используют массивы. Массивы – это тип данных, содержащий в себе набор элементов определенного типа данных. Массивы используются для упрощения передачи данных и уменьшения засорения кода от лишних переменных, массивы бывают с числовыми ключами (последовательные массивы) и ассоциативные массивы, ключами которых могут быть буквы, символы и целые фразы. Массив хранится в памяти таких образом, что можно получить значение любой ячейки без создания дополнительных переменных. Таким образом можно упростить код и не создавать слишком больших программ. Одно из самых лучших свойств массивов – возможность проходить по всем ячейкам. На рисунке 3 указан абстрактный вид одномерного массива.



Рисунок 2 – Абстрактный вид одномерного массива

* 1. Постановка задачи

Для упрощения взаимодействия с роботами необходимо реализовать API в формате библиотеки для обработки данных роботами. Каждый пользовательский метод библиотеки должен быть публичным для вызова API из других программ.

Для робота-политайзера нужно реализовать методы:

* получения информации о данных с моторов массивом – данный метод возвращает массив с информацией о температуре двигателей роботов;
* получения информации о данных с моторов по номеру мотора – данный метод запрашивает номер мотора и возвращает его температуру;
* установки положения робота – метод, который запрашивает три целочисленные цифры, преобразует их в координаты по порядку и отправляет их на робота.
* установки положения робота изменяя только одну координату - метод, запрашивает номер координаты и её значение, при вызове изменяет координату и отправляет её на робота;
* получения информации о текущих координатах робота – собирает сохраненные координаты робота и выводит их в формате массива;
* получения информации о состоянии захвата робота – возвращает данные о состоянии захвата робота (включен или выключен).

Для робота-манипулятора нужно реализовать методы:

* получения информации о данных с моторов массивом. Данный метод возвращает массив с информацией о температуре двигателей роботов;
* получения информации о данных с моторов по номеру мотора-метод, который запрашивает три целочисленные цифры, преобразует их в координаты по порядку и отправляет их на робота.;
* установки положения робота – устанавливает положение робота запрашивая все координаты у программиста;
* установки положения робота изменяя только одну координату – устанавливает положение робота запрашивая одно значение у программиста;
* получение информации о текущих координатах робота – возвращает последние сохраненные координаты робота в формате массива;
* получение информации о состоянии захвата робота – возвращает информацию о том, включен захват робота или нет.

Для сигнальной четырехцветной лампы нужно реализовать методы:

* установки состояния цветов лампы отдельно – устанавливает состояние одного цвета лампы во включенное или выключенное состояние;
* установки состояния цветов ламп массивом – устанавливает цвета лампы, запрашивая у программиста информацию о том, какие лампы включены, а какие выключены;
* получение текущего состояния цветов лампы массивом – возвращает цвета лампы программисту массивом с целочисленными ключами.

Для удаленного терминала нужно реализовать следующие методы:

* получение количества нажатий на малые кнопки – возвращает информацию о количестве нажатий на малые кнопки удаленного терминала;
* получение состояния рычага – возвращает информацию о том, зажат ли рычаг (красная большая кнопка) или нет;
* установки состояния цветов ламп отдельно – устанавливает состояние определенной лампы;
* метод установки состояния цветов ламп массивом – устанавливает все значения ламп.
  + 1. Выбор технологии реализации

Для реализации поставленной задачи необходимо выбрать один или несколько языков программирования. Язык программирования – это формальный язык для записи компьютерных программ. Язык программирования определяет набор конкретный набор инструкций для общения с компьютером и позволяет разрабатывать свои инструкции на основе базовых инструкций.

Чтобы обеспечить связь между API и роботами понадобится так же локальная вычислительная сеть. Локальная вычислительная сеть– это компьютерная сеть, которая обычно покрывает небольшую территорию, например, офис компании или небольшую группу зданий.

Чтобы получать данные с роботов нужно собирать данные из локальной вычислительной сети. Каждое устройство в площадке работает таким образом, что данные с устройства можно получить только после настройки устройств на определенный IP адрес отправки.

IP адрес – это уникальный адрес устройства в сети. IP означает «Интернет-протокол» - набор правил, который регулирует формат данных, отправляемый через интернет или локальную вычислительную сеть.

Соответственно, каждое устройство в площадке может быть идентифицировано по его IP адресу, а также после анализа данных, приходящих с него. Это означает, что для конкретной работы API должен иметь данные с каких IP адресов собирается информация и на какие адреса отправляется информация.

Данный проект должен быть напрямую связан со повышенной скоростью обработки данных, так как роботы отсылают данные по протоколу UDP для наиболее быстрой обработки будет использоваться объектно-ориентированное программирование на языке С++.

Объектно-ориентированное программирование – это парадигма разработки, набор правил и критериев, предполагающая собой, что все решения в программе будут реализованы с помощью объектов. Объектно-ориентированное программирование состоит из нескольких обязательных критериев реализации:

* абстракция данных – код программы должен содержать объекты с минимальным количеством полей, максимально исключая дублирование.
* инкапсуляция – свойство, позволяющее объединить данные и методы, работающие в одном классе. Одно из свойств инкапсуляции – сокрытие данных.
* наследование – свойство, позволяющее описать существующий класс на основе уже существующего.
* полиморфизм – свойство, позволяющее автоматически конвертировать один тип объекта в другой, более обобщенный.

С++ является компилируемым, что означает собираемым в машинный код при сборке программы, а не при выполнении, статически типизированным, подразумевая наличие строгих правил при использовании, языком программирования общего назначения, что позволяет ему быть безопасным и универсальным для разработки любого программного обеспечения.

Чтобы качественно выполнить поставленную задачу каждый из методов должен быть прикреплен к устройству, созданному виртуально– специальным объектом, названным классом. Такой подход позволяет определить максимальную интуитивно-понятную простоту общения с программой. Краткое объяснение работы класса указано на рисунке 3.



Рисунок 3 – Пример работы класса

Класс в программировании является шаблоном для создания однотипных объектов, в котором могут содержаться методы, свойства объекта, которые позволяют быстро получить доступ к определенной информации. Каждый объект класса является уникальным, поэтому такой подход обеспечит возможность удалять из сети объектов определенные и обеспечит гибкость работы API. Фактическое удаление объекта класса закрывает соединение с роботом или любым другим устройством и позволяет очистить память.

Такой подход позволяет:

* устранить большой расход памяти программой;
* увеличить расширяемость API интерфейса для других программистов;
* позволяет виртуальные объекты и контролировать их проще, чем под физическим управлением.

1. Практическая часть
   1. Описание сценариев использования

Данное для разработки API содержит методы, ускоряющие доступ другим программистам к информации, предоставляемой роботами. Некоторые из них имеют требования в виде аргументов, которые должен предоставить программист и определенный результат выполнения метода.

В качестве описание сценариев использования выступают примером документации API.

* + 1. Методы API, используемые для обработки и управления данными с робота-политайзера

Робот-политайзер требует наличия следующих методов:

* метод получения информации о данных с моторов массивом (в API get\_temperature\_array) содержится в классе robot\_politaizer и возвращает массив, который используется имеет числовые ключи и целые числовые значения, данный метод имеет возвращаемый тип int (целочисленный) и обозначен, как указатель на объект;
* метод получения информации о данных с моторов по номеру мотора (в API get\_temperature\_by\_number) содержится в классе robot\_politaizer и возвращает температуру определенного мотора, данный метод имеет возвращаемый тип int;
* метод установки положения робота (в API set\_robot\_pos) содержится в классе robot\_politaizer и возвращает true, если операция установки положения робота проведена успешно или false, если операция вызвала ошибку. Имеет возвращаемый тип bool. Требует три целочисленных числа в качестве координат, первое число – координата X, вторая – Y, третья – Z;
* метод установки положения робота изменяя только одну координату (в API set\_robot\_cord) содержится в классе robot\_politazer и возвращает true, если операция установки положения робота проведена успешно или false, если операция вызвала ошибку. Имеет возвращаемый тип bool. Требует целочисленный номер координаты 1 для X, 2 для Y, 3 для Z, а также значение координаты в целочисленном формате;
* метод получения информации о текущих координатах робота (в API get\_robot\_cord) содержится в классе robot\_politaizer и возвращает массив, который используется имеет числовые ключи и целые числовые значения, данный метод имеет возвращаемый тип int (целочисленный) и обозначен, как указатель на объект;
* метод получение информации о состоянии захвата робота (в API get\_robot\_vp) содержится в классе robot\_politaizer и возвращает true в случае, если вакуумная присоска робота включена, и false, если вакуумная присоска выключена. Имеет возвращаемый тип int (целочисленный).
  + 1. Методы API, используемые для обработки и управления данными с робота-манипулятора

Робот-манипулятор должен иметь следующие методы:

* метод получения информации о данных с моторов массивом (в API get\_temperature\_array) содержится в классе robot\_man и возвращает массив, который используется имеет числовые ключи и целые числовые значения, данный метод имеет возвращаемый тип int (целочисленный) и обозначен, как указатель на объект;
* метод получения информации о данных с моторов по номеру мотора (в API get\_temperature\_by\_number) содержится в классе robot\_man и возвращает температуру определенного мотора, данный метод имеет возвращаемый тип int;
* метод установки положения робота (в API set\_robot\_pos) содержится в классе robot\_man и возвращает true, если операция установки положения робота проведена успешно или false, если операция вызвала ошибку. Имеет возвращаемый тип bool. Требует три целочисленных числа в качестве координат, первое число – координата X, вторая – Y, третья – Z;
* метод установки положения робота изменяя только одну координату (в API set\_robot\_cord) содержится в классе robot\_man и возвращает true, если операция установки положения робота проведена успешно или false, если операция вызвала ошибку. Имеет возвращаемый тип bool. Требует целочисленный номер координаты 1 для X, 2 для Y, 3 для Z, а также значение координаты в целочисленном формате;
* метод получения информации о текущих координатах робота (в API get\_robot\_cord) содержится в классе robot\_man и возвращает массив, который используется имеет числовые ключи и целые числовые значения, данный метод имеет возвращаемый тип int (целочисленный) и обозначен, как указатель на объект;
* метод получение информации о состоянии захвата робота (в API get\_robot\_vp) содержится в классе robot\_man и возвращает true в случае, если захват включен, и false, если захват выключен. Имеет возвращаемый тип bool.
  + 1. Методы API, используемые для обработки и управления данными с сигнальной четырёхцветной лампы

Сигнальная лампа должна иметь следующие методы:

* метод установки состояния цветов лампы отдельно (в API set\_light\_enable\_first) содержится в классе signal\_lamp и возвращает true в случае, если операция установки прошла успешно, и false, если операция установки цвета прошла с ошибкой. Требует целочисленный номер лампы и булевое значение для установки состояния, которым является true или false. Имеет возвращаемый тип bool, реализация метода указана на рисунке 4;



Рисунок 4 – Реализация метода установки цветов лампы отдельно

* метод установки состояния цветов ламп массивом (в API set\_light\_enable) содержится в классе signal\_lamp и возвращает true в случае, если операция установки прошла успешно, и false, если операция установки цвета прошла с ошибкой. Требует четыре булевых значения для установки состояния, которым является true или false. Имеет возвращаемый тип bool, реализация метода указана на рисунке 5;

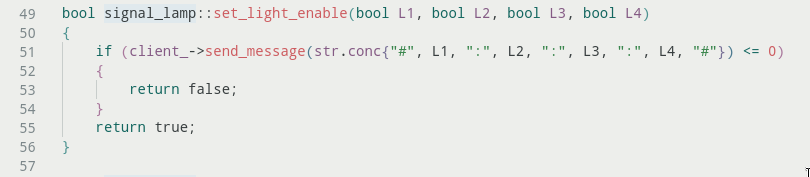


Рисунок 5 – реализация метода установки состояния цветов ламп массивом

* метод получения текущего состояния цвета лампы (в API get\_light) содержится в классе signal\_lamp и возвращает булевое значение, true, если лампа включена, false, если лампа выключена, требует ввода целочисленного номера лампы, имеет возвращает тип bool, реализация метода указана на рисунке 6;



Рисунок 6 – Реализация метода получения текущего состояния цвета лампы

* метод получение текущего состояния цветов лампы массивом (в API get\_light\_array) содержится в классе signal\_lamp и возвращает массив булевых значений, true, если лампа включена, false, если лампа выключена, имеет целочисленные ключи массива по номеру лампы, имеет возвращает тип bool, реализация метода указана на рисунке 7.

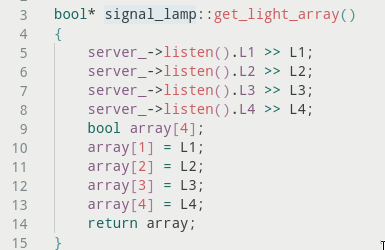


Рисунок 7 – Реализация метода получения текущего состояния цветов лампы массивом

* + 1. Методы API, используемые для обработки и управления данными с удаленного терминала
* получение количества нажатий на малые кнопки (в API get\_buttons\_clicked) содержится в классе distance\_terminal и возвращает массив целочисленных значений, имеет возвращаемый тип int\*, реализация метода указана на рисунке 8;

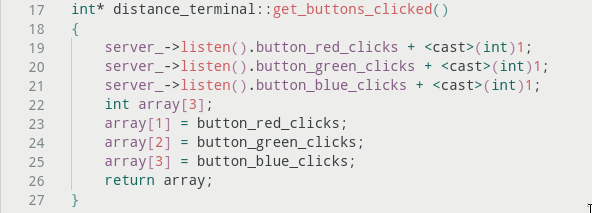


Рисунок 8 – Реализация метода для получения количества нажатий на малые кнопки

* получение состояния рычага (в API get\_red\_button\_toggle) содержится в классе distance\_terminal и возвращает булевое значение, если рычаг зажат, то true, иначе false. Имеет возвращаемый тип bool, реализация метода указана на рисунке 9;

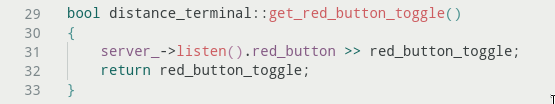


Рисунок 9 – Реализация метода получения состояния рычага

* метод установки состояния цветов лампы отдельно (в API set\_light\_enable\_first) содержится в классе distance\_terminal и возвращает true в случае, если операция установки прошла успешно, и false, если операция установки цвета прошла с ошибкой. Требует целочисленный номер лампы и булевое значение для установки состояния, которым является true или false. Имеет возвращаемый тип bool, реализация метода указана на рисунке 10;

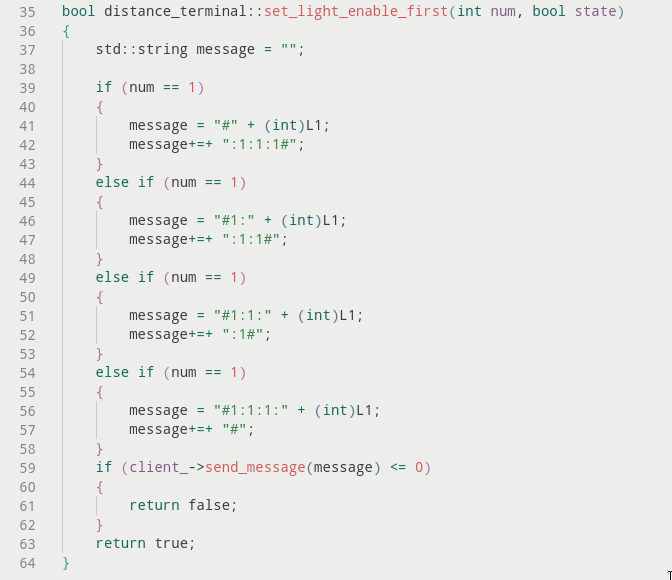


Рисунок 10 – Реализация метода получения состояния рычага

* метод получение текущего состояния цветов лампы массивом (в API get\_light\_array) содержится в классе distance\_terminal и возвращает массив булевых значений, true, если лампа включена, false, если лампа выключена, имеет целочисленные ключи массива по номеру лампы, имеет возвращает тип bool. Реализация метода указана на рисунке 11.

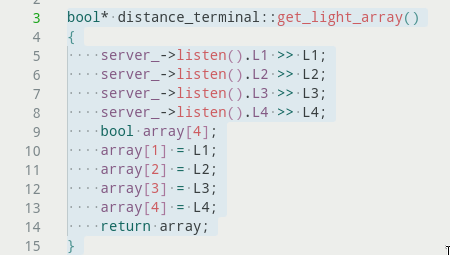


Рисунок 11 – Реализация метода получения текущего состояния цветов лампы массивом

* 1. Проектирование модели данных

Для создания интерфейса понадобится спроектировать модель данных базовых устройств при этом обеспечив дальнейшую возможность развития программного модуля.

Чтобы обеспечить развитие программного модуля нужно обеспечить будущий программный модуль свойством расширяемости, поэтому базовый модуль должен содержать в себе специальный класс, который будет назван общим названием-устройства (devices при использовании данного класса в программном коде).

Данный класс содержит в себе базовые для общения с устройством поля:

* IP адрес устройства (string ip\_address\_ при использовании его в программном коде, string означает, что IP адрес будет хранится в программе, как строка, поэтому при установке программистом этого параметра он так же должен будет использовать строку при конфигурировании своих устройств в программе, это наиболее простой способ), нужен для определения устройства в сети и отправке данных на него;
* порт устройства (int port\_ при использовании его в программном коде, это означает, что порт устройства будет использовать целочисленный тип значения), нужен для ускорения определения устройства в сети и избавления от базовой коллизии между устройствами;
* клиент связи UDP (UDPClient client\_ при использовании его в программном коде), занимается отправкой сообщений на любое из устройств, для этого нужно будет передавать в него IP адрес и порт устройства;
* сервер связи UDP (UDPServer server\_ при использовании его в программном коде), занимается приемом и обработкой внешних UDP запросов.

В связи с тем, что модель имеет древовидную структуру, базовый класс, на котором базируется API должен быть реализован таким образом, чтобы при создании новых устройств им выдавался уникальный номер, это нужно для того, чтобы избежать коллизии между двумя устройствами, обычно такой номер имеет целочисленный тип (в программе данный столбец будет указан, как id\_).

Итоговое решение шаблона устройств указано на рисунке 12 и 13 соответственно.



Рисунок 12 - Итоговое решение шаблона устройств



Рисунок 13 – Итоговое решение шаблона устройств в cpp

* 1. Проектирование интерфейса

API работает таким образом, что постоянно идёт сбор данных с площадки, после чего у каждого уникального устройства меняются данные, сохраненные в API. Каждый раз, когда программист использует методы, предполагающие собой изменение данный внешних устройств–он предоставляет аргументы, требуемые методом, а метод API обрабатывает данные и отсылает по нужному IP адресу уже обработанные данные.

Для того, чтобы программист мог расширить функционал библиотеки API нужно проектировать библиотеку таким образом, чтобы каждый класс имел два файла, один файл расширения hpp с названием устройства – этот файл будет использоваться для реализации заголовочных файлов, эти файлы можно подключить и использовать уже в стороннем проекте. Это позволяет использовать только определенные части библиотеки, не используя её полностью, что повышает скорость работы и потребление памяти программой. Второй тип файлов имеет формат cpp и используется для реализации методов исходной отдельно от класса, что делает программу более читаемой.

Итоговая файловая структура указана на рисунке 14.

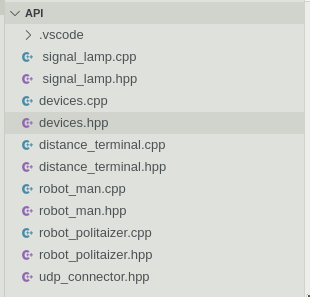


Рисунок 14 – Файловая структура после проектирования интерфейса

* 1. Разработка программы

Чтобы приступить к разработке программы нужно определится с используемым стандартом С++, этот язык программирования имеет несколько популярный стандартов, которые несовместимы между собой. Из всех предоставленных стандартов наиболее подходящим является 14 стандарт, приносящий достаточное количество базовых методов для реализации, поставленной ранее задачи.

Для создания программы на этом языке также потребуется компилятор программы. Наиболее оптимальный для этой задачи компилятор – это g++, он является кроссплатформенным (может быть использован на любой из операционных систем и платформ) и является наиболее поддерживаемым сообществом программистов.

Для реализации этого API необходимо спроектировать модель данных, а также структуру самой программы и перенести их в код. Затем нужно реализовать базовые методы.

Результатом создание программы является полноценная, легко расширяемая API библиотека, которая позволяет управлять роботами и обрабатывать информацию, приходящую с роботов и других устройств, связанных в одной площадке общей локальной вычислительной сетью.

* 1. Тестирование программы

API тестируется непосредственно подключаясь к площадке с роботами определённых моделей. Для тестирования должна быть совершенна внутренняя настройка роботов на отправку пакетов данных серверу (компьютеру, который указан как сервер) по его IP адресу, затем подключается библиотека с нужными устройствами и создаются объекты с проверкой правильности работы методов.

Заключение

Подводя итоги, можно заключить, что поставленные ранее цели и задачи были выполнены.

Необходимо было разработать API интерфейс для обработки данных роботами. Данное API было реализовано при использовании такого языка программирования, как C++.

Для достижения этой цели было необходимо совершить анализ предметной области и разработать некоторое количество методов для достижения рабочего и протестированного программного модуля.

В течении курсовой работы существовала гипотеза исследования о том, что разработка программного модуля API для обработки данных роботов, ускорит разработку программного обеспечения, разработанного на нем и по окончании анализа предметной области и завершения создания программного модуля данная гипотеза подтвердилась.

Список использованной литературы

1. Кениг Эндрю и Му Барбара Э. Эффективное программирование на C++. Практическое программирование на примерах. Серия "C++ In-Depth" / Эндрю Кёниг, Барбара Э. Му - М.: Вильямс, 2019. - 368 с.

2. Лафоре Роберт. Объектно-ориентированное программирование в С++ / Роберт Лафоре - СПб.: Питер, 2018. - 928 с.

3. О'Двайр Артур. Осваиваем C++17 STL. / Артур О'Двайр - ДМК Пресс, 2018 – 352 c.

4. Прата Стивен. Язык программирования C++. Лекции и упражнения / Стивен Прата - М.: Диалектика-Вильямс, 2018 - 1244 с.

5. Стенли Б. Липпман, Жози Лажойе и Барбара Э. Му. Язык программирования C++. Базовый курс / Стенли Б. Липпман, Жози Лажойе и Барбара Э. Му. - 5-е изд., - М.: Диалектика-Вильямс, 2019 - 1120 с.

6. Страструп Бьярне. Программирование. Принципы и практика с использованием C++ / Бьярне Страустру - М.: Вильямс, 2018 - 1328 с.

7. Яцек Галовиц. С++17 STL. Стандартная библиотека шаблонов / Галовиц Яцек СПб.: Питер, 2018. - 436 с.