Министерство науки и высшего образования Российской Федерации



Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«Санкт-Петербургский горный университет императрицы Екатерины II»

Кафедра автоматизации технологических процессов и производств

**КУРСОВАЯ РАБОТА**

по дисциплине

« Программирование и алгоритмизация »

(наименование учебной дисциплины согласно учебному плану)

**Тема:** Программирование ПЛК применением языков программирования стандарта МЭК. Установление связи между базой данных и вывод графического материала в виде приложения при помощи языка Python.

**Автор:** студент гр. АПГ-22 \_\_\_\_\_\_\_\_ /Скрябнев А.В./

(шифр группы) (подпись) (Ф.И.О.)

**Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Комиссия:** доцент /Федорова Э.Р./

Санкт-Петербург

2024

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Санкт-Петербургский горный университет имени Екатерины II

УТВЕРЖДАЮ

Заведующий кафедрой

Кульчицкий А.А.

(подпись)

« » 2024 г.

Кафедра автоматизации технологических процессов и производств

**КУРСОВАЯ РАБОТА**

По дисциплине Программирование и алгоритмизация

(наименование учебной дисциплины согласно учебному плану)

**ЗАДАНИЕ**

студенту группы АПГ-22 Скрябнев А.В.

(шифр группы) (Ф.И.О.)

Тема работы: Программирование ПЛК применением языков программирования стандарта МЭК. Установление связи между базой данных и вывод графического материала в виде приложения при помощи языка Python.

Содержание пояснительной записки: Титульный лист, лист задания, реферат, оглавление, введение, необходимое количество разделов, заключение, список использованных источников, приложения.

1. Перечень графического материала: IDE PyCharm, UnityPro XL, графическое отображение теории.

Срок сдачи законченной работы «5» апреля 2024 г.

Задание выдал (руководитель работы) \_\_\_\_\_\_\_\_\_\_ Федорова Э.Р.

(подпись) (Ф.И.О)

Задание принял к исполнению студент \_\_\_\_\_\_\_\_\_\_ Скрябнев А.В.

(подпись) (Ф.И.О)

Дата получения задания «7» марта 2024 г.АННОТАЦИЯ

В данной курсовой работе рассматривается программирование ПЛК с применением языков программирования стандарта МЭК, а именно ST и LD. Для этого применяется программа Unity Pro XL, позволяющая симулировать подключение к ПЛК и занесение в него соответствующих программ. Также в данной курсовой работе рассматривается взаимодействие языка программирования Python с базой данных MySQL и создание приложения для представления обсчитанных графиков при помощи фрэймворка QT.

ANNOTATION

In this course work, improvement of PLC programming using IEC programming languages, namely ST and LD standards. For this purpose, the Unity Pro XL program is used, which allows you to simulate connecting to the PLC and entering the corresponding programs into it. Also in this work is the interface of interaction of the Python programming language with the MySQL database and the creation of an application for displaying calculated graphs using the QT framework.

Оглавление

[Аннотация 3](#_Toc165403239)

[Ход работы 5](#_Toc165403240)

[1 Работа с Unity Pro XL 5](#_Toc165403241)

[1.1 Исходные данные 5](#_Toc165403242)

[1.2 Алгоритм решения 5](#_Toc165403243)

[1.3 Общие сведения о ST 7](#_Toc165403244)

[1.4 Решение ST через инструкцию IF 9](#_Toc165403245)

[1.5 Решение через ST инструкцию CASE 12](#_Toc165403246)

[1.6 Решение через LD 14](#_Toc165403247)

[2 Связь базы данных с python и выведение данных в виде приложения при помощи соответствующих библиотек 15](#_Toc165403248)

[2.1 Основная теория о базах данных 15](#_Toc165403249)

[2.2 Основные команды языка SQL 17](#_Toc165403250)

[2.3 Использование коннекторов (библиотек) для взаимодействия с ними через языки программирования (Python). 18](#_Toc165403251)

[2.4 Создание приложения 20](#_Toc165403252)

[2.4.1 Основы QT 20](#_Toc165403253)

[2.4.2 Layout 22](#_Toc165403254)

[2.4.3 Slider 23](#_Toc165403255)

[2.4.4 График 25](#_Toc165403256)

[2.4.5 Таблица 25](#_Toc165403257)

[2.4.6 Button 26](#_Toc165403258)

[Вывод 29](#_Toc165403259)

ВВЕДЕНИЕ

Программируемые логические контроллеры (ПЛК) являются важным элементом автоматизации промышленных процессов. Они обеспечивают управление и мониторинг различными устройствами, а также выполнение операций в зависимости от определенных условий. Для программирования ПЛК часто используются языки программирования стандарта Международной электротехнической комиссии (МЭК).

Одним из таких языков программирования является язык Python, который также широко применяется для работы с базами данных и разработки графических приложений. Установление связи между базой данных и вывод графического материала в виде приложения при помощи языка Python позволяет создавать эффективные и удобные интерфейсы для управления и мониторинга промышленными процессами.

Этот подход к программированию ПЛК не только облегчает разработку и обслуживание систем управления, но и позволяет значительно улучшить их функциональность и надежность. Использование языка Python в сочетании с программированием ПЛК по стандартам МЭК открывает перед специалистами новые возможности для создания инновационных и эффективных решений в области автоматизации промышленных процессов.

ПРАКТИЧЕСКАЯ ЧАСТЬ

1. Работа с Unity Pro XL
   1. Исходные данные

На рисунке 1 изображено расположение датчиков уровня в установке шахтного водоотлива.

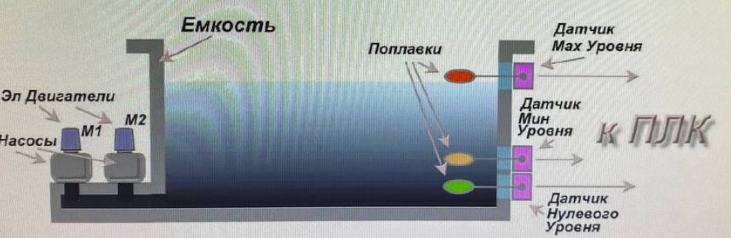


Рисунок 1 - расположение датчиков уровня в установке шахтного водоотлива.

Необходимо разработать программу, которая обеспечивала бы выполнение следующих условий:

1. Если вода находиться ниже нулевого уровня насосы стоят;
2. Как только уровень достигает минимального уровня, включается первый насос;
3. Первый насос выключается, если уровень снижается ниже нулевого уровня;
4. Если уровень воды достигается максимального уровня, включается второй насос;
5. Оба насоса отключаются, если уровень спускается ниже нулевого уровня.
   1. Алгоритм решения

Алгоритм представлен в виде графа (Рисунок 2).

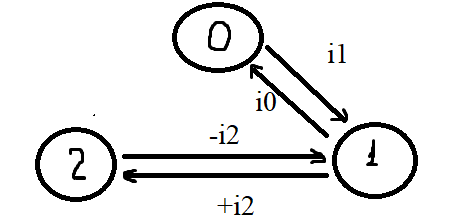


Рисунок 2 – Граф алгоритма

Нулевое состояние: «Насос1» и «Насос2» выключены.

Переходное действие i1:

* «Датчик нулевого уровня» равен 1;
* «Датчик минимального уровня» равен 1;

Возвратное действие i0:

* «Датчик нулевого уровня» равен 0;
* «Датчик минимального уровня» равен 0;

Первое состояние: «Насос1» включен.

Переходное действие +i2:

* «Датчик нулевого уровня» равен 1;
* «Датчик минимального уровня» равен 1;
* «Датчик максимального уровня» равен 1;

Возвратное действие -i2:

* «Датчик нулевого уровня» равен 1;
* «Датчик минимального уровня» равен 1;
* «Датчик максимального уровня» равен 0;

Второе состояние: «Насос1» и «Насос2» включены.

Представим данный алгоритм в виде таблицу (таблица 1). Фактически, каждый блок алгоритма будет состоять из четырех  
частей: номер текущего состояния, условие перехода, номер следующего  
состояния и список действий, осуществляемых в текущем состоянии. Небольшая сложность состоит в том, что LD состоит из набора двоичных символов, которые могут принимать только значения «0» и «1». Поэтому мы закодируем десятичный номер состояния в двоичном коде, который запишем с помощью нужного числа внутренних битов памяти %Mx. В данном случае, для системы с 3 состояниями понадобится 2 бита, так как .

Таблица 1 – Представление алгоритма в табличном виде

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Текущее состояние | Текущее значение битов | | | Событие перехода | След. Значение битов | | | Nasos2 | Nasos1 |
| Dmax | Dmin | Dnull | Dmax | Dmin | Dnull |
| 0 | 0 | 0 | 0 |  | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |  | 1 | 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 1 |  | 0 | 0 | 0 | 1 | 1 |

* 1. Общие сведения о ST

Язык ST или SCL (Structured Text, Структурированный Текст)представляет собой язык высокого уровня, имеющий черты языков «Pascal» и «Basic». Данный язык имеет те же недостатки, что и IL, однако они выражены в меньшей степени.

Программа ST (Структурированный текст) состоит из списка инструкций, которые последовательно выполняются контроллером. Используя эти инструкции, Вы можете вызывать функциональные блоки, функции и процедуры условно и безусловно, выполнять присвоения, явно выполнять инструкции, повторять инструкции и выполнять прыжки условно или безусловно в пределах секции (Рисунок 3).

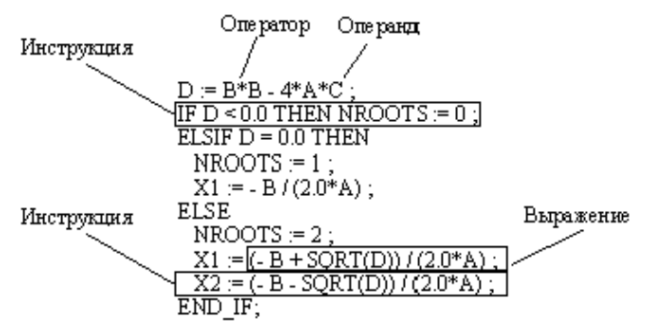


Рисунок 3 - Структурированный текст

Инструкции состоят из:

o оператора,

o операнда,

o выражения, если требуется и

o комментария, если требуется.

o В дополнение, каждая инструкция может быть снабжена меткой

(чтобы по команде Go to делать переход по строкам, например).

•Инструкции должны заканчиваться точкой с запятой.

•Несколько инструкций (разделенных точкой с запятой) могут располагаться в одной строке.

•Одиночная точка с запятой обозначает пустые инструкции (Одиночная  
точка с запятой ; изображают пустую инструкцию. Например,  
IF x THEN ; ELSE ..

В этом примере пустая инструкция следует за THEN инструкцией. Это  
означает, что программа покидает инструкцию IF как только условие IF  
= 1).

•Допускаются разрывы строк в инструкциях (многострочные  
инструкции).

•Метки, символы и комментарии могут быть помещены в любом месте  
секции. (Комментарии могут вводиться в любом месте, где разрешены  
пробелы)

•Проверка синтаксиса и семантики производится сразу же после ввода  
инструкции. Результат этой проверки отображается цветным текстом.

•Синтаксически и семантически неверные секции могут быть сохранены.

* 1. Решение ST через инструкцию IF

Описание инструкции IF:

IF [Условие] THEN

[Действие];

END\_IF;

Если [Условие] будет истинно, то [Действие] будет выполнено.

Для работы с Unity Pro XL необходимо объявление переменных. Для этого нужно нажать в окне «Structural view» на папку «Variable & FB instances» и выбрать «Elementary Variables» (Рисунок 4).

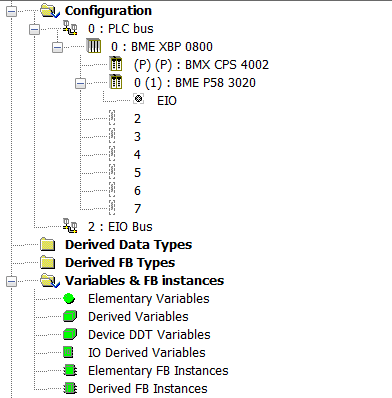


Рисунок 4 – Дерево проекта

На рисунке 5 изображена таблица с необходимыми нам переменными, с определением их типам и комментариями. Тип «EBOOL» используется для чтения дискретных входов, то есть, то есть если локализованной переменной присвоить тип «BOOL» при компиляции возникнет ошибка.

Отличие типа «BOOL» от «EBOOL» заключается в том, что «BOOL» состоит из одного значащего бита, который содержит значение «FALSE» или «TRUE», а переменная типа «EBOOL» состоит из трех битов: бита значения (value bit), бита прошлого значения (history bit) и бита форсирования (forcing bit). Бит форсирования отображает статус форсирования: «0» если значение переменной не форсируется, «1» если форсируется.

Форсирование – принудительное изменение значения переменной программным путем в онлайн-режиме, производимое с целью проверки и отладки программы.

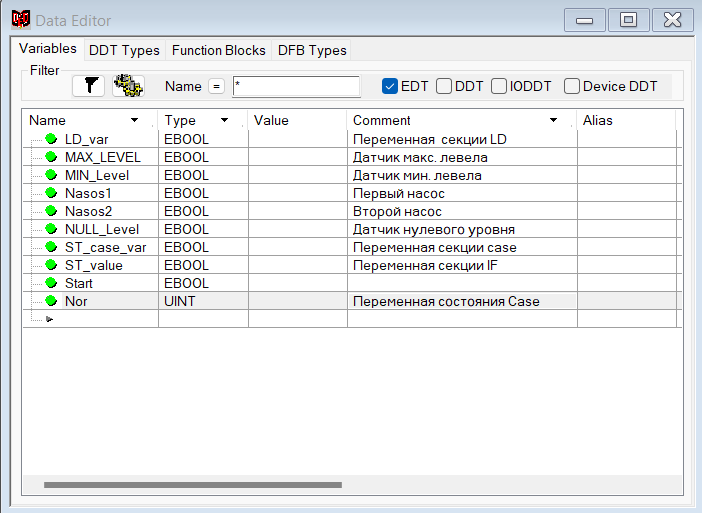


Рисунок 5 – Таблица переменных

Переменные секций нам необходимы для того, чтобы мы выбирали какие секции должны работать в данный момент и какие должны быть выключены.

Реализация алгоритма через инструкцию IF изображена на рисунке 6.

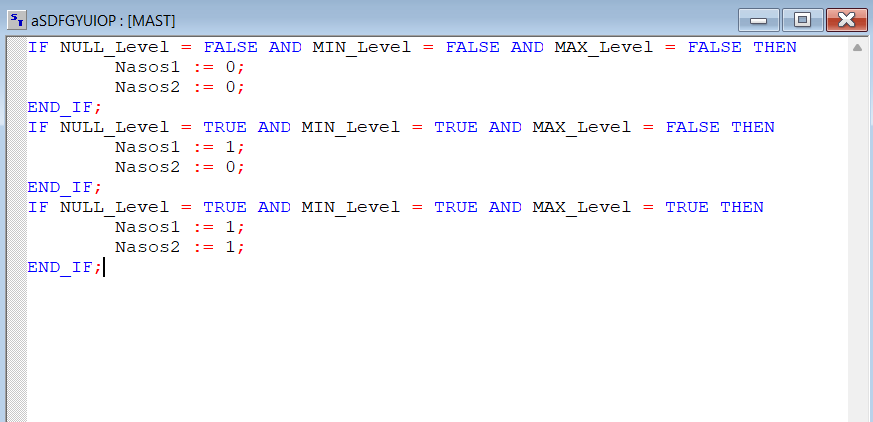


Рисунок 6 - Реализация алгоритма через инструкцию IF

Для проверки работоспособности программы сделаем симуляцию ее запуска на каком-нибудь виртуальном ПЛК. Для этого нам необходимо проверить проект на ошибки: В главном меню приложения во вкладке «Build» нажать «Analyze project». В случае отсутствия ошибок в окне оповещения об этом сообщиться (Рисунок 7).

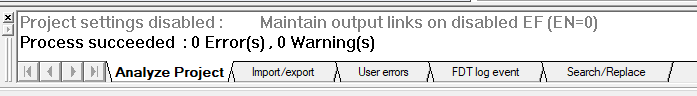


Рисунок 7 – Результат анализа проекта

Дальше необходимо перекинуть программу на ПЛК:

1. «PLC\Connect»;
2. «PLC\Transfer project to PLC».

Высветиться окно, изображенное на рисунке 8.

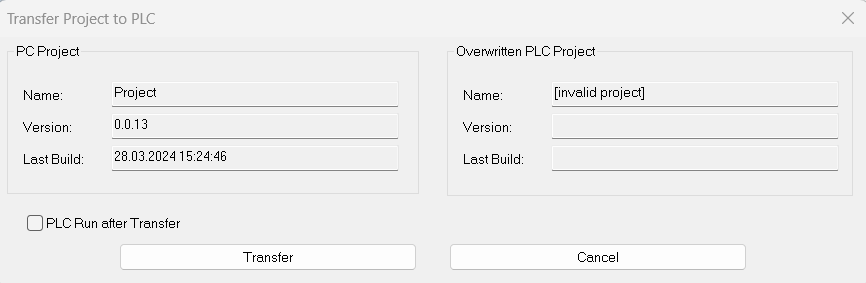


Рисунок 8 – Настройки Transfer project to PLC

Для взаимодействия с проектом необходима анимационная таблица. Для этого создаем в «Structural view\Animation Tables» новую таблицу и переносим туда созданные раннее переменные (Рисунок 9).

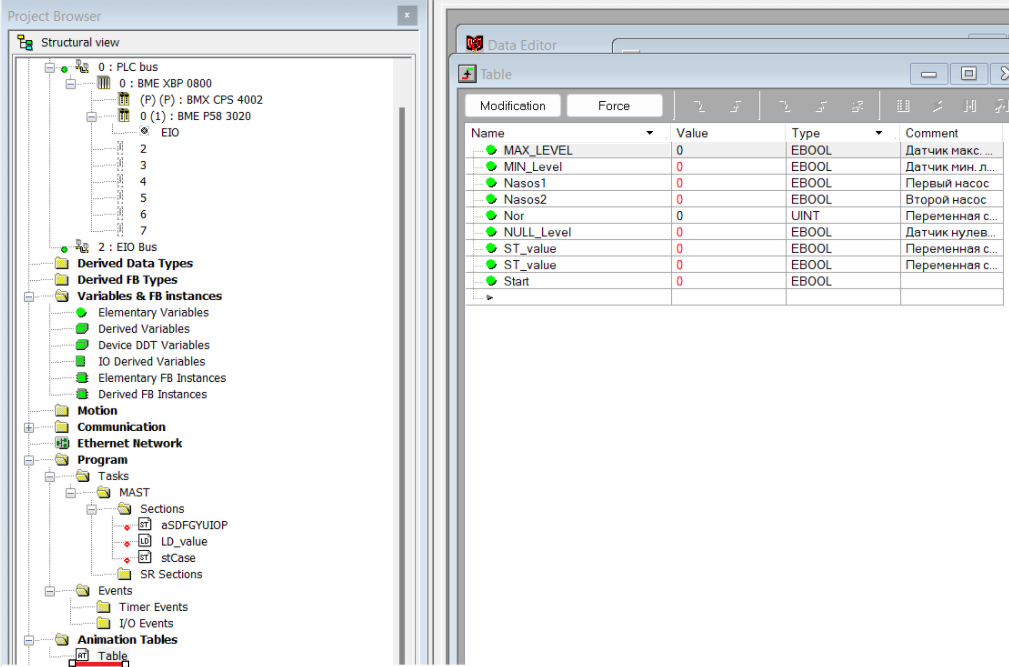


Рисунок 9 – Создание анимационной таблицы

Проверим работоспособность алгоритма, включив необходимую секцию.

Состояние 0 изображено на рисунке 10:

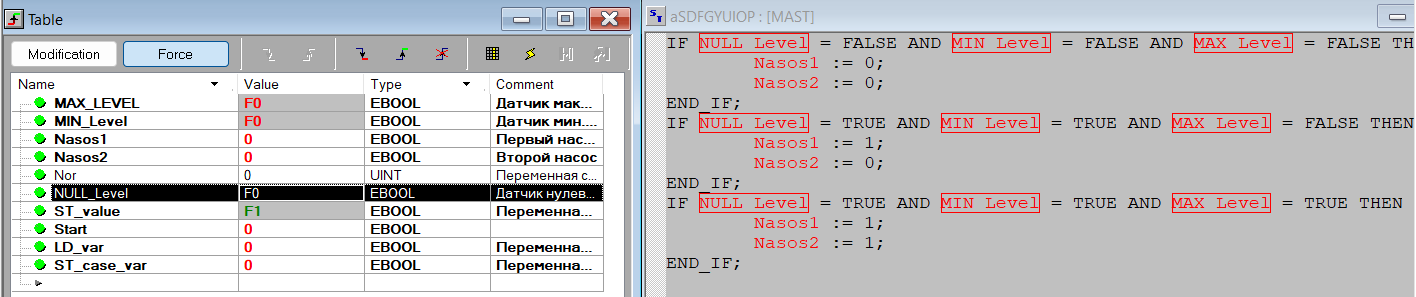


Рисунок 10 – Состояние объекта 0

Состояние 1 изображено на рисунке 11:

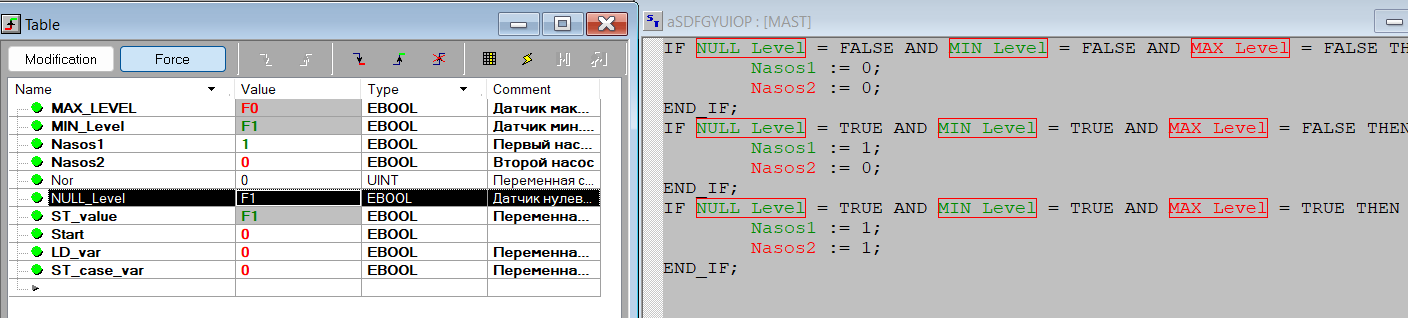


Рисунок 11 – Состояние объекта 1

Состояние 2 изображено на рисунке 12:

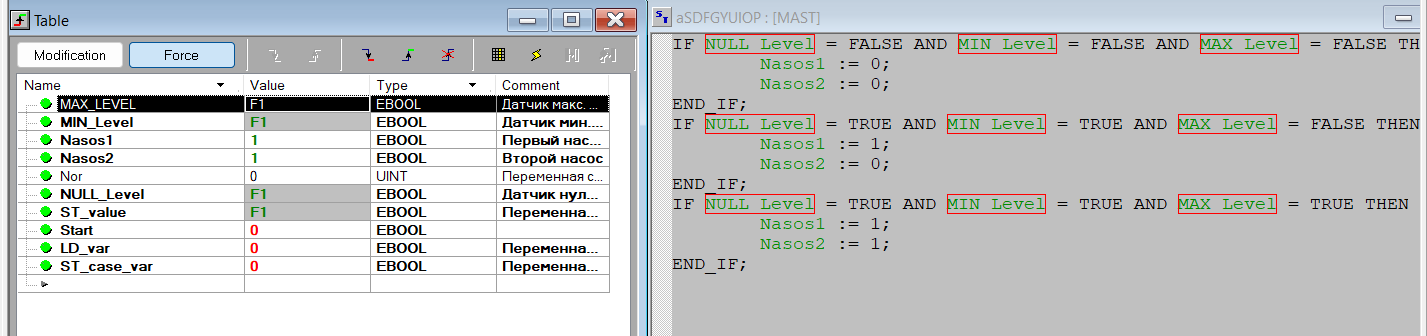


Рисунок 12 – Состояние объекта 2

Программа удовлетворяет заданным условиям.

* 1. Решение через ST инструкцию CASE

Описание инструкции CASE:

CASE [Переменная состояния] OF

0:

[Действие];

[Условие перехода на следующее состояние];

1: …

END\_CASE;

Пример инструкции изображен на рисунке 13.

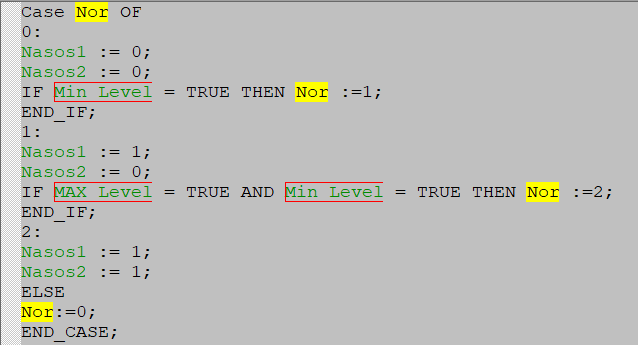


Рисунок 13 – Реализация инструкции CASE

Проверим работоспособность алгоритма, включив необходимую секцию.

Состояние 0 изображено на рисунке 14:

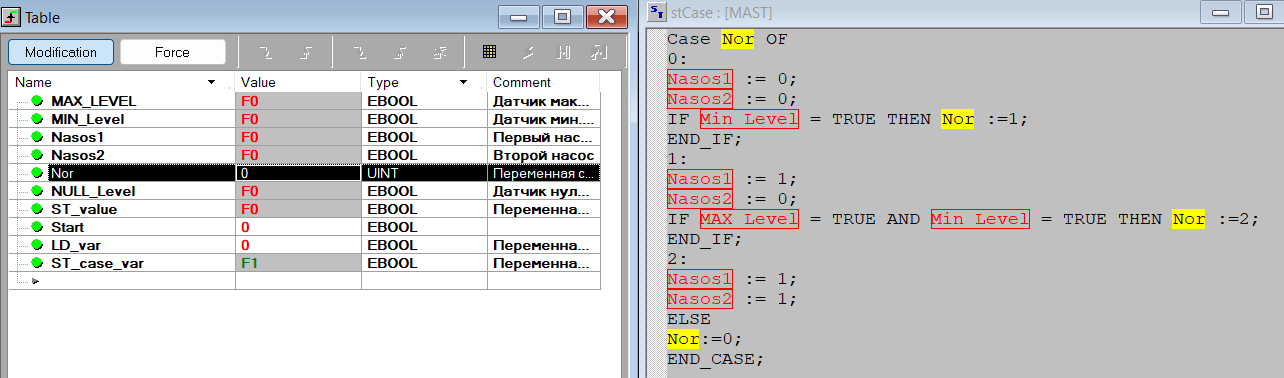


Рисунок 14 – Состояние объекта 0

Состояние 1 изображено на рисунке 15:

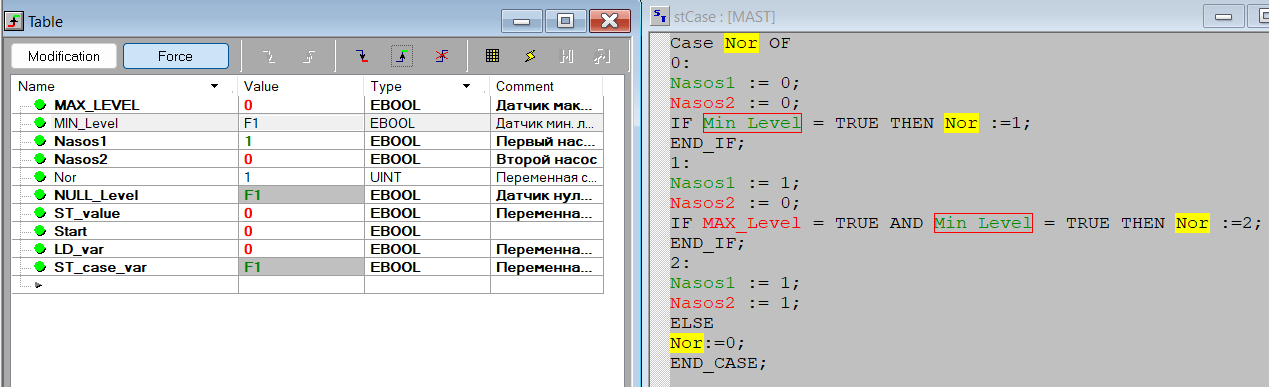


Рисунок 15 – Состояние объекта 1

Состояние 2 изображено на рисунке 16:

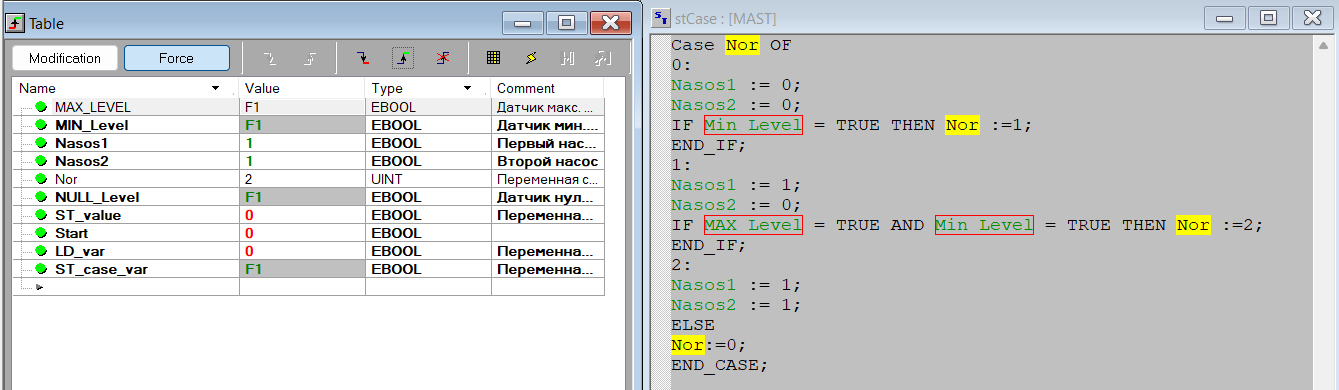


Рисунок 16 – Состояние объекта 2

Программа удовлетворяет заданным условиям.

* 1. Решение через LD

Язык LD (Ladder Diagram) – это графический язык программирования, представляющий собой диаграмму релейно-контактной логики (Рисунок 17) [6].

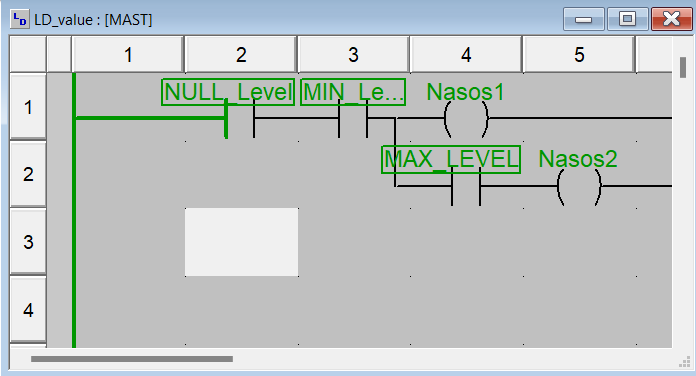


Рисунок 17 – Алгоритм на языке LD

Самая левая зеленая линия — это провод с током. Мы ограничиваем поступление тока, до исполнения какого – либо из условий. При подаче тока на исполнительные механизмы они начинают работать.

Проверим работоспособность алгоритма, включив необходимую секцию.

Состояние 0 изображено на рисунке 18:

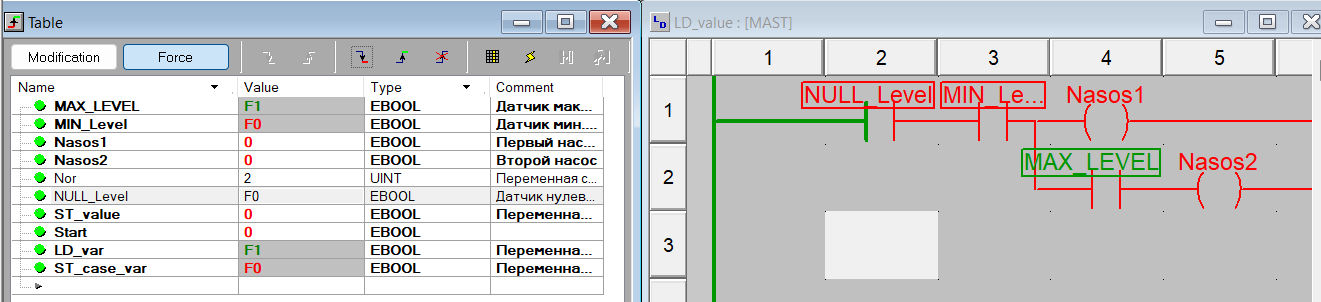


Рисунок 18 – Состояние объекта 0

Состояние 1 изображено на рисунке 19:

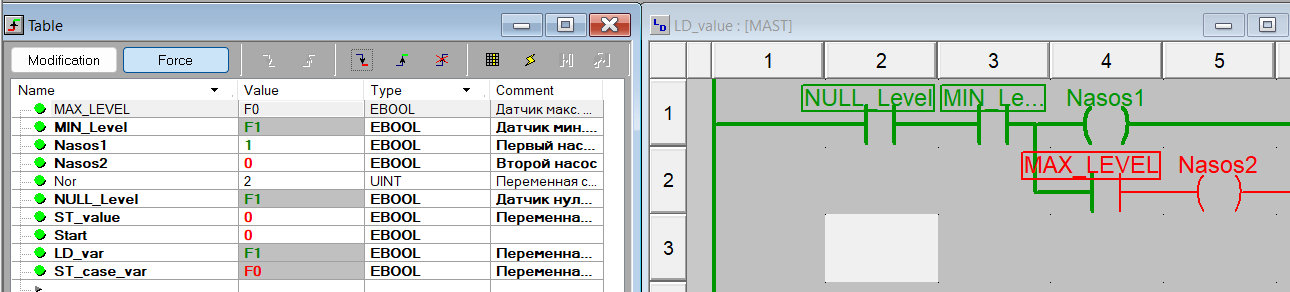


Рисунок 19 – Состояние объекта 1

Состояние 2 изображено на рисунке 20:

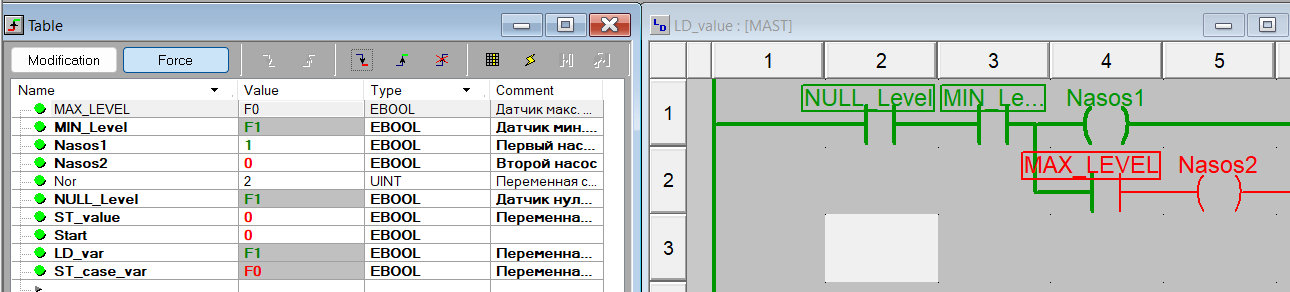


Рисунок 20 – Состояние объекта 2

Программа удовлетворяет заданным условиям.

1. Связь базы данных с python и выведение данных в виде приложения при помощи соответствующих библиотек
   1. Основная теория о базах данных

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется система управления базой данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных [1].

Данные в наиболее распространенных типах современных баз данных обычно хранятся в виде строк и столбцов, формирующих таблицу. Этими данными можно легко управлять, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов (SQL).

SQL — это язык программирования, используемый в большинстве реляционных баз данных для запросов, обработки и определения данных, а также контроля доступа [1].

Для хранения любой информации необходимо материальное записывающее устройство. Зачастую для этого используются выделенные сервера или же облака, в нашем же случае используется локальное устройство (жесткий диск на компьютере). Эта информация, нам будет необходима для пункта 2.2.

В данной курсовой работе используется реляционная база данных MySQL [2]. Взаимодействие с ней может осуществляться двумя способами:

1. Командная строка (cmd)
2. Workbench (GUI)

Для взаимодействия через командную строку необходимо:

1. Сменить директория, на ту, где хранится сервер MySQL (Рисунок 21)



Рисунок 21 - Сменить директория, на ту, где хранится сервер MySQL

1. Использовать команду mysql с параметрами: «-u», «-p», где первый отвечает за имя пользователя «User», второй соответственно пароль для входа на сервер «Password» (Рисунок 22). Регистрация в систему MySQL происходило при установке.

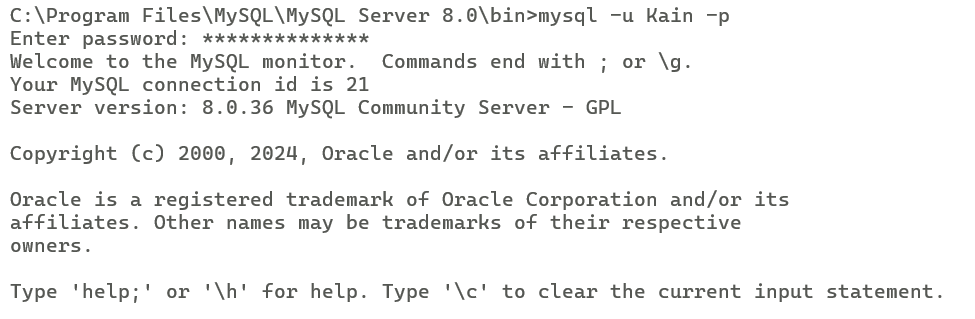


Рисунок 22 – Пример входа в систему MySQL

* 1. Основные команды языка SQL

1. Создание новой базы данных: «CREATE DATABASE [name];», где вместо [name] вписываем имя новой базы данных. Для проверки что команда выполнилась можно написать: «SHOW DATABASES;». Для дальнейшего использования созданной базы данных, используется команда: «USE [name]».
2. Реляционная база данных (РБД) – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. Для создания таблицы используется команда (Рисунок 23):

«CREATE TABLE [name] (

[nameColumn1] [typeOfValue] [flags]»

[nameColumn2] [typeOfValue] [flags]»

…);

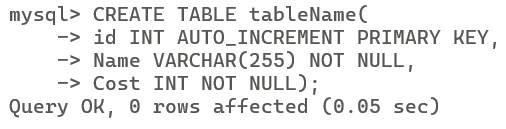


Рисунок 23 – Пример использования команды

typeOfValue – тип переменной. Пример: INT, VARCHAR(255), …

flags:

* NOT NULL – Автоматическое заполнение значение not null;
* PRIMARY KEY - это поле, которое однозначно идентифицирует каждую запись в таблице.
* AUTO\_INCREMENT - это атрибут, который автоматически генерирует уникальные целочисленные значения для столбца при вставке новой строки в таблицу.

1. Внесение значений в таблицу используется команда INSERT (Рисунок 24).

«INSER [nameTable] ([nameValue1, nameValue2…]) VALUES («Value1», «Value2»…), …»



Рисунок 24 – Пример использования команды.

1. Получение значений с таблицы осуществляется командой SELECT:

SELECT [nameColumn1], [nameColumn2], … FROM [nameTable];

Вместо перечисления названия столбцов можно напечатать символ «\*», для получения всех столбцов (Рисунок 25).

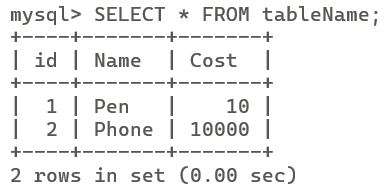


Рисунок 25 – Пример использования команды

* 1. Использование коннекторов (библиотек) для взаимодействия с ними через языки программирования (Python).

Данные об дифференциальном уравнение мы храним в табличном виде для удобного переноса информации в базу данных. Для это используется библиотека «pandas».

Для хранения данных в табличном виде используется метод «DataFrame». На рисунке 26 показано хранение коэффициентов уравнения и массив значений «x», «y».

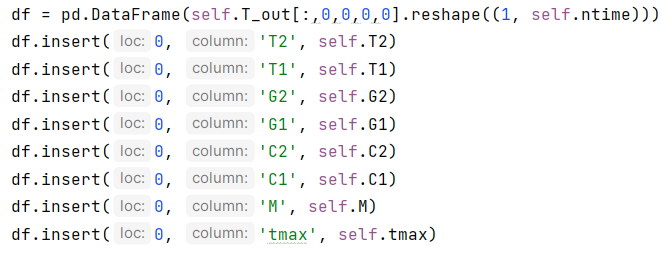


Рисунок 26 – пример применения метода «DataFrame» библиотеки «Pandas»

Для переноса таблицы в базу данный используется библиотека «SQLAlchemy». Данная библиотека работает по структуре, изображенной на рисунке 27.

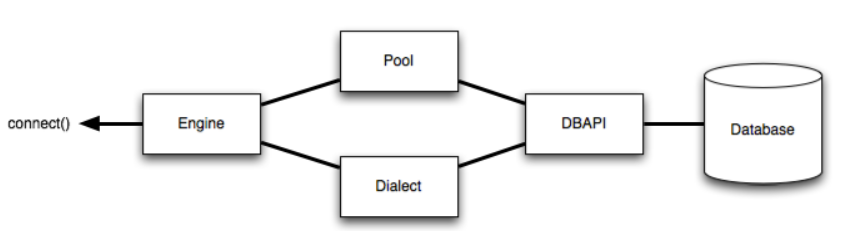


Рисунок 27 – Структура взаимодействия базы данных с языком Python

Engine является отправной точкой для любого приложения «SQLAlchemy» [3]. Представляет из себя URL, необходимое для хранения и дальнейшей передачи параметров подключения и взаимодействия с базой данной в DBAPI. В URL указывается название базы данных и название API (DataBaseAPI), к которой мы будем подключаться (Рисунок 28).



Рисунок 28 – Пример URL

К параметрам, которые необходимы для подключения относятся:

* Пользователь (Регистрируется при скачивании, вместе с паролем)
* Пароль
* Host - Адрес или же порт сервера, на котором хранится Database. Если программа, при помощи которой происходит подключение к базе данных и сама база данных хранится на одном и том же сервере (компьютере), то для этого существует стандарт. Он разнится от использования API. В нашем случае: «localhost:3306».
* Название базы данных.

Для определения необходимого API используются модули «Pool» и «Dialect». Они интерпретируют функции модуля DBAPI, а также поведение базы данных [3].

«Engine» является одним из параметров метода «to\_sql» библиотеки «pandas». Этот метод необходим для переноса целой таблицы в базу данных (Рисунок 29).

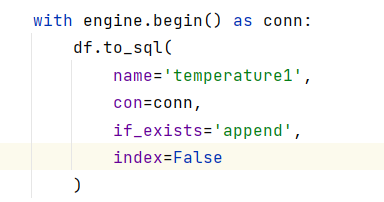


Рисунок 29 – Использование метода «to\_sql»

Он создает таблицу в базе данных аналогичную переменной «df» под название «temperature1». Параметр «if\_exists» позволяет не создавать при повторном обращении уже существующую таблицу, а добавление в нее новых значений.

Метод «with … as» автоматически разрывает соединение с базой данных, для предотвращения не нужных ошибок.

2.4 Создание приложения

2.4.1 Основы QT

Qt – фреймворк для создания кроссплатформенных приложений, в основном используется для языка программирования С++, но также может использоваться и в Python. Использование этого фреймворка происходит через импорт библиотеки.

Для разработки приложений фреймворк Qt предоставляет функциональный модуль «QtWidgets» (в обиходе просто называют "виджеты"). Исторически это более старый подход, который предполагает использование виджетов - специальных классов, который определяют некоторые визуальные компоненты, например, кнопки, текстовые поля и т.д. И с помощью виджетов пользователь может взаимодействовать с приложением. Приложения на основе виджетов будут выглядеть как нативные приложения, что упрощает работу пользователя с приложением [4].

Любое приложение в «Windows» является процессом (процессами) со своими потоками. Оно обладает свойствами:

* HANDLE
* PID
* Name
* Parent Name

…

Для создания подобного процесса используется команда «QApplication». Чтобы нас процесс (Приложение) что-то показывало необходимо ему присвоить окно. В «Qt» уже есть решение для окна — виджет «QMainWindow», имеющий стандартные функции окна для использования в приложениях, который содержит панели инструментов, меню, строку состояния, закрепляемые виджеты и многое другое [5]. Создание начального пустого окна представлен на Рисунке 30.

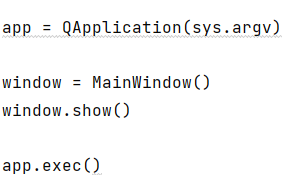


Рисунок 30 – Создание пустого окна приложения

Для отображения окна используется команда «[nameWindow].show()». Команда «[nameApplication].exec()» запускает цикл событий, он необходим для реагирования приложения на действия пользователя.

Для описания функций окна и содержащихся внутри его виджетов необходимо расписать его класс (Рисунок 31).

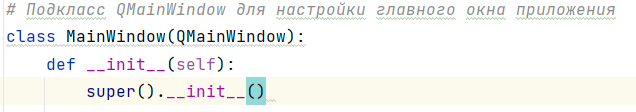


Рисунок 31 – Объявление класса QMainWindow

Для того чтобы класс окна мог использовать импортируемые модули фреймворка необходима строка «super().\_\_init\_\_()». Для установки названия окна и цвета заднего фона используются команды, представленные на рисунке 32.

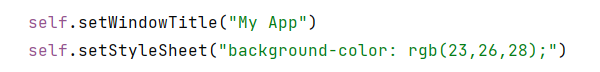


Рисунок 32 – Настройка основных параметром окна

2.4.2 Layout

В окно можно добавлять любые виджеты. Для их позиционирования существует несколько способов, например, использования структуры REST, однако в данной работы используется «layout». Они позволяют структурно размешать наши виджеты, например: вертикально на определенном расстоянии друг от друга. Для использования несколько «layout» необходимо создать один главный и уже в него добавлять в него подмакеты (Рисунок 33).



Рисунок 33 – Настройка layout

Чтобы придать макетам цвет, создается пустой виджет, который мы окрашиваем командой:

[Widget Name].setStyleSheet (f”background-color: rgb(…,…,…);”)

И помешаем необходимый макет (layout) в созданный виджет. Это работает потому что в фреймворке QT «Widgets» является главным классом, а все остальные наследуются от него. Это подразумевает, что под виджетом может пониматься как кнопка, так и макет.

На рисунке 34 показано добавление подмакетов в главный макет и расстояния между макетами.

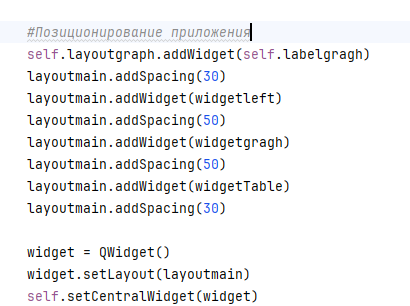


Рисунок 34 – Настройка позиционирования макетов

Чтобы созданные виджеты отображались необходимо их добавление в макеты.

2.4.3 Slider

Для задания дифференциальному уравнению коэффициентов используются «Sliders». Для удобства чтения кода и его сокращения напишем функцию шаблон. Функция создает горизонтальный слайдер. Определяет диапазон его слайдера по переданным функции значениям «minRange» и «maxRange». Определяется отметки деления слайдера при помощи команд «name.setSingleStep()» и «name.PageStep()». Первая определяет длину одного шага (расстояние между делениями), вторая количество делений. Команда «setTickPosition», отображает деления под ползунком. Дальше задается ширина и высота слайдера (Рисунок 35).

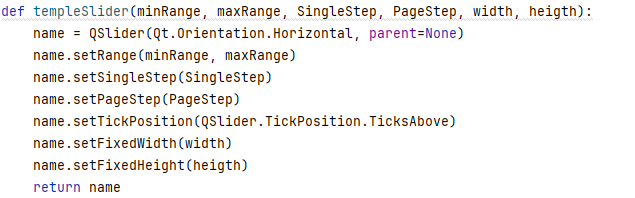


Рисунок 35 – Шаблон для создания Slider

Также слайдеру необходимо расписать реакцию на действия пользователя. Для этого создается специальная функция, к которой мы привязываем слайдер (рисунок 36). В ней уже расписываем реакцию приложения на изменения значения слайдера. Для это используется команда:

[Name of Slider].clicked.connect([Name Function])

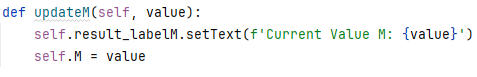


Рисунок 36 – Функция реакции приложения на нажатие кнопки

Для отображения текущего значения мы выводим надпись над слайдером. Для этого создаётся label он отображает строку: «Current Value M:», значение после двоеточия постоянно меняется, однако с самого начала мы выводим, то значение на котором слайдер указывает изначально командой: «[Slider name].value()». Для того чтобы это всё отображалось необходимо добавить в макет.

Сама функция реакции приложения на движение слайдера изображена на рисунке 37. Она нужна для изменения значения в label. А также изменения значения переменной, отвечающей за данный коэффициент.

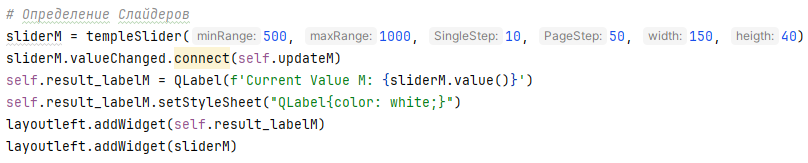


Рисунок 37 - Настройка взаимодействия со слайдером

По аналогии сделаны и другие слайдеры.

2.4.4 График

Для отображения зависимости дифференциального уравнения необходим график. Для этого используется команда «subplot» из библиотеки «matplotlib.pyplot». команда возвращает массив из двух элементов:

* Fig – нужен для сохранения графика;
* Ax – позволяет настраивать график, то есть задавать переменные оси ОХ и ОУ, заголовок графика и его легенду.

Создание и отображение в приложении пустого графика представлено на рисунке 38.

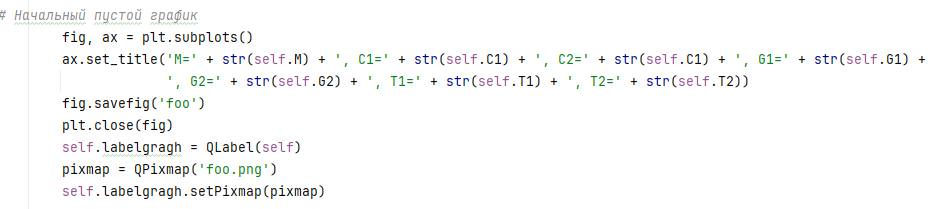


Рисунок 38 - Создание и отображение в приложении пустого графика

График изначально пустой, потому что при создании приложения ещё не обсчитано дифференциальное уравнение. Для этого нужно нажать на кнопку. Полная реакция приложения на нажатие кнопки пользователем рассмотрена позже, однако изменения графика рассмотрим сейчас. На рисунке 39 показано создание нового графика с подставлением обсчитанных значений, и пересохранение старого графика на новый, тем самым обновляя его отображение в приложении.

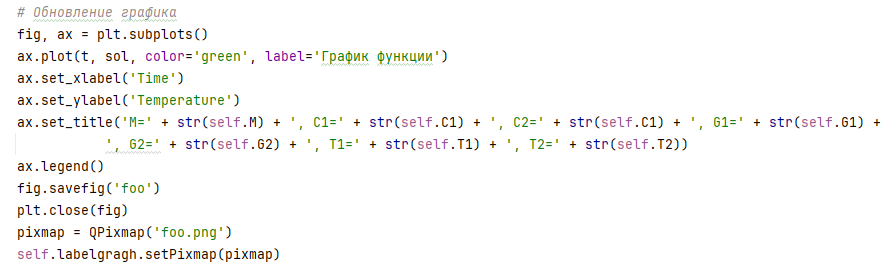


Рисунок 39 – Обновления графика

* + 1. Таблица

Таблица необходима для отображения значения Tout и соответствующему ему значение Time. Для создания виджета таблицы используется команда «QTableWidget()» (Рисунок 40). Дальше задается количество столбцов, с соответствующими заголовками и строк в таблице. И как было указано ранее изменяет цвет таблицы командой:

[Widget Name].setStyleSheet(f”background-color: rgb(…,…,…);”)

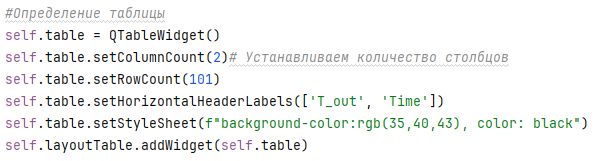


Рисунок 40 – Создание пустой таблицы

Внесение значений в таблицу происходит в функции реакции приложения на нажатие кнопки, потому что обсчет дифференциального уравнения происходит после её нажатия. Значения в таблицу присваиваются по ячейкам, поэтому используется цикл для заполнения всей таблицы (Рисунок 41).

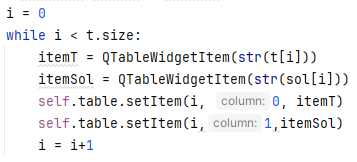


Рисунок 41 – Заполнение таблицы значениями

Команда «setItem» необходима для того удачного вноса значения в ячейку, в противном случае произойдет ошибка. Она задает вносимой информации необходимый тип.

2.4.6 Button

Для создания кнопок написана шаблонная функция для сокращения кода (Рисунок 42). Функция получает на вход 3 переменные:

* Text – текст, который будет отображаться на кнопке
* Width – ширина кнопки
* Height – высота кнопки

А возвращает макет настроенной кнопки.

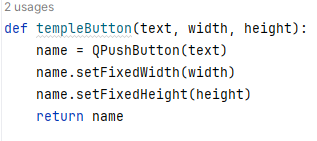


Рисунок 42 – Шаблон для создания кнопки

Также кнопке необходимо расписать реакцию на действия пользователя. Для этого создается специальная функция, к которой мы привязываем кнопку. В ней уже расписываем реакцию приложения на нажатие на кнопку. Для это используется команда:

[Name of Button].clicked.connect([Name Function])

Для того чтобы кнопка отображалась, необходимо поместить ее в layout (Рисунок 43).



Рисунок 43 – Настройка взаимодействия с кнопкой

Сама функция реакции приложения на нажатие кнопки изображена на рисунке 45. Она нужна для подставления ведённых коэффициентов в дифференциальное уравнение, его расчет, занесения решения в базу данных, и как сказано ранее обновления графиков и таблиц.

Решение дифференциального уравнения происходит при помощи модуля «odeint» из библиотеки «scipy». Он принимает макет дифференциального уравнения (Рисунок 44) и необходимые для ее решения переменные.

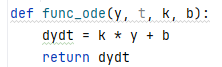
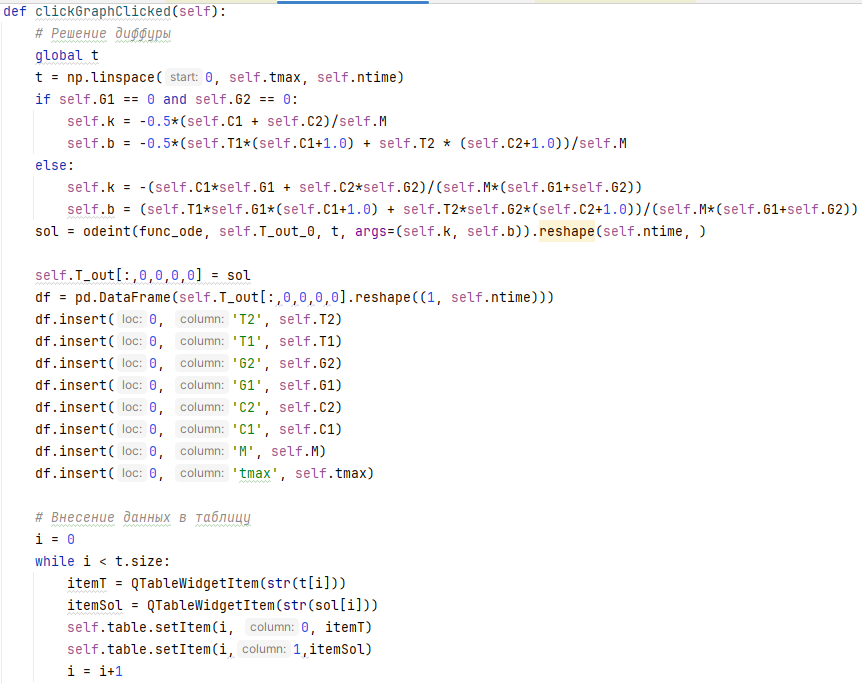


Рисунок 44 – Макет дифференциального уравнения



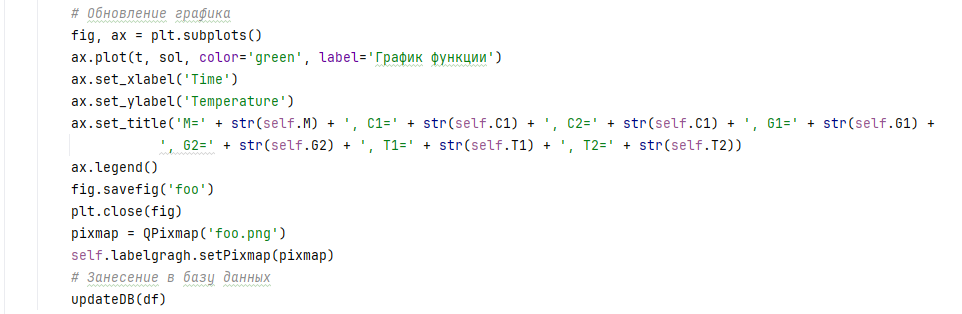


Рисунок 45 – Функция реакции приложения на нажатие кнопки

Дальше происходим представление коэффициентов и расчетных значений Tout и Time в табличный вид. Сама таблица переносится в базу данных с помощью функции, рассмотренной раннее.

Полный код, вид приложения и таблица базы данных представлены на рисунках А1 – А10.

ВЫВОД

Таким образом, программирование ПЛК с использованием языков программирования стандарта МЭК и установление связи между базой данных и вывод графического материала в виде приложения при помощи языка Python предоставляют широкие возможности для создания современных и эффективных систем управления промышленными процессами. Этот подход не только упрощает разработку и обслуживание систем, но и позволяет значительно улучшить их функциональность и надежность.

Использование языка Python в комбинации с языками программирования стандарта МЭК открывает новые перспективы для инновационных решений в области автоматизации промышленности. Благодаря этому подходу специалисты могут создавать современные интерфейсы управления, обеспечивать надежное и эффективное функционирование систем управления, а также повышать эффективность производственных процессов.

Таким образом, использование современных технологий программирования в автоматизированных системах позволяет значительно оптимизировать производственные процессы, повысить уровень безопасности и эффективности, а также обеспечить более комфортные условия работы для персонала.

Список литературы

1. URL: <https://www.oracle.com/cis/database/what-is-database/> (Дата обращения: 04.04.2024)
2. URL: <https://dev.mysql.com/downloads/installer/> (Дата обращения: 04.04.2024)
3. URL: <https://docs.sqlalchemy.org/en/20/core/engines.html#engine-configuration> (Дата обращения: 20.04.2024)
4. URL: <https://metanit.com/cpp/qt/1.1.php> (Дата обращения: 24.04.2024)
5. URL: <https://habr.com/ru/companies/skillfactory/articles/599599/> (Дата обращения: 24.04.2024)
6. Федорова Э.Р. Вводное занятие. Программирование ПЛК. Язык LD. Среда программирование UnityPro XL. // Презентация – 2023 – С. 1-39.

ПРИЛОЖЕНИЕ А

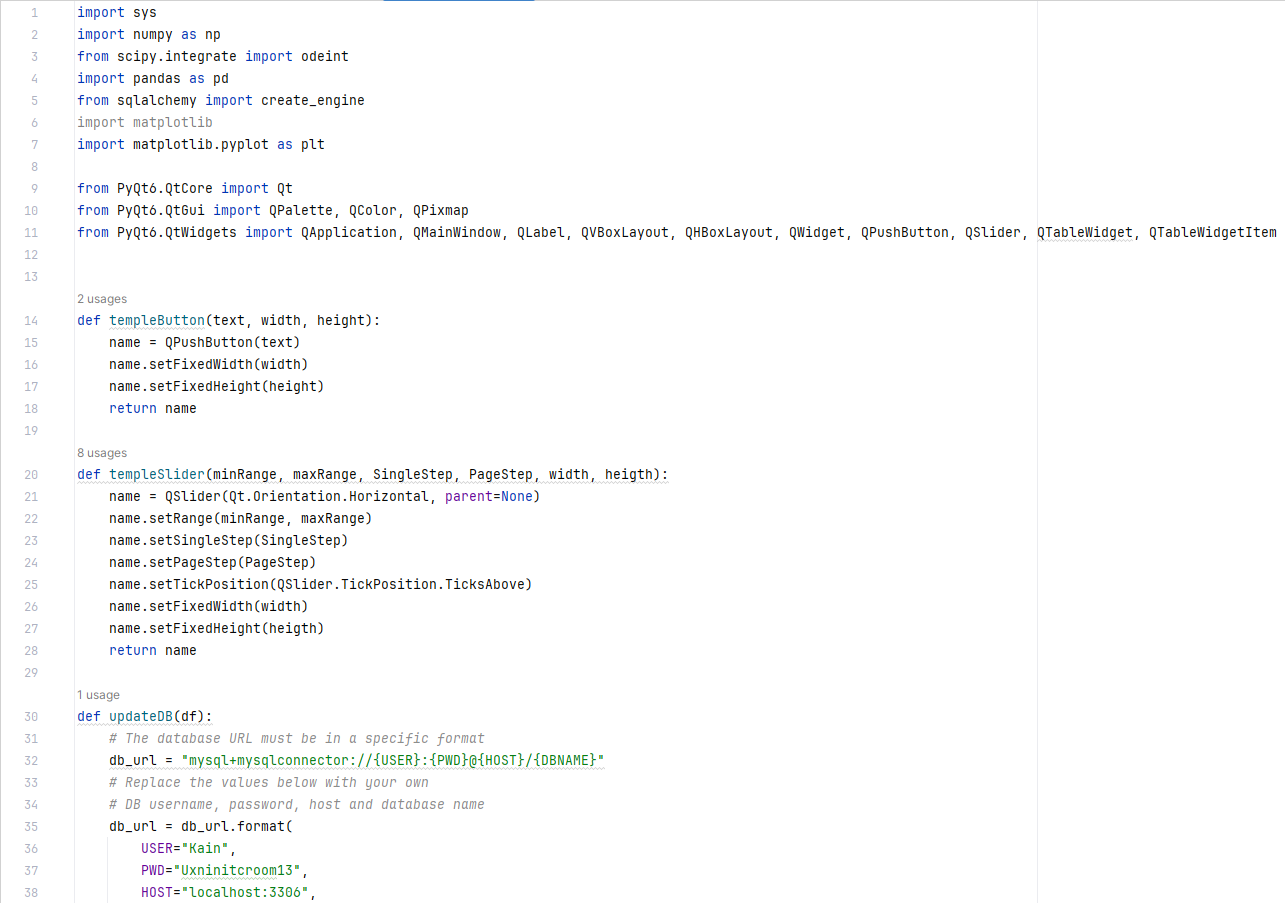


Рисунок А1 – Полный код первая часть



Рисунок А2 – Полный код вторая часть



Рисунок А3 – Полный код третья часть



Рисунок А4 – Полный код четвертая часть



Рисунок А5 – Полный код пятая часть

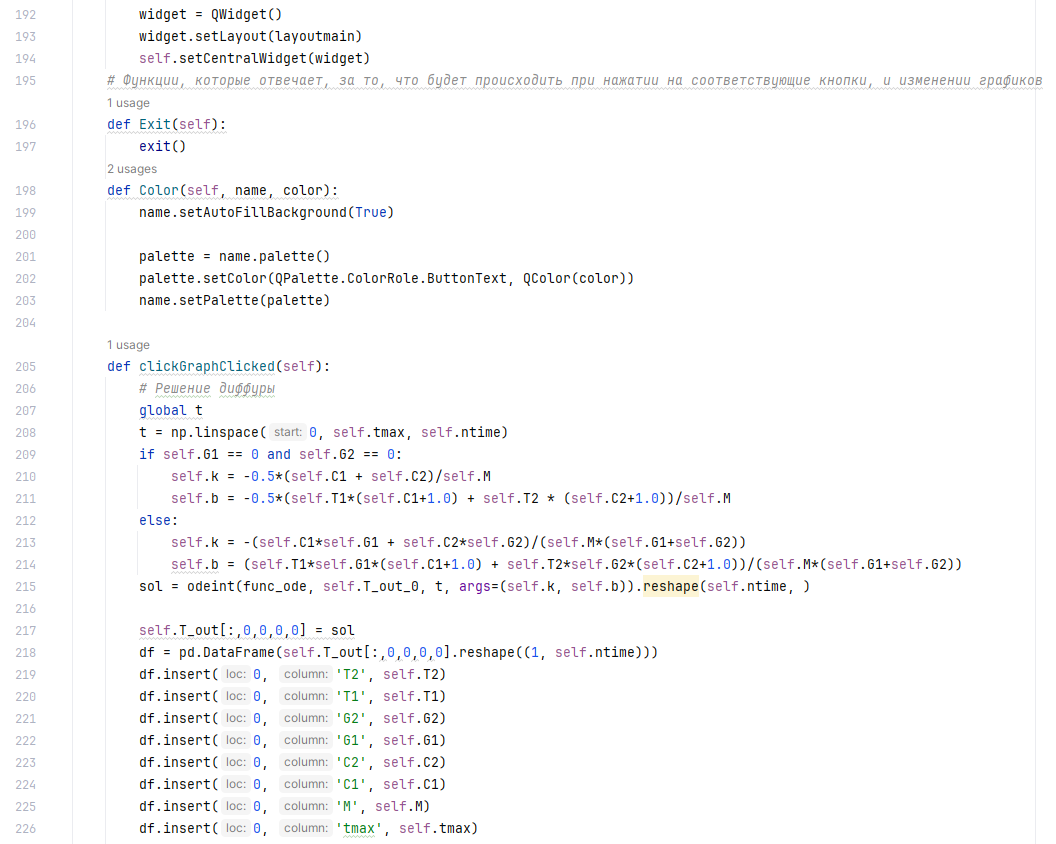


Рисунок А6 – Полный код шестая часть

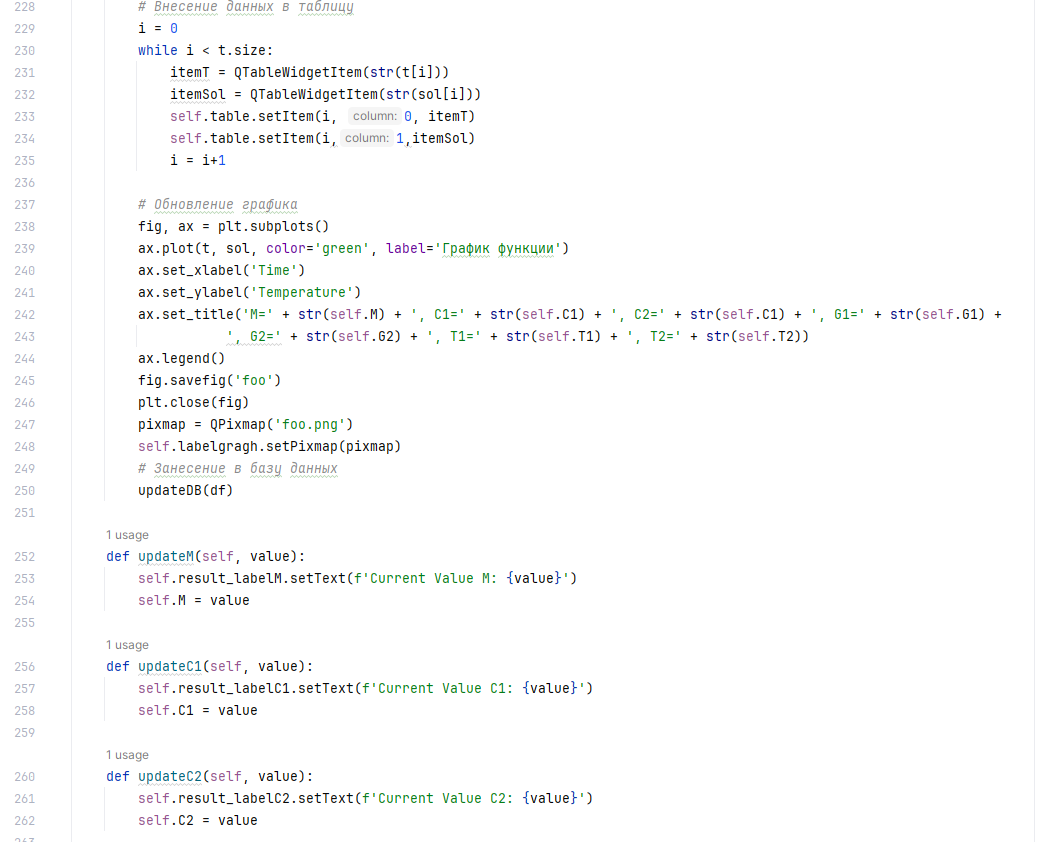


Рисунок А7 – Полный код седьмая часть



Рисунок А8 – Полный код восьмая часть

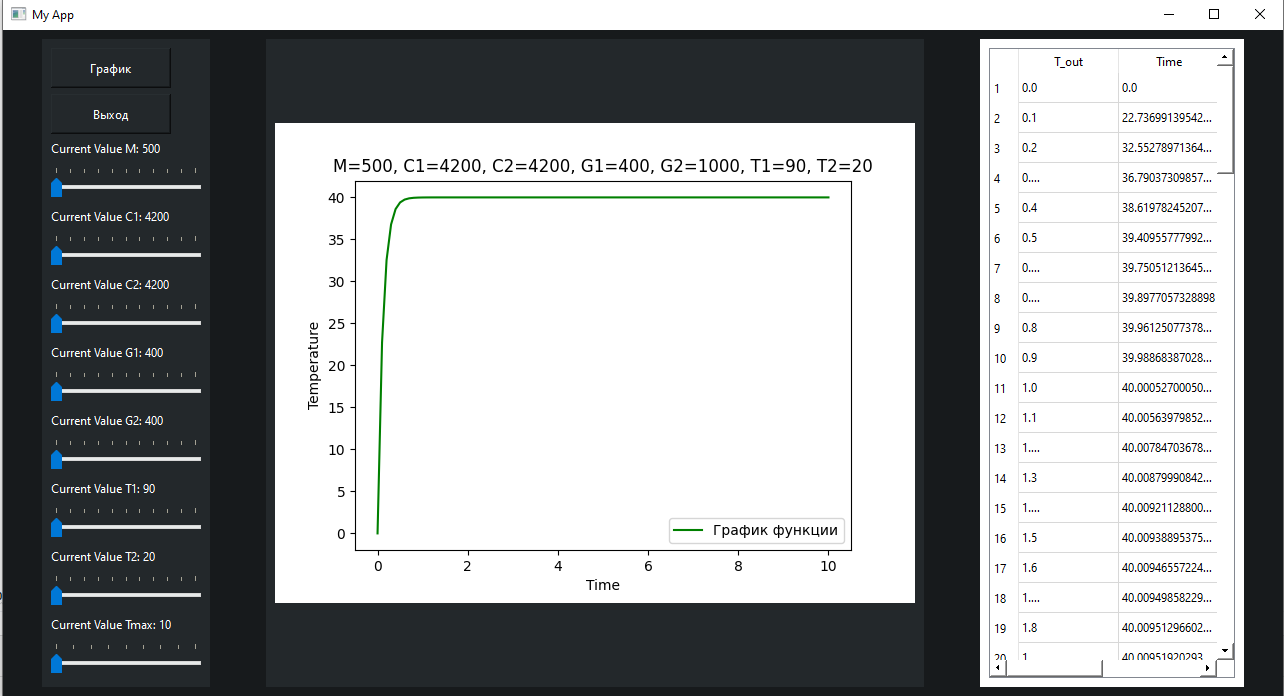


Рисунок А9 – Готовое приложение

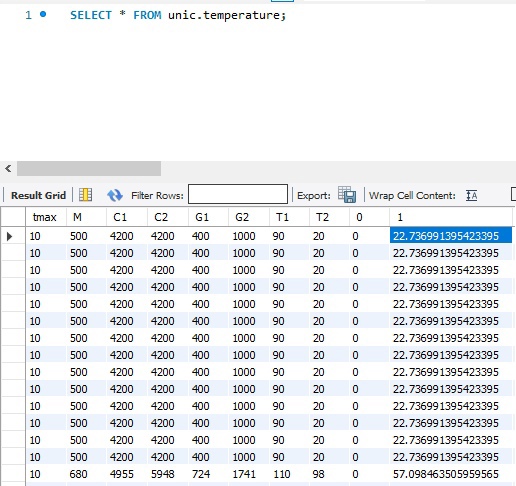


Рисунок А10 – Вывод таблицы из базы данных