Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Рыбинский государственный авиационный технический

университет имени П. А. Соловьева»

Институт Информационных технологий и систем управления

Кафедра «Вычислительных систем»

Курсовой проект

по дисциплине

«Периферийные устройства»

на тему:

«Модуль ввода-вывода информации, сопрягаемого с ЭВМ через шину USB»

Пояснительная записка

Студент группы ИВБ1-20\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Морозов А. А.  
Руководитель профессор, к.т.н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Вишняков В. А.

Рыбинск 2023

Техническое задание

Содержание

[Введение 4](#_Toc10831)

[1 Описание структурной схемы модуля 6](#_Toc9884)

[1.1 Канал ввода цифровой информации 7](#_Toc11764)

[1.2 Канал вывода аналоговой информации 8](#_Toc16351)

[2 Разработка функциональной схемы 9](#_Toc31800)

[2.1 Интерфейсные блоки модуля 12](#_Toc29453)

[2.2 Исполнительные блоки модуля 12](#_Toc17503)

[3 Разработка принципиальной схемы 1](#_Toc17062)4

[3.1 Обоснование элементной базы 1](#_Toc29453)4

[3.2 Разработка интерфейса 1](#_Toc29453)4

[3.3 Разработка преобразователя информации 1](#_Toc29453)4

[3.4 Разработка элементов модуля 1](#_Toc29453)4

[3.5 Расчёт токовой нагрузки и потребляемой мощности 1](#_Toc20307)6

[4 Разработка программного обеспечения 1](#_Toc30221)7

[4.1 Программная модель устройства 1](#_Toc29453)7

[4.2 Блок-схемы алгоритмов программы функционирования модуля 1](#_Toc29453)8

[Заключение 2](#_Toc599)7

[Список использованных источников 2](#_Toc5737)8

[Приложение А 2](#_Toc19593)9

Введение

*USB* (*Universal Serial Bus* — универсальная последовательная шина) является про­мышленным стандартом расширения архитектуры *PC*, ориентированным на ин­теграцию с телефонией и устройствами бытовой электроники. Версия стандарта 1.0 была опубликована в начале 1996 года, большинство устройств поддерживают стан­дарт 1.1, который вышел осенью 1998 года, — в нем были устранены обнаружен­ные проблемы первой редакции. Весной 2000 года опубликована спецификация *USB* 2.0, в которой предусмотрено 40-кратное увеличение пропускной способнос­ти шины. Первоначально (в версиях 1.0 и 1.1) шина обеспечивала две скорости передачи информации: полная скорость, *FS* (*full speed*) — 12 Мбит/с и низкая ско­рость, *LS* (*low speed*) — 1,5 Мбит/с. В версии 2.0 определена еще и высокая ско­рость, *HS* (*high speed*) — 480 Мбит/с, что позволяет существенно расширить круг устройств, подключаемых к шине. В одной и той же системе могут присутствовать и одновременно работать устройства со всеми тремя скоростями. Шина позволяет с использованием промежуточных хабов соединять устройства, удалённые от компьютера на расстояние до 25 м. Разработку устройств и их классификацию и стандартизацию координирует *USB*-*IF* (*USB Implemented Forum* *Inc*.).

Шина USB обеспечивает обмен данными между хост-компьютером и множеством периферийных устройств (ПУ). *USB* является единой централизованной аппарат­но-программной системой массового обслуживания множества устройств и мно­жества прикладных программных процессов. Связь программных процессов со всеми устройствами обеспечивает хост-контроллер с многоуровневой программ­ной поддержкой. Этим *USB* существенно отличается от традиционных перифе­рийных интерфейсов (портов *LPT*, *COM*, *GAME*, клавиатуры, мыши и т. п.).

Целью данной курсовой работы является разработка модуля ввода-вывода информации, сопрягаемого с ЭВМ через шину *USB*. На сегодняшний день *USB* является промышленным стандартом расширения архитектуры персонального компьютера. Для достижения поставленной цели необходимо разработать электрические структурную, функциональную и принципиальные схемы, конструкцию, программу, рассчитать устройство.

1 Описание структурной схемы модуля

По требованиям технического задания разрабатываемый модуль должен выполнять две функции:

- ввод цифровой информации (16 байт за 1 цикл обмена) со скоростью обмена 160000 байт/c, вид обмена *bulk*;

- вывод аналоговой информации (однополярное напряжение 0 – 3 В) с частотой сигнала 0 – 300 Гц, при максимальной ошибке преобразования 1 %.

Обобщённая структура модуля ввода-вывода информации показана на рисунке 1.1.

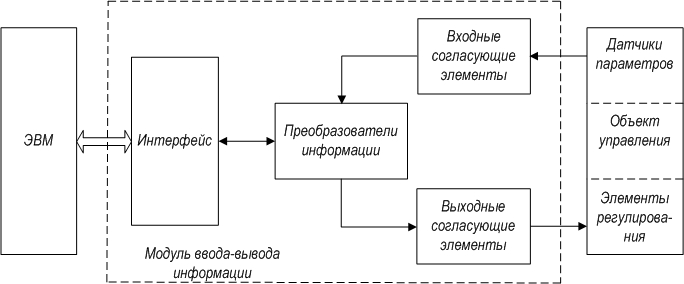


Рисунок 1.1 – Обобщённая структура модуля ввода-вывода

Анализируя техническое задание, можно сделать следующие предварительные выводы о структуре разрабатываемого модуля. Конструкция должна содержать следующие блоки:

- специальное *USB* устройство – оно обычно используется совместно с микроконтроллером и связывается с ним по высокоскоростному 8 разрядному универсальному параллельному интерфейсу. Оно поддерживает протокол обмена *USB* и снижает затраты разработчика;

- преобразователь цифрового кода в аналоговый сигнал – преобразует входной цифровой код, поступающий из микроконтроллера, в аналоговое напряжение;

- микроконтроллер – управляет работой всех вышеперечисленных блоков модуля, формируя необходимые стробы, а также хранит данные, которые необходимо передать (прочитать) в (из) Объекта управления (ОУ).

Схема электрическая структурная модуля ввода-вывода информации имеет шифр ИВБП.010100.2308Э1.

Таким образом, анализируя ТЗ и функции модуля ввода-вывода информации, можно выделить два канала передачи данных.

* 1. Канал ввода цифровой информации

Режим *bulk* используется для передачи массивов данных. В этом режиме нет обязательств по скорости передачи, задержки доставки. Она может занимать всю свободную полосу пропускания шины. Приоритет этих передач самый низкий, они могут приостанавливаться при большой загрузке шины. Доставка гарантирована, при случайной ошибке выполняется повтор.

Все пересылки данных управляются хостом. Хост-контроллер циклически с периодом 1 мс формирует кадры, в котором укладываются все транзакции.

При запросе передачи на ввод от хост контроллера (после того как закончилось конфигурирование конечных точек на ввод и вывод) микроконтроллер выдаёт сигнал *P\_RD* объекту управления. Этим сигналом инициируется передача данных из ОУ в микроконтроллер. Затем ОУ выдаёт на шину данных, связывающих ОУ и микроконтроллер, первые 8 бит информации и сигнал *P\_WR*, по приходу которого микроконтроллер считывает байт с шины данных. Затем снова выдаётся сигнал запроса на чтение из ОУ *P\_RD*, и так повторяется до тех пор, пока микроконтроллер не прочитает 16 байт информации. Далее эти 16 байт передаются в специализированное *USB* устройство, и формируется пакет, для передачи хосту.

1.2 Канал вывода аналоговой информации

При запросе передачи на вывод из хоста, специализированное *USB* устройство получает пакет с данными. Эти данные передаются микроконтроллеру по высокоскоростному восьмиразрядному параллельному каналу передачи данных. Микроконтроллер по мере поступления данных выдаёт их на цифро-аналоговый преобразователь, который в свою очередь переводит данные из цифровой формы в аналоговый сигнал. Аналоговый сигнал передаётся в объект управления. Питание к цифро-аналоговому преобразователю подводится из объекта управления (*Udac* = −5,2 В и *Uofst* = +3В) и непосредственно с шины *USB*.

Как для канала ввода, так и для канала вывода все передачи происходят по инициативе хост контроллера. Конфигурирование конечных точек специализированного *USB* устройства происходит программным путём. Скоростной режим инициализируется программным путём (подключение резистора к линии *D*+ или *D*−). При конфигурировании конечных точек размер буфера будет определён в соответствии с техническим заданием (ТЗ). Скорость обмена с хост контроллером, указанная в ТЗ, зависит от загруженности канала шины *USB*.

Так как специализированное *USB* устройство (в частности микросхема *PDIUSBD*12 фирмы *Phillips Semiconductors*, используемая в данном курсовом проекте) поддерживает связь по интерфейсу *USB* на уровне протокола, то дополнительных цифровых устройств и времязадающих схем не понадобится, выделение временных интервалов будет реализовано программным путём. Для дальнейшей реализации понадобится лишь ряд дискретных электрорадиоэлементов, таких как кварцевый резонатор, резисторы и конденсаторы.

2 Разработка функциональной схемы

Функциональную схему, исходя из структурной схемы (ИВБП.010100.2308Э1), можно представить как совокупность интерфейсных и исполнительных элементов, которые взаимодействуют между собой. Интерфейсными элементами являются блоки, которые отвечают за связь разрабатываемого модуля ввода-вывода информации с хостом (ЭВМ). Под исполнительными элементами понимаются блоки, отвечающие за непосредственное выполнение основных функций модуля, таких как преобразование цифрового кода в аналоговый сигнал.

Микроконтроллер организует функционирование модуля, путём формирования управляющих стробов, а также связывает интерфейсную и исполнительные части разрабатываемого модуля. В качестве микроконтроллера выбрана микросхема *Intel* 8051 фирмы *Intel*. На рисунке 2.1 показана структурная схема микроконтроллера семейства 8051. Назначение выводов микроконтроллера будет показано на схеме электрической функциональной модуля ввода-вывода информации.

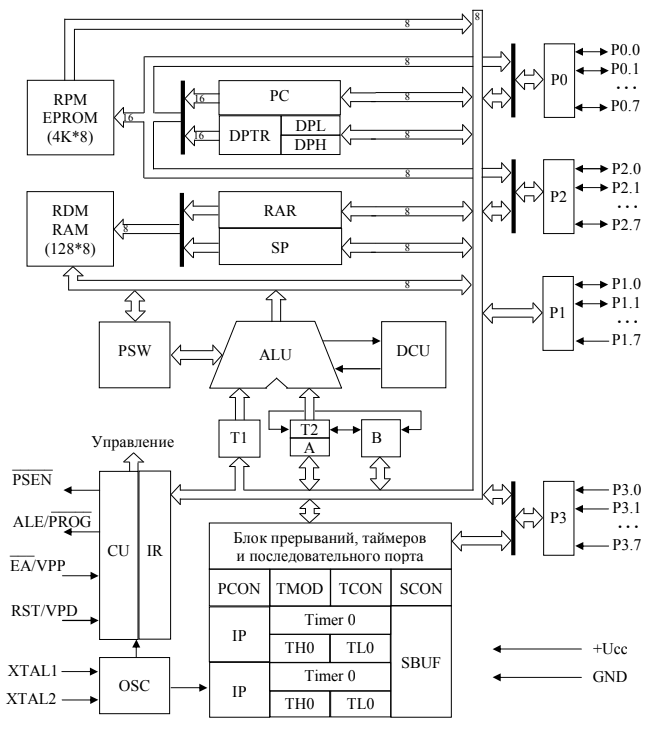


Рисунок 2.1 – Структурная схема микроконтроллера КМ1816ВЕ51 (*i*8051)

Исходя из вышеперечисленного становится очевидным следующее деление модуля:

- интерфейсная часть. Специальное *USB* устройство, в качестве которого выбрана готовая микросхема *PDIUSBD*12 фирмы *Philips Semiconductors*. *PDIUSBD*12 полностью соответствует *USB* спецификации версии 2.0. Также поддерживает связь по интерфейсу *USB* на уровне протокола;

- исполнительная часть. Преобразователь цифрового кода в аналоговый сигнал, в качестве которого выбран быстродействующий, восьмиразрядный цифро-аналоговый преобразователь (ЦАП) К1118ПА3 (время преобразования 20 нс). Техническим заданием чётко оговорена максимальная ошибка преобразования цифровых данных в аналоговый сигнал. Эта величина составляет ε = 1%. Эта величина вносит жёсткое ограничение на разрядность ЦАП и находится из соотношения:



,

.

Откуда получаем, что , таким образом точность выбранного ЦАП является приемлемой.

Другим параметром, определяющим выбор ЦАП, является частота дискретизации.

По теореме Котельникова-Шеннона получаем, что



Где  – частота дискретизации, а *fmax* – частота наивысшей гармоники, в нашем случае fmax = 300 Гц. Тогда ≥ 600 Гц. Выбор частоты дискретизации будет произведён на этапе разработки программного обеспечения, так как выбор временных промежутков будет реализован не на аппаратном, а на программном уровне и формировать временные отчёты будет *USB*-хост (ЭВМ). Для выбранного ЦАП максимальная частота дискретизации составляет 50 МГц, что удовлетворяет требованиям.

Также в техническом задании указан вид аналогового сигнала: однополярное напряжение 0-3 В. Микросхема выбранного ЦАП выдаёт выходное напряжение от –3 В до 0 В. Но выбранный ЦАП позволяет смещать диапазон выходного напряжения (для этого на соответствующий вывод подаётся напряжение смещения +3В).

Такое разбиение разрабатываемого модуля ввода-вывода информации на два типа элементов упрощает понимание его работы, так как происходит разделение управляющих и исполнительных функций.

Схема электрическая функциональная модуля ввода-вывода информации имеет шифр ИВБП.010100.2308Э2.

2.1 Интерфейсные блоки модуля

Специальное USB устройство *PDIUSBD*12.

Данный блок взаимодействует с блоком микроконтроллера *Intel* 8051 посредством следующих сигналов: *А*0 – данный сигнал отвечает за выборку фазы команд (*А*0 = 1) и фазы данных (*А*0 = 0) и должен формироваться программно, *INT*\_*N* – данный сигнал формируется *PDIUSBD*12 и сообщает микроконтроллеру о том что необходимо принять данные от хоста или передать данные в хост (активный 0 уровень), *WR* – с помощью данного строба происходит запись информации из микроконтроллера в порт *PDIUSBD*12 (активный уровень 0), *RD* – с помощью данного строба происходит чтение информации из порта *PDIUSBD*12 в микроконтроллер (активный 0 уровень), *CLK* - сигнал синхронизации для микроконтроллера. Передача данных между микроконтроллером и *PDIUSBD*12 осуществляется через восьмиразрядную шину данных *DATA*0-*DATA*7.

Ко входам *XTAL*1 и *XTAL*2 подключается отечественный кварцевый резонатор РК169МВ-7АП с частотой 6MГц. На вход *СS*\_*N* (выбор чипа) подаётся активный уровень 0 (земля). На вход *ALE* (включение закрытия адреса) также подаётся 0 для разделения конфигурации шины адрес/данные.

2.2 Исполнительные блоки модуля

Блок преобразователя цифрового кода в аналоговый сигнал.

В цифровым входам ЦАП подключён порт *P*2 микроконтроллера, а также подведено питание (+5 В от шины USB; –5,2 В и напряжение смещения от объекта управления). Преобразование цифровой информации в аналоговый сигнал происходит непрерывно. Выходное напряжения соответствует требованиям технического задания и лежит в диапазоне 0-3В и напрямую передаётся в объект управления.

3 Разработка принципиальной схемы

3.1 Обоснование элементной базы

В качестве микроконтроллера выбрана микросхема *Intel* 8051. Это простая в использовании и недорогая схема, содержащая четыре восьмиразрядных параллельных порта ввода/вывода (причём каждый бит может быть настроен на ввод или на вывод независимо от состояния других битов в порте, а некоторые – на альтернативные функции) и один последовательный порт. На принципиальной схеме эта микросхема обозначается *DD*1.

3.2 Разработка интерфейса

Специализированное *USB* устройство *PDIUSBD*12 (*DD*2) полностью поддерживает протокол USB, поэтому оно очень удобно для проектирования устройства. Оно соединяется с микроконтроллером *Intel* 8051 (*DD*1) через восьмиразрядную шину данных *Р*0.0…*Р*0.7. Узел соединяется с шиной *USB* посредством линий *D*+ и *D*–. К линии *D*+ может быть подключён согласующий резистор R номиналом 1,5 КОм, данный резистор уже встроен в *PDIUSBD*12 и может быть подключён программным путём [2]. Данный тип подключения свидетельствует, что устройство, присоединяемое к шине *USB*, является полно скоростным (*FS*-устройство).

3.3 Разработка преобразователя информации

Выбор цифро-аналогового преобразователя ЦАП (*DD*3) обоснован на этапе разработки функциональной схемы. Данный ЦАП подключён к микроконтроллеру (*DD*1) по типовой схеме, с подачей напряжения смещения на выходы для расширения диапазона выходного сигнала.

3.4 Разработка элементов модуля

Схема включения ЦАП определяет режим его работы. ЦАП включается по типовой схеме [9], с подачей напряжения смещения на выходы для расширения диапазона выходного сигнала. При этом обвязка ЦАП показана на рисунке 3.1.



Рисунок 3.1 – Типовая схема подключения обвязки к ЦАП К1118ПА3

При таком включении выходное напряжение преобразования, которое лежит в диапазоне -3…0 В можно сместить на 3 В. При этом получим выходное напряжение преобразования, указанное в техническом задании.

Для подключения к объекту управления используется разъем СНП59-16/94×11B-23-1 (*Х*1) на 16 контактов. Для подключения к шине *USB* используется разъем *USBА* – 1*J* (*Х*2).

Для предотвращения паразитных высокочастотных колебаний в цепи питания +5 В установлены керамические конденсаторы KM-5б-П33-0,1 мкФ 10 % (*С*8 – *С*10), которые размещаются в непосредственной близости от микросхем [4]. Для подавления низкочастотных помех цепи питания +5 В к ней подключается электролитический конденсатор К-53-14-25В-10 мкФ 20% (*С*7). Этот конденсатор устанавливается в непосредственной близости от разъёма [4]. (Будем считать, что линии питания ЦАП и линия подачи напряжения смещения идут от объекта управления и уже профильтрованы.)

Кварцевый резонатор *BQ*1 – резонатор РК169МA-7АП-6МГц. Для снятия пульсаций с кварцевого резонатора к нему подключается конденсатор KM-5б-П33-0,1мкФ 10 % (*С*3).

Резистор *R*1 служит для задания уровня логической единицы, обычно его выбирают равным 10 кОм [4].

В устройстве спроектированы две схемы автоматического сброса при кратковременном пропадании питания, так как микроконтроллер и специализированное *USB* устройство сбрасываются разным уровнем. Микроконтроллер сбрасывается активным уровнем, поэтому при использовании *Intel* 8051 достаточно включить конденсатор ёмкостью 0,1 мкФ (*С*1) между шиной питания и входом запуска.

*PDIUSBD*12сбрасывается низким потенциалом, поэтому используется классическая схема сброса, построенная на конденсаторе *С*2, диоде *VD*1 и резисторе *R*2 [4].

3.5 Расчёт токовой нагрузки и потребляемой мощности

Перед разработкой программного обеспечения необходимо провести расчёт токовой нагрузки для выбора источника питания (от шины или независимого). ЦАП требует дополнительного питания –5,2 В и +3 В, поэтому это питание должен обеспечивать объект управления, на разъёме (*X*1) для него предусмотрены контакты.

*DD*1: *Imax* = 160 мА, *U*пит = +5 В, *P*max = 800 мВт [3],

*DD*2: *Imax* = 100 мА, *U*пит = +5 В, *P*max = 500 мВт [2],

*DD*3: *Imax* = 20 мА, *U*пит = +5 В, *P*max = 80 мВт [9].

Таким образом, суммарный потребляемый ток по шине USB не превышает 280 мА, следовательно, можно запитывать логические устройства от шины USB.

Максимальный ток потребления ЦАП по питанию –5,2 В составляет *Imax* = 80 мА [8]. Таким образом *Pmax* = 80 мА ∙ 5,2 В = 416 мВт.

4 Разработка программного обеспечения

4.1 Программная модель устройства

В микросхеме *PDIUSBD*12 предусмотрены три конечные точки, с которыми взаимодействует хост. Нулевая конечная точка предназначена для конфигурации устройства (первоначально хост посылает ей пакет *SETUP*). Первая и вторая конечные точки служат для обмена данными между устройством и хостом. Каждая конечная точка содержит входной и выходной буфер. Для каждой конечной точки определены четыре режима обмена, исходя из условий технического задания, будем использовать неизохронный режим 0. Этот режим предназначен для режимов обмена *bulk* и с прерываниями. В данном режиме микросхема формирует все сигналы квитирования (*ACK*, *NAK*, *STALL*) в зависимости от состояний буферов и обрабатывает маркеры пакетов (*IN*, *OUT*, *SETUP*) помещая выходные данные от хоста в соответствующие буферы данной конечной точки (*OUT*) и передавая хосту данные из соответствующих буферов конечной точки (*IN*). Когда хост формирует один из пакетов микросхема генерирует прерывание (*INT*), микроконтроллер анализирует регистр прерываний и переходит к выполнению процедуры считывания или записи данных в один из буферов определённой конечной точки. В регистре прерываний шесть битов соответствуют входному или выходному буферу каждой из трёх точек. Поэтому разработчик может не заботится о дешифрации пакетов. Основная задача сводится к следующему:

- конфигурирование конечных точек (настройка режимов, временных интервалов, реализация ответов на основные запросы хоста);

- ввод цифрового сигнала от объекта управления в хост;

- получение от хоста цифрового сигнала, преобразование его в аналоговый и передача его в объект управления по выше оговорённому интерфейсу.

Перейдём к реализации программной модели устройства.

Для конечной точки 1 будем использовать режим вывода по прерыванию. В техническом задании оговорена частота выходного сигнала 0 – 300 Гц. Так как максимальная частота опроса по прерыванию равно 1 кГц (время опроса 1 мс), то этого достаточно для условий технического задания. Максимальный размер пакета для 1 конечной точки составляет 16 байт.

Для конечной точки 2 будем использовать режим ввода *bulk*. Пересылка данных от объекта управления в микроконтроллер осуществляется последовательно, и рассматривалась выше.

Сброс устройства реализован аппаратно таким образом, что все устройства будут сброшены в первоначальное состояние при отключении питания, поэтому по начальному адресу в микроконтроллере (с которого начинает выполняться программа) должна храниться процедура инициализации, далее микроконтроллер должен находиться в бесконечном цикле и выходить из него только по прерыванию.

4.2 Блок-схемы алгоритмов программы функционирования модуля

На рисунке 4.1 представлена блок-схема макроуровня программы функционирования модуля. Блок «возможные прерывания» отмечен курсивом, так как прерывания возникают лишь в определенные промежутки времени. Фактически тело программы представляет собой бесконечный цикл без каких либо действий (ожидание возникновения прерывания). При этом ввод и вывод данных в хост осуществляется по запросу хоста, который вызывает прерывание *PDIUSBD*12. В регистре прерывания находится код транзакции (*IN*, *OUT*) и номер конечной точки.



Рисунок 4.1 – Макроуровень программы функционирования модуля

В качестве обработчика событий прерывания процесс чтения цифрового канала, блок-схема алгоритма представлена на Рисунке 4.2.



Рисунок 4.2 – Блок-схема обработчика прерываний.

Наличие обращения к той или иной конечной точке устанавливается путём анализа регистра прерываний микросхемы *PDIUSBD*12. Обработчиков прерываний на ввод конечной точки 1 и на вывод конечной точки 2 не предусмотрено, так как для каждой из конечных точек (кроме нулевой) определён только один режим при конфигурации конечной точки

Рассмотрим обработку ввода информации подробнее (рисунок 4.3)



Рисунок 4.3 – Блок-схема ввода цифровой информации с ОУ

Условие «конечная точка доступна» и «буфер ввода пуст» проверяется путём анализа соответствующих битов, после выполнения команды «чтение состояния конечной точки». «Запись любого байта» – формальная операция, так как байт в буфере по нулевому смещению зарезервирован и может принимать любое значение [2].

Рассмотрим обработку вывода информации подробнее (рисунок 4.4)



Рисунок 4.3 – Блок-схема вывода цифровой информации от хоста

Всеми передачами управляет хост. Все пакеты с запросами на передачу хост посылает нулевой конечной точке, так как используется неизохронный режим 0. Причём на нулевую конечную точку может придти пакет *SETUP* или пакет с данными. Пакет *SETUP* (8 байт) содержит код стандартного запроса. Пришедший пакет SETUP разбирается обработчиком стандартных запросов. Если же пришёл пакет с данными, то он не обрабатывается, то есть выполняется команда «Очистить буфер».

Устройство должно отвечать на стандартные запросы от хоста. Это необходимо для согласования условий передачи данных от хоста и в хост. Из всех стандартных запросов будем обрабатывать только минимально необходимые: получить/установить конфигурацию устройства (*GET*/*SET\_CONFIGURATION*) (рисунок 4.4 и рисунок 4.5), установить адрес устройства (*SET\_ADRESS*) (рисунок 4.6) и получить дескриптор устройства (*GET\_DESCRIPTOR*). Если обрабатываемый запрос на установку конфигурации либо адреса, то необходимо сформировать пакет подтверждение хосту. Процедура «Заполнение буфера ввода нулями» подразумевает запись нуля только во второй байт пакета, что означает пакет нулевой длины.



Рисунок 4.4 – Блок-схема получения конфигурации устройства



Рисунок 4.5 – Блок-схема установки конфигурации устройства



Рисунок 4.6 – Блок-схема установки адреса устройства



Рисунок 4.7 – Блок-схема получение дескриптора

Во всех устройствах должен быть предусмотрен дескриптор устройства и

по крайней мере один дескриптор конфигурации. Если устройство не поддерживает запрошенный дескриптор, оно отвечает, останавливая канал, используемый для запроса. Ненулевое значение первого байта дескриптора указывает, что буфер содержит допустимый дескриптор. Запрос о конфигурации дескриптора возвращает в одном запросе дескриптор конфигурации, все дескрипторы интерфейса, и дескрипторы конечной точки для всех интерфейсов. Первый дескриптор интерфейса немедленно следует за дескриптором конфигурации. Дескрипторы конечной точки для первого интерфейса следуют за первым дескриптором интерфейса. Если имеются дополнительные интерфейсы, их дескриптор интерфейса, и дескрипторы конечной точки следуют за дескрипторами конечной точки первого интерфейса [1].

Заключение

В результате проделанной работы был разработан модуль ввода-вывода, сопрягаемый с ЭВМ через шину *USB*, к данном модулю были разработаны структурная, функциональная и принципиальные схемы. Основные требования технического задания можно считать выполненными. Вся документация представлена в приложении и в пояснительной записке.

Список использованных источников

1. Вишняков В. А., Беляев О. А. Универсальная последовательная шина

USB: Пособие. – Рыбинск: РГАТА, 2004.

1. Вишняков В.А., Рахманин Д. А. Специализированное USB устройство

PDIUSBD12: Пособие. – Рыбинск: РГАТА, 2004.

1. Вишняков В.А., Соколов С. Ю. Микроконтроллеры семейства MCS-51:

Пособие. – Рыбинск: РГАТА, 2004.

Приложение А

Чертежи

Для реализации заданных функций возьмём микроконтроллер *PIC18F2550*, который поддерживает высокоскоростной режим передачи.

*PIC18F2550* – специализированный микроконтроллер фирмы *MicroChip* со встроенным *USB* интерфейсом. Кроме реализации протокола обмена, используя *USB* интерфейс, МК реализует необходимые преобразования информации и управляет всеми остальными узлами модуля.

Данный микроконтроллер обладает следующими достоинствами:

- высокопроизводительное процессорное ядро на основе RISC-архитектуры;

- интегрированный *USB*-интерфейс, соответствующий стандарту *V*2.0;

- возможность работы процессорного ядра и *USB*-модуля на разных тактирующих частотах;

- поддержка основных протоколов обмена при работе в высокоскоростном режиме (12 Mбит/с);

- интегрированный *USB*-трансивер;

- 10-канальный 10-битный высокопроизводительный АЦП;

- четыре программируемых таймера с программируемым делителем частоты;

- два модуля захвата, сравнения и широтно-импульсной модуляции;

- 32 *Kb Flash*-памяти для хранения управляющей программы, 2 *Kb* ОЗУ.

- низкое энергопотребление;

- широкий диапазон питающих напряжений от 2 до 5,5 В;

- возможность питания как от шины, так от внешнего источника.

На рисунке 1.2 приведена структурная схема микроконтроллера *PIC18F2550*.

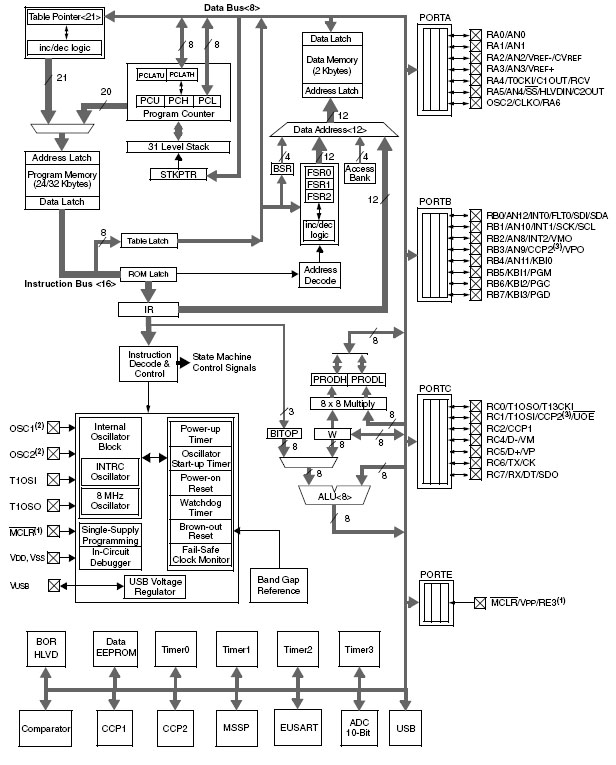


Рисунок 1.2 – Структурная схема микроконтроллера *PIC18F2550*

* 1. Цифровой ввод информации

Приём данных от хоста осуществляет интегрированный USB – модуль

(рисунок 1.3).

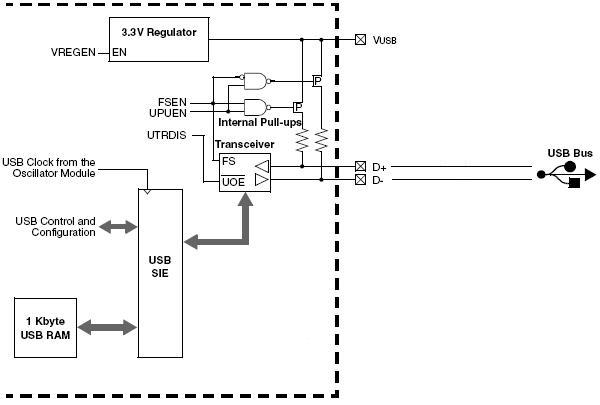


Рисунок 1.3 – Структура USB – модуля

В данном модуле используется приемопередатчик (*Transceiver*), внутренний регулятор (*Regulator*) и блок последовательного интерфейса (*USB SIE*), который производит кодирование-декодирование данных, вставку и извлечение бита, формирует биты проверки на четность, идентифицирует тип передатчика, автоматически формирует сигналы *ACK* и *NACK.* Опорные синхроимпульсы формируются в модуле регулятора и поступают на *USB* *SIE*. Регулятор может работать на частоте 48 МГц, а его параметры настраиваются в регистре *OSCTUNE* битами *TUNE*4 – TUNE0.

* 1. Аналоговый вывод информации

Выводимый аналоговый сигнал представляет собой импульсное напряжение переменной частоты. Для его формирования целесообразно использовать модуль широтно-импульсной модуляции, входящий в состав данного микроконтроллера. Модуль позволяет задавать период и длительность импульса. Для формирования частоты выходного сигнала используются программируемый делитель частоты и таймер для точной подстройки частоты. Структура модуля ШИМ представлена на рисунке 1.4.



Рисунок 1.4 – Структурная схема широтно-импульсного модулятора

Таймер (*TMR*2) используется для счета тактовых импульсов. Значение таймера сравнивается при помощи двух компараторов со значениями регистров длительности импульса и периода импульса. Выходные сигналы с компараторов подаются на *RS*-триггер формирующий выходной сигнал модуля ШИМ.

Электрическая структурная схема находится в приложении А.

2 Разработка функциональной схемы модуля

Функциональную схему, исходя из структурной схемы (ИВБП.010100.2308Э1), можно представить как совокупность интерфейсных и исполнительных элементов, которые взаимодействую между собой. Интерфейсными элементами являются блоки, которые отвечают за связь разрабатываемого модуля ввода-вывода информации с ЭВМ. Под исполнительными элементами понимаются блоки, отвечающие за непосредственное выполнение основных функций модуля, таких как преобразование аналогового сигнала в цифровой код, согласование сигнала и так далее.

2.1 Обоснование функциональной схемы

Для решения поставленной задачи необходим цифро-аналоговый

преобразователь. Чтобы обеспечить максимальную ошибку преобразования в 1 %, необходимо выполнить условие:



где n – разрядность ЦАП;

 – максимальная ошибка преобразования.

Так как (1 % по техническому заданию), то получаем:



из чего  . Отсюда следует, что .

Схема согласования представляет собой инвертирующий усилитель (рисунок 2.1). Коэффициент усиления такого звена [4].



Рисунок 2.1 – Инвертирующий усилитель

1 Основные понятия *Matplotlib*

Из работы [1] можно сделать вывод, что при разработке данной библиотеки было заложено множество понятий для более гибкой и глубокой работы с графиками. На рисунке 1.1 изображён график и отмечены компоненты, которые программист может модифицировать.

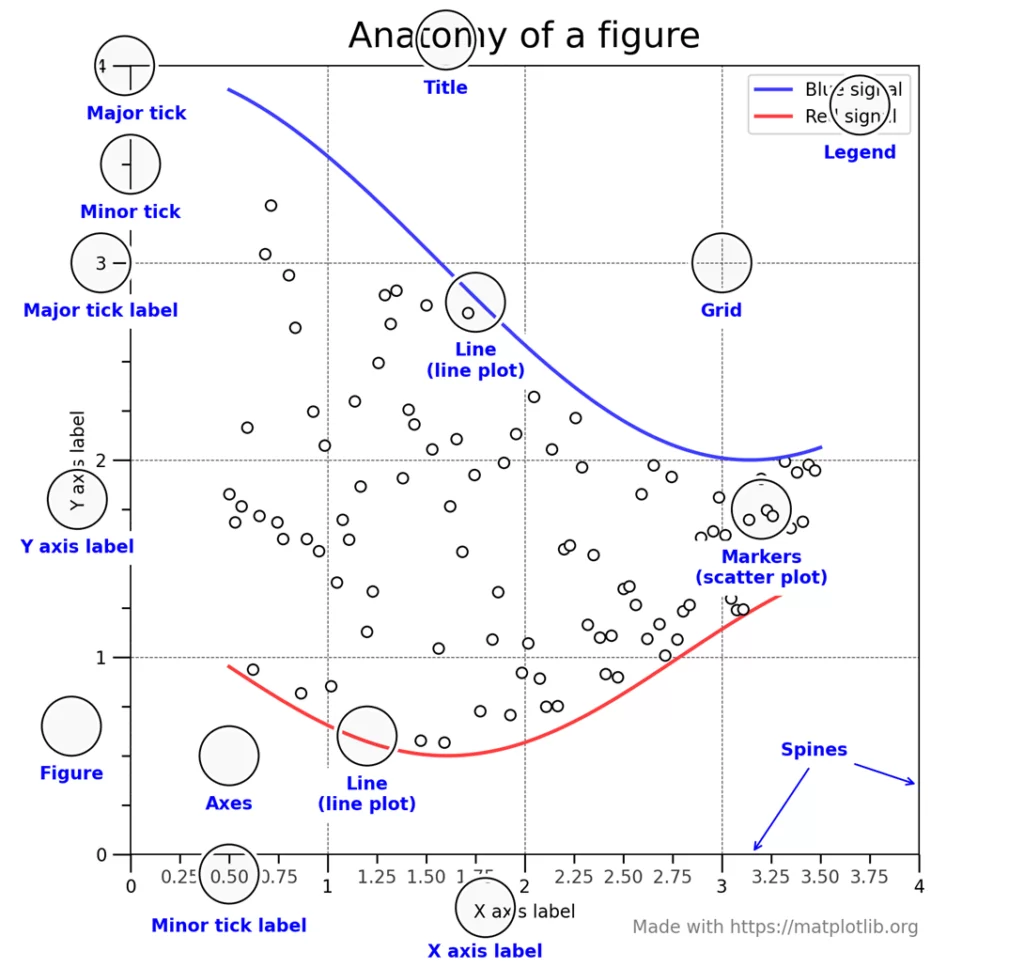


Рисунок 1.1 – Все компоненты графика *Matplotlib*

Всё пространство (*Figure*) включает в себя график (*Axes*), который, в свою очередь, состоит из таких элементов, как границы (*Spines*), сетки (*Grid*), описания (*Legend*), самого графика (*Line*) и других.

Графики отражают зависимость, при этом элементы графика – оси, подписи и др. – открыты к настройке.

Базовый элемент *Figure* может содержать в себе как один график, так и несколько.

2 Основные модули и функции *Matplotlib API*

2.1 Модуль *pyplot*

Данный модуль представляет из себя коллекцию функций, которые позволяют использовать библиотеку *Matplotlib* так же как и пакет прикладных программ *MATLAB*. Это основа основ для построения всевозможных графиков и диаграмм. Вот несколько основных целей и возможностей модуля *pyplot*:

- создание графиков: *pyplot* позволяет легко создавать различные типы графиков, такие как линейные графики, точечные графики, столбчатые диаграммы, круговые диаграммы и т.д. Это особенно полезно при визуализации данных для анализа и понимания их распределения или зависимостей;

- настройка внешнего вида графиков: *pyplot* предоставляет множество функций для настройки внешнего вида графиков, включая заголовки, метки осей, легенды, цвета и стили линий. Это позволяет создавать профессионально выглядящие графики;

- работа с множеством графиков: *pyplot* обеспечивает возможность создания нескольких графиков на одной фигуре, что полезно для сравнения данных или отображения различных аспектов информации;

- сохранение графиков: *pyplot* позволяет сохранять созданные графики в различных форматах, таких как *PNG*, *PDF*, *SVG* и других. Это полезно, если вы хотите использовать графики в презентациях, документах или веб-сайтах.

2.2 Функция *plot*

Функция *plot* библиотеки *Matplotlib*используется для создания различных типов графиков. Эта функция является одним из ключевых инструментов для визуализации данных. Вот основные задачи, которые решает функция *plot*:

- создание линейных графиков: функция применяется для построения линейных графиков, которые отображают зависимость одной переменной от другой. Это может быть полезно для визуализации трендов, изменений со временем и других подобных зависимостей;

- настройка внешнего вида графика: функция позволяет настраивать внешний вид графика, такой как цвет линии, стиль линии, маркеры данных и другие параметры. Это позволяет создавать графики с нужным визуальным эффектом;

- отображение нескольких линий на одном графике: функция может использоваться для отображения нескольких линий на одном графике. Это может быть полезно при сравнении различных наборов данных или результатов;

- создание различных типов графиков: хотя функция *plot* чаще всего используется для линейных графиков, он также может строить другие типы графиков, такие как точечные графики, ступенчатые графики и др., в зависимости от переданных параметров.

Исходя из этого, можно сделать вывод, что данная функция предоставляет множество параметров для настройки внешнего вида графиков и обеспечивает широкий спектр возможностей для визуализации данных в *Matplotlib*.

2.3 Функция *subplot*

Функция *subplot* библиотеки Matplotlib используется для создания сетки графиков, позволяя размещать несколько графиков в одном окне. Это полезно для отображения различных аспектов данных или сравнения нескольких наборов данных на одном изображении. Вот основные задачи и возможности функции *subplot*:

- создание сетки графиков: функция *subplot*позволяет разбить фигуру на сетку из подграфиков, определяемых числами строк и столбцов. Это делает возможным размещение нескольких графиков на одной фигуре;

- отображение нескольких графиков: *subplot* позволяет отображать несколько графиков на одной фигуре, что полезно для сравнения данных или отображения различных аспектов информации;

- настройка расположения графиков: функция *subplot*позволяет управлять расположением и размером подграфиков в сетке. Параметры *rowspan*и *colspan*могут быть использованы для указания количества строк или столбцов, которые должен занимать подграфик.

Исходя из этого, можно сделать вывод, что данная функция предоставляет гибкую возможность управления размещением и отображением нескольких графиков на одной фигуре в библиотеке *Matplotlib*.

2.4 Функции *pie, bar* и *barh*

Функции *pie*, *bar* и *barh*в библиотеке *Matplotlib* предназначены для создания различных типов графиков и представления данных. Вот их краткое описание:

- *pie*: позволяет строить круговые диаграммы, представляющие из себя относительные доли частей в целом;

- *bar*: используется для построения вертикальных столбчатых диаграмм, где каждый столбец представляет отдельное значение;

- *barh*: позволяет строить горизонтальные столбчатые диаграммы, где каждый столбец расположен горизонтально.

Эти функции полезны при визуализации и сравнении данных различных категорий. Выбор между ними зависит от специфики данных и предпочтений при отображении информации.

2.5 Функции *show* и *savefig*

Функция *show* из библиотеки *Matplotlib* используется для отображения графиков и диаграмм, которые были созданы с использованием этой библиотеки. После того, как все необходимые изменения в графике были проведены, вызов функции отображает окно с самим графиком, который программист ранее создал.

Функция *savefig*из библиотеки *Matplotlib* используется для сохранения текущего графика в файле. Эта функция предоставляет возможность сохранить изображение в различных форматах, таких как *PNG*, *PDF*, *SVG* и другие. У данной функции есть несколько параметров:

- *fname.* Данный параметропределяет имя файла в котором будет сохранённый график. Можно указать полный путь для сохранения файла в определённом месте;

- *dpi*. Данный параметрпозволяет сохранить график с определённым разрешением;

- *format*. Определяет формат файла в котором сохранится график. Если не определён, то используется формат из параметра *fname*. Если в параметре *fname* не указан формат файла, то по умолчанию используется *PNG.*

3 Примеры использования *Matplotlib*

*Matplotlib* можно использовать для отображения широкого круга разнообразных графиков. Простая аппроксимация показана на рисунке 3.1 и листинг программы находится в приложении А.

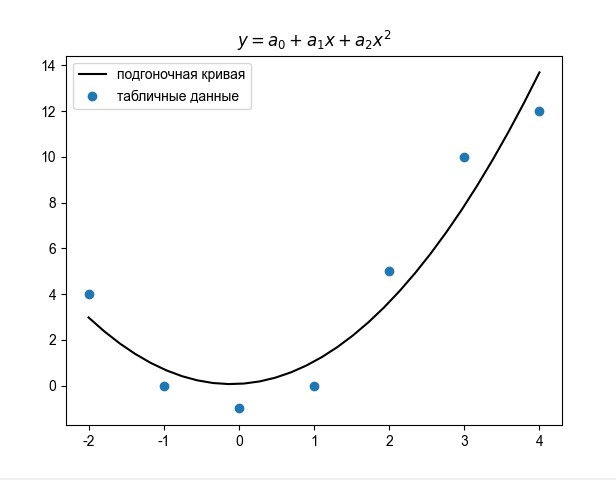


Рисунок 3.1 – График аппроксимации

Библиотека позволяет изображать не только простенькие графики аппроксимации или интерполяции, но и создать график триангуляции Делоне (рисунок 3.2). Листинг программы находится в приложении Б.

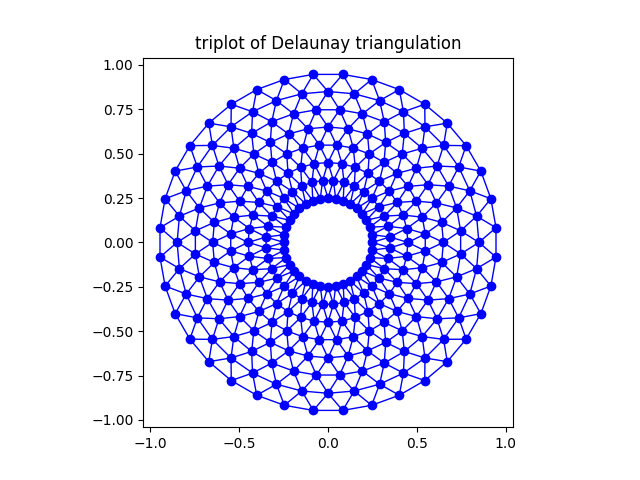


Рисунок 3.2 – Триангуляция Делоне

В работе [2] представлен график сигнала пульсара (рисунок 3.3) известный тем, что он изображён на обложке альбома «*Unknown Pleasures*»группы *Joy Division.* Листинг программы находится в приложении В.



Рисунок 3.3 – Сигнал пульсара

Заключение

*Matplotlib API* – это мощный инструмент для создания красивых и информативных графиков и диаграмм в *python*. В данной работе был представлен и изучен основной функционал библиотеки, который позволяет построить график, изобразить его и сохранить с нужным расширением для дальнейшего пользования.Были представлены несколько графиков, реализованных с помощью *Matplotlib API.* Таким образом, можно сделать вывод, что благодаря *Matplotlib API* пользователи могут визуализировать данные эффективно и привлекательно, делая их анализ более понятным и убедительным.

Список использованных источников

1. *Matplotlib: Visualization with Python* [электронный ресурс] *URL*:

*https*://*matplotlib.org*/*stable*/*users*/*explain*/*quick\_start*.*html*

1. *Matplotlib UNCHAINED* [электронный ресурс] *URL*:

https://matplotlib.org/stable/gallery/animation/unchained.html

Приложение А

Исходный текст программы для отображения графика аппроксимации

def grafik():  
 global x, y  
 x1 = numpy.linspace(-2, 4, 30)  
 y1 = numpy.linspace(-1, 12, 30)  
 data = podgonka(x, y)  
 print(data)  
 data1 = [data[0][-1:], data[1][-1:], data[2][-1:]]  
 data2 = [data[0][:-1], data[1][:-1], data[2][:-1]]  
 print("Значения a0, a1 и a2:")  
 param = numpy.linalg.solve(data2, data1)  
 print(numpy.linalg.solve(data2, data1))  
 print("Новые Y")  
 # newData = calculate(x, [param[2][0], param[1][0], param[0][0]])  
 newData = calculate(x1, [param[2][0], param[1][0], param[0][0]])  
 draw(x, newData, y)  
 form = r'$\left\{\begin{matrix} a\_{0}n+a\_{1}\sum x + a\_{2}\sum x^{2}=\sum y\\ ' \  
 r'a\_{0}\sum x+a\_{1}\sum x^{2} + a\_{2}\sum x^{3}=\sum yx \\ ' \  
 r'a\_{0}\sum x^{2}+a\_{1}\sum x^{3} + a\_{2}\sum x^{4}=\sum yx^{2} \end{matrix}\right.$'  
 mpl.rcParams['font.sans-serif'] = ['Arial']  
 mpl.rcParams['axes.unicode\_minus'] = False  
 plt.title(r"$y=a\_{0}+a\_{1}x+a\_{2}x^{2}$")  
 plt.legend(loc="upper left")  
 plt.show()

def podgonka(data\_x, data\_y):  
 size = len(data\_x)  
 i = 0  
 sum\_x = 0  
 sum\_sqare\_x = 0  
 sum\_third\_power\_x = 0  
 sum\_four\_power\_x = 0  
 average\_x = 0  
 average\_y = 0  
 sum\_y = 0  
 sum\_xy = 0  
 sum\_sqare\_xy = 0  
 while i < size:  
 sum\_x += data\_x[i]  
 sum\_y += data\_y[i]  
 sum\_sqare\_x += math.pow(data\_x[i], 2)  
 sum\_third\_power\_x += math.pow(data\_x[i], 3)  
 sum\_four\_power\_x += math.pow(data\_x[i], 4)  
 sum\_xy += data\_x[i] \* data\_y[i]  
 sum\_sqare\_xy += math.pow(data\_x[i], 2) \* data\_y[i]  
 i += 1  
 average\_x = sum\_x / size  
 average\_y = sum\_y / size  
 # print([[size, sum\_x, sum\_sqare\_x, sum\_y],  
 # [sum\_x, sum\_sqare\_x, sum\_third\_power\_x, sum\_xy],  
 # [sum\_sqare\_x,sum\_third\_power\_x,sum\_four\_power\_x,sum\_sqare\_xy]])  
 return [[size, sum\_x, sum\_sqare\_x, sum\_y],  
 [sum\_x, sum\_sqare\_x, sum\_third\_power\_x, sum\_xy],  
 [sum\_sqare\_x, sum\_third\_power\_x, sum\_four\_power\_x, sum\_sqare\_xy]]  
  
# Вычислить значение подобранной кривой  
def calculate(data\_x, parameters):  
 data\_y = []  
 for x in data\_x:  
 data\_y.append(parameters[2] + parameters[1] \* x + parameters[0] \* x \* x)  
 print("data\_y:")  
 print(data\_y)  
 return data\_y  
  
# Функция draw рисует наши кривые на координатной плоскости  
def draw(data\_x, data\_y\_new, data\_y\_old):  
 global x, y  
 x1 = numpy.linspace(-2, 4, 30)  
 plt.plot(x1, data\_y\_new, label="подгоночная кривая", color="black")  
 plt.scatter(data\_x, data\_y\_old, label="табличные данные")  
x = [-2, 0, 1, 3, 5, 6, 8]  
y = [5, -1, 2, 10, 24, 36, 38]  
grafik()

Приложение Б

Исходный текст программы для отображения триангуляции Делоне

import matplotlib.pyplot as pltimport

matplotlib.tri as tri

import numpy as np

# Сначала создаём координаты x и y Для точек  
n\_angles = 36  
n\_radii = 8  
min\_radius = 0.25  
radii = np.linspace(min\_radius, 0.95, n\_radii)  
  
angles = np.linspace(0, 2 \* np.pi, n\_angles, endpoint=False)  
angles = np.repeat(angles[..., np.newaxis], n\_radii, axis=1)  
angles[:, 1::2] += np.pi / n\_angles  
  
x = (radii \* np.cos(angles)).flatten()  
y = (radii \* np.sin(angles)).flatten()  
  
# Производим триангуляцию  
triang = tri.Triangulation(x, y)  
  
# Убираем ненужные треугольники  
triang.set\_mask(np.hypot(x[triang.triangles].mean(axis=1),  
 y[triang.triangles].mean(axis=1))  
 < min\_radius)

# Строим график

fig1, ax1 = plt.subplots()  
ax1.set\_aspect('equal')  
ax1.triplot(triang, 'bo-', lw=1)  
ax1.set\_title('triplot of Delaunay triangulation')

Приложение В

Исходный текст программы для отображения сигнала пульсара

import numpy as np  
import matplotlib  
import matplotlib.pyplot as plt  
import matplotlib.animation as animation  
  
# Fixing random state for reproducibility  
np.random.seed(19680801)  
  
# Create new Figure with black background  
fig = plt.figure(figsize=(8, 8), facecolor='black')  
  
# Add a subplot with no frame  
ax = plt.subplot(frameon=False)  
  
# Generate random data  
data = np.random.uniform(0, 1, (64, 75))  
X = np.linspace(-1, 1, data.shape[-1])  
G = 1.5 \* np.exp(-4 \* X \*\* 2)  
  
# Generate line plots  
lines = []  
for i in range(len(data)):  
 # Small reduction of the X extents to get a cheap perspective effect  
 xscale = 1 - i / 200.  
 # Same for linewidth (thicker strokes on bottom)  
 lw = 1.5 - i / 100.0  
 line, = ax.plot(xscale \* X, i + G \* data[i], color="w", lw=lw)  
 lines.append(line)  
  
# Set y limit (or first line is cropped because of thickness)  
ax.set\_ylim(-1, 70)  
  
# No ticks  
ax.set\_xticks([])  
ax.set\_yticks([])  
  
# 2 part titles to get different font weights  
ax.text(0.5, 1.0, "MATPLOTLIB ", transform=ax.transAxes,  
 ha="right", va="bottom", color="w",  
 family="sans-serif", fontweight="light", fontsize=16)  
ax.text(0.5, 1.0, "UNCHAINED", transform=ax.transAxes,  
 ha="left", va="bottom", color="w",  
 family="sans-serif", fontweight="bold", fontsize=16)  
  
def update(\*args):  
 # Shift all data to the right  
 data[:, 1:] = data[:, :-1]  
  
 # Fill-in new values  
 data[:, 0] = np.random.uniform(0, 1, len(data))  
  
 # Update data  
 for i in range(len(data)):  
 lines[i].set\_ydata(i + G \* data[i])  
  
 # Return modified artists  
 return lines