МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

ИНСТИТУТ МАШИНОСТРОЕНИЯ

(наименование института полностью)

Кафедра «Промышленная электроника»

(наименование)

11.04.04 «Электроника и наноэлектроника»

(код и наименование направления подготовки)

Электронные приборы и устройства

(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Система диагностики для электроники»

Студент С.Н. Стычев

(И.О. Фамилия) *(личная подпись)*

Научный руководитель к.т.н., Е.С. Глибин

(ученая степень, звание, И.О. Фамилия)

Тольятти 2022

Содержание

Введение

Несмотря на то, что контролю качества изделий электроники в настоящее время уделяется большое внимание, отказы отдельных компонентов или целых систем продолжаются.

Снимать показания и анализировать поведение электронных устройств необходимо для предотвращения поломок или определения их причин с дальнейшей оперативной передачей обратной связи производителю. Современный рынок требует электронные устройства высокой надежности.

Чтобы гарантировать надежность, нужно анализировать электронный компонент не только на стадии проектирования и производства, но и после начала выполнения им своих функций [4].

Как правило, в контрольных точках схемы известны номинальные значения напряжений или есть возможность произвести их с приемлемой точностью. Поэтому сравнив эти значения с реальными, полученными со схемы, можно оценить состояние электронного компонента. Однако такой подход к диагностике не всегда подходит, есть системы, выход из строя которых, может даже нанести вред человеку, поэтому особенно важно уметь спрогнозировать поломку заранее, используя полученные данные и сравнив их с данными, полученными в другой промежуток времени с этой же схемы.

Прогнозирование узлов, работающих по принципам механики, активно применяется уже достаточно долго. Но электронные системы сложнее поддаются анализу, потому что в основном меняют свои свойства не постепенно как в механике, а скачкообразно.

Надежность — это способность продукта или системы работать в соответствии с назначением (т. е. без сбоев и в установленных пределах производительности) в течение определенного времени в среде своего жизненного цикла. Диагностика работоспособности схемы и электрические показания, полученные до выхода элемента из строя, позволяют больше

говорить о характере поломки и дают дополнительные возможности в предотвращении повторных поломок на этом или аналогичном компоненте.

Так как современное электронное оборудование становится все более сложным, отказы в отдельных компонентах приводят к сложным отказам всей системы. Анализ отказа без учета этих факторов может привести к ошибочным корректирующим действиям.

Отказы компонентов происходят из-за механических, тепловых, экологических, тепловых, электрических, упаковочных факторов и факторов старения. Важно знать все подробности и причины.

Механическое разрушение печатной платы включает в себя упругую и пластическую деформацию, зарождение и распространение разрушения. Деформация — это просто искажение, которое может изменить форму и размер объекта. Она бывает двух видов: эластичная и пластичная. Эластичная деформация носит временный характер и исчезает после устранения внешних сил. Однако пластическая деформация является постоянной и сохраняет искажение даже после устранения внешних сил, вызывающих напряжение.

Термическое разрушение происходит, когда компонент нагревается выше его критических температур. Это может привести к возгоранию.

Также возникает ситуация, когда элемент цепи выходит из строя в результате пробоя. Одной из причин подобного поведения может служить неправильная эксплуатация

Реальные сигналы, такие как температура, давление, положение обычно обрабатываются аппаратной цепочкой сигналов. Такая цепь включает в себя аналоговую электронику, преобразование в цифровые данные и дальнейшие операции цифровой обработки. Обработка и анализ таких сигналов способствует предотвращению поломок.

В данной работе разработана система сбора, передачи и хранения напряжения и токов, протекающих в электронных схемах с целью дальнейшего анализа и определения работоспособности схемы.

1 Общие сведения об анализе и диагностике схем

Для анализа поломки или ее предотвращения за устройством устанавливается наблюдение. Состояние, когда устройство перестает выполнять свои функции, называется отказом. Отказы предполагают полную потерю функций и различные уровни деградации.

Анализ отказа — это исследование характера и механизма отказа с использованием оптических, электрических, физических и химических методик анализа. Прежде чем начать анализ, необходимо собрать подробности об обстоятельствах и симптомах отказа. Сюда входит исследование изменений электрических характеристик и других предшествующих отказу данных, среды, условий нагрузки, монтажа и возможность человеческих ошибок. Анализ этих факторов позволит сделать предположение о потенциальном характере и механизме отказа. На основе этого предположения определяются наиболее подходящие методы и процедуры. Недостаточная информация относительно обстоятельств и симптомов отказа может привести к неподобающему выбору методики анализа, а, следовательно, к значительным затратам труда и времени [10].

Напряжения и токи, полученные со схемы и переданные в базу данных, позволяют говорить не только о работоспособности схемы, но и дают много другой информации об устройстве и процессах, связанных с ним.

Например, кривые тока устройства важны для понимания инженерами, поскольку они графически показывают реакцию устройства на различные уровни перегрузки по току. Кривые позволяют инженеру энергосистемы графически представить селективную координацию устройств перегрузки по току в электрической системе. Также различные нелинейные элементы, который участвуют в построении электрических цепей, могут приводить к искажениям формы кривой тока и напряжения, тем самым делая ее форму отличной от синусоидальной. Такими элементами, которые искажают форму сигнала, могут быть как резисторы с нелинейным сопротивлением, так и

катушки индуктивности или конденсаторы. Сигналы искаженной формы находят практическое применение, например, в связи. На практике обычно все кривые токов и напряжений бывают в большей или меньшей степени отличны от синусоидальных. Периодическая несинусоидальная кривая может быть разложена в тригонометрический ряд Фурье и затем могут быть вычислены амплитуды и фазы гармоник. Для этого производится запись кривых тока и напряжения на фотопленке или бумаге осциллографом [8].

Графической зависимостью какого-либо процесса от времени называется осциллограмма и снимается она при помощи осциллографа — прибора, который графически отображает электрические сигналы и показывает, как эти сигналы изменяются во времени. Осциллографы используются для измерения электрических явлений и быстрого тестирования, проверки и отладки своих схем. Основная функция осциллографа заключается в измерении волн напряжения. Эти волны отображаются на графике, который может многое рассказать о сигнале, например:

— значения времени и напряжения сигнала;

— частота колебательного сигнала;

— «движущиеся части» цепи, представленные сигналом;

— частота, с которой возникает конкретная часть сигнала по отношению к другим частям;

— независимо от того, искажает ли неисправный компонент сигнал;

— какая часть сигнала представляет собой постоянный ток (DC) или переменный ток (AC);

— какая часть сигнала является шумом и меняется ли шум со временем [23].

2 Выбор инструментов

2.1 Плата микроконтроллера

Для снятия и передачи показаний необходимо собрать схему из датчиков и микроконтроллера, который будет принимать входные значения и подавать управляющие сигналы. Микроконтроллеры способны обнаруживать цифровые сигналы. Когда микроконтроллер питается от пяти вольт, он может принимать на вход ноль вольт (0 В) как двоичный 0 и пять вольт (5 В) как двоичную 1. Однако порой необходимо использовать какие-то другие значения. Эти значения называются аналоговыми сигналами. Многие микроконтроллеры имеют встроенное устройство, которое позволяет преобразовывать эти напряжения в значения, которые мы можем использовать в программе.

Как правило, аналого-цифровой преобразователь делает снимок аналогового напряжения в один момент времени и выдает цифровой выходной код, который представляет это аналоговое напряжение. Количество двоичных цифр или битов, используемых для представления этого аналогового значения напряжения, зависит от разрешения аналого-цифрового преобразователя.

Аналого-цифровое преобразование является операцией, устанавливающей соотношение двух величин: входной аналоговой Vi и эталонной Vr. Цифровой сигнал преобразователя есть кодовое выражение этого соотношения. Это соотношение иллюстрируется на рисунке 1. Если цифровой выход является n – разрядным, то число дискретных выходных уровней равно 2n. Таким образом, процесс аналого-цифрового преобразования является совокупностью квантования по уровню и кодирования.

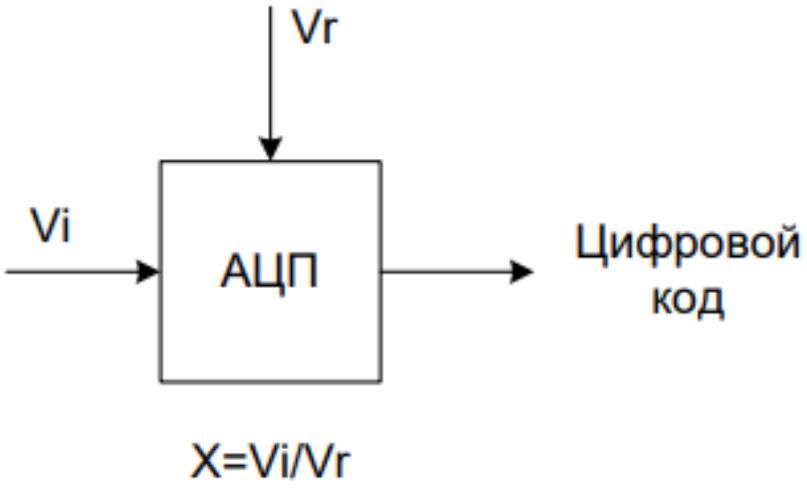


Рисунок 1 – АЦП

На рисунке 2 изображена передаточная характеристика АЦП (ломаная) и линия абсолютной точности (прямая, проходящая через 0 и точку максимального значения входного сигнала). Из этого рисунка видно, что шаг квантования по уровню, в сущности, определяет разрешающую способность преобразователя. Для квантованного сигнала характерно наличие скачков на величину шага квантования в моменты времени, когда непрерывный аналоговый входной сигнал проходит среднее между двумя уровнями значение. Между этими моментами времени значение выходного сигнала не изменяется.

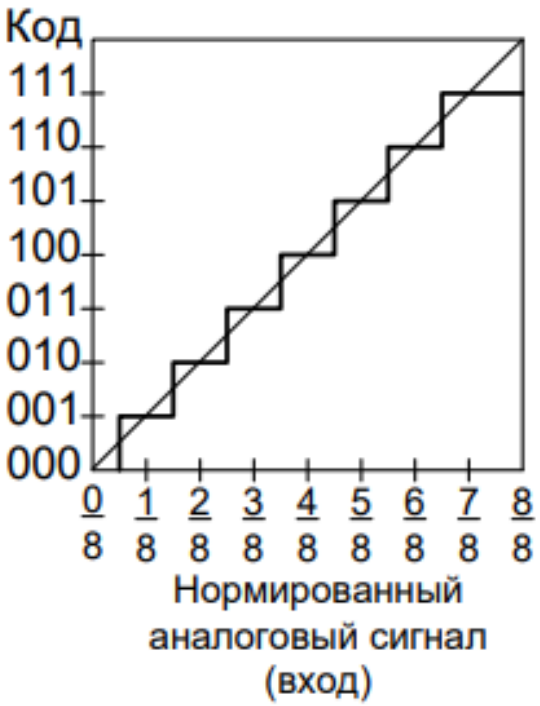


Рисунок 2 – Передаточная характеристика АЦП

Аналого-цифровой преобразователь (АЦП) - очень полезная функция, которая преобразует аналоговое напряжение на выводе в цифровое число.

Не каждый вывод микроконтроллера может выполнять аналого-цифровое преобразование. На плате Arduino эти контакты имеют перед меткой букву «A» (от A0 до A5), чтобы указать, что эти контакты могут считывать аналоговые напряжения.

АЦП могут сильно различаться между микроконтроллерами. АЦП на Arduino представляет собой 10-разрядный АЦП, что означает, что он может обнаруживать 1024 (2 ^ 10) дискретных аналоговых уровня. Некоторые микроконтроллеры имеют 8-битные АЦП (2 ^ 8 = 256 дискретных уровней), а некоторые - 16-битные АЦП (2 ^ 16 = 65 536 дискретных уровней).

Принцип работы АЦП сложен. Есть несколько разных способов добиться этого, но один из наиболее распространенных методов использует аналоговое напряжение для зарядки внутреннего конденсатора, а затем измерения времени, необходимого для разряда через внутренний резистор.

Микроконтроллер отслеживает количество тактов, которые проходят перед разрядкой конденсатора. Это количество циклов — это число, которое возвращается после завершения работы АЦП.

АЦП сообщает логометрическое значение. Это означает, что АЦП предполагает, что 5 В составляет 1023, а значение, меньшее 5 В, будет иметь отношение между 5 В и 1023 (формула 1).

$$\frac{R}{Umax} = \frac{X}{U} \qquad (1)$$

где R – Разрешение АЦП,

X – искомое значение,

Umax – максимальное напряжение,

U – измеренное напряжение.

Аналого-цифровое преобразование зависит от напряжения в системе. Поскольку мы преимущественно используем 10-битный АЦП Arduino в системе 5 В, мы можем немного упростить это уравнение (формула 2):

$$\frac{1023}{5} = \frac{X}{U} \qquad (2)$$

Если рассматриваемая система составляет 3,3 В, то просто необходимо заменить 5 В на 3,3 В в уравнении. Если аналоговое напряжение составляет 3 В, какое значение будет выдавать АЦП?

Решаем уравнение и получаем, что X=613,8

Чтобы понять, как это работает в реальной схеме воспользуемся Arduino для определения аналогового напряжения.

В плату Arduino встроен микроконроллер Atmega328p. Это высокопроизводительный 8-битный микроконтроллер, который сочетает в себе флэш-память ISP объемом 32 КБ с возможностью чтения во время записи, EEPROM 1024 Б, 2 КБ SRAM, 23 линии ввода-вывода общего назначения, 32 рабочих регистра общего назначения, три гибких таймера / счетчики с

режимами сравнения, внутренними и внешними прерываниями, последовательным программируемым USART, байтовым 2-проводным последовательным интерфейсом, последовательным портом SPI, 6-канальным 10-разрядным аналого-цифровым преобразователем (8 каналов в TQFP и QFN / MLF пакетов), программируемый таймер с внутренним генератором и пять программно выбираемых режимов энергосбережения. Устройство работает в диапазоне 1,8-5,5 вольт [13].

2.2 Программная среда

Для того чтобы запрограммировать микроконтроллер на считывание данных с датчиков, необходимо использовать среду разработки Arduino (рисунок 3). В этой среде используется язык программирования C++.

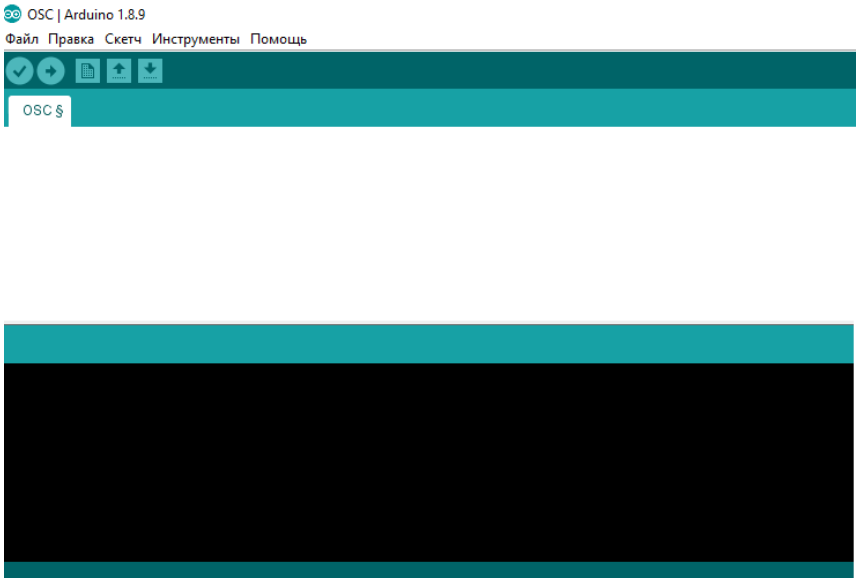


Рисунок 3 – Среда разработки Arduino

Данные выводились в плоттер последовательного соединения. Это позволило оценить их в реальном времени. Но при закрытии программы или отключении питания схемы все полученные данные терялись. То есть

проанализировать полученные данные можно только в режиме реального времени, что не всегда удобно, поэтому для решения данной проблемы было решено воспользоваться базами данных.

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

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. SQL был разработан в IBM в 1970-х годах. Со временем у стандарта SQL ANSI появились многочисленные расширения, разработанные такими компаниями как IBM, Oracle и Microsoft. Хотя в настоящее время SQL все еще широко используется, начали появляться новые языки программирования запросов [11].

В нашей работе будет использоваться реляционная база данных, то есть данные будут храниться в виде таблицы. Перед тем как начать писать запросы на языке SQL, нужно определиться с выбором СУБД, наиболее распространенные СУБД изображены на рисунке 4.

Рисунок 4 – Наиболее распространенные СУБД

Из рисунка видно, что сейчас существует большое количество современных СУБД. Так как в данном проекте данные из Arduino передаются при помощи языка программирования Python, то было решено использовать компактную встраиваемую Sqlite. В Python уже есть встроенная библиотека Sqlite3 с набором функций и методов, позволяющим создавать таблицы, добавлять данные и изменять их. Например, можно создать базу данных, наполнить ее пустыми таблицами, затем добавить туда записи. При помощи специальных команд можно извлекать из таблицы записи, соответствующие каким-то заданным нами условиям. Это все можно реализовать при помощи библиотеки sqlite. Еще одним ее преимуществом является то, что ее не нужно устанавливать и настраивать, можно сразу приступать к написанию программного кода. Поэтому все программное обеспечение было написано с использованием двух языков программирования: C++ и Python.

3 Сборка схемы и написание программного обеспечения для снятия напряжений и токов с элементов

3.1 Считывание напряжения с элементов схемы при помощи микроконтроллера Atmega328p.

Основываясь на способности микроконтроллера обнаруживать сигналы, было решено использовать его в качестве осциллографа. В результате анализа в качестве инструмента была выбрана плата Arduino с микроконтроллером ATmega328P (рисунок 5).

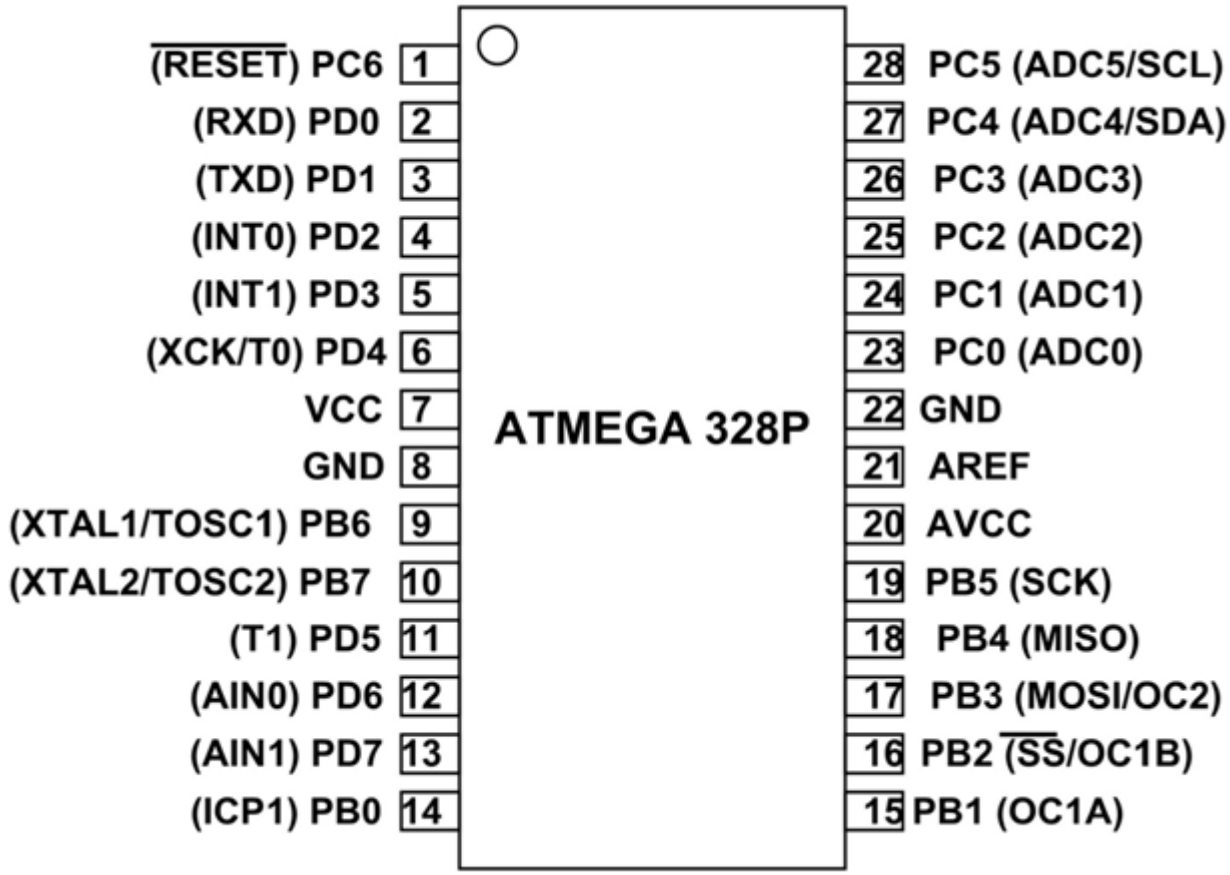


Рисунок 5 – Распиновка ATmega328P

ATmega328 — это высокопроизводительный контроллер с низким энергопотреблением. Он имеет модифицированное ядро 8-битного RISC-процессора с гарвардской архитектурой. Микроконтроллер Atmega328 используется в платах Arduino UNO, Arduino Pro Mini и Arduino Nano [16].

Осциллограф (рисунок 6) представляет собой электронный прибор, который графически отображает изменяющийся сигнал, обычно в виде двухмерного графика одного или нескольких сигналов в зависимости от времени. Осциллографы отображают изменение электрического сигнала с течением времени, с напряжением и временем по осям Y и X, соответственно, на калиброванной шкале. Затем форму волны можно проанализировать на предмет таких свойств, как амплитуда, частота, время нарастания, временной интервал, искажение и другие. Современные цифровые инструменты могут вычислять и отображать эти свойства напрямую. Первоначально расчет этих значений требовал ручного измерения формы волны по шкале, встроенной в экран прибора.

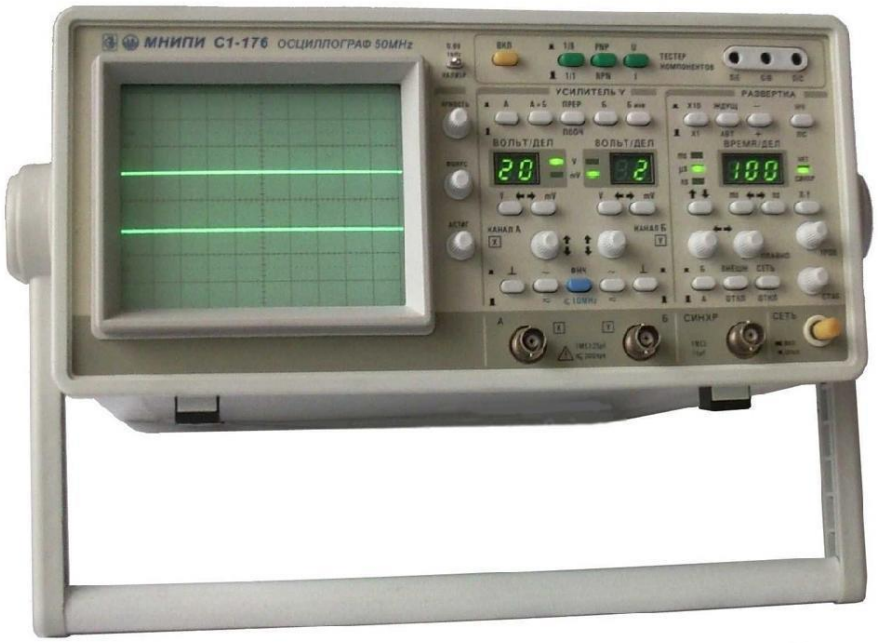


Рисунок 6 – Осциллограф

Но лучше всего в разработке идти от простого к сложному, поэтому первым делом необходимо протестировать плату на возможность работы в режиме вольтметра и считать изменяющееся напряжение. Для изменения напряжения можно использовать подстроечный резистор, датчик освещенности или простой делитель напряжения.

В этих целях была симулирована схема с потенциометром в системе моделирования Tinkercad (рисунок 7).

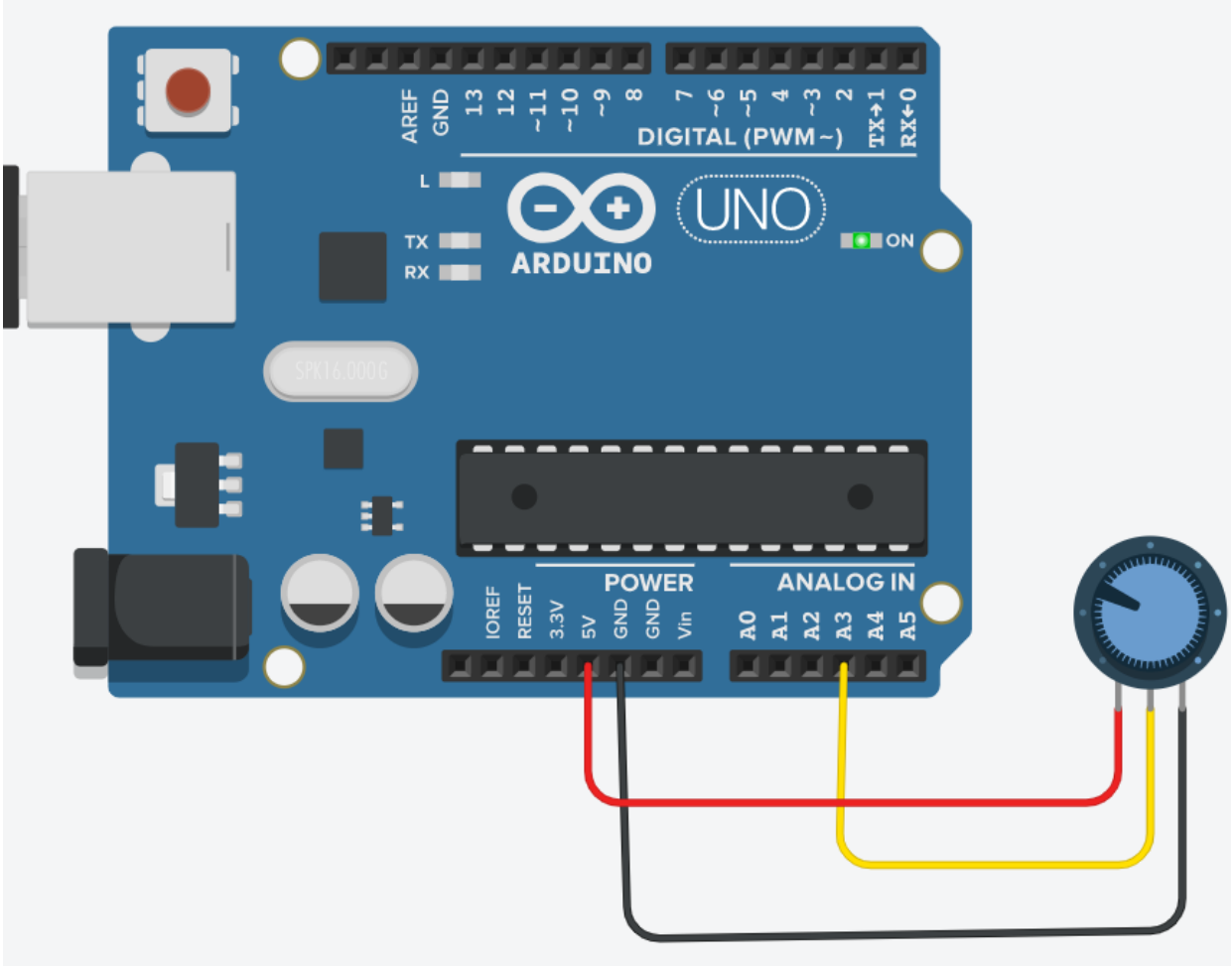


Рисунок 7 – Смоделированная схема в Tinkercad

Перед написанием кода была составлена блок-схема программы, представленная на рисунке 8.

Рисунок 8 – Блок-схема программы

Для начала нам нужно определить контакт как вход. Будем использовать контакт A3:

pinMode(A3, INPUT);

В затем выполните преобразование аналоговой версии в цифровую с помощью команды analogRead() :

int x = analogRead(A3); // Считывает напряжение с контакта A3

Значение, которое возвращается и сохраняется в x, будет значением от 0 до 1023. Arduino имеет 10-битный АЦП (2 ^ 10 = 1024). Мы сохраняем это значение в int, потому что x больше (10 бит), чем может содержать байт (8 бит).

Напечатаем это значение, чтобы посмотреть, как оно меняется:

Serial.print(“Analog value: “);

Serial.println(x);

Таким образом, можно использовать плату микроконтроллера Arduino как вольтметр.

Для работы осциллографа на плате микроконтроллера Arduino необходимо написать программу. Но для начала были проанализированы готовые решения.

3.2 Считывание силы тока со схемы

Чтобы сделать выводы по поводу работы схемы недостаточно знать лишь напряжение, нужно оперировать как можно большим количеством величин, поэтому было собрана схема для определения силы тока элементов.

Для этих целей был использован датчик ACS712 (рисунок 9). Это полностью интегрированный линейный датчик тока на основе эффекта Холла с изоляцией по напряжению 2,1 кВэфф и встроенным токопроводом с низким сопротивлением. Помимо технических терминов, это просто датчик тока, который использует свой проводник для расчета и измерения величины приложенного тока.

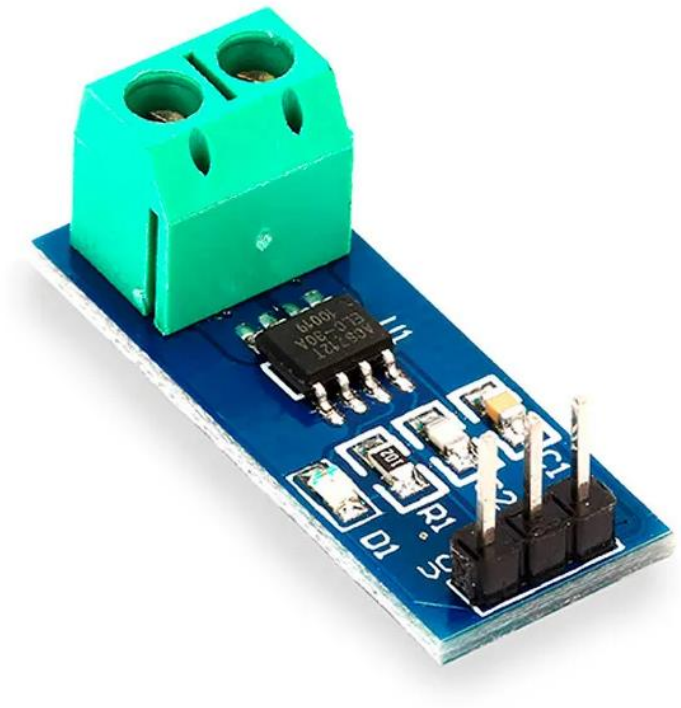


Рисунок 9 – Датчик тока ACS712

Особенности ACS712:

— полоса пропускания 80 кГц;

— выходная чувствительность от 66 до 185 мВ/А;

— путь аналогового сигнала с низким уровнем шума;

— полоса пропускания устройства устанавливается с помощью нового вывода FILTER;

— внутреннее сопротивление проводника 1,2 мОм;

— общая погрешность вывода 1,5% при TA = 25°C;

— стабильное выходное напряжение смещения;

— Почти нулевой магнитный гистерезис.

Расположение выводов датчика ACS712 и схема включения представлена на рисунке 10 [3].

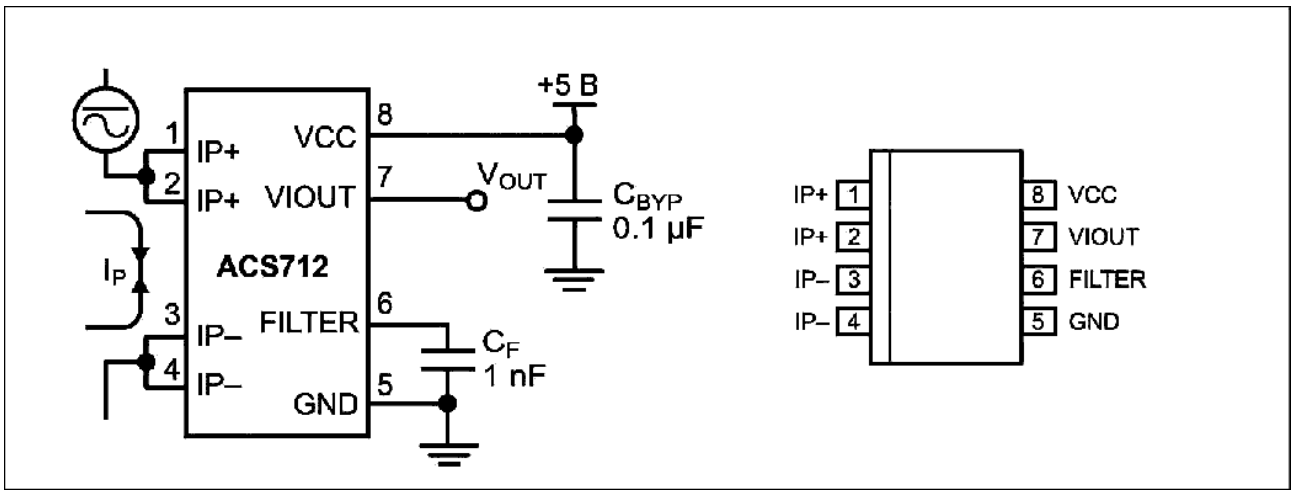


Рисунок 10 – Расположение выводов датчика ACS712 и схема включения

Принцип работы датчика заключается в следующем:

— ток протекает по цепи бортового датчика холла в его ИС;

— датчик Холла обнаруживает входящий ток за счет генерации магнитного поля;

— после обнаружения датчик Холла генерирует напряжение, пропорциональное его магнитному полю, которое затем используется для измерения силы тока.

Для подключения датчика тока к Arduino используется три контакта: A0, 5V, GND (рисунок 11).

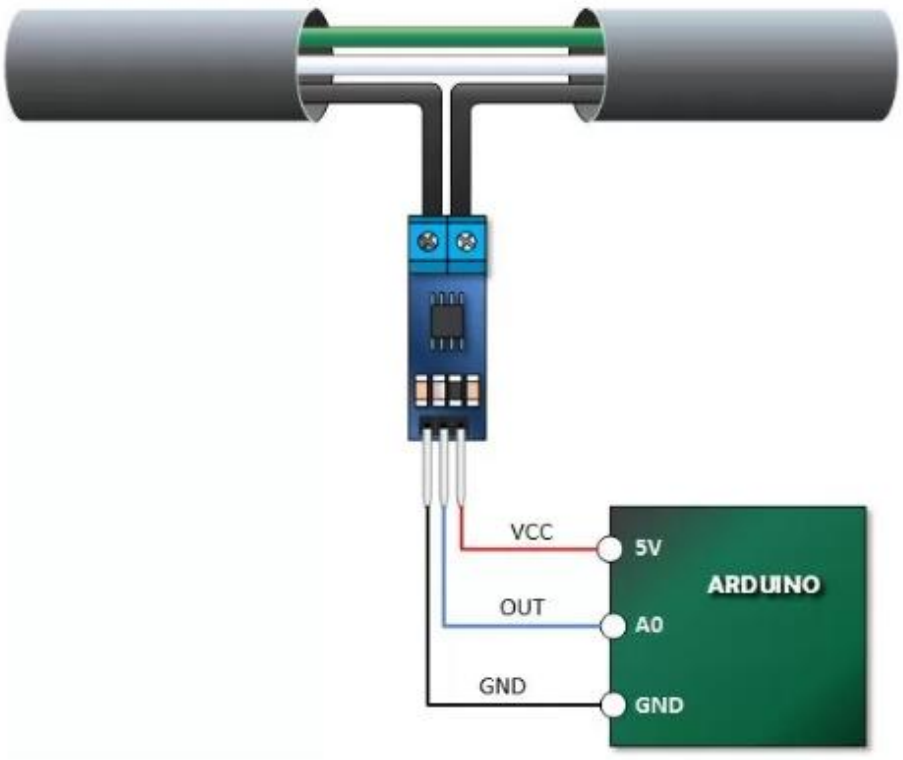


Рисунок 11 – Подключение датчика тока ACS712 к Arduino

Блок-схема программы изображена на рисунке 12 [7].

Рисунок 12 – Блок-схема программы для работы датчика тока

Для начала нужно назначить контакт OUT датчика тока:

#define PIN_OUT A0

Затем произвести подключение библиотеки:

#include <TroykaCurrent.h>

Потом нужно создать объект для работы датчика:

ACS712 dataI(PIN_OUT);

Затем запустить последовательный порт:

Serial.begin(9600);

И, наконец, в void loop считывать и выводить показания в последовательный порт:

Serial.print(dataI.readCurrentDC());

Аналоговый контакт нужно подключить к выводу исследуемого элемента, например, к аноду светодиода.

3.3 Анализ готовых решений

Большинство имеющихся решений схожи тем, что имеют код, написанный на языке C++, а также использующие программное обеспечение для вывода сигнала на экран компьютера.

Для того чтобы реализовать простейший осциллограф понадобятся:

— Arduino Leonardo или Arduino Micro,

— два зажима типа крокодил,

— конденсатор 0.1 мкФ (опционально),

— стабилитрон 5.1В (опционально),

— ПК с установленным компилятором языка Processing.

На рисунке 13 можно видеть схему осциллографа на Arduino с делителями напряжения.

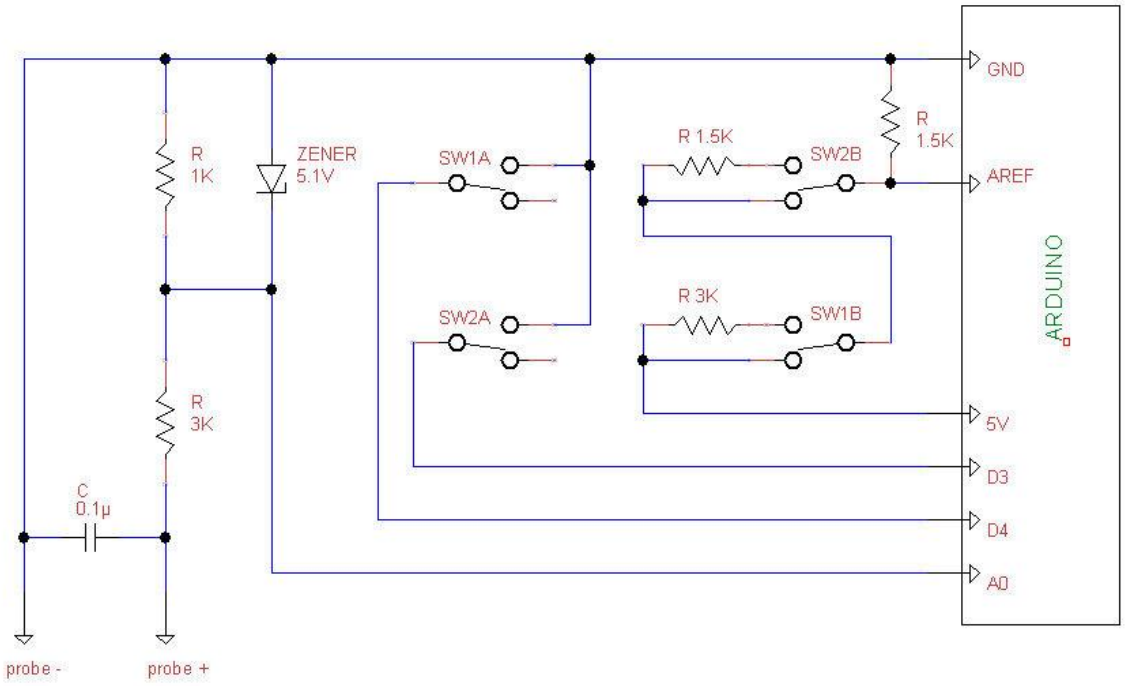


Рисунок 13 – схема осциллографа на Arduino с делителями напряжения

В левой стороне схемы расположен делитель напряжения с коэффициентом 1:4. То есть к нему можно подключать напряжение да 20 В.

Справа расположен делитель напряжения, переключающийся между линиями 5 В и опорного напряжения (Aref). Вы можете использовать переключатели для установки диапазона измерения: 5В, 6.64В, 10В или 20В. Второй контакт каждого переключателя соединяется с цифровым входом Arduino (D3 и D4). Это работает следующим образом. Если программа настроена на работу с опорным напряжением, АЦП сравнивает напряжение аналоговых входов с Aref вместо 5В. Например, мы измеряем 5 В, тогда напряжение на A1 будет 5В/4=1.25 В. Если оба переключателя разомкнуты, то напряжение на Aref будет 5В, АЦП прочитает 1.25/5=25%. Если первый переключатель разомкнут, а второй замкнут, напряжение на Aref будет 2.5В, АЦП прочитает 1.25/2.5=50%. Если первый переключатель замкнут, а второй разомкнут, напряжение на Aref будет 1.66В, АЦП прочитает 1.25/1.66=75%.

Если оба переключателя замкнуты, напряжение на Aref будет 1.25В, АЦП прочитает 1.25/1.25=100% [2].

Пример результата сборки схемы представлен на рисунке 14:

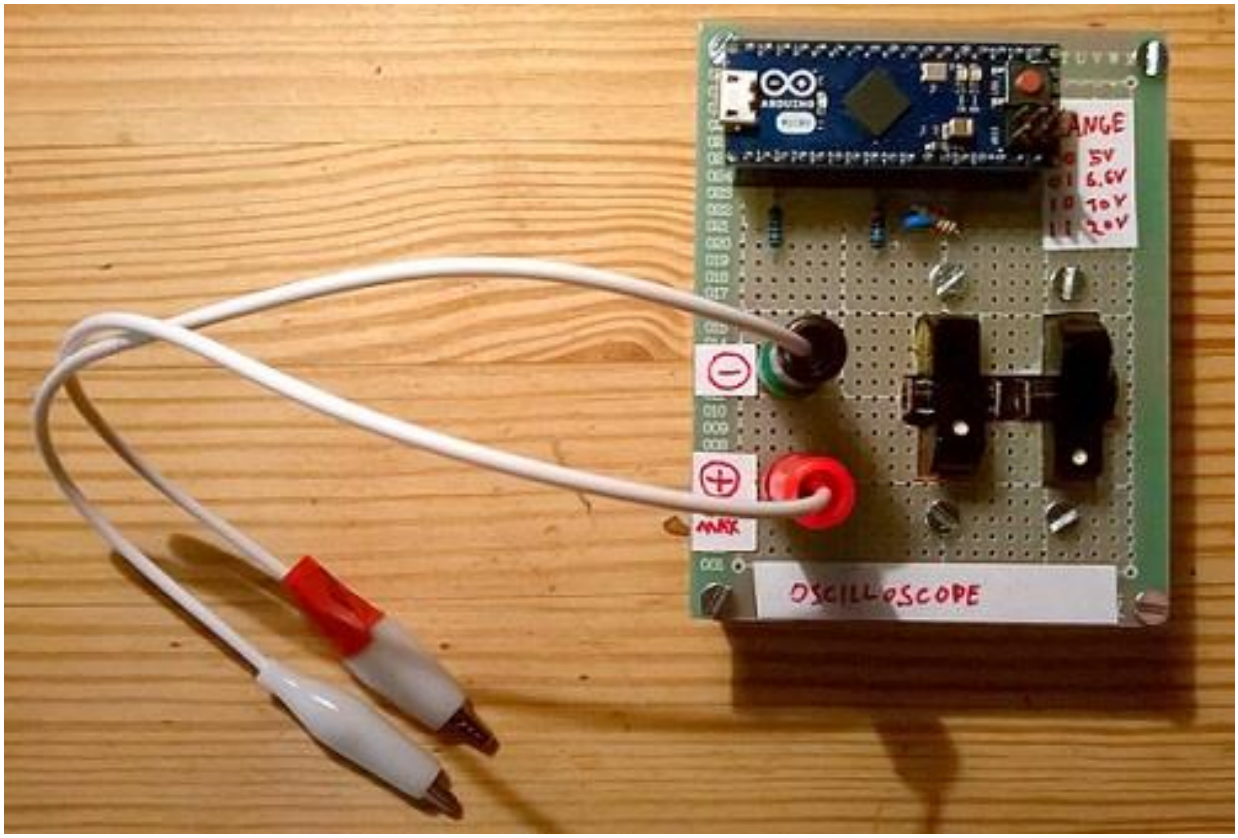


Рисунок 14 – осциллограф на Arduino

Дополнительными элементами в схеме являются конденсатор между линией щупа и землей и Стабилитрон 5.1V. Конденсатор позволяет уменьшить шумы измеряемых сигналов, а стабилитрон защищает Arduino от перенапряжения [9].

В результате выполнения кода на экране компьютера отобразится сигнал в формате, представленном на рисунке 15.

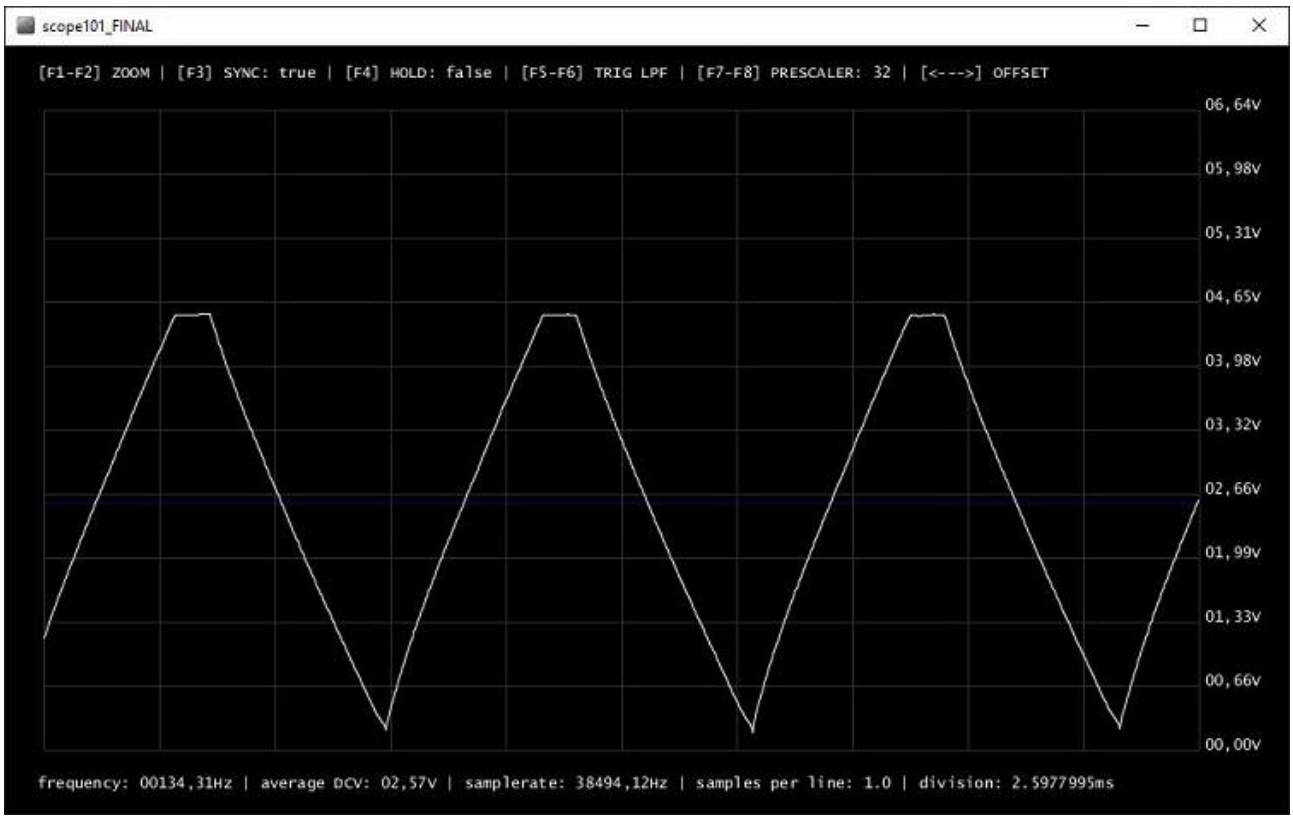


Рисунок 15 – Пример вывода сигнала на экран

Другое решение отличается простым интерфейсом и хорошим набором настроек (рисунок 16).

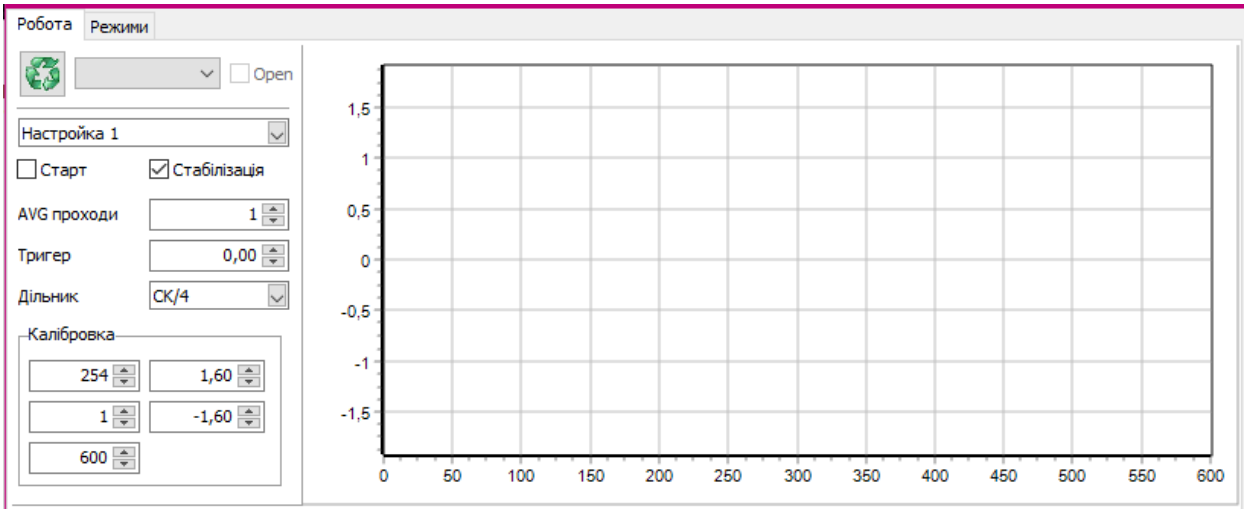


Рисунок 16 – Интерфейс программы для осциллографа

На рисунке 17 представлена схема сборки такого осциллографа.

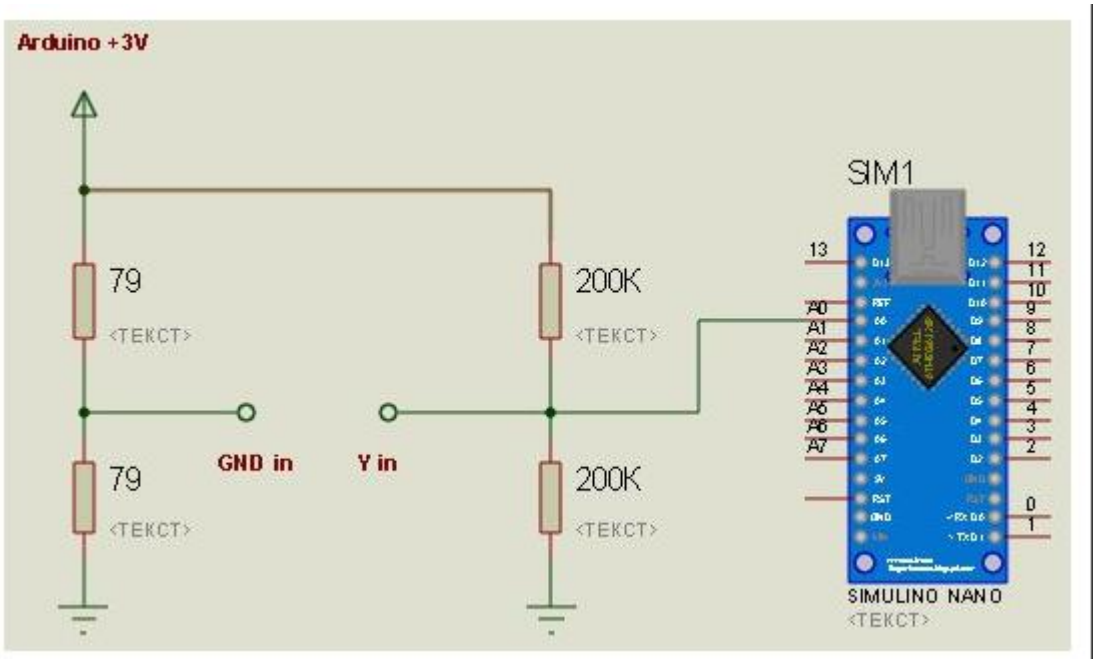


Рисунок 17 – Схема подключения микроконтроллера

Другой способ создания осциллографа на Arduino отличается тем, что вывод данных производится на отдельный дисплей (рисунок 18). Для работы понадобится:

— Arduino,

— дисплей,

— резисторы,

— кнопки.

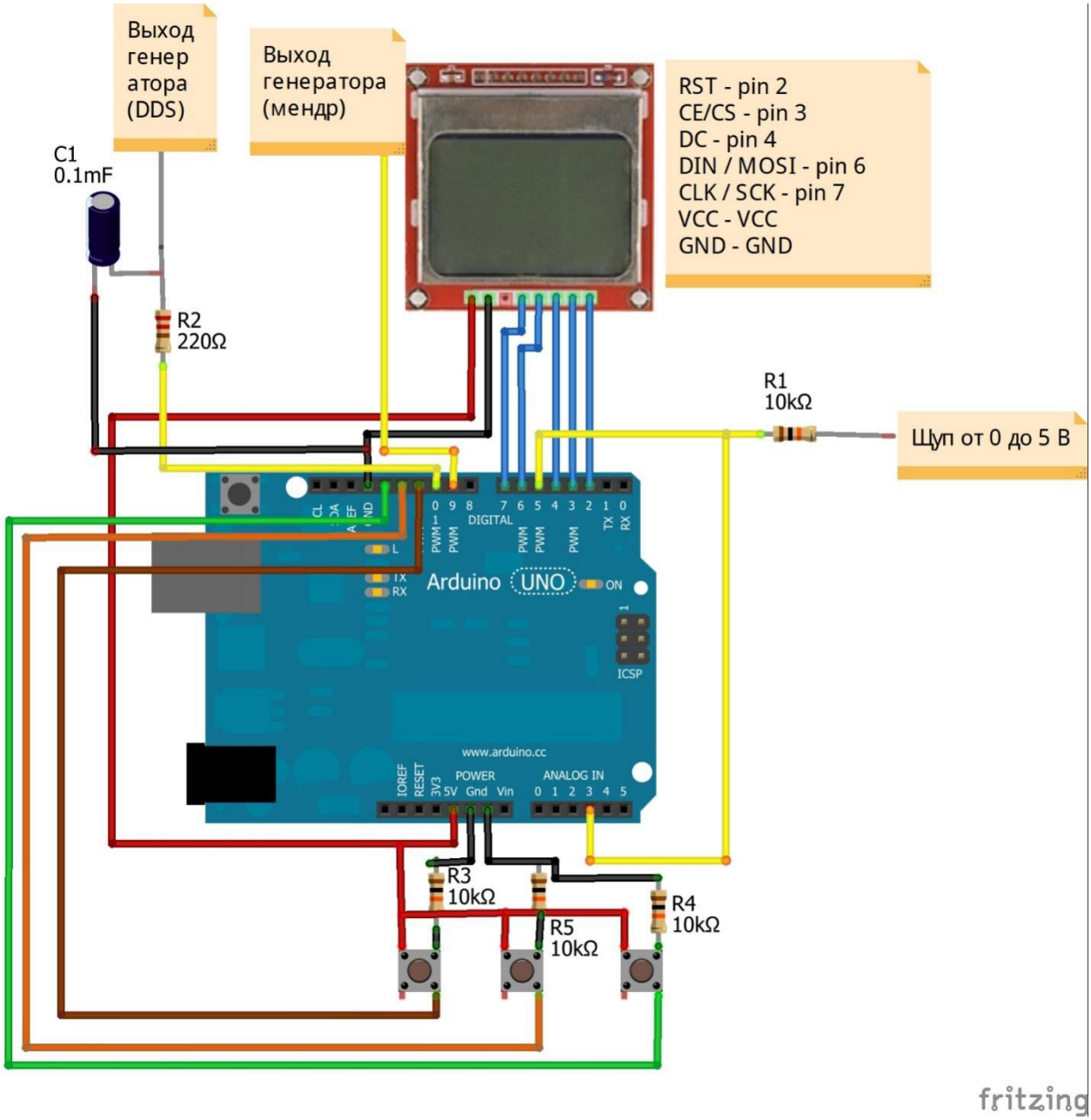


Рисунок 18 – Схема сборки осциллографа

Таким образом, преимуществом этого проекта является вывод данных из компьютера на отдельный экран, что позволяет улучшить портативность проекта.

3.4 Разработка программы для работы платы в режиме осциллографа

В результате поиска и анализа готовых решений было решено взять одну из программ и модифицировать ее.

Программный код имитирует 6-лучевой осциллограф. Ожидается, что входные сигналы, представленные на аналоговых выводах, будут находиться в диапазоне от 0 до 5 вольт с максимальной частотой 1 кГц.

Можно выбрать количество входов 1-6, которые отображают напряжения на выводах A0-A5 соответственно. Незадействованные открытые аналоговые выводы выдают ложные напряжения.

Осциллограф работает в двух режимах: «непрерывный» (свободная работа) и «принудительный запуск» («ждущий», запуск развертки при выполнении критерия). Критерий (триггер) срабатывания выполняется, когда входной сигнал, считанный с A0, пересекает предопределенное напряжение запуска, в зависимости от того, будет ли это «фронт» или «спад», когда он пересекает это предопределенное напряжение. Это определяется при помощи условных операторов if и else [6].

В «ждущем» режиме общее время развертки может быть установлено в миллисекундах. Начало запуска развертки указывается, когда синхронизирующая отметка доскакивает до уровня 5 В.

При развертке выборка на аналоговых выводах будет производиться каждые 'SampleInterval' миллисекунд:

В нижней части графика синхронизирующие отметки (прямоугольный сигнал) будут переключаться каждый 10-ые 'SampleInterval' миллисекунд.

Встроенный светодиод (вывод 13) является индикатором состояния осциллографа: (1) включен, непрерывный режим или развертка в «ждущем» режиме, (2) мигает, запущен «ждущий» режим, ожидает срабатывания триггера, (3) выключен, все операции приостановленны (с помощью кнопки).

Когда происходит выборка более одного сигнала, отображения сигналов могут «накладываться» или «разделяться по каналам». Когда используются каналы, напряжения на вертикальной оси не откалиброваны.

Опционально между цифровым выводом 12 и землей может быть подключена кнопка. При ее нажатии цифровая выборка и развертка останавливаются. Повторное нажатие кнопки возобновляет развертку (но с разрывом в графиках сигналов).

Порядок и цвета выводимых графиков следующие: метки времени (синий), уровень запуска (красный, если включен ждущий режим), аналоговые сигналы A0-A5, соответственно (многоцветные) (рисунок 19) [15].

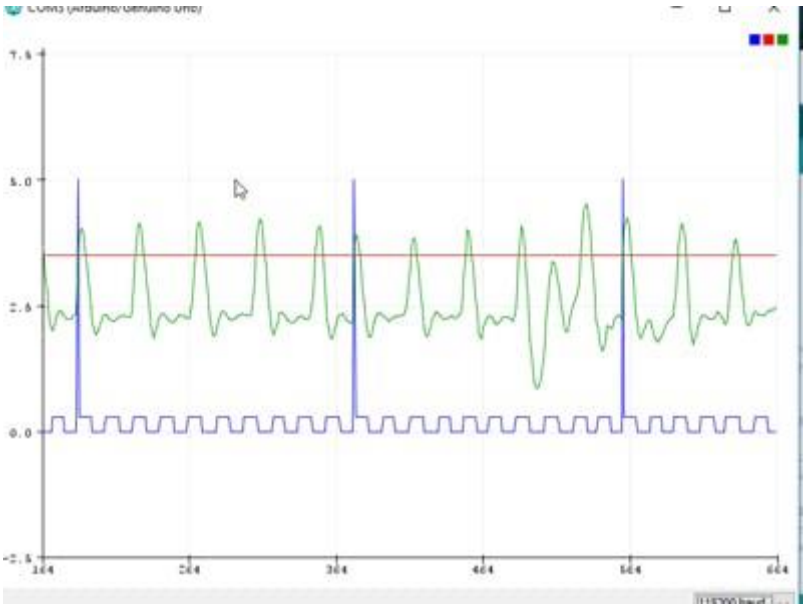


Рисунок 19 – Порядок и цвета выводимых графиков

В начале программы нужно определить переменные и константы.

На языке C++ создание констант производится с помощью выражения “#define”. Переменные же создаются при помощи указания их типа. Переменная – поименованная, либо адресуемая иным способом область памяти, адрес которой можно использовать для осуществления доступа к

данным. В C++ они бывают несколько типов, главные из которых представлены в таблице 1.

Таблица 1 – Типы переменных.

Тип	Определение	Значение
Int	Целое число	-32 768 / 32 767
Long	Целое число	-2 147 483 648 / 2 147 483 647
Boolean	Логический	true/false
Float	Число с плавающей точкой	-2 147 483 648.0 / 2 147 483 647.0

Определяем буквенное сокращение ul для переменной типа unsigned long.

#define ul unsigned long

Затем определяем состояние запуска и кнопки.

#define armed true

Указываем режим развертки(непрерывный).

#define continuous true

Задаем переменные для запуска по спаду или по фронту. Указываем режим развертки, определяющий, когда происходит срабатывание. Настраиваем прочие параметры осциллографа.

Затем нужно определить прерывание. Оно задает функцию, которую необходимо вызвать при возникновении внешнего прерывания или заменяет предыдущую функцию, если таковая была ранее ассоциирована с прерыванием. В большинстве плат Arduino существует два внешних прерывания: номер 0 (цифровой вывод 2) и 1 (цифровой вывод 3).

Синтаксис написания для платы Arduino выглядит так:

attachInterrupt(interrupt, function, mode)

Где interrupt – номер прерывания, а function – функция, которую необходимо вызвать при возникновении прерывания; функция должна быть без параметров и не возвращать никаких значений. Эту функцию иногда называют обработчиком прерывания. Mode определяет условие, при котором должно срабатывать прерывание. Принимает одно из четырех предопределенных значений:

— LOW – будет срабатывать всякий раз при низком уровне сигнала;

— CHANGE – при изменении состояния вывода прерывание будет срабатывать всякий раз;

— RISING – это прерывание, которое будет работать, если состояние вывода изменится с низкого уровня на более высокий;

— FALLING – это прерывание, которое будет работать, если состояние вывода изменится с низкого уровня на высокий.

Структура программы в среде Arduino обладает особенностью, есть часть кода, выполняющаяся один раз, и соответственно прописывается в функции “setup”, есть часть кода, которая выполняется, постоянно повторяясь и записывается в функции loop.

Настройку прерывания, а именно время генерации прерывания, частота выдачи импульсов, стоит производить один раз. А выполнение выборки и отображение сигнала, временных меток и напряжения запуска, изменения состояния индикатора нужно производить постоянно.

3.5 Тестирование работы осциллографа

Для того, чтобы протестировать схему на работоспособность было решено собрать схему мультивибратора.

На рисунке 20 представлена схема симметричного мультивибратора.

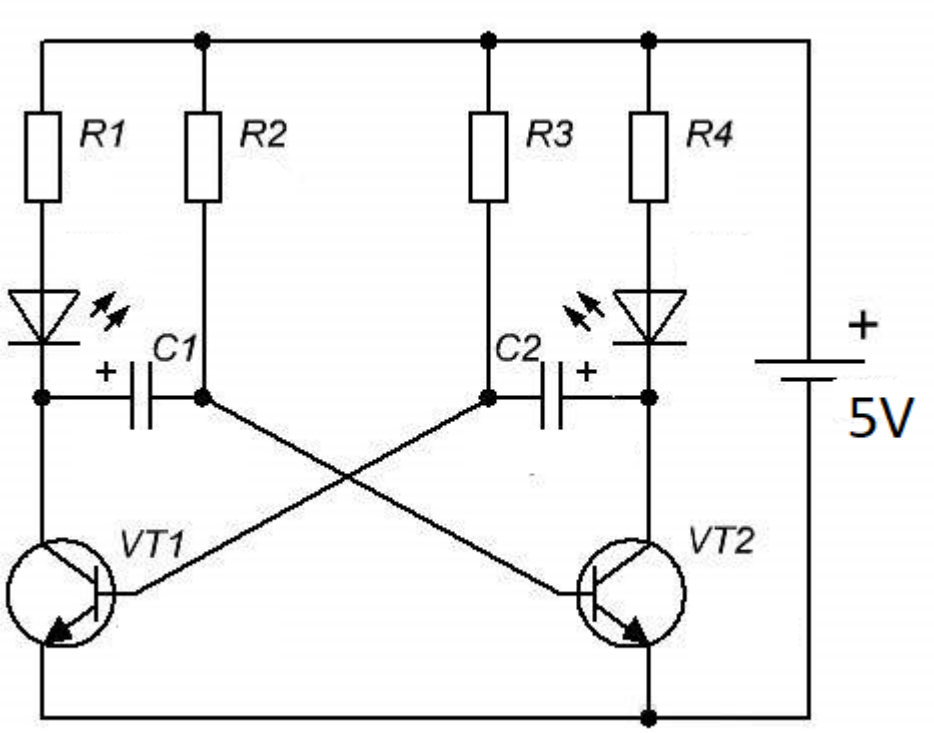


Рисунок 20 – Схема мультивибратора

Мультивибратор – это релаксационный генератор сигналов электрических прямоугольных колебаний с короткими фронтами. Из всех генераторов импульсов прямоугольной формы, наиболее распространенным является мультивибратор. Он представляет собой двухкаскадный резистивный усилитель с глубокой положительной обратной связью.

На двух транзисторах или логической схеме собираются мультивибратор и дополнительные элементы. Фактически это двухкаскадный усилитель, в котором цепь положительной обратной связи (ПОС) является двухкаскадной. В этом случае выход второго каскада связан через конденсатор со входом первого каскад. В результате усилитель за счет положительной обратной связи превращается в генератор. Для того чтобы мультивибратор стал генерировать импульсы, необходимо подключить напряжение питания.

Мною была собрана схема мультивибратора (рисунок 21). К аноду светодиода был подключен провод, идущий в аналоговый контакт Arduino.

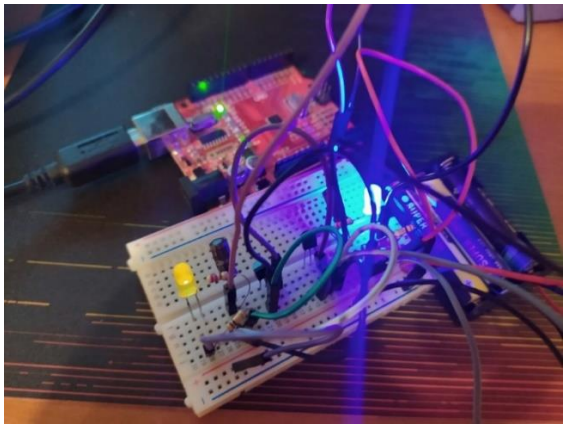


Рисунок 21 – Экспериментальная схема мультивибратора.

В результате выполнения кода и загрузки его в плату Arduino в плоттере последовательного соединения отобразилась осциллограмма и была сохранена как png файл. Она представлена на рисунке 22.

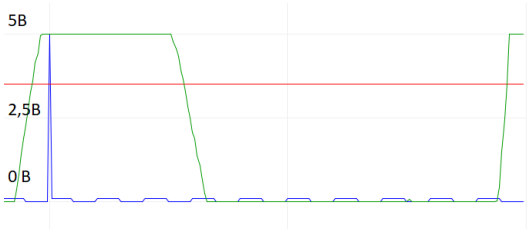


Рисунок 22 – Осциллограмма в плоттере последовательного соединения.

4 Передача и хранение полученных со схемы данных

4.1 Разработка программы для передачи данных на компьютер

Так как результаты выполнения кода выводятся не только в плоттер, но и в монитор порта в виде численных значений, можно передавать их на компьютер. Для этого мною была написана программа на языке Python, которая представлена на рисунке 23.

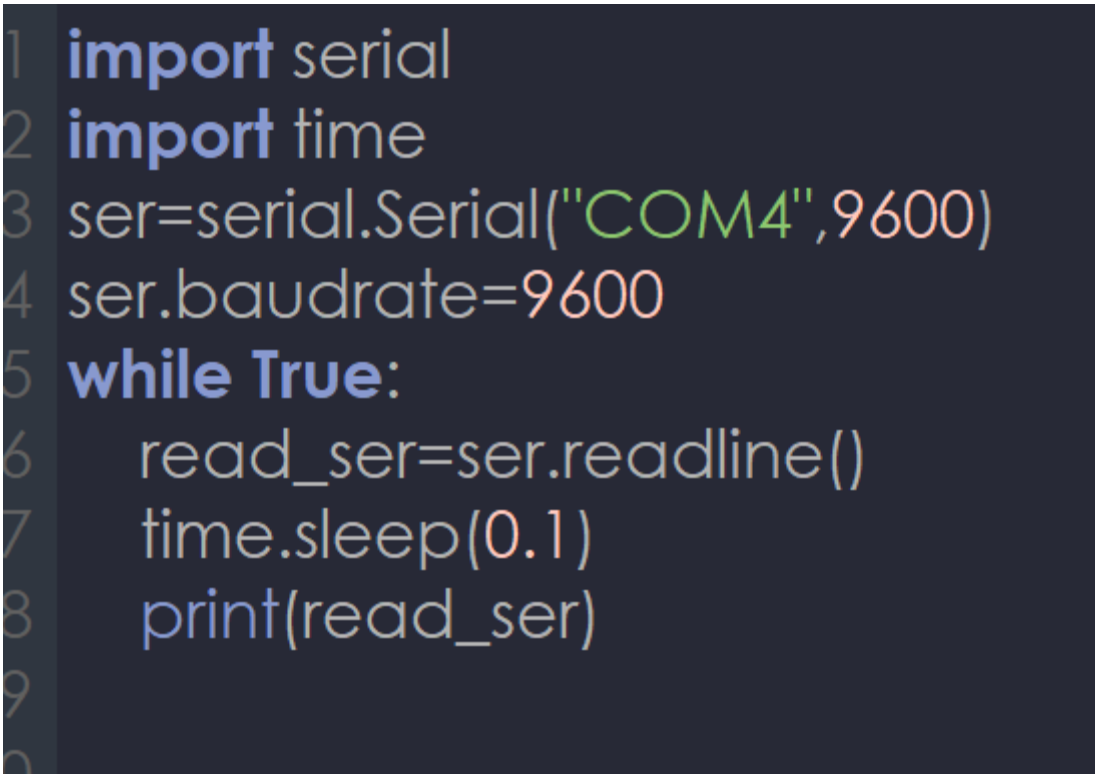


Рисунок 23 – Код программы.

Сначала импортируются необходимые библиотеки для работы с последовательным портом и временными задержками. Затем выбирается порт для работы и скорость передачи данных. После этого в бесконечном цикле, создаваемом с помощью конструкции while, происходит считывание данных с небольшой временной задержкой. В дальнейшем с помощью инструментов

языка можно обрабатывать эти данные, сортировать, группировать и экспортировать их.

4.2 Автоматизация запуска программного обеспечения

Для получения данных с электронной схемы, последующей передачи, хранения и анализа, необходимо сначала установить пакет программного обеспечения: интерпретатор Python, среда разработки Arduino IDE. Затем нужно запустить код, который будет считывать данные и сохранять их в базу данных.

Проблема заключается в том, что за считывание данных отвечает среда Arduino IDE с графическим интерфейсом, а за сохранение и обработку – программа, написанная на языке Python. На рисунке 24 изображен пакет файлов, необходимых для запуска осциллографа, считывания и сохранения значений в базу данных.

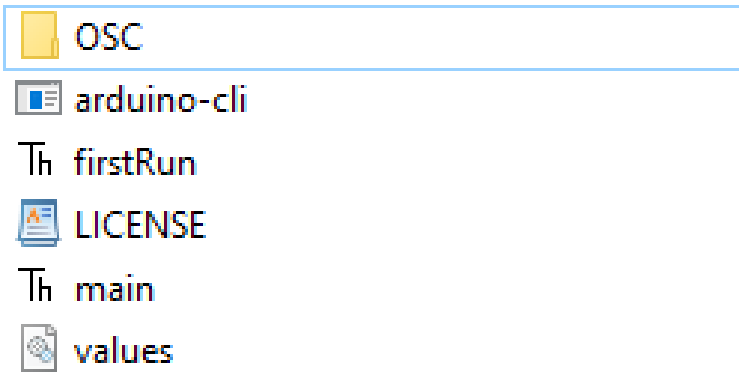


Рисунок 24 – Пакет файлов

Запуск кода производится в два этапа, на что уходит неоправданно много времени. На рисунке 25 показан графический интерфейс среды Arduino.

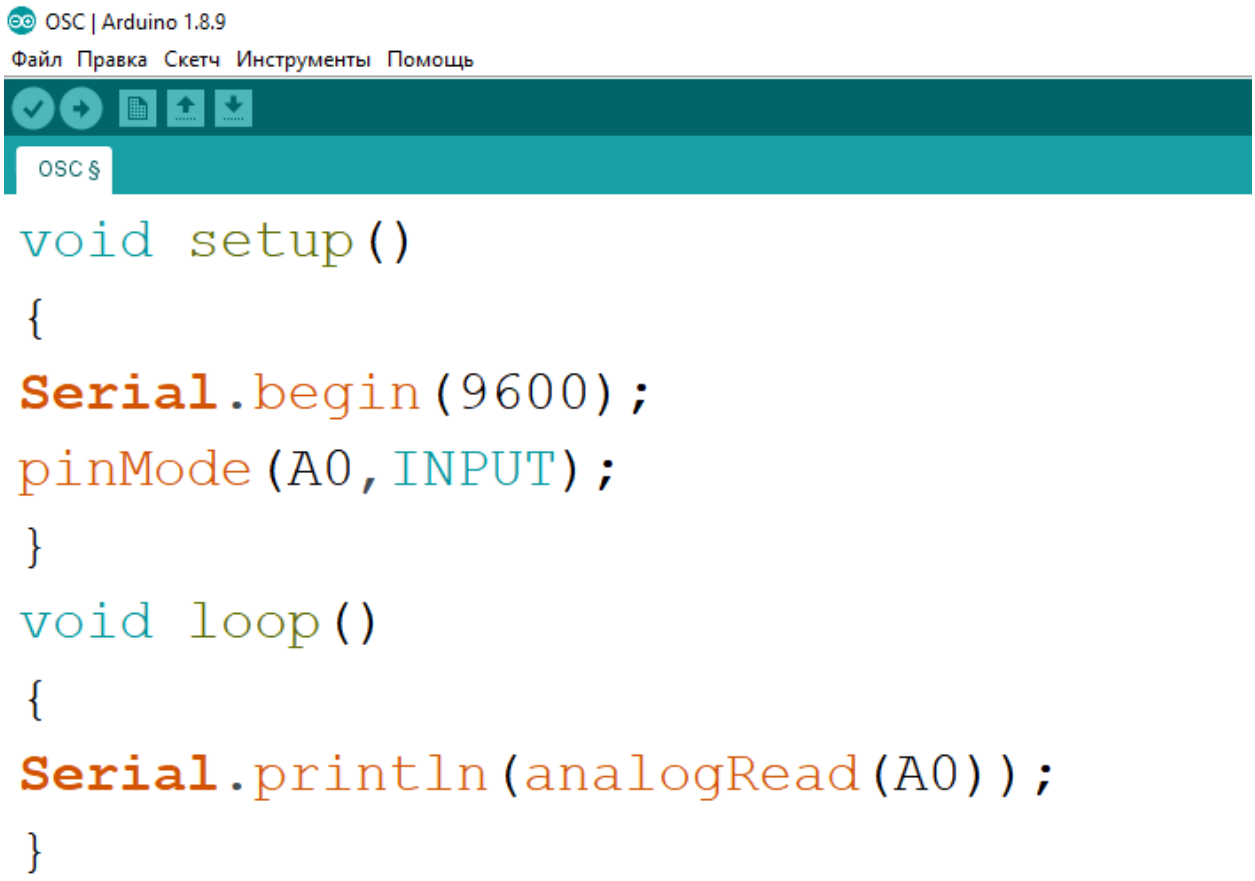


Рисунок 25 – Интерфейс среды Arduino IDE

Чтобы уменьшить количество действий от пользователя можно автоматизировать первый этап. Один из способ сделать это – воспользоваться модулем Pyautogui на Python.

PyAutoGUI позволяет вашим сценариям Python управлять мышью и клавиатурой для автоматизации взаимодействия с другими приложениями. API разработан, чтобы быть простым. PyAutoGUI работает в Windows, macOS и Linux, а также – на Python версии 2 и 3 [22].

Мы будем пользоваться Python 3, так как он более современный и обладает лучшей поддержкой разработчиков и сообщества пользователей.

PyAutoGUI имеет несколько функций:

— перемещение мыши и нажатие в окнах других приложений;

— отправка нажатий клавиш приложениям (например, для заполнения форм);

— создание скриншотов и изображений (например, кнопки или флажка) и поиск его на экране;

— поиск окна, изменение размера, перемещение (в настоящее время только для Windows);

— отображение окон предупреждений и сообщений.

То есть, этот модуль позволит указать координаты нужных кнопок из среды Arduino.

Но сначала нужно научиться открывать программу, изображенную на рисунке 2, не вручную, а при помощи Python. Можно воспользоваться модулем os и написать следующий метод:

os.startfile('OSC\OSC.ino')

Этот метод открывает указанную в качестве аргумента программу.

Теперь перейдем к загрузке программы в микроконтроллер. Она производится в несколько этапов.

1. Выбрать в инструментах нужный последовательный порт
2. Загрузить порт

Для выполнения этой задачи первым делом нужно создать новый python файл и импортировать модуль pyautogui, предварительно установив его:

import pyautogui

Затем нужно определить координаты нужных кнопок и написать алгоритм нужной последовательности кликов. Клик производится методом:

pyautogui.click(координаты клика)

А определить нужную кнопку для клика можно с помощью метода locateOnScreen, который находит нужный объект по фото и возвращает его координаты.

Итоговый код представлен на рисунке 26.

```
os.startfile('OSC\OSC.ino')
z =p.locateOnScreen('but.png')

while z == None:
    z =p.locateOnScreen('but.png')
x=p.center(z)
p.click(x)
time.sleep(8)
```

Рисунок 26 – Программа для загрузки скетча в плату Arduino

Недостатком этого метода является то, что на всех компьютерах разрешение экрана разное, поэтому изображение нужной кнопки может искажаться. Поэтому программа может дать сбой. Чтобы избавиться от зависимости от разрешения экрана, нужно отказаться от графического интерфейса Arduino IDE. Для этого существует специальный инструмент Arduino CLI. Он предназначен для загрузки кода в микроконтроллер через командную строку. По сути, это инструмент командной строки, который содержит все необходимое для простого создания приложений в экосистеме Arduino [14].

Чтобы работать с этим инструментом, нужно скачать его на компьютер и поместить в папку с нашим Python файлом. Этот инструмент позволяет загружать в плату Arduino программу, не используя графический интерфейс. Для этого нужно запустить три команды для установки драйверов и загрузки скетча (рисунок 27).

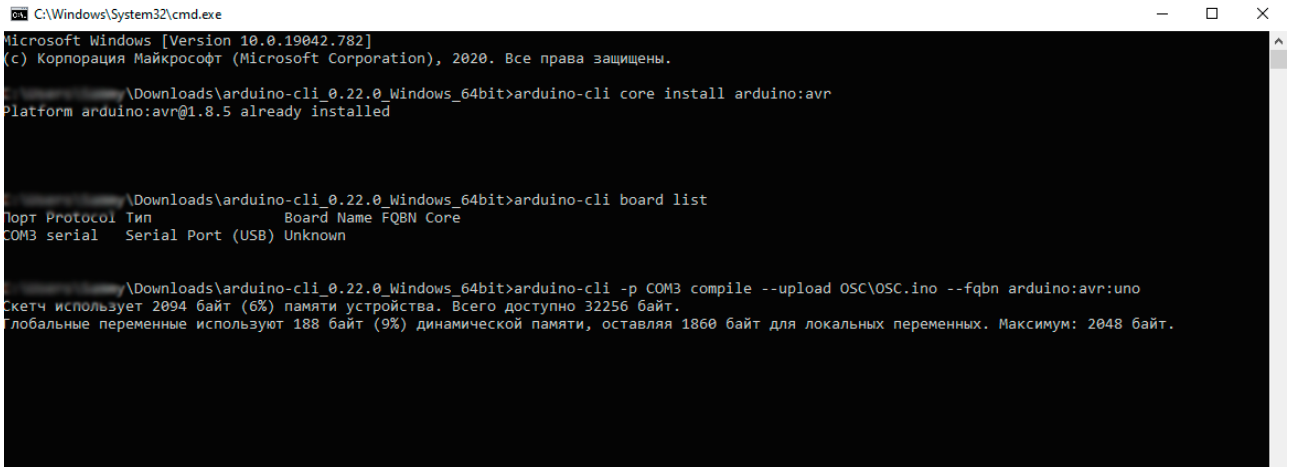


Рисунок 27 – Командная строка

После выполнения первой команды можно увидеть сообщение, что драйвера уже установлены или увидеть сообщение об установке драйверов в данный момент. Далее нужно узнать последовательный порт, к которому подключен Arduino. Последовательный порт или COM-порт двунаправленный последовательный интерфейс, предназначенный для обмена байтовой информацией.

Последовательный потому, что информация через него передаётся по одному биту, бит за битом (в отличие от параллельного порта). Наиболее часто для последовательного порта персональных компьютеров используется стандарт RS-232C.

Ранее последовательный порт использовался для подключения терминала, позже для модема или мыши. Сейчас он используется для соединения с источниками бесперебойного питания, для связи с аппаратными средствами разработки встраиваемых вычислительных систем [12].

Поэтому сегодня при помощи микросхемы FTDI COM-порт создается виртуально при подключении платы к компьютеру через USB.

Эти команды можно выполнить также, написав программу на Python. Для этого можно обратиться к библиотекам os и subprocess. Os – модуль для работы с операционной системы, который уже был использован выше, когда

нужно было запустить Arduino файл. Subprocess же – это модуль, используемый для запуска новых кодов и приложений путем создания новых процессов. Он позволяет запускать новые с помощью программы написанной на Python. Чтобы запустить новый процесс или, другими словами, новый подпроцесс в Python, нужно использовать вызов функции Popen. В вызове функции можно передать два параметра. Первый параметр — это программа, которую мы хотим запустить, а второй — аргумент файла [5].

Сначала нужно импортировать библиотеку:

```
import subprocess
```

Затем – по очереди вызвать нужные команды при помощи функции Popen:

```
subprocess.Popen("arduino-cli core install arduino:avr", shell=True, stdout=subprocess.PIPE)
```

Но чтобы вводить команды автоматически за пользователя нужно избавиться от всех изменяемых компонентов. Один из таких компонентов – выбор последовательного порта. В моем случае COM равен 3. На других компьютерах это значение может быть другим. Поэтому нужно при помощи функций Python анализировать вывод командной строки и определять, какой COM там указан.

Для этого нужно написать следующий код:

```
subprocess_return = str(subprocess_do.stdout.read())
com_first_num = str(subprocess_return[str(subprocess_return).find('COM') + 3])
com_second_num = str(subprocess_return[str(subprocess_return).find('COM') + 4])
```

Он определит весь текст, полученный из командной строки, и найдет 2 следующих символа, идущих за словом «COM». Это нужно для того, чтобы учесть тот случай, когда значение последовательного порта – двузначное. После этого нужно сохранить это значение в переменную и передать его в функцию system для отправки в командную строку:

```
COM = "COM" + COM
system(f'"
arduino-cli -p {COM} compile --upload {path} --fqbn arduino:avr:uno"')
```

Так, удалось автоматизировать процесс загрузки программы и сохранения значений в базу данных.

4.3 Сортировка и группировка значений

После того как данные были переданы на компьютер, необходимо было научиться хранить и обрабатывать данные при помощи Python и SQL и произвести сортировку и группировку произвольных значений в базе данных.

Для создания базы данных первым делом нужно создать отдельную python программу и подключить библиотеки:

```
import sqlite3, datetime
```

Первая библиотека позволяет работать с базами данных, а вторая – в удобном виде получить текущую дату и время, что позволит сортировать данные по датам их добавления.

Затем необходимо создать базу данных с именем values.db:

```
conn = sqlite3.connect('values.db')
```

Если база данных уже была создана нами ранее, то есть мы повторно запустили программу, то база данных не будет создаваться заново, а просто произойдет подключение к существующей БД. В любом случае результат выполнения функции сохранится в переменную conn. С ее помощью теперь мы создадим курсор:

```
cur = conn.cursor()
```

Курсор — это временная память или временная рабочая станция. Он выделяется сервером базы данных во время выполнения пользователем операций над таблицей. Курсоры используются для хранения таблиц базы данных [24].

Затем нужно использовать метод execute, который выполняет определенный SQL запрос, который необходимо написать в круглых скобках и в кавычках. В этом запросе мы создаем таблицу с названием values_table и добавляем в нее три поля – id, volt, time, date.

В конце необходимо закрыть соединение с базой данных и сохранить все изменения:

```
conn.commit()
```

Таким образом, в базе данных была создана таблица для хранения id – номера записи, volt – напряжение, которое будет передаваться с осциллографа, time – момент времени, исчисляемый с начала передачи данных и date – сегодняшняя дата.

Для того чтобы добавлять в таблицу значения нужно сначала разобраться, какие типы данных можно туда передавать. При создании полей id, volt, time, date я указал типы данных – int, int, int и date соответственно. То есть в эту таблицу можно передавать целые числа и дату. Id – это уникальный номер поля, он создается сам и автоматически увеличивается на 1, его можно не передавать.

Чтобы добавить первое значение в таблицу нужно создать две переменные, в первой передать SQL запрос на добавление данных, а во второй – сами данные:

```
sql = "INSERT INTO values_table (volt,time,date) VALUES (?, ?,?)"
val = (5,0, datetime.date.today())
```

Затем с помощью уже известного метода execute передать эти переменные в базу данных.

```
cur.execute(sql, val)
```

После этого необходимо сделать коммит изменений и проверить, получилось ли добавить данные. Для этого нужно написать:

```
cur.execute("select * from values_table")
print(cur.fetchall())
```

Первая строчка запрашивает все данные из таблицы, а вторая выводит все на экран.

Для лучшего понимания, как располагаются данные в созданной таблице, можно посмотреть рисунок 28.

ID	VOLT	TIME	DATE
1	1	0	10-11-2021
...	...	...	...
...	...	...	...
999	999	999	10-11-2021

Рисунок 28 – База данных

Теперь для примера нужно добавить какие-то значения уже описанным выше способом. После этого можно начать работать с этими данными, например, выводить на экран данные, полученные только в определенный день или все значения напряжения, которые превышали какой-то порог, например, 4.5 вольта.

Для этого нужно понимать структуру запроса на языке SQL, которая представлена на рисунках 29 - 30.

Рисунок 29 – SQL запрос без условия

На рисунке 29 видно, что в качестве значения может выступать знак умножения, что означает «все поля».

Рисунок 30 – SQL запрос с условием

Помимо этого, можно получать не все данные, а лишь подходящие под какое-то условие, в данном случае под условие – напряжение, записанное в поле volt, должно быть больше 5. В результате этого запроса будут получены id, время, дата и значение напряжение, но только для тех случаев, когда напряжение подходило под это условие. На рисунке 31 показано, как получить только поле напряжение, которое к тому же должно подходить по условию volt>5.

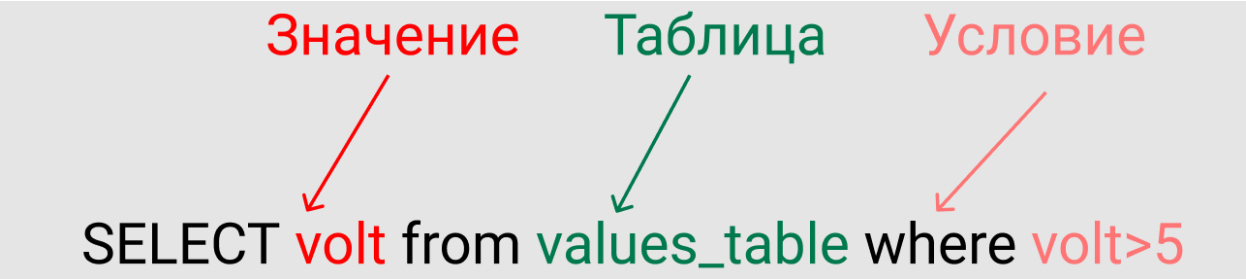


Рисунок 31– SQL запрос с условием и получением одного поля

С полученными значениями можно производить любые арифметические операции как с помощью SQL, так и при помощи Python. Например, можно перемножить или поделить между собой любые значения или вычесть из одной даты другую, чтобы найти разницу в количестве дней между ними. Все зависит от целей и задач.

Однако выполнять какие-то операции удобнее с теми данными, которые отсортированы и сгруппированы каким-то образом, а не выданы в хаотичном порядке. Стандартно все значения будут выводиться в том порядке, в котором были добавлены в таблицу. А если нужно посмотреть на значения по убыванию или по возрастанию? Здесь понадобится оператор order by.

Чтобы получить все значения напряжения, отсортированные по возрастанию, необходимо написать следующий запрос:

select volt from values_table order by volt asc

Точно также можно получить значения по убыванию, поменяв одно слово:

select volt from values_table order by volt desc

Также иногда необходимо получить группировку данных по какому-то признаку. Например, сгруппировать данные по дате и определить максимальное значение напряжения, полученное в этот день. Для этого нужно написать следующий запрос:

select max(volt), date from values_table group by date order by volt desc

Функция max вычисляет максимальное значение в данном столбце, однако далее происходит группировка данных по дате, поэтому максимальное значение вычисляется не по всему столбцу, а отдельно по дням.

4.4 Разработка интерфейса для хранения данных

Чтобы сохранить эти данные в конкретном месте и иметь возможность в любой момент обратиться к ним, нужно загрузить получившийся файл базы данных в какую-то систему с графическим интерфейсом понятным любому, даже не самому опытному, пользователю персонального компьютера.

Для этих целей подойдет любая известная библиотека для создания графического интерфейса на Python:

— Tkinter,

— WxPython,

— PyQT.

Tkinter — это библиотека для создания графических пользовательских интерфейсов (GUI) в Python, который включен во все стандартные дистрибутивы Python. Это единственный фреймворк, встроенный в стандартную библиотеку Python [31].

Этот фреймворк предоставляет пользователям Python простой способ создания элементов графического интерфейса с помощью виджетов из набора инструментов Tk. Виджеты Tk можно использовать для создания кнопок, меню, полей данных и т. д. в приложении Python. После создания эти графические элементы могут быть связаны или взаимодействовать с функциями, функциями, методами, данными или даже другими виджетами.

Например, виджет-кнопка может принимать щелчки мышью, а также может быть запрограммирован на выполнение какого-либо действия, например на выход из приложения.

На рисунке 32 изображен пример программного кода и результата его выполнения.

```
import tkinter as tk

def callback(event):
    root.title('x={0:2d}, y ={1:2d}'.format(event.x, event.y))

root = tk.Tk()
bg_image = tk.PhotoImage(file ="bg.gif")
lbl = tk.Label (root, image = bg_image)
lbl.pack()
lbl.bind("<Button-1>", callback)
root.mainloop()
```

Рисунок 32 – Пример работы библиотеки Tkinter

Другая популярная библиотека – WxPython — это кроссплатформенный инструментарий с графическим интерфейсом для языка программирования Python (рисунок 33). Он позволяет создавать программы с надежным, многофункциональным графическим пользовательским интерфейсом. Он реализован в виде набора модулей расширения Python, которые обертывают компоненты графического интерфейса популярной кроссплатформенной библиотеки wxWidgets, написанной на C++. При этом одна и та же программа будет работать на разных платформах, без изменений. По состоянию на данный момент поддерживаются платформы Microsoft Windows, Mac OS X и macOS, а также Linux или другие Unix-подобные системы с библиотеками GTK2 или gtk3. Обычно собственные виджеты используются на каждой платформе, чтобы обеспечить 100% естественный внешний вид [19].

Рисунок 33– Пример работы библиотеки WxPython

И, наконец, PyQt – библиотека для создания кроссплатформенных интерфейсов, которая широко используется для создания крупномасштабных программ на основе графического интерфейса (рисунок 34). Она также предоставляет множество хороших готовых дизайнов [30].

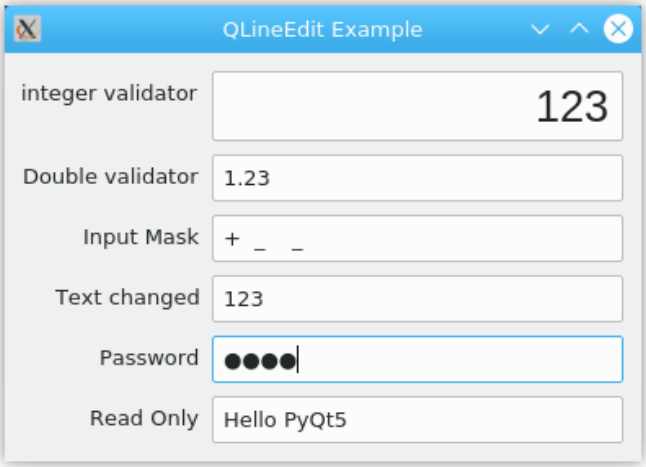


Рисунок 34 – Пример работы библиотеки PyQt

Однако если мы хотим передавать данные между компьютерами, не используя сторонних сервисов, то, в таком случае, нужно публиковать приложение на удаленном сервере. И тогда удобнее всего воспользоваться не

перечисленными выше библиотеками, а общепринятым языком разметки HTML для создания веб-сайтов.

HTML или язык гипертекстовой разметки — это язык разметки для Интернета, который определяет структуру веб-страниц [27].

Гипертекст - текст (часто также с вставками, такими как изображения), который организован для соединения связанных элементов.

Разметка - руководство по стилю для набора всего, что будет напечатано в печатном или электронном формате.

Язык - язык, который компьютерная система понимает и использует для интерпретации команд.

Как и любой язык, HTML выглядит как набор команд и текстовых блоков, прежде чем превратиться во фронтальную визуализацию. Пример html кода представлен на рисунке 35.

Рисунок 35– Пример HTML кода

Большая часть HTML построена с использованием «блоков элементов», которые представляют собой фрагменты HTML-кода, разделяющие различные элементы на странице.

Пример типичной структуры HTML страницы представлен на рисунке 36.

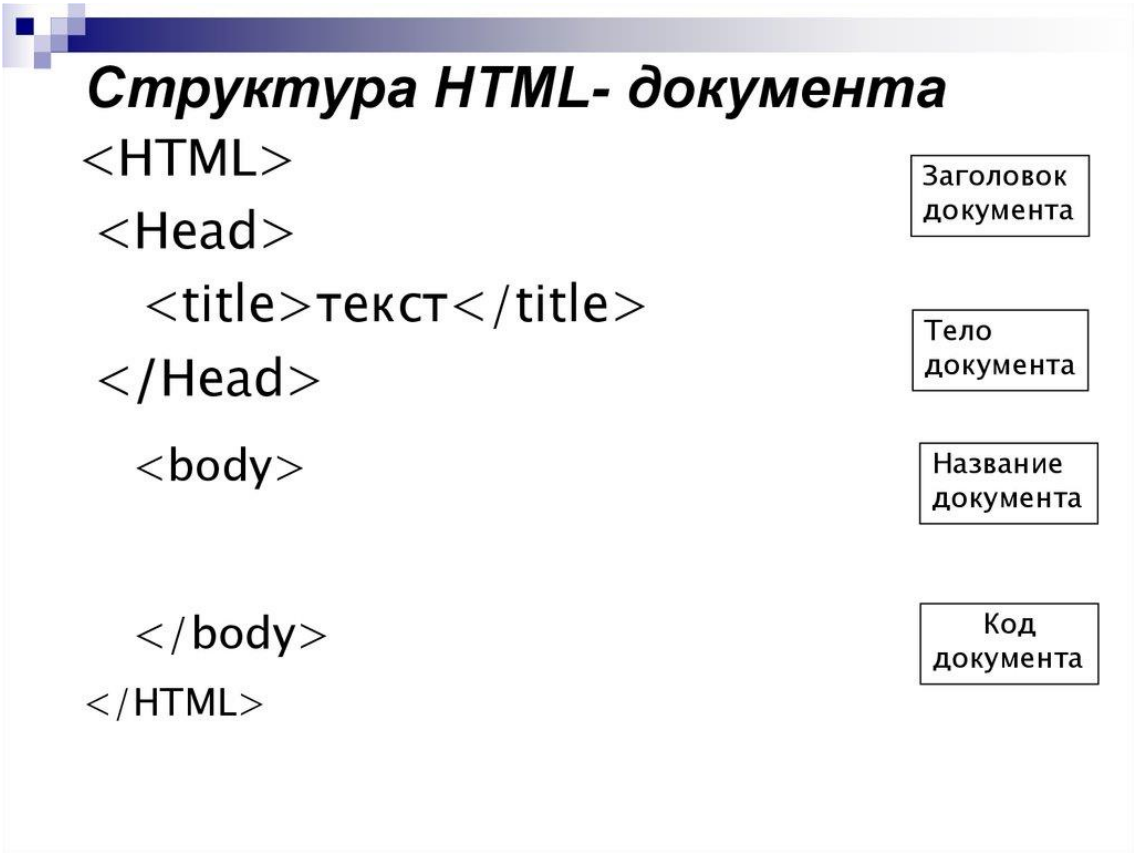


Рисунок 36 – Структура HTML страницы

Элемент состоит из открывающего тега, символа, содержимого и закрывающего тега (рисунок 37). Некоторые элементы пусты, то есть у них нет закрывающего тега, а вместо этого есть источник или ссылка на контент, встраиваемый в веб-страницу.

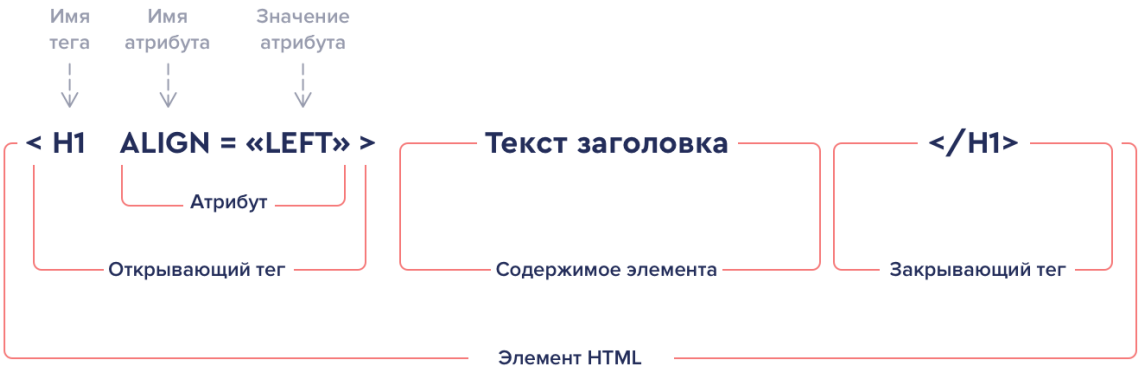


Рисунок 37 – HTML элемент

Важнейшей частью создания HTML-страниц является использование каскадных таблиц стилей (CSS). Это документы, которые определяют, как должны выглядеть конкретные элементы страницы. Например, насколько большими должны быть изображения, какие шрифты должны отображаться на странице и как веб-страница должна реагировать на изменение размера или растяжение. Все это имеет решающее значение для создания привлекательных, целостных и стильных веб-сайтов.

Смысл и предназначение CSS:

— CSS расшифровывается как каскадные таблицы стилей;

— CSS описывает, как HTML-элементы должны отображаться на экране, бумаге или других носителях;

— CSS экономит много работы. Он может одновременно управлять макетом нескольких веб-страниц;

— внешние таблицы стилей хранятся в файлах CSS.

Таким образом, с помощью CSS можно добавлять стили, то есть позиционировать элементы на странице, менять цвет и размер элементов и т.д. Пример страницы с использованием CSS и без его использования представлен на рисунке 38.

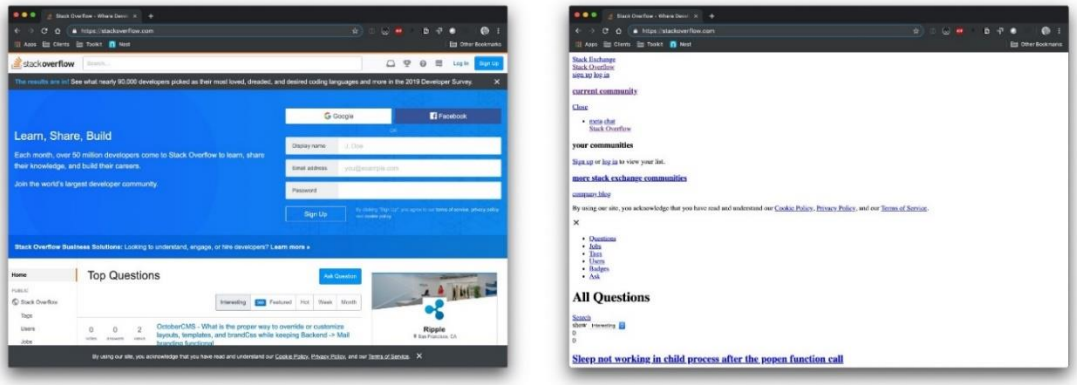


Рисунок 38 – Пример страницы с использованием CSS и без него.

Описанные выше технологии лишь позволяют сделать видимую оболочку. Чтобы добавить сайту необходимый нам функционал (загрузку считанных данных, вывод изображений графиков на странице, поиск загруженных ранее данных), необходимо написать программу серверной части. Чтобы не увеличивать количество используемых языков, воспользуемся библиотеками языка Python. Здесь есть две наиболее популярные библиотеки: Django и Flask.

Django — это бесплатный фреймворк с открытым исходным кодом, впервые публично выпущенный в 2005 году. Django обеспечивает «быструю разработку и чистый, прагматичный дизайн». Веб-инфраструктура Django, развернутая на веб-сервере, может помочь разработчикам быстро создать многофункциональный, безопасный и масштабируемый веб-интерфейс.

Использование веб-фреймворка Django — более эффективный способ создания веб-приложения, чем запуск с нуля, который требует создания серверной части, API, javascript и карт сайта. С помощью веб-фреймворка Django веб-разработчики могут сосредоточиться на создании уникального приложения и воспользоваться большей гибкостью, чем при использовании инструмента веб-разработки [25].

Веб-фреймворк Django используется уже более десяти лет и был тщательно протестирован и улучшен очень активным сообществом. Самая большая сила Django — это его большой набор функций — с более чем 10 000 пакетов Django, платформа охватывает практически все, что вам нужно для веб-приложения.

Другим известным веб-фреймворком является Flask. Это модуль Python, который позволяет легко разрабатывать веб-приложения. У него есть много интересных функций, таких как маршрутизация URL-адресов, механизм шаблонов. Это фреймворк для веб-приложений WSGI [26].

В отличие от фреймворка Django, Flask проще. Начать работу с Flask легко, потому что у него нет длительного обучения.

Кроме того, читабельность кода во Flask выше чем в Django.

На рисунке 39 представлен код простейшего приложения на Flask

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

Рисунок 39 – Простое приложение на Flask

Таким образом, можно выбрать Flask, так как данный проект не обладает таким большим масштабом и количеством одновременных подключений от пользователей.

Таким образом, разработка сайта велась при помощи технологий html, css и модуля Python - Flask.

Для начала нужно создать несколько Python и html файлов, а также папки для хранения определенных файлов. Итоговое содержимое папки со всеми файлами необходимыми для функционирования сайта изображены на рисунке 40.

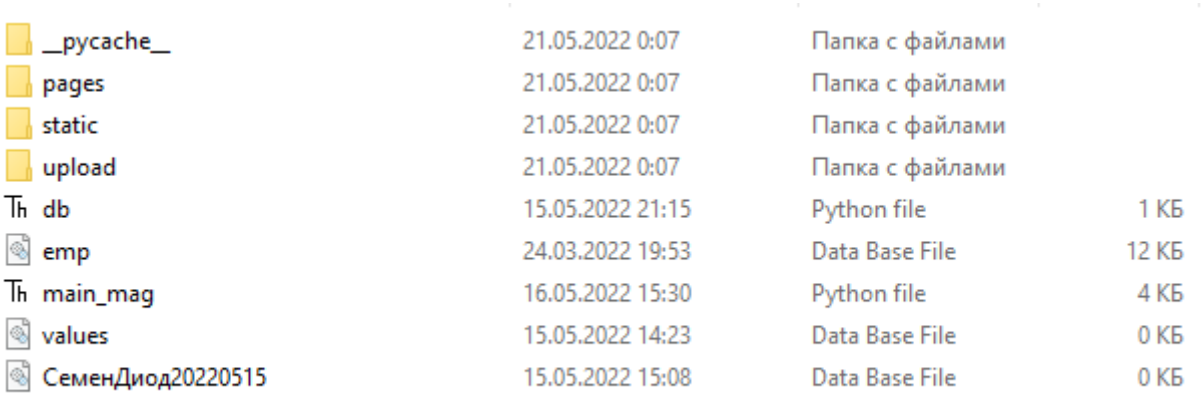


Рисунок 40 – Содержимое папки Site

Папка pages предназначена для хранения html файлов, папка static – для хранения изображений и файлов css, а в папку upload будут попадать данные, которые пользователи будут загружать на сайт.

Файл main_mag это основной файл, в котором прописаны запуск сервера, отображение html страниц при переходах на нужный адрес и вся логика сайта. Первым делом в этом Python файле импортируются все необходимые для работы модули и библиотеки:

```
from flask import Flask, render_template, request, session, redirect,flash,url_for
import os
import db
import matplotlib.pyplot as plt
```

```
import pylab
```

Затем создается объект класса Flask с указанием папки, где будут храниться html файлы:

```
app = Flask(__name__, template_folder="pages")
```

После этого необходимо использовать декоратор route, с помощью которого можно задать url, который должен запускать ту или иную функцию. Первый url будет отвечать за главную страницу сайта и выводить содержимое файла main.html на экран:

```
@app.route('/')
def main():
    return render_template('main.html')
```

Чтобы приложение запустилось необходимо задать секретный ключ и использовать метод run:

```
if __name__ == "__main__":
    app.secret_key = 'key'
    app.run()
```

Теперь необходимо в папке pages создать html файл с именем main.html. Создание структуры любой HTML-страницы уже было подробно описано в разделе 5.3.1, поэтому здесь будут описаны только не затронутые ранее детали.

На главной странице будет располагаться меню с ссылками на другие страницы. Оно располагается внутри тега header, говорящего о том, что это заглавная часть страницы. Внутри находится тег nav, предназначенный для хранения навигационной части сайта. И, наконец, теги ul, li и a создают перечисление из ссылок, ведущих на другие страницы:

```
<header>
    <nav>
        <ul>
        <li><a class="menu" href ='/'>На главную</a></li>
```

```
            <li><a class="menu" href ='/upload'>Загрузить новые данные</a></li>
            <li><a class="menu" href ='/find'>Найти прошлые измерения</a></li>
            </ul>
          </nav>
        </header>
```

Далее внутри тега main располагается блок div с абзацем p, содержащем текст «Главная страница»:

```
        <div class="blockcenter">
          <p class="blockcenter">Главная страница</p>
         </div>
```

Внизу страницы находится тег footer, являющийся так называемым «подвалом» сайта, где находится служебная информация, например, контакты, адрес или имя создателя сайта:

```
        <footer>
          <div class="blockcenterfooter">
          ТГУ, Стычев С.Н. 2022
        </div>
        </footer>
```

Для красивого отображения всей указанной выше информации в папке static необходимо создать файл main.css и в нем указать все необходимые стили, например, черный цвет фона для всей страницы кроме тех элементов, где не задан свой цвет:

```
        html{
            background-color: rgb(0, 0, 0);
          }
```

Размер шрифта задается свойством font-size, а семейство шрифта при помощи – font-family. За цвет шрифта отвечает свойство color. Чтобы применить общие свойства для всего текста нужно написать это для body:

```
body{
    font-family: 'Roboto', sans-serif;
    font-size: 30px;
color:white;}
```

За равномерное распределение меню по горизонтали отвечает специальное свойство display и justify-content и flex-direction. Они позволяют расположить элементы меню строго по горизонтали с равными отступами между собой:

```
ul{
 display: flex;
 justify-content:space-around ;
 flex-direction: row;
}
```

Теперь при заходе по стандартному адресу http://127.0.0.1:5000/ можно увидеть получившуюся главную страницу (рисунок 41).

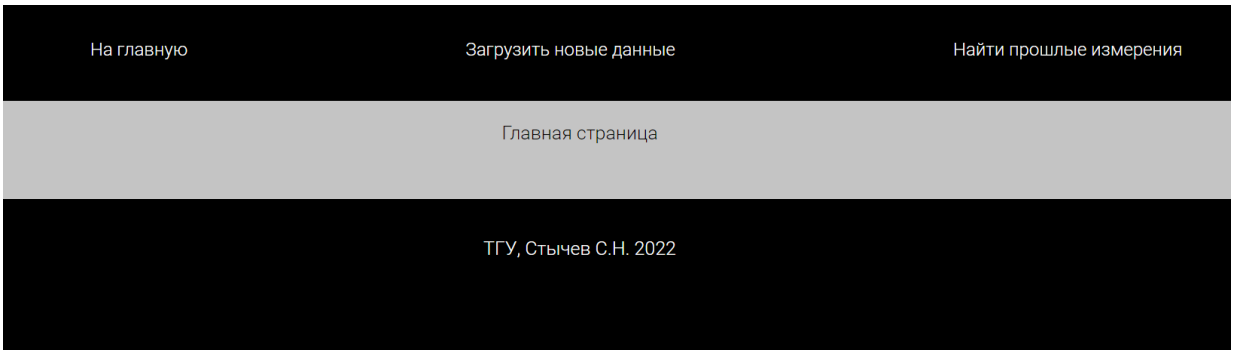


Рисунок 41 – Главная страница сайта

Теперь необходимо разработать страницу для загрузки данных, полученных при помощи платы микроконтроллера Arduino и сохраненных в

файл базы данных на компьютере. В python файле добавим еще один путь при помощи декоратора route, обработав url адрес “upload”. При переходе по нему, нужно вывести страницу upload.html, которая будет описана ниже.

```
@app.route('/upload', methods=['POST', 'GET'])
def upload():
return render_template('upload.html')
```

Если пользователь пытается нажать на кнопку загрузки, при этом не добавив файл, то нужно вывести об этом сообщение:

```
if request.method == 'POST':
    if 'file' not in request.files:
        flash('Не могу прочитать файл')
        return redirect(request.url)
```

Если пользователь прикрепил файл, то нужно сохранить его в папку upload и перенаправить его на страницу find.html для дальнейшей работы с полученными данными:

```
        if file:
        filename = file.filename
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        return redirect(url_for('checkinfo', name=filename))
```

Теперь нужно создать файл upload.html, добавить в него все основные моменты, которые не изменяются от страницы к странице, например, меню и футер. Далее, форма создается при помощи тега form:

```
<form action = "" method = "POST"
  enctype = "multipart/form-data">
```

А поле для загрузки файла и кнопка отправки при помощи тегов input с определенным атрибутом type:

```
<input type="file" name="file">
<input class='button' type="submit"><br>
```

Исходный код сайта не будет передаваться пользователю, а останется только у администратора. Все что нужно пользователю, это пройти на сайт,

нажать на кнопку «Добавить файл», выбрать файл и нажать «Отправить» (рисунок 42).

Рисунок 42 – Форма загрузки файла на сайт

Чтобы просмотреть загруженные данные, а также увидеть график, нужно перейти во вкладку «Найти прошлые измерения».

На этой странице нужно ввести в поиск название файла, который был загружен. После этого на странице отобразится график и поля, позволяющие поменять шаг и ограничения на графике (рисунок 43).

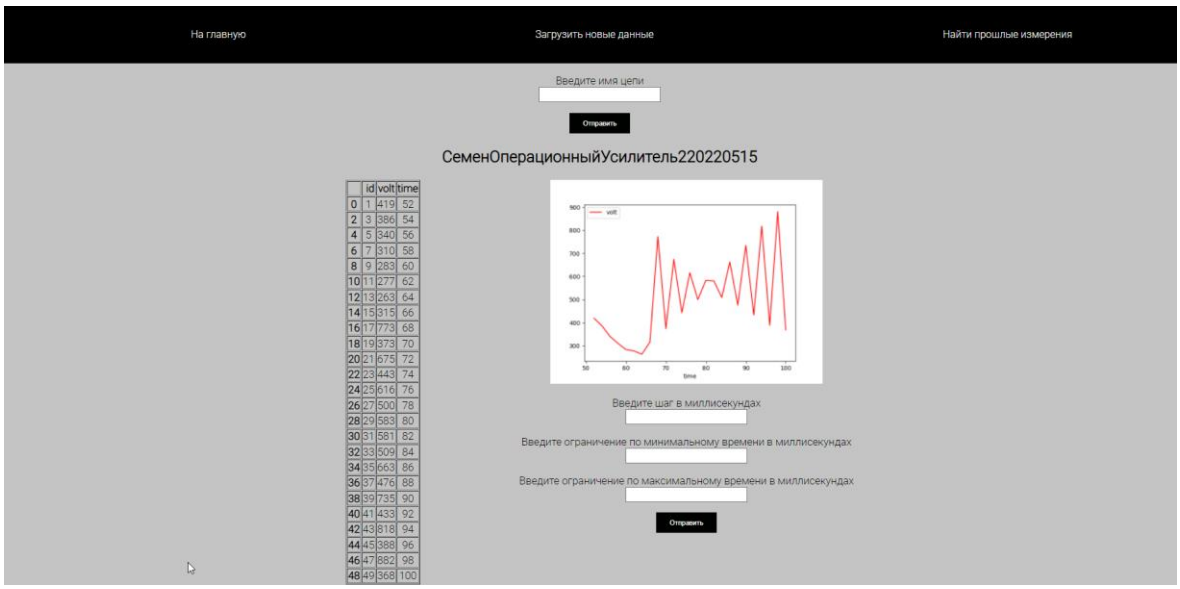


Рисунок 43 – Отображение графика и значений на странице

Если ввести в поля желаемые пороги, то страница сразу же обновится и отобразит новый график.

Эта страница включает в себя все те же основные теги с двух других страниц. Большое отличие в том, что здесь все результаты выводятся в виде таблицы и графиков. Это реализовано при помощи двух популярных библиотек языка Python – Pandas и MatPlotLib.

Pandas — это пакет Python с открытым исходным кодом, который наиболее широко используется для обработки данных/анализа данных и задач машинного обучения. Он построен поверх другого пакета под названием Numpy , который обеспечивает поддержку многомерных массивов. Как один из самых популярных пакетов обработки данных, Pandas хорошо работает со многими другими модулями обработки данных в экосистеме Python [29].

Pandas упрощает выполнение многих трудоемких повторяющихся задач, связанных с работой с данными, в том числе:

— очистка данных,

— заполнение данных,

— нормализация данных,

— сливание и объединение,

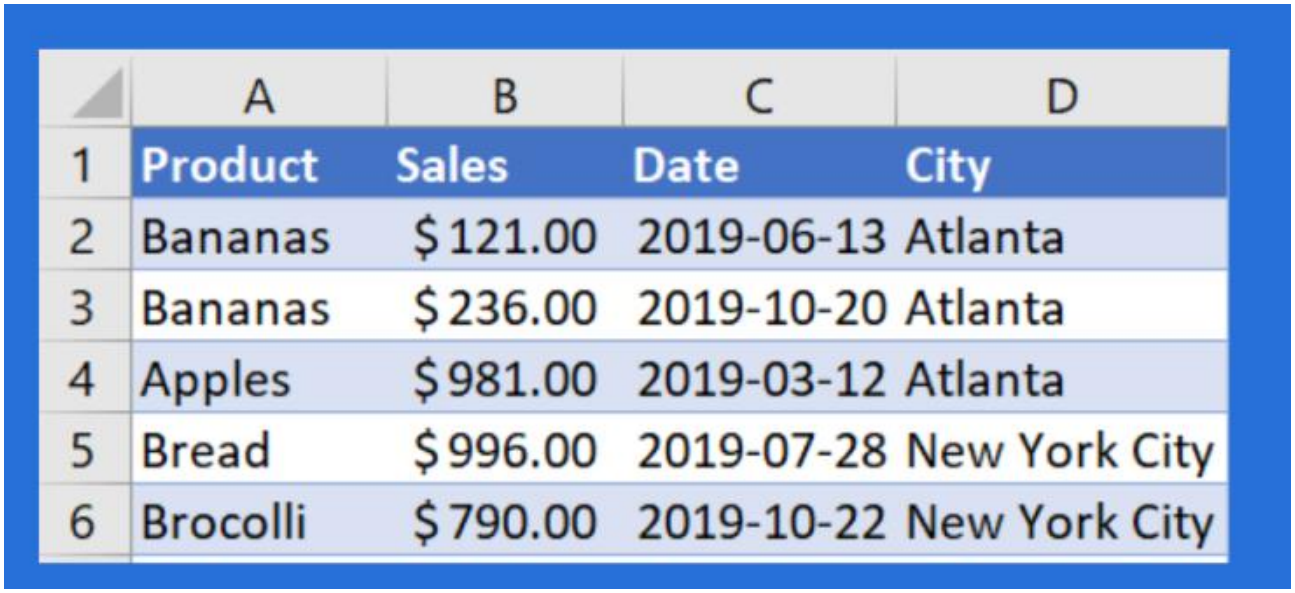
— визуализация данных,

— статистический анализ,

— проверка данных,

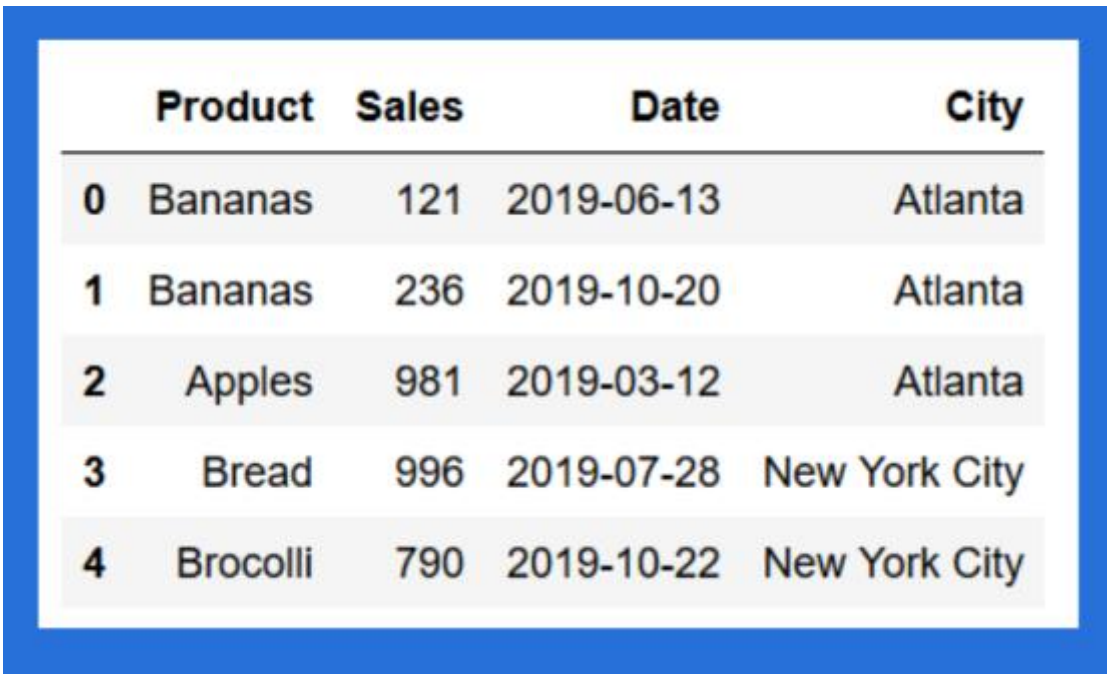
— загрузка и сохранение данных.

Самый популярный тип данных в Pandas это DataFrame — двумерная помеченная структура данных, похожая на таблицу SQL или электронную таблицу со столбцами и строками. Каждый столбец DataFrame может содержать разные типы данных. На рисунке 44 изображена Excel таблица с некоторыми данными, а на рисунке 45 показан DataFrame, полученный из той же таблицы.



	A	B	C	D
1	Product	Sales	Date	City
2	Bananas	$121.00	2019-06-13	Atlanta
3	Bananas	$236.00	2019-10-20	Atlanta
4	Apples	$981.00	2019-03-12	Atlanta
5	Bread	$996.00	2019-07-28	New York City
6	Brocolli	$790.00	2019-10-22	New York City

Рисунок 44 – Пример Excel таблицы



	Product	Sales	Date	City
0	Bananas	121	2019-06-13	Atlanta
1	Bananas	236	2019-10-20	Atlanta
2	Apples	981	2019-03-12	Atlanta
3	Bread	996	2019-07-28	New York City
4	Brocolli	790	2019-10-22	New York City

Рисунок 45 – Пример объекта DataFrame

Синтаксис Pandas DataFrame включает в себя функции «loc» и «iloc», например, data_frame.loc[] и data_frame.iloc[] . Обе функции используются для доступа к строкам и/или столбцам, где «loc» — для доступа по меткам, а «iloc» — для доступа по положению, т. е. числовым индексам.

Одна из наиболее распространенных операций, которые люди используют с Pandas, — это чтение некоторых данных, таких как файл CSV, файл Excel, таблица SQL или файл JSON. Например, чтобы прочитать CSV-файл, нужно ввести следующее:

data_frame = pd.read_csv ("имя_файла_файла.csv")

В нашем случае используются данные из SQL таблицы, которые можно прочитать при помощи метода read_sql передав в качестве аргумента подходящий SQL запрос:

```
df = pandas.read_sql(f'''
   select * from {name};
   ''',conn)
```

Чтобы отобразить данные в виде таблицы необходимо запросить у пользователя имя нужной базы данных, которую он загружал ранее:

```
name_of_file = request.form.get('fname')
```

Затем найти таблицу среди существующих:

```
list_of_tables = db.select_names_of_tables(name_of_file)
```

После этого в переменную result сохраняются найденные результаты:

```
result = db.selectall(session['list_of_tables'][0][0])
```

Затем нужно определить нужные временные границы исследования считав их на сайте при помощи тега input:

```
time_min = request.form.get('xMin')
time_max = request.form.get('xMax')
```

Если пользователь не впишет эти данные, будут выведены все результаты. В любом случае обращение по времени происходит при помощи описанных ранее методов loc и iloc:

```
result = result.iloc[::int(xVal)//10]
result = result.loc[result["time"] >= int(time_min)]
result = result.loc[result["time"] <= int(time_max)]
```

Для отрисовки графиков используется библиотека MatPlotLib — кроссплатформенная библиотека для визуализации данных и графического построения графиков для Python и его числового расширения NumPy. Таким образом, он предлагает жизнеспособную альтернативу MATLAB с открытым исходным кодом. Разработчики также могут использовать API-интерфейсы matplotlib (интерфейсы прикладного программирования) для встраивания графиков в приложения с графическим интерфейсом [28].

Удобство этой библиотеки в том, что график можно построить прямо по данным DataFrame, для этого нужно использовать метод plot и в качестве атрибутов указать оси и цвет графика:

```
first_plot = result.plot(x='time', y='volt',color='red')
```

Затем график сохраняется в виде изображения:

```
plt.savefig(f'static/images/{name_of_file}.png')
```

И выводится на странице при помощи HTML тега img:

<img src="{{url_for('static',filename = 'images/' + fname + '.png')}}" >

Здесь также используется шаблонизатор Jinja — это библиотека Python, которая позволяет создавать шаблоны и подставлять динамически изменяющиеся переменные, например, путь к изображению, как в нашем случае.

Использовать переменные внутри шаблонов можно выполнив следующие два шага:

Объект предоставляется как именованный аргумент render_template функции.

return render_template("index.html", my_object = Object)

Затем значение этого объекта извлекается внутри шаблона (founded.html) с использованием специального синтаксиса:

{{ my_object }}

Таким образом, на сайте отображается нужное изображение.

5 Разработка программы для диагностики работы схемы

Так, до начала этого этапа была решена задача сбора, передачи, хранения данных и даже вывода графической зависимости полученных значений от времени.

Теперь, необходимо разработать логику диагностирования работоспособности схемы.

Для начала необходимо снять показания с рабочей схемы, в качестве примера будет также использоваться схема мультивибратора из раздела 4.5. Полученные значения загружаем на сайт через форму загрузки. Теперь нужно снять показания с другой рабочей схемы, чтобы произвести сравнение двух исправных схем. Сначала нужно наглядно вывести на графике зависимости обеих схем и наглядно оценить, насколько они похожи и действительно ли исследуемая рабочая схема исправна. Для этого вновь воспользуемся инструментами pandas и matplotlib, а именно методом plot [21].

Построение графиков позволяет наглядно увидеть, насколько те или иные данные соответствуют ожиданиям [17].

Однако перед этим создадим новую страницу и новый путь в главном python файле:

```
@app.route('/diagnose', methods=['POST', 'GET'])
def diagnose()
```

Эта страница частично повторяет страницу find и всю ее логику. Отличия в том, что здесь можно не выводить значения в таблице, потому что сравниваться будут графики. Также важное изменение заключается в том, что теперь полей для ввода текста должно быть два – для двух экспериментов (рисунок 46).

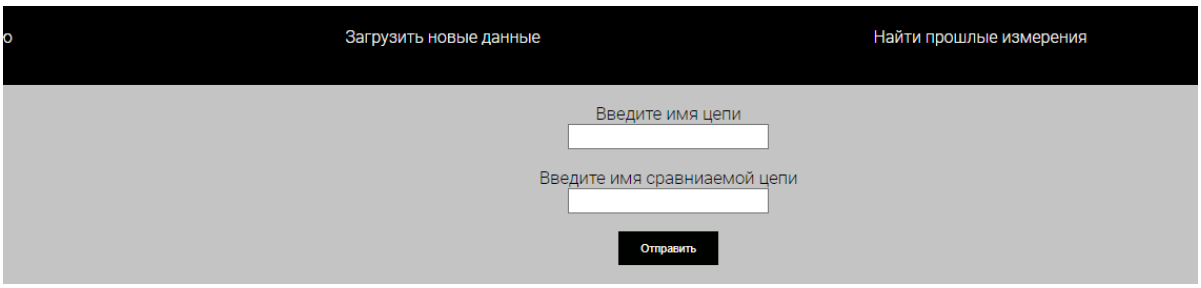


Рисунок 46 – Страница diagnose

Теперь пропишем логику для обработки запрашиваемых экспериментов, добавим на график первую зависимость, загружаемую из первой формы:

first_plot = result.plot(x='id', y='volt', color='yellow', label='result')

Затем добавим на этот график вторую зависимость:

result2.plot(x='id', y='volt', ax=first_plot, color='blue')

Как будет производиться диагностика? При помощи метода diff, который предназначен для вычисления разницы между строками, можно сравнить значения одного и того же периода времени двух исследуемых схем и определить среднюю разность между ними и тем самым понять, какая погрешность допустима, чтобы считать схему рабочей [20].

Прежде чем обрабатывать данные, массивы нужно подготовить, а именно, при помощи метода shift сдвинуть первые несколько значений, которые не всегда являются точными в следствии длительного запуска программы [1].

После этого нужно взять значения по абсолютной величине и найти среднее значение разности:

mean_diff = (abs_df.iloc[1]).abs().mean()

Далее используя шаблонизатор Jinja вывести график и итоговое значение на сайт (рисунок 47) [18].

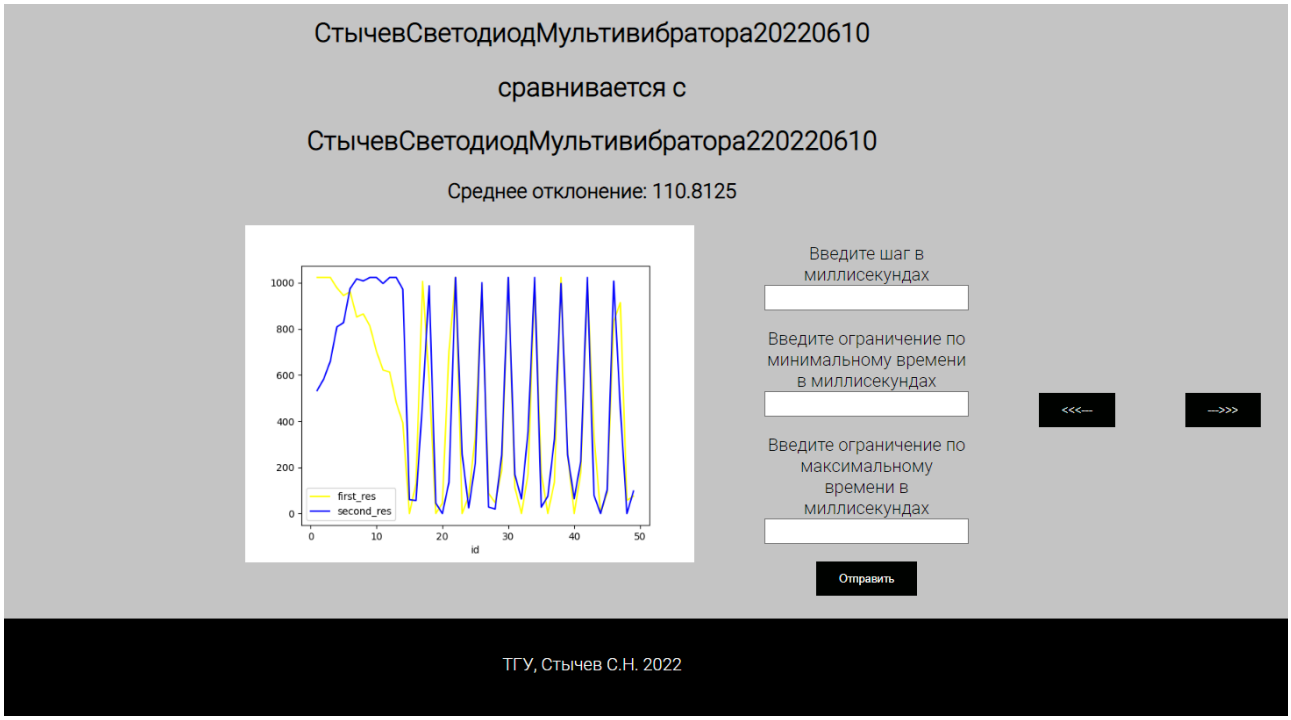


Рисунок 47 – Страница diagnose после отправки данных

Из рисунка видно, что, судя по графику значения действительно очень схожи и схемы работают практически одинаково. Отличия связаны как с помехами, так и с индивидуальными особенностями каждого отдельно взятого компонента схемы.

Среднее отклонение составило 105, учитывая разрядность АЦП, нужно разделить 105 на 1023 и умножить на 5, и тем самым получить приблизительное значение 0.5 вольт – реальная погрешность измерений, означающая, что диагностируемая схема работает исправно.

Также можно проанализировать работу светодиодов и посмотреть, как отличается работа двух светодиодов из одной партии, один из которых ранее не эксплуатировался, а второй эксплуатировался довольно долго. На рисунке 48 приведен результат сравнения работы одной и той же схемы с целью получения допустимого значения погрешности. Как видно из рисунка оно примерно равно 8.

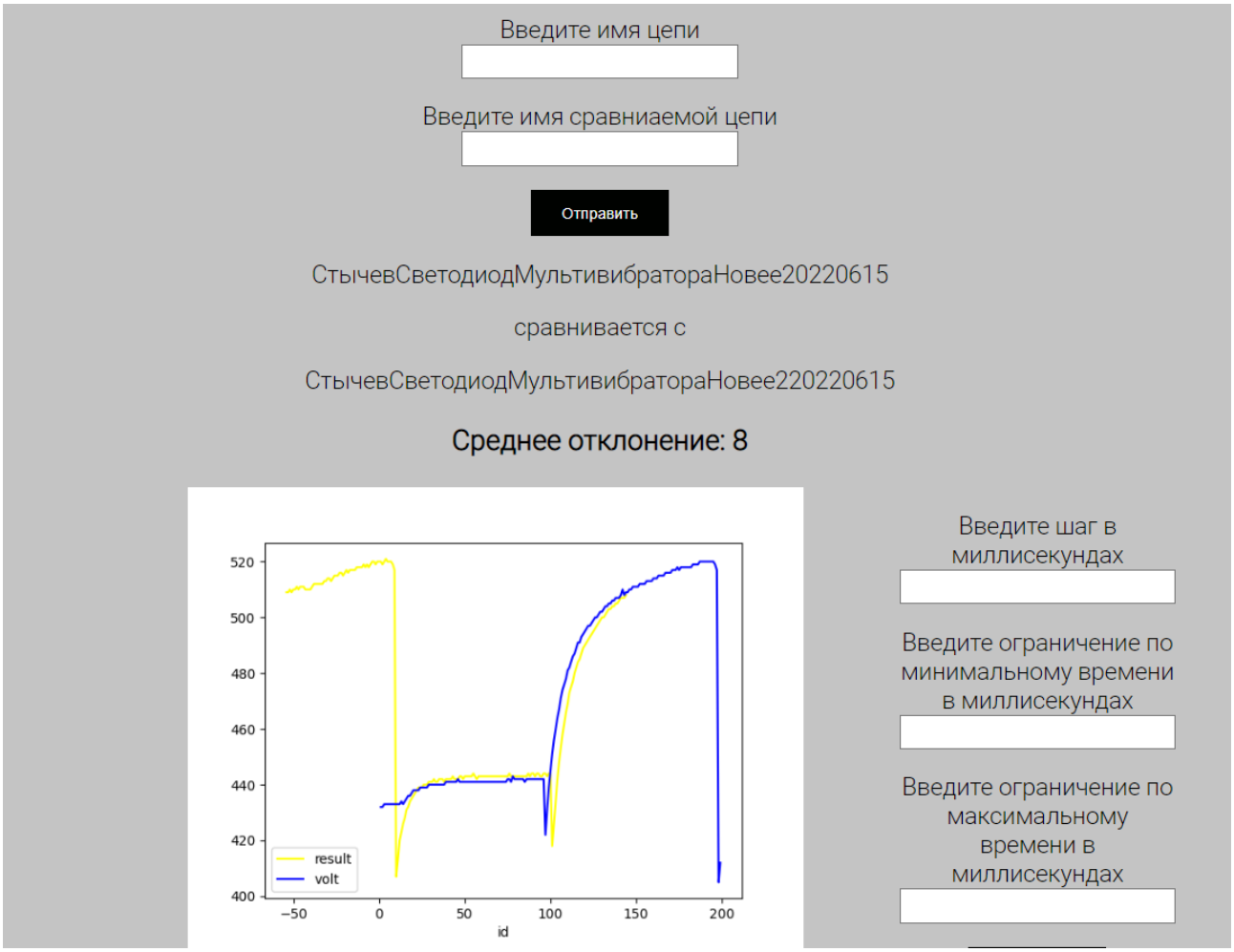


Рисунок 48 – Результат сравнения графиков двух одинаковых светодиодов

На рисунке 49 показан результат сравнения двух светодиодов, работающих в схеме мультивибратора. Желтый график отвечает за показания старого светодиода и синий – за показания нового, не бывавшего в эксплуатации.

Рисунок 49 – Результат диагностики работы разных светодиодов

Из рисунка можно увидеть, что графики достаточно сильно отличаются. Светодиод, который ранее не использовался, имеет более низкое напряжение включения. Также можно сделать вывод, что поведение графиков в целом достаточно похоже, значит, схема работает исправно.

Заключение

В рамках выполнения магистерской диссертации была разработана система диагностики электроники, включающая в себя две связанные части, первая из которых направлена на сбор, передачу и хранение данных с электронной схемы, а вторая – на анализ полученных данных с целью определения работоспособности схемы.

Во введении проанализировано текущее состояние проблемы диагностики и прогнозирования поломок в электронике и отмечена актуальность разработки систем диагностики.

Первый раздел посвящен общим сведениям об анализе и диагностике схем, необходимости снятия тех или иных электрических параметров и тому, какие выводы можно сделать по известным значениям токов и напряжений.

Второй раздел посвящен выбору комплектующих для выполнения магистерской диссертации и обоснован выбор программных и электронных средств, используемых в данной работе.

В третьем разделе была реализована связь микроконтроллера с датчиком тока и написано программное обеспечение для снятия напряжений и токов с элементов схемы. Проанализированы готовые решения в этой области и подробно описаны их преимущества и недостатки. Протестирована собственная реализация выполнения данной задачи.

В четвертом разделе написано программное обеспечение для передачи и хранения полученных со схемы данных. Обоснован выбор инструментов, используемых для написания программного кода и сбора электронной схемы. Произведен анализ способов автоматизации запуска программы. Реализован и обоснован способ автоматизирован запуска программы для считывания напряжений и токов со схемы при помощи командной строки компьютера. Разработан веб-интерфейс для хранения данных. Продемонстрирован способ загрузки и чтения данных при помощи интерфейса.

В пятом разделе описан процесс разработки новой страницы сайта, реализующей анализ загружаемых значений. Написано программное обеспечение, которое самостоятельно проводит сравнение двух загруженных временных зависимостей, путем попарного сравнения значений, соответствующих одинаковым временным промежуткам. Приведены результаты анализа исправных и неисправных схем, приведено обоснование заключений по поводу работоспособности схемы. Результаты анализа представлены в виде графиков и численных значений отклонений между снятыми со схемы показаниями.

Таким образом, результатом выполнения магистерской работы является исправно работающая система сбора, хранения и передачи электрических данных, предназначенная для определения работоспособности электронных схем и отдельных компонентов.

Список используемой литературы

1. Аналитикам: большая шпаргалка по Pandas // smysl URL: https://smysl.io/blog/pandas/ (Дата обращения: 05.04.2022).
2. Выжимаем максимум. USB осциллограф на Arduino // Arduino URL: http://arduino.ru/forum/proekty/vyzhimaem-maksimum-usb-ostsillograf-na-arduino (дата обращения: 17.10.2021).
3. Датчик тока ACS712 // 3d-diy URL: https://3d-diy.ru/wiki/arduino-datchiki/datchik-toka-acs712/ (дата обращения: 08.03.2022)
4. Зачем нужен анализ отказов в микроэлектронике // sernia инжиниринг URL: https://sernia.ru/training/zachem_nuzhen_analiz_otkazov_v_mikroelektronike/ (дата обращения: 17.05.2022).
5. Модуль подпроцесса Python // pythonbyte URL: https://pythobyte.com/python-subprocess-module-efb29876/
6. Описание основных функций языка Arduino URL: http://freeduino.ru/arduino/lang.html (дата обращения: 11.02.2021)
7. Подключение датчика тока ACS712 к Arduino // robotchip URL: https://robotchip.ru/podklyuchenie-datchika-toka-acs712-k-arduino/ (дата обращения: 08.03.2022)
8. Построение вольтамперных кривых. // studref. URL: https://studref.com/462620/matematika_himiya_fizik/postroenie_voltampernyh_krivyh (дата обращения: 11.06.2022).
9. Простой осциллограф на Arduino своими руками // DIGITRODE URL: http://digitrode.ru/computing-devices/mcu_cpu/330-prostoy-oscillograf-na-arduino-svoimi-rukami.html (дата обращения: 17.10.2021).
10. Спектральный анализ токов и напряжений выпрямителей // poznayka URL: https://poznayka.org/s95451t1.html (дата обращения: 27.12.2021).
11. Что такое база данных? // ORCALE URL: https://www.oracle.com/cis/database/what-is-database/ (дата обращения:

27.10.2021).

12. Что такое последовательный (COM-порт) // Encont URL: https://us800.ru/com.htm (дата обращения: 12.04.2022).

13. Электрик в доме // ratingservices URL: https://ratingservices.ru/lighting/kalibrator-osc-generatora-i-ustrojstvo-vosstanovlenia-fuzov-avr-mikrokontrollerov.html (дата обращения: 07.11.2021).

14. Arduino CLI (Command Line Interface) Application // arduino URL: https://www.arduino.cc/pro/cli (дата обращения: 11.03.2022).

15. Arduino Oscilloscope (6-Channel) // ProjectHUB URL: https://create.arduino.cc/projecthub/Meeker6751/arduino-oscilloscope-6-channel-674166?ref=platform&ref_id=424_trending___&offset=3 (дата обращения: 17.02.2022).

16. ATMega328P Microcontroller // Components101 URL: https://components101.com/microcontrollers/atmega328p-pinout-features-datasheet (дата обращения: 27.09.2021).

17. Creating Diagnostic Plots in Python // Robert Alvarez URL: https://robert-alvarez.github.io/2018-06-04-diagnostic_plots/ (Дата обращения: 15.05.2022).

18. Customization with Jinja2 // pyembed URL: https://pyembed.github.io/customization/jinja2/ (Дата обращения: 05.05.2021).

19. Overview of wxPython // wxpython URL: https://wxpython.org/pages/overview/index.html (дата обращения: 07.04.2022).

20. Pandas Diff: Calculate the Difference Between Pandas Rows // datagy URL: https://datagy.io/pandas-diff/ (Дата обращения: 05.04.2022).

21. Pandas.DataFrame.diff // pandas URL: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.diff.html (Дата обращения: 03.04.2022).

22. Welcome to PyAutoGUI’s documentation! // pyautogui URL: https://pyautogui.readthedocs.io/en/latest/ (дата обращения: 08.03.2022).

23. What does an oscilloscope measure? // Tektronix URL: https://www.tek.com/en/blog/what-can-an-oscilloscope-measure (дата обращения:

07.05.2022).

24. What is Cursor in SQL? // geeksforgeeks URL: https://www.geeksforgeeks.org /what-is-cursor-in-sql/ (дата обращения: 07.03.2022).

25. What is Django? // IBM URL: https://www.ibm.com/cloud/learn/django-explained (Дата обращения: 02.02.2021).

26. What is Flask Python // Python Tutorial URL: https://pythonbasics.org/what-is-flask-python/ (Дата обращения: 02.02.2021).

27. What is HTML – Definition and Meaning of Hypertext Markup Language // freeCodeCamp URL: https://www.freecodecamp.org/news/what-is-html-definition-and-meaning/ (дата обращения: 12.04.2022).

28. What Is Matplotlib In Python? // Activestate URL: https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/ (Дата обращения: 05.03.2022).

29. What Is Pandas In Python? Everything You Need To Know // activestate URL: https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/ (Дата обращения: 05.02.2021).

30. What is PyQt? // pythonpyqt URL: https://pythonpyqt.com/what-is-pyqt/ (дата обращения: 07.04.2022).

31. What Is Tkinter Used For And How To Install It? // activestate URL: https://www.activestate.com/resources/quick-reads/what-is-tkinter-used-for-and-how-to-install-it/ (дата обращения: 07.04.2022).