Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
Санкт-Петербургский горный университет

Кафедра информатики и компьютерных технологий

ИНФОРМАТИКА

ПРОГРАММИРОВАНИЕ ПЕРЕХОДНЫХ ПРОЦЕССОВ В ИНТЕГРИРОВАННОЙ СРЕДЕ РАЗРАБОТКИ ARDUINO

Методические указания к курсовым работам
для студентов специальности 21.05.04

САНКТ–ПЕТЕРБУРГ
2020

УДК 004.9 (073)

ИНФОРМАТИКА. Программирование переходных процессов в интегрированной среде разработки arduino. Методические указания к курсовой работе /Санкт-Петербургский горный университет. Сост.: *Е.Г. Водкайло, О.В. Косарев.* СПб, 2020. 34 с.

Рассмотрен пример программирования работы переходных процессов в RC-цепи на базе аппаратно-программного комплекса Arduino. Приведены общие сведения об Arduino UNO и программной среде Arduino IDE, необходимые для начала работы с комплексом.

Предназначены для студентов специальности 21.05.04 «Горное дело» профиль «Электрификация и автоматизация горного производства»

Научный редактор доц. *А.Б. Маховиков*

Рецензент канд. техн. наук *К.В. Столяров* («Telum Inc»)

ВВЕДЕНИЕ

Курсовая работа посвящена расчету и исследованию переходных процессов в цепях первого порядка. Расчет параметров переходного процесса выполнен в математическом пакете PTC Mathcad 15, моделирование переходных процессов реальной выполнено на базе аппаратно-программного комплекса Arduino Uno. Пакет Mathcad 15 позволяет выполнить расчет параметров цепи с использованием встроенных единиц измерения физических величин. Использование Arduino позволят собрать простую лабораторную установку и закрепить навыки программирования.

Допустимо выполнять расчеты в SMath Studio или другом пакете, а также использовать любой доступный аппаратно-программный комплекс для моделирования.

ЗАДАНИЕ

В методических указаниях к курсовой работе описан процесс компьютерного моделирования переходных процессов, возникающих при коммутациях в цепях первого порядка, содержащих сопротивление и емкость. В курсовой работе необходимо исследовать зависимости напряжения в емкости RC-цепи при заряде и разряде конденсатора, построить зарядные и разрядные кривые конденсатора, т.е. показать четыре зависимости – две теоретических и две экспериментальных. Из экспериментальной разрядной кривой конденсатора определить экспериментальное значение постоянной времени *RC*–цепи $\tau_{экс}$ и сравнить полученное значение с теоретическим значением постоянной времени $\tau_{теор}$. Все теоретические расчеты произвести в математическом пакете MathCad, экспериментальные данные получить с помощью программного комплекса Arduino.

В процессе выполнения курсовой работы студенту необходимо: подключить сопротивление и конденсатор к платформе Arduino, запрограммировать работу на считывание физических параметров в течение определенного периода времени. По полученным данным построить графики в MathCad. Результаты работы оформить по ГОСТ 7.32. Программный код и блок-схему алгоритма работы RC-цепи вынести в приложение. Схему подключения выполнить в программе Fritzing и подтвердить фото своей собранной установки. Следует помнить, что все подключения проводов к плате необходимо выполнять после отключения питания! Все примеры приведены для операционной системы Windows 10.

1. ИСХОДНЫЕ ДАННЫЕ

Исходными данными для выполнения расчетов являются: физические величины, подлежащие регистрации (емкость, сопротивление, напряжение на источнике, зависимость напряжения на конденсаторе от времени его заряда и разряда, постоянная времени *RC*–цепи), форма представления результата (график изменения величин и линия тренда). Необходимое оборудование: аппаратно-программный комплекс Arduino, сопротивление и конденсатор. Необходимое программное обеспечение: среда программирования Arduino IDE или любая другая, MathCad и MS Word любой версии.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В любой реальной электрической цепи при ее включении и выключении происходят переходные процессы. Переходными процессами, в электрической цепи, принято называть процессы возникающие в результате различных воздействий (например: включений или отключений цепи от источника питания, обрывах или коротких замыканиях, импульсных возмущающих воздействий и так далее) и переводящих её из одного стационарного (установившегося) состояния в новое (другое) стационарное состояние [1].

При переходных процессах могут возникать большие перенапряжения, сверхтоки, электромагнитные колебания, которые способны нарушить работу систем автоматики и других устройств, вплоть до выхода их из строя. С другой стороны, переходные процессы находят практическое применение, например, в различного рода электронных генераторах, в схемах электроники и автоматики. Таким образом, переходные процессы в электрических схемах играют важное значение. Это обуславливает актуальность изучения явлений, рассматриваемых в данной курсовой работе [1, 2].

Рассмотрим переходный процесс в RC-цепи (рисунок 1), в состав которой входят резистор *R*, конденсатор *C*, ключ К и источник питания, на зажимах которого поддерживается постоянное напряжение *E=U*.

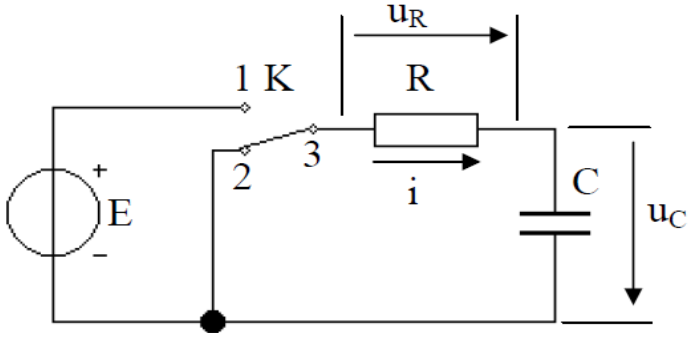


Рис. 1. Схема RC-цепи

Установив переключатель К в положение 1, мы будем заряжать конденсатор, а в положении 2 – разряжать конденсатор.

Рассмотрим вначале процесс заряда. Для образовавшейся цепи, согласно второму правилу Кирхгофа [1, 3], будет справедливо соотношение:

$$RI + U_C = U, \tag{1}$$

где I – мгновенное значение силы тока в цепи, U_c – мгновенное значение напряжения на конденсаторе. Но

$$U_c = \frac{q}{C}, \tag{2}$$

$$I = \frac{dq}{dt}, \tag{3}$$

где q – заряд конденсатора, t – время, прошедшее с момента начала заряда, т.е. с момента включения ключа [1, 3].

Из (1) – (3) получается дифференциальное уравнение с постоянными коэффициентами для нахождения U_c:

$$RC\frac{dU_c}{dt} + U_c = U, \tag{4}$$

Так как на конденсаторе напряжение скачком изменяться не может, то в момент (t=0) подключения цепи к источнику питания всё напряжение источника окажется на резисторе R, то есть $U_R = U$, $U_c = 0$. С учетом этих начальных условий ($t = 0$, $U_c = 0$) запись полного решения дифференциального уравнения [2, 3] относительно $U_c(t)$ будет иметь вид:

$$U_c = U(1 - e^{-\frac{t}{RC}}). \quad (5)$$

В начальный момент времени заряда конденсатора, ток в RC-цепи будет иметь наибольшее значение: I=U/R. Конденсатор начнёт заряжаться, напряжение на нём “постепенно” повышается, что, в свою очередь, приведёт к уменьшению падения напряжения на резисторе $U_R = U — U_C$, а следовательно и уменьшению тока в RC-цепи, вплоть до его ”полного” прекращения. Напряжение на конденсаторе, во время заряда, нарастает по экспоненциальной зависимости согласно формуле (5).

Значения напряжения на резисторе и общего тока RC-цепи уменьшаются также по экспоненциальному закону:

$$U_R = Ue^{-\frac{t}{RC}}, \quad I = \frac{U}{R}e^{-\frac{t}{RC}}. \quad (6)$$

Если установить ключ К в положение 2 (рисунок 1), то начнётся новый переходный процесс — разряд конденсатора C через резистор R. В этом случае предварительно заряженный конденсатор становится фактическим источником напряжения, т.к. источник внешнего напряжения $E=U$ перестаёт действовать и для любого момента времени становится действительным соотношение $U_C + U_R = 0$, то есть $U_C = -U_R$.

В случае разряда конденсатора исходное уравнение будет иметь вид:

$$-IR = U_c, \quad (7)$$

Из (2), (3)– (6) получим дифференциальное уравнение:

$$-RC\frac{dU_c}{dt} = U_c. \quad (8)$$

С учетом того, что перед разрядом на конденсаторе было начальное напряжение $U_c = U_{нач}$ при $t = 0$ и интегрирования уравнения (7) получим зависимость напряжения на конденсаторе от времени в процессе его разряда:

$$U_c = U_{нач}e^{-\frac{t}{RC}}. \quad (9)$$

Качественно процесс зарядки и разрядки конденсатора во времени показан на рис. 2.

Из результатов видно, что процессы заряда конденсатора и разряда конденсатора происходят не мгновенно, а с конечной быстротой. Скорость этих процессов зависит от произведения RC и обозначается τ:

$$\tau = RC\,, \tag{10}$$

где τ имеет размерность времени [с] и называется постоянной времени RC – цепи (рис.2) [1,3].

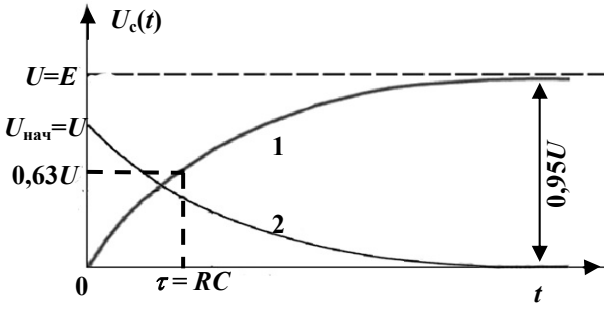


Рис.2. Зарядка (1) конденсатора от источника и его разрядка (2) через резистор

При $t = \tau$, после подключения RC-цепи к источнику постоянного напряжения, напряжение на конденсаторе достигнет значения:

$$U_C = U\,(1 - e^{-1}) = 0{,}63U. \tag{11}$$

Процесс заряда конденсатора будет продолжаться до тех пор, пока напряжения на его выводах не достигнет значения равного напряжению источника питания U. Когда заряд конденсатора закончится - ток в RC-цепи становится равным нулю.

Для полного заряда конденсатора, потребуется бесконечно большое время, поэтому, принято считать, что процесс заряда и разряда емкости, практически, полностью заканчивается за время 3τ или, когда напряжение на конденсаторе достигает значения 95 % величины напряжения источника питания $U=E$.

3. ИССЛЕДОВАНИЕ ПЕРЕХОДНОГО ПРОЦЕСС В RC-ЦЕПИ В MATHCAD

Рассмотрим использование пакета PTC Mathcad для расчета переходных процессов и построения графиков с учетом физических размерностей токов, напряжений и номиналов элементов RC-цепей. При расчетах будем использовать готовые аналитические выражения, оставив за скобками процесс их вывода.

Сначала произведем расчет теоретического значения напряжения на конденсаторе в процессе заряда и разряда по формулам (5) и (9) соответственно (рис.3). Рассчитать теоретическое значение постоянной времени $\tau_{теор}$ по формуле (10) (рис.3). По полученным данным на одном координатном поле (в одних координатных осях) построим зарядные и разрядные кривые конденсатора, т.е. зависимости $U_c(t)$ (рис.4).

Исходные данные:

$U := 5V$

$R := 10K\Omega$

$C := 22\mu F$

Постоянная времени: $\tau := R \cdot C = 0.22\,s$

Диапазон времени для заряда кондесатора:

$t := 0s, 0.01s .. 3 \cdot \tau$

Напряжение на кондесаторе во время заряда:

$$Uc(t) := U \cdot \left(1 - e^{\frac{-t}{R \cdot C}}\right)$$

$Uc(\tau) = 3.161\,V$

Напряжение на кондесаторе во время разряда:

$$Ucp(t) := U \cdot e^{\frac{-t}{R \cdot C}}$$

Рис. 3. Параметры цепи

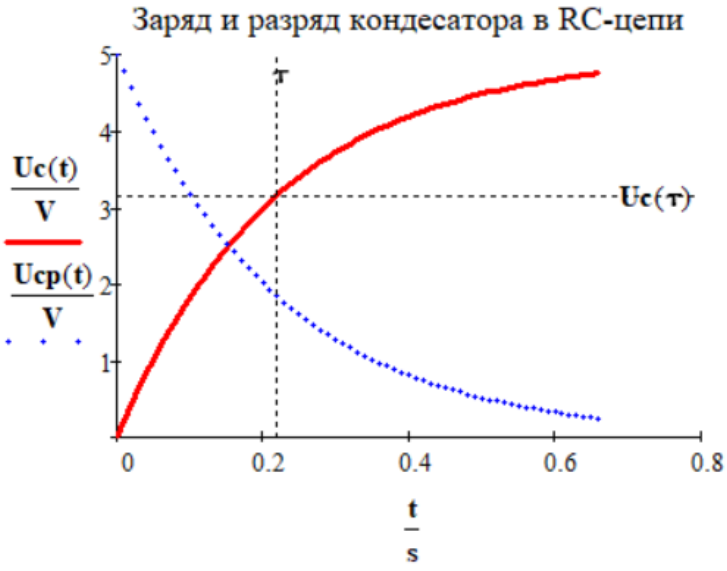


Рис. 4. Графики переходных процессов

Обратите внимание рисунке 3 показан код, где заданы номиналы элементов цепи с указанием единиц измерения, рассчитана постоянная времени τ. Значение τ автоматически выводится в секундах. При необходимости секунды можно заменить на миллисекунды, процесс пересчета мантиссы параметра происходит автоматически. Буквы нижнего регистра в именах переменных вводятся после точки. Не стоит путать нижний регистр текста с индексом переменной, это не одно и то же.

Особенности графика в Mathcad: функция и аргумент заданы как числитель дроби на осях, знаменатель дроби – единица измерения; в знакоместа маркеров по осям (черные точки по осям, включаются в настройках рисунка) записаны характерные точки переходного процесса.

4. ОБЩИЕ СВЕДЕНИЯ ОБ ARDUINO

Проект Arduino представляет собой совокупность аппаратного (электронная плата с микроконтроллером и интерфейсами, как правило это Arduino UNO) и программного обеспечения (среда программирования, например, Arduino IDE) для создания проектов в области робототехники, интернета вещей, умного дома. Проект открытый, бесплатный, предназначен для тех, кто не имеет познаний в электронике, компьютерах и программировании. Сайт проекта – www.arduino.cc. Проект развивается под этим названием с 2008 года. Поэтому сейчас существует бесчисленное множество клонов платы Arduino UNO, примеров программ (скетчей), датчиков, двигателей, корпусов и других запчастей. Стоимость самой платы Arduino UNO – 300-400 рублей, датчиков – от 50 рублей. Программное обеспечение Arduino IDE – бесплатное. Для сборки проекта не нужен паяльник – все соединения выполнены на разъемах. Таким образом, стоимость простого проекта не превышает стоимости обеда в Макдональдсе. Рассмотрим более подробно плату Arduino UNO и ее подключение к компьютеру.

Внешний вид платы Arduino UNO (один из возможных вариантов) показан на рисунке 5. Это рисунок из руководства «Быстрый старт. Первые шаги по освоению Arduino» с сайта Maxkit.ru, являющегося, в свою очередь, переводом с английского языка руководства «SIK GUIDE» с сайта www.sparkfun.com/SIK. Для изучения можно использовать любую другую доступную книгу [4,5] и информацию из интернета [6]. На сайте проекта www.arduino.cc есть много полезной информации и форум разработчиков устройств.

На рисунке 5 цифрами обозначены:

1 – разъем питания. Можно не использовать, если плата подключена к порту USB компьютера;

2 – разъем USB для подключения к компьютеру;

3, 4 – индикаторы, сигнализирующие о том, что плата принимает (3, Rx) или передает (4, Tx) данные;

5 – контрольный индикатор. Используется для контроля правильности работы загруженной программы. Мы будем мигать этим индикатором;

6 – разъемы с цифровыми портами ввода-вывода (DIGITAL, пронумерованы цифрами от 0 до 13). К этим портам будут подключаться цифровые датчики. Порты работают как на ввод информации, так и на вывод;
7 – индикатор питания. Если горит – значит, питание на плату подано. Этот индикатор может быть расположен в любом другом месте платы;
8 – кнопка сброса. Жмите её, если ничего не получается ☺;
9 – разъем для программирования платы (порт ICSP). В рамках курсовой работы этот разъем использоваться не будет;
10 – разъемы питания датчиков (POWER) и аналоговые порты ввода A0-A5 (ANALOG IN). К портам A0-A5 обычно подключаются разнообразные приводы (исполнительные механизмы) или на них подаются аналоговые сигналы.

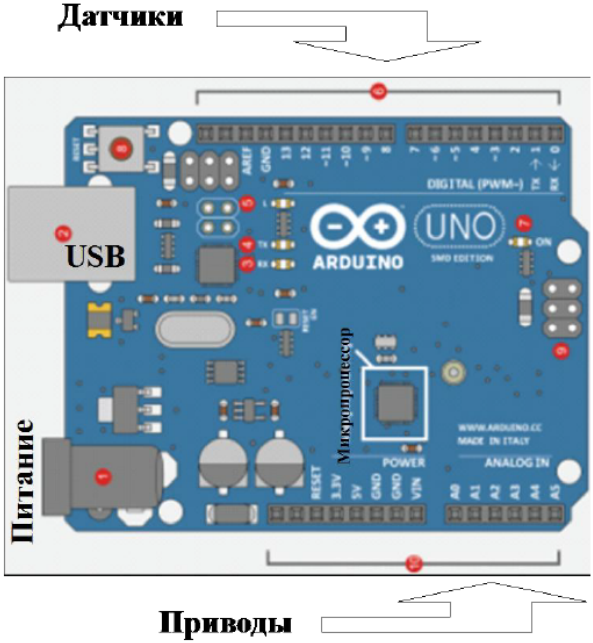


Рис.5. Внешний вид платы Arduino Uno

Плата работает следующим образом. В память микропроцессора записывается программа. По алгоритму, заложенному в программе, опрашиваются подключенные к плате датчики или подаются сигналы управления на двигатели, светодиоды и другие устройства. Управляет процессом сбора информации с датчиков и выполнением промежуточных расчетов микропроцессор, расположенный на плате. Результаты выполнения программы сохраняются в памяти микропроцессора. На рисунке 5 микропроцессор находится примерно посередине платы. Это такая маленькая квадратная микросхема, обведенная белым прямоугольником. Микропроцессор может быть и в другом корпусе, например длинном прямоугольном. На работу это не влияет.

В платах Arduino UNO используется 8-битный RISC-микропроцессор ATmega328P фирмы Atmel. Микропроцессор построен по гарвардской архитектуре с раздельными памятью и шинами команд и данных. Обратите внимание, что она отличается от архитектуры фон Неймана (рисунок 6), используемой в классических компьютерах и ноутбуках. На рисунке 7 показана архитектура семейства микропроцессоров ATmegaXXXX. Сравните ее с архитектурой на рисунке 6 и найдите все элементы. Если плата построена на другом процессоре – ничего страшного, все будет работать.

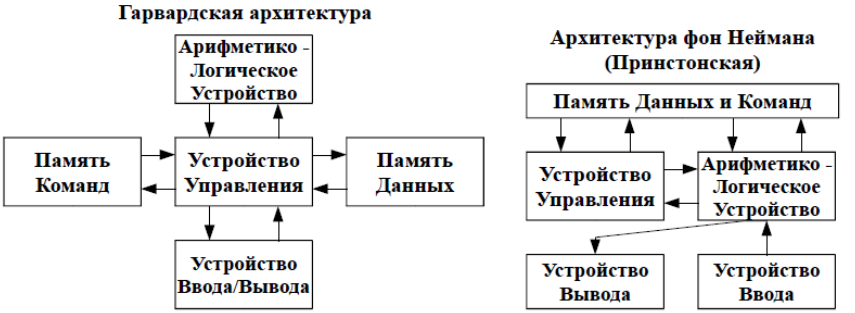


Рис. 6. Гарвардская и Принстонская архитектуры ЭВМ

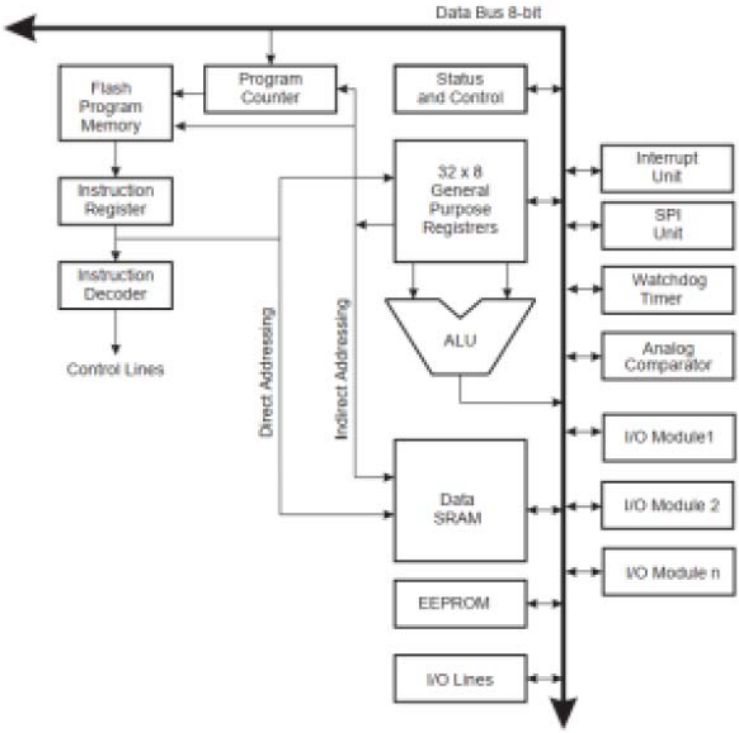


Рис. 7. Архитектура семейства микропроцессоров ATmegaXXXX

Теперь рассмотрим, как подключить плату к компьютеру и загрузить программу в память микропроцессора.

5. ПРОГРАММИРОВАНИЕ ARDUINO

Для работы с Arduino на компьютер необходимо установить специальную программу. Существует несколько вариантов. На занятиях будет использоваться Arduino IDE (от англ. Integrated Development Environment – интегрированная среда разработки). Программу можно скачать с сайта по адресу https://www.arduino.cc/en/Main/Software. Доступны версии под операционные системы Windows, Linux и Mac OS. В пособии все примеры приведены для версии 1.8.10.

Программа устанавливается обычным образом. Если права пользователя не позволяют установить программу, следует скачать автономную версию (Windows ZIP file for non admin install). После установки ее необходимо запустить, подключить плату Arduino к компьютеру по USB и указать в Arduino IDE к какому COM-порту подключена плата. Для связи платы с компьютером по USB на плате установлена специальная микросхема – преобразователь интерфейса USB в последовательный интерфейс UART. В недорогих платах это обычно микросхема CH340G или ATmega16U2, которая полностью эмулирует работу стандартного COM порта. Для корректного определения платы Arduino в операционной системе Windows необходимо установить драйвер этой микросхемы. Если плата определилась сразу после установки – ничего делать не нужно.

Окно установленной Arduino IDE выглядит, как показано на рисунке 8. Если плата подключена правильно, то можно прочитать информацию о ней по команде:

Инструменты\Получить_информацию_о_плате (рисунок 9).

Рассмотрим структуру программы. Программа для Arduino называется скетчем. Для записи скетчей практически не требуется никаких знаний о программировании. Тем не менее, будет полезно вспомнить три типа вычислительных процессов, определение типов переменных и функций, объекты, методы и свойства объектов.

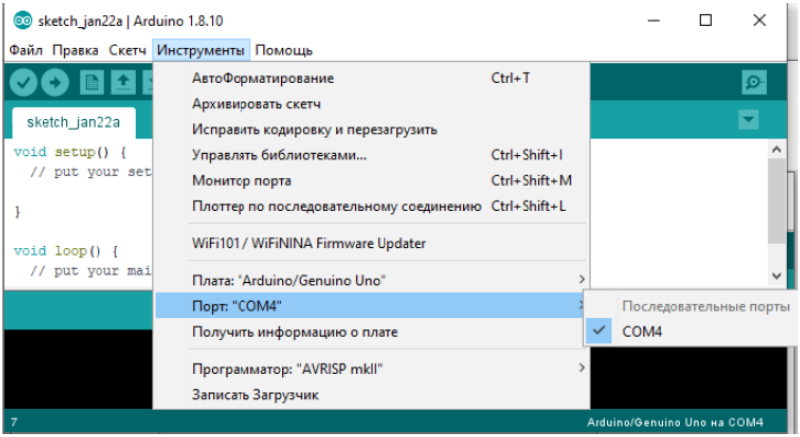


Рис. 8. Подключение платы в Arduino IDE

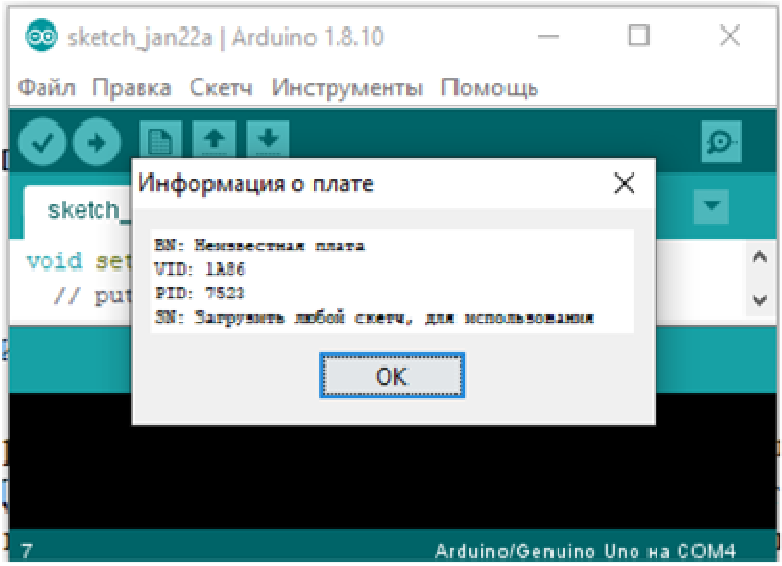


Рис.9. Информация о плате в Arduino IDE

Структура скетча для Arduino содержит:

1. Заголовок в виде комментария. Описано название и предназначение скетча. Может быть указана другая вспомогательная информация. **Это необязательная часть программы.**

// Так пишется однострочный комментарий

/ А так пишется многострочный комментарий.*

*Давайте помигаем светодиодом на плате. */*

2. Инструкция #define или константа const. Подключение библиотек #include. Инструкция #define используется для упрощения кода. Синтаксис: #define <что меняем> <на что меняем>. Очень похожа на переменную или константу. Замена происходит во всем коде перед компиляцией (преобразованием текста программы в двоичный код, вспоминаем принципы функционирования ЭВМ). Например, будем использовать для удобства название PIN_LED для обозначения порта, к которому подключен светодиод: #define PIN_LED 13. Теперь дальше в коде мы будем использовать имя PIN_LED, а вместо него будет подставляться число 13. Аналогичный результат модно получить при использовании константы: *const int PIN_LED = 13;*. Здесь «int» – тип константы PIN_LED, целочисленный тип. Аналогично подключаются специальные программы для работы с датчиками и приводами – библиотеки. Для подключения библиотеки используется инструкция #include. В этом же разделе при необходимости создаются Объекты подключаемых Классов (библиотек). Подключение библиотек и создание Объектов будет рассмотрено ниже. **Это необязательная часть программы.**

3. Функция void setup(). Функция называется «setup», слово «void» обозначает, что функция не возвращает никаких данных (в противном случае было бы необходимо вместо «void» написать тип возвращаемых данных, например «int» как при константе в п.2 выше). Все команды в этой функции (строки кода) должны находиться между фигурных скобок {}. Команды функции выполняются только один раз во время загрузки скетча. Это дает возможность пользователю поучаствовать в процессе загрузки. Здесь могут описываться номера портов, устанавливаться скорость соединения с COM-портом, подключаться внешние программы (например NI LabVIEW) и др. Пример функции void setup() показан ниже. Здесь «pinMode» -

функция определения режима работы цифрового порта; «PIN_LED» - константа номера порта; «OUTPUT» - режим работы порта, режим=ВЫХОД. **Это ОБЯЗАТЕЛЬНАЯ часть программы.**

```
// Программа мигания светодиодом
//#define PIN_LED 13 // Можно так задать порт
const int PIN_LED = 13; // А можно через константу
void setup() {
// инициализируем цифровой порт PIN_LED как выход.
pinMode(PIN_LED, OUTPUT);
}
```

4. Функция void loop(). Функция называется «loop» (петля). Все команды, расположенные внутри этой функции, выполняются бесконечное количество раз, пока плата Arduino не будет выключена. Это основной блок программы, вся пользовательские алгоритмы заложены здесь. **Это ОБЯЗАТЕЛЬНАЯ часть программы.**

```
// Программа мигания светодиодом
//#define PIN_LED 13 // Можно так задать порт
const int PIN_LED = 13; // А можно через константу
void setup() {
// инициализируем цифровой порт PIN_LED как выход
pinMode(PIN_LED, OUTPUT);
}
void loop() {
// зажигаем светодиод, HIGH – высокий уровень напряже-
ния
digitalWrite(PIN_LED, HIGH);
// ждем 1000 мс
delay(1000);
// гасим светодиод
digitalWrite(PIN_LED, LOW);
// ждем 1000 мс
delay(1000);
// А дальше бесконечный повтор кода в скобках
}
```

5. Функции пользователя. В этом разделе можно написать код своей функции, обращение к которой будет в программе выше. Здесь же можно сделать вывод данных. **Это необязательная часть программы.**

Загрузим получившийся код в плату и помигаем светодиодом. Плата уже подключена, в окне редактора кода открыт скетч по умолчанию (рис.8). Сохраним наш скетч под любым именем, например, «Svetodiod», через меню «Сохранить как». Вместо пустого шаблона вставим код из пункта 4 и нажмем кнопку «Проверить» на панели инструментов (под кнопкой «Файл», круглая кнопка с пиктограммой галочки). Arduino IDE скомпилирует код и если все в порядке, выдаст сообщение «Компиляция завершена» (рисунок 10). Если в коде есть ошибки, то в нижней части редактора кода появятся сообщения об ошибках на английском языке оранжевым цветом. Если ошибок нет, то можно загружать код в плату. Для этого необходимо нажать кнопку «Загрузка» - вторая слева круглая кнопка с пиктограммой стрелки вправо (рисунок 11). После загрузки скетча светодиод на плате начнет равномерно мигать. Попробуйте изменить в коде одно из времен задержки и загрузить код заново. Что изменилось?

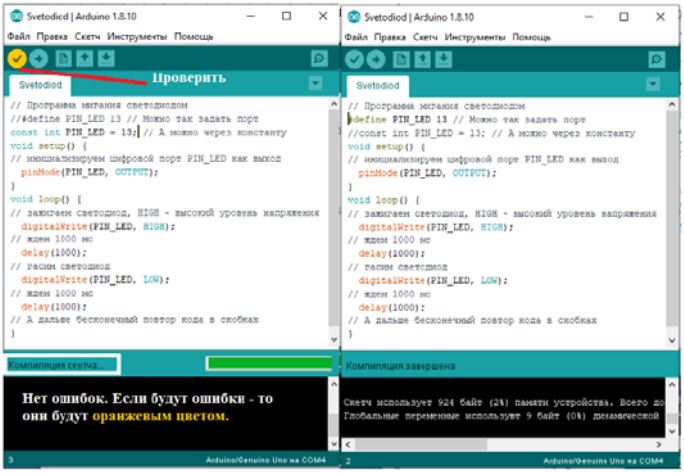


Рис. 10. Проверка (компиляция) скетча в Arduino IDE

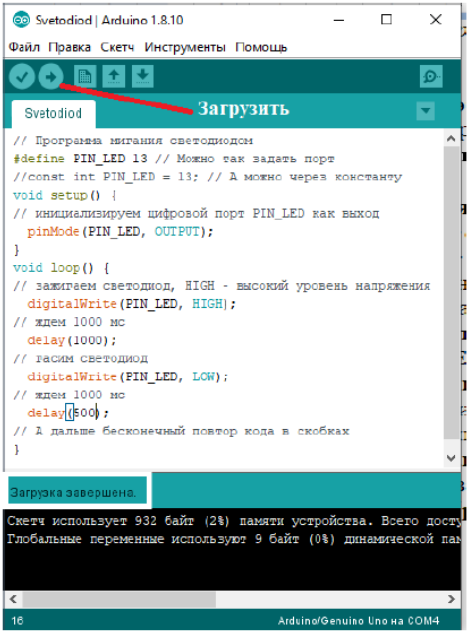


Рис. 11. Загрузка скетча в Arduino IDE

6. РАБОТА С АЦП

Аналого-цифровой преобразователь (АЦП) – это такое устройство, которое преобразует мгновенное значение непрерывной аналоговой величины в целое число. Предназначено для преобразования непрерывных аналоговых величин в цифровой код. Представьте, что на вход АЦП подано напряжение в диапазоне от 0 до 5 В. Уровню напряжения 0 В на входе АЦП будет соответствовать число 0 на выходе, а уровню 5 В – число 1023. Число 1023 появилось как степень двойки: 210=1024 (от 0 до 1023). Число 10 – это разрядность АЦП. В Arduino Uno встроен именно 10-разрядный АЦП. В совместимых платах может использоваться 12-разрядный АЦП. Узнать какой АПЦ можно в документации к плате или опытным путем. Другая важная характеристика АЦП – уровень опорного напряжения. С этим уровнем сравнивается входной сигнал. Входной сигнал не может быть больше опорного напряжения. В Arduino Uno опорное напряжение по умолчанию равно 5 В. В платах с АЦП разрядностью 12 бит опорное напряжение равно 3.3 В. Более подробно прочитать про принцип работы АЦП можно в [7].

В Arduino Uno к АЦП подключены аналоговые входы A0-A5. Подадим на этот вход напряжение 5 В и считаем его функцией analogRead() [8]. Результат выведем в монитор порта. Скетч и монитор порта показаны на рисунке 12. Как видно на рисунке, в монитор порта выводится некоторое число. Это число – номер уровня из диапазона от 0 до 1023 в АЦП, в который попало мгновенное значение напряжения на входе A0. Так как мы ничего к порту A0 не подключали, значит это помехи или наводки от других устройств. Нас интересует не номер уровня, а напряжение в вольтах. Поэтому уровень необходимо преобразовать в напряжение. Максимальное напряжение, которое мы можем измерить, равно опорному – 5 В. А количество уровней на выходе АЦП – 1024. Отношение 5 В/1024 показывает, сколько вольт приходится на каждый уровень (4.88 мВ/уровень). Умножим на этот коэффициент переменную sensorValue в коде и получим в мониторе порта значения в вольтах (рисунок 13).

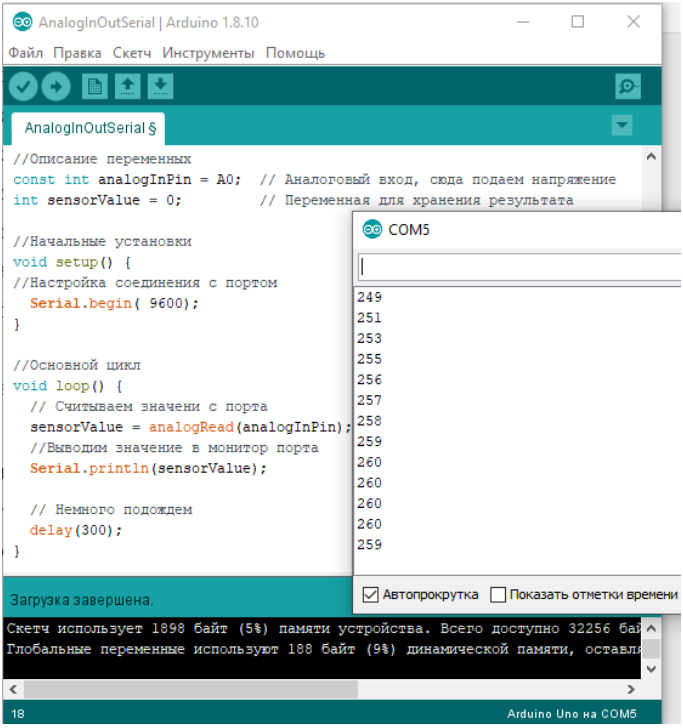


Рис. 12. Скетч и монитор порта

Чтобы повысить точность представления измеренного результата переменную sensorValue зададим с типом float.

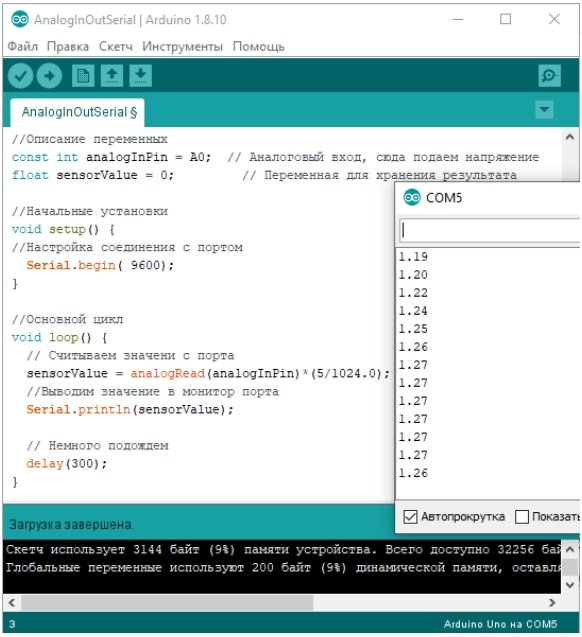


Рис. 13. Введение клэффициента (5 В/1024) в переменную sensorValue для получения в мониторе порта значения в вольтах

Теперь в мониторе порта мы видим напряжение в вольтах. Именно такая помеха наводится на вход A0 без принят специальных мер. Подключите ко входу A0 провод и потрогайте его руками. Напряжение будет изменяться еще сильнее. Проведем ряд экспериментов с подключением, а результат выведем в Плоттер по последовательному соединению (монитор порта нужно перед этим закрыть).

На рисунке 14 показан Плоттер по последовательному соединению. Цифрами на графике обозначены входные сигналы, поданные на порт A0:

1 – просто провод в качестве антенны
2 – провод подключен к разъему GND на плате
3 – провод подключен к разъему 3.3V на плате
4 – провод подключен к разъему 5V на плате
5 – провод приложен к кабелю питания компьютера

Объясните поведение графика и рассчитайте качественно период сигнала на участках 1 и 5 (подсказка: в Мониторе порта можно включить опцию «Показать отметки времени»).

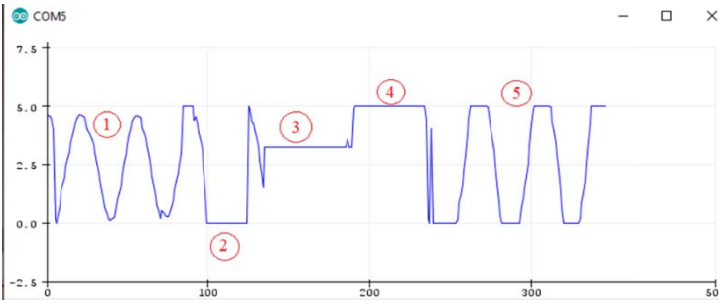


Рис. 14. Плоттер по последовательному соединению

7. ПРОГРАММИРОВАНИЕ RC-ЦЕПИ В ARDUINO

Теперь соберем на макетной плате интегрирующую RC-цепь. Номиналы цепи: емкость (электролитический конденсатор) 22 мкФ, сопротивление – 10 кОм. На вход подадим вручную 5 В, выход цепи (точка между сопротивлением емкостью) подключим ко входу A0. Задержку в скетче уменьшим до 2 мс, в остальном скетч остался неизменным. Схема подключения приведена на рисунке 15.

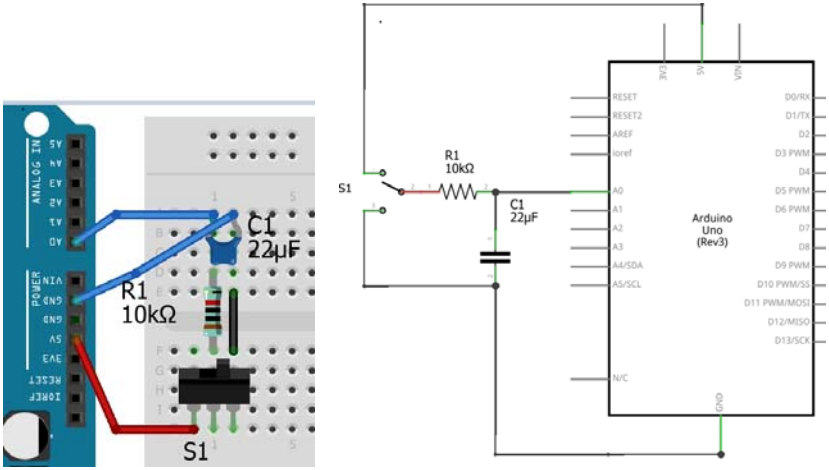


Рис. 15 Схема подключения RC-цепи. Нарисован во Fritzing

На рисунке 16 показан переходной процесс (заряд конденсатора). По Плоттеру оценить временные характеристики не удастся. Поэтому проведем опыт еще раз (предварительно разрядив конденсатор) и выведем результат в Монитор порта. В Мониторе порта обязательно включим отметки времени. Скопируем содержимое монитора порта в Excel и вычислим постоянную времени цепи. По расчетам в Mathcad мы знаем, что $\tau_ц$=220 мс, а напряжение на конденсаторе в этот момент $U_C(\tau_ц)$=3.161 В.

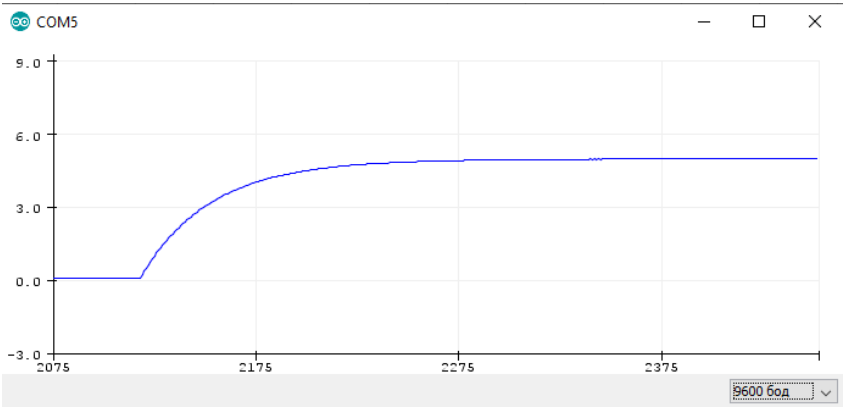


Рис.16. Переходной процесс

На рисунке 17 показан фрагмент данных из Монитора порта. Данные скопированы в Excel и размещены таким образом, чтобы посчитать постоянную времени цепи. Как видно из рисунка, темп поступления данных в монитор порта составляет 5-6 мс. А в нашем скетче задержка была 2 мс. Поэтому следует сделать вывод о том, что необходимо учитывать временные задержки на выполнение кода в плате. Также из рисунка видно, что ближайшая точка к значению 3.161 В – 3.17 В. Разница по времени между моментом началом процесса и этой точкой – 206 мс. Неточность составляет примерно 10-12 процентов. Такой результат можно считать удовлетворяющим условиям нашего опыта. Для получения более точного результата необходимо откалибровать показания АЦП в скетче, а также точно измерить номиналы цепи.

37	14:26:46.841 -> 3.12		
38	14:26:46.841 -> 3.17		14:26:46.635 -> 0.10
39	14:26:46.841 -> 3.22	Пост. Цепи	
40	14:26:46.841 -> 3.27	206	
41	14:26:46.876 -> 3.32		

Рис.17. Данные из Монитора порта

Следующим шагом необходимо сравнить теоретические (рассчитанные в MathCad) и практические (измеренные в RC-цепи на базе Arduino) значения напряжения на конденсаторе в процессе заряда и разряда. Для этого необходимо значения времени в таблице MS Excel подготовить для построения в одном масштабе на графике в MathCad.

Сначала заменим знак точки на запятую для этого: выделим столбец со значениями, перейдем во вкладку Главная\ Найти и выделить, выбираем Заменить, в открывшееся окно вписываем что мы хотим заменить и нажимаем Заменить все (рис. 18).

Теперь разделим по столбцам значения времени и напряжения. Выделим столбец с данными, перейдем во вкладку Данные\Текст по столбцам, выбираем формат данных фиксированной ширины (рис. 19), нажимаем далее, устанавливаем на образце разбора данных стрелки по столбцам (рис. 20), нажимаем далее, в образце разбора данных выбираем столбец (рис. 21), который не требуется выводить и ставим флажок пропустить столбец (при необходимости повторить с другими столбцами), нажимаем готова и подтверждаем замену данных.

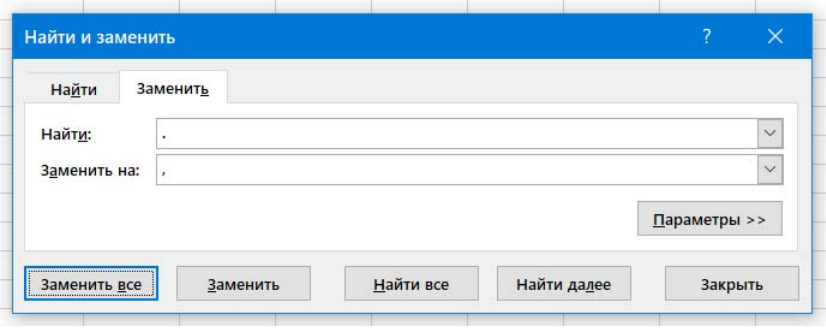


Рис. 18 Замена точки на запятую

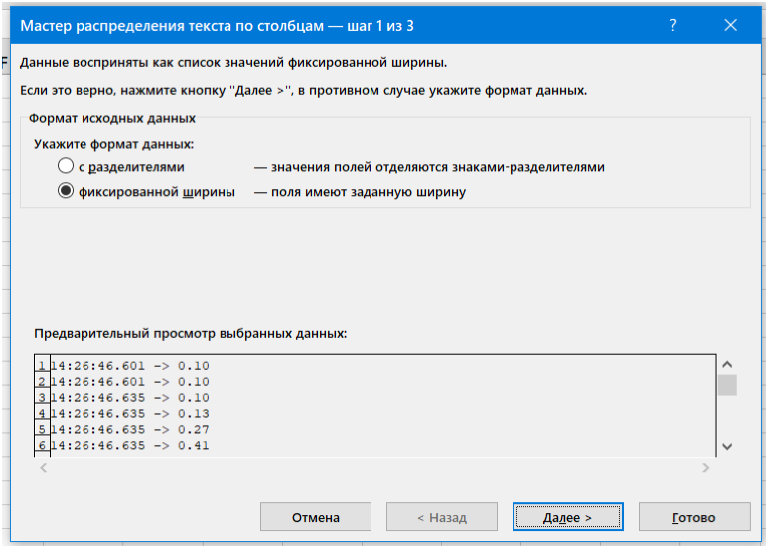


Рис. 19 Выбор фиксированной ширины для разделения по столбцам

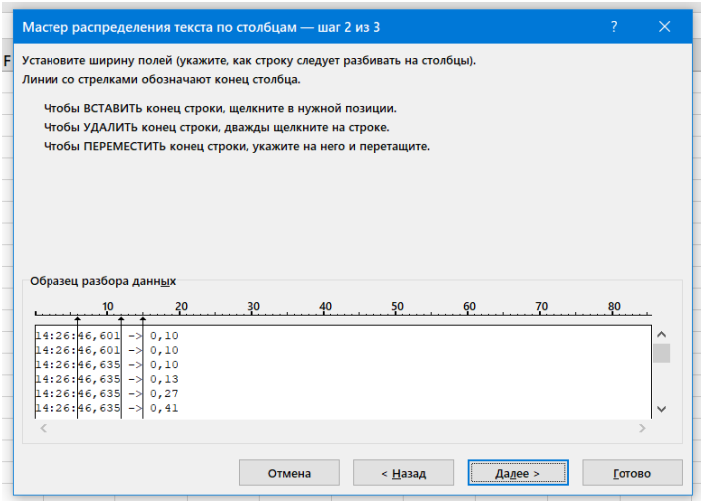


Рис. 20 Установка ширины полей

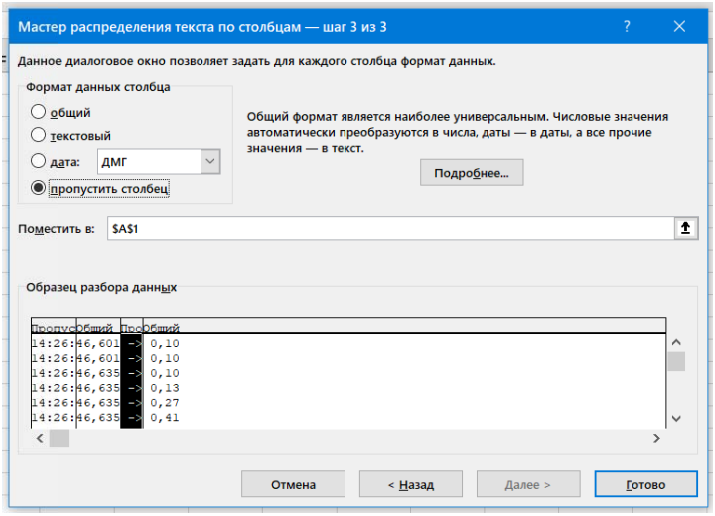
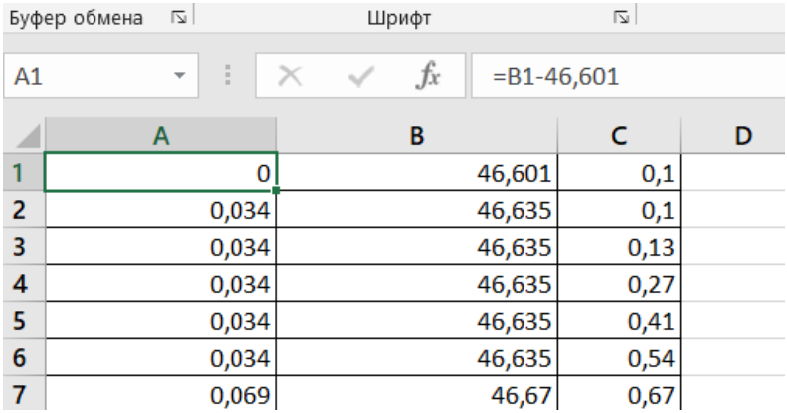


Рис. 21 Пропуск столбцов для вывода на лист Excel

В результате преобразований на листе получится два столбца со значениями времени и напряжения, но время все еще не подходит для сравнения с теоретическими данными, так как начинается не с нуля. Добавим столбец и примем начальное значение времени за ноль, вычитая это значение из следующих (рис. 22).



	A	B	C	D
1	0	46,601	0,1	
2	0,034	46,635	0,1	
3	0,034	46,635	0,13	
4	0,034	46,635	0,27	
5	0,034	46,635	0,41	
6	0,034	46,635	0,54	
7	0,069	46,67	0,67	

Рис. 22 Изменение времени

Перенесем таблицу в MathCad с помощью команды READEXCEL(“полный путь с именем файла“) (рис.23) и выделим столбцы времени и напряжения для построения на общем графике (рис. 24).

$$\mathbf{M} := \mathbf{READEXCEL}(\text{"C:\Users\Admin\Desktop\ACP.xlsx"})$$

$\mathbf{M} =$

	0	1	2
0	0	0.1	46.601
1	0.034	0.1	46.635
2	0.034	0.13	46.635
3	0.034	0.27	46.635
4	0.034	0.41	46.635
5	0.034	0.54	46.635
6	0.069	0.67	46.67
7	0.069	0.8	46.67
8	0.069	0.91	46.67
9	0.069	1.04	46.67
10	0.069	1.15	46.67
11	0.069	1.25	46.67
12	0.103	1.36	46.704
13	0.103	1.47	46.704
14	0.103	1.57	46.704
15	0.103	1.67	...

$$\mathbf{tArduino} := \mathbf{M}^{\langle 0 \rangle}$$

$$\mathbf{UcArduino} := \mathbf{M}^{\langle 1 \rangle}$$

Рис. 23 Перенос таблицы из MS Excel в MathCad и выделение столбцов времени и напряжения

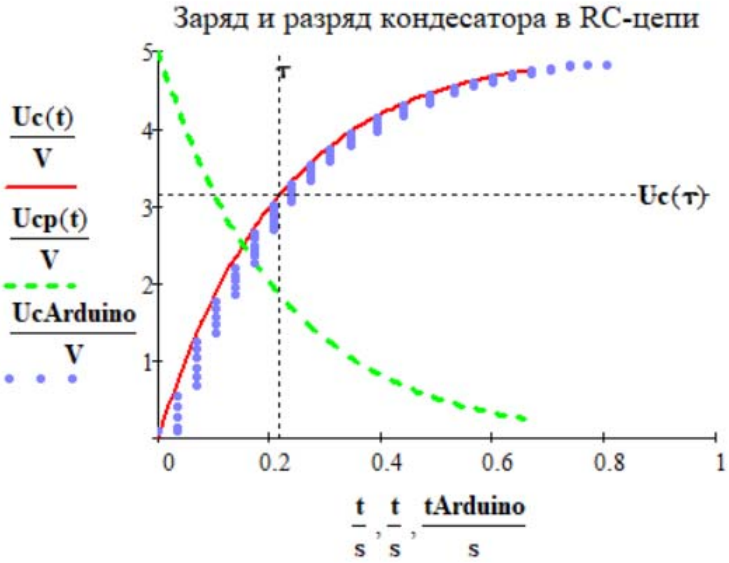


Рис. 24 Сравнение теоретических и практических значений напряжения на конденсаторе в процессе заряда и разряда

ЗАКЛЮЧЕНИЕ

Курсовую работу необходимо выполнять по принципу «от простого – к сложному», двигаясь маленькими шажками к цели работы. Сначала необходимо добиться устойчивого выполнения кода примеров. Разобравшись в каждой строчке кода примера, переходить к его адаптации к заданию. Следует максимально полно использовать циклы и функции.

Оформление работы должно соответствовать государственным требованиям, предъявляемым к оформлению документации [9-12]. Внимательно прочитайте следующие строки и следуйте им при оформлении курсовой работы.

Содержание курсовой работы отражает ход мысли и способности студента. Оформление работы демонстрирует отношение студента к процессу обучения, преподавателю и лично к себе. Все разделы работы последовательно раскрывают этапы достижения цели. В работе содержится только информация, непосредственно связанная с этапами работы. Все иллюстрации раскрывают содержание выполненных действий. Текстовые пояснения помогают студенту отвечать на вопросы на защите, а не порождают новые уточняющие вопросы. Выводы по работе содержат личные впечатления студента и констатируют достижение цели работы, а не перечисляют очевидные факты. Переписывать теорию из учебника и интернета в курсовую работу не нужно. Из текста пояснительной записки должно быть видно, какие новые навыки приобретены студентом. Время, затраченное на выполнение курсовой работы, не является критерием оценки. Удачи!

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Калашников С.Г. Электричество: Учебн. пособие. – 6-е изд., ст-реот. – М.: ФИЗМАЬЛИТ, 2008. – 624 с.

2. Фролов Ю.М., Шелякин В.П. Регулируемый асинхронный электропривод: Учебное пособие. – 2-е изд., стер. – СПб.: Издательство «Лань», 2018. -464 с.: ил.

3. Теоретические основы электротехники: в 3-х т. Учебник для вузов. Том 2. – 4-е изд./К.С. Демирчян, Л.Р. Нейман, Н.В. Коровкин, В.Л. Чечурин – СПб.: Питер, 2003. -576 с.: ил.

4. Блум Д. Изучаем Arduino: инструменты и методы технического волшебства / Джереми Блум; [перевод с английского В. Петина]. - Санкт-Петербург: БХВ-Петербург, 2017. - 336 с.

5. Монк С. Программируем Arduino: профессиональная работа со скетчами / Саймон Монк; [пер. с англ. А. Киселев]. – Санкт-Петербург [и др.]: Питер, 2017. - 272 с.

6. Arduino сайт на русском для начинающих мастеров ардуино [Электронный ресурс]. – URL: https://arduinomaster.ru/#1508255432022-58d0f83f-1be8 (дата обращения 22.01.2020).

7. Топольский В.Б. Схемотехника аналого-цифровых преобразователей. Учебное издание Москва: ТЕХНОСФЕРА, 2014. – 288 с.

8. Описание функции analogRead() [Электронный ресурс]. – URL: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/ (дата обращения 22.01.2020).

9. SPI Arduino - подключение устройств к платам ардуино [Электронный ресурс]. – URL: https://arduinomaster.ru/datchiki-arduino/podklyuchenie-spi-arduino/ (дата обращения 22.01.2020).

10. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Стандартинформ, 2018. – 33 с.

11. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. – М.: Стандартинформ, 2007. – 32 с.

12. ГОСТ Р 7.0.5-2008. Библиографическая ссылка. Общие требования и правила составления. – М.: Стандартинформ, 2008. – 23 с.

СОДЕРЖАНИЕ

ИНФОРМАТИКА

ПРОГРАММИРОВАНИЕ ПЕРЕХОДНЫХ ПРОЦЕССОВ В ИНТЕГРИРОВАННОЙ СРЕДЕ РАЗРАБОТКИ ARDUINO

Методические указания к курсовым работам для студентов специальности 21.05.04

Сост.: *Е.Г. Водкайло, О.В. Косарев*

Печатается с оригинал-макета, подготовленного кафедрой информатики и компьютерных технологий

Ответственный за выпуск *Е.Г. Водкайло*

Лицензия ИД № 06517 от 09.01.2002

Подписано к печати 30.06.2020. Формат 60×84/16.
Усл. печ. л. 2,1. Усл.кр.-отт. 2,1. Уч.-изд.л. 1,9. Тираж 50 экз. Заказ 452.

Санкт-Петербургский горный университет
РИЦ Санкт-Петербургского горного университета
Адрес университета и РИЦ: 199106 Санкт-Петербург, 21-я линия, 2