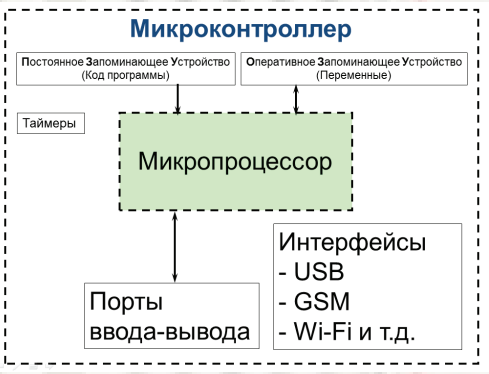
Министерство образования и науки Российской Федерации
Тольяттинский государственный университет
Институт энергетики и электротехники
Кафедра «Промышленная электроника»

Е.С. Глибин
В.И. Чепелев

РАЗРАБОТКА ИЗМЕРИТЕЛЬНЫХ СИСТЕМ С ПРИМЕНЕНИЕМ КОНТРОЛЛЕРОВ ARDUINO



ISBN 978-5-8259-0952-3

УДК 621.317.75(075.8)
ББК 31.221.043-5я73

Рецензенты:
засл. работник высшей школы РФ, д-р пед. наук,
канд. техн. наук, профессор кафедры «Сервис технических
и технологических систем» Поволжского государственного
университета сервиса *Н.П. Бахарев*;
д-р техн. наук, доцент Тольяттинского государственного
университета *В.П. Певчев*.

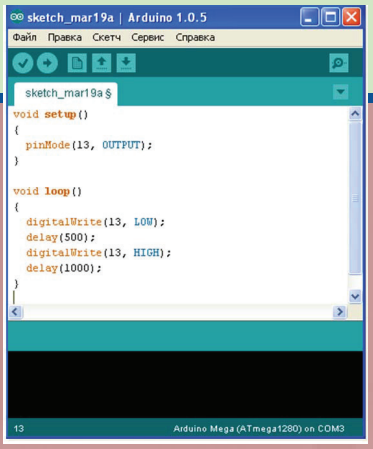
Глибин, Е.С. Разработка измерительных систем с применением контроллеров Arduino : электрон. учеб.-метод. пособие / Е.С. Глибин, В.И. Чепелев. – Тольятти : Изд-во ТГУ, 2016. – 1 оптический диск.

Учебно-методическое пособие содержит задания по работе с электронными осциллографами, генераторами и по макетированию устройств, работающих с датчиками информации, а также описание использования среды программирования Arduino для разработки и отладки программного обеспечения измерительных систем на базе микроконтроллеров AVR-архитектуры.

Предназначено для студентов направлений подготовки бакалавров 210100.62 (11.03.04) «Электроника и наноэлектроника», 140400.62 (13.03.02) «Электроэнергетика и электротехника» при изучении ими дисциплины «Электроизмерительные приборы и датчики информации».

Текстовое электронное издание

Рекомендовано к изданию научно-методическим советом Тольяттинского государственного университета.



Редактор *Т.Д. Савенкова*
Корректор *Л.Н. Ворожцова*
Технический редактор *Н.П. Крюкова*
Компьютерная верстка: *Л.В. Сызганцева*
Художественное оформление,
компьютерное проектирование: *Г.В. Карасева*

Издательство Тольяттинского государственного университета
445020, г. Тольятти, ул. Белорусская, 14,
тел. 8 (8482) 53-91-47, www.tltsu.ru

Содержание

ВВЕДЕНИЕ

Современное развитие автоматизированного производства требует применения электронных устройств для управления технологическим процессом и определения параметров производимой продукции. При переходе от ручного к автоматизированному производству датчики должны сообщать автоматизированной системе о положении обрабатываемых деталей в пространстве, температуре рабочей среды, токах и напряжениях двигателей и т. д. Повышение качества производимой продукции осуществляется с помощью контроля параметров процесса измерительными системами на базе микропроцессоров.

Целью изучения дисциплины является формирование у студентов профессиональных компетенций (ПК-1, ПК-3, ПК-5, ПК-9, ПК-10, ПК-33, ПК-35, ПК-36), необходимых для разработки и эксплуатации устройств с использованием электронных датчиков.

Задачи:

- ✓ научить студентов разрабатывать электронные схемы с использованием датчиков промышленной электроники различных типов;
- ✓ развить у студентов навыки разработки алгоритмов функционирования измерительных систем;
- ✓ развить у студентов навыки обращения с измерительной техникой для анализа работы реальных систем; умение принимать решения при поиске и устранении неисправностей;
- ✓ выработать умения создавать устройства на основе современной элементной базы с применением микропроцессорной техники; написания программного обеспечения для опроса датчиков; обработки полученной информации и её передачи по стандартным каналам данных.

В результате изучения дисциплины приобретаются

– знания:

- основ измерения физических величин;
- методов и принципов измерений;
- структуры современных измерительных систем, созданных на базе микропроцессоров;
- устройства и принципа работы основных датчиков промышленной электроники: температуры, тока и напряжения, магнитного поля, давления, влажности, расхода жидкостей и газов, расстояния и приближения;

– умение правильно выбирать датчики первичной информации для решения поставленной задачи;

– навыки:

- проектирования электронных систем сбора данных;
- проверки электронных узлов на наличие работоспособности и анализа возможных неисправностей в их работе с последующим их устранением;

– компетенции:

- способность представлять адекватную современному уровню знаний научную картину мира на основе знания основных положений, законов и методов естественных наук и математики (ПК-1);
- готовность учитывать современные тенденции развития электроники, измерительной и вычислительной техники, информационных технологий в своей профессиональной деятельности (ПК-3);
- способность владеть основными приемами обработки и представления экспериментальных данных (ПК-5);
- способность осуществлять сбор и анализ исходных данных для расчета и проектирования электронных приборов, схем и устройств различного функционального назначения (ПК-9);
- готовность выполнять расчет и проектирование электронных приборов, схем и устройств различного функционального назначения в соответствии с техническим заданием с использованием средств автоматизации проектирования (ПК-10);
- способность владеть современными методами расчета и проектирования электронных приборов и устройств, способность к восприятию, разработке и критической оценке новых способов их проектирования (ПК-33);
- способность идентифицировать новые области исследований, новые проблемы в сфере физики, проектирования, технологии изготовления и применения электронных приборов и устройств (ПК-35);
- способность разрабатывать модели исследуемых процессов, материалов, элементов, приборов и устройств электронной техники (ПК-36).

Лабораторная работа 1

ИЗУЧЕНИЕ ЭЛЕКТРОННОГО ОСЦИЛЛОГРАФА И ГЕНЕРАТОРОВ ЭЛЕКТРИЧЕСКИХ СИГНАЛОВ

Цель работы – научиться работать с электронным осциллографом, настраивать режимы работы генераторов для создания периодических электрических сигналов произвольной формы.

Программа работы

1. Подготовить аппаратное и программное обеспечение лабораторного стенда к работе.
2. Научиться задавать форму электрического сигнала с помощью генератора.
3. Снять осциллограмму генерируемого сигнала, предварительно зафиксировав ее с помощью триггера.
4. Измерить максимальное, среднее и среднеквадратичное значения напряжения на генераторе.
5. Рассчитать среднеквадратичное значение сигнала при заданных параметрах, сравнить его с отображаемым программой и заданным (если показывается) в окне настроек генератора.
6. Повторить пункты 2–5 для второго сигнала при использовании встроенного генератора, для третьего – при использовании внешнего генератора.
7. Оформить и защитить отчет о лабораторной работе.

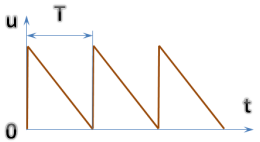
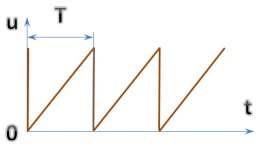
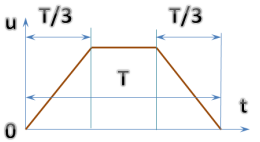
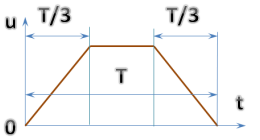
Варианты заданий

Для каждого из трех сигналов, приведенных в табл. 1, необходимо настроить соответствующий генератор, зафиксировать осциллограмму полученного сигнала, измерить его параметры согласно программе работы.

В отраслевом стандарте связи ***ОСТ 45.159-2000 «Отраслевая система обеспечения единства измерений. Термины и определения»*** термин «амплитуда» определяется как наибольшее значение, которое принимает какая-либо величина, изменяющаяся по гармоническому закону.

Таблица 1

Формы и параметры генерируемых сигналов

№	№ сигнала		
	1	2	3
1	Синусоида, амплитуда 1,5 В, частота 2 кГц, постоянная составляющая 1 В	 размах 2,2 В, частота 10 кГц	Униполярный меандр, размах 1 В, частота 5 кГц
2	Двухполярный меандр, минимальное значение –1,5 В, максимальное значение 1 В, частота 15 кГц	 размах 1,8 В, частота 5,5 кГц	Синусоида, амплитуда 2,5 В, частота 900 Гц
3	Симметричный треугольный сигнал, размах 3 В, частота 500 Гц	 размах 3 В, частота 1400 Гц	Прямоугольный сигнал, размах 2 В, частота 7,5 кГц, постоянная составляющая 1,5 В
4	Прямоугольный сигнал, размах 4 В, частота 1,5 кГц, постоянная составляющая 2 В, скважность – 3	 размах 2 В, частота 3600 Гц	Симметричный треугольный сигнал, размах 3 В, частота 200 Гц

Термин «амплитуда» следует отличать от терминов, применимых к любым сигналам.

Максимальное и минимальное значения сигнала – наибольшее и наименьшее мгновенные значения сигнала на протяжении заданного интервала времени.

Размах сигнала – разность между максимальным и минимальным значениями сигнала на протяжении заданного интервала времени.

Согласно ***ГОСТ 16465-70 «Сигналы радиотехнические измерительные. Термины и определения»*** не следует в общем случае заменять термин «максимальное значение сигнала» термином «амплитуда». ГОСТ отдельно регламентирует использование таких терминов, как «амплитуда импульса» или «амплитуда прямоугольного сигнала».

Приведем некоторые термины, встречающиеся в задании, определения которых приведены в ГОСТ 16465–70.

Постоянная составляющая сигнала – среднее значение сигнала

$$\bar{x} = \lim_{Ty \to \infty} \frac{1}{Ty} \int_0^{Ty} \bar{x}(t)dt ,$$

где *Ty* – интервал времени усреднения.

Скважность – отношение периода прямоугольного сигнала к длительности импульса.

При скважности 2 (длительность импульса равна длительности паузы) прямоугольный сигнал называется меандром. Меандры бывают униполярными (величина напряжения принимает значения одного знака) и двухполярными.

Следует отметить, что в программном обеспечении зарубежного производства определения ряда применяемых терминов могут не соответствовать ГОСТу – воспользуйтесь инструкцией и справкой!

Номер варианта соответствует номеру бригады, назначенной на вводном занятии. Если количество бригад больше 4, то пятая выполняет задание первого варианта, шестая – второго и т. д.

Содержание отчета

1. Титульный лист.
2. Цель работы, программа работы и задание ТОЛЬКО для своего варианта.
3. Текстовое описание выполняемых действий в процессе подготовки генератора и осциллографа к работе.
4. Описание значений выбираемых настроек программы или установки элементов управления внешнего прибора.
5. Осциллограммы всех сигналов задания с ОБЯЗАТЕЛЬНЫМ указанием масштаба и единиц измерения по осям, а также нулевого уровня.
6. Результаты измерений среднеквадратичного, среднего и максимального значений напряжений для каждого сигнала.

7. Результаты расчета среднеквадратичных значений для каждого сигнала.
8. Выводы о работе.

Указания к выполнению работы

Предварительно самостоятельно ознакомьтесь с инструкцией по работе со стендом и его программным обеспечением, используя материалы, полученные на вводном занятии.

Включите компьютер, а также лабораторный стенд, генератор и осциллограф с помощью соответствующих выключателей. Убедитесь, что лампочки «Сеть», «Генератор» и «Осциллограф» горят. После этого запустите программу PCLAB2000SE с настройками по умолчанию. Появится окно, подобное изображенному на рис. 1. Если при запуске возникают ошибки, смените учетную запись на администраторскую.

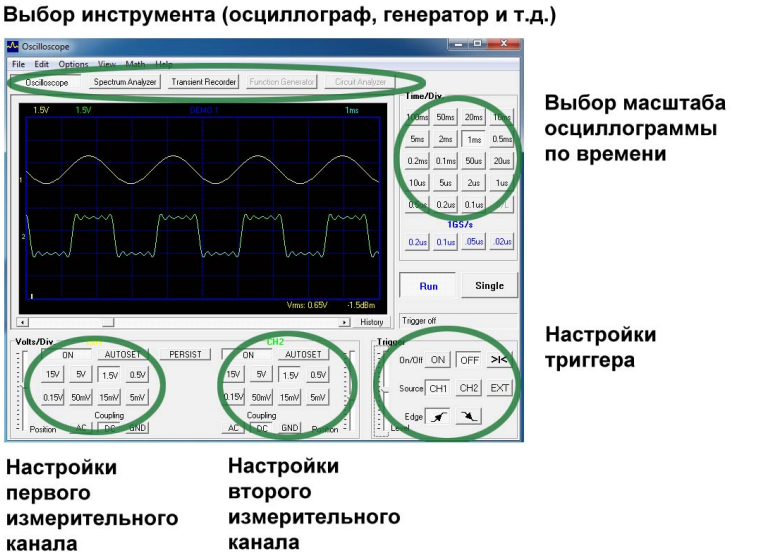


Рис. 1. Главное окно программы-осциллографа PCLAB2000SE

Соедините выход генератора со входом первого канала осциллографа. Нажатием кнопки **Function Generator** откройте окно настройки сигнала (рис. 2) и установите параметры согласно заданию.

ВНИМАНИЕ! Если закрыть окно настроек сигнала, работа генератора прекращается. Другими словами, если сначала вы зададите сигнал требуемой формы, а затем нажатием на **X** закроете окно, так как оно закрывает окно осциллографа и мешает работе, сигнал пропадёт. Просто переместите окно в свободную область экрана.

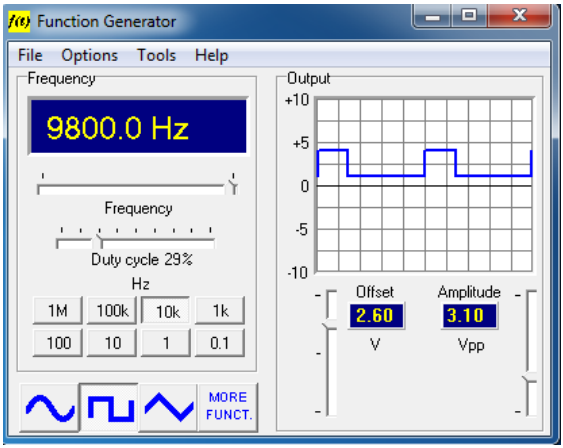


Рис. 2. Окно настроек генератора

Вручную (то есть без использования функции AUTOSET) настройте осциллограф, чтобы получить устойчивое изображение сигнала на экране: режим измерения, масштабы по осям, триггер (рис. 3). Для получения стабильного изображения все осциллографы содержат систему, называемую схемой синхронизации, которую в зарубежной литературе не совсем корректно часто называют триггером. Назначение схемы синхронизации – задерживать запуск развертки до тех пор, пока не произойдет некоторое событие. Система синхронизации имеет как минимум три важные настройки:

1) уровень запуска (задается бегунком trigger): задаёт напряжение исследуемого сигнала, при достижении которого запускается развёртка;
2) источник исследуемого сигнала: первый или второй канал, либо внешний;
3) тип запуска: по фронту или по спаду.

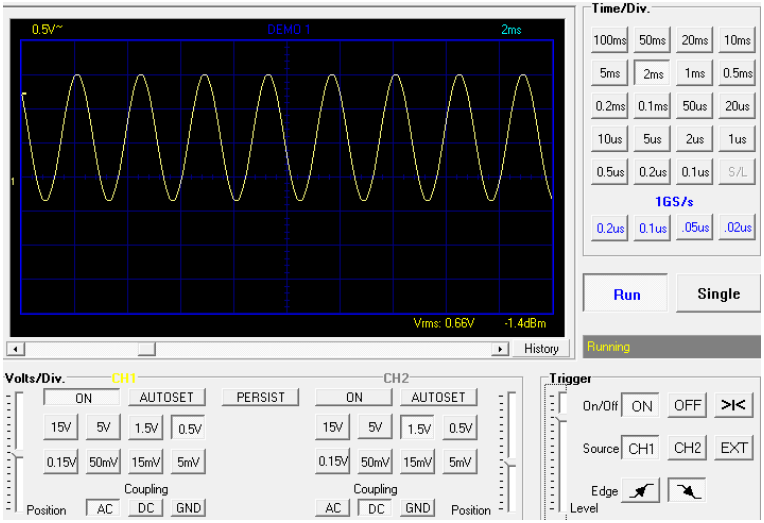


Рис. 3. Настройка триггера

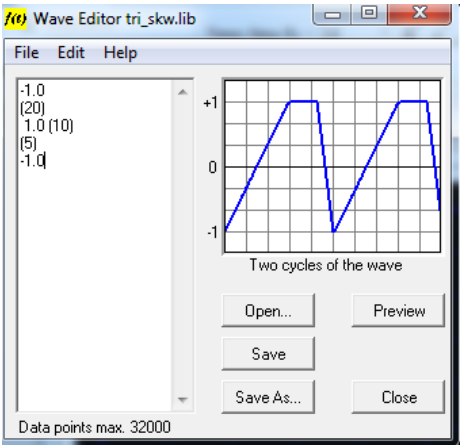


Рис. 4. Задание периодического сигнала произвольной формы

Правильная настройка этих органов управления обеспечивает запуск развёртки всегда в одном и том же месте сигнала, поэтому изображение сигнала на осциллограмме выглядит стабильным и неподвижным.

Сохраните, сфотографируйте или зарисуйте полученную осциллограмму, запишите масштабы, RMS (**R**oot **M**ean **S**quare, среднеквадратичное значение), измерьте амплитуду. Дополнительные параметры сигнала могут быть отображены с использованием пункта меню View → Waveform Parameters...

Для того чтобы создать сигнал формы пилы и трапеции, воспользуйтесь меню Tools → Wave Editor (рис. 4). Для образца используйте кривые из встроенной библиотеки.

Среднеквадратичное значение напряжения (иногда называемое действующим или эффективным) удобно для практических расчетов. На линейной активной нагрузке (например, лампа накаливания) оно совершает точно такую же работу, как и равное ему постоянное напряжение:

$$U = \sqrt{\frac{1}{T}\int_0^T u^2(t)dt},$$

где T – период напряжения.

Для расчета среднеквадратичного значения каждого сигнала $u(t)$ постройте кривую $u^2(t)$, определите площадь под ней и разделите на период.

Вопросы для защиты лабораторной работы

Для ответа на вопросы к этой и последующим лабораторным работам воспользуйтесь конспектом лекций.

1. Что такое измерение? Зачем необходимы измерения в промышленности и науке?
2. Что такое электрический сигнал? Что такое датчик? Какие задачи решают датчики в промышленности?
3. Какие типы измерений вы знаете?
4. Поясните термины «единица», «эталон». Какие системы единиц вам известны?
5. Что может являться эталоном физической величины? Приведите примеры.
6. Назовите причины использования датчиков.
7. Каково различие между контактным и бесконтактным устройством измерения?

Лабораторная работа 2

ЗНАКОМСТВО С ПРОГРАММИРОВАНИЕМ МИКРОКОНТРОЛЛЕРОВ

Цель работы – научиться составлять, компилировать и загружать в микроконтроллер простейшие программы на Arduino.

Программа работы

1. Подготовить аппаратное и программное обеспечение (установить драйверы) лабораторного стенда к работе.
2. Изучить краткие теоретические сведения.
3. Ввести, скомпилировать, загрузить в микроконтроллер и отладить программу мигающего светодиода, представленную в кратких теоретических сведениях.
4. Разработать программу согласно заданию и номеру бригады.
5. Оформить и защитить отчет о лабораторной работе.

Краткие теоретические сведения

Сегодня зачастую электронные устройства, включающие в свой состав датчики, создаются с применением микроконтроллеров. Микроконтроллер представляет собой электронную схему, которая может выполнять самые разные задачи: контроль температуры в помещении, управление автомобильным двигателем или промышленным роботом и многие другие. Весьма упрощенная структурная схема микроконтроллера показана на рис. 5. Обратите внимание: так как микроконтроллер состоит из модулей общего назначения (микропроцессор, память, модули ввода – вывода и т. д.), то, взглянув на схему, нельзя сказать, какую задачу он сейчас выполняет: возможно, следит за состоянием автоматизированной линии, а может быть, загружает интернет-страницу в планшете. Функционирование микроконтроллера определяется программой, загруженной в него.

Программа представляет собой последовательность команд, которые выполняет микропроцессор. Команды определяются выбранным языком программирования, которых достаточно много (рис. 6, 7): низкоуровневые (ассемблеры), высокоуровневые

общего назначения (Бейсик, Паскаль, Си и Си++), высокоуровневые специального назначения (Simatic Step 7, PHP, Arduino, Algorithm Builder).

В этой и последующих лабораторных работах будет использоваться Arduino, практически идентичный языку программирования Си++ с очень небольшими особенностями из-за ориентированности на обучение программированию микроконтроллеров. В наши дни многие программы написаны именно на Си++. Этот язык включает практически все конструкции, которые можно встретить в других языках программирования: владея Си++, освоить любой другой можно в кратчайшие сроки. Богатые возможности языка позволяют создавать программы различной сложности – от управления светодиодами до операционных систем (Windows, Android и т. д.). Конечно, он является не самым простым для изучения. Поэтому мы рассмотрим поверхностно лишь те конструкции языка, которые позволят создавать несложные программы для работы с датчиками и выполнять лабораторные работы. В случае затруднений всегда можно воспользоваться литературой из списка, полученного на вводном занятии.

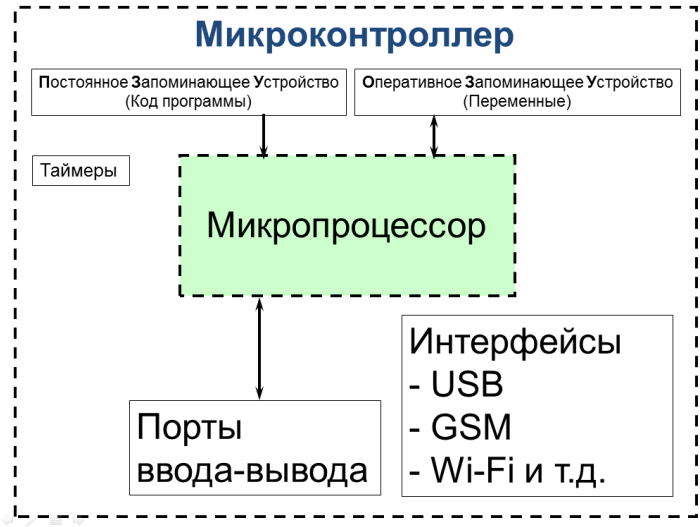


Рис. 5. Упрощенная структура микроконтроллера

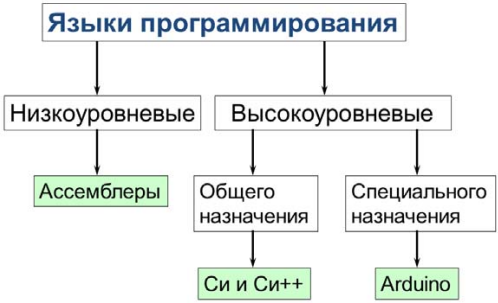


Рис. 6. Классификация языков программирования

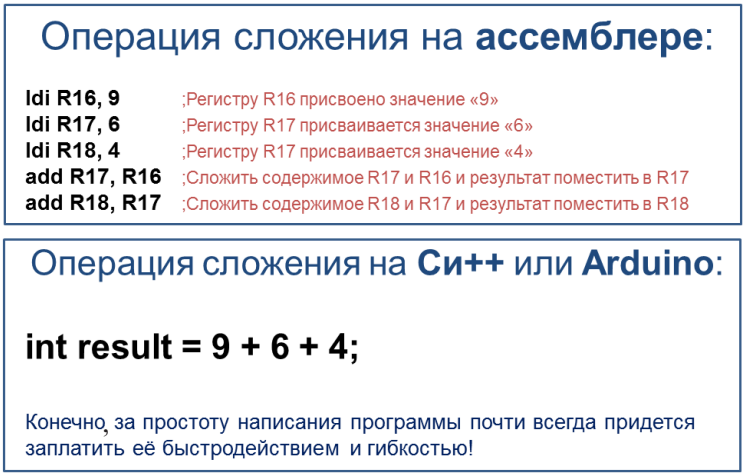


Рис. 7. Сравнение операций сложения на низкоуровневом и высокоуровневых языках программирования

На рис. 8 изображен один из вариантов контроллера Arduino. Контроллер включает следующие основные узлы.

1. Микроконтроллер, часто Atmega1280 или Atmega32U2.
2. Разъем питания 6–12 вольт.
3. Интегральный стабилизатор напряжения LM7805, преобразующий напряжение с разъема питания от 6 до 12 вольт в 5 вольт, которое подается на выводы питания микроконтроллера.
4. USB-разъем (может быть разных типов: обычный, mini, micro).

5. Интерфейсная микросхема FT232 с необходимой обвязкой, организующая связь с персональным компьютером через USB. Стоит отметить, что эта микросхема составляет значительную долю в стоимости всего контроллера. В микроконтроллер Atmega32U2 встроена аппаратная поддержка, USB и FT232 отсутствуют.
6. Обвязка контроллера: фильтры питания аналого-цифрового преобразователя, кварцевый резонатор с конденсаторами, обеспечивающий тактовую частоту процессора, токоограничивающие резисторы и другие защитные элементы.
7. Разъемы для удобного подключения устройств, в том числе датчиков, к выводам микроконтроллера.
8. Индикаторный светодиод, соединенный с 13-м выводом микроконтроллера. Если программа установит напряжение 5 вольт на 13-м выводе, светодиод загорится, если ноль вольт – погаснет.

Рис. 8. Контроллер Arduino DFROBOT MEGA2560

Сердцем платы является микроконтроллер Atmega1280, имеющий 100 выводов. Некоторые выводы имеют строго определенное назначение, например питание. Назначение других определяется программой. Чтобы поместить программу во флеш-память кристалла микроконтроллера, используется USB-интерфейс. На рис. 9 представлен скриншот интегрированной среды разработки Arduino,

с помощью которой программа может быть написана, проверена и скомпилирована, прошита в микроконтроллер и отлажена.

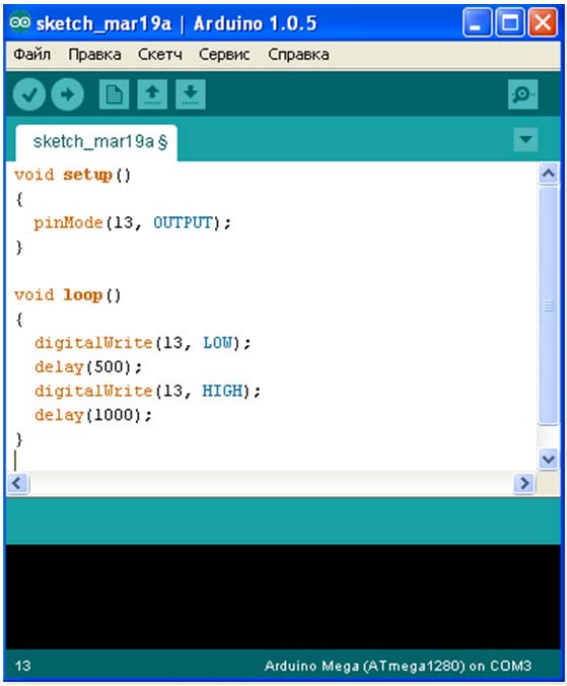


Рис. 9. Среда разработки Arduino

Рассмотрим очень простую программу, работу которой можно наблюдать визуально: мигающий светодиод, подключенный к 13-му выводу. Для ее реализации потребуется выполнить следующие шаги.

Получи́те на бригаду у преподавателя один микроконтроллер с соединительным шнуром. Во избежание повреждений контроллера в результате короткого замыкания ни в коем случае **не размещайте плату на металлическом корпусе компьютера** или металлических поверхностях лабораторного оборудования: нижняя сторона платы имеет выступающие паяные выводы элементов. Подключите контроллер к свободному USB-разъему, загорится индикатор питания. Питание контроллера может осуществляться как через специальный разъем 6–12 В, так и от компьютера через USB.

Проверьте, что контроллер обнаружен компьютером. Щелкните с помощью правой кнопки мыши на иконке «**Компьютер**», выберите «**Свойства**». Вызовите **диспетчер оборудования**. Откройте порты (COM и LPT). В списке должно быть устройство, подключенное к COM-порту с номером больше 1 (последовательный порт COM1 присутствует всегда). Это может быть что-то вроде **Arduino Leonardo (COM3)**. Запомните номер порта и закройте диспетчер.

Если плата подключена к компьютеру впервые, то устройство не будет установлено без соответствующих драйверов. Драйверы расположены в папке с Arduino – для контроллеров на основе FT232 и без нее. В случае необходимости установите их и повторите предыдущий шаг.

Запустите среду разработки. Введите текст программы, представленный ниже. Количество пробелов, символов табуляций, пустых строк между отдельными элементами программы не важно для ее работы и влияет только на визуальное оформление текста. С другой стороны, регистр букв («маленькие» или «большие») имеет значение – не ошибитесь! Обратите внимание на то, что некоторые строчки заканчиваются символом «;».

```
void setup()
{
        pinMode(13, OUTPUT);
}
void loop()
{
        digitalWrite(13, HIGH);
        delay(1000);
        digitalWrite(13, LOW);
        delay(500);
}
```

Щелкните на кнопке «**Проверить**» на панели инструментов Arduino. Если ошибки не найдены, программа успешно скомпилируется, о чем будет выведено соответствующее сообщение в нижней черной части окна.

Теперь необходимо ее загрузить в контроллер. Выполните команду меню Сервис → Плата и выберите название используемой платы или ее аналога. Далее выполните команду меню Сервис →

Последовательный порт и выберите номер используемого порта, который вы запомнили, посмотрев предварительно в диспетчере устройств. Нажмите кнопку «**Загрузить**» на панели инструментов. Через несколько секунд, если не возникли ошибки, программа будет загружена в микроконтроллер, а индикаторный светодиод начнет мигать.

Разберемся в работе программы. Перед рассмотрением ее текста необходимо определиться с некоторыми используемыми далее терминами.

«Пин» – вход или выход, подключённый к чему-либо: светодиоду на выходе или кнопке на входе.

«Цифровой» – значение HIGH или LOW (как вкл/выкл или один/ноль). Пример – состояние выключателя.

Большинство пинов являются цифровыми и могут работать в режиме входа или выхода.

Все выходы в микроконтроллерах, используемых в лабораторных работах, являются цифровыми. То есть на выходе программа может установить единицу (5 В) или ноль (ноль вольт), но установить, например, 2 В не получится.

То же самое при чтении входов: на большинстве выводов нельзя определить промежуточные значения напряжений. Исключение составляют входы аналого-цифрового преобразователя (АЦП), их немного, вынесены на контроллере в отдельный разъем, обозначаются в программах как A0, A1 и т. д. и могут быть использованы для измерения напряжений в диапазоне от ноля до пяти вольт.

Блок-схема алгоритма загруженной программы приведена на рис. 10. Программа состоит из двух частей: инициализации, которая выполняется один раз при включении контроллера, и бесконечного повторения процесса включения и выключения светодиода.

Программы могут состоять из миллионов отдельных команд. Чтобы в них ориентироваться, программы разбивают на отдельные части – подпрограммы. Примером подпрограммы может являться вывод символа на текстовый дисплей, подключенный к контроллеру. Чтобы вывести отдельный символ, необходимо сформировать импульсы на разных входах дисплея. Для этого требуется изучить документацию на дисплей, его интерфейс, команды и т. д.

Не очень удобно обращаться к документации, когда необходимо вывести очередной символ. С другой стороны, можно написать подпрограмму – последовательность команд, предназначенных для многократного выполнения. Подпрограмма будет формировать управляющие сигналы для любого символа, который будет передан ей. Затем можно будет вызывать ее из основной программы, передавая отдельные символы. При этом уже не будет необходимости даже помнить об интерфейсе дисплея, а подпрограмма и основная программа могут быть написаны разными людьми. Можно создать и другую подпрограмму – вывод текста, она будет вызывать подпрограмму вывода символа. В свою очередь, работа с текстовым дисплеем из основной программы упростится. Любую, даже самую сложную программу можно свести к подпрограммам, имеющим разную степень вложенности (как матрешка), образующим иерархическую структуру. Такой подход называется структурным программированием и является основным в языке Си.

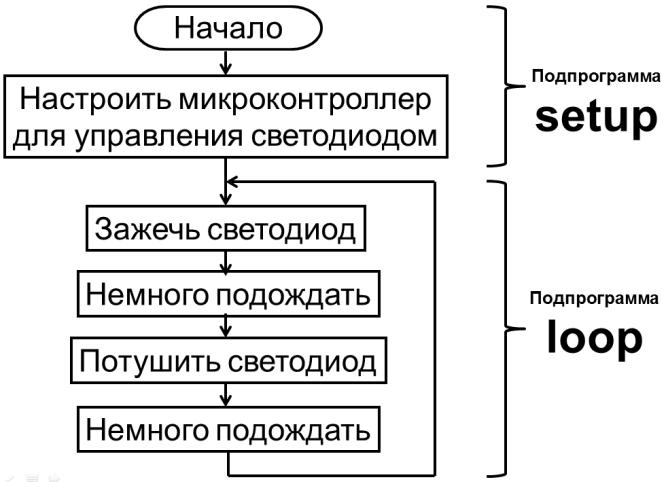


Рис. 10. Блок-схема алгоритма тестовой программы

В языках программирования высокого уровня используются два типа подпрограмм – процедуры и функции. Процедуры принимают данные в виде одного или нескольких аргументов (выводимый символ в примере выше) и выполняют определенный код. Функции, в отличие

от процедур, дополнительно возвращают значение (например, результат вывода – успешно или произошла ошибка в примере выше). В языке Си используются только функции. Следующая функция принимает вещественное число и возвращает число в два раза больше.

```
float mult2(float x)
{
        return x * 2;
}
```

Первая строчка – заголовок функции.

float – тип данных для хранения вещественных чисел, занимает 4 байта в оперативной памяти микроконтроллера.

mult2 – уникальное название (идентификатор) функции, выбираемое самостоятельно. Может состоять только из латинских букв, цифр и символа подчеркивания. Не может начинаться с цифры! А также не может совпадать с командой Си++ или с именем функции, объявленной ранее.

В скобках перечисляются аргументы через запятую: тип и имя переменной. В нашем случае аргументом будет переменная *x* типа **float**. Некоторые другие варианты функций:

```
float function2() ← аргументы могут отсутствовать;
int function3(float x, int y) ← может быть несколько аргументов.
```

Обратите внимание на новый тип **int**: он предназначен для хранения только целых чисел. Разрядность **int** зависит от платформы и сегодня обычно варьируется от 16 до 64 бит. А вот тип **long** является аналогом **int**, но с фиксированной разрядностью 32 бита. Операции с целыми числами намного быстрее, чем с вещественными.

Что делать, если нет необходимости возвращать результат работы подпрограммы, то есть необходима процедура? Для этого используется специальный тип **void** (пустота, ничего). То есть формально функция что-то возвращает, но результат не занимает в памяти места, не существует: void function4(int x) ← процедура по сути.

Далее в фигурных скобках следует тело функции – выполняемые ею команды. Кроме заголовков функции и скобок, вероятно, все конструкции, с которыми придется столкнуться при выполнении работ, заканчиваются точкой с запятой. Функция mult2 состо-

ит только из одной команды (return *x* * 2;). Команда return возвращает управление к точке, откуда функция была вызвана. При этом возвращает значение выражения *x* умноженное на 2. Команда return должна присутствовать в любой функции, возвращающей значение типа, отличного от void. Если возвращается void, можно записать так:

```
void function5()
{
    return;
}
```

Впрочем, в этом случае return можно пропустить, что и было сделано в первой рассмотренной программе. Управление вернется после выполнения всех команд функции:

```
void function6()
{
}
```

Теперь рассмотрим, как вызываются функции.

```
void main()
{
    float y;

    y = mult2(4);
    y = mult2(y);
    function4(10);
    function5();

    return;
}
```

Очевидно, что в любой программе, состоящей только из функций, должна существовать функция, вызывающаяся «сама» (на самом деле операционной системой или микроконтроллером), с которой начнется работа программы – главная функция. В Си у нее имеется уникальное имя main (главная). Первая строчка программы «float y;» определяет новую переменную в программе с именем *y* и типа float. Любые переменные перед первым использованием должны быть определены, микроконтроллер должен выделить необходимое количество ячеек памяти для нее. Кроме варианта, описанного выше, приведем другие способы.

float x, y, pressure; ← определяются переменные x, y, pressure (давление), все типа float, начальные значения неизвестны.

int temperature = 20; ← определяется переменная temperature целочисленного типа и ей сразу же присваивается начальное значение.

float y = mult2(4); ← В Си++ можно и так.

Следующая строчка программы – вызов функции mult2. Запись и действие, как в математике, переменной *y* будет присвоено значение 8, а после выполнения следующей строчки *y* станет равной 16. И даже такой вариант вызова функций является верным:

```
y = mult2(5) + mult2(2);
```

Вызов будет происходить слева направо, возвращаемые значения сложатся, и результат будет присвоен переменной *y*.

Переменные, определенные внутри функции, существуют только во время одного вызова этой функции.

```
void setup()
{
    int variable1 = 5;
}
void loop()
{
    variable1 = variable1 + 1;
}
```

Данная программа ошибочна: в функции loop() переменная не определена! Даже если ее заново определить в loop(), это будет совсем другая переменная variable1.

Стоит еще раз отметить: если функция завершается, то все переменные, определенные в ней, уничтожаются. Иногда требуются глобальные переменные, доступ к которым необходим из любой части программы. В этом случае их необходимо определить вне функций:

```
int variable1 = 5;

void setup()
{
}

void loop()
{
```

```
    variable1 = variable1 + 1;
}
```

Данная программа при каждом шаге будет увеличивать значение переменной на 1. Но в общем случае стоит избегать большого числа таких переменных.

Вернемся теперь к первой программе. Первая функция выглядит так:

```
void setup()
{
    pinMode(13, OUTPUT);
}
```

Как можно догадаться, функция состоит из единственной команды, эта команда – всего лишь вызов другой функции pinMode («режим пина»). Описана эта функция в среде разработки Arduino. Можно попробовать отыскать ее код в текстовых файлах. На самом деле функция записывает соответствующие биты регистра направления микропроцессора, что, в свою очередь, изменяет состояние транзисторных ключей порта. Это приводит к появлению высокого сопротивления между выводом и землей в случае работы пина в качестве входа, снижая потребляемый ток, и низкого – в случае работы в качестве выхода, обеспечивая достаточный ток примерно в 20 мА для свечения светодиода.

Сейчас же достаточно запомнить, что у функции два аргумента – номер пина и его режим работы. Режим работы задается константами OUTPUT (выход), INPUT (вход). Если подключать источник напряжения к пину (например, датчик), используется INPUT. Обратите внимание: подключать к микроконтроллеру можно только датчики с диапазоном выходных напряжений от ноля до пяти вольт. Многие промышленные датчики не удовлетворяют этому условию, необходимо применять специальные согласующие схемы, например на операционных усилителях.

Если подключать устройство, принимающее сигналы от микроконтроллера, используется OUTPUT. Однако максимальный ток одного вывода микроконтроллера типа Atmega – только 40 мА, поэтому напрямую подключить десяток светодиодов к одному выводу

не получится. Впрочем, к разным тоже – максимальный ток по линии питания VCC и GND – 200 мА.

Превышение максимальных токов в результате неверно выбранных режимов в программе приведет к повреждению контроллера.

Вторая функция loop состоит уже из 4 вызовов функций.

```
void loop()
{
    digitalWrite(13, HIGH); ← установка высокого уровня на 13-м пине, светодиод загорелся
    delay(1000); ← задержка 1000 миллисекунд, т. е. 1 секунда
    digitalWrite(13, LOW); ← установка низкого уровня на 13-м пине, светодиод погас
    delay(500); ← задержка 500 миллисекунд, т. е. полсекунды
}
```

Две новые функции также являются стандартными в Arduino. Функция выполнит лишь одно мигание, ее необходимо вызывать постоянно из главной функции main. Она описана в среде Arduino примерно так (точный текст чуть более сложен, его можно найти в файлах среды):

```
void main()
{
    setup();

    while (1)
        loop();
}
```

где while – оператор цикла следующего формата:

```
while (выражение)
    команда;
```

Команда будет вызываться, пока выражение не ноль, а в этом случае всегда. В операторах цикла и условий (далее в работе 4) можно задавать составные команды:

```
while (выражение)
{
    команда1;
    команда2;
```

команда3;
}

Всё в фигурных скобках считается одной командой.

Функции setup и loop не определены в Arduino. Их необходимо написать самостоятельно. Таким образом, программа на Си должна включать минимум одну функцию main, а программа на Arduino – минимум две функции – setup и loop.

Варианты заданий

Составьте программу, циклически формирующую световые импульсы светодиодом в последовательности, указанной в табл. 2.

Таблица 2

№ бригады	Последовательность световых импульсов
1	*Короткий* – **длинный** – **длинный**
2	*Короткий* – *короткий* – **длинный**
3	*Короткий* – **длинный**
4	*Короткий* – *короткий* – **длинный** – **длинный**
5	**Длинный** – **длинный** – **длинный** – *короткий*
6	*Короткий* – *короткий* – *короткий* – **длинный**

Длительность паузы всегда постоянна и равна половине секунды, длительность короткого импульса – полсекунды, длительность длинного импульса – полторы секунды.

Содержание отчета

1. Титульный лист.
2. Цель работы, программа работы и задание для своего варианта.
3. Текстовое описание выполняемых действий в процессе отладки программы из пункта «Краткие теоретические сведения».
4. Блок-схема алгоритма **вашей** программы согласно заданию.
5. Исходный текст программы.
6. Выводы о работе.

Вопросы для защиты лабораторной работы

1. Поясните работу программ, приведенных в отчете.
2. Поясните основные конструкции языка Си, приведенные в кратких теоретических сведениях.
3. Перечислите основные методы измерений.
4. В чем заключаются компенсационный метод, метод отклонений? Приведите примеры.
5. В чем заключаются методы аналогий, повторений и перечисления? Приведите примеры.
6. В чем заключаются методы чередований и подстановки? Приведите примеры.
7. Что такое когерентная выборка? Как данная стратегия измерений используется в осциллографах?
8. Каковы различия между дискретными и аналоговыми датчиками?

Лабораторная работа 3

ПОДКЛЮЧЕНИЕ УЛЬТРАЗВУКОВОГО ДАТЧИКА РАССТОЯНИЯ К МИКРОКОНТРОЛЛЕРУ

Цель работы – научиться работать с ультразвуковым датчиком расстояния, формировать управляющие импульсы с помощью микроконтроллера, определять программным способом длительность импульсов.

Программа работы

1. Подготовить аппаратное и программное обеспечение (установить драйверы) лабораторного стенда к работе.
2. Изучить краткие теоретические сведения.
3. Ввести, скомпилировать, загрузить в микроконтроллер и отладить программу опроса датчика, представленную в кратких теоретических сведениях.
4. Модифицировать программу согласно заданию.
5. Оформить и защитить отчет о лабораторной работе.

Краткие теоретические сведения

1. Ультразвуковой дальномер HC-SR04

Ультразвуковой дальномер HC-SR04 (рис. 11) представляет собой простое и дешевое решение задачи измерения расстояния, не превышающего 4 м. Принцип действия HC-SR04 основан на явлении эхолокации. При его использовании излучатель формирует акустический сигнал, который, отразившись от преграды, возвращается к датчику и регистрируется приемником. Зная скорость распространения ультразвука в воздухе (примерно 340 м/с) и время запаздывания между излученным и принятым сигналом, легко рассчитать расстояние до предмета.

Рис. 11. Ультразвуковой дальномер HC-SR04

Датчик имеет следующие характеристики:

- напряжение питания: +5 В;
- ток ожидания: <2 мА;
- ток, потребляемый в рабочем режиме: 15 мА;
- измеряемое расстояние: 2–400 см;
- ✓ частота импульсов: 40 кГц;
- ✓ разрешение: 0,3 см;
- ✓ угол измерения: 30°;
- ✓ импульс запуска измерения: 10 мкс.

Для подключения к схеме измерения датчик оснащен 4 выводами. Два из них служат для подключения питания. На вход Trig подается запускающий импульс, а с выхода Echo снимается сигнал, длительность которого пропорциональна измеренному расстоянию.

Алгоритм работы с датчиком следующий:

- для старта измерения на вход Trig подается запускающий импульс длительностью 10 микросекунд;
- после обнаружения запускающего импульса датчик излучает 8 импульсов с частотой 40 кГц;
- обнаружив отраженный сигнал, датчик устанавливает высокий уровень на выходе Echo. Длительность данного состояния в микросекундах будет пропорциональна измеренному расстоянию в метрах;
- запускающие импульсы рекомендуется формировать 1 раз в 50 мс.

Всё, что требуется от управляющего микроконтроллера, – сформировать запускающий импульс и измерить значение эхо-сигнала.

2. Подключение датчика с помощью макетной платы

Изображение ультразвукового датчика расстояния приведено на рис. 11. Датчик имеет 4 вывода, расположенных в один ряд, – земля «GND», +5В «Vcc», Trig и Echo. (Выводы подписаны на корпусе датчика!) Первые два необходимо подключить к GND и Vcc контроллера, последние два – к любым цифровым пинам. Сделать это без дополнительных приспособлений (проводов) не получится. Удобным способом подключения является применение макетных плат, широко используемых в электронике. Плата имеет гнезда для подключения электронных компонентов и проводов. Гнезда соединены между собой в особой последовательности (рис. 12).

На рис. 13 и 14 приведены примеры правильного и неправильного подключений датчика. Во втором случае все выводы датчика соединены между собой и никак не соединены с проводами, а все провода (+5 В, 0 В, пины) также соединены между собой. **Это приведет к короткому замыканию при включении контроллера!** Другие концы проводов подключаются к разъемам контроллера.

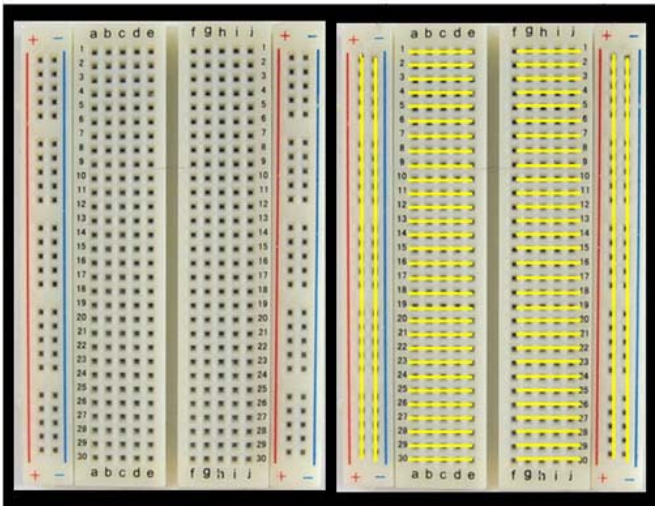


Рис. 12. Макетная плата (желтым цветом обозначены внутренние соединения)

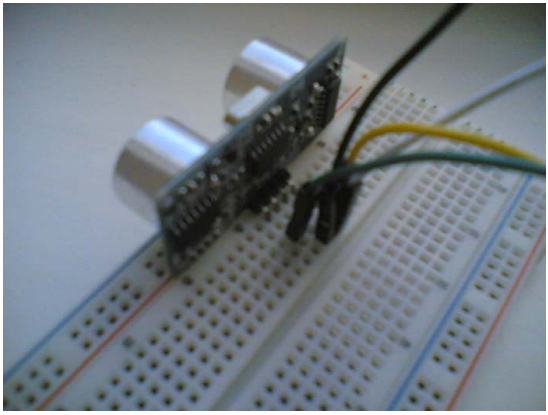


Рис. 13. Правильное подключение датчика с помощью макетной платы

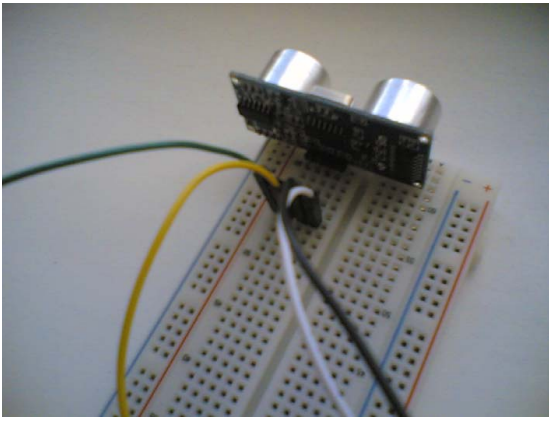


Рис. 14. **НЕПРАВИЛЬНОЕ** подключение датчика

3. Подключение датчика к микроконтроллеру

Соедините вывод Trig датчика с 10-м пином контроллера, а Echo – с 11-м. Загрузите программу, представленную ниже.

```
int Trig_pin = 10;
int Echo_pin = 11;

void setup()
{
    pinMode(Trig_pin, OUTPUT);
    pinMode(Echo_pin, INPUT);

    Serial.begin(9600);
}

void loop()
{
    long duration;

    digitalWrite(Trig_pin, LOW);
    delayMicroseconds(2);
    digitalWrite(Trig_pin, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig_pin, LOW);

    duration = pulseIn(Echo_pin, HIGH);

    Serial.println(duration);
    delay(1000);
}
```

Количество функций, в которые передаются номера пинов, увеличилось, поэтому стоит объявить переменные (в будущем лучше использовать константы, см. литературу) в начале программы, обозначающие номера. Таким образом, если датчик подключить к другим разъемам, достаточно будет изменить только две эти строчки. В функции setup() устанавливаются режимы соответствующих пинов и инициализируется интерфейс связи с компьютером. В функции loop() сначала формируется импульс на вход Trig датчика, что заставляет излучатель создать звуковую волну:

```
digitalWrite(Trig_pin, LOW);   ← 1
delayMicroseconds(2);          ← 2
digitalWrite(Trig_pin, HIGH);  ← 3
delayMicroseconds(10);         ← 4
digitalWrite(Trig_pin, LOW);   ← 5
```

Задаем низкий логический уровень (1), ждем 2 микросекунды (2), чтобы завершились переходные процессы и действительно установилось требуемое напряжение. Устанавливаем высокий логический уровень (3), ждем 10 мкс согласно документации на датчик (4). Задаем низкий уровень (5). Датчик установил высокий уровень на Echo в момент излучения звуковой волны. Как только волна отражается от объекта и обнаруживается приемником, на выходе Echo датчика устанавливается низкий уровень. Таким образом, длительность высокого уровня на Echo после формирования импульса на Trig равна длительности движения ультразвуковой волны до объекта и обратно. Можно непрерывно отслеживать в цикле состояние входа № 11 микроконтроллера или даже установить прерывание в будущем, но пока воспользуемся встроенной функцией pulseIn():

```
duration = pulseIn(Echo_pin, HIGH);
```

Данная функция ожидает появления на пине № 11 (= Echo_pin) высокого логического уровня, запускает отсчет времени, ожидает появления низкого уровня и останавливает отсчет. Измеренное время в микросекундах возвращается и присваивается переменной duration типа long. В работе функции существует таймаут 1 секунда по умолчанию на случай, если звуковая волна не вернулась обратно.

И наконец, выводим длительность прохождения звуковой волной расстояния до объекта и обратно в монитор последовательно-

го порта компьютера с задержкой в 1 секунду для удобства чтения результата:

```
Serial.println(duration);
delay(1000);
```

Задание для лабораторной работы

Задание к этой работе едино для всех бригад: модифицировать приведенную выше программу, чтобы она отображала расстояние в сантиметрах. Используйте скорость звука, равную 340 м/с, для пересчета длительности импульса Echo в сантиметры.

Содержание отчета

1. Титульный лист.
2. Цель работы, программа работы и задание.
3. Текстовое описание выполняемых действий в процессе отладки программы из пункта «Краткие теоретические сведения».
4. Блок-схема алгоритма **вашей** программы.
5. Исходный текст программы.
6. Выводы о работе.

Вопросы для защиты лабораторной работы

1. Объясните работу ультразвукового датчика расстояния.
2. Поясните работу программ, приведенных в отчете.
3. Объясните работу датчика приближения индуктивного типа.
4. Опишите схему, используемую в индуктивных датчиках приближения, для обнаружения металлических объектов.
5. Чем определяется уровень вихревых токов в объекте, помещенном в электромагнитное поле индуктивного датчика?
6. Как влияет экранирование на работу индуктивных датчиков?
7. Объясните работу емкостного датчика приближения. Каких типов они бывают?
8. Опишите структуру датчика расстояния фотоэлектрического типа.
9. Какие режимы применяются в датчиках фотоэлектрического типа? Опишите несколько на выбор.
10. Какие светодиоды применяются в фотоэлектрических датчиках?

Лабораторная работа 4

ИЗМЕРЕНИЕ РАССТОЯНИЙ ПРИ ПОМОЩИ УЛЬТРАЗВУКОВОГО ДАТЧИКА

Цель работы – научиться задавать реакции микропроцессорной системы на показания измерительных датчиков.

Программа работы

1. Подготовить аппаратное и программное обеспечение лабораторного стенда к работе.
2. Изучить работу условного оператора с помощью кратких теоретических сведений и литературы.
3. Разработать, ввести, скомпилировать, загрузить в микроконтроллер и отладить программу согласно заданию.
4. Оформить и защитить отчет о лабораторной работе.

Краткие теоретические сведения

Для того чтобы заставить программу выполнять некие действия при определенных показаниях измерительных датчиков, необходимо в программе воспользоваться условным оператором.

Общий вид условного оператора следующий:

```
if (выражение)
      оператор1;
else
      оператор2;
```

Здесь часть { else оператор2 } является необязательной, можно применять и одиночный оператор:

```
if (выражение)
      оператор1;
```

Вначале вычисляется значение выражения, оператор1 выполняется, если значение выражения истинно. Если выражение ложно (его значение равно нулю) и если есть часть с else, то выполняется оператор2.

Как всегда, можно использовать составные команды.

Примеры выражений

1. Вызвать функцию, если значение переменной temperature > 100.

```
if (temperature > 100)
{
    Func1();
}
```

2. Вызвать функцию, если значение переменной temperature > 100, но меньше 150. && – логическая операция И.

```
if (temperature > 100 && temperature < 150)
{
    Func1();
}
```

3. Вызвать функцию, если значение переменной temperature > 100 или меньше 50. || – логическая операция ИЛИ.

```
if (temperature > 100 | | temperature < 50)
{
    Func1();
}
```

4. Вызвать функцию, если значение переменной temperature равно 100 или меньше 50, или больше 150. == – операция сравнения (именно два раза знак равенства!).

```
if (temperature == 100 | | temperature < 50 || temperature > 150)
{
    Func1();
}
```

5. Вызвать функцию, если значение переменной temperature равно 100, а значение переменной pressure < 3. == – операция сравнения (именно два раза знак равенства!).

```
if (temperature == 100 && pressure < 3)
{
    Func1();
}
```

Задания для лабораторной работы

Варианты заданий приведены в табл. 3.

Таблица 3

№ бригады	Задание
1	Если в сторону датчика двигается объект, индикаторный светодиод загорается (в противном случае не горит)
2	Если от датчика удаляется объект, индикаторный светодиод загорается (в противном случае не горит)
3	Индикаторный светодиод загорается, если расстояние до объекта менее 10 см, если больше 1 м – мигает, в остальных случаях не горит
4	Индикаторный светодиод мигает, если расстояние до объекта менее 15 см, если больше 80 см – непрерывно горит, в остальных случаях не горит
5	См. задание для бригады № 1
6	См. задание для бригады № 3

Содержание отчета

1. Титульный лист.
2. Цель работы, программа работы и задание.
3. Блок-схема алгоритма **вашей** программы согласно варианту.
4. Исходный текст программы.
5. Выводы о работе.

Вопросы для защиты лабораторной работы

1. Какие предосторожности необходимы при использовании ультразвуковых датчиков?
2. Опишите, как работают выходы высокого и низкого уровней. Почему выходы низкого уровня связаны с npn-, а высокого – с pnp-транзисторами?
3. Опишите области применения систем технического зрения.
4. Каковы требования к интерфейсу связи датчиков и контроллеров?

Лабораторная работа 5

ДАТЧИКИ ТЕМПЕРАТУРЫ

Цель работы – изучение принципов работы датчиков температуры, знакомство с основными типами датчиков и экспериментальное снятие их основных характеристик.

Программа работы

1. Ознакомиться с теоретическим материалом по теме работы.
2. Измерить зависимость температуры радиатора транзистора от величины потребляемого от источника питания тока с помощью термопары и мультиметра.
3. Снять зависимость сопротивления терморезистора от тока потребления.
4. Измерить зависимость выходного напряжения аналогового датчика от температуры.
5. Разработать и отладить программу микроконтроллера, осуществляющего индикацию с помощью светодиода превышения температуры 60 градусов.
6. Измерить температуру с помощью цифрового датчика температуры, снять временные диаграммы импульсов датчика.

Краткие теоретические сведения

Температура определяет степень нагрева или охлаждения среды, которая выражается в градусах по шкалам Цельсия, Фаренгейта или Кельвина. В промышленности обычно используется шкала Цельсия, реже – Фаренгейта. Преобразование осуществляется по следующим формулам:

$$^{\circ}C = \frac{5}{9}(^{\circ}F - 32),$$

$$^{\circ}F = \frac{9}{5} \times {}^{\circ}C + 32.$$

Измерение температуры используется, чтобы поддерживать её в некотором диапазоне, выполнять определенные операции при превышении ею установленного предела и снижении ниже его. Из-

мерение температуры используется также для контроля процесса или определения его эффективности.

К наиболее распространенным бытовым устройствам для измерения температуры относятся жидкостные термометры и биметаллические пластины. Последние применяются при двухпозиционном регулировании температуры.

Более подробно остановимся на датчиках, применяемых в промышленности. Они классифицируются по типу выходной величины:

1) датчики, у которых сопротивление зависит от температуры, – резистивные детекторы температуры и терморезисторы;
2) датчики, у которых напряжение зависит от температуры, – термопары и твердотельные датчики температуры;
3) датчики, у которых излучение зависит от температуры, – пирометры.

Резистивные детекторы температуры (РДТ) – датчики, которые используют изменение сопротивления металлов в зависимости от температуры. Внутри некоторого температурного диапазона сопротивление изменяется практически линейно и может быть выражено так:

$$R2 = R1(1 + \alpha \cdot \Delta T),$$

где $R2$ – сопротивление при любой температуре; $R1$ – сопротивление при определенной температуре; α – коэффициент сопротивления; ΔT – изменение температуры.

Таблица 4

Свойства резистивных детекторов температуры

№ п/п	Свойство		
	Материал	Диапазон измерения, °С	Коэффициент сопротивления, Ом/°С
1	Медь	от –151 до +149	0,0042
2	Никель	от –73 до +149	0,0067
3	Платина	от –184 до +815	0,0039
4	Вольфрам	от –73 до +276	0,0045

Изменение сопротивления датчика должно быть преобразовано в изменение напряжения или тока, поскольку именно эти величины могут быть использованы в других узлах системы. Для этого используется мостовая схема, показанная на рис. 15. При комнатной температуре мост сбалансирован так, чтобы разность потенциалов

на его выходе была равна нулю. Сопротивление резистивного детектора температуры можно определить из выражения

$$R_T = R3\frac{R1}{R2}.$$

Кроме резистивного детектора с двумя выводами существуют варианты с тремя и четырьмя выводами. В схеме с тремя выводами с помощью дополнительного вывода устраняются ошибки, вызванные сопротивлением подводящего провода, когда датчик должен быть размещен на большом расстоянии от моста.

Сопротивление детектора с четырьмя выводами устраняет ошибки, вызванные сопротивлением подводящих проводов, и уменьшает ошибки, вызванные шумом, если требуется высокая степень точности измерений.

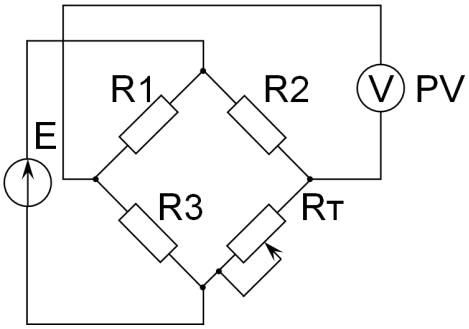


Рис. 15. Мостовая схема с двухвыводным резистивным детектором температуры

В качестве датчиков температуры могут использоваться **термопары**, которые создают небольшое напряжение в несколько милливольт, зависящее от температуры. Свойства термопар, которые изготавливаются с использованием двух различных металлических проводов, приведены в табл. 5.

Термопары обязаны своим появлением Томасу Зеебеку (Seebeck), который в начале XIX века открыл эффект, позже названный его именем. Зеебек обнаружил, что если два провода, сделанные из металлов, спаяны в кольцо с двумя спаями, то в этой цепи при нагревании одного из спаев возникает небольшой ток. Выравнивающее

напряжение на одном из спаев пропорционально количеству теплоты, выделяющейся на другом переходе.

Таблица 5

Свойства термопар

№ п/п	Свойство			
	Тип	Термоэлектрический материал	Диапазон температуры, °С	Коэф. Зеебека α_1, мкВ/К
1	B	Платина и 6 или 30 % родия	от 0 до +1800	–
2	E	Хромель – константан	от –190 до +1000	61
3	J	Железо – константан	от –190 до +800	52
4	K	Хромель – алюминий	от –190 до +1370	40
5	R	Платина или платина и 13 % родия	от 0 до +1700	6
6	S	Платина или платина и 10 % родия	от 0 до +1765	6
7	T	Медь – константан	от –190 до +400	41

Точка соединения провода термопары с проводом вольтметра образует дополнительный переход из неодинаковых металлов, что создает небольшое напряжение, направленное навстречу напряжению рабочего спая и вызывающее ошибку. Однако если температура этого перехода равна опорной (0 °С), то вольтметр измеряет правильное значение напряжения. В конце XIX века свободный спай начали называть холодным, поскольку его помещали в ванну со льдом, которая сохраняла температуру 0 °С.

В наши дни используют следующие методы.

1. Для поддержания заданной температуры холодного спая он помещается в термостат.
2. В контур термопары последовательно включаются компенсирующие цепи, создающие сигнал, таким образом, чтобы суммарный сигнал был таким же, как при опорной температуре.
3. Напряжения обоих переходов преобразуются в цифровую форму, передаются на компьютер, программа которого их учитывает.

Работа датчиков температуры другого типа, называемых **интегральными**, основывается на свойствах биполярного транзистора. В таком датчике используется фундаментальное свойство кремния,

состоящее в том, что ширина запрещенной зоны зависит от температуры. Датчик реализуется следующим образом. В двух биполярных транзисторах, расположенных близко друг к другу в одной интегральной схеме, протекают разные коллекторные токи. Если отношение плотностей токов равно r, то разность напряжений база – эмиттер двух транзисторов будет равна $(kT/q)\ln r$, где k – постоянная Больцмана, T – абсолютная температура, q – заряд электрона. Эта разность напряжений линейно зависит от температуры. С помощью дополнительных электронных схем это напряжение усиливается для того, чтобы получить на выходе необходимую величину.

Типичный датчик в интегральном исполнении имеет следующие параметры: температурный диапазон от –55 °С до +150 °С, нелинейность во всем диапазоне – около 0,3 К.

Описание лабораторной установки

Лабораторная установка включает регулируемый источник стабилизированного напряжения с ограничением по току и стенд для изучения датчиков температуры. С помощью переменного резистора задается ток базы биполярного транзистора. Увеличение тока базы приводит к увеличению коллекторного тока, следовательно, и суммарного потребляемого стендом тока. Температура транзистора изменяется от комнатной при отсутствии тока до 90 градусов при максимальном токе потребления.

Разъемы на стенде предназначены для подключения лабораторного источника питания, термопары и терморезистора к мультиметрам, датчиков – к микроконтроллеру.

Указания к выполнению работы

К пункту 2. Подключите схему питания биполярного транзистора к лабораторному источнику питания, соблюдая полярность. Свободный вывод термопары подключите к мультиметру. Включите источник питания. Установите минимальный ток базы и напряжение питания, равное 15 В. Изменяя переменным резистором ток базы, увеличьте ток транзистора.

К пункту 3. Измерьте температуру с помощью термопары и электрическое сопротивление терморезистора в 10–15 точках. Интервал между измерением тока и измерением температуры должен состав-

лять не менее 3 минут. Внесите данные в таблицу и постройте графики зависимости температуры транзистора от тока источника питания и сопротивления термистора – от температуры.

К пункту 4. Подключите аналоговый датчик температуры к микроконтроллеру. Для этого соедините +5 В стенда с Vcc микроконтроллера, 0 В – с GND, а информационный вывод – с АЦП (разъем An, где *n* – произвольное целое число). С помощью функции analogRead() считайте данные с датчика и перешлите в программу «Монитор порта» через виртуальный COM-порт.

```
int result; // объявление переменной целого типа
result = analogRead(A0); // чтение напряжения с вывода A0
// значение переменной result линейно зависит от напряжения
// и равно нулю, если напряжение равно 0 В, и 1023 – если 5 В
Serial.println(result); // отправить значение через COM-порт в ПК
```

К пункту 5. Используйте условный оператор if для разработки и отладки программы микроконтроллера, осуществляющего индикацию с помощью светодиода превышения температуры 60 градусов.

Содержание отчета

1. Титульный лист.
2. Цель и программа работы.
3. Описание лабораторной установки.
4. Ход работы с результатами измерения.
5. Программа для микроконтроллера.
6. Выводы о результатах работы.

Вопросы для защиты лабораторной работы

1. Объясните принцип работы резистивного детектора температуры.
2. Когда следует выбирать термистор с положительным, а не с отрицательным тепловым коэффициентом?
3. Поясните работу термопары.
4. Почему опорный переход термопары назван холодным спаем?
5. Расскажите, как работает интегральный датчик температуры, применяемый в микроэлектронике.
6. Объясните работу датчиков давления двух типов на выбор.
7. Объясните работу датчиков расхода двух типов на выбор.

Лабораторная работа 6

ДАТЧИКИ МАГНИТНОГО ПОЛЯ

Цель работы – изучение принципов работы датчиков магнитного поля, знакомство с основными типами датчиков и экспериментальное снятие их основных характеристик.

Программа работы

1. Ознакомиться с теоретическим материалом по теме работы.
2. Измерить зависимость выходного напряжения аналогового датчика Холла от тока катушки индуктивности, используя микроконтроллер со встроенным АЦП.
3. Рассчитать зависимость магнитной индукции от тока катушки индуктивности.
4. Снять релейную характеристику цифрового датчика Холла.
5. Оформить и защитить отчет.

Описание лабораторной установки

Лабораторная установка включает регулируемый источник стабилизированного напряжения с ограничением по току, стенд для изучения аналогового датчика магнитного поля и стенд для изучения цифрового датчика магнитного поля. Каждый стенд состоит из переключателя для изменения направления тока через катушку индуктивности и разъемов для подключения датчика к микроконтроллеру. Ток катушки регулируется изменением напряжения питания, индикация величины тока осуществляется с помощью дисплея источника.

Указания к выполнению работы

К пункту 2. Подключите схему питания катушки индуктивности к лабораторному источнику питания, соблюдая полярность. Установите рукоятку тока в среднее положение, а напряжения – в крайнее левое (ноль). Переключатель направления установите в левое положение. Подключите аналоговый датчик магнитного поля к микроконтроллеру. Для этого соедините +5 В стенда с Vcc микроконтроллера, 0 В – с GND, а информационный вывод – с АЦП (разъем

An микроконтроллера, где *n* – произвольное целое число). Включите источник питания. Измерьте зависимость выходного напряжения аналогового датчика Холла от тока катушки индуктивности, используя программу для микроконтроллера. Учитывайте явление магнитного гистерезиса. Проведите измерения в 10–12 точках.

К пункту 3. Приняв за нулевую точку 2,5 В и изменение выходного напряжения за 1,4 мВ/Гс, рассчитайте величину магнитной индукции для каждой из измеренных точек.

К пункту 4. Составьте программу для определения выходного уровня цифрового датчика. Подключите схему подобно схеме аналогового датчика. Обратите внимание, что вывод датчика подключается к цифровому входу микроконтроллера.

Задайте режим работы вывода на ввод в функции setup():

```
pinMode(4, INPUT);
```

4 – номер цифрового входа микроконтроллера, к которому подключен информационный вывод датчика.

Осуществите чтение состояния входа в функции loop():

```
uint8_t result;
result = digitalRead(4);
```

Значение переменной равно LOW, если магнитное поле отсутствует, и HIGH, если магнитная индукция превысит некоторый порог, численное значение которого необходимо определить с помощью результатов работы с аналоговым датчиком.

Содержание отчета

1. Титульный лист.
2. Цель и программа работы.
3. Описание лабораторной установки.
4. Ход работы с результатами измерения.
5. Программы для микроконтроллера с описанием.
6. Блок-схемы алгоритмов.
7. Графики зависимостей магнитной индукции и выхода цифрового датчика от тока катушки.
8. Выводы о результатах работы.

Вопросы для защиты лабораторной работы

1. Какие частицы создают электрическое поле? Какие частицы создают магнитное поле? Что такое индукция магнитного поля?
2. Поясните работу датчика магнитного поля на эффекте Холла.

ЗАКЛЮЧЕНИЕ

В рамках учебно-методического пособия рассмотрена работа с генераторами электрических сигналов и осциллографами, разработка измерительных систем с применением микропроцессоров AVR-архитектуры и основные приемы составления, отладки программ на языке Си для опроса электронных датчиков.

Полученные навыки и умения могут быть использованы при разработке технологических систем для производителей различной продукции, технической поддержке заказчиков, продаже и внедрении датчиков.

Библиографический список

1. Павловская, Т.А. C/C++ : Программирование на языке высокого уровня : учебник для вузов / Т.А. Павловская. – СПб. : Питер, 2007. – 460 с.
2. Подбельский, В.В. Программирование на языке Си / В.В. Подбельский, С.С. Фомин. – СПб. : Финансы и статистика, 2005. – 600 с.
3. Рег, Дж. Промышленная электроника / Дж. Рег. – М. : ДМК Пресс, 2011. – 1136 с.
4. Клаассен, К.Б. Основы измерений. Электронные методы и приборы в измерительной технике / К.Б. Клаассен. – М. : Постмаркет, 2002. – 352 с.