МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Физические методы контроля»

ПРОГРАММИРУЕМЫЕ ЦИФРОВЫЕ УСТРОЙСТВА

Методические рекомендации к лабораторным работам для студентов направления подготовки 12.03.04 «Биотехнические системы и технологии» и 1-54 01 02 «Методы и приборы контроля качества и диагностики состояния объектов» дневной формы обучения

Могилев 2019

УДК 004.4:621.37
ББК 32.973-018.2:32.844
П 78

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Физические методы контроля» «5» октября 2018 г., протокол № 3

Составитель канд. техн. наук, доц. А. А. Афанасьев

Рецензент канд. техн. наук, доц. С. В. Болотов

В методических рекомендациях кратко изложены изучаемые теоретические сведения, приведен порядок выполнения экспериментальных исследований, указана структура отчета о выполненной работе и дан список контрольных вопросов для самопроверки по каждой теме. Составлены в соответствии с рабочей программой по дисциплине «Программируемые цифровые устройства» для студентов направления подготовки 12.03.04 «Биотехнические системы и технологии» и 1-54 01 02 «Методы и приборы контроля качества и диагностики состояния объектов».

Учебно-методическое издание

ПРОГРАММИРУЕМЫЕ ЦИФРОВЫЕ УСТРОЙСТВА

Ответственный за выпуск	С. С. Сергеев
Технический редактор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 16 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилев.

Содержание

1 Лабораторная работа № 1. Исследование работы учебного стенда НТЦ-31.100

Цель работы: изучить состав, функциональную схему стенда, ознакомиться со структурой и принципом работы микроконтроллера, изучить порядок работы стенда.

1.1 Основные теоретические положения

Внешний вид передней панели стенда приведен на рисунке 1.1.

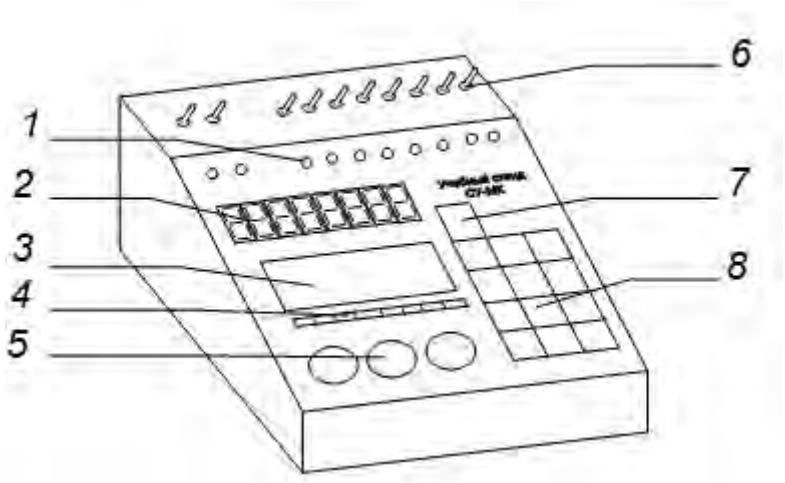

Рисунок 1.1 – Внешний вид передней панели учебного стенда

На передней панели учебного стенда расположены:

– дискретный светодиодный индикатор (набор из 10 светодиодов ДСИ1…ДСИ10) (*1*);

– светодиодный семисегментный динамический индикатор ССДИ (*2*);

– матричный жидкокристаллический индикатор МЖКИ (*3*);

– линейка светодиодов – индикатор аналогового сигнала на выходе ЦАП-ЛСИ (*4*);

– имитаторы аналогового сигнала на входах АЦП АД1…АД3 (*5*);

– датчики дискретных сигналов (набор из 10 переключателей ДД1…ДД2) (*6*);

– кнопка сброса (7);

– двенадцатикнопочная клавиатура КЛ (*8*).

1.2 Порядок выполнения работы

1.2.1 Подключить стенд НТЦ -31.100 к персональному компьютеру.

1.2.2 Включить стенд.

1.2.3 Загрузить в стенд программу тестирования стенда, для чего запустить на персональном компьютере файл \samples\lab1\LoadTestNTC31_100.bat.

1.2.4 Дождаться закрытия окна командной консоли. Выполнить сброс

стенда, нажав кнопку «Сброс».

1.2.5 С помощью ДД1…ДД10 выставить двоичный код десятиразрядного числа в соответствии с заданием преподавателя (верхнее положение тумблера – логическая «1», нижнее – логический «0»). Состояние тумблеров проверить на ДСИ.

1.2.6 Установить АД1, АД2, АД3 поочередно в крайнее левое, среднее, крайнее правое положение. Результат преобразования аналогового сигнала в цифровой код следует наблюдать на ССДИ (АД1, АД2) и на МЖКИ (АД3).

1.2.7 Результат работы ЦАП, на вход которого подается дискретный выходной сигнал АЦП3, нужно наблюдать на ЛСИ.

1.2.8 Исследовать работу клавиатуры, поочередно нажимая клавиши и наблюдая результат на МЖКИ.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, краткое описание стенда, выводы по результатам экспериментальных исследований.

Контрольные вопросы

1 Каковы основные структурные элементы учебного стенда НТЦ -31.100?

2 Опишите органы управления лабораторного стенда.

2 Лабораторная работа № 2. Изучение программного обеспечения лабораторного стенда и системы команд микроконтроллера семейства AVR

Цель работы: изучить среду программирования С++.

2.1 Основные теоретические положения

2.1.1 Среда программирования CodeBlocks. В этой среде есть минимально необходимый комплект (редактор, компилятор и отладчик) для разработки программ. Скачать CodeBlocks можно из Интернета. Далее следует:

– распаковать скачанный архив и запустить инсталляционный файл, согласиться с лицензией. В окошке выбора компонентов для установки выбрать либо standart, либо full (принципиальной разницы нет);

– выбрать путь установки;

– процесс установки завершить, нажав Finish.

Запуск среды программирования CodeBlocks и создание проекта.

Запускается среда программирования CodeBlocks через Пуск –> Все программы –> CodeBlocks –> CodeBlocks или путем щелчка по соответст-

вующему ярлыку на рабочем столе. Главное окно программы после запуска будет иметь следующий вид (рисунок 2.1).

Рисунок 2.1 – Главное окно программы CodeBlocks после запуска

При первом запуске программа выдаст сообщение-вопрос: «Хотите ли Вы ассоциировать файлы исходного кода на C и C++ со средой CodeBlocks», – выбирается третий вариант, т. е. ассоциировать. Впоследствии файлы с расширениями **.c** и **.cpp** будут автоматически открыты в этой среде.

Создание проекта. Проект – это набор файлов, генерируемых средой программирования, необходимых для последующей компиляции программы (исходные, заголовочные, объектные файлы), а также различные вспомогательные файлы (сохраняющие рабочее пространство и др.). По умолчанию в CodeBlocks при запуске открывается стартовая страница, на которой располагаются кнопки для создания проектов, открытия через проводник, открытия последних проектов.

Щелкаем по «Create a new project» (Создать новый проект). В появившемся окошке выбираем значок «Console application» и нажимаем GO. Таким образом можно создавать проекты для построения консольных приложений на языке программирования C++ (рисунок 2.2).

Рисунок 2.2 − Стартовая страница, на которой располагаются кнопки для создания проектов

Далее в следующем окошке нажимаем «Next», в следующем окне выбираем язык C++, нажимаем «Next». Далее вводим название проекта и папку,

в которой он будет находиться. Можно, например, для всех проектов создать папку в корне какого-либо диска на компьютере с названием «My project C++».

В следующем окошке выбора компилятора оставляем все, как есть, и нажимаем «Finish».

Структуру проекта можно посмотреть на панели «Management» (вкладка «Projects»). Здесь видно, что в проект (в папку «Sourses») средой автоматически добавлен файл «main.cpp» (файл, содержащий исходный код программы). В нем и пишется код разрабатываемой программы. Открываем его для редактирования в редакторе CodeBlocks (щелкаем по нему левой кнопкой мыши два раза). Заметим, что файл не пуст: среда программирования сама в него добавила заготовку программы. Делается это для облегчения работы программисту. Набираем в редакторе CodeBlocks следующий текст программы:

```
#include <iostream>
using namespace std;
int main()
{
cout << "This is my first program!" << endl;
return 0;
}
```

Теперь программу нужно скомпилировать, построить, а затем запустить на выполнение. Все это можно сделать через меню программы, но удобнее делать через панель (она расположена сверху). Находим кнопку, которая внешне напоминает шестеренку (при наведении курсора всплывает подсказка «build»), она служит для компиляции и построения проекта. Нажимаем и наблюдаем за процессом внизу на панели «Logs» (вкладка «build messages»). Если панель не видна, то нажимаем клавишу F2. Если ошибок нет, то компиляция (проверка на синтаксические ошибки) и построение (объединение всех нужных файлов в единый объектный модуль для дальнейшего запуска) прошли успешно.

Далее можно запустить программу на выполнение. Для этого нажимаем рядом кнопку в виде треугольника («Run»). Должно появиться окошко с результатом выполнения написанной программы.

Рассмотрим устройство программы подробнее.

Первая строка:

```
#include <iostream>
```

#include – это директива препроцессора, она подключает заголовочный файл (файл с расширением **.h**) **iostream.h**, который содержит объявления функций и переменных для потокового ввода и вывода. Имя подключаемого модуля указывается в <> (если заголовочный файл находится в каталоге \INCLUDE\ конкретной среды разработки), либо в " " (если заголовочный файл находится в том же каталоге, в котором и включающий его модуль разрабатываемой программы с расширением **.cpp)**. Заголовочные файлы с помощью директивы препроцессора будут подключаться всегда, в зависимости от того, что нужно в программе выполнить. В данном случае необходима функция

вывода на экран, которая будет рассматриваться далее.

Так как язык С++ относится к языкам высокого уровня, то многие инструкции (команды) для процессора скрыты, чтобы программисту не приходилось каждый раз программировать то, что уже придумано до него. Поэтому в заголовочных файлах хранятся объявления функций, а эту функцию просто используют, не зная ее реализации. Так и в написанной программе применяют функцию **cout**, которая управляет потоковым выводом (в данном случае выводит строку "This is my first program!" на экран).

Вторая строка *using namespace std;* всегда должна быть в программе.

Далее следует описание единственной в программе функции **main()**. Абсолютно любая консольная программа на языке С++ обязательно включает в себя эту функцию, именно с нее и начинается выполнение программы. Ключевое слово **int**, расположенное перед именем функции, указывает на то, что по завершении своей работы функция **main** вернет операционной системе целочисленное значение. Это даже можно наблюдать в окне программы: ниже строки, запрограммированной на вывод, среда выводит строку, в которой видно returned 0, что означает нормальное завершение работы программы (функции **main**). Если это значение отличное от нуля, то значит, что в программе есть ошибка.

Тело самой функции (оно всегда заключено между скобками { }) содержит строку

```
cout << "This is my first program! " << endl;
```

в которой **cout <<** – это оператор консольного вывода последовательности символов (в данном случае это предложение "This is my first program!"), **<< endl**, – оператор который переводит курсор на следующую строку.

Любая инструкция в языке С++ заканчивается *точкой с запятой*.

return – это оператор возврата из функции. **return 0** – означает, что по окончании своей работы функция (main) вернет значение 0.

2.1.2 Основные операторы и структуры программ на языке С++.

Понятие «**переменная**» в языке С++. «Переменная» – это выделенная программистом при составлении программы область в памяти компьютера, в которой будут храниться её конкретные значения. Значения переменных могут быть числовыми, символьными, логическими.

Объявить переменную можно в любом месте программы до ее первого использования. Однако, если объявлять переменные в начале функции (**main**), читаемость программы будет лучше. При запуске программа резервирует в оперативной памяти компьютера столько места, сколько необходимо для хранения объявленных переменных. Типы переменных в С++, их допустимые значения и объём занимаемой памяти в байтах представлены в таблице 2.1.

Операторы, используемые в С++. Математические функции библиотеки **math.h** языка программирования С++ приведены в таблице 2.2. Операторы символьных констант приведены в таблице 2.3.

Таблица 2.1 – Типы переменных языка С++

Имя типа	Размер, байт	Допустимое значение
char	1	От –128 до 127
int	*Зависит от реализации*	
short int	2	От –32 768 до 32 767
long int	4	От –2 147 483 648 до 2 147 483 647
unsigned char	1	От 0 до 255
unsigned int	*Зависит от реализации*	
unsigned short int	2	От 0 до 65 535
unsigned long int	4	От 0 до 4 294 967 295
float	4	От ~3,4·e^{-38} до ~3,4·e^{+38}
doubl	8	От ~1,7·e^{-308} до ~1,7·e^{+308}
long doubl	10	От ~3,4·e^{-4932} до ~1,1·e^{+4932}

Таблица 2.2 – Математические функции библиотеки **math.h** в С++

Функция	Описание работы функции
double **sqrt** (double **x**);	Данная математическая функция вычисляет и возвращает корень из положительного числа, принимая его в качестве аргумента **x**
double **pow** (double **x**, double **y**);	Возводит число, принимаемое в качестве аргумента **x**, в степень, принимаемую в качестве аргумента **y**
double **fabs** (double **x**);	Вычисляет абсолютное значение числа **x** (иными словами, его модуль)
double **fmod** (double **x**, double **y**);	Вычисляет остаток от деления **x** на **y**
double **ceil** (double **x**);	Вычисляет наименьшее целое, значение которого не будет меньшим, чем **x**. Например, функция **ceil (4.68)** вернет значение **5.00**
double **floor** (double **x**);	В отличии от предыдущей функции эта функция вычисляет наибольшее целое, по значению не превосходящее **x**. Например, функция **floor (4.68)** вернет значение **4.00**
double **modf** (double **value**, double***ptr**);	Разбивает значение аргумента **value** на целую и дробные части. Целую часть функция сохраняет в объекте, на который указывает указатель ***ptr**, а дробную возвращает
double **cos** (double **x**);	Вычисляет косинус аргумента **x**, который задается в **радианах**
double **sin** (double **x**);	То же, но синус
double **tan** (double **x**);	То же, но тангенс
double **acos** (double **x**);	Вычисляет главное значение арккосинуса **x**. Аргумент **x** должен быть из интервала [–1 ; +1]. Функция возвращает значение в радианах из интервала [0; π]
double **asin** (double **x**);	Вычисляет главное значение арксинуса **x**. Аргумент **x** должен быть из интервала [–1 ; +1]. Функция возвращает значение в радианах из интервала [–π/2; +π/2]
double **atan** (double **x**);	Вычисляет главное значение арктангенса **x**
double **exp** (double **x**);	Вычисляет значение показательной функции аргумента **x**
double **log** (double **x**);	Вычисляет натуральный логарифм аргумента **x**
double **log10** (double **x**);	Вычисляет десятичный логарифм аргумента **x**

Таблица 2.3 – Операторы символьных констант, используемые в С++

Символьные константы	Обозначение действий
\n	Переход на новую строку
\t	Табуляция (на 6 символов)
\0	Нулевой символ
\\	Обратная косая черта
\v	Вертикальная табуляция (на 6 позиций вниз)
\f	На новой строке в том же месте
\a	Подача звукового сигнала

Для выполнения арифметических и логических действий над переменными, в языке С++ используются операторы, приведенные в таблице 2.4.

Таблица 2.4 – Арифметические и логические операторы, используемые в языке С++

Название операции	Символ, используемый в коде
Арифметическое сложение	+
Арифметическое вычитание	-
Умножение	*
Деление	/
Отрицание	!
Присваивание	=
Вычисление остатка	%
Логическое сложение	&&
Логическое умножение	||
Проверка на равенство	
Равно	= =
Не равно	! =
Проверка отношения	
Больше	>
Меньше	<
Больше или равно	> =
Меньше или равно	< =

Математические функции в языке С++. Все математические функции принимают в качестве аргументов и возвращают числа с плавающей точкой двойной точности **double**. Иными словами, принимают дробные числа, которые могут содержать после запятой 15…16 разрядов. Это не значит, что при использовании данных функций нужно обязательно применять дробные числа – можно использовать любые числа. Программа сама выполнит необходимые приведения типов.

2.2 Порядок выполнения работы

2.2.1 Выполнить программу «PCULab.2.1», код которой приведен далее. Записать в коде программы комментарии к выполняемым операциям.

/* Программа «PCULab.2.1», которая просит от пользователя ввести два числа, а потом выводит на экран результаты вычислений: сумму, разность, произведение и частное */

```
#include <iostream>
using namespace std;
int main()
{
setlocale(LC_CTYPE,"Russian"); // вывод русского текста
int a, b;
cout<< "Введите число a: " <<endl;
cin>> a;
cout<< "Введите число b: " <<endl;
cin>> b;
cout<< "Сумма " << a << " и " << b << " равна " << a + b <<endl;
cout<< "Разность " <<a<< " и " <<b<< " равна " <<a - b<<endl;
cout<< "Произведение " <<a<< " и " <<b<< " равно " <<a * b<<endl;
cout<< "Частное " << a << " и " << b << " равно " << a / b <<endl;
return 0;
}
```

2.2.2 Составить программу «PCULab.2.2», которая просит пользователя ввести два числа, а потом выводит на экран результаты логических операций над ними: И, ИЛИ, НЕ. Записать в коде программы комментарии к выполняемым операциям.

/* Программа «PCULab.2.2», которая просит пользователя ввести два числа, а потом выводит на экран результаты логических операций над ними: И, ИЛИ, НЕ */

```
#include <iostream>
using namespace std;
int main()
{
setlocale(LC_CTYPE,"Russian"); // вывод русского текста
int a, b;
a=8;
b=4;
//cout<< "Введите  число a: " <<endl;
//cin>> a;
//cout<< "Введите число b: " <<endl;
//cin>> b;
unsignedint c=a&b;
unsignedint d=a|b;
unsignedint e=~a;
unsignedint g=~b;
cout<< "Логическое И над числами " <<a<< " и " <<b<< " равно " <<c<<endl;
cout<< "Логическое ИЛИ над числами " <<a<< " и " <<b<< " равно " <<d<<endl;
```

```
cout<< "Логическое НЕ над числом " <<a<< " равно " <<e<<endl;
cout<< "Логическое НЕ над числом " <<b<< " равно " <<g<<endl;
return 0;
}
```

2.2.3 В программе «PCULab.2.2» измените код, чтобы числа a и b вводились с клавиатуры.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, измененный код программ «PCULab.2.1», «PCULab.2.2», результаты их выполнения, блок-схемы алгоритмов этих программ, выводы.

Контрольные вопросы

1 Опишите результат выполнения логической операции И над одноразрядными двоичными числами.

2 Опишите результат выполнения логической операции ИЛИ над одноразрядными двоичными числами.

3 Опишите результат выполнения логической операции НЕ над одноразрядными двоичными числами.

4 Опишите результат выполнения логической операции ИСКЛЮЧАЮЩЕЕ ИЛИ над одноразрядными двоичными числами.

3 Лабораторная работа № 3. Разработка и исследование программного обеспечения на C++ для цифрового прибора с линейной функцией преобразования

Цель работы: исследовать работу программного обеспечения цифрового прибора на микроконтроллере с линейной функцией преобразования

3.1 Основные теоретические положения

Чтобы проектируемый прибор – рабочее средство измерения (РСИ) – стал измерительным, необходимо предусмотреть разработку специального программного обеспечения, обеспечивающего выполнение градуировочных операций. Сущность их заключается в передаче РСИ единицы измеряемой физической величины от образцового средства измерения (ОСИ). Это может быть сделано путем одновременного воздействия измеряемой величиной X на РСИ и ОСИ (рисунок 3.1).

При этом на их ЦОУ появятся числа N_p и N_o соответственно, причем N_o будет действительным значением X. Выполнив несколько таких экспериментов, по полученным числовым значениям N_{p1}, N_{p2}, $N_{o1} = X_1$ и $N_{o2} = X_2$ может быть

построена графическая зависимость $N_p = f(X)$, которая и будет являться градуировочной характеристикой РСИ (рисунок 3.2).

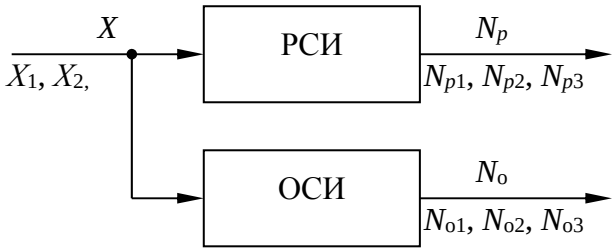

Рисунок 3.1 – Градуировка РСИ с использованием ОСИ

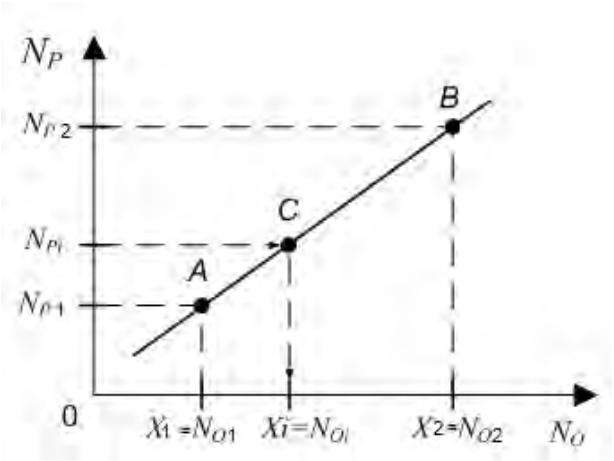

Рисунок 3.2 – Градуировочная характеристика РСИ

Если зависимость $N_p = f(X)$ линейная, то можно записать математическое выражение для градуировочной характеристики, которое будет представлять собой уравнение прямой, проходящей через две точки A и B с известными координатами. Тогда нахождение неизвестной измеряемой величины X_i может быть осуществлено путем решения уравнения:

$$X_i = \frac{(N_{o2} - N_{o1})(N_{pi} - N_{p1})}{N_{p2} - N_{p1}} + N_{o1}, \qquad (3.1)$$

где N_{pi} – числовое значение, получаемое с помощью РСИ при воздействии на него величиной X_i при её измерении.

Координаты точек $A(N_{O1}, N_{P1})$ и $B(N_{O2}, N_{P2})$ – величины известные, которые определяются при градуировке РСИ и записываются в его ППЗУ.

3.2 Порядок выполнения работы

3.2.1 Разработайте программу, которая будет обеспечивать:

– отображение в диалоговом окне выражения «Vvedite pokazanie pribora *Npi* =»;

– ввод с клавиатуры числа *Npi*;

– вычисление результата измерения с использованием выражения (3.1);

– отображение в диалоговом окне выражения «Rezultat izmereniy X =».

В качестве исходных данных используйте координаты точек А и В: N_{01} = 10, N_{P1} = 70, N_{02} = 14, N_{P2} = 83.

3.2.2 Запустите среду программирования CodeBlocks.

3.2.3 Создайте новый проект и присвойте ему имя «PCULab.3».

3.2.4 Вставьте в созданный проект разработанный код программы.

3.2.5 Запустите процесс компиляции программы. Если ошибок нет, то компиляция (проверка на синтаксические ошибки) и построение (объединение всех нужных файлов в единый объектный модуль для дальнейшего запуска) прошли успешно и внизу появится запись

```
Process terminated with status 0 (0 minutes, 1 seconds)
0 errors, 0 warnings
```

3.2.6 Запустите процесс выполнения программы, выбрав курсором кнопку в виде треугольника («Run»).

3.2.7 С помощью клавиатуры сделайте ввод числа, представляющего собой виртуальную реакцию РСИ на воздействие на него измеряемой величины X. Далее нажмите клавишу «Enter». В окошке должен появиться результат выполнения написанной программы.

3.2.8 Проделайте 3…5 раз действия, прописанные в п. 3.2.6, с разными числами, вводимыми с помощью клавиатуры.

3.2.9 Составьте блок-схему алгоритма программы «PCULab.3».

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, код программы «PCULab.3», блок-схему алгоритма этой программы, выводы по результатам её выполнения.

Контрольные вопросы

1 Какое значение имеет запись в программе «#include<math.h»?

2 Какие типы данных используются в исследуемой программе?

3 Какие действия выполняются по команде «\n"»?

4 Какие действия выполняются по команде «cin >> Npi;»?

5 Какие действия выполняются по команде «X = (((($N_{02} - N_{01}$) * ($Npi - N_{P1}$)) / / ($N_{P2} - N_{P1}$)) + N_{01};»?

6 Каково назначение в программе комментариев? Как они отделяются от кода программы?

4 Лабораторная работа № 4. Разработка и исследование программного обеспечения на C++ для цифрового прибора с линейной функцией преобразования с использованием массивов

Цель работы: разработать и исследовать работу программного обеспечения цифрового прибора на микроконтроллере с линейной функцией преобразования с использованием массивов.

4.1 Основные теоретические положения

Массив – это совокупность переменных, содержащих данные одного типа (например, int), объединенных одним общим именем. Каждая отдельно взятая переменная называется элементом (ячейкой) массива. В памяти элементы массива всегда располагаются строго последовательно, благодаря чему повышается скорость доступа к данным. Доступ к элементам массива осуществляется через индексы, которые указываются после имени в квадратных скобках и обязательно нумеруются начиная с нуля.

4.2 Порядок выполнения работы

4.2.1 Запустите среду программирования CodeBlocks.

4.2.2 Создайте новый проект и присвойте ему имя «PCULab.4».

4.2.3 Разработайте программу на C++, с помощью которой можно:

– выполнить ввод числа точек N на градуировочной характеристике с проверкой правильности ввода;

– выполнить ввод числа результатов наблюдений n из диапазона 5…7 с проверкой правильности ввода;

– сформировать массив N_O[ ] показаний образцового прибора путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– сформировать массив N_{PG}[ ] показаний рабочего средства измерения путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– сформировать массив N_P[ ] результатов наблюдений с помощью рабочего средства измерения n = 5…7 путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– найти сумму элементов массива N_P[ ] sumN_P[ ];

– найти среднее арифметическое *Nps* элементов массива N_P[ ]:

$$Nps = \text{sum}Np / n ,$$

и отобразить *Nps* на экране;

– рассчитать результат измерения по формуле (3.1), используя выражение

RezIzmer = ((massivNo[1]-massivNo[0])*(Nps-massivNpg[0]))/

/(massivNpg[1]-massivNpg[0])+massivNo[0]

и отобразить RezIzmer на экране.

4.2.4 Запустите процесс компиляции программы.

4.2.5 Запустите процесс выполнения программы, выбрав курсором кнопку в виде треугольника («Run»).

4.2.6 С помощью клавиатуры сделайте ввод числа точек N, ввод числа результатов наблюдений n, ввод элементов массива N_O[ ], ввод элементов массива N_{PG}[ ], ввод элементов массива N_P[ ].

Далее нажмите клавишу «Enter».В окошке должен появиться результат выполнения написанной программы (рисунок 4.1).

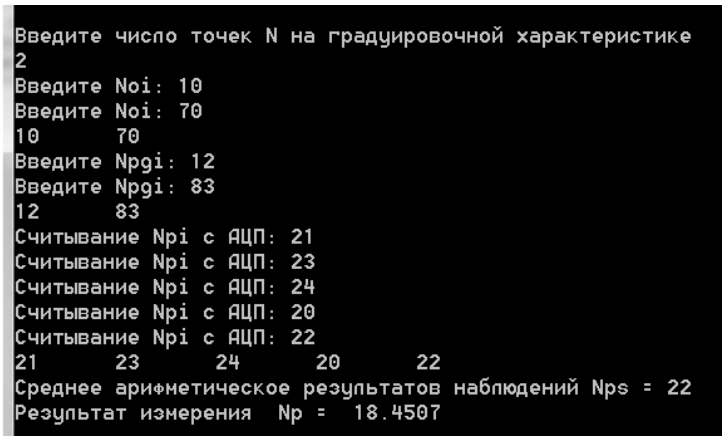

Рисунок 4.1 − Окно с результатом выполненной программы «PCULab.4»

4.2.7 Проделайте 3…5 раз действия, прописанные в п. 4.2.6, с разными числами, вводимыми с помощью клавиатуры.

4.2.8 Составьте блок-схему алгоритма программы «PCULab.4».

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, код программы «PCULab.4», блок-схему алгоритма этой программы, выводы по результатам её выполнения.

Контрольные вопросы

1 Какие типы данных используются в исследуемой программе?

2 Как отделяются от кода программы комментарии?

5 Лабораторная работа № 5. Разработка и исследование программного обеспечения на C++ для цифрового прибора с нелинейной функцией преобразования

Цель работы: разработать и исследовать работу программного обеспечения цифрового прибора на микроконтроллере с нелинейной функцией преобразования.

5.1 Основные теоретические положения

В большинстве случаев функция преобразования РСИ является нелинейной (рисунок 5.1).

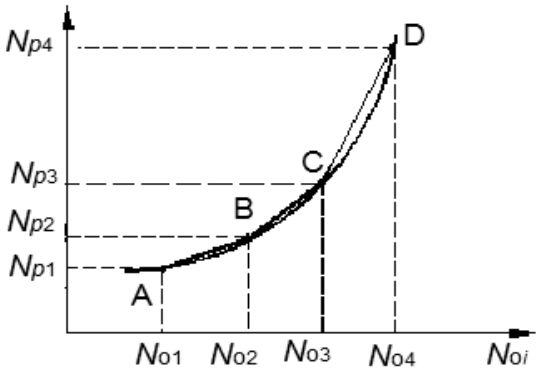

Рисунок 5.1 – Градуировочная характеристика РСИ с нелинейной функцией преобразования

Поэтому определение координат двух точек при его градуировке оказывается недостаточным. При градуировке РСИ с нелинейной функцией преобразования необходимо определить координаты возможно большего числа точек в диапазоне измерения величины X и записать их значения в его ППЗУ.

Нелинейная функция преобразования может быть аппроксимирована набором прямолинейных отрезков, каждый из которых описывается выражением, аналогичным (3.1) (см. рисунок 5.1).

Координаты двух соседних точек позволяют составить уравнение отрезка, соединяющего эти точки.

Для отрезка АВ с учетом уравнение прямой запишется в виде

$$X = \frac{(N_{o2} - N_{o1})(N_p - N_{p1})}{N_{p2} - N_{p1}} + N_{o1}. \quad (5.1)$$

Для отрезка ВС

$$X = \frac{(N_{o3} - N_{o2})(N_p - N_{p2})}{N_{p3} - N_{p2}} + N_{o2}. \quad (5.2)$$

Для отрезка CD

$$X = \frac{(N_{o4} - N_{o3})(N_p - N_{p3})}{N_{p4} - N_{p3}} + N_{o3}. \quad (5.3)$$

Алгоритм вычисления величины X при нелинейной зависимости $N_p = f(X)$ будет отличаться выполнением дополнительных операций, связанных с определением координат двух соседних точек, между числовыми значениями которых окажется измеряемая величина. Определение двух ближайших координат N_{Pi} и $N_{P(i+1)}$ осуществляется путем сравнения полученного РСИ числа N_P и значений, зафиксированных в его ППЗУ при градуировке. Для числовых значений N_{Pi} и $N_{P(i+1)}$ в ППЗУ хранятся также соответствующие им числовые значения N_{Oi} и $N_{O(i+1)}$. Тогда измеряемая величина X может быть определена с помощью выражения

$$X = \frac{N_{O(i+1)} - N_{Oi}}{N_{P(i+1)} - N_{Pi}}(N_P - N_{pi}) + N_{oi}. \quad (5.4)$$

5.2 Порядок выполнения работы

5.2.1 Запустите среду программирования CodeBlocks.

5.2.2 Создайте новый проект и присвойте ему имя «PCULab.5».

5.2.3 Разработайте программу на C++, с помощью которой можно:

– выполнить ввод числа точек N (N = 3…5) на градуировочной характеристике с проверкой правильности ввода.

– выполнить ввод числа результатов наблюдений n из диапазона 5…7 с проверкой правильности ввода;

– сформировать массив N_O[ ] показаний образцового прибора путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– сформировать массив N_{PG}[ ] показаний рабочего средства измерения путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– сформировать массив N_P[ ] результатов наблюдений с помощью рабочего средства измерения n = 5…7 путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– найти сумму элементов массива N_P[ ] sumN_P[ ];

– найти среднее арифметическое Nps элементов массива N_P[ ]:

$$Nps = \text{sum}Np / n,$$

и отобразить Nps на экране;

– выполнить проверку попадания Nps в диапазон (Npi...$Np(i+1)$) и при выполнении условия вычислить результат измерения по формуле (5.4), используя выражение

RezIzmer = ((massivNo[i+1]-massivNo[i])*(Nps-massivNpg[i]))/

/(massivNpg[i+1]-massivNpg[i])+massivNo[i]

и отобразить RezIzmer на экране.

5.2.4 Запустите процесс компиляции программы.

5.2.5 Запустите процесс выполнения программы, выбрав курсором кнопку «Run».

5.2.6 С помощью клавиатуры сделайте ввод числа точек *N*, ввод числа результатов наблюдений *n*, ввод элементов массива N_O[ ], ввод элементов массива N_{PG}[ ], ввод элементов массива N_P[ ].

Далее нажмите клавишу «Enter». В окошке должен появиться результат выполнения написанной программы (рисунок 5.2).

```
Введите число точек N на градуировочной характеристике
3
Введите Noi: 10
Введите Noi: 40
Введите Noi: 90
10      40      90
Введите Npgi: 12
Введите Npgi: 36
Введите Npgi: 78
12      36      78
Считывание Npi с АЦП: 23
Считывание Npi с АЦП: 21
Считывание Npi с АЦП: 20
Считывание Npi с АЦП: 24
Считывание Npi с АЦП: 22
23      21      20      24      22
Среднее арифметическое результатов наблюдений Nps = 22
Результат измерения  Np =  22.5
```

Рисунок 5.2 − Окно с результатом выполненной программы «PCULab.4»

5.2.7 Проделайте 3…5 раз действия, прописанные в п. 5.2.6, с разными числами, вводимыми с помощью клавиатуры.

5.2.8 Составьте блок-схему алгоритма программы «PCULab.5».

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, код программы «PCULab.5», блок-схему алгоритма этой программы, выводы по результатам её выполнения.

Контрольные вопросы

1 Каким количеством прямолинейных отрезков аппроксимирована кривая в программе «PCULab.5»?

2 Какие типы данных используются в исследуемой программе?

3 Какие действия выполняются по команде «cout << "ChitdatacADC: \n"»?

4 Какие действия выполняются по команде «if (*Np*1 <= Np&&Np <= *Np*2);»?

5 Какие действия выполняются по команде «cout << endl << '\t';»?

6 Лабораторная работа № 6. Разработка и исследование программного обеспечения на C++ для цифрового прибора с округлением результата измерения

Цель работы: разработать и исследовать работу программного обеспечения цифрового прибора на микроконтроллере с использованием функции округления результата измерения.

6.1 Основные теоретические положения

Программу надо составить для цифрового прибора с нелинейной функцией преобразования.

Алгоритм вычисления величины X при нелинейной зависимости $N_p = f(X)$ аналогичен рассмотренному в лабораторной работе № 5.

Число точек на кривой функции преобразования прибора и число наблюдений измеряемой величины вводится с клавиатуры при его градуировке; указаны диапазоны их значений, и после ввода происходит проверка правильности ввода; если вводимое значение не попадает в рекомендуемый диапазон, предлагается повторить ввод. При вычислении среднего арифметического минимальное и максимальное значения результатов наблюдений отбрасываются. Результат измерения округляется до заданного числа разрядов после запятой.

6.2 Порядок выполнения работы

6.2.1 Запустите среду программирования CodeBlocks.

6.2.2 Создайте новый проект и присвойте ему имя «PCULab.6.1».

6.2.3 Разработайте программу на C++, с помощью которой надо:

– выполнить ввод числа точек N на кривой нелинейной функции преобразования для ее аппроксимации из диапазона 3…10 с проверкой правильности ввода;

– выполнить ввод числа результатов наблюдений n из диапазона 5…7 с проверкой правильности ввода;

– сформировать массив $N_O[\ ]$ показаний образцового прибора N_O = 3…10 путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– сформировать массив $N_{PG}[\ ]$ показаний рабочего средства измерения N_{PG} = 3…10 путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– сформировать массив $N_P[\ ]$ результатов наблюдений с помощью рабочего средства измерения n = 5…7 путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– найти сумму элементов массива $N_P[\ ]$ sum$N_P[\ ]$;

– в массиве $N_P[\ ]$ найти minN_P и maxN_P и отобразить их на экране;

– найти среднее арифметическое *Nps* элементов массива N_P[ ], отбросив минимальное и максимальное значения minN_P и max N_P:

$$Nps = (\text{sum}Np - \text{min}Np - \text{max}Np) / (n - 2),$$

и отобразить *Nps* на экране;

– рассчитать результат измерения по формуле

RezIzmer=((massivNo[i+1]-massivNo[i])*(Nps-massivNpg[i]))/

/(massivNpg[i+1]-massivNpg[i])+massivNo[i]

и отобразить RezIzmer на экране;

– используя функцию «Округление», округлить результат измерения до заданного числа разрядов после запятой и отобразить скорректированный RezIzmer на экране.

6.2.4 Запустите процесс компиляции программы.

6.2.5 Запустите процесс выполнения программы.

6.2.6 С помощью клавиатуры сделайте ввод числа точек *N*, ввод числа результатов наблюдений *n*, ввод элементов массива N_O[ ], ввод элементов массива N_{PG}[ ], ввод элементов массива N_P[ ].

Далее нажмите клавишу «Enter». В окошке должен появиться результат выполнения написанной программы (рисунок 6.1).

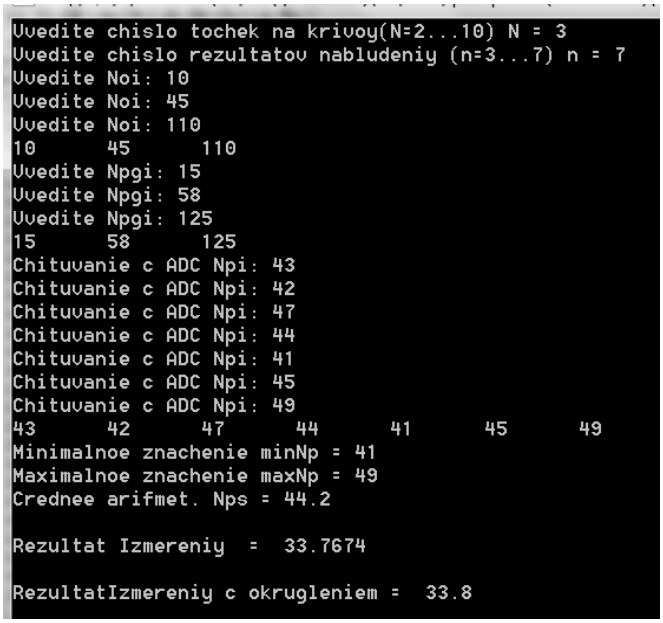

Рисунок 6.1 – Окно с результатами выполненной программы «PCULab.6.1»

6.2.7 Проделайте 2…3 раза действия, прописанные в п. 6.2.6, с разными числами, вводимыми с помощью клавиатуры.

6.2.8 Составьте блок-схему алгоритма программы «PCULab.6.1».

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, код программы «PCULab.6.1», блок-схему алгоритма этой программы, выводы по результатам её выполнения.

Контрольные вопросы

1 Как задается количество наблюдений измеряемой величины в программе «PCULab.6.1» ?

2 Как задается количество точек на кривой функции преобразования в программе «PCULab.6.1»?

3 Каким количеством прямолинейных отрезков аппроксимируется кривая функции преобразования в программе «PCULab.6.1» ?

4 Как задается число разрядов после запятой в программе «PCULab.6.1» ?

7 Лабораторная работа № 7. Разработка и исследование программного обеспечения на C++ для цифрового прибора с автоматическим выбором диапазона измерения

Цель работы: разработать и исследовать работу программного обеспечения цифрового прибора на микроконтроллере с автоматическим выбором диапазона измерения.

7.1 Основные теоретические положения

Необходимо составить программу для нелинейной функции преобразования РСИ.

Алгоритм вычисления величины X при нелинейной зависимости $N_p = f(X)$ аналогичен рассмотренному в лабораторной работе № 4.

Число точек на кривой функции преобразования прибора и число наблюдений измеряемой величины вводится с клавиатуры при его градуировке; указаны диапазоны их значений, и после ввода происходит проверка правильности ввода; если вводимое значение не попадает в рекомендуемый диапазон, предлагается повторить ввод. При вычислении среднего арифметического минимальное и максимальное значения результатов наблюдений отбрасываются.

7.2 Порядок выполнения работы

7.2.1 Запустите среду программирования CodeBlocks.

7.2.2 Создайте новый проект и присвойте ему имя «PCULab.7.1».

7.2.3 Разработайте программу на C++, с помощью которой можно:

– выполнить ввод числа точек N на кривой нелинейной функцией

преобразования для ее аппроксимации из диапазона 3…10 с проверкой правильности ввода;

– выполнить ввод числа результатов наблюдений n из диапазона 5…7 с проверкой правильности ввода;

– сформировать массив N_O[ ] показаний образцового прибора

– N_O = 3…10 путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– сформировать массив N_{PG}[ ] показаний рабочего средства измерения N_{PG} = 3…10 путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– сформировать массив N_P[ ] результатов наблюдений с помощью рабочего средства измерения n = 5…7 путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

– найти сумму элементов массива N_P[ ] sumN_P[ ];

– в массиве N_P[ ] найти minN_P и maxN_P и отобразить их на экране;

– найти среднее арифметическое Nps элементов массива N_P[ ], отбросив минимальное и максимальное значения minN_P и maxN_P:

$$Nps = (\text{sum}Np - \text{min}Np - \text{max}Np) / (n - 2),$$

и отобразить Nps на экране.

– рассчитать результат измерения по формуле

$$\text{RezIzmer} = ((\text{massiv}No[i+1] - \text{massiv}No[i]) * (Nps - \text{massiv}Npg[i])) / / (\text{massiv}Npg[i+1] - \text{massiv}Npg[i]) + \text{massiv}No[i]$$

и отобразить RezIzmer на экране;

– весь диапазон измерения разбить на три поддиапазона, например, 0-3, 3-30, 30-100;

– дополнить программу функцией

```
setlocale(LC_CTYPE,"Russian"); // вывод русского текста.
```

В поддиапазоне 0–3 предусмотрите округление результата измерения до двух цифр после запятой, в поддиапазоне 3–30 – округление результата измерения до одной цифры после запятой, в поддиапазоне 30–100 –- округление результата измерения до целого значения.

7.2.5 Запустите процесс компиляции программы.

7.2.6 Запустите процесс выполнения программы.

7.2.7 С помощью клавиатуры сделайте ввод числа точек N, ввод числа результатов наблюдений n, ввод элементов массива N_O[ ], ввод элементов массива N_{PG}[ ], ввод элементов массива N_P[ ].

Далее нажмите клавишу «Enter». В окошке должен появиться результат выполнения написанной программы (рисунок 7.1).

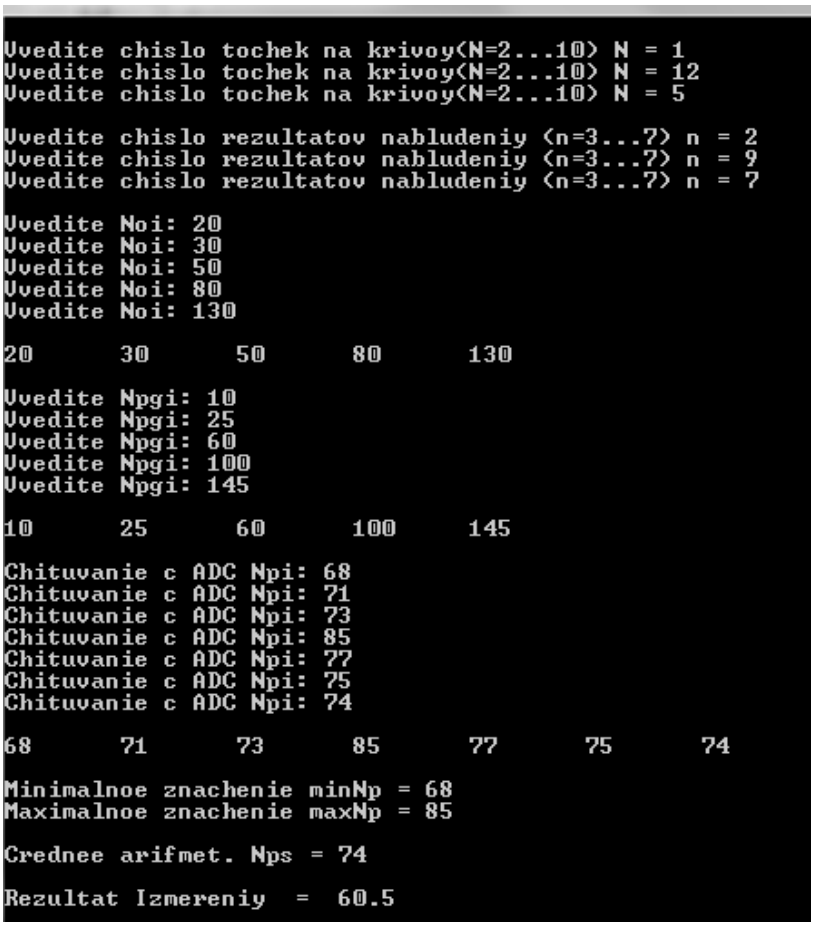

Рисунок 7.1 – Окно с результатом выполненной программы «PCULab.7.1»

7.2.8 Проделайте 2…3 раза действия, прописанные в п. 7.2.7, с разными числами, вводимыми с помощью клавиатуры.

7.2.9 Составьте блок-схему алгоритма программы «PCULab.7.1».

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, код программы «PCULab.7.1», блок-схему алгоритма этой программы, выводы по результатам её выполнения.

Контрольные вопросы

1 Как задается количество наблюдений измеряемой величины в программе «PCULab.7.1» ?

2 Как задается количество точек на кривой функции преобразования в программе «PCULab.7.1»?

3 Каким количеством прямолинейных отрезков аппроксимируется кривая функции преобразования в программе «PCULab.7.1»?

8 Лабораторная работа № 8. Разработка и исследование программного обеспечения на C++ для ввода данных от аналоговых датчиков

Цель работы: разработать и исследовать программу на C++ для ввода данных в микроконтроллер от аналогового датчика.

8.1 Основные теоретические положения.

В качестве контроллера используется плата Arduino Leonardo, созданная на базе микросхемы ATmega32U4. В Arduino Leonardo предусмотрен обмен данными с ЭВМ через USB-порт.

Контроллер Arduino Leonardo имеет:

− 20 цифровых входов-выходов (семь из которых могут использоваться как выходы ШИМ и 12 как аналоговые входы);

− кварцевый генератор на 16 МГц;

− разъем микроUSB;

− силовой разъем;

− разъем ICSP;

− кнопку перезагрузки.

Для работы необходимо подключить контроллер к компьютеру посредством кабеля USB либо подать питание при помощи адаптера AC/DC или батареи.

В качестве аналогового датчика используется потенциометр.

Потенциометр – механическое устройство, у которого изменяется сопротивление при повороте его вала. Если потенциометр подключить к источнику питания (ИП), то он будет выполнять роль делителя напряжения и с его помощью можно плавно изменять напряжение *Uвых* от 0 до + 5 В, которое затем подается на аналоговый вход Arduino (рисунок 8.1).

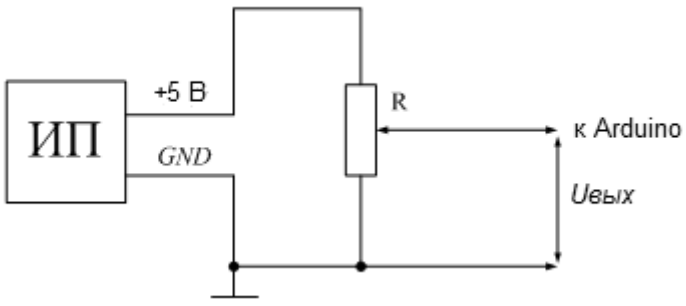

Рисунок 8.1 – Схема подключения потенциометра к источнику питания

8.2 Порядок выполнения работы

8.2.1 Присоедините три провода, припаянные к потенциометру, в соответствии со схемой, приведенной на рисунке 8.2. Один из крайних выводов потенциометра соедините с выводом GND Arduino. Второй крайний вывод

потенциометра соедините с выводом 5 В Arduino. Третьим, оставшимся, проводом соедините аналоговый вход A0 Arduino и средний вывод потенциометра.

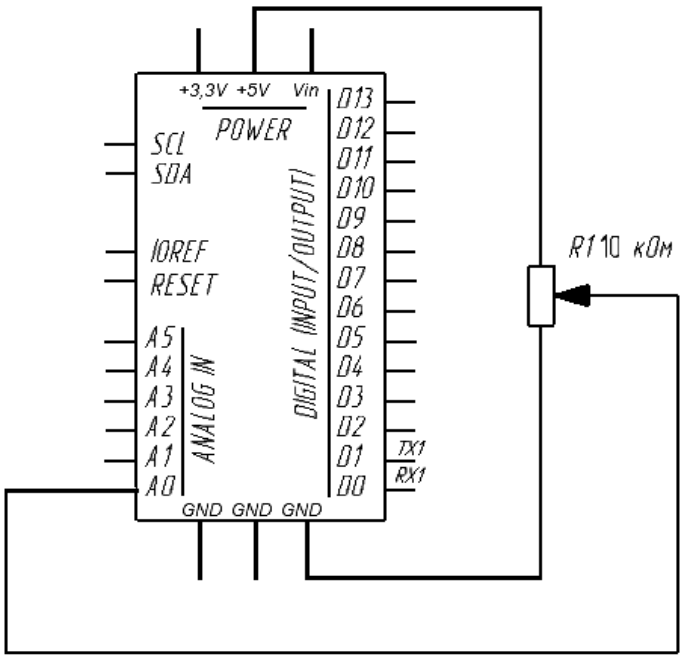

Рисунок 8.2 – Схема подключения потенциометра к контроллеру Arduino

При вращении вала потенциометра можно изменять сопротивление по обе стороны от центрального вывода. Это приводит к изменению напряжения на центральном выводе. Когда напряжение между центральным выводом и выводом, подключенным к 5 В, близится к нулю, напряжение на центральном выводе приближается к 5 В. Если повернуть ручку в другую сторону, то сопротивления поменяются местами и напряжение на центральном контакте приблизится к 0 В. Это напряжение является аналоговым сигналом, который подаётся на аналоговый вход A0 Arduino.

Arduino имеет встроенный 10-разрядный аналого-цифровой преобразователь (АЦП), который считывает значения напряжения и преобразовывает их в числа от 0 до 1023.

Если ручка потенциометра повернута до упора в одну сторону, то на вход АЦП подается 0 В и результат преобразования равен 0. Если ручка потенциометра повернута до упора в другую сторону, то на вход АЦП подается 5 В и результат преобразования равен 1023. В промежуточных положениях ручки потенциометра АЦП возвращает число между 0 и 1023, которое пропорционально напряжению на его среднем выводе.

8.2.2 Подключите платуArduino к ПК через USB-порт.

8.2.3 Запишите программу «Lab.8.1» в память контроллера Arduino, переслав её из ПК. В данной программе в функции установки надо начать последовательную передачу между Arduino и ПК со скоростью 9600 бит данных в секунду командой

```
Serial.begin(9600);
```

Далее, в основном цикле программы, надо создать переменную для хранения значения напряжения, которая может изменяться от 0 до 1023 (лучше всего подойдет тип int) и приходит с потенциометра.

```
intsensorValue = analogRead(A0);
```

Наконец, следует наблюдать эту информацию на мониторе в виде десятичных (DEC) значений. Можно сделать это, используя команду Serial.Println(), с помощью следующей строки кода:

```
Serial.println(sensorValue, DEC)
```

Теперь, открыв SerialMonitor в среде Arduino, можно увидеть поток цифр от нуля до 1023, соответствующих положению ручки потенциометра. Если повернуть ручку, эти показания изменятся почти мгновенно.

Полный текст программы.

```
/*
Считывание аналоговых значений с вывода A0 контроллера Arduino, вывод значений результата аналого-цифрового преобразования на монитор ПК.
*/
// установки:
voidsetup()
{
 // инициализируем последовательную передачу данных со скоростью 9600 бит в секунду:
Serial.begin(9600);
}
// основной цикл:
voidloop()
{
  // читаем значение на аналоговом входе A0:
intsensorValue = analogRead(A0);
  // выводим на монитор считанное значение:
Serial.println(sensorValue);
delay(1);
  // задержка в промежутке между считываниями для стабильности:
}
```

8.2.4 Вращая ручку потенциометра, наблюдайте на мониторе последовательного порта ПК поток цифр от 0 до 1023, соответствующих положению ручки потенциометра.

8.2.5 Измените программу «Lab.8.1» так, чтобы на мониторе последовательного порта ПК отображались значения напряжений, подаваемые на аналоговый вход Arduino.

8.2.6 Используя фрагменты программы лабораторной работы № 7, измените программу «Lab.8.1» так, чтобы на мониторе последовательного порта ПК отображались значения напряжений, подаваемые на аналоговый вход Arduino, с округлением до двух значащих цифр после запятой в диапазоне от 0 до 1 В,

до одной значащей цифры после запятой в диапазоне от 1 до 2 В, до целого значения в диапазоне от 2 до 5 В.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, схему подключения потенциометра к Arduino, выводы по результатам экспериментальных исследований.

Контрольные вопросы

1 Какова разрядность встроенного в контроллер Arduino АЦП?
2 В каком диапазоне изменяется напряжение на выходе потенциометра?
3 В каком диапазоне формируется результат преобразования на выходе АЦП?
4 На какую величину должно измениться напряжение на входе АЦП, чтобы результат преобразования на выходе АЦП изменился на единицу младшего разряда?

9 Лабораторная работа № 9. Разработка и исследование программного обеспечения на C++ для ввода данных от дискретных датчиков

Цель работы: разработать и исследовать программу на C++ для ввода данных в микроконтроллер от дискретного датчика.

9.1 Основные теоретические положения

В качестве контроллера используется плата Arduino Leonardo, в качестве дискретного датчика – кнопка. Схема её подключения к Arduino приведена на рисунке 9.1.

При нажатии кнопки соединяются две точки цепи. Когда кнопка не нажата, нижний её вывод подключен к GND (общей шине) Arduino через резистор *R*1 и находится в состоянии LOW, или логического 0. При нажатии на кнопку устанавливается соединение между двумя выводами, на нижний её вывод подается +5 В, и он переходит в состояние HIGH, или логической 1.

Если не использовать резистор *R*1 и оставить нижний вывод кнопки неподключенным к общей шине, то напряжение на нем будет изменяться хаотично. Это приведет к случайному формированию на нижнем выводе кнопки напряжения высокого или низкого уровня.

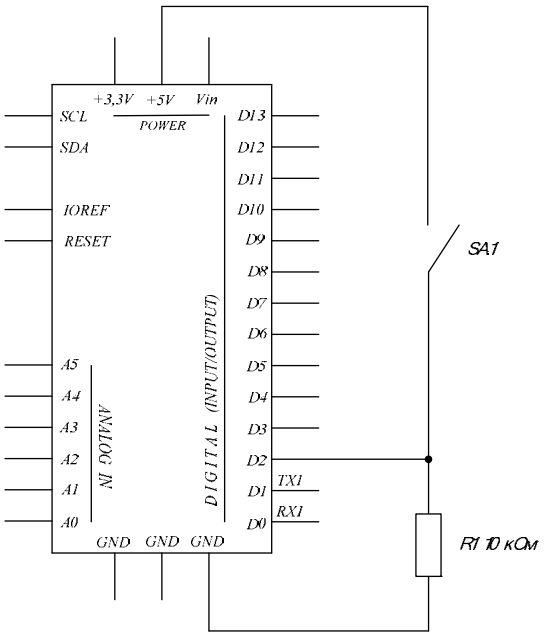

Рисунок 9.1 – Схема подключения кнопки к контроллеру Arduino

9.2 Порядок выполнения работы

9.2.1 Подключите кнопку SA1 и резистор *R1* (10 кОм) к Arduino, как показано на рисунке 9.1.

9.2.2 Подключите плату Arduino к ПК через USB-порт.

9.2.3 Запишите программу «Lab.9.1» в память контроллера Arduino, переслав её из ПК.

Полный текст программы «Lab.9.1».

```
/*
Считывание дискретных значений с вывода D2 контроллера Arduino, вывод значений результата считывания на монитор ПК.
 */
// цифровой вход D2 присоединен к кнопке. Назовем его:
int pushButton = 2;
// проведем необходимые установки:
voidsetup()
 // инициализируем последовательную передачу данных со скоростью 9600 бит в секунду:
Serial.begin(9600);
 // назначим вывод D2 входом:
pinMode(pushButton, INPUT);
}
 // основной цикл:
voidloop()
{
 // читаем значение на входе D2:
int buttonState = digitalRead(pushButton);
 // выводим значение на монитор:
Serial.println(buttonState);
```

```
delay(1);       // задержка для стабильного считывания
}
}
```

9.2.4 Нажимая или отпуская кнопку, наблюдайте на экране дисплея ПК появление 0 или 1, соответствующих положению кнопки.

9.2.5 Дополните программу «Lab.9.1» командами, обеспечивающими включение встроенного в плату Arduino светодиода (подключен к выводу D13) при нажатой кнопке и отключение его при отпускании кнопки.

9.2.6 Подключите потенциометр к контроллеру Arduino, как показано на рисунке 8.2 (см. лабораторную работу № 8). Измените программу «Lab.9.1» так, чтобы результаты преобразования сигнала от аналогового датчика (потенциометра) появлялись на мониторе последовательного порта ПК при нажатой кнопке и исчезали при отпускании кнопки.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, схему подключения кнопки и потенциометра к Arduino, выводы по результатам экспериментальных исследований.

Контрольные вопросы

1 Какое напряжение соответствует логическому 0?
2 Какое напряжение соответствует логической 1?
3 Зачем на схеме, приведенной на рисунке 9.1, используется резистор *R*1?

10 Лабораторная работа № 10. Разработка и исследование программного обеспечения на C++ для управления устройствами сигнализации

Цель работы: исследовать работу индикаторных устройств цифрового прибора на микроконтроллере.

10.1 Основные теоретические положения

Макет цифрового прибора с индикаторными устройствами собирается на макетной плате на основе контроллера Arduino.

В качестве индикаторных устройств используется светодиод HL1 и динамик BA1. Схема их подключения к Arduino приведена на рисунке 10.1.

10.2 Порядок выполнения работы

10.2.1 Соедините один вывод резистора *R*1 = 300 Ом с цифровым выводом D12 на Arduino, второй вывод – с длинным выводом (анодом) светодиода HL1 (рисунок 10.1). Короткий вывод (катод) светодиода HL1 соедините с выводом GND Arduino. Динамик подключите одним выводом к выводу D4 Arduino, другим – к общей шине GND Arduino (см. рисунок 10.1).

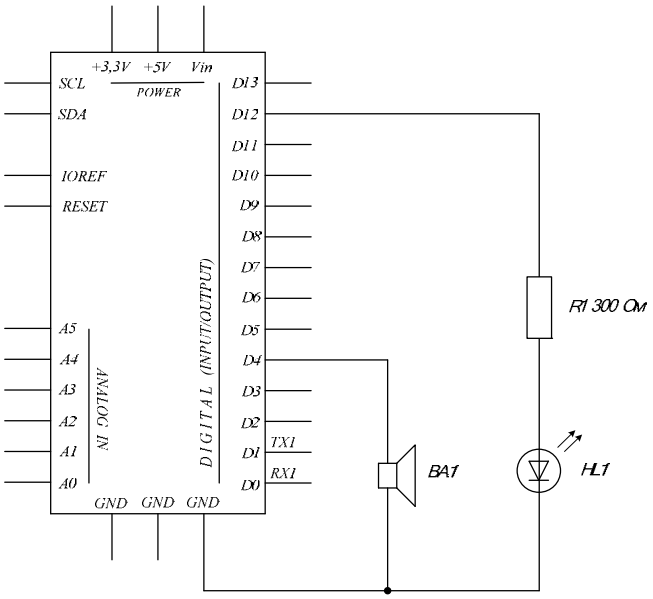

Рисунок 10.1 – Схема подключения светодиода и динамика к микроконтроллеру Arduino

10.2.2 Подключите плату Arduino к ПК через USB-порт.

10.2.3 Запишите программу «Indikator1» в память контроллера Arduino, переслав её из ПК.

Полный текст программы «Indikator1».

```
/*
Включает светодиод на одну секунду, затем выключает на одну секунду.
*/
// К контакту D12 подключен светодиод
// Дадим ему имя:
int led = 12;
// Процедура установки запускается один раз при нажатии кнопки сброса:
voidsetup()
{
// инициализируем цифровой вывод D12 как выход:
pinMode(led, OUTPUT);
}
// основной цикл:
voidloop()
{
digitalWrite(led, HIGH);   // включить светодиод (HIGH означает высокий уровень напряжения )
```

```
delay(1000);              // ждать секунду
digitalWrite(led, LOW);       // выключить светодиод (LOW означает низкий уровень напряжения )
delay(1000);              //ждать секунду
}
```

10.2.4 Изменяя 3…4 раза число в операторе **delay(1000),** наблюдайте за работой светодиода.

10.2.5 Подключите к контроллеру Arduino герконовое реле вместо кнопки SA1 (см. рисунок 9.1, лабораторная работа № 9).

10.2.6 Разработайте программу «Indikator2» так, чтобы светодиод, встроенный в плату Arduino, включался после срабатывания герконового реле при приближении к нему постоянного магнита и погасал при удалении магнита от реле.

10.2.7 Запишите программу «Indikator2» в память контроллера Arduino, переслав её из ПК, и исследуйте работу светодиода, используемого в качестве индикатора состояния герконового реле.

10.2.8 Запишите программу «Dinamik1» в память контроллера Arduino, переслав её из ПК, и исследуйте работу динамика, используемого в качестве звукового индикатора. Для этого 3…4 раза измените в операторе **delay()** параметр, устанавливающий длительность временной задержки.

```
/*
Полный текст программы «Dinamik1»:
Включает динамик, который выполняет функцию звукового индикатора.
*/
// К контакту D4 подключен динамик
// Дадим ему имя:
int Dinamik = 4;
// Процедура установки запускается один раз при нажатии кнопки сброса:
void setup()
{
// инициализируем цифровой вывод D4 как выход:
pinMode(Dinamik, OUTPUT);
}
// основной цикл:
void loop()
{
 digitalWrite(Dinamik, HIGH);   // включить динамик (HIGH означает высокий уровень напряжения )
 delay(500);              // временная задержка
 digitalWrite(Dinamik, LOW);     // выключить динамик (LOW означает низкий уровень напряжения )
 delay(500);              //ждать секунду
}
```

10.2.9 Запишите программу «Dinamik2» в память контроллера Arduino, переслав её из ПК, и исследуйте работу динамика, используемого в качестве звукового индикатора.

Полный текст программы «Dinamik2».

```
/*
Включает динамик, который выполняет функцию звукового индикатора.
*/
// К контакту D4 подключен динамик
// Дадим ему имя:
int Dinamik = 4;
// К контакту D13 подключен светодиод
// Дадим ему имя:
int led = 13;
int frequency = 1200; // задаем частоту сигнала в герцах:
int duration = 300; //задаем длительность звукового сигнала в миллисекундах:
// Процедура установки запускается один раз при нажатии кнопки сброса:
void setup()
{
// инициализируем цифровой вывод D4 как выход:
pinMode(Dinamik, OUTPUT);
}
// основной цикл:
void loop()
{
  tone(Dinamik, frequency, duration);
  delay(10000); // задаем длительность паузы в миллисекундах;
}
```

10.2.10 Измените программу «Dinamik2» так, чтобы устройство звуковой сигнализации включалось после срабатывания герконового реле при приближении к нему постоянного магнита и выключалось при удалении магнита от реле.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, схему подключения светодиода, динамика и кнопки к Arduino, программу «Indikator1» и «Dinamik», выводы по результатам экспериментальных исследований.

Контрольные вопросы

1 Как программно изменяется временная задержка?

2 Как программно изменить частоту сигнала, подаваемого на динамик?

3 Опишите, как с помощью светодиода определяется нажатое состояние кнопки.

4 Опишите, как с помощью динамика определяется состояние герконового реле.

11 Лабораторная работа № 11. Разработка и исследование программного обеспечения на C++ для считывания данных с клавиатуры

Цель работы: разработать и исследовать схему подключения клавиатуры к микроконтроллеру и программное обеспечение на C++ для её сканирования.

11.1 Основные теоретические положения

Макет цифрового прибора собирается на макетной плате на основе контроллера Arduino. Сенсорный модуль на базе микросхемы TTP-224 приведен на рисунке 11.1.

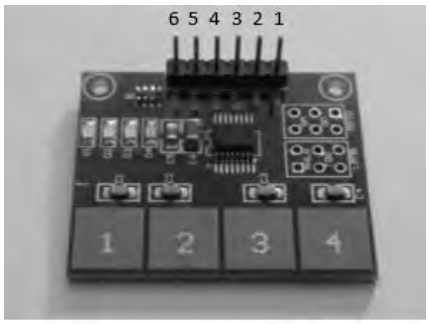

Рисунок 11.1 − Сенсорный модуль на базе микросхемы TTP-224

Он имеет следующее обозначение и назначение выводов:

− 1 (VCC) – подключается к положительному выводу источника питания с постоянным напряжением от 2 до 5,5 В;

− 2 (GND) – общая шина;

− 3 (OUT4) – выход клавиши 4;

− 4 (OUT3) – выход клавиши 3;

− 5 (OUT2) – выход клавиши 2;

− 6 (OUT1) – выход клавиши 1.

Схема подключения сенсорной клавиатуры к Arduino Leonardo приведена на рисунке 11.2.

Рисунок 11.2 – Схема подключения сенсорной клавиатуры к Arduino Leonardo

11.2 Порядок выполнения работы

11.2.1 Подключите плату Arduino к ПК через USB-порт.

11.2.2 Запишите программу «Claviatura1» в память контроллера Arduino, переслав её из ПК.

Полный текст программы «Claviatura1».

```
/* Считывает значения напряжения с клавиатуры (с вывода OUT1), если напряжение высокое, то диод загорается.*/
//Задаем номера входов
int i;
// К контакту D13 подключен светодиод
// Дадим ему имя:
int led = 13;
voidsetup() {
// инициализируем последовательную передачу данных
Serial.begin(9600);
for(i=0; i<4;i++)
{
// инициализируем цифровые выводыD0-D3 как входы:
pinMode(i, INPUT);
// инициализируем цифровой выводD13 как выход:
pinMode(led, OUTPUT);
}
}
// основной цикл:
void loop()
{
for(i=0; i<4;i++)
  {
if(digitalRead(0) == HIGH)
{
digitalWrite(led, HIGH); //включить светодиод Serial.println(i+1);//отображение на мониторе порта нажатой клавиши
}
elseif(digitalRead(0) == LOW)
{
digitalWrite(led, LOW);// выключить светодиод
}
}
delay(100);// задержка в промежутке между считываниями для стабильности
}
```

11.2.3 Изменяя цифровой вход**,** наблюдайте за работой светодиода.

11.2.4 Подключите дополнительно динамик к плате Arduino, как показано на рисунке 10.1.

11.2.5 Запишите программу «Claviatura2» в память контроллера Arduino, переслав её из ПК, и исследуйте работу динамика, используемого в качестве звукового индикатора.

/* Текст программы «Claviatura2»

Считывает значения напряжения с клавиатуры (с вывода OUT4), если напряжение высокое, то подается звуковой сигнал.*/

/*

Полный текст программы «Claviatura2»:

При нажатии клавиши «4» включается динамик, который выполняет функцию звукового индикатора.

```
int i;      //задаем номера входов
int led = 13;// к контакту D13 подключен светодиод
int Dinamik = 4; // к контакту D4 подключен динамик
int frequency = 1200; // задаем частоту сигнала в герцах
int duration = 300;// задаем длительность звукового сигнала в миллисекундах
void setup()
{
for (i=0; i<4; i++)
pinMode(i, INPUT);// инициализируем цифровые выводы D0-D3 как входы
pinMode(led, OUTPUT);// инициализируем цифровой вывод D13 как выход
pinMode(Dinamik, OUTPUT);// инициализируем цифровой вывод D4 как выход
Serial.begin(9600);
}
void loop()       // основной цикл:
{
for(i=0; i<4; i++)
  {if(digitalRead(i) == HIGH)
Serial.println(i+1);
}
if (digitalRead(0) == HIGH)
{
digitalWrite(led, HIGH);// включить светодиод
}
elseif (digitalRead(0) == LOW)
{
digitalWrite(led, LOW); // выключить светодиод
}
if (digitalRead(3) == HIGH)
{
tone(Dinamik, frequency, duration);
delay(1);// задаем длительность паузы в миллисекундах;
}
elseif (digitalRead(3) == LOW)
{
digitalWrite(Dinamik, LOW);// выключить динамик
}
delay(100);
}
```

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, схему подключения светодиода и сенсорной клавиатуры к Arduino, программу «Claviatura1», программу «Claviatura2», выводы по результатам экспериментальных исследований.

Контрольные вопросы

1 Как программно включается светодиод?
2 Как программно включается динамик?
3 Как программно изменить клавишу, подающую сигнал на светодиод?
4 Опишите, каким образом с помощью светодиода определяется нажатое состояние клавиши.

12 Лабораторная работа № 12. Разработка и исследование программного обеспечения на C++ для выработки временных интервалов заданной длительности

Цель работы: разработать и исследовать работу программного обеспечения на C++ для выработки на микроконтроллере временных интервалов заданной длительности.

12.1 Основные теоретические положения

Макет генератора временных интервалов собирается на основе контроллера Arduino.

В качестве индикаторного устройства используется светодиод, расположенный на плате Arduino и присоединенный к выводу D13.

12.2 Порядок выполнения работы

12.2.1 Подключите плату Arduino к ПК через USB - порт.

12.2.2 Запишите программу «Zaderjka1» в память контроллера Arduino, переслав её из ПК, и исследуйте её выполнение.

```
// Программа «Zaderjka1» включает светодиод на время, указанное в операторе delay().
// К контакту D13 подключен светодиод
// Дадим ему имя:
int led = 13;
// Процедура установки запускается один раз при нажатии кнопки сброса:
voidsetup()
{
// инициализируем цифровой вывод D13 как выход:
pinMode(led, OUTPUT);
```

```
}
// основной цикл:
voidloop()
{
digitalWrite(led, HIGH);   // включить светодиод
delay(3000);               // ждать 3 секунды
digitalWrite(led, LOW);    // выключить светодиод
// выход из основного цикла
M1:
delay(1000);               //ждать секунду
go to M1;
}
```

12.2.3 Подключите к выводу D13 Arduino осциллограф.

12.2.4 Подставляя поочередно в операторе **delay**() числа 2000, 5000, 8000, 10000**,** наблюдайте за работой генератора временной задержки с помощью осциллографа и светодиодного индикатора.

12.2.5 Аппаратно-программную задержку можно реализовать, если к Arduino подключить кнопку SA1 (см. рисунок 9.1).

12.2.6 Исследуйте программу «Zaderjka2», в которой задержка начинает вырабатываться после нажатия кнопки SA1. Длительность временного интервала индицируется свечением светодиода, расположенного на плате Arduino и присоединенного к выводу D13. Полный текст программы «Zaderjka2» приведен далее.

```
//Программа "Zaderjka2"
// К контакту D13 подключен светодиод
// Дадим ему имя:
int led = 13;
// цифровой вход D2 присоединен к кнопке. Назовем его:
int pushButton = 2;
// Процедура установки запускается один раз при нажатии кнопки:
voidsetup()
{
  // назначим вывод D2 входом:
pinMode(pushButton, INPUT);
  // инициализируем цифровой вывод D13 как выход:
pinMode(led, OUTPUT);
}
// основной цикл:
voidloop()
{
M1:
  // читаем значение на цифровом входе D2:
intbuttonState = digitalRead(pushButton);
if (digitalRead(buttonState) == HIGH)
{
 // включить светодиод
digitalWrite(led, HIGH);
    // ждать 5 секунд после нажатия кнопки
delay(5000);
```

```
// выключить светодиод
digitalWrite(led, LOW);
  }
else
{
goto M1;
}
}
```

12.2.7 Запишите программу «Zaderjka2» в память контроллера Arduino, переслав её из ПК, и исследуйте её выполнение.

12.2.8 Составьте программу «Zaderjka3», в которой после нажатия кнопки SA1 начинает вырабатываться задержка Δt_1 = 5 с, после чего включается светодиод, время свечения которого Δt_2 = 7 с.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, схему подключения светодиодов и кнопки к Arduino, программу «Zaderjka3», выводы по результатам экспериментальных исследований.

Контрольные вопросы

1 Как программно изменяется временная задержка?
2 Как определить временную задержку с помощью осциллографа?
3 Как можно реализовать аппаратно-программную задержку?

13 Лабораторная работа № 13. Разработка и исследование программного обеспечения на C++ для вывода информации на светодиодные индикаторы

Цель работы: разработать и исследовать схему подключения к микроконтроллеру средств отображения информации, используемых в цифровых приборах, и программное обеспечение на C++ для вывода данных.

13.1 Основные теоретические положения

На рисунке 13.1 показан 4-разрядный семисегментный LED-индикатор с I2C драйвером TM1637.

Особенности рассматриваемого индикатора:
− четыре семисегментные цифры и разделительные точки с общим анодом;
− всего четыре контакта подключения;
− регулируемая яркость – до восьми уровней яркости;
− логические уровни – 5 или 3,3 В;
− ток потребления – до 80 мА.

Рисунок 13.1 – Внешний вид 4-разрядного LED-дисплея с I2C драйвером TM1637

Интерфейс управления:
− GND – общий;
− VCC – питание;
− DIO − вход данных;
− CLK − вход стробирования данных.

Схема подключения к контроллеру Arduino Nano 4-разрядного LED-дисплея с I2C драйвером TM1637 показана на рисунке 13.2.

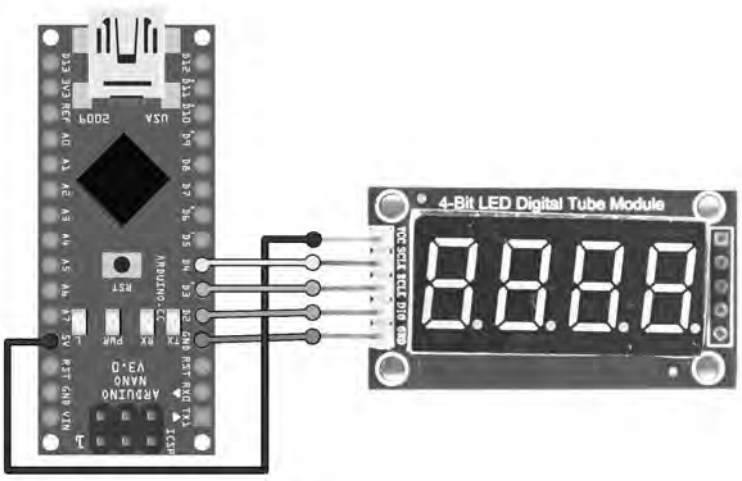

Рисунок 13.2 – Схема подключения к контроллеру Arduino Nano 4-разрядного LED-дисплея с I2C драйвером TM1637

13.2 Порядок выполнения работы

13.2.1 Подключите к плате Arduino LED-дисплей с I2C драйвером TM1637. Подключите плату Arduino к ПК через USB-порт.

13.2.2 Запишите программу D-led-display в память контроллера Arduino, переслав её из ПК, и исследуйте её выполнение.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, схему подключения к контроллеру Arduino Nano 4-разрядного LED-дисплея с I2C драйвером TM1637, результаты экспериментальных исследований, выводы.

Контрольные вопросы

1 Как осуществляется управление работой индикаторов в микроконтроллерных системах?

2 Опишите устройство и принцип работы 4-разрядного LED-дисплея с I2C драйвером TM1637.

14 Лабораторная работа № 14. Разработка и исследование программного обеспечения на C++ для вывода информации на ЖК-дисплей

Цель работы: разработать и исследовать схему подключения к микроконтроллеру средств отображения информации, используемых в цифровых приборах, и программное обеспечение на C++ для вывода данных.

14.1 Основные теоретические положения

На рисунке 14.1 показана микросхема жидкокристаллического индикатора (ЖКИ) WM-C1602N. Она имеет две строки по 16 символов. Индикатор работает под управлением встроенного контроллера.

Рисунок 14.1 – Внешний вид жидкокристаллического индикатора (ЖКИ) WM-C1602N

Назначение выводов у этой микросхемы следующее:

− VCC – питание +5 В;

− GND – общий вывод;

− Contrast – вход для регулировки контрастности изображения символов на индикаторе;

− RS – вход выбора регистра: «0» − регистр команд; «1» − регистр данных;

− RW – выбор режима записи или чтения;

− E – вход разрешения чтения/записи;

− DB0…DB7 – выводы для передачи данных.

Схема подключения к контроллеру Arduino LCD дисплея 1602 показана на рисунке 14.2.

На рисунке 14.3 показана микросхема жидкокристаллического индикатора (ЖКИ) WM-C1602N с клавиатурой.

Сочетание в одном модуле жидкокристаллического индикатора, прини-

мающего отображаемую информацию по шине 4 бит, и клавиатуры упрощает разрабатываемый прибор. Применение модуля для сборки передней панели прибора упрощает его конструкцию. ЖКИ-дисплей имеет регулировку подсветки. Предусмотрены отверстия для установки на переднюю панель прибора.

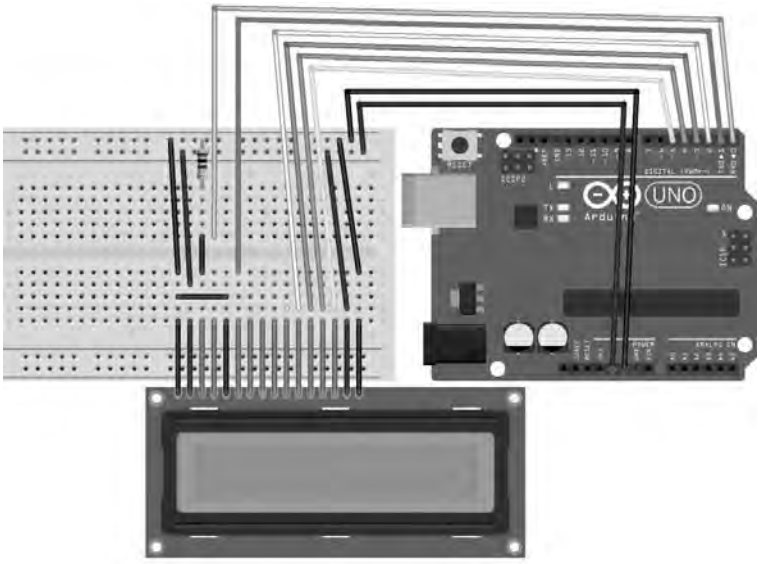

Рисунок 14.2 − Схема подключения к контроллеру Arduino LCD дисплея 1602

Рисунок 14.3 – Внешний вид жидкокристаллического индикатора (ЖКИ) WM-C1602N с клавиатурой

При подаче питания ЖКИ-индикатор на несколько секунд входит в режим инициализации. В это время отображаются квадратики – полное заполнение знакоместа. Переменным резистором, расположенным рядом с индикатором, осуществляется настройка контрастности. Для вывода данных на дисплей надо использовать библиотечную программу LiquidCrystal. Она преобразовывает типы данных (из числовых в символьные) перед их выводом на экран, также можно указать и формат преобразования (DEC, BIN, HEX).

14.2 Порядок выполнения работы

14.2.1 Подключите ЖК-дисплей к плате Arduino.
14.2.2 Подключите плату Arduino к ПК через USB-порт.

14.2.3 Запишите программу «TIME» в память контроллера Arduino, переслав её из ПК, и исследуйте её выполнение.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, схему подключения к контроллеру Arduino дисплея 1602, результаты экспериментальных исследований, выводы.

Контрольные вопросы

1 Как осуществляется управление работой индикаторов в микроконтроллерных системах?

2 Опишите устройство и принцип работы ЖКИ-индикатора.

15 Лабораторная работа № 15. Разработка и исследование программного обеспечения на C++ для управления исполнительными устройствами

Цель работы: разработать и исследовать схемы подключения к микроконтроллеру исполнительных устройств, используемых в цифровых приборах, и программное обеспечение на C++ для управления их работой.

15.1 Основные теоретические положения

15.1.1 Реле SRD-05VDC. Программное управление мощными исполнительными устройствами цифрового устройства может быть реализовано с помощью электромагнитного реле. На рисунке 15.1 показан внешний вид реле SRD-05VDC.

Рисунок 15.1 – Внешний вид реле SRD-05VDC

Данное реле управляется напряжением 5 В и способно коммутировать до 10 А 30 В DC и 10 А 250 В AC.

Реле имеет две раздельные цепи: цепь управления, представленную контактами A1, A2, и управляемую цепь, представленную контактами *1*, *2*, *3*. Цепи никак не связаны между собой (рисунок 15.2).

Релейный модуль имеет три вывода:

1) VCC: «+» питания;

2) GND: «–» питания;
3) IN: вывод входного сигнала.
Подключение модуля к плате Arduino:
− VCC на +5 В на Arduino;
− GND на любой из GND-выводов Arduino;
− IN на любой из цифровых входов/выходов Arduino.

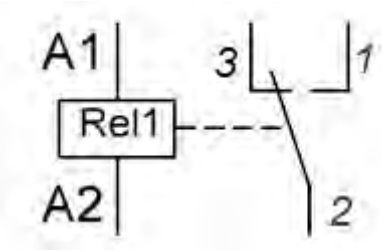

Рисунок 15.2 – Коммутационная схема реле SRD-05VDC

Схема подключения реле к контроллеру Arduino приведена на рисунке 15.3.

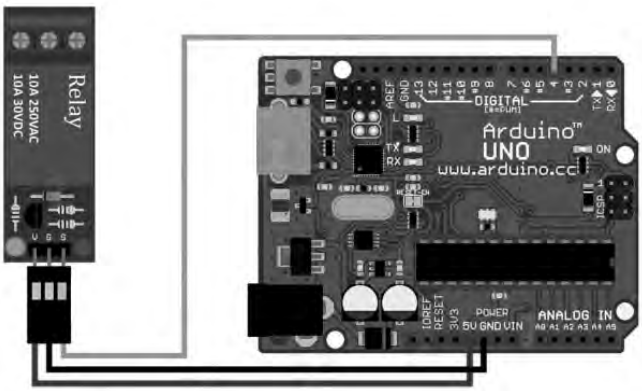

Рисунок 15.3 − Схема подключения реле к контроллеру Arduino

Приведем код программы включения и выключения реле с интервалом в 2 с. Пример программного кода «Rele1».

```
// Реле модуль подключен к цифровому выводу 4
  int Relay = 4;
  void setup()
  {
  pinMode(Relay, OUTPUT);
  }
  void loop()
  {
  digitalWrite(Relay, LOW);   // реле включено
  delay(2000);
  digitalWrite(Relay, HIGH);  // реле выключено
  delay(2000);
  }
```

15.1.2 Управление электродвигателем. Напрямую к цифровым контактам Arduino можно подключать только устройства, потребляющие небольшую мощность, например, светодиод или динамик небольшой мощности. Для доказательства рассмотрим возможность подключения электродвигателя с сопротивлением 10 Ом. При напряжении +5 В (выдаваемым Arduino) в соответствии с законом Ома $U = I \cdot R$ через мотор потечет ток $I = U / R = 5 / 10 =$ = 0,5 А = 500 мА, что намного больше, чем допустимый ток через цифровой вывод Arduino, равный 40 мА. Поэтому при подключении электродвигателя непосредственно к контакту Arduino, контроллер может выйти из строя. Чтобы этого не случилось, напряжение на электродвигатель необходимо подавать от внешнего источника.

Простым вариантом управления электродвигателем является использование реле. Схема подключения электродвигателя к контроллеру Arduino через реле приведена на рисунке 15.4.

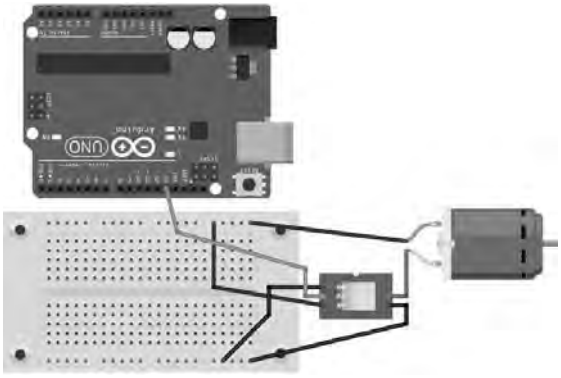

Рисунок 15.4 – Схема подключения электродвигателя к контроллеру Arduino через реле

Код программы, обеспечивающий периодическое включение электродвигателя на 8 с с интервалом в 4 с, представлен далее.

Код программы «Motor1».

```
int Relay1 = 5;
void setup()
{
pinMode(Relay1, OUTPUT);
}
void loop()
{
digitalWrite(Relay1, HIGH);  // реле включено (двигатель работает)
delay(8000);
digitalWrite(Relay1, LOW);   // реле выключено (двигатель не работает)
delay(4000);
}
```

Также можно реализовать работу электродвигателя в реверсивном режиме (обеспечить вращение в обе стороны). Для этого нужно использовать два реле (рисунок 15.5).

Код программы, обеспечивающий реверсивный режим работы

электродвигателя, представлен далее.

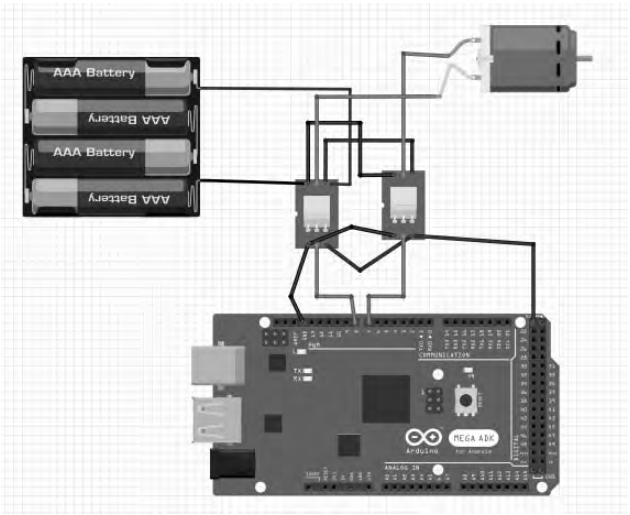

Рисунок 15.5 – Схема подключения электродвигателя к контроллеру Arduino, обеспечивающая реверс вращения

Код программы «Motor2».

```
int Relay3 = 8;
int Relay4 = 9;
void setup()
{
pinMode(Relay3, OUTPUT);
pinMode(Relay2, OUTPUT);
}
void loop()
{
digitalWrite(Relay3, LOW);   // реле включено
delay(1000);
digitalWrite(Relay3, HIGH);  // реле выключено
delay(1000);
digitalWrite(Relay4, LOW);   // реле включено
delay(1000);
digitalWrite(Relay4, HIGH);  // реле выключено
delay(1000);
}
```

Порядок выполнения работы

15.2.1 Подключить реле SRD-05VDC к контроллеру Arduino, контроллер Arduino подключить к ПК.

15.2.2 Загрузить в контроллер Arduino программу «Rele1», исследовать её выполнение.

15.2.3 Составить и загрузить в контроллер Arduino программу «Motor1», обеспечивающую работу электродвигателя в обычном режиме, и исследовать её выполнение.

15.2.4 Загрузить в контроллер Arduino программу «Motor2», обеспечивающую работу электродвигателя в реверсивном режиме, и исследовать её выполнение.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, схемы подключения к Arduino реле, электродвигателя в обычном и реверсивном режимах, программы и результаты экспериментальных исследований, выводы.

Контрольные вопросы

1 Опишите устройство и принцип работы реле.
2 Опишите принцип управления работой электродвигателя в обычном режиме.
3 Опишите принцип управления работой электродвигателя в реверсивном режиме.

Список литературы

1 **Иванов, В. Н.** Электроника и микропроцессорная техника : учебник / В. Н. Иванов, И. О. Мартынова. – Москва : Академия, 2016. – 288 с.
2 **Титов, В. С.** Проектирование аналоговых и цифровых устройств : учебное пособие / В. С. Титов, В. И. Иванов, М. В. Бобырь. – Москва : ИНФРА-М, 2016. – 143 с.
3 **Миленина, С. А.** Электротехника, электроника и схемотехника : учебник и практикум для академ. бакалавриата / С. А. Миленина ; под ред. Н. К. Миленина. – Москва : Юрайт, 2015. – 399 с.
4 **Бладыко, Ю. В.** Электроника. Практикум : учебное пособие / Ю. В. Бладыко. – Минск : ИВЦ Минфина, 2016. – 190 с. : ил.
5 **Horowitz, P.** The art of electronics. Third Edition / P. Horowitz, W. Hill. – Cambridge : University Press, 2015. – 1192 с.