МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Физические методы контроля»

ПРОГРАММИРУЕМЫЕ ЦИФРОВЫЕ УСТРОЙСТВА

Методические рекомендации к лабораторным работам для студентов специальности 1-54 01 02 «Методы и приборы контроля качества и диагностики состояния объектов» дневной формы обучения

Могилев 2024

УДК 621.3.049.77
ББК 32.844+32.85
П78

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Физические методы контроля» «7» марта 2024 г., протокол № 8

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

Рецензент канд. техн. наук, доц. С. О. Парашков

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 специальности 1-54 01 02 «Методы и приборы контроля качества и диагностики состояния объектов» дневной формы обучения.

Учебное издание

ПРОГРАММИРУЕМЫЕ ЦИФРОВЫЕ УСТРОЙСТВА

Ответственный за выпуск	А. В. Хомченко
Корректор	И. В. Голубцова
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 36 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 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

1 Лабораторная работа № 1. 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.

1.1 Основные теоретические положения

Чтобы проектируемый цифровой 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 1.1).

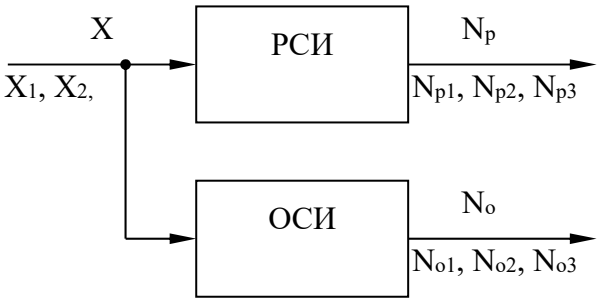

Рисунок 1.1 – Градуировка РСИ с использованием ОСИ

При этом на их дисплеях появятся числа N_p и N_o соответственно, причем N_o будет действительным значением измеряемой величины *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 1.2).

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

$$X_i = \frac{(N_{o2} - N_{o1})(N_{pi} - N_{p1})}{N_{p2} - N_{p1}} + N_{o1}, \qquad (1.1)$$

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

N_{o1}, N_{p1}, N_{o2}, N_{p2} – координаты точек $A(N_{O1}, N_{P1})$ и $B(N_{O2}, N_{P2})$, величины известные, которые определяются при градуировке РСИ и записываются в его постоянное запоминающее устройство (ПЗУ).

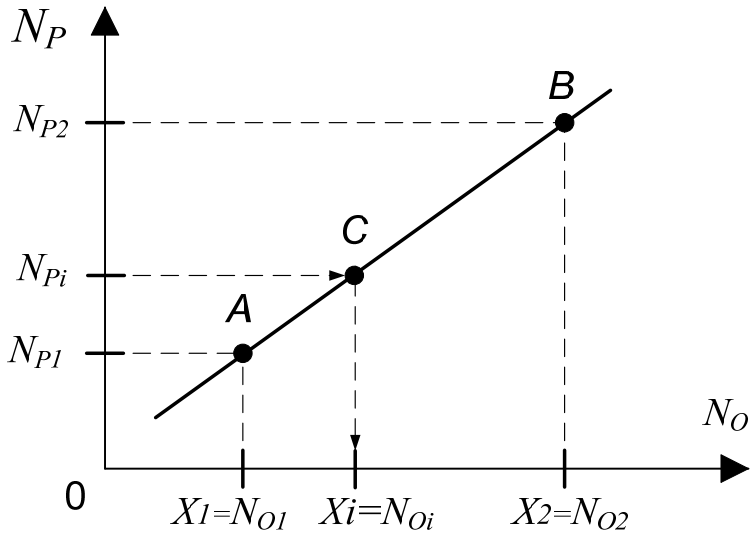

Рисунок 1.2 – Градуировочная характеристика РСИ

При разработке программы переменные объявляются в виде массивов. 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.

1.2 Порядок выполнения работы

1.2.1 Запустите среду программирования CodeBlocks.

1.2.2 Создайте новый проект и присвойте ему имя «PCULab.1».

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

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

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

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

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

5) 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;

6) найти сумму элементов массива N_p[ ] sumN_p[ ];

7) найти среднее арифметическое N_{ps} элементов массива N_p[ ]:

$$N_{ps} = \text{sum}N_p / n$$

и отобразить N_{ps} на экране;

8) рассчитать результат измерения по формуле (1.1), используя выражение

$$\text{RezIzmer} = ((\text{massiv}N_o[1] - \text{massiv}N_o[0]) \cdot (N_{ps} - \text{massiv}N_{pg}[0])) /$$

$$/ (\text{massiv}N_{pg}[1] - \text{massiv}N_{pg}[0]) + \text{massiv}N_o[0],$$

и отобразить RezIzmer на экране.

1.2.4 Запустите процесс компиляции программы.

1.2.5 Запустите процесс выполнения программы, выбрав курсором кнопку в виде треугольника (Run).

1.2.6 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 1.3).

```
Введите число точек N на градуировочной характеристике
2
Введите Noi: 10
Введите Noi: 70
10      70
Введите Npgi: 12
Введите Npgi: 83
12      83
Считывание Npi с АЦП: 21
Считывание Npi с АЦП: 23
Считывание Npi с АЦП: 24
Считывание Npi с АЦП: 20
Считывание Npi с АЦП: 22
21      23      24      20      22
Среднее арифметическое результатов наблюдений Nps = 22
Результат измерения  Np =  18.4507
```

Рисунок 1.3 − Окно с результатом выполненной программы «PCULab.1»

1.2.7 Проделайте 3–5 раз действия, прописанные в п. 1.2.6, с разными числами, вводимыми с помощью клавиатуры.

1.2.8 Составьте блок-схему алгоритма программы «PCULab.1».

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, код программы «PCULab.1», 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?

2 Лабораторная работа № 2. 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.

2.1 Основные теоретические положения

В большинстве случаев функция преобразования РСИ является нелинейной (рисунок 2.1).

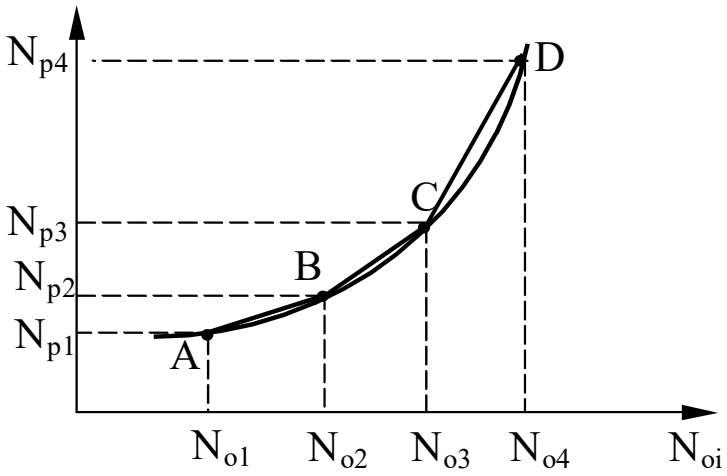

Рисунок 2.1 – Градуировочная характеристика РСИ с нелинейной функцией преобразования

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 (1.1) (см. рисунок 2.1).

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

Для отрезка *AB* уравнение прямой запишется в виде

$$X = \frac{(N_{o2} - N_{o1})(N_p - N_{p1})}{N_{p2} - N_{p1}} + N_{o1}. \qquad (2.1)$$

Для отрезка *BC*

$$X = \frac{(N_{o3} - N_{o2})(N_p - N_{p2})}{N_{p3} - N_{p2}} + N_{o2}. \qquad (2.2)$$

Для отрезка *CD*

$$X = \frac{(N_{o4} - N_{o3})(N_p - N_{p3})}{N_{p4} - N_{p3}} + N_{o3}. \qquad (2.3)$$

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

$$X = \frac{N_{O(i+1)} - N_{Oi}}{N_{P(i+1)} - N_{Pi}}(N_P - N_{pi}) + N_{oi}. \qquad (2.4)$$

2.2 Порядок выполнения работы

2.2.1 Запустите среду программирования CodeBlocks.

2.2.2 Создайте новый проект и присвойте ему имя «PCULab.2».

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

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

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

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

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

5) 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;

6) найти сумму элементов массива N_p[ ] sumN_p[ ];

7) найти среднее арифметическое N_{ps} элементов массива N_p[ ]:

$$N_{ps} = \text{sum}N_p / n$$

и отобразить N_{ps} на экране;

8) выполнить проверку попадания N_{ps} в диапазон (N_{P_i} ... $N_{p(i+1)}$ и при выполнении условия вычислить результат измерения по формуле (2.4), используя выражение

$$\text{RezIzmer} = ((\text{massiv}N_o[1] - \text{massiv}N_o[0]) \cdot (N_{ps} - \text{massiv}N_{pg}[0])) /$$

$$/(\text{massiv}N_{pg}[1] - \text{massiv}N_{pg}[0]) + \text{massiv}N_o[0],$$

и отобразить RezIzmer на экране.

2.2.4 Запустите процесс компиляции программы.

2.2.5 Запустите процесс выполнения программы, выбрав курсором кнопку Run.

2.2.6 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 2.2).

2.2.7 Проделайте 3–5 раз действия, прописанные в п. 2.2.6, с разными числами, вводимыми с помощью клавиатуры.

2.2.8 Составьте блок-схему алгоритма программы «PCULab.2».

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, код программы «PCULab.2», блок-схему алгоритма этой программы, выводы по результатам её выполнения.

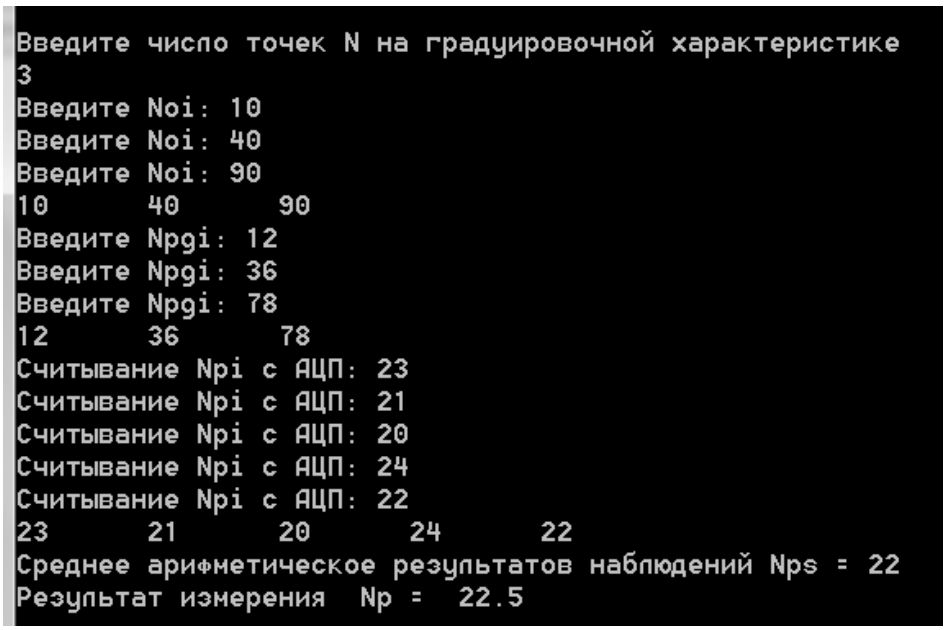

Рисунок 2.2 − Окно с результатом выполненной программы «PCULab.2»

Контрольные вопросы

1 Каким количеством прямолинейных отрезков аппроксимирована кривая в программе «PCULab.2»?

2 Какие типы данных используются в исследуемой программе?

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

4 Какие действия выполняются по команде «if (N_{p1} <= N_p&&N_p <= N_{p2});»?

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

3 Лабораторная работа № 3. Разработка и исследование программного обеспечения на С++ 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.

3.1 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 № 2.

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 проверяется 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.

3.2 Порядок выполнения работы

3.2.1 Запустите среду программирования CodeBlocks.

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

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

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

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

3) сформировать массив N_o[ ] показаний образцового прибора N_o = (3…10) путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

4) 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;

5) 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;

6) найти сумму элементов массива N_p[ ] sumN_p[ ];

7) в массиве N_p[ ] найти minN_p и maxN_p и отобразить их на экране;

8) найти среднее арифметическое N_{ps} элементов массива N_p[ ], отбросив минимальное и максимальное значения minN_p и maxN_p:

$$N_{ps} = (\text{sum}N_p - \text{min}N_p - \text{max}N_p) / (n - 2),$$

и отобразить N_{ps} на экране;

9) рассчитать результат измерения по формуле

$$\text{RezIzmer} = ((\text{massiv}N_o[1] - \text{massiv}N_o[0]) \cdot (N_{ps} - \text{massiv}N_{pg}[0])) /$$

$$/ (\text{massiv}N_{pg}[1] - \text{massiv}N_{pg}[0]) + \text{massiv}N_o[0]$$

и отобразить RezIzmer на экране;

10) используя функцию округления, округлить результат измерения до заданного числа разрядов после запятой и отобразить скорректированный RezIzmer на экране.

3.2.4. Запустите процесс компиляции программы.

3.2.5 Запустите процесс выполнения программы.

3.2.6 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 3.1).

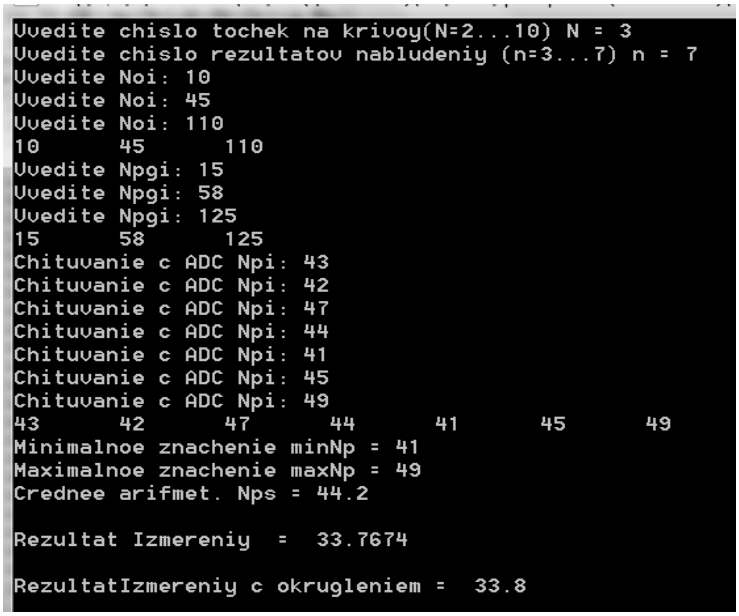

Рисунок 3.1 − Окно с результатами выполненной программы «PCULab.3»

3.2.7 Проделайте 2–3 раза действия, прописанные в п. 3.2.6, с разными числами, вводимыми с помощью клавиатуры.

3.2.8 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 Как задается количество наблюдений измеряемой величины в программе «PCULab.3»?

2 Как задается количество точек на кривой функции преобразования в программе «PCULab.3»?

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

4 Как задается число разрядов после запятой в программе «PCULab.3»?

4 Лабораторная работа № 4. 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.

4.1 Основные теоретические положения

Программу надо составить для нелинейной функции преобразования РСИ.

Алгоритм вычисления величины X при нелинейной зависимости $N_p = f(X)$ аналогичен рассмотренному в лабораторной работе № 2.

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 проверяется 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.

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:

1) выполнить ввод числа точек N на кривой нелинейной функцией преобразования для ее аппроксимации из диапазона (3…10) с проверкой правильности ввода;

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

3) сформировать массив N_o[ ] показаний образцового прибора N_o = (3…10) путем ввода элементов массива с помощью клавиатуры и отображения вводимых данных на экране;

4) 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;

5) 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;

6) найти сумму элементов массива N_p[ ] sumN_p[ ];

7) в массиве N_p[ ] найти minN_p и maxN_p и отобразить их на экране;

8) найти среднее арифметическое N_{ps} элементов массива N_p[ ], отбросив минимальное и максимальное значения minN_p и maxN_p:

$$N_{ps} = (\text{sum}N_p - \text{min}N_p - \text{max}N_p) / (n - 2),$$

и отобразить N_{ps} на экране;

9) рассчитать результат измерения по формуле

$$\text{RezIzmer} = ((\text{massiv}N_o[1] - \text{massiv}N_o[0]) \cdot (N_{ps} - \text{massiv}N_{pg}[0])) /$$

$$/ (\text{massiv}N_{pg}[1] - \text{massiv}N_{pg}[0]) + \text{massiv}N_o[0]$$

и отобразить RezIzmer на экране;

10) весь диапазон измерения разбить на три поддиапазона, например, 0…3, 3…30, 30…100;

11) дополнить программу функцией setlocale(LC_CTYPE,"Russian"); // вывод русского текста.

В поддиапазоне 0…3 предусмотреть 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.

4.2.5 Запустите процесс компиляции программы.

4.2.6 Запустите процесс выполнения программы.

4.2.7 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 4.1).

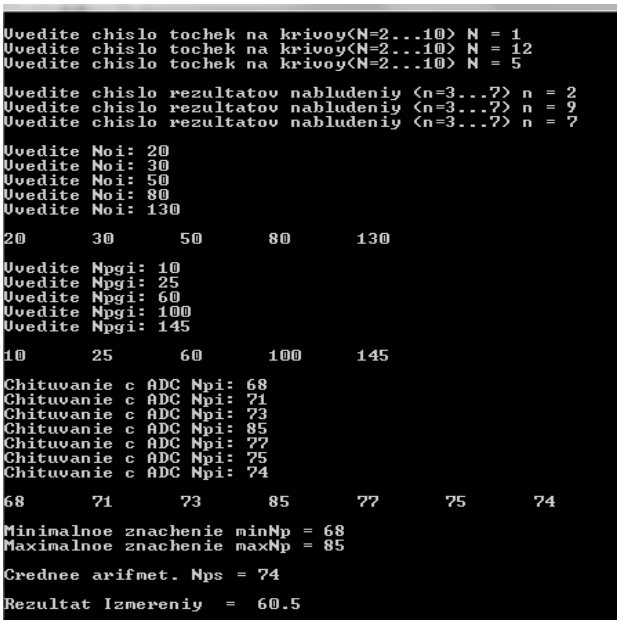

Рисунок 4.1 – Окно с результатом выполненной программы «PCULab.4»

4.2.8 Проделайте 2–3 раза действия, прописанные в п. 4.2.7, с разными числами, вводимыми с помощью клавиатуры.

4.2.9 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 Как задается количество наблюдений измеряемой величины в программе «PCULab.4»?

2 Как задается количество точек на кривой функции преобразования в программе «PCULab.4»?

3 Каким количеством прямолинейных отрезков аппроксимируется кривая функции преобразования в программе «PCULab.4»?

5 Лабораторная работа № 5. Разработка и исследование программного обеспечения на C++ для ввода данных от аналоговых датчиков

Цель работы: исследовать работу цифрового прибора на микроконтроллере с аналоговым датчиком.

5.1 Основные теоретические положения

Макет цифрового прибора собирается на плате Arduino, созданного на базе микроконтроллера ATmega32U4, с использованием в качестве аналоговых датчиков потенциометра и фоторезистивного датчика KY-018.

Плата Arduino Leonardo имеет внешний вид, показанный на рисунке 5.1.

Рисунок 5.1 – Внешний вид платы Arduino Leonardo

Она имеет следующие основные технические характеристики:

− рабочее напряжение – +5 В;

− напряжение питания (рекомендуемое) – +(7…12) В;

− напряжение питания (предельное) – +(6…20) В;

− количество цифровых выводов − 20;

− количество цифровых выводов с ШИМ − 7;

− количество аналоговых входов − 12;

− максимальный выходной ток одного вывода – 40 мА;

− максимальный выходной ток вывода 3,3 V – 50 мА;

− Flash-память – 32 кБ (ATmega32U4) постоянной памяти для хранения выполняемых программ, из которых 4 кБ используются загрузчиком;

− SRAM – 2,5 кБ (ATmega32U4) оперативной памяти для временного хранения данных;

− EEPROM – 1 кБ (ATmega32U4) энергонезависимой памяти для длительного хранения данных;

− тактовая частота – 16 МГц.

Плата Arduino может быть запитана от микроUSB либо от внешнего источника питания. В качестве внешнего источника питания (не USB) может использоваться сетевой AC/DC-адаптер или аккумулятор/батарея. Штеккер адаптера (диаметр – 2,1 мм, центральный контакт − положительный) необходимо вставить в соответствующий разъем питания на плате. В случае питания от аккумулятора/батареи, ее провода необходимо подсоединить к выводам GND и Vin разъема POWER.

Рекомендуемое напряжение внешнего источника питания может быть в пределах от 7 до 12 В. Уменьшение напряжения питания ниже 7 В приводит к уменьшению напряжения на выводе 5V, что может стать причиной нестабильной работы устройства. Использование напряжения больше 12 В может приводить к перегреву стабилизатора напряжения и выходу платы из строя.

Обозначение контактов (пинов) на Arduino показано на рисунке 5.2. Они имеют следующее назначение:

− **Vin** – пин для подачи напряжения на плату Arduino от внешнего источника питания (пин не связан с +5 В от USB или другим стабилизированным напряжением). Через этот пин можно как подавать внешнее питание, так и потреблять ток, если плата Arduino запитана от внешнего адаптера;

− **5V** – на этом пине присутствует напряжение +5 В, поступающее от стабилизатора напряжения, расположенного на плате Arduino. Может поступать как с выхода стабилизатора напряжения питания Vin, так и от USB или другого стабилизированного источника +5 В;

− **GND** – пин, соединенный с минусовым выводом источника питания (его ещё называют общим выводом или выводом «земля»);

− **IOREF** – этот пин предоставляет платам расширения информацию о рабочем напряжении на микроконтроллере. В зависимости от напряжения, плата расширения может переключиться на соответствующий источник питания либо задействовать преобразователи уровней, что позволит ей работать с устройствами, питающимися как от источников +5 В или от +3,3 В;

− **цифровые (digital) входы/выходы 0 – 13** – пины с логическим уровнем единицы − +5 В, нуля − 0 В; максимальный ток выхода с пина − 40 мА. К контактам подключены подтягивающие резисторы, которые по умолчанию выключены, но могут быть включены программно. На выходах пинов 3, 5, 6, 9, 10, 11 и 13 может быть программно реализована широтно-импульсная модуляция (ШИМ) выходной последовательности прямоугольных импульсов;

− **аналоговые (analog in) входы/выходы** A0 – A5 и A6 – A11 (на цифровых пинах 4, 6, 8, 9, 10 и 12). Микроконтроллер ATmega32U4 имеет встроенный 10-разрядный аналого-цифровой преобразователь (АЦП), который считывает значения напряжений, поступающие на аналоговые входы и преобразовывает их в числа в двоичном формате от 0000000000 (при $U_{вх}$ = 0 В) до 1111111111 (при $U_{вх}$ = +5 В). В десятичном формате это числа от 0 до 1023;

− **SDA и SCL** – пины для передачи данных в последовательном коде по двухпроводной шине **TWI/I²C** (для обмена данными с периферийными устройствами с использованием библиотеки Wire);

− **RX1←0** и **TX1→1** – пины, через которые осуществляется обмен данными в последовательном коде по протоколу UART между платой Arduino и другими устройствами (на плате Arduino Leonardo эти пины используются для связи с компьютером через порт микроUSB);

− **RESET (RST)** – пин для подачи сигнала «Сброс» от внешнего устройства (перезапуска программы). Для выполнения этой функции на плате имеется кнопка «**RESET».**

Потенциометр – механическое устройство, у которого изменяется сопротивление при повороте его вала. Если потенциометр R1 подключить к источнику питания (ИП), то он будет выполнять роль делителя напряжения и с его помощью можно плавно изменять напряжение $U_{вых}$ от 0 до +5 В, которое затем подается на аналоговый вход Arduino (рисунок 5.3).

Фоторезистивный аналоговый датчик **KY-018**. Его внешний вид и назначение выводов показаны на рисунке 5.4.

Принцип работы фоторезистивного аналогового датчика KY-018 основан на уменьшении сопротивления светочувствительного полупроводникового слоя при освещении. Он применяется для управления освещением в зависимости от уровня яркости и интенсивности света. Сопротивление фоторезистора увеличивается при уменьшении освещенности и, наоборот, чем больше света, тем сопротивление меньше.

Основные характеристики фоторезистивного датчика KY-018:

− напряжение питания − +5 В;

− сопротивление датчика варьируется (в зависимости от освещенности) от 1 до 100 кОм.

5.2 Порядок выполнения работы

5.2.1 Исследование устройства с потенциометром.

5.2.1.1 Присоедините три провода, припаянные к потенциометру *R*1, к плате Arduino в соответствии со схемой, приведенной на рисунке 5.5.

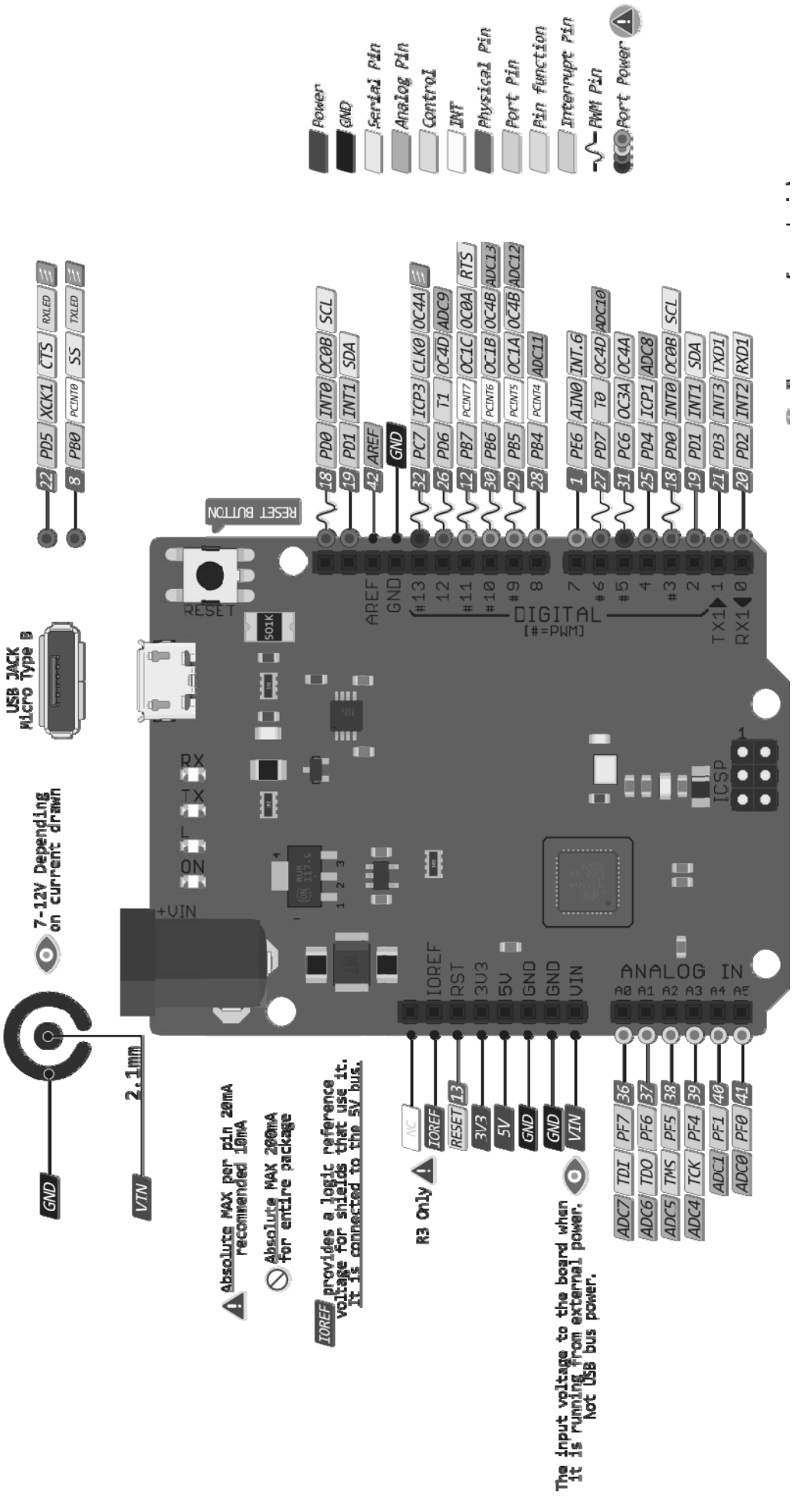

Рисунок 5.2 – Обозначение контактов (пинов) на плате Arduino и их связь с выводами микроконтроллера ATmega32U4

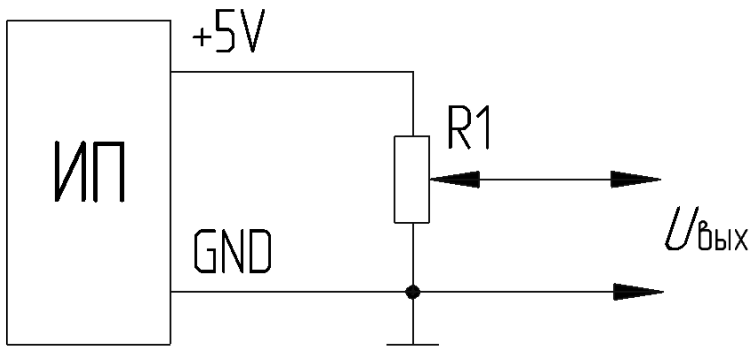

Рисунок 5.3 – Схема подключения потенциометра к источнику питания

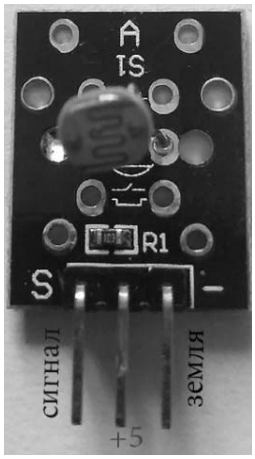

Рисунок 5.4 – Внешний вид и назначение выводов фоторезистивного аналогового датчика KY-018

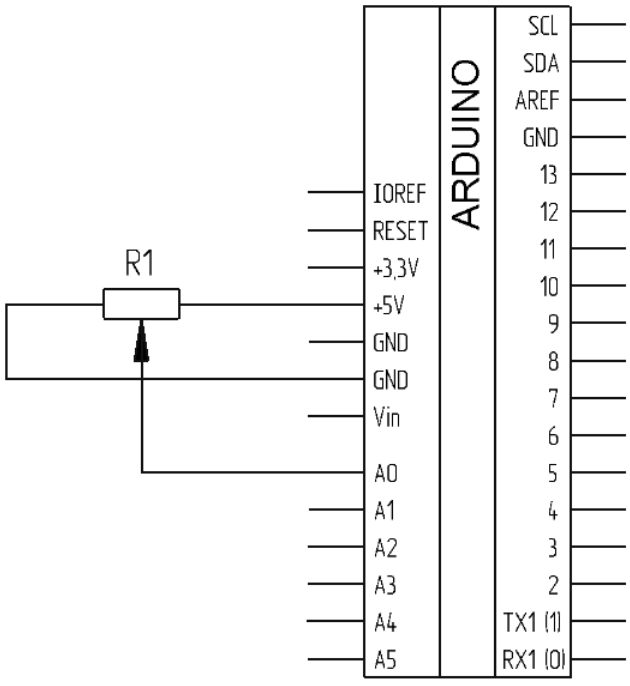

Рисунок 5.5 – Схема подключения потенциометра к плате Arduino

Один из крайних выводов потенциометра соедините с землей (GND) Arduino, второй крайний вывод потенциометра соедините с выводом 5V Arduino, третьим соедините аналоговый вход A0 и средний вывод потенциометра.

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 5V, 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 следует подавать на аналоговый вход A0 Arduino.

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.

5.2.1.2 Подключите плату Arduino к ПК через USB-порт.

5.2.1.3 Запишите программу ReadAnalogVoltage 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 создаем переменную **sensorValue** для хранения значения напряжения (которое может изменяться от 0 до 1023), которое приходит с потенциометра. Лучше всего подойдет тип **int**:

int sensorValue = analogRead(A0);

Эта информация отразится на мониторе в виде десятичных (DEC) значений. Можно сделать это посредством команды Serial.Println() с помощью следующей строки кода:

Serial.println(sensorValue, DEC)

Открыв **Serial Monitor** 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

```
/*
  AnalogReadSerial
Считывает аналоговые значения с вывода A0, выводит значения на монитор.
*/
// установки:
void setup()
{
```

```
 // инициализируем последовательную передачу данных со скоростью
 9600 бит в секунду:
  Serial.begin(9600);
}
// основной цикл:
void loop()
{
  // читаем значение на аналоговом входе A0:
  int sensorValue = analogRead(A0);
  // выводим на монитор считанное значение:
  Serial.println(sensorValue);
  delay(10);     // задержка в промежутке между считываниями для
стабильности:
}
```

5.2.1.4 Вращая ручку потенциометра, наблюдайте на экране дисплея ПК поток цифр от 0 до 1023, соответствующих положению ручки потенциометра *R*1.

5.2.2 Исследование устройства с фоторезистивным датчиком KY-018.

5.2.2.1 В данной работе с помощью фоторезистивного датчика KY-018 будем управлять включением/выключением светодиода, расположенного на плате Arduino и подключенного к её пину 13. Схема подключения фоторезистивного датчика KY-018 к плате Arduino приведена на рисунке 5.6.

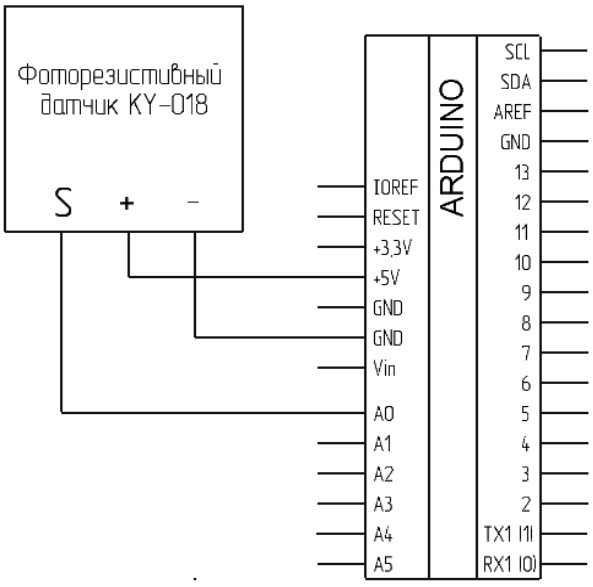

Рисунок 5.6 – Схема подключения фоторезистивного датчика KY-018 к плате Arduino

Алгоритм работы устройства следующий.

1 Определяется (программно) уровень сигнала, поступающего на вход A0 Arduino.

2 Сравнивается полученное значение сигнала с пороговым значением. Максимальное значение будет соответствовать темноте, минимальное –

максимальной освещенности. В программе задается пороговое значение val_porog = 300.

3 Если уровень сигнала окажется меньше порогового, включается светодиод. Иначе – светодиод выключается.

В соответствии с этим алгоритмом составлена программа «SVET1».

Полный текст программы «SVET1»

```
#define PIN_LED 13
#define PIN_PHOTO_SENSOR A0
void setup() {
  Serial.begin(9600);
  pinMode(PIN_LED, OUTPUT);
}
void loop() {
  int val_porog = analogRead(PIN_PHOTO_SENSOR);
  Serial.println(val);
  if (val_porog < 300) {
    digitalWrite(PIN_LED, LOW);
  } else {
    digitalWrite(PIN_LED, HIGH);
  }
}
```

5.2.2.2 С помощью трех проводов подключите датчик KY-018 к плате Arduino (см. рисунок 5.6): вывод «-» (земля) – к выводу GND платы Arduino; вывод «+» (+5) – к выводу +5V платы Arduino; вывод «S» (сигнал) – к аналоговому входу A0 платы Arduino.

5.2.2.3 Подключите плату Arduino к ПК через USB-порт.

5.2.2.4 Запишите программу «SVET1» в память контроллера Arduino, переслав её из ПК.

Прикрывая фоторезистор (рукой или светонепроницаемым предметом), наблюдайте за включением/выключение светодиода, расположенного на плате Arduino.

5.2.2.5 Изменяя в программе пороговый параметр val_porog, повторите действия, описанные в п. 5.2.2.4.

5.2.2.6 Яркость свечения светодиода можно менять, используя широтно-импульсную модуляцию (ШИМ) сигнала, посылая с помощью команды analogWrite() на пин 13 Arduino значения от 0 до 255.

Для преобразования цифрового значения уровня освещения от датчика освещенности (от 0 до 1023) в диапазон ШИМ яркости светодиода (от 0 до 255) надо использовать функцию map().

Полный текст программы «SVET2»

```
#define PIN_LED 13
#define PIN_PHOTO_SENSOR A0
void setup() {
  Serial.begin(9600);
  pinMode(PIN_LED, OUTPUT);
}
void loop() {
  int val_porog = analogRead(PIN_PHOTO_SENSOR);
  Serial.println(val_porog);
  int ledPower = map(val_porog, 0, 1023, 0, 255); // Преобразуем полученное значение в уровень ШИМ-сигнала. Чем меньше значение освещенности, тем больше мощности мы должны подавать на светодиод через ШИМ.
  analogWrite(PIN_LED, ledPower); // Меняем яркость
}
```

Примечание – В случае другого способа подключения фоторезистивного датчика, при котором сигнал с A0 пропорционален степени освещенности, надо пороговое значение изменить, вычитая его из максимального:

int val_porog = 1023 – analogRead(PIN_PHOTO_RESISTOR).

5.2.2.7 Запишите программу «SVET2» в память контроллера Arduino, переслав её из ПК.

Прикрывая фоторезистор (рукой или светонепроницаемым предметом), наблюдайте за свечением светодиода, расположенного на плате Arduino.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, схемы подключения потенциометра и фоторезистивного датчика к плате 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?

5 Как осуществляется управление светодиодом с помощью фоторезистивного датчика?

6 Лабораторная работа № 6. Разработка и исследование программного обеспечения на C++ для ввода данных от дискретных датчиков

Цель работы: исследовать работу цифрового прибора на микроконтроллере с дискретным датчиком.

6.1 Основные теоретические положения

В качестве дискретных датчиков будем использовать кнопку KY-004 (рисунок 6.1) и датчик наклона KY-027 (рисунок 6.2).

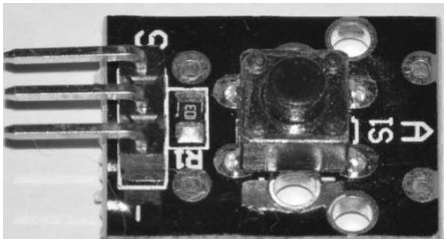

Рисунок 6.1 – Внешний вид кнопки KY-004 в качестве дискретного датчика

Рисунок 6.2 – Внешний вид датчика наклона KY-027

При нажатии кнопки соединяются две точки цепи. Схема её подключения к плате Arduino приведена на рисунке 6.3.

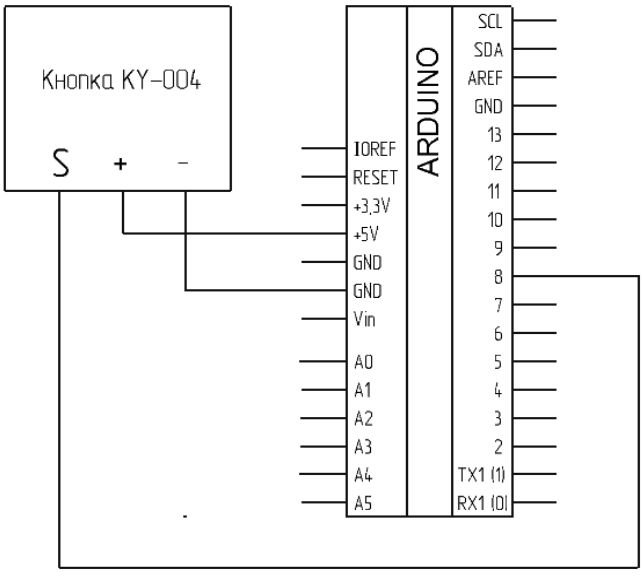

Рисунок 6.3 – Схема подключения кнопки KY-004 к плате Arduino

Когда кнопка не нажата, нет никакой связи между двумя ее выводами, так что нижний её вывод подключен к общей шине через резистор $R1$ и на пин 8 подается сигнал LOW, или логический нуль. При нажатии на кнопку устанавливается соединение между двумя выводами кнопки и на пин 8 подается +5 В, он переходит в состояние HIGH, или логическую единицу.

Датчик наклона KY-027 представляет собой электронное устройство, которое определяет ориентацию объекта в пространстве и, соответственно, выдает на выходе сигнал высокого логического уровня (HIGH или «1») или низкого (LOW или «0»). В принципе его работы лежит тот факт, что в нем есть ртутный шарик, который перемещается и замыкает два контакта. Таким образом, датчик наклона может включать или выключать схему в зависимости от ориентации объекта в пространстве. Схема подключения датчика наклона KY-027 к плате Arduino приведена на рисунке 6.4.

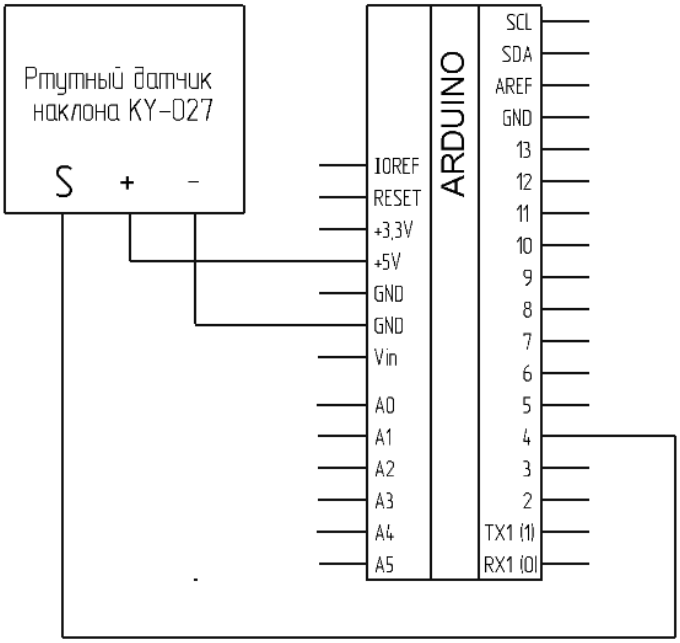

Рисунок 6.4 – Схема подключения датчика наклона KY-027 к плате Arduino

6.2 Порядок выполнения работы

6.2.1 Соедините проводниками кнопку (датчик KY-004) и плату Arduino в соответствии с рисунком 6.3.

6.2.2 Подключите плату Arduino к ПК через USB-порт.

6.2.3 Запишите программу DigitalSensor1 в память контроллера Arduino, переслав её из ПК. В этой программе прежде всего нужно в функции настройки включить последовательную передачу данных между ПК и Arduino со скоростью 9600 бит в секунду следующей строкой:

Serial.begin(9600);

Далее инициализируйте пин 8. Он должен читать состояние кнопки, чтобы быть настроенным как вход:

pinMode(8,INPUT);

Далее, когда установки завершены, следует перейти к основному циклу кода. Когда кнопка нажата, +5 В будут свободно проходить через цепь, а когда она не нажата, вход будет соединен с общей шиной через резистор *R*1 (10 кОм). Это цифровой вход, следовательно, он может быть только во включенном (читается контроллером Arduino как 1 или HIGH) или в выключенном состоянии (читается контроллером Arduino как 0 или LOW). Промежуточных состояний не существует.

Первое, что нужно сделать в основном цикле, – это создать переменную для хранения информации, поступающей от кнопки. Поскольку сигнал, приходящий от кнопки, может иметь только значения 1 или 0, можно использовать целочисленный тип данных **int**. Назовем эту переменную **sensorValue** и присвоим ей значение, равное значению, считанному с пина 8. Это можно сделать с помощью следующей строки:

int sensorValue = digitalRead(8);

Как только Arduino считает значение на пине 8, передаем его в ПК и выводим на экран как десятичное значение. Это можно сделать, используя команду Serial.println(), как показано в следующей строке:

Serial.println(sensorValue);

При открытии монитора последовательной передачи (**Serial Monitor** в программной среде Arduino) можно увидеть последовательность нулей, когда кнопка разомкнута, и последовательность единиц, когда кнопка замкнута.

Полный текст программы DigitalSensor1

```
/* Читает состояние пина 8, выводит результат на монитор */
// цифровой пин 8 присоединен к кнопке. Назовем его:
int pushButton = 8;
// проведем необходимые установки:
void setup()
{
// инициализируем последовательную передачу данных со скоростью 9600 бит в секунду:
Serial.begin(9600);
// назначим пин 8 входом:
pinMode(pushButton, INPUT);
}
// основной цикл:
void loop()
{
// читаем значение на пине 8:
int buttonState = digitalRead(pushButton);
// выводим значение на монитор:
Serial.println(buttonState);
```

delay(10); // задержка для стабильного считывания
}

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

6.2.5 Соедините проводниками датчик наклона KY-027 и плату Arduino в соответствии с рисунком 6.4.

6.2.6 Подключите плату Arduino к ПК через USB-порт.

6.2.7 Запишите программу DigitalSensor2 в память контроллера Arduino, переслав её из ПК.

Полный текст программы DigitalSensor2

```
/* Читает состояние пина 4, выводит результат на монитор */
// цифровой пин 4 присоединен к датчику наклона. Назовем его:
int Sensor2 = 4;
// проведем необходимые установки:
void setup()
{
// инициализируем последовательную передачу данных со скоростью 9600 бит в секунду:
Serial.begin(9600);
// назначим пин 4 входом:
pinMode(Sensor2, INPUT);
}
// основной цикл:
void loop()
{
// читаем значение на пине 4:
int Sensor2State = digitalRead(Sensor2);
// выводим значение на монитор:
Serial.println(Sensor2State);
delay(10);  // задержка для стабильного считывания
}
```

6.2.8 Делая наклоны датчика KY-027 так, чтобы ртутный шарик перемещался внутри стеклянной колбы и замыкал или размыкал контакты, также расположенные внутри колбы, наблюдайте на экране дисплея ПК появление 0 или 1, соответствующих положению датчика наклона.

Содержание отчета

Отчет о проделанной работе должен содержать название работы, цель работы, схемы подключения дискретных датчиков к плате Arduino, выводы по результатам экспериментальных исследований.

Контрольные вопросы

1 Какое напряжение соответствует логическому 0?
2 Какое напряжение соответствует логической 1?
3 Какую функцию выполняет ртутный шарик в датчике наклона?

7 Лабораторная работа № 7. 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.

7.1 Основные теоретические положения

На рисунке 7.1 показан жидкокристаллический (ЖК) дисплей WM-C1602N. Он имеет 2 строки по 16 символов. Дисплей работает под управлением встроенного контроллера.

Рисунок 7.1 – Внешний вид ЖК-дисплея WM-C1602N

Назначение выводов у этого дисплея следующее:
− 1 (VSS) − питание контроллера (-);
− 2 (VDD) − питание контроллера (+);
− 3 (VO) − вывод управления контрастом;
− 4 (RS) − выбор регистра;
− 5 (R/W) − чтение/запись (режим записи при соединении с землей);
− 6 (E) − enable (строб по спаду);
− 7–10 (DB0-DB3) − младшие биты 8-битного интерфейса;
− 11–14 (DB4-DB7) − старшие биты интерфейса;
− 15 (A) − анод (+) питания подсветки;
− 16 (K) − катод (-) питания подсветки.

Схема подключения к Arduino ЖК-дисплея WM-C1602N (*HG*1) показана на рисунке 7.2.

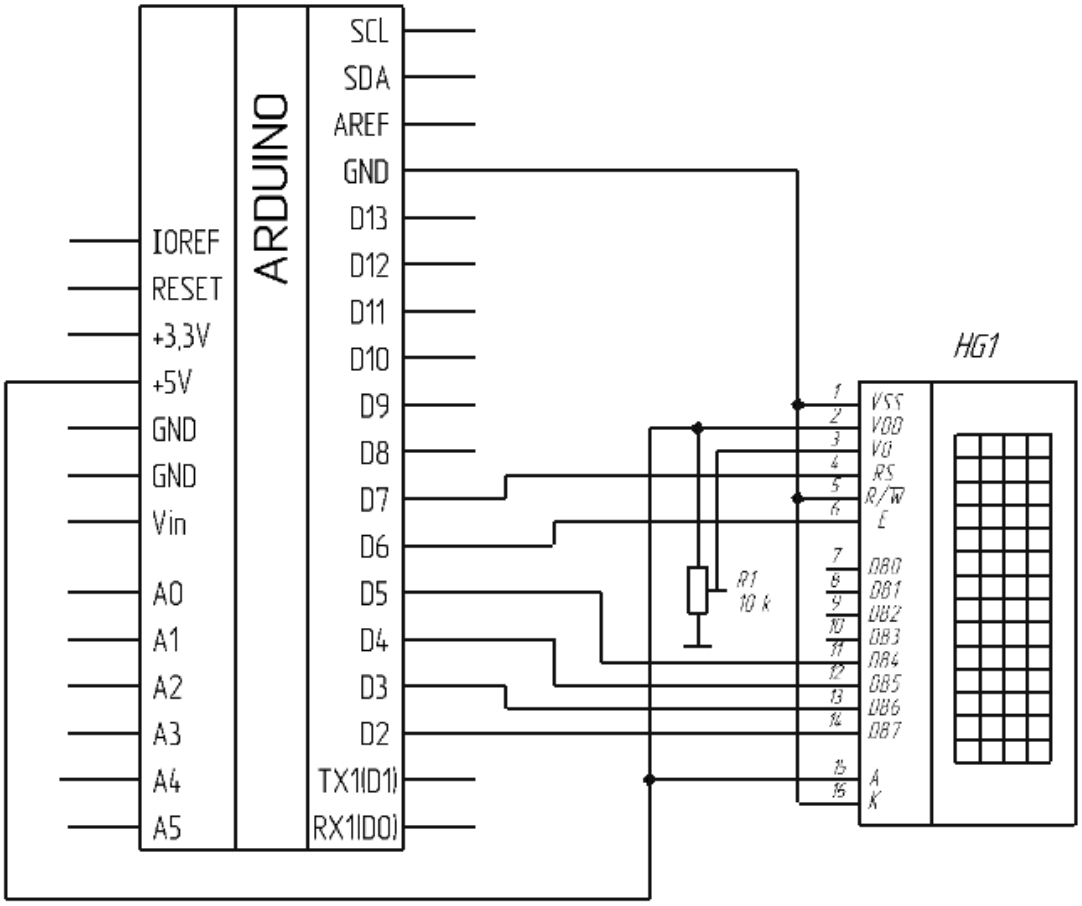

Рисунок 7.2 − Схема подключения к Arduino ЖК-дисплея WM-C1602N

На рисунке 7.3 показана микросхема ЖК-дисплея WM-C1602N с клавиатурой.

Рисунок 7.3 – Внешний вид ЖК-дисплея WM-C1602N с клавиатурой

Сочетание в одном модуле ЖК-дисплея, принимающего отображаемую информацию по шине 4 бит, и клавиатуры упрощает разрабатываемый прибор. Применение модуля для сборки передней панели прибора упрощает его конструкцию. ЖК-дисплей имеет регулировку подсветки. Предусмотрены отверстия для установки на переднюю панель прибора.

При подаче питания ЖК-дисплей 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).

7.2 Порядок выполнения работы

7.2.1 Подключите ЖК-дисплей к плате Arduino.

7.2.2 Подключите плату Arduino к ПК через USB-порт.

7.2.3 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 WM-C1602N, результаты экспериментальных исследований, выводы.

Контрольные вопросы

1 Как осуществляется управление работой дисплеев в микроконтроллерных системах?

2 Опишите устройство и принцип работы ЖК-дисплея.

8 Лабораторная работа № 8. Разработка и исследование программного обеспечения на C++ для управления исполнительными устройствами

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

8.1 Основные теоретические положения

8.1.1 **Реле SRD-05VDC**. Программное управление мощными исполнительными устройствами цифрового устройства может быть реализовано с помощью электромагнитного реле. На рисунке 8.1 показан 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, представленная контактами A1, A2, и управляемая цепь, представленная контактами 1–3. Цепи никак не связаны между собой (рисунок 8.2).

Рисунок 8.1 – Внешний вид реле SRD-05VDC

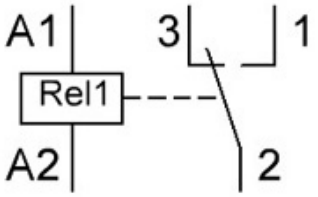

Рисунок 8.2 – Коммутационная схема реле SRD-05VDC

Релейный модуль имеет три вывода:
1) VCC: "+" питания;
2) GND: "-" питания;
3) IN: вывод входного сигнала.
Подключение модуля к плате Arduino:
− VCC на + 5 V на Arduino;
− GND на любой из GND выводов Arduino;
− IN на любой из цифровых входов/выходов Arduino.
Схема подключения реле к контроллеру Arduino приведена на рисунке 8.3.

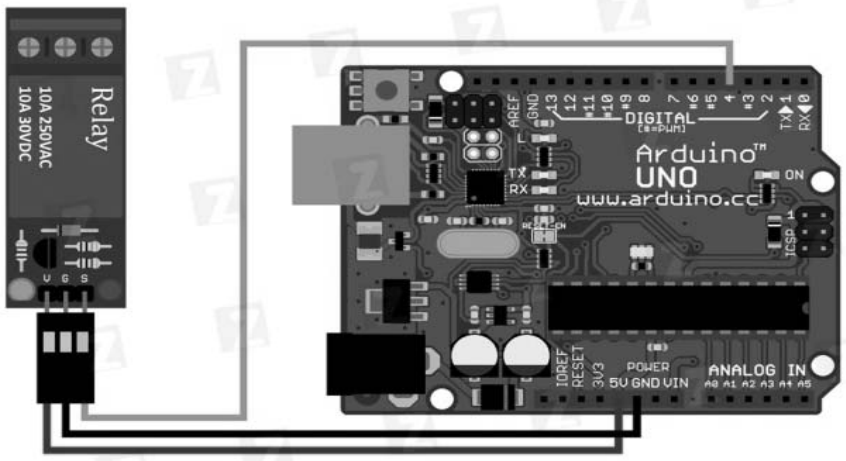

Рисунок 8.3 − Схема подключения реле к контроллеру Arduino

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);
}
```

8.1.2 **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 надо подавать от внешнего источника.

Простым вариантом управления электродвигателем является использование реле. Схема подключения электродвигателя М1 к контроллеру Arduino через реле К1 приведена на рисунке 8.4.

Код программы «Motor1» обеспечивает периодическое включение электродвигателя М1 на 8 с. и выключение на 4 с.

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);
}
```

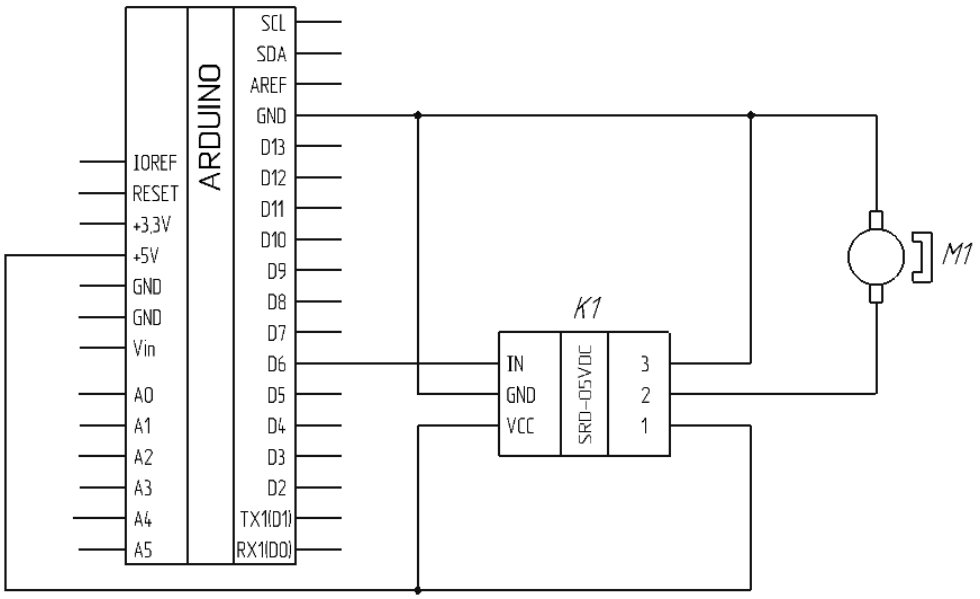

Рисунок 8.4 – Схема подключения электродвигателя к контроллеру Arduino через реле

Также можно реализовать работу электродвигателя М1 в реверсивном режиме (обеспечить вращение вала в обе стороны). Для этого нужно использовать два реле: К1 и К2 (рисунок 8.5).

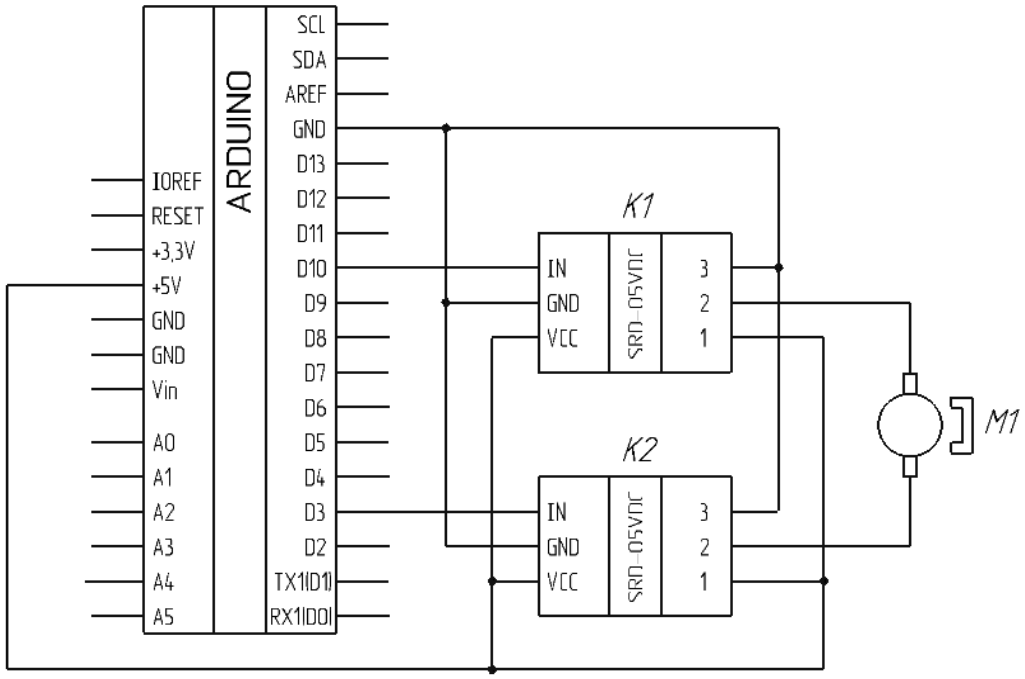

Рисунок 8.5 – Схема подключения электродвигателя к контроллеру Arduino, обеспечивающая реверс вращения его вала

Код программы, обеспечивающий реверсивный режим работы электродвигателя, представлен далее.

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

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

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

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

8.2.4 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.

4 В каких случаях совместно с контроллером используется реле?

Список литературы

1 **Гусев, В. Г.** Электроника и микропроцессорная техника: учебник / В. Г. Гусев, Е. М. Гусев. – Москва: КноРус, 2022. – 798 с.

2 **Шишкин, Г. Г.** Электроника: учебник для бакалавров / Г. Г. Шишкин, А. Г. Шишкин. – 2-е изд., испр. и доп. – Москва: Юрайт, 2019. – 703 с.