МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫХ СИСТЕМ

Методические рекомендации
к выполнению лабораторных работ по дисциплине
для студентов очной формы обучения по направлению
12.04.01 «Приборостроение»

Севастополь
СевГУ
2024

УДК 004.45:681.518.3(076.5)
ББК 32.973-018.2я73
П784

Р е ц е н з е н т :
А. Ю. Тараховский – канд. техн. наук,
аведующий кафедрой «Цифровое проектирование»

Составители: А. И. Балакин, Н. А. Балакина

П784 Программное обеспечение информационно-измерительных систем : методические рекомендации к выполнению лабораторных работ по дисциплине для студентов очной формы обучения по направлению 12.04.01 «Приборостроение» / Севастопольский государственный университет ; сост.: А. И. Балакин, Н. А. Балакина. – Севастополь : СевГУ, 2024. – 32 с. – Текст : электронный.

Предназначены для учебно-методического обеспечения дисциплины «Программное обеспечение информационно-измерительных систем».

Целью методических рекомендаций является оказание помощи студентам при выполнении лабораторных работ по дисциплине «Программное обеспечение информационно-измерительных систем», овладение практическими навыками создания программного обеспечения для измерительных устройств.

УДК 004.45:681.518.3(076.5)
ББК 32.973-018.2я73

Методические рекомендации рассмотрены и утверждены на заседании кафедры «Приборные системы и автоматизация технологических процессов», протокол № 4 от 30 ноября 2023 г.

Рекомендованы в качестве методических рекомендаций для студентов очной формы обучения по направлению 12.04.01 «Приборостроение».

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1
ОСНОВЫ РАБОТЫ С ARDUINO

***Цель работы*:** Получение первичных навыков программирования платформы Arduino на базе микроконтроллера Atmega.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

Функция setup

Функция setup() вызывается, когда стартует скетч. Используется для инициализации переменных, определения режимов работы выводов, запуска используемых библиотек и т.д. Функция setup запускает только один раз, после каждой подачи питания или сброса платы Arduino.

Функция Loop

После вызова функции setup(), которая инициализирует и устанавливает первоначальные значения, функция loop(), выполняется в цикле, позволяя вашей программе совершать вычисления и реагировать на них. Используется для активного управления платой Arduino.

Цифровые выводы

Выводы платформы Arduino могут работать как входы или как выходы.

Выводы Arduino (Atmega) стандартно настроены как порты ввода, таким образом, не требуется явной декларации в функции pinMode(). Сконфигурированные порты ввода находятся в высокоимпедансном состоянии. Это означает то, что порт ввода дает слишком малую нагрузки на схему, в которую он включен. Эквивалентом внутреннему сопротивлению будет резистор 100 МОм подключенный к выводу микросхемы. Таким образом, для перевода порта ввода из одного состояния в другое требуется маленькое значение тока. Это позволяет применять выводы микросхемы для подключения емкостного датчика касания, фотодиода, аналогового датчика со схемой, похожей на RC-цепь.

С другой стороны, если к данному выводу ничего не подключено, то значения на нем будут принимать случайные величины, наводимые электрическими помехами или емкостной взаимосвязью с соседним выводом.

Если на порт ввода не поступает сигнал, то в данном случае рекомендуется задать порту известное состояние. Это делается добавлением подтягивающих резисторов 10 кОм, подключающих вход либо к +5 В (подтягивающие к питанию резисторы), либо к земле (подтягивающие к земле резисторы).

Микроконтроллер Atmega имеет программируемые встроенные подтягивающие к питанию резисторы 20 кОм. Программирование данных резисторов осуществляется следующим образом.

```
pinMode(pin, INPUT);          // назначить выводу порт ввода
digitalWrite(pin, HIGH);      // включить подтягивающий резистор
```

Подтягивающий резистор пропускает ток достаточный для того, чтобы слегка светился светодиод подключенный к выводу, работающему как порт ввода. Также легкое свечение светодиодов означает то, что при программировании вывод не был настроен как порт вывода в функции pinMode().

Подтягивающие резисторы управляются теми же регистрами (внутренние адреса памяти микроконтроллера), что управляют состояниями вывода: HIGH или LOW. Следовательно, если вывод работает как порт ввода со значением HIGH, это означает включение подтягивающего к питанию резистора, то конфигурация функцией pinMode() порта вывода на данном выводе микросхемы передаст значение HIGH. Данная процедура работает и в обратном направлении, т.е. если вывод имеет значение HIGH, то конфигурация вывода микросхемы как порта ввода функцией pinMode() включит подтягивающий к питанию резистор.

Примечание: Затруднительно использовать вывод микросхемы 13 в качестве порта ввода из-за подключенных к нему светодиода и резистора. При подключении подтягивающего к питанию резистора 20 кОм на вводе будет 1.7 В вместо 5 В, т.к. происходит падение напряжения на светодиоде и включенном последовательно резисторе. При необходимости использовать вывод микросхемы 13 как цифровой порт ввода требуется подключить между выводом и землей внешний подтягивающий резистор.

Выводы, сконфигурированные как порты вывода, находятся в низкоимпедансном состоянии. Данные выводы могут пропускать через себя достаточно большой ток. Выводы микросхемы Atmega могут быть источником (положительный) или приемником (отрицательный) тока до 40 мА для других устройств. Такого значения тока достаточно чтобы подключить светодиод (обязателен последовательно включенный резистор), датчики, но недостаточно для большинства реле, соленоидов и двигателей.

Короткие замыкания выводов Arduino или попытки подключить энергоемкие устройства могут повредить выходные транзисторы вывода или весь микроконтроллер Atmega. В большинстве случаев данные действия приведут к отключению вывода на микроконтроллере, но

остальная часть схемы будет работать согласно программе. Рекомендуется к выходам платформы подключать устройства через резисторы 470 Ом или 1 кОм, если устройству не требуется больший ток для работы.

2. ЗАДАНИЯ И ПОРЯДОК ИХ ВЫПОЛНЕНИЯ

Задание 1. Запрограммировать мигание светодиода.
Собрать схему, представленную на рисунке 1.

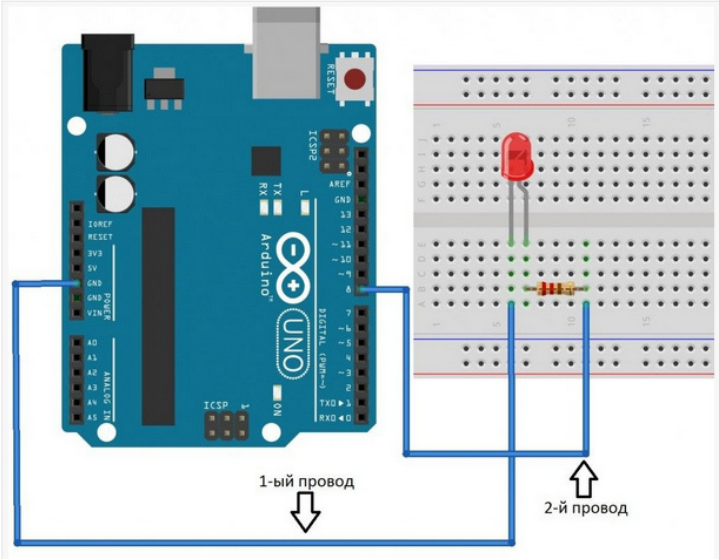


Рисунок 1 – Схема подключения светодиода при использовании 8 пина.

На рисунке1 приведена схема подключения при использовании 8 пина.

В зависимости от номера варианта подключить светодиод к соответствующему выходу Arduino. Написать программу, позволяющую задать время свечения светодиода и временя между включениями в соответствии с вариантом.

Пример программы приведен ниже:

```
int led = 8;  //объявление переменной целого типа, содержащей номер порта к которому мы подключили второй провод
void setup()  //обязательная процедура setup, запускаемая в начале программы; объявление процедур начинается словом void
{
pinMode(led, OUTPUT); //объявление используемого порта, led - номер порта, второй аргумент - тип использования порта - на вход (INPUT) или на выход (OUTPUT)
}
```

void loop() //обязательная процедура *loop*, запускаемая циклично после процедуры *setup*

{

digitalWrite(led, HIGH); //эта команда используется для включения или выключения напряжения на цифровом порте; *led* - номер порта, второй аргумент - включение (*HIGH*) или выключение (*LOW*)

delay(1000); //эта команда используется для ожидания между действиями, аргумент - время ожидания в миллисекундах

digitalWrite(led, LOW);

delay(1000);

}

Таблица 1 – Варианты заданий

Номер варианта	1	2	3	4	5	6	7	8	9	10
Номер вывода	2	3	4	5	6	7	8	9	10	11
Продолжительность импульса, с	1	2	0,5	0,1	1,5	0,3	0,4	1,2	1,8	1,9
Время между импульсами, с	2	0,5	1	0,3	0,8	0,4	1,2	1,3	0,9	0,6

Задание 2. Запрограммировать работу светодиода при включении кнопки

Схема подключения приведена на рисунке 2. Выводы для подключения светодиода использовать в соответствии с заданием 1. Кнопку подключить к любому свободному входу.

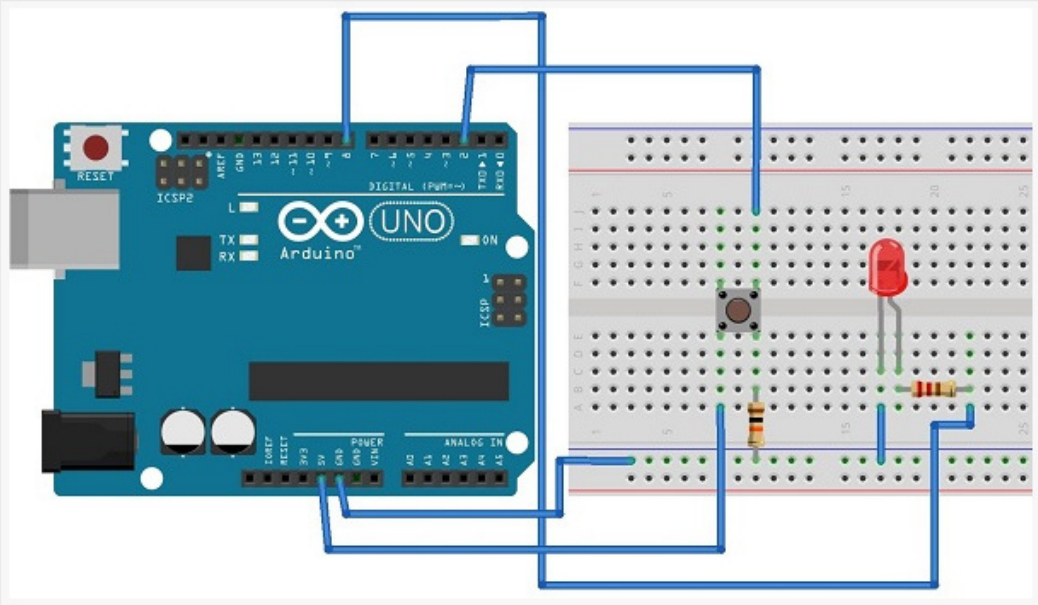


Рисунок 2 – Схема подключения кнопки для управления светодиодом

Пример программы приведен ниже.

```
int button = 2;
int led = 8;
void setup() {
pinMode(led, OUTPUT);
pinMode(button, INPUT);
}
void loop(){
if (digitalRead(button) == HIGH) {
digitalWrite(led, HIGH);
}
else {
digitalWrite(led, LOW);
}
}
```

Задание 3. Запрограммировать мигание светодиода при включении кнопки.

Схему подключения использовать из задания 2.

3. СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

Отчет по лабораторной работе должен содержать:

- титульный лист;
- наименование и цель работы;
- схемы подключения элементов и устройств к Arduino;
- программы для выполнения соответствующих заданий;
- вывод.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение функция setup?
2. Назначение функция Loop?
3. В каком состоянии находятся сконфигурированные порты ввода?
4. Команды для работы с цифровыми портами?
5. Назначение резистора при подключении светодиода?
6. Из каких элементов состоит ваша программа?
7. Как работает ваша программа?

ЛАБОРАТОРНАЯ РАБОТА №2
РАБОТА С ПОСЛЕДОВАТЕЛЬНЫМ ПОРТОМ

Цель работы: Научится работать с последовательным портом платформы Arduino.

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Для связи микроконтроллера с компьютером чаще всего применяют COM-порт.

Для запуска работы порта UART служит функция Serial.begin(). Единственный ее параметр – это скорость. Скорость необходимо указывать на передающей и приемной стороне заранее, так как протокол передачи асинхронный.

Для записи значения в порт используются три функции:

Serial.write() – записывает в порт данные в двоичном виде.

Serial.print() – может иметь много значений, но все они служат для вывода информации в удобной для человека форме. Например, если информация, указанная как параметр для передачи, выделена кавычками – терминальная программа выведет ее без изменения. Если вы хотите вывести какое-либо значение в определенной системе исчисления, то необходимо добавить служебное слово: *BIN*-двоичная, *OCT* – восьмеричная, *DEC* – десятичная, *HEX* – шестнадцатеричная. Например, Serial.print(25,HEX).

Serial.println() делает то же, что и Serial.print(), но еще переводит строку после вывода информации.

Для проверки работы программы необходимо, чтобы на компьютере была терминальная программа, принимающая данные из COM-порта. В Arduino IDE такая программа имеется рисунок 1.

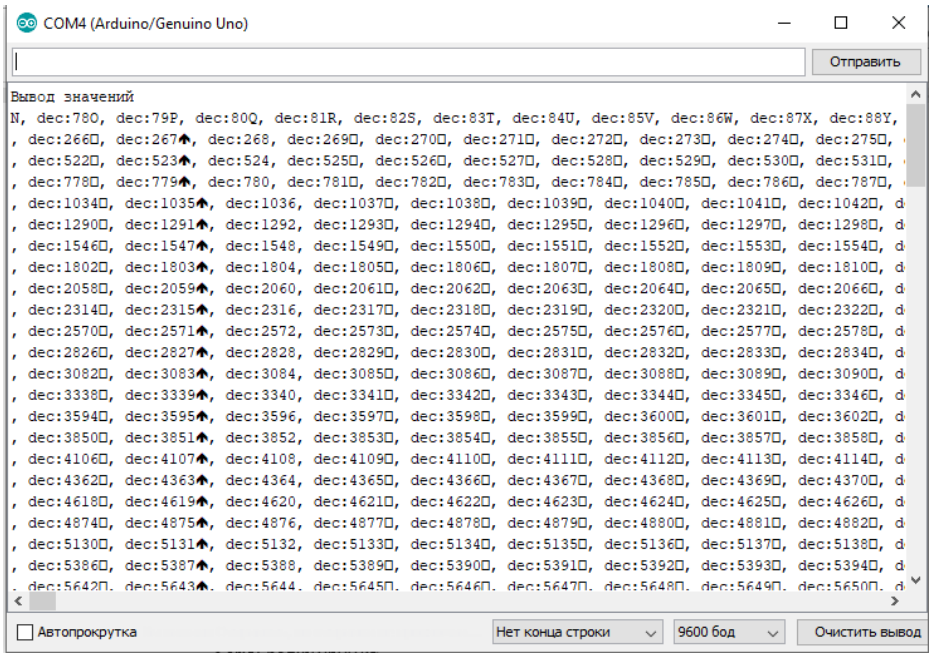


Рисунок 1 – Пример монитора последовательного порта в программе Arduino IDE

Для ее вызова выберите в меню Сервис->Монитор порта.

2. ЗАДАНИЯ И ПОРЯДОК ИХ ВЫПОЛНЕНИЯ

Задание 1. Загрузить на плату код, выводящий таблицу *ASCII*.

ASCII представляет собой кодировку для представления десятичных цифр, латинского и национального алфавитов, знаков препинания и управляющих символов.

```
int symbol = 33;

void setup() {
  Serial.begin(9600);
  Serial.println("ASCII Table ~ Character Map");
}

void loop() {
  Serial.write(symbol);
  Serial.print(", dec: ");
  Serial.print(symbol);
  Serial.print(", hex: ");
  Serial.print(symbol, HEX);
  Serial.print(", oct: ");
  Serial.print(symbol, OCT);
```

```
    Serial.print(", bin: ");
    Serial.println(symbol, BIN);
    if(symbol == 126) {
      while(true) {
        continue;
      }
    }
    symbol++;
  }
```

Переменная symbol хранит код символа. Таблица начинается со значения 33 и заканчивается на 126, поэтому изначально переменной symbol присваивается значение 33.

Следующая часть кода:

```
if(symbol == 126) {
   while(true) {
     continue;
   }
 }
```

предназначена для остановки выполнение программы. Если ее исключить, то таблица будет выводиться бесконечно.

Задание 2. Написать бесконечный цикл, который будет раз в секунду отправлять в последовательный порт имя с новой строки.

Отправка команд с ПК

При отправке информации с ПК устройству, данные помещаются в специальный раздел памяти. Как только устройство готово – оно вычитывает данные из буфера. Проверить состояние буфера позволяет функция *Serial.avaliable()*. Эта функция возвращает количество байт в буфере. Чтобы вычитать эти байты необходимо воспользоваться функцией *Serial.read()*.

Задание 3.

Выполните пример:

```
int val = 0;
void setup() {
      Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
```

```
    val = Serial.read();
    Serial.print("I received: ");
    Serial.write(val);
    Serial.println();
  }
}
```

После того, как код будет загружен в память микроконтроллера, откройте монитор COM-порта. Введите один символ и нажмите Enter. В поле полученных данных вы увидите: *“I received: X”*, где вместо *X* будет введенный вами символ. Программа бесконечно крутится в основном цикле. В тот момент, когда в порт записывается байт функция Serial.available() принимает значение 1, то есть выполняется условие *Serial.available() > 0*. Далее функция *Serial.read()* вычитывает этот байт, тем самым очищая буфер. После чего при помощи уже известных функций происходит вывод.

Использование встроенного в Arduino IDE монитора COM-порта имеет некоторые ограничения. При отправке данных из платы в COM-порт вывод можно организовать в произвольном формате. А при отправке из ПК к плате передача символов происходит в соответствии с таблицей *ASCII*. Это означает, что, когда вы вводите, например, символ “1”, через COM-порт отправляется в двоичном виде “00110001” (то есть “49” в десятичном виде).

Задание 4. Управление устройством через COM-порт

Напишите программу, управляющую работой светодиода.

При отправке в COM-порт символа “H” происходит зажигание светодиода на выводе 8, а при отправке “L” светодиод будет гаснуть

```
int val = 0;
void setup() {
      Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
    val = Serial.read();
    if (val=='H') digitalWrite(8,HIGH);
    if (val=='L') digitalWrite(8,LOW);
  }
}
```

Задание 5. Написать программу, которая по результатам приема данных из COM-порта в основном цикле должна выполнять разные действия. Для этого выполнить проверку условий в основном цикле

```
int val = '0';
```

```
void setup() {
      Serial.begin(9600);
}
void loop() {
 if (Serial.available() > 0) {
   val = Serial.read();}
   if (val=='1') {
     digitalWrite(13,HIGH); delay (100);
     digitalWrite(13,LOW); delay (100);
   }
   if (val=='0') {
     digitalWrite(13,HIGH); delay (500);
     digitalWrite(13,LOW); delay (500);
   }
}
```

Если в мониторе порта отправить значение “1” светодиод будет мигать с частотой 5Гц. Если отправить “0” – частота изменится на 1Гц.

Задание 6. Выполнить индивидуальные задания, приведенные ниже

1. Придумайте три светодиодных эффекта, переключение между которыми можно осуществлять при отправке различных символов с ПК.
2. Напишите программу, мигающую светодиодом с частой, заданной пользователем с ПК.

3. СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

Отчет по лабораторной работе должен содержать:

- титульный лист;
- наименование и цель работы;
- схемы подключения элементов и устройств к Arduino;
- программы для выполнения соответствующих заданий;
- вывод.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение последовательного порта?
2. Особенности передачи данных по последовательному порту?
3. Как осуществляется инициализация последовательного порта?
4. Как осуществить вывод данных в последовательный порт?
5. Как осуществляется чтение данных из последовательного порта?
6. В чем отличие между командами Serial.println()и Serial.print()?

ЛАБОРАТОРНАЯ РАБОТА №3
МОДЕЛИРОВАНИЕ ШИМ

Цель работы: Получить сведения о широтно-импульсной модуляции и возможности ее применения на платформе Arduino.

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Широтно-Импульсная модуляция, сокращенно ШИМ (англ. PWM)

Широтно-Импульсная модуляция, или ШИМ, это операция получения изменяющегося аналогового значения посредством цифровых устройств. Устройства используются для получения прямоугольных импульсов - сигнала, который постоянно переключается между максимальным и минимальным значениями. Данный сигнал моделирует напряжение между максимальным значением (5 В) и минимальным (0 В), изменяя при этом длительность времени включения 5 В относительно включения 0 В. Длительность включения максимального значения называется шириной импульса. Для получения различных аналоговых величин изменяется ширина импульса. При достаточно быстрой смене периодов включения-выключения можно подавать постоянный сигнал между 0 и 5 В на светодиод, тем самым управляя яркостью его свечения.

На графике (рис. 1) зеленые линии отмечают постоянные временные периоды.

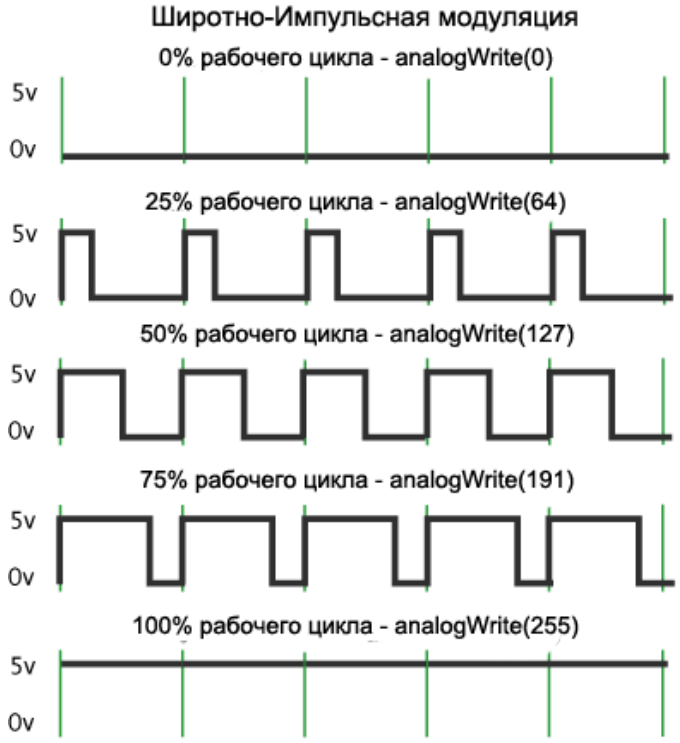


Рисунок 1 – Примеры широтно-импульсной модуляции

Длительность периода обратно пропорциональна частоте ШИМ. Т.е. если частота ШИМ составляет 500 Гц, то зеленые линии будут отмечать интервалы длительностью в 2 миллисекунды каждый. Вызов функции analogWrite() с масштабом 0 – 255 означает, что значение analogWrite(255) будет соответствовать 100% рабочему циклу (постоянное включение 5 В), а значение analogWrite(127) – 50% рабочему циклу.

2. ЗАДАНИЯ НА ЛАБОРАТОРНУЮ РАБОТУ

Задание 1. Используя ШИМ, написать программу изменения яркости светодиода с определенным шагом, задаваемым в программе.

Задание 2. Написать программу для управления яркость светодиода, подключенного через резистор к пину *D3*, используя ШИМ. Потенциометр подключен к пину *A0.*

3. СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

Отчет по лабораторной работе должен содержать:

- титульный лист;
- наименование и цель работы;
- схемы подключения элементов и устройств к Arduino;
- программы для выполнения соответствующих заданий;
- вывод.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое ШИМ?
2. Как используется ШИМ для управления устройствами?
3. Как получить различные значения аналоговой величины используя ШИМ?
4. С помощью каких команд осуществляется ШИМ в Arduino?
5. Какие выводы используются для реализации ШИМ?

ЛАБОРАТОРНАЯ РАБОТА №4
СОЗДАНИЕ ВОЛЬТМЕТРА НА БАЗЕ АРДУИНО

Цель работы: Создать вольтметр на платформе Arduino с использованием делителя напряжения.

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Плата Arduino UNO содержит 6 аналоговых входов, предназначенных для измерения напряжения сигналов. Измерение напряжение на входах производится аналого-цифровым преобразователем (АЦП) с коммутатором на 6 каналов. АЦП имеет разрешение 10 бит, что соответствует коду на выходе преобразователя 0…1023. Погрешность измерения не более 2 единиц младшего разряда. Для сохранения максимальной точности (10 разрядов) необходимо, чтобы внутреннее сопротивление источника сигнала не превышало 10 кОм. Это требование особенно важно при использовании резисторных делителей, подключенных к аналоговым входам платы.

Программные функции аналогового ввода

int analogRead(port)

Считывает значение напряжения на указанном аналоговом входе. Входное напряжение диапазона от 0 до уровня источника опорного напряжения (часто 5 В) преобразовывает в код от 0 до 1023.

При опорном напряжении равном 5 В разрешающая способность составляет 5 В / 1024 = 4,88 мВ.

```
int inputCod;          // код входного напряжения
float inputVoltage; // входное напряжение в В
inputCod= analogRead(A3); // чтение напряжения на входе A3
inputVoltage= ( (float)inputCod * 5. / 1024. ); // пересчет кода в напряжение (В)
```

void analogReference(type)

Задает опорное напряжение для АЦП. Оно определяет максимальное значение напряжения на аналоговом входе, которое АЦП может корректно преобразовать. Величина опорного напряжения также определяет коэффициент пересчета кода в напряжение:

*Напряжение на входе = код АЦП * опорное напряжение / 1024.*

Аргумент *type* может принимать следующие значения:

DEFAULT – опорное напряжение равно напряжению питания контроллера (5 В или 3,3 В).

INTERNAL – внутреннее опорное напряжение 1,1 В для плат с контроллерами ATmega168 и ATmega328, для ATmega8 – 2,56 В.

INTERNAL1V1 – внутреннее опорное напряжение 1,1 В для контроллеров Arduino Mega.

INTERNAL2V56 – внутреннее опорное напряжение 2,56 В для контроллеров Arduino Mega.

EXTERNAL – внешний источник опорного напряжения, подключается к входу *AREF*.

При использовании Arduino UNO напряжение питание равно 5 В.

```
analogReference(DEFAULT); // опорное напряжение равно 5 В
```

Если задать опорное напряжение равным 5 В, то аналоговые входы платы будут измерять напряжение в пределах 0…5 В. Если необходимо обеспечить, измерение напряжения в больших пределах, например 0…20 В, то необходимо использовать делитель напряжения.

Схема делителя напряжения приведена на рисунке 1

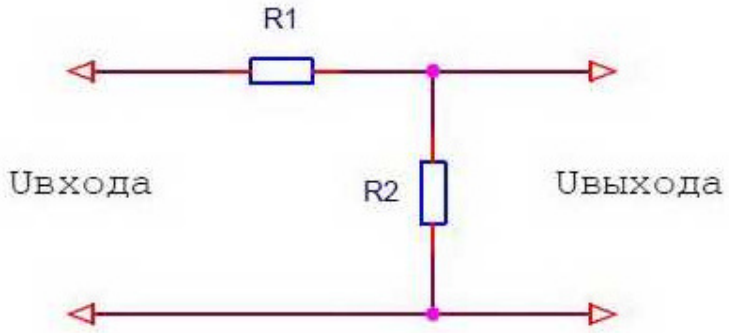


Рисунок 1. Схема делителя напряжения

Напряжение на входе и выходе делителя связаны соотношением:

$$U_{вых} = U_{вх} \cdot \frac{R_2}{(R_1 + R_2)}$$

Коэффициент передачи:

$$K = \frac{U_{вых}}{U_{вх}} = \frac{R_2}{(R_1 + R_2)}$$

2. ЗАДАНИЯ НА ЛАБОРАТОРНУЮ РАБОТУ

Задание 1. Создать устройство, измеряющее напряжения на аналоговом входе платы, и передающее измеренные значения на ЭВМ по последовательному порту. Вольтметр должен измерять напряжение в пределах не меньше 0…20 В.

Для решения данной задачи воспользуемся выражениями, приведенными выше. Коэффициент передачи в согласно заданию равен 0,25.

Для сохранения максимальной точности (10 разрядов) рекомендуется, чтобы внутреннее сопротивление источника сигнала не превышало 10

кОм. Поэтому выбираем резистор R2 равным 4,7 кОм, а затем рассчитываем сопротивление резистора R1.

Пример программы вольтметра приведен ниже

```
// программа измерения напряжения
#define MEASURE_PERIOD 500  // время периода измерения
#define R1  4.8  // сопротивление резистора R1
#define R2  1. // сопротивление резистора R2
uint32_t timer = 0;   // переменная таймера
float u1;   // измеренные напряжения
void setup() {
  Serial.begin(9600);  // инициализируем порт, скорость 9600
 }
void loop() {
  // период 500 мс
   if (millis() - timer >= MEASURE_PERIOD) { // таймер на millis()
    timer = millis(); // сброс
    u1=  ((float)analogRead(A0)) * 5. / 1024. /R2 * (R1 + R2);
      // передача данных через последовательный порт
    Serial.print("U1 = ");     Serial.println(u1, 2);
  }
}
```

Задание 2. Разработать программу двухканального вольтметра, с разными пределами измерения: 1 канал от 0 до 10 В., 2 канал от 0 до 35 В.

3. СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

Отчет по лабораторной работе должен содержать:

- титульный лист;
- наименование и цель работы;
- схемы подключения элементов и устройств к Arduino;
- программы для выполнения соответствующих заданий;
- вывод.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как осуществляется измерение напряжения с помощью микроконтроллера?
2. Что такое делитель напряжения?
3. Назначение делителя напряжения?
4. Как определить коэффициент делителя напряжения?
5. Как определить номиналы резисторов, входящих в делитель напряжения?
6. Приведите схему делителя напряжения?

ЛАБОРАТОРНАЯ РАБОТА №5
СОЗДАНИЕ И ПРОГРАММИРОВАНИЕ ИЗМЕРИТЕЛЬНОЙ СИСТЕМЫ КОНТРОЛЯ ТЕМПЕРАТУРЫ И ДАВЛЕНИЯ

Цель работы: Создать измерительное устройства на платформе Arduino для контроля температуры и давления.

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Рассмотрим создание системы измерения атмосферного давления и температуры на основе датчика BMP180 и платы Arduino Uno. Чтобы иметь возможность подключения датчика BMP180 к плате Arduino, необходимо установить библиотеку, предназначенную для работы с датчиком BMP180 (BMP180 Breakout Arduino Library).

Датчик BMP180 (3.3В, GY-68) является сенсорным датчиком, позволяющим измерить атмосферное давления и температуру окружающей среды. Технические характеристики датчика следующие:

- Напряжение питания: 3.3 В – 5 В
- Рабочий ток: 0.5 мА
- Диапазон измеряемого давления: 300 гПа – 1100 гПа
- Интерфейс: I2C
- Время срабатывания: 4.5 мс.
- Точность измерения давления: 0,1 гектопаскаль;
- Точность измерения температуры: 0,1°C;
- Габариты: 15 мм x 14 мм

Рассмотрим модуль подробно, в левой части расположен сам сенсорный датчик BMP180 фирмы Bosch. Так как датчик BMP 180, работает от 3.3В, на плате предусмотрен стабилизатор напряжения XC6206P332MR в корпус SOT-23, который выдает на выходе напряжение в 3.3В, рядом установлена обвязка стабилизатора, состоящая из двух керамических конденсаторов на 1 мкФ. Подключение осуществляется по интерфейсу I2C, линии SCL и SDA выведены на группу контактов на другой стороне модуля, туда же выведено и питание. Последние два резистора на 4.7 кОм, необходимы подтяжки линии SCL и SDA к питанию.

Назначение контактов:

- SCL — линия тактирования (Serial CLock)
- SDA — линия данных (Serial Data)
- VCC — «+» питание
- GND — «-» питание

Принципиальная схема датчика BMP180, показана на рисунке 1.

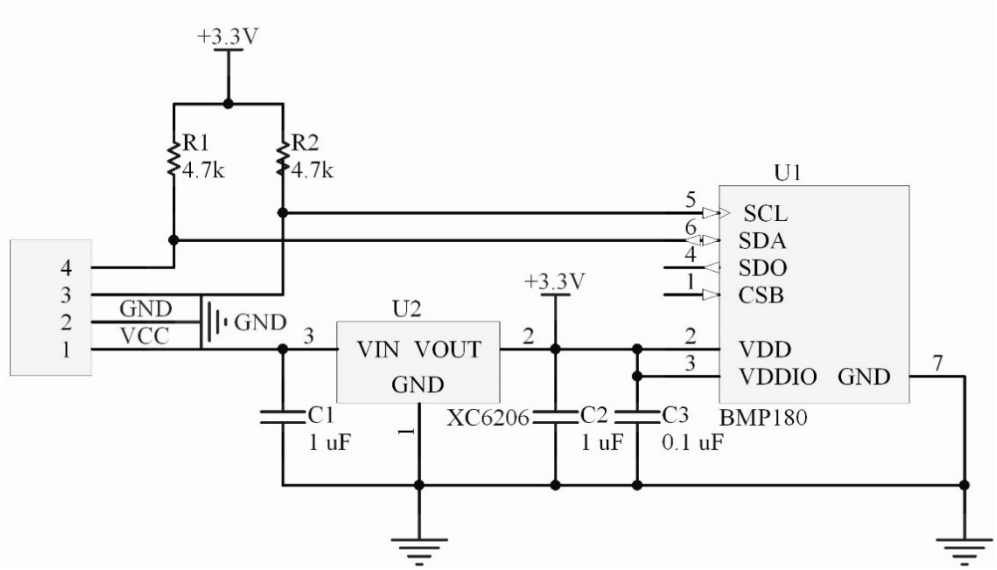


Рисунок 1 – Принципиальная схема датчика BMP180

2. ЗАДАНИЯ НА ЛАБОРАТОРНУЮ РАБОТУ

Задание 1. Подключить датчик BMP 180 к плате Arduino UNO, все полученные показания отправлять в «Serial порт».

Схема представлена на рисунке 2. Для интерфейса I2C на плате arduino предусмотрено только два вывода A4 и A5, другие вывода не поддерживают I2C.

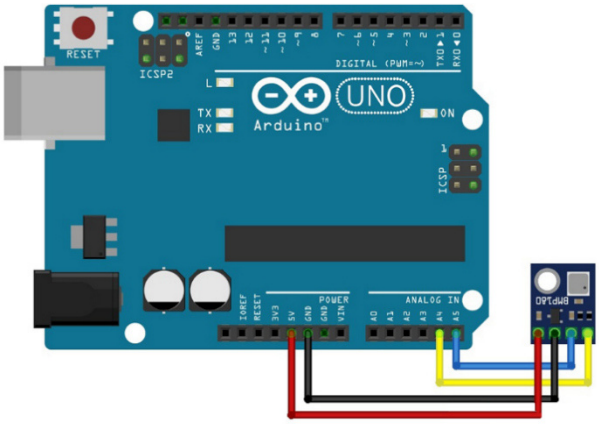


Рисунок 2 – Схема подключения датчика BMP180 к плате Arduino

Порядок выполнения лабораторной работы

Загружаем скетч в плату Arduino

```
#include <SFE_BMP180.h>
#include <Wire.h>

SFE_BMP180 pressure;  // Объявляем переменную для доступа к SFE_BMP180
```

```
void setup()
{
 Serial.begin(9600);                // Задаем скорость передачи данных
 Serial.println("REBOOT");          // Печать текста "Перезагрузка"

 if(pressure.begin())               // Инициализация датчика
   Serial.println("BMP180 init success");     // Печать текста "BMP180 подключен"
  else{                             // В противном случаи, датчик не подключен
   Serial.println("BMP180 init fail\n\n");   // Печать текста "BMP180 не подключен"
   while(1);                        // Пауза.
    }
}

void loop()
{
 char status;
 double T,P,p0,a;
```

Так как давление зависит от температуры, надо сначала узнать температуру. Считывание температуры занимает некоторое время. Если ошибок нет, функция pressure.startTemperature вернет status с количеством миллисекунд которые нужно подождать. При возникновении проблем функция вернет 0.

```
  status = pressure.startTemperature();       // Считывание показания
  if(status!=0){                    // Если значение status не 0, выполняем следующую команду.
   delay(status);                   // Ждем
   status = pressure.getTemperature(T);        // Полученные показания, сохраняем в переменную T
    if(status!=0){                  // Если ошибок нет, функция вернет 1, иначе вернет 0
      Serial.print("Temperature: ");       // Печать текста "Температура"
      Serial.print(T,2);            // Печать показания переменной "T"
      Serial.println(" C, ");       // Печать текста "C"
```

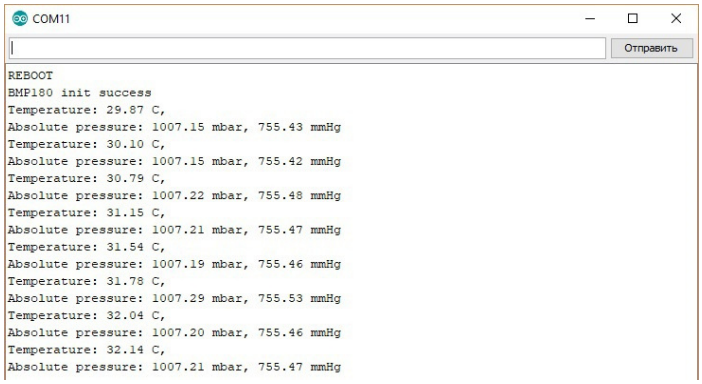
Далее определяем показания атмосферного давления. Параметр указывает расширение, от 0 до 3 (чем больше расширение, тем больше точность, тем дольше ждать)

Если ошибок нет, функция pressure.startTemperature вернет status с количеством миллисекунд которые нужно подождать. При возникновении проблем функция вернет 0.

```
status = pressure.startPressure(3);        // Считывание показания
if(status!=0){// Если значение status не 0, выполняем следующую команду.
    delay(status);                         // Ждем
    status = pressure.getPressure(P,T);          // Полученные показания, сохраняем в переменную P
     if(status!=0){ // Если ошибок нет, функция вернет 1, иначе вернет 0
       Serial.print("Absolute pressure: "); // Печать текста "Атмосферное давление"
          Serial.print(P,2);               // Печать показания переменной mBar
          Serial.print(" mbar, ");           // Печать текста "mBar"
          Serial.print(P*0.7500637554192,2);  // Печать показания в mmHg
          Serial.println(" mmHg");}          // Печать текста "mmHg"

     else Serial.println("error retrieving pressure measurement\n");}        // Ошибка получения давления
     else Serial.println("error starting pressure measurement\n");}          // Ошибка запуска получения давления
     else Serial.println("error retrieving temperature measurement\n");} // Ошибка получения температуры
     else Serial.println("error starting temperature measurement\n");       // Ошибка запуска получения температуры
     delay(5000);                                                  // Пауза в 5с
    }
```

Если все правильно подключено, в окне монитора порта, можно увидеть температуру и атмосферное давление



Задание 2. Получить значение давления и вывести их на ЖК дисплей.

Принципиальная схема подключения представлена на рисунке 3.

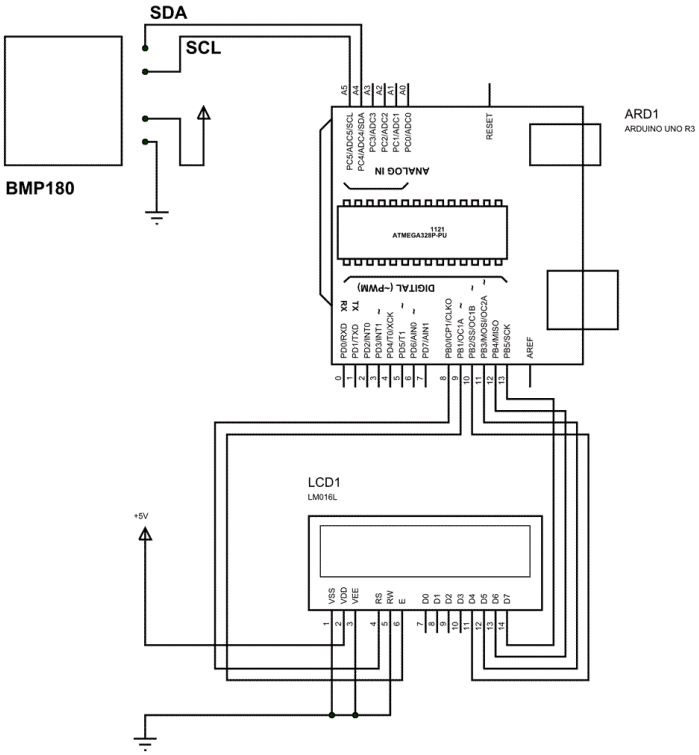


Рисунок 3 – Принципиальная схема подключения датчика BMP180 и ЖК дисплея к плате Arduino

Необходимо инициализировать ЖК дисплей и вывести на него полученные от датчика значения.

ЖК дисплей 16x2 имеет 16 контактов. Если не применять черный цвет можно использовать 14 контактов. Контакты черного цвета можно запитать или оставить "пустыми". Таким образом из 14 контактов 8 контактов данных (7-14 или D0-D7), 2 контакта питания (1&2 или VSS&VDD или GND&+5v), 3-й контакт для управления контрастностью (VEE-controls определяет насколько толстыми будут выглядеть символы на экране дисплея) и 3 контакта управления (RS&RW&E).

В схеме на рисунке 3 используется только 2 контакта управления ЖК дисплея. Контакт контраста и контакт READ/WRITE можно замкнуть на землю. Это обеспечивает ЖК дисплею максимальную контрастность и переводит его в режим чтения данных. В этом случае будет необходимо только управлять контактами ENABLE и RS чтобы передавать на ЖК дисплей символы и данные.

В схеме необходимо сделать следующие соединения платы Arduino с ЖК дисплеем:

PIN1 или VSS на землю

PIN2 или VDD или VCC к источнику питания +5 В

PIN3 или VEE на землю
PIN4 или RS (Register Selection – выбор регистра) к контакту PIN8 платы ARDUINO UNO
PIN5 или RW (Read/Write) на землю
PIN6 или E (Enable) к контакту PIN9 платы ARDUINO UNO
PIN11 или D4 к контакту PIN10 платы ARDUINO UNO
PIN12 или D5 к контакту PIN11 платы ARDUINO UNO
PIN13 или D6 к контакту PIN12 платы ARDUINO UNO
PIN14 или D7 к контакту PIN13 платы ARDUINO UNO

Программная среда ARDUINO IDE позволяет взаимодействовать с ЖК дисплеем в 4-битном режиме. Этот режим заложен в Arduino по умолчания и сокращает число контактов, необходимых для взаимодействия с ЖК дисплеем.

Для подключения датчика BMP180 к плате Arduino Uno необходимо его инициализировать:

```
#include <Adafruit_BMP085.h>
#include <Wire.h>
#include <LiquidCrystal.h>
Serial.begin(9600);
String PRESSUREVALUE = String(bmp.readPressure());
String TEMPARATUREVALUE = String(bmp.readTemperature());
```

Необходимо подключить заголовочный файл библиотеки (#include <Adafruit_BMP085.h>) чтобы получить доступ к специальным функциям для работы с датчиком BMP180.

Далее необходимо разрешить последовательное соединение (C communication) с помощью подключения заголовочного файла “#include <Wire.h>”.

После этого можно считывать значение давления с датчика с помощью вызова функции ”String PRESSUREVALUE = String(bmp.readPressure());”. В результате этой команды считанное значение давления будет сохранено в строке “PRESSUREVALUE”.

Также можно считать с датчика значение температуры с помощью вызова функции ”String TEMPARATUREVALUE = String(bmp.readTemperature());”. В результате этой команды считанное значение температуры будет сохранено в строке “TEMPARATUREVALUE”.

Также в программе необходимо подключить заголовочный файл для работы с ЖК дисплеем - #include <LiquidCrystal.h>. По умолчанию ЖК дисплей будет работать в 4-битном режиме.

Далее требуется указать плате Arduino тип ЖК дисплея, с которым мы будем работать. При использовании ЖК дисплея 16x2 эта команда будет выглядеть следующим образом: ‘lcd.begin(16,2);‘.

После этого необходимо сообщить плате Arduino номера ее контактов, к которым мы подключили ЖК дисплей. Т.е. это будет команда “LiquidCrystallcd(8, 9, 10, 11, 12, 13);”.

Текст программы выглядит следующим образом

```
#include <Adafruit_BMP085.h>
#include <Wire.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 10, 11, 12, 13); // номера контактов платы Arduino, к которым подключены контакты RS,EN,D4,D5,D6,D7 ЖК дсиплея

char PRESSURESHOW[4]; // инициализация массива символов на 4 элемента для хранения значения давления
char TEMPARATURESHOW[4]; // инициализация массива символов на 4 элемента для хранения значения температуры
 Adafruit_BMP085 bmp;

void setup()
{
 lcd.begin(16, 2);
 lcd.print("  CIRCUITDIGEST"); // вывод приветственной строки на ЖК дсиплей
 lcd.setCursor(0, 1);
 delay (2500);
 delay (2500);
 lcd.clear(); // очистить ЖК дисплей
 Serial.begin(9600); // старт последовательного порта на скорости 9600 бод/с
 if (!bmp.begin())
 {
  Serial.println("ERROR"); // если возникла ошибка при связи датчика с последовательным портом
  while (1) {}
 }
}
```

```
void loop()
{
 lcd.print("Pressure= ");
 String PRESSUREVALUE = String(bmp.readPressure()); // считывание значения давления с датчика
 PRESSUREVALUE.toCharArray(PRESSURESHOW, 4); // конвертация строки в символьный массив
 lcd.print(PRESSURESHOW); // отображение на ЖК дисплее значения давления
 lcd.print("hPa  ");
 lcd.setCursor(0, 1); // установка курсора в нулевой столбец первой строки
 lcd.print("Tempar.=");
 String TEMPARATUREVALUE = String(bmp.readTemperature()); // считывание значения температуры с датчика
 TEMPARATUREVALUE.toCharArray(TEMPARATURESHOW, 4); // конвертация строки в символьный массив
 lcd.print(TEMPARATURESHOW); // отображение на ЖК дисплее значения температуры
 lcd.print("C  ");
 lcd.setCursor(0, 0); // установка курсора в нулевой столбец нулевой строки
 delay(1000);
}
```

3. СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

Отчет по лабораторной работе должен содержать:

- титульный лист;
- наименование и цель работы;
- схемы подключения элементов и устройств к Arduino;
- программы для выполнения соответствующих заданий;
- вывод.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какой датчик используется для определения давления и температуры?
2. Принципиальная схема датчика давления и температуры?
3. Схема подключения датчика давления и температуры к Arduino?
4. Схема подключения ЖК дисплея?
5. Как подключить библиотеку для работы с датчиком давления?
6. Какие команды используются для опроса датчика?

ЛАБОРАТОРНАЯ РАБОТА №6
ИСПОЛЬЗОВАНИЕ СЕРВОПРИВОДОВ В ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫХ СИСТЕМАХ В КАЧЕСТВЕ ИСПОЛНЯЮЩИХ УСТРОЙСТВ

Цель работы: Получить навыки использования сервоприводов в измерительных системах.

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Сервопривод (сервомотор) является важным элементом при конструировании различных исполнительных устройств информационно-измерительных систем. Это точный исполнитель, который имеет обратную связь, позволяющую точно управлять движениями механизмов. Другими словами, получая на входе значение управляющего сигнала, сервомотор стремится поддерживать это значение на выходе своего исполнительного элемента.

Сервомотор имеет встроенный потенциометр, который соединен с выходным валом рисунок 1. Поворотом вала, сервопривод меняет значение напряжения на потенциометре. Плата анализирует напряжение входного сигнала и сравнивает его с напряжением на потенциометре, исходя из полученной разницы, мотор будет вращаться до тех пор, пока не выровняет напряжение на выходе и на потенциометре.

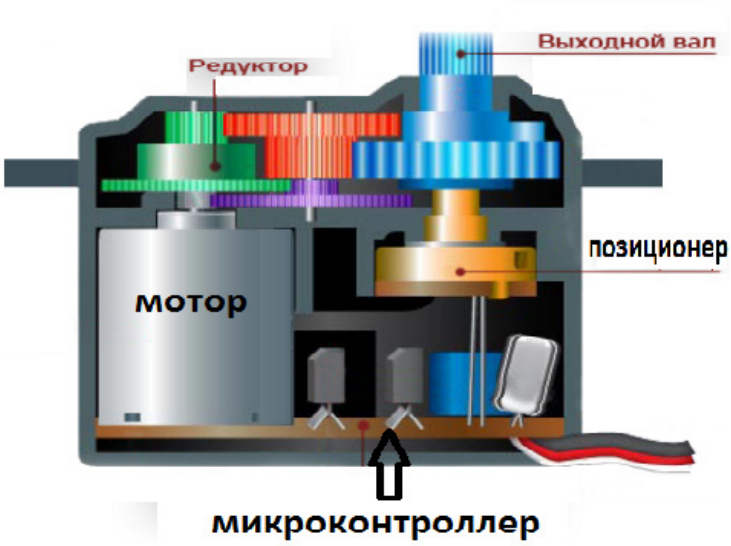


Рисунок 1 – Упрощенный вид сервопривода

Принцип работы сервопривода основан на обратной связи с одним или несколькими системными сигналами. Выходной показатель подается на вход, где сравнивается его значение с задающим действием и выполняются необходимые действия – например, выключается двигатель.

Самым простым вариантов реализации является переменный резистор, который управляется валом – при изменении параметров резистора меняются параметры питающего двигатель тока.

В реальных сервоприводах механизм управления гораздо сложнее и использует встроенные микросхемы-контроллеры. В зависимости от типа используемого механизма обратной связи выделяют **аналоговые** и **цифровые** сервоприводы. Первые используют что-то, похожее на потенциометр, вторые – контроллеры.

Выделяют два основных вида серводвигателей – с непрерывным вращением и с фиксированным углом (чаще всего, 180 или 270 градусов). Отличие серво ограниченного вращения заключается в механических элементах конструкции, которые могут блокировать движение вала вне заданных параметрами углов. Достигнув угла 180, вал окажет воздействие на ограничитель, а тот отдаст команду на выключение мотора. У серводвигателей непрерывного вращения таких ограничителей нет.

Управление сервоприводом

Вся схема управления серво находится внутри корпуса, управляющие сигналы и питание подаются, как правило, идут по трем проводам: земля, напряжение питания и управляющий сигнал.

Решающее значение в управлении сервоприводами выполняет управляющий сигнал, который представляет собой импульсы постоянной частоты и переменной ширины. Длина импульса – это один из важнейших параметров, который определяет положение сервопривода. Эту длину можно задать в программе вручную методом подбора через угол или использовать команды библиотеки.

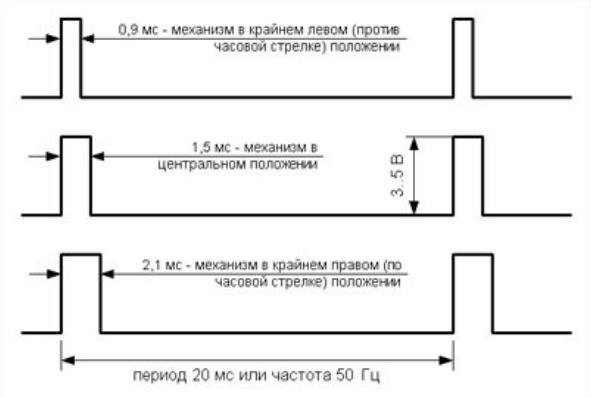


Рисунок 2 – Управляющие сигнал сервопривода

Когда сигнал попадает в управляющую схему, генератор подает свой импульс, длительность которого определяется с помощью потенциометра. В другой части схемы происходит сравнение длительности поданного сигнала и сигнала с генератора. Если эти сигналы разные по длительности, включается электромотор, направление вращения которого определяется

тем, какой из импульсов короче. При равенстве длины импульсов мотор останавливается.

Стандартная частота, с которой подаются импульсы, равна 50 Гц, то есть 1 импульс в 20 миллисекунд.

При таких значениях длительность составляет 1520 микросекунд, и сервопривод занимает среднее положение. Изменение длины импульса приводит к повороту сервопривода – при увеличении длительности поворот осуществляется по часовой стрелке, при уменьшении – против часовой стрелки. Имеются границы длительности – в Arduino в библиотеке Servo для 0° установлено значение импульса в 544 мкс (нижняя граница), для 180° – 2400 мкс (верхняя граница).

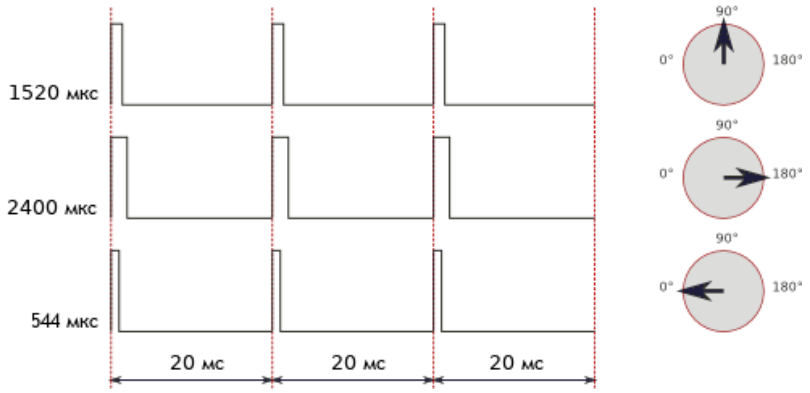


Рисунок 3 – Зависимость угла поворота от длительности импульсов

Подключение серводвигателя к Arduino

Сервопривод обладает тремя контактами, которые окрашены в разные цвета. Коричневый или черный провод ведет к земле, красный – к питанию +5В, провод оранжевого или желтого цвета – сигнальный.

К Arduino устройство подключается через макетную плату рисунок 4.

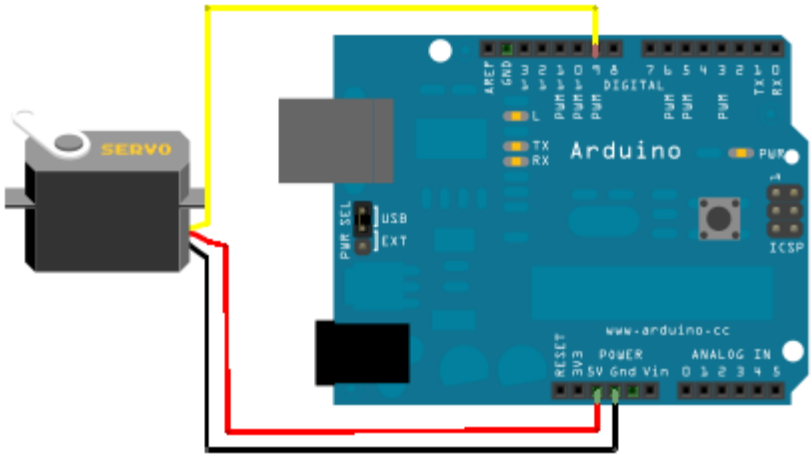


Рисунок 4 – Подключение сервопривода к Arduino

Оранжевый провод (сигнальный) подключается к цифровому пину, черный и красный – к земле и питанию соответственно. Для управления серводвигателем не требуется подключение именно к шим-пинам.

Управление сервоприводом напрямую через изменение в скетче длительности импульсов – достаточно сложная задача. Обычно для этих целей используют библиотеку Servo, встроенную в среду разработки Arduino.

Рассмотрим порядок использования библиотеки Servo.

Алгоритм работы следующий:

- Подключаем библиотеку Servo.h
- Создаем объект класса Servo
- В блоке setup указываем, к какому пину подключен сервопривод
- Используем методы объекта обычным для C++ способом. Например, метод write, которому мы подаем целочисленное значение в градусах.

2. ЗАДАНИЯ НА ЛАБОРАТОРНУЮ РАБОТУ

Задание 1. Установить серводвигатель на нулевой угол, а затем осуществить поворот на 90 градусов.

```
#include <Servo.h>
Servo servo;  // Создаем объект
void setup() {
  servo.attach(9);      // Указываем объекту класса Servo, что серво присоединен к пину 9
  servo1.write(0);   // Выставляем начальное положение
  }

void loop() {
  servo.write(90); // Поворачиваем серво на 90 градусов
  delay(1000);
  servo.write(1800);
  delay(100);
  servo.write(90);
```

3. СОДЕРЖАНИЕ ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

Отчет по лабораторной работе должен содержать:

- титульный лист;
- наименование и цель работы;
- схемы подключения элементов и устройств к Arduino;

- программы для выполнения соответствующих заданий;
- вывод.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение сервопривода?
2. Принцип работы сервопривода?
3. Способы управления сервоприводом?
4. Схема подключения сервопривода к Arduino?
5. Как подключить библиотеку для работы с сервоприводом?
6. Какие команды используются для управления сервоприводом?

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аппаратная платформа Arduino : официальный сайт. – Url: https://arduino.ru/ (дата обращения: 13.12.2023). – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

2. Боровский, А. С. Программирование микроконтроллера Arduino в информационно-управляющих системах : учебное пособие / А. С. Боровский, М. Ю. Шрейдер. – Оренбург : ОГУ, 2017. – 113 с. – Текст: непосредственный.

3. Глибин, Е. С. Разработка измерительных систем с применением контроллеров Arduino : учебно-методическое пособие / Е. С. Глибин, В. И. Чепелев. – Тольятти : ТГУ, 2016. – 48 с.

4. Григорьев, Е. К. Разработка систем анализа и обработки информации на базе Arduino : учебно-методическое пособие / Е. К. Григорьев, В. А. Ненашев, А. М. Сергеев. – Санкт-Петербург : ГУАП, 2022. – 63 с. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/263945 (дата обращения: 13.12.2023). – Режим доступа: для авториз. пользователей.

5. Кульченко, А. Е. Using the Arduino platform in robotic development : учебное пособие / А. Е. Кульченко, М. Ю. Медведев. – Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2022. – 134 с. – ISBN 978-5-9275-4254-3. – Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/129088.html (дата обращения: 22.03.2023). – Режим доступа: для авторизир. пользователей.

6. Петин, В. А. Практическая энциклопедия Arduino : практическое руководство / В. А. Петин, А. А. Биняковский. - 2-е изд., доп. - Москва : ДМК Пресс, 2020. - 166 с. – Текст : непосредственный.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫХ СИСТЕМ

Методические рекомендации
к выполнению лабораторных работ

Составители: **Балакин** Алексей Игоревич, **Балакина** Наталья Анатольевна

В авторской редакции

Изд. № 226/2023. Объем 2 п.л. Усл. печ. л. 1,86. Уч.-изд. л. 1,96.
РИИЦМ ФГАОУ ВО «Севастопольский государственный университет»