Лабораторная работа № 2.
“ Аналоговые порты. ШИМ ”

Цель работы

Изучение аналоговых портов. Знакомство с широтно-импульсной модуляцией.

Задача

Познакомиться с теоретическим материалом и выполнить задание по вариантам.

Необходимый материал для выполнения

1. Arduino UNO.
2. Breadboard.
3. Светодиод.
4. Резистор.
5. Монтажные провода.
6. Потенциометр.

Содержание отчёта

1. Цель работы.
2. Задание на работу по варианту.
3. Собранная схема подключения.
4. Блок-схема алгоритма.
5. Код, написанный на языке программирования С.
6. Выводы по работе.

Содержание

Теоретическая часть

Аналоговый сигнал - сигнал, который может быть представлен непрерывной линией из множества значений, определенных в каждый момент времени относительно временной оси. Иллюстрация, отражающая отличие аналогового сигнала от цифрового была приведена в прошлой лабораторной, ниже она продублирована (рисунок 1). Аналоговым сигналом можно назвать, например, звуковой сигнал, генерируемый обмоткой электромагнитного микрофона, поскольку такой сигнал непрерывен и его значения (напряжение или ток) сильно отличаются друг от друга в каждый момент времени, также примером может послужить значение температуры, которое передает термопара на контроллер.

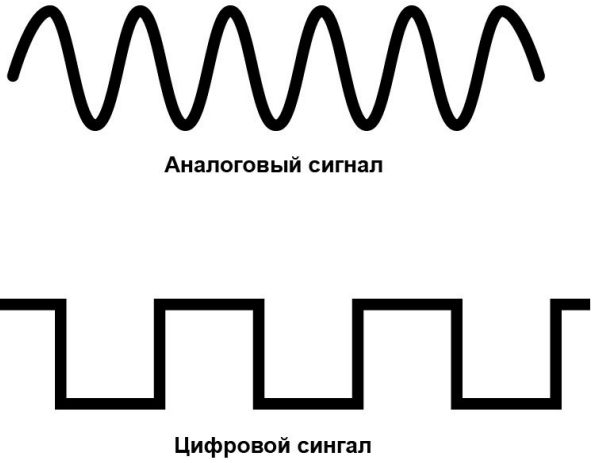

Рисунок 1 - Изображение аналогового и цифрового сигналов

Аналоговые порты на Arduino Uno

Микроконтроллер Arduino Uno содержит 6 аналоговых ***входов***, т.е. входов, подключенных к АЦП – аналогово-цифровому преобразователю (ADC), которые маркированы буквой A (A0 - A5) (рисунок 2) . Правильнее сказать, что эти 6 выводов платы могут работать в режиме, как *дискретных выводов*, так и *аналоговых входов*.

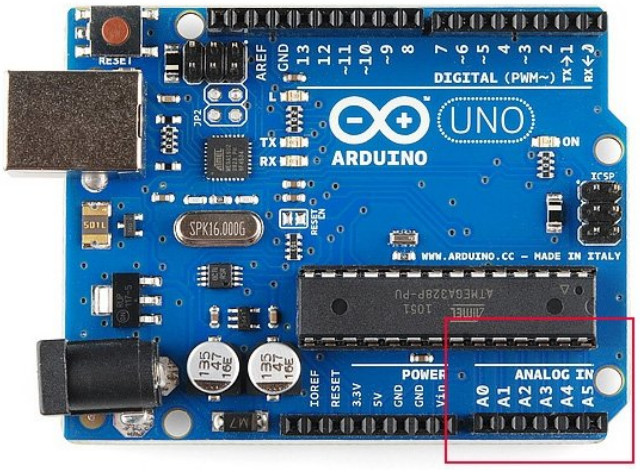

Рисунок 2 - Обозначение аналоговых портов на плате Arduino Uno.

Зачем нужно читать аналоговый сигнал? Микроконтроллер может выступать в роли вольтметра, измерять собственное напряжение питания, например от аккумулятора, может измерять ток через шунт, можно измерять сопротивление, а также работать с потенциометрами (крутильными, линейными, джойстиками), которые являются очень удобными органами управления.

Чтение аналогового сигнала на Arduino Uno

Аналоговые пины могут принимать напряжение от 0 (GND) до опорного* напряжения и преобразовывать его в цифровое значение, просто в какие-то условные единицы. АЦП имеет разрядность в 10 бит, т.е. измеренное напряжение будет представлено в виде числа от 0 до 1023. Функция, которая оцифровывает напряжение, называется analogRead(pin), данная функция принимает в качестве аргумента номер аналогового пина и возвращает полученное значение. Сам пин должен быть сконфигурирован как INPUT (вход).

Листинг 1 - Примеры чтения аналогового сигнала

```
1.    int value1 = analogRead(0); //считать напряжение с пина A0
2.    int value2 = analogRead(A0); //считать напряжение с пина A0
3.    int value3 = analogRead(14); //считать напряжение с пина A0
```

Хранить полученное значение разумно в переменной типа int, потому что значение варьируется от 0 до 1023.

Опорное напряжение*

*От опорного напряжения зависит максимальное измеряемое напряжение, возможность и точность перевода полученного значения 0-1023 в Вольты. По умолчанию опорное напряжение равно напряжению питания микроконтроллера. Также, в качестве опрного напряжения может выступать напряжение встроенного источника питания или напряжение поданное на пин AREF. В рамках данной лабораторной работы в качестве опрного напряжения будет использоваться напряжение питания микроконтроллера 5 В.

Потенциометр

Потенциометр - переменный резистор с регулируемым уровнем сопротивления, представленный на рисунке 3. При использовании потенциометра появляется возможность менять параметры электрических схем, гибко подстраивая их под определенные условия: например, регулировать чувствительность датчика или громкость звука в динамике. Потенциометры получили широкое распространение в схемах регулировки громкости, напряжения, контрастности и т.д., за свою простоту и практичность.

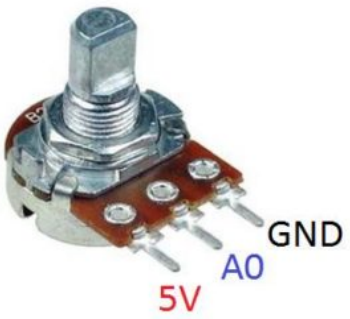

Рисунок 3 - Потенциометр

К Arduino потенциометр подключается следующим образом: средний вывод на любые А-пины, крайние – на GND и питание. От порядка подключения GND и питания зависит направление изменения значения.

Измерение напряжения

Пример измерения напряжения:

```
float voltage = (float)(analogRead(0) * 5.0) / 1024;
```

Переменная voltage получает значение в Вольтах, от 0 до 5.

Широтно-импульсная модуляция (ШИМ)

Часто в робототехнике возникает необходимость плавно управлять каким-то процессом, будь то яркость светодиода, мощность обогревателя или скорость вращения двигателя. Управление напрямую связано с изменением значения напряжения на потребителе: светодиод будет по-другому светить, двигатель крутиться с другой скоростью. Управлять напряжением может только ЦАП – цифро-аналоговый преобразователь, в Arduino UNO встроенного ЦАПа нет, есть только цифровой сигнал, т.е. либо подано напряжение (1), либо нет (0).

Однако добиться плавного управления цифровым сигналом можно. Представьте себе вентилятор, вращающийся на полной мощности, напряжение постоянно. Представим теперь, что секунду напряжение подаётся, и секунду – нет, и так продолжается “по кругу”. Вентилятор начнёт крутиться в два раза медленнее, но мы скорее всего будем замечать моменты включения и выключения, особенно если вентилятор маленький. Большой вентилятор более инертен и там можно даже не заметить изменений скорости в пределах двух секунд. Можно теперь включать напряжение на 0.5 секунды, а на остальные 1.5 секунды – выключать. Вентилятор будет крутиться со скоростью 25% от максимальной. Так

получается ШИМ сигнал, широтно-импульсную модуляция. На рисунке 4 приведен пример с лампой накаливания.

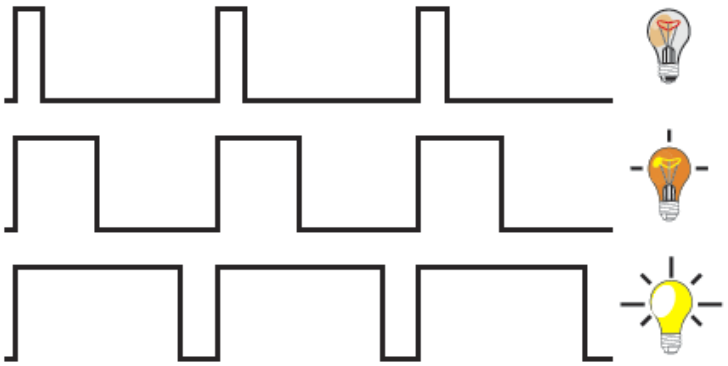

Рисунок 4 - Зависимость яркости лампочки от времени подачи напряжения

Что касается работы со светодиодом, то у него практически нет задержки включения/выключения, поэтому при периоде в 2 секунды (частоте 0,5 Гц), как представлено в эксперименте выше, визуально будет заметно как светодиод загорается и гаснет. Чтобы исправить это, необходимо увеличить частоту. При высокой частоте человеческий глаз не будет улавливать вспышек, человеком будет восприниматься это как увеличение или уменьшение яркости.

Важно отметить, что в теории выделяются такие коэффициенты ШИМ, как скважность - $S = T/t_и$ ($t_и$ - время длительности импульса; T - период одного импульсного колебания) и обратный ему коэффициент заполнения - $D = t_и /T$. Он, как правило, выражается в процентах.

Фактически коэффициент заполнения показывает, на сколько процентов импульс заполняет весь период колебания T. Если коэффициент заполнения D равен 1 (100%), то время импульса целиком заполняет период и фактически это постоянное напряжение. Если уменьшить коэффициент заполнения D, например, до 0,25 (25%), то длительность

импульса будет всего 25% от всего периода, а результирующее напряжение будет уже в 4 раза меньше, как показано на рисунке 5.

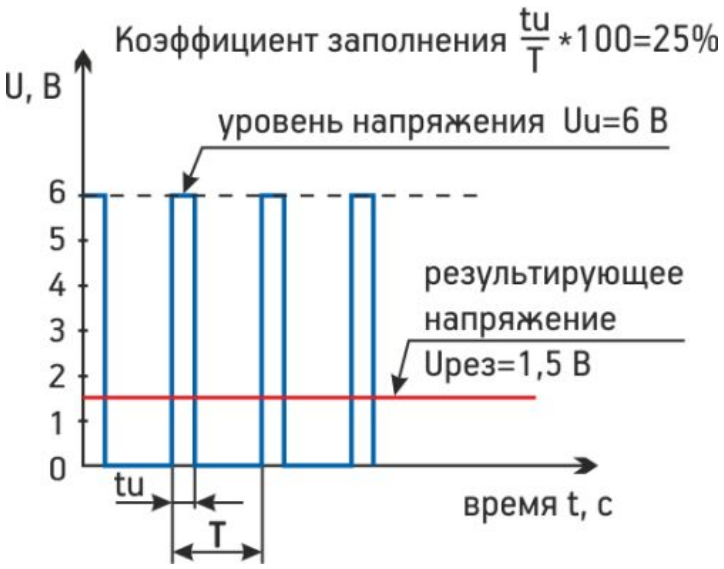

Рисунок 5 - Принцип работы ШИМ

Таким образом, при помощи ШИМ сигнала можно даже модулировать сложные аналоговые сигналы, например – синусоиду (Рисунок 6).

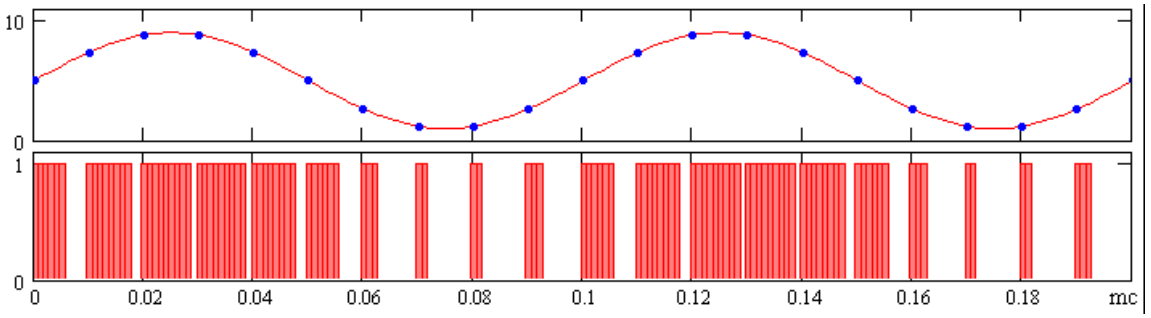

Рисунок 6 - Модуляция синусоиды с помощью ШИМ

ШИМ и Arduino Uno

Для генерации ШИМ-сигнала на Arduino Uno предусмотрены 6 цифровых портов. В английском варианте ШИМ имеет название pulse-width modulation (PWM), поэтому пины, генерирующие ШИМ-сигнал обозначены как Digital PWM (~) (Рисунок 7).

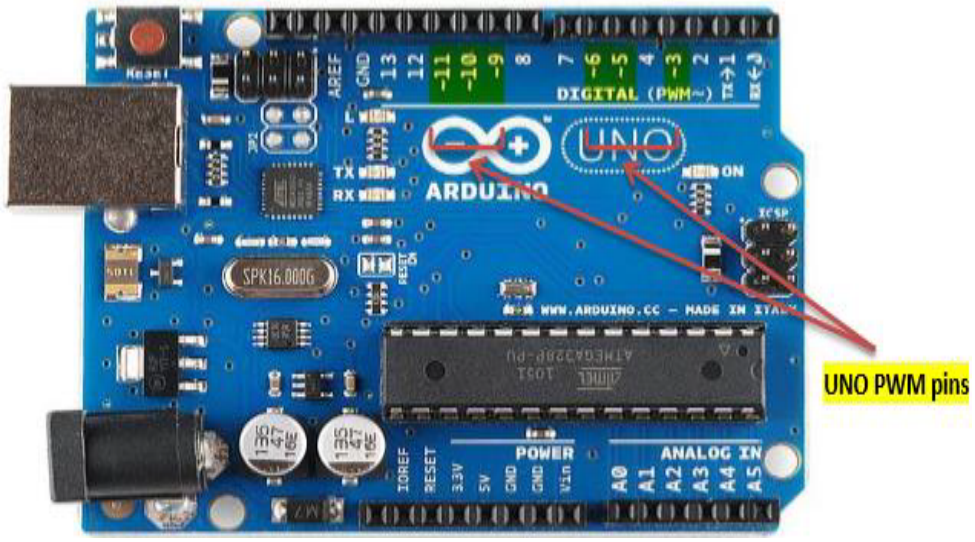

Рисунок 7 - Обозначение пинов, генерирующих ШИМ, на Arduino Uno

Для генерации ШИМ предусмотрена готовая функция analogWrite(pin, duty). В данном случае, pin - номер пина, с которого генерируется ШИМ-сигнал, а duty - заполнение ШИМ сигнала. По умолчанию все “выходы” ШИМ 8-битные, то есть duty может принимать значение с “разрешением” 8 бит, а это 0-255.

Листинг 2 - Пример записи ШИМ-сигнала на вывод светодиода

```
1.    #define pwmPin 3   //Обозначение пина, на который будет
подаваться значение заполнения ШИМ сигнала
2.
3.    void setup() {
4.      pinMode(pwmPin, OUTPUT); //Настраиваем 3 пин как выход
5.    }
6.
7.    void loop() {
8.      analogWrite(pwmPin, 102); //Записываем в 3 порт значение
заполнения ШИМ
9.    }
```

ВАЖНО!!! При выполнении заданий не забудьте, что на выходе АЦП значения изменяются от 0 до 1023 бит, а ШИМ выходы могут выдавать от 0 до 255 бит. Для удобства преобразования чисел можно

использовать функцию под названием “map”. Синтаксис этой функции следующий:

map(value, fromLow, fromHigh, toLow, toHigh)

Функция пропорционально переносит значение (value) из текущего диапазона значений (fromLow .. fromHigh) в новый диапазон (toLow .. toHigh), заданный параметрами.

Листинг 3 - Пример использования функции “map”

```
1.    /*Переносим значение с аналогового входа (возможные
значения от 0 до 1023) в 8 бит (0..255)*/
2.    #define adcPin 0 //Обозначение пина, с которого будем
считывать аналоговые значения
3.    #define pwmPin 9
4.
5.    void setup() {
6.      pinMode(pwmPin, OUTPUT); //Настраиваем пин на выход
7.      pinMode(adcPin, INPUT); //Настраиваем пин на вход
8.    }
9.    void loop() {
10.     int val = analogRead(adcPin);
11.     val = map(val, 0, 1023, 0, 255);
12.     analogWrite(pwmPin, val); //Записываем в 9 порт значение
заполнения ШИМ, которое получаем из val
13.   }
```

Практическая часть

Первый вариант

Реализовать управление яркостью светодиода с помощью потенциометра: при вращении ручки потенциометра по часовой стрелке яркость светодиода должна увеличиваться, при вращении против часовой стрелки яркость светодиода должна уменьшаться.

Второй вариант

Реализовать управление яркостью светодиода с помощью потенциометра: при вращении ручки потенциометра против часовой стрелки яркость светодиода должна увеличиваться, при вращении по часовой стрелке яркость светодиода должна уменьшаться.

Третий вариант

Реализовать управление рядом светодиодов, состоящим из 4-х светодиодов, с помощью потенциометра. При повороте ручки потенциометра вправо светодиоды должны последовательно загораться, пока не будут включены все. При обратном вращении светодиоды должны последовательно выключаться.

Четвертый вариант

Реализовать управление яркостью светодиода с помощью 2-х кнопок. При удержании первой кнопки изменяется яркость светодиода. При однократном нажатии на вторую кнопку изменяется направление изменения яркости.

Пятый вариант

Реализовать управление яркостью светодиода с помощью потенциометра и кнопки. При прокручивании ручки потенциометра яркость светодиода должна изменяться. Направление изменения яркости должна задавать кнопка.

Шестой вариант

Реализовать управление 2-мя светодиодами, с помощью потенциометра. При повороте ручки потенциометра вправо с крайнего левого положения до середины первый светодиод должен увеличивать свою яркость до максимальной, при вращении ручки потенциометра вправо с середины до крайнего правого положения второй светодиод должен увеличивать свою яркость до максимальной.

Седьмой вариант

Реализовать управление яркостью светодиода с помощью 2-х кнопок. При удержании первой кнопки яркость светодиода изменяется в большую сторону, при удержании второй кнопки яркость светодиода изменяется в меньшую сторону.

Восьмой вариант

Реализовать управление рядом светодиодов, состоящим из 6-ти светодиодов, с помощью потенциометра. При повороте ручки потенциометра вправо должны одновременно увеличивать свою яркость светодиоды №1, №3, №5, а при повороте влево постепенно гаснуть. При повороте ручки потенциометра в левую сторону диоды №2, №4 и № 6

должны одновременно увеличивать свою яркость, а при повороте вправо постепенно гаснуть.

Девятый вариант

Реализовать управление рядом светодиодов, состоящим из 5-ти светодиодов, с помощью потенциометра. При повороте ручки потенциометра вправо светодиоды должны загораться по очереди, изменяя свою яркость с минимальной до максимальной, при повороте ручки потенциометра влево светодиоды должны гаснуть по очереди, изменяя свою яркость с максимальной до минимальной.