Лабораторная работа № 1

Аналого-цифровое преобразование и широтно-импульсная модуляция в Arduino

Цель лабораторной работы: познакомиться с принципами аналого-цифрового преобразования и технологией широтно-импульсной модуляции в Arduino. Научиться настраивать выводы микроконтроллера на соответствующий задаче режим работы.

Теоретическая часть

Аналого-цифровой преобразователь (АЦП, англ. Analog-to-digital converter, ADC) — устройство, преобразующее входной аналоговый сигнал в дискретный код (цифровой сигнал).

Плата **Arduino UNO** содержит 6 аналоговых входов предназначенных для измерения напряжения сигналов. Правильнее сказать, что шесть **GPIO** платы могут работать в режиме, как дискретных выводов, так и аналоговых входов. Эти выводы имеют номера от 14 до 19. Изначально они настроены как аналоговые входы, и обращение к ним можно производить через имена **A0-A5** (Рисунок 1). В любой момент их можно настроить на режим дискретных выходов.

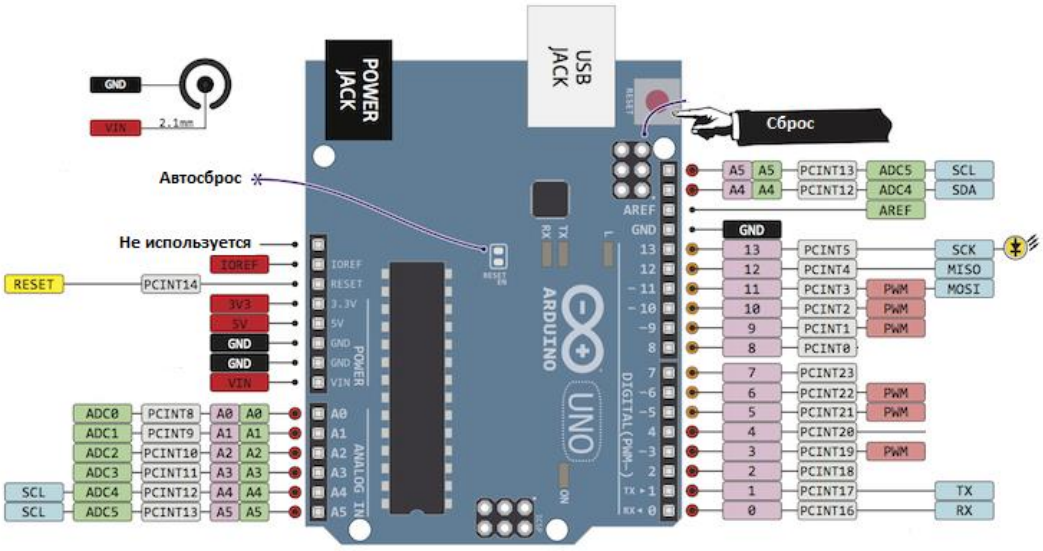

Рисунок 1 – Нумерация GPIO платы Arduino UNO

Данные GPIO могут принимать напряжение от 0 (**GND**) до **опорного** напряжения и преобразовывать его в цифровое значение. **АЦП** у **Arduino UNO** имеет разрядность в **10 бит**, т.е. мы получаем измеренное напряжение в виде числа от 0 до 1023. В относительных единицах одно деление шкалы будет соответствовать 5В/1024 = 4.9 мВ.

Для оцифровки напряжения на аналоговых GPIO существует функция:

```
analogRead(pin)
```

Данная функция принимает в качестве аргумента номер аналогового GPIO и возвращает полученное значение. Сам пин (GPIO) должен быть сконфигурирован как INPUT (вход).

Аргумент функции можно определить несколькими способами:

- Номером A-пина (например, 0)
- Номером с буквой A (например, A0)
- Порядковым номером GPIO: A0 – 14 пин, A1 – 15 пин

Пример:

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

Опорное напряжение

В общем случае диапазон измерений АЦП имеет ширину от 0 В до опорного напряжения. Это изменение повлечет за собой изменение формулы расчет точности (E) АЦП:

$$E = U_{оп}/1024$$

Опорное напряжение определяет границу диапазона, с которым будет работать АЦП.

Обычно, опорное напряжение будет равно напряжению питания Arduino Uno, которое получили от USB порта.

Допустим, что есть четыре NiMh аккумулятора на 1.2 Вольта от которых запитана плата Arduino. В сумме они дадут 4.8 Вольта (пусть они немного разряжены, ведь в действительности их заряжают до 1.4 Вольта). Точность измерения будет равна 4.8/1024. Это следует учесть в нашей программе.

Рассмотрим случай, когда мы питаем Arduino Uno одним напряжением, а в качестве опорного хотим установить другое, например, 3.3 В. Для такого варианта на Arduino Uno есть специальный вывод Vref. Чтобы сформировать опорное напряжение для АЦП необходимо подать на этот контакт напряжение 3.3 В, и разрешить использование внешнего источника опорного напряжения функцией:

```
analogReference(EXTERNAL);
```

Также следует учитывать, что результат измерения значения напряжения не может превышать границы диапазона. Если выбрать в качестве опорного напряжения 3.3 В, а поступающий сигнал будет 4В, то мы получим неправильное значение напряжения, поскольку АЦП уже от 3.3 В будет выдавать значение 1023.

Рассмотрим пример работы АЦП. Для этого необходимо собрать схему (Рисунок 2).

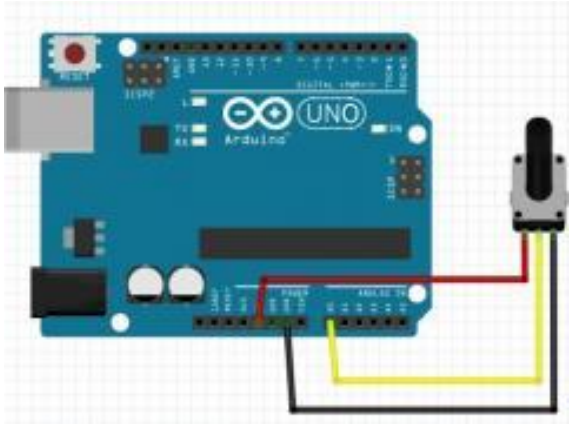

Рисунок 2 – Пример подключения потенциометра к Arduino

На среднем контакте переменного резистора можно выставить любое напряжение в диапазоне от 0 до 5В. Это напряжение преобразуется в числовое значение встроенным в микроконтроллер АЦП.

```
const int analogInPin = A0; //включаем аналоговый порт

void setup()
{
      Serial.begin(9600); //открываем Serial порт со
      скоростью 9600 бит/с
}

void loop()
{
      Serial.println(analogRead(A0)); //записываем в Serial
      порт значения считанные из АЦП
      delay(100);
}
```

Рассмотрим следующий пример: измерение напряжения элемента питания. Чтобы измерить напряжение на элементе питания, необходимо подключить его к Arduino двумя контактами (Рисунок 3).

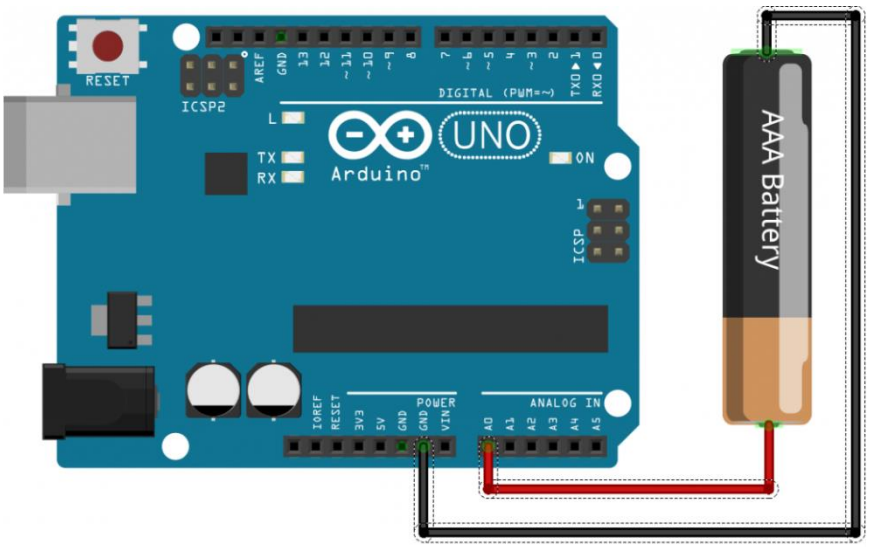

Рисунок 3 – Пример подключения внешнего элемента питания к АЦП

Теперь необходимо открыть окно COM-монитора в Arduino IDE, и посмотреть какие значение выдает нам АЦП (Рисунок 4):

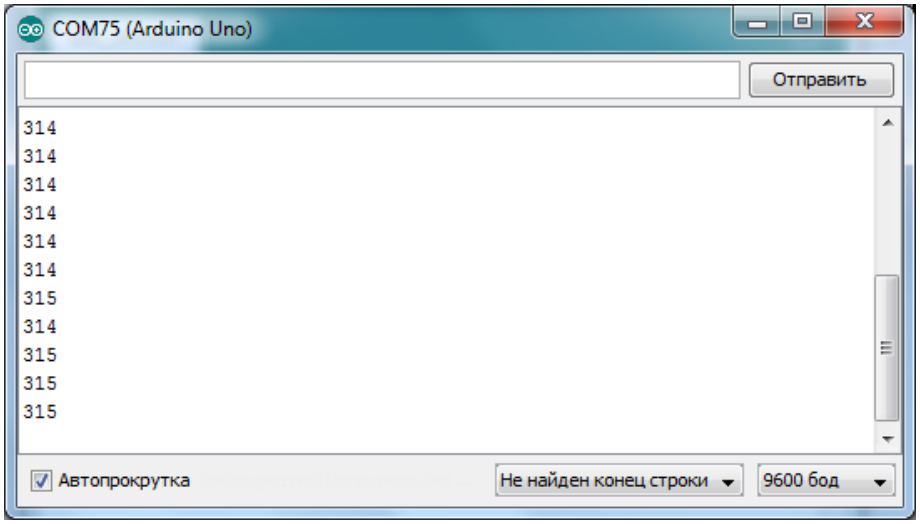

Рисунок 4 – Окно монитора порта Arduino IDE

В итоге получаем некоторое число (в примере 314). Вспомним, что 10-битный АЦП разбивает диапазон от 0 до 5 вольт на 1024 части.

Зная точность, мы можем записать формулу для преобразования показаний АЦП к напряжению:

V = (5/1024)*ADC,

где **V** — измеренное напряжение на батарейке;
ADC — результат работы функции analogRead.

Напишем эту формулу в программе и снова попробуем измерить напряжение на элементе питания:

```
const int analogInPin = A0; //включаем аналоговый порт

void setup()
{
      Serial.begin(9600); //открываем Serial порт со
      скоростью 9600 бит/с
}

void loop()
{
      Serial.println((5/1024.0) * analogRead(A0));
      delay(100);
}
```

В результате получим следующие значения:

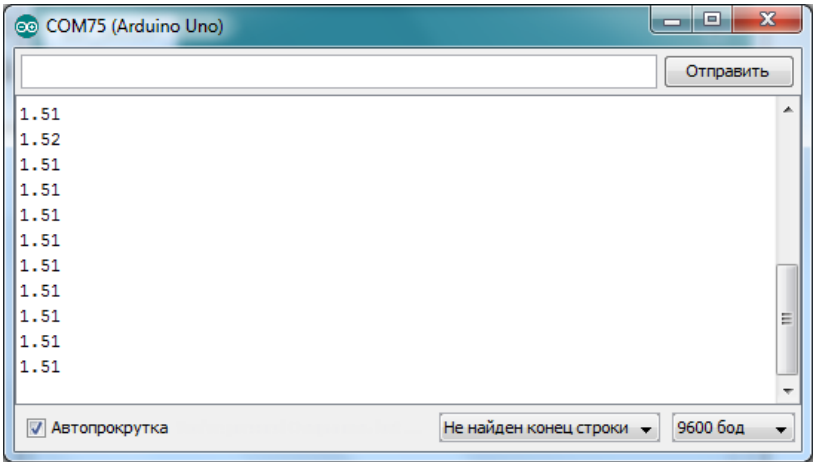

Рисунок 5 – Результат математического преобразования значений АЦП в напряжение

Широтно-импульсная модуляция

Широтно-импульсная модуляция (ШИМ) это способ управления мощностью на нагрузке с помощью изменения скважности импульсов при постоянной амплитуде и частоте импульсов.

Можно выделить две основные области применения широтно-импульсной модуляции:

- Во вторичных источниках питания, различных регуляторах мощности, регуляторах яркости источников света, скорости вращения коллекторных двигателей и т.п. В этих случаях применение ШИМ позволяет значительно увеличить КПД системы и упростить ее реализацию.

- Для получения аналогового сигнала с помощью цифрового выхода микроконтроллера. Своеобразный цифро-аналоговый преобразователь (ЦАП). Очень простой в реализации, требует минимума внешних компонентов. Часто достаточно одной RC цепочки.

Принцип регулирования с помощью ШИМ – изменение ширины импульсов при постоянной амплитуде и частоте сигнала.

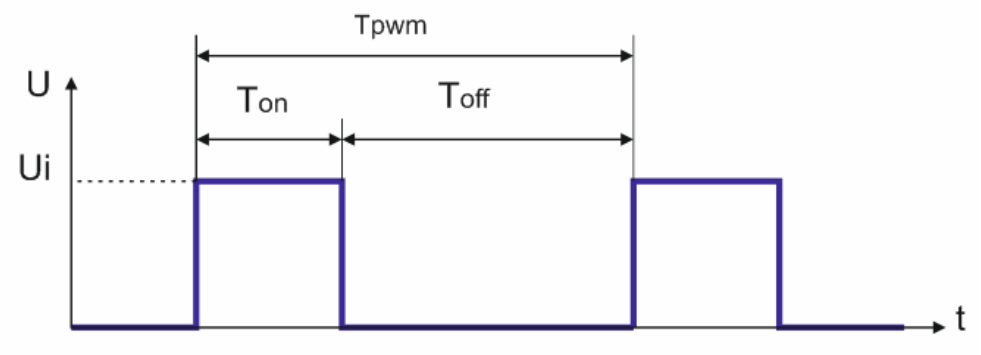

Рисунок 6 – Параметры ШИМ сигнала

На диаграмме (Рисунок 6) можно увидеть основные параметры ШИМ сигнала:

- U_i - амплитуда импульсов ;
- T_{on} – время активного (включенного) состояния сигнала;
- T_{off} – время отключенного состояния сигнала;
- T_{pwm} – время периода ШИМ.

Даже интуитивно понятно, что мощность на нагрузке пропорциональна соотношению времени включенного и отключенного состояния сигнала.

Это соотношение определяет коэффициент заполнения ШИМ:

$$\mathbf{K_w = T_{on} / T_{pwm}.}$$

Он показывает, какую часть периода сигнал находится во включенном состоянии. Может меняться:

- от 0 – сигнал всегда выключен;
- до 1 - сигнал все время находится во включенном состоянии.

Чаще используют процентный коэффициент заполнения. В этом случае он находится в пределах от 0 до 100%.

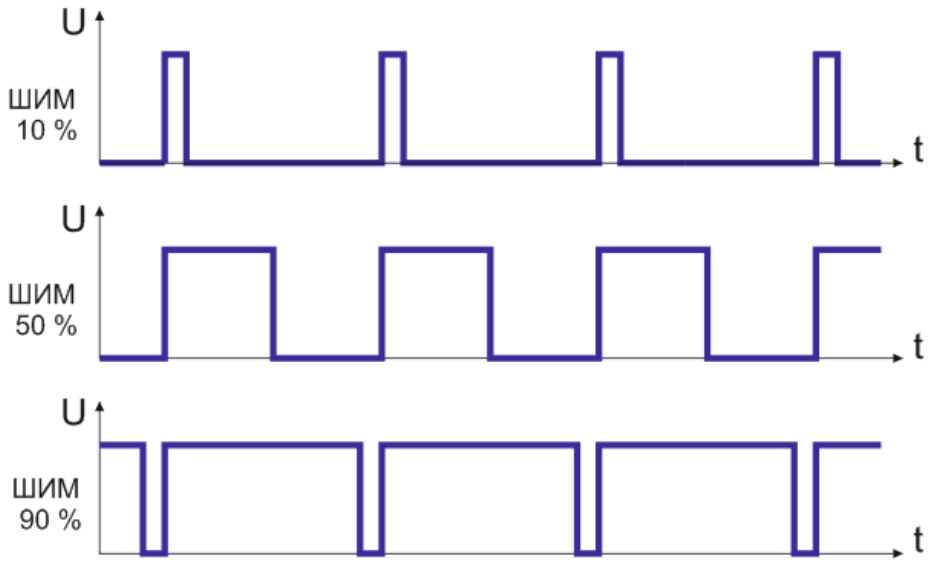

Рисунок 7 – ШИМ с разным коэффициентом заполнения.

Изменяя коэффициент заполнения ШИМ сигнала можно менять напряжение (интегрированное) за некоторый период. Чем больше заполнение ШИМ, тем выше напряжение (Рисунок 8).

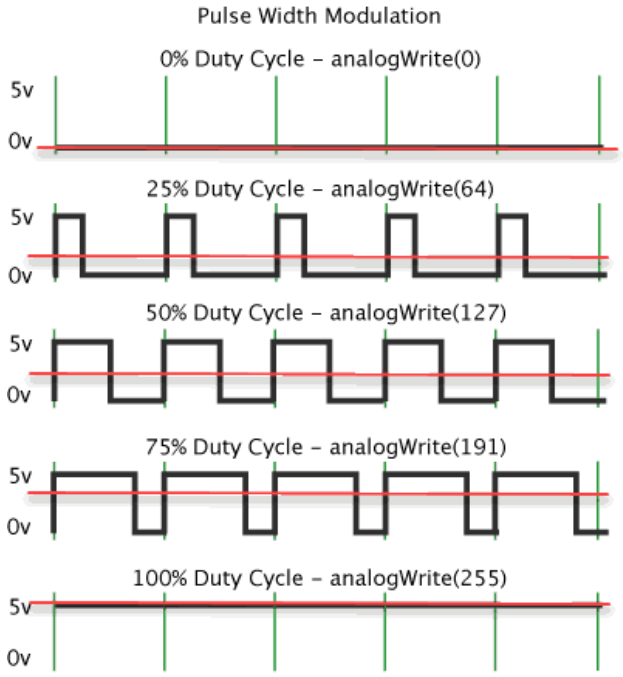

Рисунок 8 – Уровни напряжения при разных коэффициентах заполнения

При помощи ШИМ сигнала можно формировать сложные аналоговые сигналы, например – синусоиду. На рисунке 9 показан ШИМ (снизу) и полученное фильтрацией напряжение сверху:

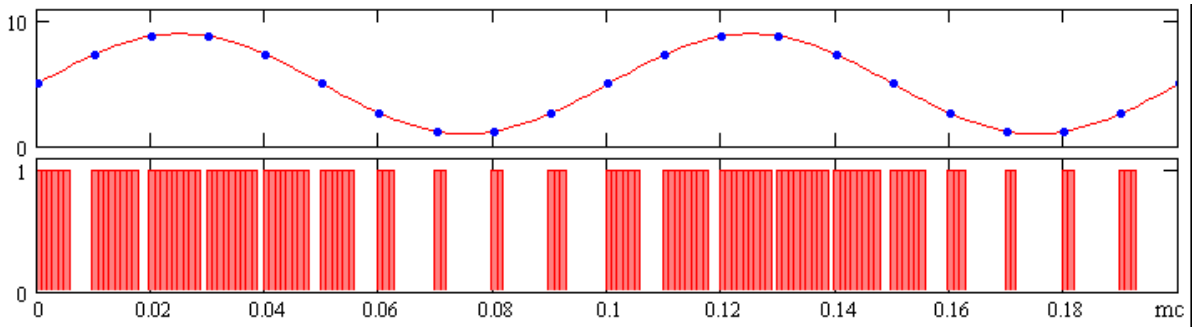

Рисунок 9 – Формирование синусоиды с использованием ШИМ

Если сигнал ШИМ пропустить через фильтр низких частот (ФНЧ) (Рисунок 10), то на выходе фильтра мы получим аналоговый сигнал, напряжение которого пропорционально коэффициенту заполнения ШИМ.

U = Kw * Ui

В качестве ФНЧ можно использовать простейшую RC цепочку.

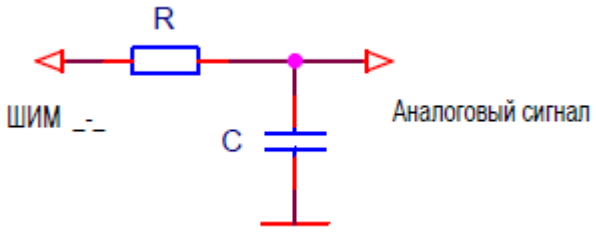

Рисунок 10 – Схема фильтра низких частот

ШИМ в Arduino

Платы Arduino на базе микроконтроллеров **ATmega168/328** имеют **6** аппаратных широтно-импульсных модуляторов. Сигналы ШИМ могут быть сгенерированы на выводах 3, 5, 6, 9, 10, 11. Данные выводы на плате Arduino UNO отмечены волной (Рисунок 11).

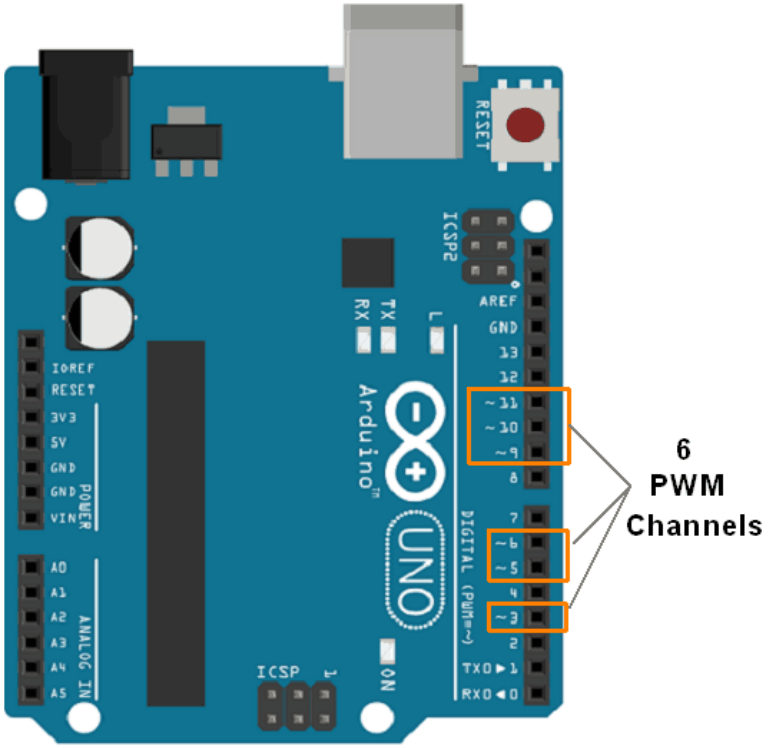

Рисунок 11 – Выводы Arduino UNO для работы с ШИМ

Управление аппаратными ШИМ осуществляется с помощью системной функции **analogWrite().**

```
void analogWrite(pin, val)
```

Функция переводит вывод в режим ШИМ и задает для него коэффициент заполнения. Перед использованием **analogWrite()** функцию **pinMode()** для установки вывода в режим **OUTPUT** вызывать необязательно.

Аргументы:

- **pin** – номер вывода для генерации ШИМ сигнала.
- **val** – коэффициент заполнения ШИМ.

Без дополнительных установок диапазон val от 0 до 255 и соответствует коэффициенту заполнения от 0 до 100 %. Т.е. разрядность системных ШИМ в Arduino **8 разрядов**.

```
analogWrite(9, 25); // на выводе 9 ШИМ = 10%
```

Система Ардуино устанавливает на всех выводах ШИМ параметры:

- частота 488,28 Гц;
- разрешение 8 разрядов (0…255).

Практическая часть

Упражнение 1

1. Собрать схему и написать программу для организации автоматического фонаря на датчике освещенности, включающего или выключающего светодиод в зависимости от освещенности в комнате.

2. Собрать схему и написать программу индикатора напряжения на светодиодах. Если потенциометр находится в крайнем левом положении, то светодиоды не горят. При вращении ручки потенциометра вправо, количество светящихся светодиодов постепенно увеличивается. В крайнем правом положении должны светиться все 5 светодиодов.

3. Собрать схему (см. рисунок 12) и написать программу сканера клавиатуры на аналоговом пине. По нажатию одной из кнопок должен зажечься соответствующий светодиод.

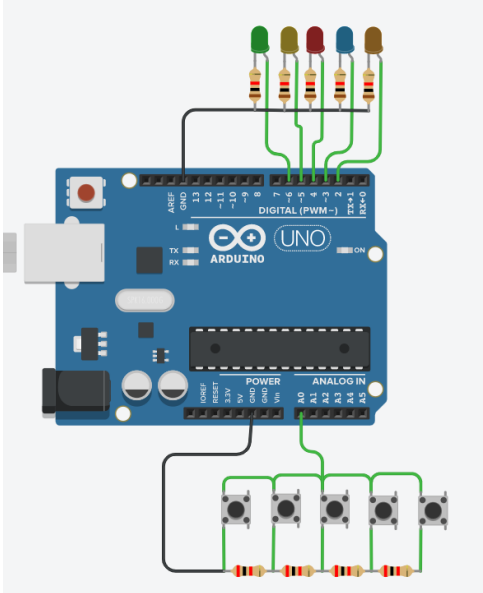

Рисунок 12 – Схема к заданию 3

Упражнение 2

1. Собрать схему и написать программу для управления RGB-светодиодом с плавным перебором цветов от красного до фиолетового.
2. Собрать схему и написать программу гирлянды с использованием ШИМ для управления 6 светодиодами в разных режимах (бегущая единица, чередующийся огонь, бегущий ноль, расходящийся от центра огонь и т.п.). Эффекты должны переключаться кнопкой. Дополнить гирлянду демо-режимом (последовательный перебор всех эффектов).