Лабораторная работа 2

Изучение принципов формирования управляющих воздействий на исполнительные устройства через дискретный и аналоговый интерфейсы

Цель и задачи лабораторной работы

Цель: Получить практические навыки формирования управляющих воздействий на исполнительные устройства через дискретный и аналоговый интерфейсы Arduino (микроконтроллера).

Задачи:

1) Ознакомиться с теорией.
2) Изучить способы подключения исполнительных устройств к дискретному и аналоговому интерфейсам контроллера.
3) Освоить методику написания программ для управления исполнительными устройствами через дискретный и аналоговый интерфейсы.
4) Выполнить задание своего варианта.
5) Ответить на контрольные вопросы.
6) Подготовить и защитить отчет по лабораторной работе.

Теория

Плата Arduino Uno

Контроллер Arduino Uno построен на микроконтроллере ATmega328. Платформа имеет 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, разъем питания, разъем внутрисхемного программирования ICSP и кнопку перезагрузки. Для работы необходимо подключить платформу к компьютеру посредством кабеля USB, либо подать питание при помощи адаптера AC/DC или батареи [1].

Основные технические характеристики Arduino Uno:

- Микроконтроллер ATmega328
- Рабочее напряжение..... 5 В
- Внешнее напряжение (рекомендуемое)..... 7-12 В
- Цифровые входы/выходы..... 14 (6 из них выходы ШИМ)
- Аналоговые входы 6
- Постоянный ток через вход/выход..... 40 мА
- Постоянный ток для вывода 3.3 В..... 50 мА
- Флеш-память ..... 32 Кб из них 0.5 Кб для загрузчика
- ОЗУ ..... 2 Кб (ATmega328)
- EEPROM ..... 1 Кб (ATmega328)
- Тактовая частота ..... 16 МГц

Контроллер Arduino Uno может получать питание через порт USB или от внешнего источника питания. Источник питания выбирается автоматически. Также к Arduino Uno можно подключить внешний источник питания (например, батарею питания или аккумулятор) к гнездовому разъёму **Vin**. Датчики и исполнительные устройства, подключённые к гнездовым разъёмам **5V**, **3.3V** и **GND**, напротив, сами могут получать питание от Arduino Uno:

- *Вход питания* **Vin**. Используется для подачи питания от внешнего источника через встроенный регулятор напряжения (в отсутствие 5 В от разъема USB или другого источника питания).
- *Выход питания* **5V** (анод). Регулируемый источник напряжения, используемый для питания микроконтроллера и компонентов на плате. Питание может подаваться от вывода VIN через регулятор напряжения, или от разъема USB, или другого регулируемого источника напряжения 5 В.
- *Выход питания* **3V3** (анод). Напряжение на выводе 3,3 В генерируемое встроенным регулятором на плате. Максимальное потребление тока 50 мА.
- *Вход/выход* **GND**. Выводы заземления.

Каждый из 14 цифровых выводов Arduino Uno может быть настроен как вход или выход, используя функции **pinMode(…)**, **digitalWrite(…)**, и **digitalRead(…)**. Выходное напряжение для лог. 0 составляет 0В, а для лог. 1 – 5В. Некоторые цифровые выводы имеют дополнительные функции:

- Последовательная шина: **0 (RX)** и **1 (TX)**. Выводы используются для получения (RX) и передачи (TX) данных через интерфейс RS-232C с TTL-уровнями. Данные выводы подключены к микросхеме преобразователя интерфейсов USB-to-TTL.
- Внешнее прерывание: **2** и **3**. Данные выводы могут быть сконфигурированы на вызов прерывания либо на младшем значении, либо на переднем или заднем фронте, или при изменении значения (функция **attachInterrupt(…)**).
- ШИМ: **3**, **5**, **6**, **9**, **10**, и **11**. Любой из выводов обеспечивает ШИМ с разрешением 8 бит при помощи функции **analogWrite(…)**.
- SPI: **10 (SS)**, **11 (MOSI)**, **12 (MISO)**, **13 (SCK)**. Посредством данных выводов осуществляется связь через интерфейс SPI, для чего используется соответствующая библиотека.
- LED: **13**. Встроенный светодиод, подключенный к цифровому выводу 13. Если значение на выводе имеет высокий уровень, то светодиод «загорается».

На платформе Arduino Uno установлены 6 аналоговых входов (обозначенных как **A0..A5**), каждый разрешением 10 бит (т.е. может принимать 1024 различных значения). Стандартно выводы имеют диапазон измерения от 0 до 5 В относительно земли, тем не менее имеется возможность изменить верхний предел посредством вывода **AREF** (Опорное напряжение для аналоговых входов) и функции **analogReference(…)**. Некоторые аналоговые выводы также имеют дополнительные функции.

Цифро-аналоговый преобразователь

Цифро-аналоговый преобразователь (ЦАП) предназначен для преобразования числа в виде двоичного кода, в напряжение или ток, пропорциональные значению цифрового кода. Схемотехника цифро-аналоговых преобразователей весьма разнообразна. На рисунке 1 представлена классификационная схема ЦАП по схемотехническим признакам [2].

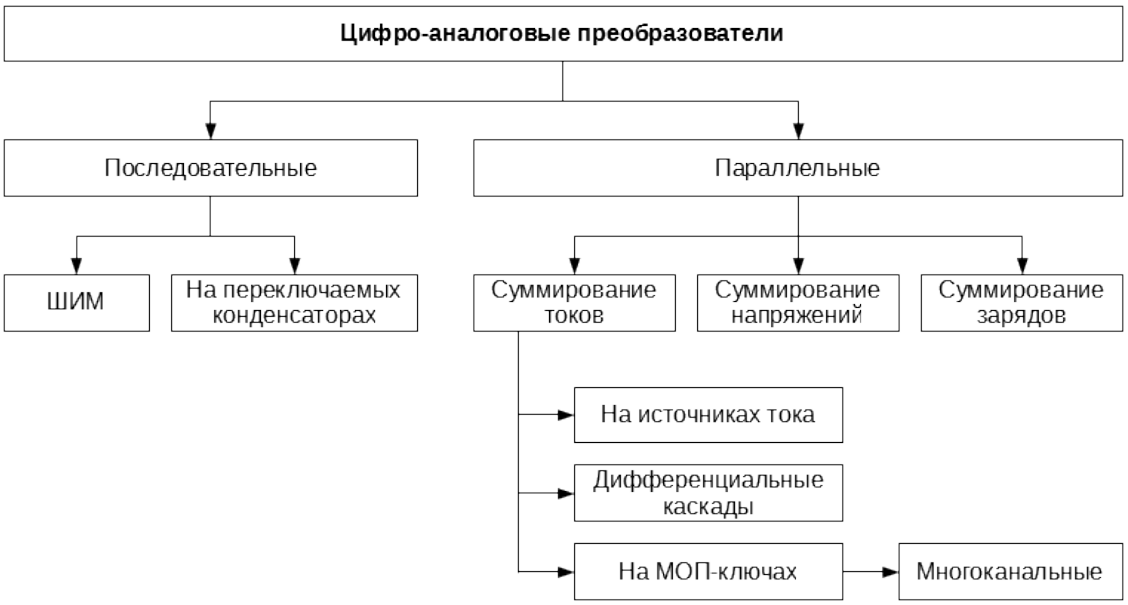


Рисунок 1 – Классификация ЦАП по схемотехническим признакам

Кроме этого, интегральные микросхемы (ИМС) цифро-аналоговых преобразователей классифицируются по следующим признакам [2]:

- По виду выходного сигнала:
  - с токовым выходом;
  - выходом в виде напряжения.
- По типу цифрового интерфейса:
  - с последовательным вводом;
  - с параллельным вводом входного кода.
- По числу ЦАП на кристалле:
  - одноканальные;
  - многоканальные.
- По быстродействию:
  - умеренного быстродействия;
  - высокого быстродействия.
- По разрядности:
  - с малым разрешением;
  - с высоким разрешением.

Основными характеристиками ЦАП являются [3]:

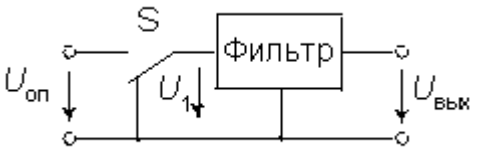
- диапазон выходных напряжений;
- динамический диапазон;
- выходной ток (или характеристики номинальной нагрузки);

- полоса частот воспроизведения выходного сигнала;
- период (частота) преобразования, для асинхронных ЦАП нормируется время преобразования;
- время установления выходного сигнала,
- коэффициент нелинейных искажений,
- погрешность воспроизведения напряжения постоянного и переменного тока.

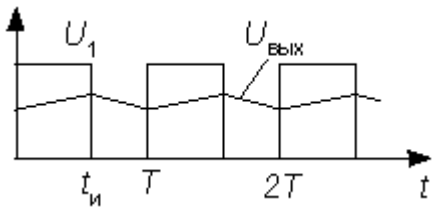
Последовательные ЦАП

ЦАП с широтно-импульсной модуляцией

Очень часто ЦАП входит в состав микропроцессорных систем. В этом случае, если не требуется высокое быстродействие, цифро-аналоговое преобразование может быть очень просто осуществлено с помощью широтно-импульсной модуляции (ШИМ, Pulse-Width Modulation (PWM)). Схема ЦАП с ШИМ приведена на рисунке 2а. Широтно-импульсный модулятор – простейший тип ЦАП. Стабильный источник тока или напряжения периодически включается на время, пропорциональное преобразуемому цифровому коду, далее полученная импульсная последовательность фильтруется аналоговым фильтром нижних частот (рисунок 2б). Такой способ часто используется для управления скоростью вращения электромоторов.



а) Схема ЦАП с ШИМ;



б) Последовательность импульсов на выходе ЦАП с ШИМ.

Рисунок 2 – ЦАП с широтно-импульсной модуляцией

Наиболее просто организуется цифро-аналоговое преобразование в том случае, если микроконтроллер имеет встроенную функцию широтно-импульсного преобразования. Контроллер с помощью своего таймера/счетчика формирует последовательность импульсов, относительная длительность которых $g = t_и/T$ (рисунок 2б) определяется соотношением:

$$\gamma = \frac{D}{2^N} \tag{1}$$

где: N – разрядность преобразования,
D – преобразуемый код.

Фильтр нижних частот сглаживает импульсы, выделяя среднее значение напряжения. В результате выходное напряжение преобразователя:

$$U_{вых} = \gamma U_{оп} = \frac{D U_{оп}}{2^N} \tag{2}$$

где: N – разрядность преобразования,
D – преобразуемый код,
$U_{оп}$ – опорное напряжение ЦАП.

Рассмотренная схема обеспечивает почти идеальную линейность преобразования, не содержит прецизионных элементов (за исключением источника опорного напряжения). Основной ее недостаток – низкое быстродействие [2].

Последовательный ЦАП на переключаемых конденсаторах

Рассмотренная выше схема ЦАП с ШИМ вначале преобразует цифровой код во временной интервал, который формируется с помощью двоичного счетчика квант за квантом, поэтому для получения *N*-разрядного преобразования необходимы 2^N временных квантов (тактов). Схема последовательного ЦАП, приведенная на рисунке 3, позволяет выполнить цифро-аналоговое преобразование за значительно меньшее число тактов.

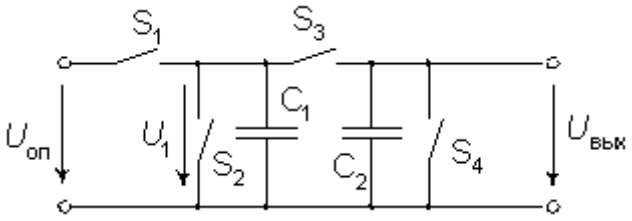


Рисунок 3 – Схема последовательного ЦАП на переключаемых конденсаторах

В этой схеме емкости конденсаторов C_1 и C_2 равны. Перед началом цикла преобразования конденсатор C_2 разряжается ключом S_4. Входное двоичное слово задается в виде последовательного кода. Его преобразование осуществляется последовательно, начиная с младшего разряда *D0*. Каждый такт преобразования состоит из двух полутактов. В первом полутакте конденсатор C_1 заряжается до опорного напряжения $U_{\text{оп}}$ при *D0=1* посредством замыкания ключа S_1 или разряжается до нуля при *D0=0* путем замыкания ключа S_2. Во втором полутакте при разомкнутых ключах S_1, S_2 и S_4 замыкается ключ S_3, что вызывает деление заряда пополам между C_1 и C_2. В результате получаем:

$$U_{\text{вых}}(0) = U_1(0) = (D_0/2)U_{\text{оп}} \tag{3}$$

Пока на конденсаторе C_2 сохраняется заряд, процедура заряда конденсатора C_1 должна быть повторена для следующего разряда D_1 входного слова. После нового цикла перезарядки напряжение на конденсаторах будет:

$$U_{\text{вых}}(1) = U_1(1) = \frac{(D_1 + D_0/2)U_{\text{оп}}}{2} = \frac{(2D_1 + D_0)U_{\text{оп}}}{4} \tag{4}$$

Точно также выполняется преобразование для остальных разрядов слова. В результате для *N*-разрядного ЦАП выходное напряжение будет равно:

$$U_{\text{вых}}(N-1) = U_1(N-1) = \frac{U_{\text{оп}}}{2^N}\sum_{k=0}^{N-1} D_k 2^k = \frac{U_{\text{оп}}}{2^N} D \tag{5}$$

Если требуется сохранять результат преобразования сколь-нибудь продолжительное время, к выходу схемы следует подключить устройство выборки и хранения (УВХ). После окончания цикла преобразования следует провести цикл выборки, перевести УВХ в режим хранения и вновь начать преобразование.

Таким образом, представленная схема выполняет преобразование входного кода за *2N* квантов, что значительно меньше, чем у ЦАП с ШИМ. Здесь требуется только два согласованных конденсатора небольшой емкости. Конфигурация аналоговой части схемы

не зависит от разрядности преобразуемого кода. Однако по быстродействию последовательный ЦАП значительно уступает параллельным цифро-аналоговым преобразователям, что ограничивает область его применения [2].

Параллельные ЦАП

Среди параллельных ЦАП наблюдается наибольшее разнообразие. К тому же по многим параметрам они значительно превосходят последовательные ЦАП.

Дельта-сигма ЦАП

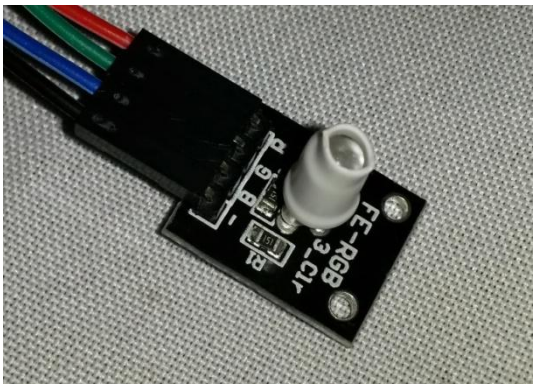
Дельта-сигма ЦАП относятся к классу ЦАП передискретизации. Их работа основана на изменяемой плотности импульсов. Такие преобразователи при небольшой собственной разрядности обеспечивают достаточно большую разрядность итогового преобразования. На ЦАП поступает импульсный сигнал с постоянной длительностью импульсов, но с изменяемой скважностью, который формируется с помощью цепи отрицательной обратной связи. Цепь отрицательной обратной связи выполняет функцию фильтра высоких частот для шума квантования. Быстродействие дельта-сигма ЦАП достигает сотни тысяч отсчетов в секунду, разрядность – 24 бит. [4].

Ход работы

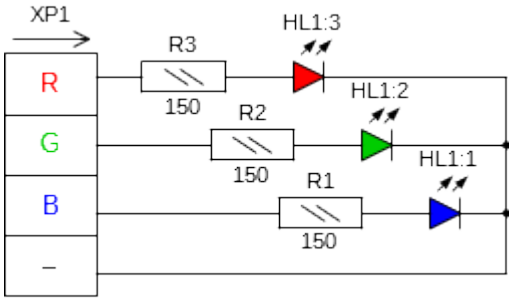
Если лабораторная работа выполняется не в аудитории, то её можно выполнить в симуляторе, например, Virtual Breadboard (http://www.virtualbreadboard.com/), Virtronics Simulator for Arduino (http://www.virtronics.com.au/), Autodesk 123D Circuits (https://circuits.io/), Fritzing (http://fritzing.org/home/) или любом другом.

Управление RGB-светодиодом с общим катодом через дискретные порты

Изучение принципов работы с исполнительными устройствами через дискретные интерфейсы контроллера Arduino начнём с простого. Роль исполнительного устройства (а точнее даже трёх исполнительных устройств) будет играть RGB-светодиод на печатной плате (рисунок 4а). RGB-светодиод содержит в одном корпусе три светодиода (триаду) – красный (R, Red), зелёный (G, Green) и синий (B, Blue).



а) RGB-светодиод с общим катодом на печатной плате;



б) электрическая принципиальная схема печатной платы с RGB-светодиодом с общим катодом

Рисунок 4 – RGB-светодиод с общим катодом

У RGB-светодиода с общим катодом отрицательные выводы (катоды) светоизлучающих диодов (СИД) соединены общей точкой, а положительные выводы (аноды) через токоограничивающие резисторы выведены на три штырька разъёма (рисунок 4б).

Для того, чтобы управлять включением и выключением отдельных светодиодов триады, подключим также кнопку на печатной плате с подтягивающим резистором (рисунок 5).

Рисунок 5 – Кнопка с подтягивающим резистором на печатной плате

Электрическая схема подключения RGB-светодиода с общим катодом и кнопки к Arduino показана на рисунке 6, а соответствующий код – в листинге 1.

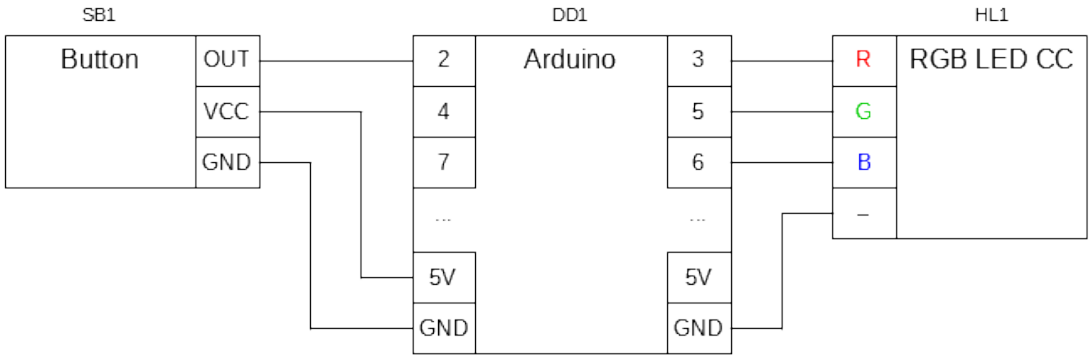


Рисунок 6 – Электрическая схема подключения RGB-светодиода с общим катодом и кнопки к Arduino

Листинг 1 – Управление одним исполнительным устройством на примере RGB-светодиода с общим катодом

```
#define RLED_PIN 3 //Номер порта для подключения красного светодиода
#define GLED_PIN 5 //Номер порта для подключения зелёного светодиода
#define BLED_PIN 6 //Номер порта для подключения синего светодиода
#define BTN_PIN 2  //Номер порта для подключения кнопки

bool btnState; //Состояние кнопки (нажата/не нажата)

void setup()
{
  // put your setup code here, to run once:
  pinMode(RLED_PIN, OUTPUT); //Настроить порты
  pinMode(GLED_PIN, OUTPUT); //светодиодов на вывод
  pinMode(BLED_PIN, OUTPUT); //дискретных сигналов
  pinMode(BTN_PIN, INPUT);   //Настроить порт кнопки на ввод сигнала
}
```

```
void loop()
{
  // put your main code here, to run repeatedly:
  btnState = digitalRead(BTN_PIN); //Считать состояние кнопки в переменную
  if (btnState == 1) digitalWrite(RLED_PIN, HIGH);
    else digitalWrite(RLED_PIN, LOW);
  delay(50); //Задержка программы длительностью 50 мс
}
```

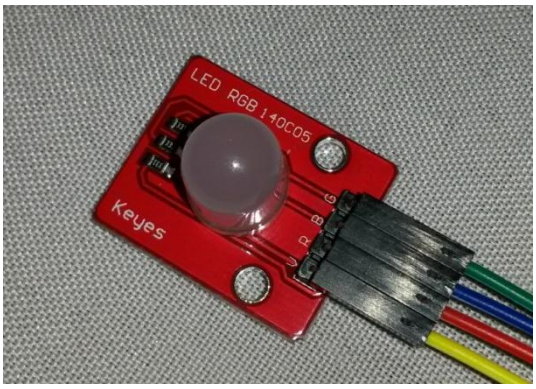
После загрузки программы в плату Arduino можно посмотреть результат её выполнения – при нажатии на кнопку, подачей на выход контроллера высокого логического уровня (лог. 1), включается красный светодиод, а после отпускания кнопки он гаснет (на выход подаётся низкий логический уровень, лог. 0).

Модифицируйте программу так, чтобы:

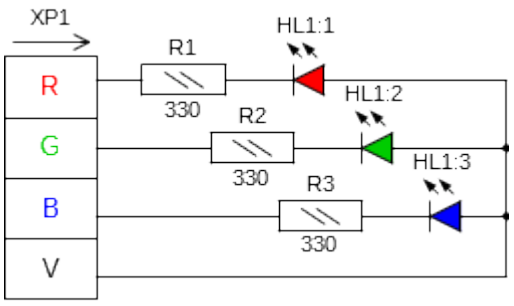
1) включался светодиод другого цвета;
2) включалось одновременно несколько светодиодов;
3) при первом нажатии и отпускании кнопки зажигался один (или несколько) светодиод, а при втором – он гас.

Управление RGB-светодиодом с общим анодом через дискретные порты

Помимо RGB-светодиодов с общим катодом существуют также и RGB-светодиоды с общим анодом (общий «+»). У RGB-светодиода с общим анодом (рисунок 7а), наоборот, положительные выводы (аноды) светоизлучающих диодов соединены общей точкой, а отрицательные выводы (катоды) через токоограничивающие резисторы выведены на три штырька разъёма (рисунок 7б). Теперь для того, чтобы зажечь отдельный светодиод триады, нужно будет вывести на порт, к которому подключён светодиод, сигнал низкого уровня (лог. 0), а чтобы погасить – сигнал высокого уровня (лог. 1).



а) RGB-светодиод с общим анодом на печатной плате;



б) электрическая принципиальная схема печатной платы с RGB-светодиодом с общим анодом

Рисунок 7 – RGB-светодиод с общим анодом

Электрическая схема подключения RGB-светодиода с общим анодом и кнопки к Arduino показана на рисунке 8, а модифицированный код программы – в листинге 2.

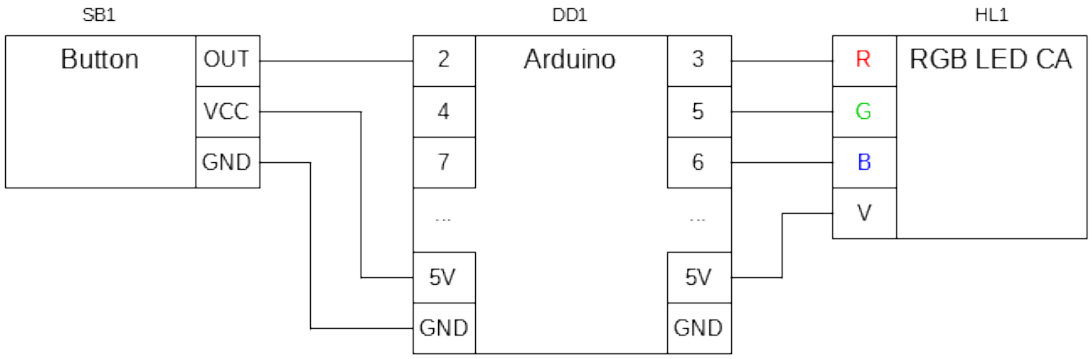


Рисунок 8 – Электрическая схема подключения RGB-светодиода с общим анодом и кнопки к Arduino

Листинг 2 – Управление одним исполнительным устройством на примере RGB-светодиода с общим анодом

```
#define RLED_PIN 3 //Номер порта для подключения красного светодиода
#define GLED_PIN 5 //Номер порта для подключения зелёного светодиода
#define BLED_PIN 6 //Номер порта для подключения синего светодиода
#define BTN_PIN 2  //Номер порта для подключения кнопки

bool btnState; //Состояние кнопки (нажата/не нажата)

void setup()
{
  // put your setup code here, to run once:
  pinMode(RLED_PIN, OUTPUT); //Настроить порты
  pinMode(GLED_PIN, OUTPUT); //светодиодов на вывод
  pinMode(BLED_PIN, OUTPUT); //дискретных сигналов
  digitalWrite(RLED_PIN, HIGH); //Погасить все
  digitalWrite(GLED_PIN, HIGH); //светодиоды
  digitalWrite(BLED_PIN, HIGH); //триады
  pinMode(BTN_PIN, INPUT);   //Настроить порт кнопки на ввод сигнала
}

void loop()
{
  // put your main code here, to run repeatedly:
  btnState = digitalRead(BTN_PIN); //Считать состояние кнопки в переменную
  if (btnState == 1) digitalWrite(RLED_PIN, LOW);
    else digitalWrite(RLED_PIN, HIGH);
  delay(50); //Задержка программы длительностью 50 мс
}
```

Предложите другие способы модифицировать программы.

Усложним задачу. Теперь будем управлять всей триадой светодиодов. Единственной кнопкой будем перебирать все возможные цвета RGB-светодиода, подключённого к дискретным портам вывода, а всего их можно воспроизвести 2^3-1=7 цветов и одно состояние «погашены все». Можно написать программу, где бы устанавливалось каждое состояние триады светодиодов, путём установки сигналов (**HIGH/LOW**) на трёх дискретных выводах. Но можно найти более изящное решение. Для этого построим куб соседних чисел, описывающий все возможные состояния триады светодиодов (рисунок 9).

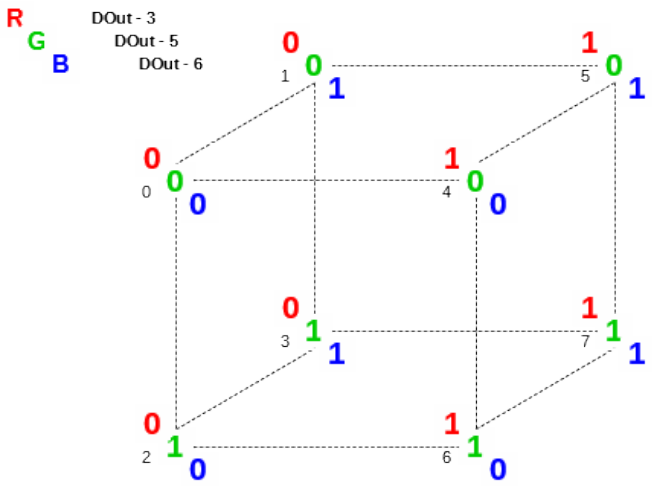


Рисунок 9 – Куб соседних чисел состояний триады RGB-светодиода

Для RGB-светодиода с общим катодом состояние лог. 1 соответствует зажжённому светодиоду, а лог. 0 – погашенному. Для RGB-светодиода с общим катодом наоборот – состояние лог. 1 соответствует погашенному светодиоду, а лог. 0 – зажжённому. Для обхода куба нужно выбрать начальное состояние, пройти по всем дугам по одному разу, и вернуться в начальное состояние. Код программы приведён в листинге 3.

Листинг 3 – Управление тремя исполнительным устройством на примере RGB-светодиода с общим анодом

```
#define RLED_PIN 3 //Номер порта для подключения красного светодиода
#define GLED_PIN 5 //Номер порта для подключения зелёного светодиода
#define BLED_PIN 6 //Номер порта для подключения синего светодиода
#define BTN_PIN 2  //Номер порта для подключения кнопки

bool btnState = 0; //Состояние кнопки (нажата/не нажата)
byte cntClick = 0; //Счётчик количества нажатий кнопки

void setup()
{
  // put your setup code here, to run once:
  pinMode(RLED_PIN, OUTPUT); //Настроить порты
  pinMode(GLED_PIN, OUTPUT); //светодиодов на вывод
  pinMode(BLED_PIN, OUTPUT); //дискретных сигналов
  digitalWrite(RLED_PIN, HIGH); //Погасить все
  digitalWrite(GLED_PIN, HIGH); //светодиоды
  digitalWrite(BLED_PIN, HIGH); //триады
  pinMode(BTN_PIN, INPUT); //Настроить порт кнопки на ввод сигнала
}

void loop()
{
  // put your main code here, to run repeatedly:
  btnState = digitalRead(BTN_PIN); //Считать состояние кнопки в переменную
  if (btnState == 1) //Если кнопка нажата, то...
  {
    while (btnState == 1) //ждать, пока кнопка не отпущена
    {
      delay(10); //Через 10 мс
      btnState = digitalRead(BTN_PIN); //прочитать состояние кнопки ещё раз
    }
    cntClick++; //Теперь можно сосчитать нажатие на кнопку
    if (cntClick > 7) cntClick = 0; //Если это 8-е нажатие, считать его 0-вым
  }
```

```
  //Переключать состояние СИД согласно порядку обхода куба соседних чисел
  switch (cntClick)
  {
    case 0: digitalWrite(BLED_PIN, HIGH); break; //Состояние 7 - Выключены
    case 1: digitalWrite(RLED_PIN, LOW); break;  //Состояние 3 - Красный
    case 2: digitalWrite(GLED_PIN, LOW); break;  //Состояние 1 - Жёлтый
    case 3: digitalWrite(RLED_PIN, HIGH); break; //Состояние 5 - Зелёный
    case 4: digitalWrite(BLED_PIN, LOW); break;  //Состояние 4 - Голубой
    case 5: digitalWrite(RLED_PIN, LOW); break;  //Состояние 0 - Белый
    case 6: digitalWrite(GLED_PIN, HIGH); break; //Состояние 2 - Фиолетовый
    case 7: digitalWrite(RLED_PIN, HIGH); break; //Состояние 6 - Синий
  }
  delay(50); //Задержка программы длительностью 50 мс
}
```

Используйте другие способы обхода куба соседних чисел.

Управление RGB-светодиодом с общим анодом через аналоговые порты

Продолжим изучение принципов работы с исполнительными устройствами, теперь через аналоговые выходы контроллера Arduino. Роль исполнительного устройства (а точнее трёх) будет по-прежнему играть RGB-светодиод с общим анодом на печатной плате (рисунок 7а). В качестве датчика будем использовать энкодер на печатной плате с токоограничивающими резисторами (рисунок 10). Энкодер имеет 20 положений на один полный оборот с шагом 18°.

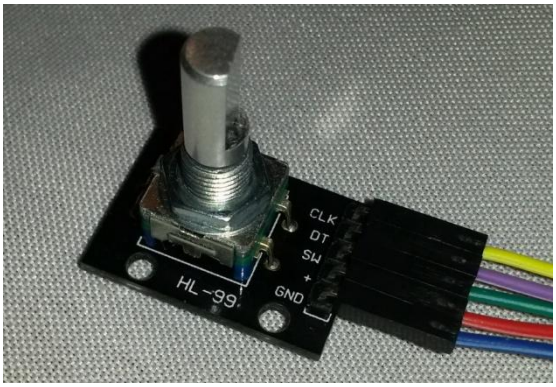


Рисунок 10 – Внешний вид печатной платы с энкодером

Электрическая схема подключения RGB-светодиода с общим анодом и энкодера к Arduino показана на рисунке 11, а код программы приведён в листинге 4.

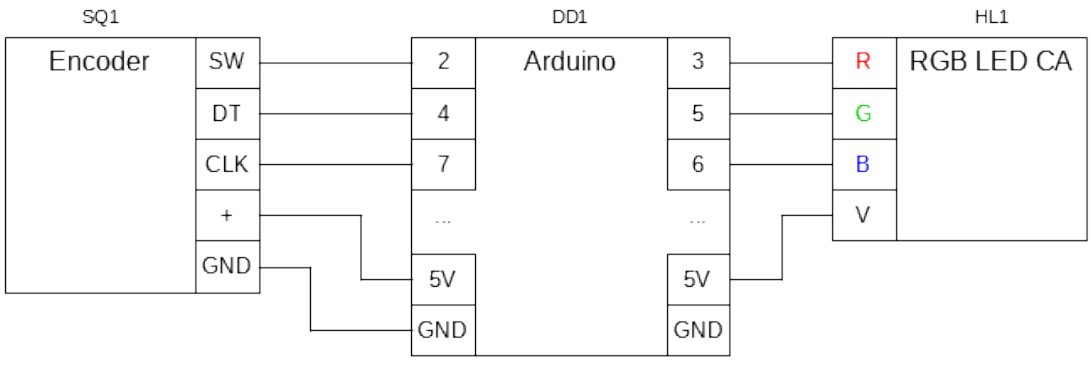


Рисунок 11 – Электрическая схема подключения RGB-светодиода с общим анодом и энкодера к Arduino

Листинг 4 – Управление тремя исполнительным устройством на примере RGB-светодиода с общим анодом через аналоговые выходы

```
#define SW_PIN 2      //SW - кнопка энкодера
#define CLK_A_PIN 7   //CLK - сигнал A энкодера
#define DAT_B_PIN 4   //DT - сигнал B энкодера
#define R_PIN 3       //Номер порта для подключения красного светодиода
#define G_PIN 5       //Номер порта для подключения зелёного светодиода
#define B_PIN 6       //Номер порта для подключения синего светодиода

int swState = 0;      //Состояние кнопки (нажата/не нажата)
bool clkAPre = HIGH; //Предыдущее значение сигнала на линии A энкодера
bool clkACur = LOW;  //Текущее значение сигнала на линии A энкодера
int rgbLed[3] = {255, 255, 255}; //0 - максимальная яркость, 255 - погашены

void setup()
{
  // put your setup code here, to run once:
  pinMode(SW_PIN, INPUT); //Настроить порт кнопки на ввод дискретного сигнала
  digitalWrite(SW_PIN, HIGH); //Подключить внутренний подтягивающий резистор
  pinMode(CLK_A_PIN, INPUT);  //Сигнал A энкодера
  pinMode(DAT_B_PIN, INPUT);  //Сигнал B энкодера
  analogWrite(R_PIN, rgbLed[0]); //Погасить все
  analogWrite(G_PIN, rgbLed[1]); //светодиоды
  analogWrite(B_PIN, rgbLed[2]); //триады
}

void loop()
{
  // put your main code here, to run repeatedly:
  if (digitalRead(SW_PIN) == LOW) //Если кнопка нажата, то...
  {
    while (digitalRead(SW_PIN) == LOW); //...ждать, пока кнопка не отпущена
    swState++; //Теперь можно сосчитать нажатие на кнопку
    if (swState > 2) swState = 0; //Если это 3-е нажатие, считать его нулевым
  }

  clkACur = digitalRead(CLK_A_PIN); //Считать текущее значение сигнала A
  //Если предыдущий уровень на линии A был низким, а сейчас стал высоким, ...
  if ((clkAPre == LOW) && (clkACur == HIGH))
  {
    //...и, если сигнал B низкого уровня, то...
    if (digitalRead(DAT_B_PIN) == LOW)
    {
      //...значит энкодер вращается по часовой стрелке, ...
      rgbLed[swState]--; //Прибавить яркость выбранного кнопкой светодиода
      //Яркость RGB-светодиода растёт при уменьшении скважности ШИМ
      if (rgbLed[swState] < 230) rgbLed[swState] = 230; //Ограничить яркость
    } else
    {
      //...а иначе - против часовой стрелки
      rgbLed[swState]++; //Убавить яркость выбранного кнопкой светодиода
      //Яркость RGB-светодиода уменьшается при увеличении скважности ШИМ
      if (rgbLed[swState] > 255) rgbLed[swState] = 255; //Погасить светодиод
    }
    analogWrite(R_PIN, rgbLed[0]); //Установить яркость красного светодиода
    analogWrite(G_PIN, rgbLed[1]); //Установить яркость зелёного светодиода
    analogWrite(B_PIN, rgbLed[2]); //Установить яркость синего светодиода
  }
  clkAPre = clkACur; //Сохранить текущее значение сигнала на линии A энкодера
}
```

Подключите RGB-светодиод с общим катодом и измените программу так, чтобы она работала правильно.

Задания по вариантам

Напишите программу для Arduino, где:

1) Светодиод зажигается и гаснет, если держать нажатой кнопку в течение 1 секунды.
2) Светодиод зажигается и гаснет двойным нажатием на кнопку.
3) Пока кнопка нажата цвета меняются, а при отпускании – остаётся последний цвет.
4) При одинарном нажатии кнопки зажигается красный светодиод, при двойном – зелёный, а при тройном – синий. Через 2 секунды светодиод гаснет.
5) Для включения и выключения светодиода нужно держать кнопку нажатой в течение 1 секунды.
6) При нажатии кнопки светодиод плавно зажигается. При повторном нажатии кнопки сразу выключается.
7) Пока кнопка нажата яркость светодиода плавно нарастает, после отпускания кнопки светодиод плавно гаснет.
8) При одинарном нажатии кнопки светодиод загорается сразу, а при двойном – плавно. Через 5 секунд светодиод выключается.
9) При одинарном нажатии кнопки красный светодиод загорается плавно, при двойном нажатии – плавно зелёный, а при тройном нажатии – плавно синий. Через 5 секунд светодиод гаснет.
10) При первом нажатии и удержании кнопки плавно возрастает яркость красного светодиода, а после отпускания кнопки яркость запоминается. При втором нажатии кнопки плавно возрастает яркость синего светодиода, а после отпускания кнопки яркость запоминается. При третьем нажатии получается результирующий цвет, а при четвёртом – светодиод гаснет.
11) При вращении энкодера по часовой стрелке меняются цвета от красного до фиолетового, а против часовой – наоборот.
12) При вращении энкодера светодиод мигает с частотой, пропорциональной скорости вращения.
13) При вращении энкодера светодиод светит с яркостью, пропорциональной скорости вращения энкодера.
14) При низкой скорости вращения энкодера загорается зелёный светодиод, а при высокой – красный.
15) При вращении энкодера по часовой стрелке частота мигания светодиода возрастает, а при вращении против часовой стрелки – уменьшается.
16) При вращении энкодера выбирается яркость красного светодиода, после нажатия кнопки – синего, ещё после нажатия появляется результирующий цвет.
17) Сейфовый замок на энкодере: светодиод зажигается, если энкодер повёрнут на два шага по часовой стрелке, затем на три – против, и затем на четыре опять по часовой стрелке. Через 5 секунд светодиод гаснет. При нажатии на кнопку состояние сбрасывается.

Требования к содержанию отчета

Отчет по лабораторной работе должен содержать следующие пункты:

1) Цель и задачи лабораторной работы.
2) Краткие теоретические сведения.
3) Управление RGB-светодиодом с общим катодом через дискретные порты.
4) Управление RGB-светодиодом с общим анодом через дискретные порты.
5) Управление RGB-светодиодом с общим анодом через аналоговые порты.
6) Задание согласно своему варианту.
7) Ответы на контрольные вопросы.
8) Вывод по лабораторной работе.

Контрольные вопросы

1) Какое количество дискретных выходов доступно у Arduino Uno? Перечислите их.
2) Какое количество аналоговых выходов доступно у Arduino Uno? Перечислите их.
3) Приведите классификацию цифро-аналоговых преобразователей.
4) Назовите основные характеристики ЦАП.
5) Перечислите преимущества и недостатки ЦАП с широтно-импульсной модуляцией.
6) Перечислите преимущества и недостатки последовательных ЦАП на переключаемых конденсаторах.
7) Назовите виды параллельных ЦАП.
8) Где используются дельта-сигма ЦАП?

Список использованных источников

1) Аппаратная платформа Arduino (Arduino Uno) [Электронный ресурс]. URL: http://arduino.ru/Hardware/ArduinoBoardUno (дата обращения: 01.09.2017)
2) Цифро-аналоговые преобразователи [Электронный ресурс]. URL: http://www.limi.ru/dacs/dacsindex.htm (дата обращения: 01.09.2017)
3) Терминология ЦАП [Электронный ресурс]. URL: http://www.lcard.ru/lexicon/dac (дата обращения: 01.09.2017)
4) Электроника НТБ - научно-технический журнал; цифроаналоговые преобразователи [Электронный ресурс]. URL: http://www.electronics.ru/journal/article/319 (дата обращения: 01.09.2017)