

Лабораторная работа № 4

Подключение 7-сегментных индикаторов: динамическая индикация. Подключение матричной клавиатуры к Arduino

Цель лабораторной работы: познакомиться с принципами организации динамической индикации на Arduino, а также научиться использовать матричное подключение кнопок (матричной клавиатуры).

Теоретическая часть

Динамическая индикация

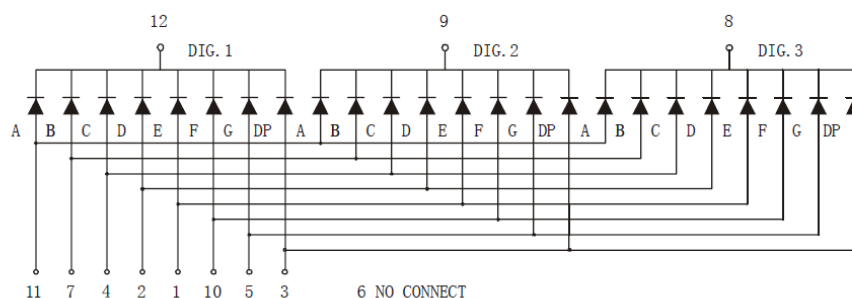
В настоящее время для отображения информации всё чаще используются графические дисплеи, однако, 7-сегментные индикаторы также не утратили своего значения. Если требуется лишь отображение чисел, то они могут стать более предпочтительным вариантом, т.к. просты в управлении и могут использоваться совместно с любым микроконтроллером с достаточным количеством выводов. Жидкокристаллические 7-сегментные индикаторы обладают сверхнизким энергопотреблением (например, в электронных часах, вместе со схемой управления работают от одной батарейки в течение нескольких лет).

Существует большое разнообразие индикаторов: с разным цветом свечения сегментов, разного размера, отличающиеся схемой подключения светодиодов. При необходимости отображения нескольких разрядов можно установить несколько одноразрядных индикаторов рядом на печатной плате либо выбрать нужный вариант многоразрядного индикатора.

По количеству разрядов (цифр) динамические 7-сегментные индикаторы бывают одноразрядные, двухразрядные, трехразрядные, четырехразрядные и очень редко – шестиразрядные. Изготавливаются они с общим анодом и общим катодом. Схемы соединения светодиодов отдельных сегментов представлены на рисунке 1.

Свое название 7-сегментные индикаторы получили в связи с тем, что изображение символа формируется с помощью семи отдельно управляемых (подсвечиваемых светодиодом) элементов - сегментов. Эти элементы позволяют отобразить любую цифру 0..9, а также некоторые другие символы, например: '-', 'A', 'b', 'C', 'd', 'E', 'F' и другие. Это даёт возможность использовать индикатор для вывода положительных и отрицательных десятичных и шестнадцатеричных чисел и даже текстовых сообщений. Обычно индикатор имеет также восьмой элемент - точку, используемую при отображении чисел с десятичной точкой. Сегменты индикатора обозначают буквами a, b, ..., g (Рисунок 2).

SP20361



SP10361

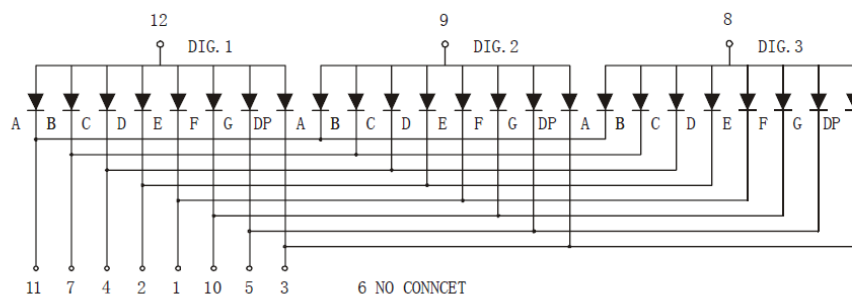


Рисунок 1 – Схема соединения светодиодов трёхразрядного 7-сегментного индикатора

В том случае, если светодиоды в индикаторе имеют соединённые вместе аноды (схема с общим анодом), общий анод подключается к источнику напряжения VDD, а катоды светодиодов - сегментов подключаются к схеме управления (например, микроконтроллеру), которая отвечает за формирование изображения на индикаторе. Зажигаются сегменты низким уровнем (лог. “0”) на выводе схемы управления. По отношению к схеме управления ток светодиодов является втекающим, так что могут использоваться интегральные схемы, которые имеют выходы с открытым стоком. Изменяя величину питающего индикатор напряжения VDD, можно регулировать яркость свечения. В противном случае (индикаторы с общим катодом), сегменты зажигаются высоким уровнем сигнала (лог. “1”), а общий катод подключается к GND.

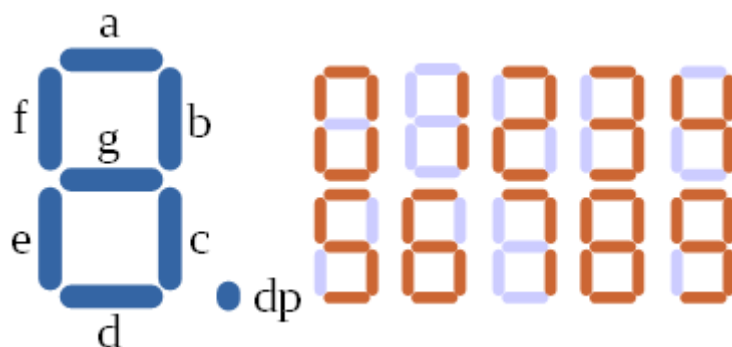


Рисунок 2 – Обозначение сегментов индикатора

Самый простой вид индикации — **статический**. При ее использовании каждый сегмент индикатора постоянно находится в одном из двух состояний — включен или выключен. Ее основное достоинство в том, что после вывода информации, например в сдвигающий регистр, состояние индикатора не изменится, пока не будут изменены данные в этих регистрах. Так же т.к. напряжение на сегментах присутствует постоянно, яркость индикатора будет максимальной. Кроме этих двух плюсов есть также минус: требуется большое число пинов микроконтроллера (один пин – один сегмент), а также большое количество токоограничивающих резисторов — по одному на каждый сегмент.

Как известно, наше зрение обладает свойством инерции, или персистенции. Это способность глаза соединять быстро сменяющиеся изображения в одно. Таким образом, чтобы человек видел на индикаторе трехзначное число, вовсе необязательно зажигать все цифры разом. Достаточно в один момент времени включать только один отдельный индикатор. Переключение между соседними индикаторам должно происходить с большой частотой, чтобы получить эффект персистенции.

Такой подход к выводу данных называется **динамической индикацией**: сегменты зажигаются по очереди. Из ее основных плюсов — требуется гораздо меньше внешних элементов. Основной минус — для нее постоянно требуется внимание процессора. Частота смены сегментов выбирается обычно не ниже 50 Гц. Лучше использовать частоты не кратные 50, иначе при искусственном освещении может появиться мерцание. Частота прерываний считается как:

$$f = \text{Кол-во разрядов} * \text{Частота обновления}$$

Так для 8 разрядов с частотой 60Гц нужно вызывать прерывание с $F=8 \times 60=480\text{Гц}$.

Есть два вида такой индикации - поразрядная и посегментная. Первая наиболее известна и популярна, вторая лучше подходит когда нужно большое количество разрядов (больше 10).

Для формирования изображения символа на индикаторе используют таблицу, которая ставит в соответствие коду символа набор отображаемых сегментов. Приведём пример такой таблицы (Рисунок 3). В этой таблице код символа - его порядковый номер в таблице. Набор сегментов, формирующих символ, рассматривается как двоичное число, сегменту А соответствует младший бит числа. Если бит числа равен 0, то соответствующий сегмент не зажигается при отображении символа, а если равен 1, то зажигается. В

таблице также приводится запись числа, определяющего набор зажигаемых сегментов, в шестнадцатеричной форме.



















#	Символ	Рис.	g	f	e	d	c	b	a	HEX
0	0		0	1	1	1	1	1	1	0x3F
1	1		0	0	0	0	1	1	0	0x06
2	2		1	0	1	1	0	1	1	0x5B
3	3		1	0	0	1	1	1	1	0x4F
4	4		1	1	0	0	1	1	0	0x66
5	5		1	1	0	1	1	0	1	0x6D
6	6		1	1	1	1	1	0	1	0x7D
7	7		0	0(1)	0	0	1	1	1	0x07 (0x27)
8	8		1	1	1	1	1	1	1	0x7F
9	9		1	1	0	1	1	1	1	0x6F
10	A		1	1	1	0	1	1	1	0x77
11	b		1	1	1	1	1	0	0	0x7C
12	C		0	1	1	1	0	0	1	0x39
13	d		1	0	1	1	1	1	0	0x5E
14	E		1	1	1	1	0	0	1	0x79
15	F		1	1	1	0	0	0	1	0x71
16	<Пробел>		0	0	0	0	0	0	0	0x00
17	-		1	0	0	0	0	0	0	0x40

Рисунок 3 – Таблица соответствия для 7-сегментного индикатора

Рассмотрим пример организации динамической индикации на Arduino:

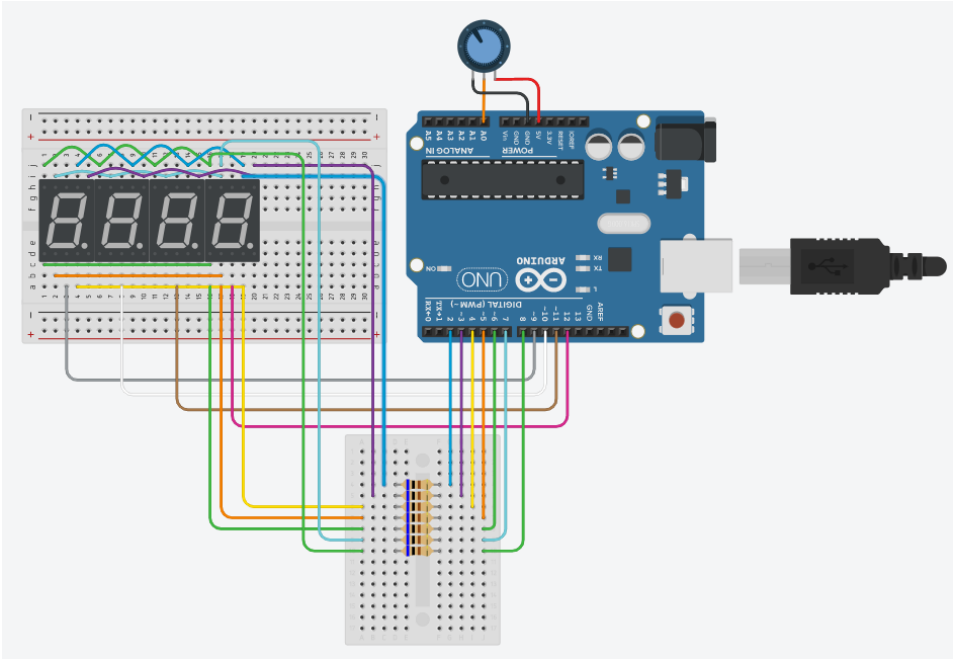


Рисунок 4 – Макет для изучения принципов организации динамической индикации

Для начала необходимо в соответствии с datasheet на индикатор составить таблицу соответствия (истинности) сегментов и пинов микроконтроллера:

```
// таблица истинности для индикатора с ОА
//
//      1  2  4  8  16 32 64 - разряды числа
//
//      7  8  9  10 11 12 13 - пин
//      a  b  c  d  e  f  g - сегмент
//
// 1| 1 0 0 0 1 1 1 - 121
// 2| 0 0 1 0 0 1 0 - 36
// 3| 0 0 0 0 1 1 0 - 48
// 4| 1 0 0 1 1 0 0 - 25
// 5| 0 1 0 0 1 0 0 - 18
// 6| 0 1 0 0 0 0 0 - 2
// 7| 0 0 0 1 1 1 1 - 120
// 8| 0 0 0 0 0 0 0 - 0
// 9| 0 0 0 0 1 0 0 - 16
// 0| 0 0 0 0 0 0 1 - 64
//
//
// таблица истинности для индикатора с ОК
//
//      1  2  4  8  16 32 64 - разряды числа
//
//      7  8  9  10 11 12 13 - пин
//      a  b  c  d  e  f  g - сегмент
//
// 1| 0 1 1 0 0 0 0 - 6
// 2| 1 1 0 1 1 0 1 - 91
// 3| 1 1 1 1 0 0 1 - 79
// 4| 0 1 1 0 0 1 1 - 102
// 5| 1 0 1 1 0 1 1 - 109
// 6| 1 0 1 1 1 1 1 - 125
// 7| 1 1 1 0 0 0 0 - 7
// 8| 1 1 1 1 1 1 1 - 127
// 9| 1 1 1 1 0 1 1 - 111
// 0| 1 1 1 1 1 1 0 - 63
//
//
```

Рисунок 5 – Таблица истинности для индикаторов с общим анодом и общим катодом

```

//формируем массив исходя из значений таблицы для индикаторов с ОА и ОК

//byte digit[10] = {64, 121, 36, 48, 25, 18, 2, 120, 0, 16};
byte digit[10] = {63, 6, 91, 79, 102, 109, 125, 7, 127, 111};

byte pos[4] = {0, 0, 0, 0};      // храним разряды

int data = 0;                    // данные для отображения

long time0;                      //начальное значение времени

//Включение сегментов по значениям из массива
void ledsOn(byte key) {

    byte mask = 1;
    if (key) {
        for (byte i = 2; i <= 8; i++) {
            digitalWrite(i, key & mask);
            mask = mask << 1;
        }
    }
}

//преобразователь двоичных значений в двоично-десятичные
void binToDec(int data, byte *mas) {

    int pos = 1000;

    for (int i = 0; i < 4; i++) {

        mas[i] = data / pos;

        data %= pos;

        pos /= 10;

    }
}

// динамическая индикация
void dinam(int data) {
    int razr = data;
    byte k = 0;
    binToDec(data, pos);

    //гашение нулей
    for (int i = 4; i > 0; i--) {
        if (razr /= 10)
            k++;
    }

    for (byte i = 0; i < k + 1; i++) {
        delay(7);
        digitalWrite(12 - i, LOW);
        ledsOn(digit[pos[3 - i]]);
        delay(7);
        digitalWrite(12 - i, HIGH);
    }
}

```

```

void setup() {

    Serial.begin(9600);

    // сюда включены аноды (катоды) разрядов
    for (int i = 9; i <= 12; i++) {
        pinMode(i, OUTPUT);
        digitalWrite(i, HIGH);
        //digitalWrite(i, LOW);
    }

    //сегменты индикатора
    for (int i = 2; i <= 8; i++) {
        pinMode(i, OUTPUT);
    }
    time0 = millis();
}

void loop() {

    //-----работа с ацп
    if ((millis() - time0) >= 200) {
        data = analogRead(A0);
        time0 = millis();
    }
    Serial.println(data);
    dinam(data);
}

```

Матричная клавиатура

Клавиатурой называют устройство, состоящее из отдельных кнопок, которое позволяет вводить информацию в компьютер или в иной электронный прибор.

Клавиатуры имеют массу вариантов исполнения. На старых персональных компьютерах использовались клавиатуры с механическими клавишами, которые имели большой ход и были почти также удобны как клавиши на старых печатных машинках. Последние 15-20 лет широко применяются мембранные клавиатуры, которые проще и дешевле в изготовлении.

Матричная клавиатура состоит из кнопок, образующих матрицу $m \times n$, т.е. таблицу, где m — количество строк, а n — количество столбцов. Для примера мы воспользуемся клавиатурой 4x4. Если мы рассмотрим её шлейф, то увидим, что он состоит из 8 дорожек. Дорожки с номерами 1-4 (обозначим их Cols) — это столбцы матрицы с первой по четвёртую, а дорожки с номерами 5-8 (Rows) — строки с первой по четвёртую.

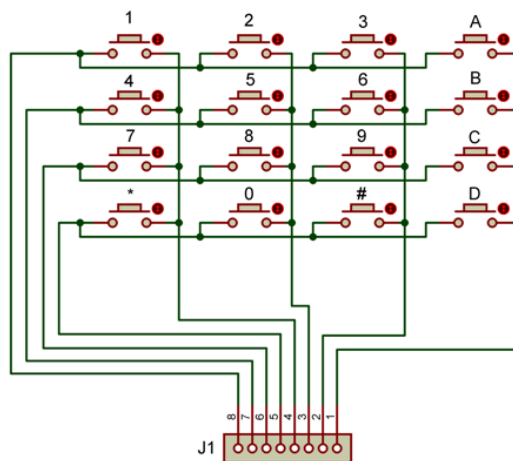


Рисунок 6 – Схема подключения кнопок в матричной клавиатуре

Каждая кнопка — это область пересечения токопроводящих дорожек. При нажатии на участок, обозначающий кнопку, происходит соединение дорожек и замыкание одного из выводов строк 5-8 с одним из выводов столбцов 1-4.

Подавая сигнал на один провод из первой четвёрки, и снимая его на проводах второй четвёрки, можно определить, какие кнопки в определённой группе зажаты в данный момент.

Рассмотрим пример использования матричной клавиатуры без использования библиотечных функций. Схема подключения представлена на рисунке 7.

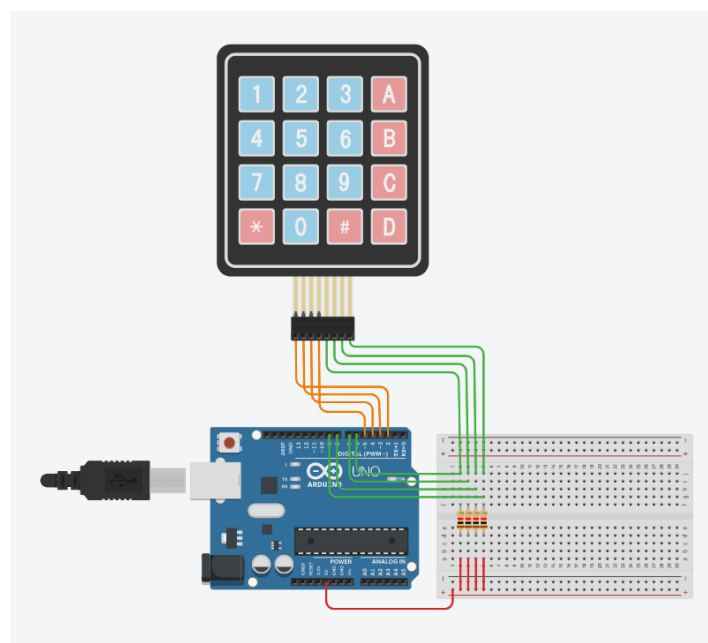


Рисунок 7 – Схема макета для работы с матричной клавиатурой

Далее, формируем программу, которая выводит в последовательный порт номер нажатой кнопки:

```
int tmp, tmp1 = 0, z;
void setup() {
  Serial.begin(9600);
  for (int i = 6; i <= 9; i++){
    pinMode(i, INPUT);
  }
  for (int i = 2; i < 6; i++)
    pinMode (i,OUTPUT);
}

void loop() {
  for ( int i = 2; i < 6; i++) {
    digitalWrite (i, 0);
    for (int j = 6; j <= 9; j++){
      if (digitalRead(j) == LOW){
        for ( int i = 2; i < 6; i++)
          if (digitalRead(i) == LOW) tmp1 = i;
        delay (10);
        if (digitalRead(j) == LOW){
          tmp = ((j-5) + (tmp1-2)*4);
          Serial.println(tmp);
        }
      }
    }
    digitalWrite(i,HIGH);
  }
}
```

Практическая часть

Упражнение 1

1. Цифровой секундомер. Собрать схему с четырёхразрядным индикатором. Добавить в схему кнопку. При нажатии на кнопку, секундомер должен запускать отсчет. При повторном нажатии — останавливать. Дополнительно, к секундомеру можно добавить дробную часть, отображаемую в младшем разряде индикатора через точку.
2. Цифровой вольтметр для напряжений от 0 до 10 В. Собрать схему с четырёхразрядным индикатором. Добавить в схему делитель напряжения из двух резисторов на 10 кОм, подключенный к аналоговому входу Arduino. Написать программу, которая будет каждые 100 мс считывать значение на аналоговом входе, переводить его в напряжение и выводить на индикатор.

Упражнение 2

3. Консольный калькулятор. Собрать схему и написать программу консольного калькулятора, позволяющего выполнять следующие операции: умножение, деление, сложение и вычитание двух чисел. Вывод осуществлять в последовательный порт.
4. Кодовый замок. Собрать схему с клавиатурой и двумя светодиодами: первый светодиод отображает совпадение кодов, а второй – процесс программирования замка. Для включения режима программирования замка предусмотреть кодовую последовательность.