Практическое занятие №
Тема: «Семисегментный индикатор и сдвиговый регистр 74HC595N»

Цель работы: приобрести практические навыки по подключению и программированию Семисегментного индикатора и сдвигового регистра 74HC595N на платформе Arduino.

Последовательность выполнения работы:

– Собрать схемы на макетной плате, иначе при отсутствии набора Arduino в web-приложениях (https://wokwi.com/projects/new/arduino-uno или https://www.tinkercad.com/) для приведенных примеров.

– Запрограммировать микроконтроллер согласно заданию в примере.

– Выполнить задание для самостоятельной работы.

Содержание отчета:

– Название практического занятия, его цель.

– Фото или скриншоты собранной схемы.

– Написанный программный код вставить текстом, Courier New, 12 кегль, одинарный отступ без абзацев.

– Вывод о проделанной работе.

– Файл Fritzing с принципиальной и монтажной схемой.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

7-сегментный цифровой LED индикатор

7-сегментный цифровой LED индикатор – это индикатор, состоящий из семи светодиодов, установленных в форме цифры 8. Зажигая или выключая соответствующие LED-сегменты, можно отображать цифры от нуля до девяти, а также некоторые буквы.

Расположение и электросхема сегментов LED индикатора представлена ниже:

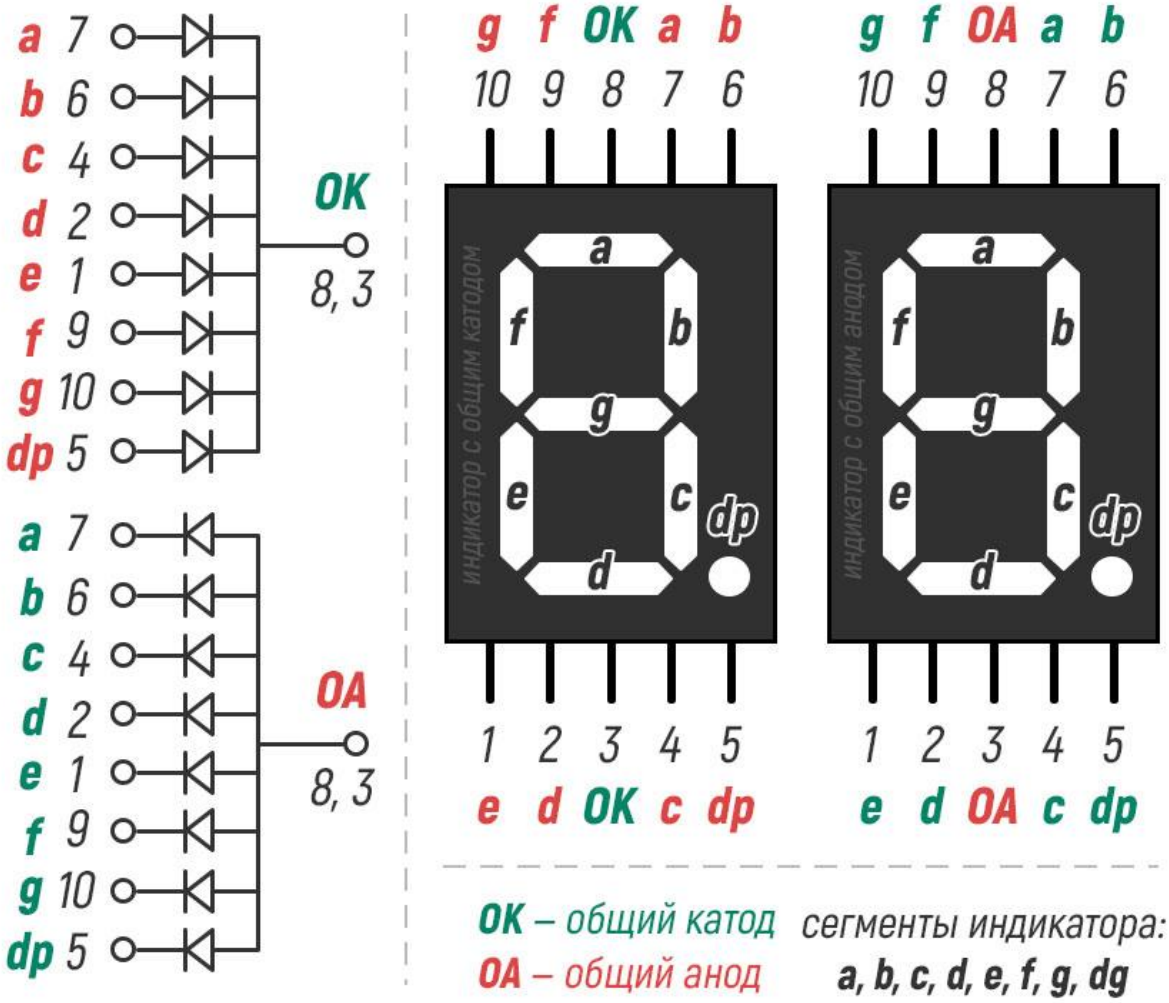

Рисунок 1 – 7-сегментный цифровой LED индикатор

Восьмиразрядный сдвиговый регистр

74HC595N – восьмиразрядный сдвиговый регистр с последовательным вводом, последовательным или параллельным выводом информации, с триггером-защелкой и тремя состояниями на выходе. Самое распространенное применение данного регистра – экономия выходов микроконтроллера. Данный сдвиговый регистр позволяет управлять напряжением на своих восьми выходах, заняв всего три выхода микроконтроллера. Таким образом количество рабочих выводов увеличивается на пять.

Кроме того, регистры 74HC595 можно подключать каскадом один за другим (через пин **Q7'**), и таким образом из всё тех же 3 входящих линий получать 16, 24, 32 и т.д. цифровых выходов.

74HC595N имеет следующие входы:

– **[10] MR** — сброс регистра, при подаче логического нуля на MR и единицы на STCP переводит все выходы в состояние логического нуля;

– **[11] SH_CP** — вход для тактовых импульсов;

– **[12] ST_CP** — линия прерываний;

– **[13] OE** — вход, переводящий выходы из высокоимпедансного состояния в рабочее;

– **[14] DS** — вход данных;

– **[8] GND** — Ground. Земля

– **[16] VCC** — Питание +5 В.

Рисунок 2 – Восьмиразрядный сдвиговый регистр

ЗАДАНИЯ

Задание 1: Управление 7-сегментным цифровым LED индикатором

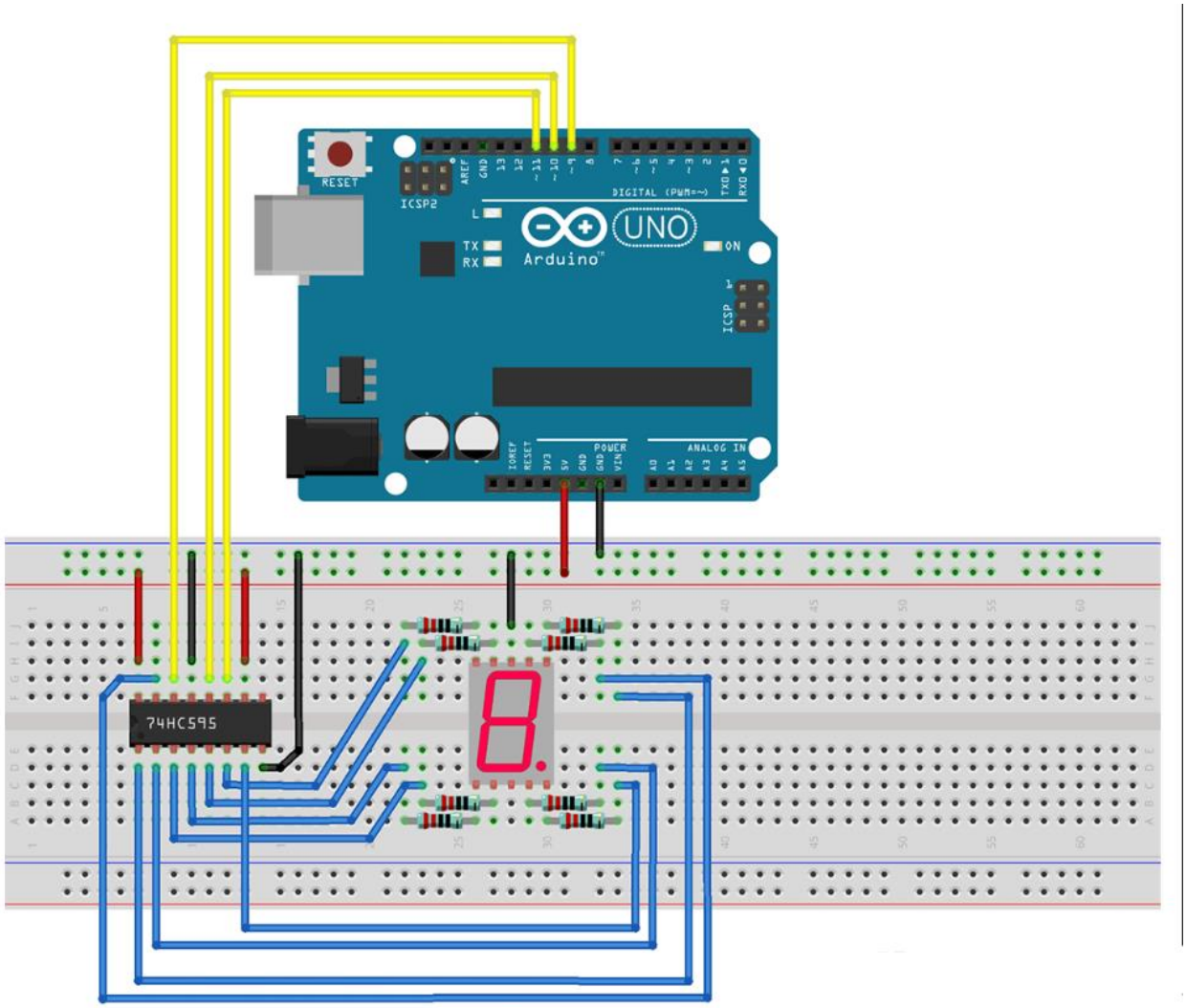

Рисунок 3 – Схема подключения

```
#define CLOCK 11  //SH_CP
#define DATA 9    //DS
#define LATCH 10  //ST_CP

void setup() {
 //настраиваем контакты на выход
 pinMode(CLOCK, OUTPUT);
 pinMode(DATA, OUTPUT);
 pinMode(LATCH, OUTPUT);

 //отключаем LATCH (чтобы регистр не ждал данных)
 digitalWrite(LATCH, HIGH);
}
```

```
void loop() {
 //включаем LATCH (Начинаем общение)
 digitalWrite(LATCH, LOW);

 //посылаем данные функцией shiftOut
 /*
Символ | байт
   .   | 0b00000001
   0   | 0b00000000
   1   | 0b01100000
   2   | 0b11011010
   3   | 0b11110010
   4   | 0b01100110
   5   | 0b10110110
   6   | 0b10111110
   7   | 0b11100000
   8   | 0b11111110
   9   | 0b11110110
 */
 shiftOut(DATA, CLOCK, LSBFIRST, 0b10110110);

 //включаем LATCH (Начинаем общение)
 digitalWrite(LATCH, HIGH);
}
```

Задание 2: Таймер с кнопочным управлением и 7-сегментным цифровым LED индикатором

Для данного задания требуется добавить 3 кнопки к пинам 2,3,4.

```
#define CLOCK 11  // SH_CP
#define DATA 9    // DS
#define LATCH 10  // ST_CP

// Пины кнопок
#define BTN_UP 2
#define BTN_DOWN 3
#define BTN_START 4

// Состояния кнопок
bool btnUpState = false;
bool btnDownState = false;
bool btnStartState = false;
bool lastBtnStartState = false;

// Таймер и состояния
unsigned long previousMillis = 0;
const long interval = 1000;
```

```
int currentTime = 0;
bool isCounting = false;
bool isPaused = false;

// Коды символов для семисегментного индикатора (0-9)
const byte digits[10] = {
  0b11111100, // 0
  0b01100000, // 1
  0b11011010, // 2
  0b11110010, // 3
  0b01100110, // 4
  0b10110110, // 5
  0b10111110, // 6
  0b11100000, // 7
  0b11111110, // 8
  0b11110110  // 9
};

void setup() {
  pinMode(CLOCK, OUTPUT);
  pinMode(DATA, OUTPUT);
  pinMode(LATCH, OUTPUT);

  pinMode(BTN_UP, INPUT_PULLUP);
  pinMode(BTN_DOWN, INPUT_PULLUP);
  pinMode(BTN_START, INPUT_PULLUP);

  digitalWrite(LATCH, HIGH);
  displayNumber(currentTime);
}

void loop() {
  // Обработка кнопок
  handleButtons();

  // Логика таймера
  if (isCounting && !isPaused) {
    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;
      if (currentTime > 0) {
        currentTime--;
        displayNumber(currentTime);
      } else {
        isCounting = false;
        isPaused = false;
      }
    }
  }
}
```

```
 void handleButtons() {
   // Кнопка увеличения
   if (digitalRead(BTN_UP) == LOW) {
     if (!btnUpState && !isCounting) {
       btnUpState = true;
       if (currentTime < 9) {
         currentTime++;
         displayNumber(currentTime);
       }
       delay(200);
     }
   } else {
     btnUpState = false;
   }

   // Кнопка уменьшения
   if (digitalRead(BTN_DOWN) == LOW) {
     if (!btnDownState && !isCounting) {
       btnDownState = true;
       if (currentTime > 0) {
         currentTime--;
         displayNumber(currentTime);
       }
       delay(200);
     }
   } else {
     btnDownState = false;
   }


  // Кнопка старта/паузы
  bool currentStartState = digitalRead(BTN_START);
  if (currentStartState == LOW && lastBtnStartState == HIGH)
{
    if (!isCounting) {
      // Запуск таймера
      isCounting = true;
      isPaused = false;
      previousMillis = millis();
    } else {
      // Пауза/возобновление
      isPaused = !isPaused;
      if (!isPaused) {
        previousMillis = millis();
      }
    }
    delay(200);
  }
  lastBtnStartState = currentStartState;
}
```

```
void displayNumber(int number) {
  digitalWrite(LATCH, LOW);
  shiftOut(DATA, CLOCK, LSBFIRST, digits[number]);
  digitalWrite(LATCH, HIGH);
}
```