Практическое занятие №
Тема: Подключение и программирование датчика шума

Цель работы: приобрести практические навыки по подключению и программированию датчика шума 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

Датчик звука KY-037

Датчик звука KY-037 предназначен для обнаружения звука и определения его порогового значения. Чувствительный микрофон, встроенный компаратор напряжения, аналоговый и цифровой выходы делают этот модуль привлекательным для применения в системах «Умный дом» и робототехнике. Порог срабатывания компаратора регулируется потенциометром.

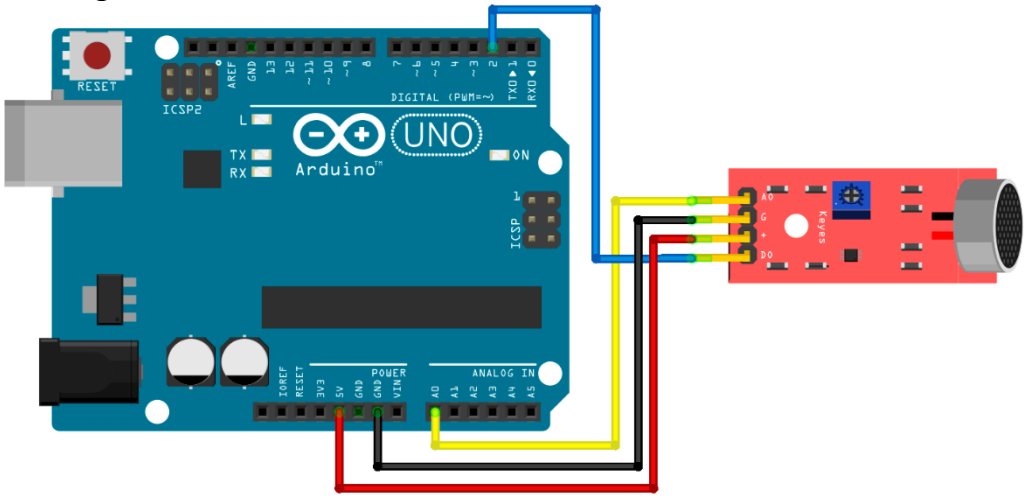

Рисунок 1 – 7-сегментный цифровой LED индикатор

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

!!! ДЛЯ ВСЕХ ЗАДАНИЙ СДЕЛАТЬ МОНТАЖНУЮ СХЕМУ !!!

Задание 1: Вывод уровня звука в дБ в монитор порта

```
const int analogPin = A0;  // Аналоговый пин датчика

void setup() {
  Serial.begin(9600);     // Инициализация последовательного
соединения
}

void loop() {
  int analogValue = analogRead(analogPin);         // Чтение
аналогового значения (0-1023)
  float voltage = analogValue * (5.0 / 1023.0);     //
Преобразование в напряжение (0-5В)
  float dB = 20 * log10(voltage / 0.00631);          //
Преобразование в дБ (формула для популярного датчика)

  Serial.print("Аналоговое значение: ");
  Serial.print(analogValue);
  Serial.print(" | Напряжение: ");
  Serial.print(voltage);
  Serial.print(" | dB: ");
  Serial.println(dB);

  delay(100);  // Задержка для стабильности чтений значений
}
```

Задание 1: Светодиодная индикация с тремя уровнями громкости

```
const int analogPin = A0;  // Аналоговый пин датчика
const int greenLed = 3;    // Зеленый светодиод
const int yellowLed = 5;   // Желтый светодиод
const int redLed = 6;      // Красный светодиод

// Пороговые значения (настройте под вашу среду)
const int lowThreshold = 300;   // Нижний порог для желтого
const int highThreshold = 700;  // Нижний порог для красного

void setup() {
  pinMode(greenLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(redLed, OUTPUT);
}

void loop() {
  int analogValue = analogRead(analogPin);

  // Зеленая зона (0 - lowThreshold)
```

```
  int greenBrightness = map(analogValue, 0, lowThreshold, 0,
255);
  greenBrightness = constrain(greenBrightness, 0, 255);
  analogWrite(greenLed, greenBrightness);

  // Желтая зона (lowThreshold - highThreshold)
  if (analogValue > lowThreshold) {
    int yellowBrightness = map(analogValue, lowThreshold,
highThreshold, 0, 255);
    yellowBrightness = constrain(yellowBrightness, 0, 255);
    analogWrite(yellowLed, yellowBrightness);
  } else {
    analogWrite(yellowLed, 0);
  }

  // Красная зона (highThreshold - 1023)
  if (analogValue > highThreshold) {
    int redBrightness = map(analogValue, highThreshold, 1023,
0, 255);
    redBrightness = constrain(redBrightness, 0, 255);
    analogWrite(redLed, redBrightness);
  } else {
    analogWrite(redLed, 0);
  }

  delay(10);
}
```

Задание 3: Светодиодная шкала на сдвиговом регистре 74HC595

Подключение 74HC595:
DS (Data) → 11
ST_CP (Latch) → 12
SH_CP (Clock) → 13
VCC → 5V
GND → GND

Светодиоды через резисторы (220 Ом) к выходам Q0-Q7

```
const int analogPin = A0;  // Аналоговый пин датчика
const int dataPin = 11;    // DS (Data)
const int latchPin = 12;   // ST_CP (Latch)
const int clockPin = 13;   // SH_CP (Clock)

void setup() {
  pinMode(dataPin, OUTPUT);
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop() {
```

```
  int analogValue = analogRead(analogPin);
  int level = map(analogValue, 0, 1023, 0, 8);  // Преобразуем
в уровень (0-8)
  level = constrain(level, 0, 8);

  // Формируем битовую маску для светодиодов
  byte pattern = 0;
  for (int i = 0; i < level; i++) {
    pattern |= (1 << i);
  }

  // Отправляем данные в регистр
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, pattern);
  digitalWrite(latchPin, HIGH);

  delay(10);
}
```