Датчик звука (Trema-модуль v2.0)

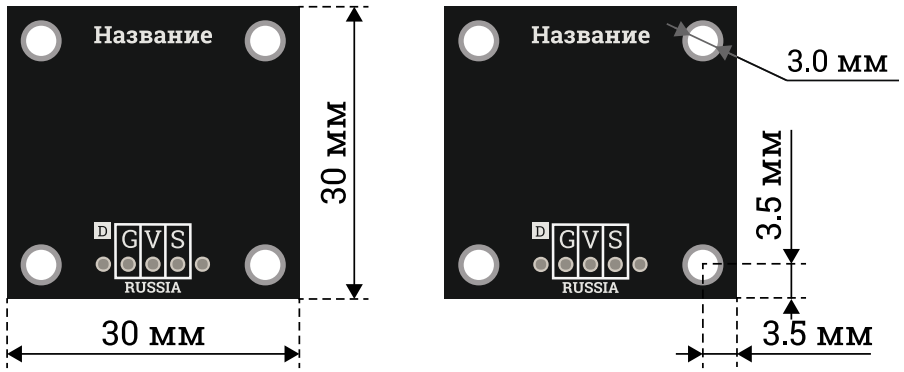
Общие сведения:

Trema-модуль Датчик звука - позволяет получить аналоговое значение, соответствующее уровню громкости звука. Модуль обладает высокой чувствительностью. Сигнал на выходе модуля не повторяет форму звукового сигнала, а соответствует уровню его громкости в любой промежуток времени.

Спецификация:

- Напряжение питания: 5 В постоянного тока.
- Потребляемый ток: 3,3 … 3,7 мА (зависит от уровня громкости).
- Чувствительность: -40 дБ ± 2 дБ.
- Частотный диапазон: 35 Гц … 10 кГц.
- Габариты: 30х30 мм.

Все модули линейки "Trema" выполнены в одном формате



Подключение:

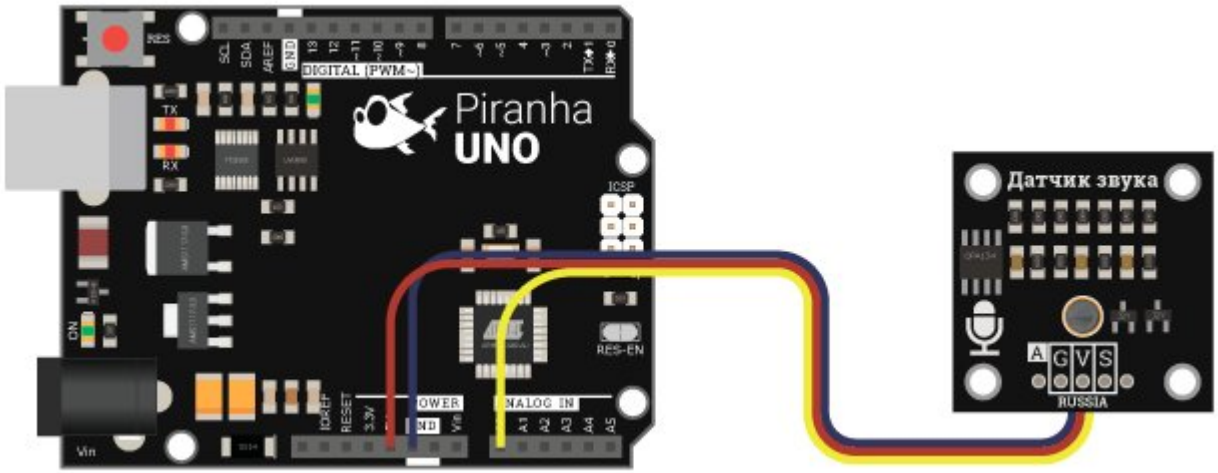
Trema датчик звука является аналоговым модулем, а значит его выход «S» (Signal) подключается к любому аналоговому входу Arduino.

Схема установки модуля при его подключении через Trema Set Shield.

Модуль удобно подключать 3 способами, в зависимости от ситуации:

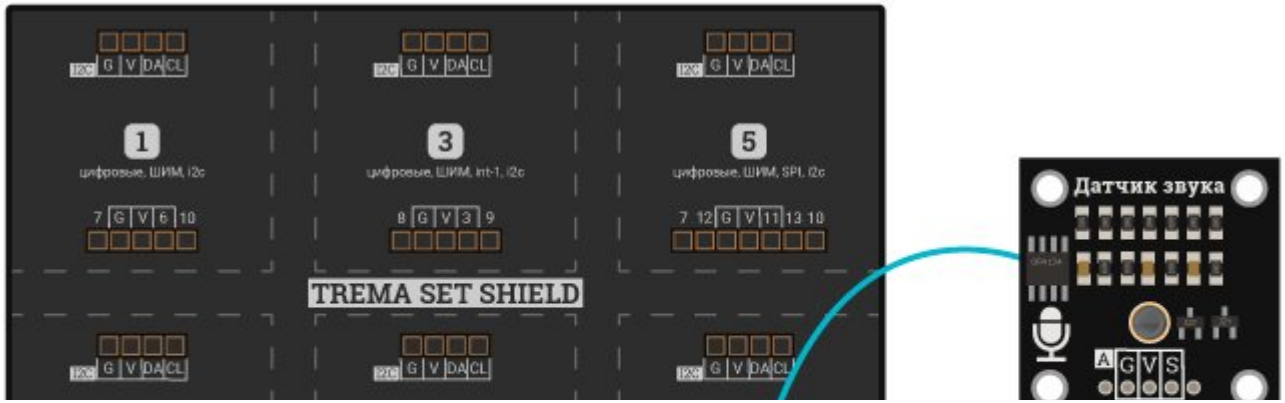
Способ - 1 : Используя проводной шлейф и Piranha UNO

Используя провода «Папа — Мама», подключаем напрямую к контроллеру Piranha UNO



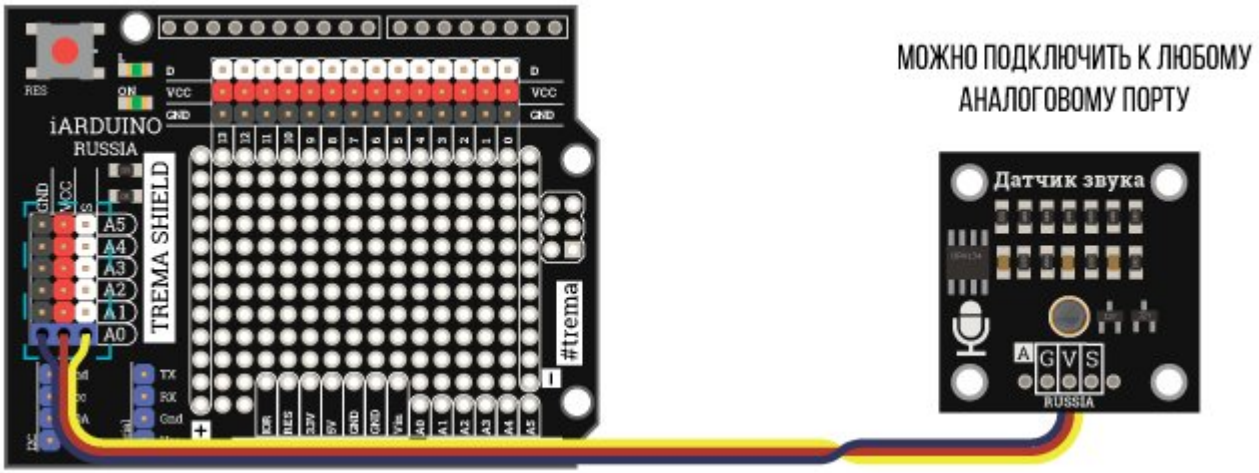
Способ - 2 : Используя Trema Set Shield

Модуль можно подключить к любому аналоговому входу Trema Set Shield.



Способ - 3 : Используя проводной шлейф и Shield

Используя 3-х проводной шлейф, к Trema Shield, Trema-Power Shield, Motor Shield, Trema Shield NANO и тд.



Питание:

Входное напряжение питания 5 В постоянного тока, подаётся на выводы «V» (Vcc) и «G» (GND) модуля.

Подробнее о модуле:

Звук это механические колебания распространяемые в твёрдых, жидких и газообразных средах. Микрофон Trema датчика звука преобразует механические колебания воздушной среды в электрические колебания и усиливает их, так как имеет встроенный предварительный усилитель. Полученный сигнал усиливается основным усилителем модуля построенным на чипе OPA134. В следующем блоке модуля из полученного сигнала удаляется постоянная составляющая и срезаются отрицательные полупериоды, после чего он опять усиливается. Далее сигнал поступает на сглаживающий фильтр и через согласующий блок поступает на выход модуля.

Модуль Благодаря наличию нескольких каскадов усиления обладает высокой чувствительностью. Сигнал на выходе модуля соответствует уровню громкости звука в любой промежуток времени, это значит что для измерения уровня громкости достаточно однократного чтения показаний датчика.

Примеры:

Управление светодиодами по хлопку.

```
const uint8_t  pinSensor   = A0;                        // Номер вывода к которому подключён датчик звука (можно изменить на люб
const uint16_t varVolume   = 400;                       // Минимальный уровень громкости (значение от 0 до 1023). Чем ниже значе
const uint16_t varDuration = 2000;                      // Время, которое отводится на хлопки (в миллисекундах), за это время Вы
const uint8_t  pinLED[3]  = {9, 6, 3};                  // Номера выводов к которым подключены светодиоды (первый светодиод подк
const uint8_t  varLED[3]  = {2, 3, 4};                  // Количество хлопков для реакции светодиодов (первый светодиод реагируе
bool           flgLED[3]  = {0, 0, 0};                  // Флаги состояния светодиодов (1-вкл / 0-выкл). Установленные в данной
uint8_t        varSum;                                  // Переменная для подсчёта количества хлопков за время varDuration.
uint32_t       varTimeOut;                              // Переменная для хранения времени завершения сессии хлопков.

void setup() {
  for (uint8_t i = 0; i < sizeof(pinLED); i++) {        // Проходим по всем выводам к которым подключены светодиоды ...
    pinMode      (pinLED[i], OUTPUT);                   // Конфигурируем вывод очередного светодиода как выход (OUTPUT).
    digitalWrite (pinLED[i], flgLED[i]);                // Устанавливаем на этом выводе состояние flgLED.
  }
}
void loop() {
  //Если зафиксирован хлопок:
  if (analogRead(pinSensor) > varVolume) {              // Если зафиксирован уровень звука выше значения varVolume, то ...
    //Считаем количество хлопков за время varDuration:
    varSum = 0;                                         // Сбрасываем счетчик количества хлопков.
    varTimeOut = millis() + varDuration;                // Определяем время завершения текущей сессии хлопков (текущее время + v
    while (varTimeOut > millis()) {                     // Уходим в цикл пока не наступит время завершения текущей сессии хлопко
      if (analogRead(pinSensor) > varVolume) {          // Если зафиксирован уровень звука выше значения varVolume, то ...
        while (analogRead(pinSensor) > varVolume) {     // Уходим в цикл ожидания завершения текущего хлопка.
```

```
          delay(50);                                  // Задержка delay(50) подавляет дребезг начала хлопка.
        }
        varSum++;                                     // Учитываем этот хлопок увеличивая значение varSum.
        delay(50);                                    // Задержка delay(50) подавляет дребезг окончания хлопка.
      }
    }
    // Время varDuration вышло, количество хлопков подсчитано и хранится в varSum, выполняем действия:
    for (uint8_t i = 0; i < sizeof(varLED); i++) {    // Проходим по всем лампам ...
      if (varSum == varLED[i]) {                      // Если количество хлопков varSum совпало со значением varLED одного из
        flgLED[i] = ! flgLED[i];                      // Меняем состояние флага flgLED для этого светодиода.
        digitalWrite(pinLED[i], flgLED[i]);           // Устанавливаем логический уровень на выводе pinLED в соответствии со з
      }
    }
  }
  //  В этом месте можно написать свой код ...       // Этот код будет выполняться в то время, пока не фиксируются хлопки.
}
```

Применение:

- Сигнализации;
- Системы, управляемые звуком (свет, двери и т.д.);