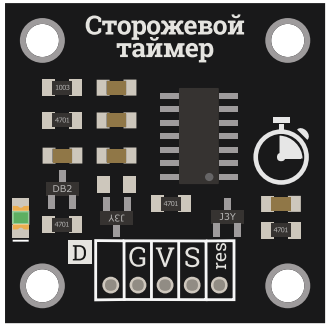
Сторожевой таймер | WatchDog Timer (Trema-модуль v2.0)



Общие сведения:

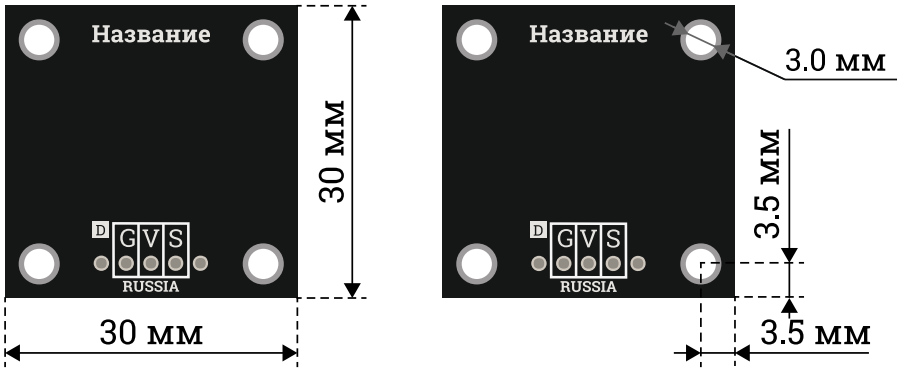
Trema-модуль Сторожевой таймер - это модуль позволяющий контролировать «зависание» Arduino. Если Arduino «зависнет» то Trema-модуль её перезагрузит как при нажатии кнопки «reset».

Сторожевые таймеры также называют WDT (WatchDog Timer - сторожевой пёс).

Спецификация:

- Входное напряжение: 5 В (постоянного тока)
- Потребляемый ток:
- Длительность импульса на входе S (Signal):
- Тайм-аут сторожевого таймера: 40 с. ±5 с.
- Длительность импульса на выходе RES (reset):
- Рабочая температура:
- Габариты: 30х30 мм.

Все модули линейки "Trema" выполнены в одном формате

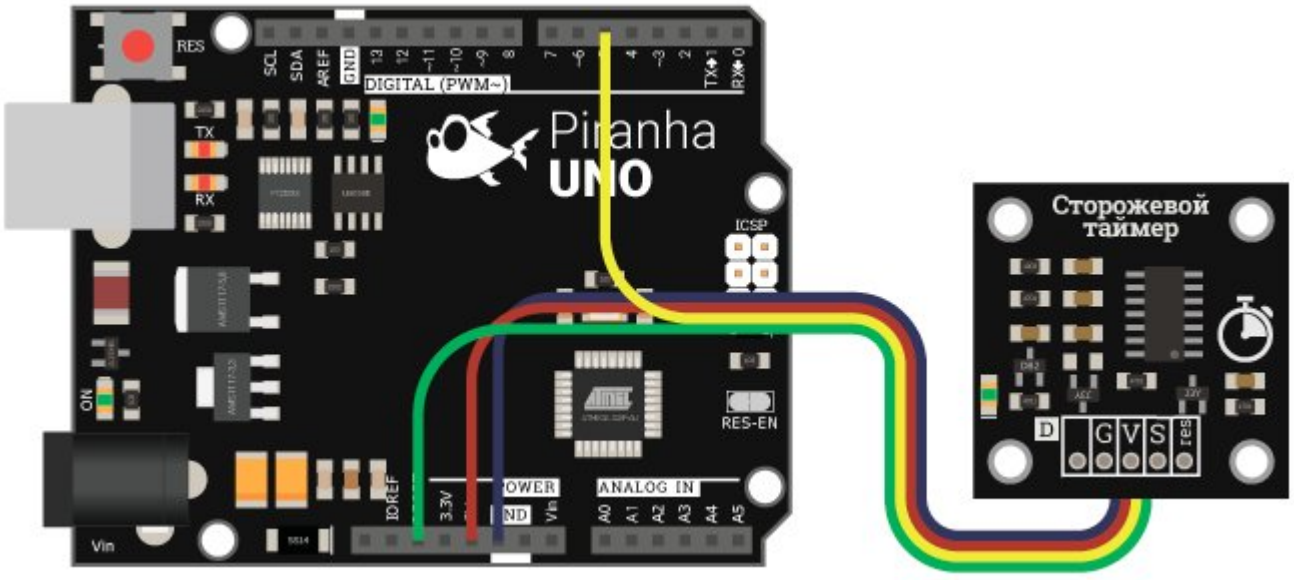


Подключение:

- Вход модуля S (Signal) подключается к любому выводу Arduino (нужен для сброса таймера).
  Если к Arduino в Вашем устройстве подключён модуль получающий данные от Arduino, не реже тайм-аута сторожевого таймера, то его вход S (Signal) можно подключить к Arduino вместе со входом этого модуля. Например, если Вы выводите данные в LCD дисплей по шине I2C, то вход S (Signal) сторожевого таймера можно подключить к выводу SDA или SCL этой шины. При таком подключении отпадает необходимость в написании кода для сброса таймера и не используется отдельный вывод для его работы.
- Выход модуля RES (Reset) подключается к выводу RESET Arduino (нужен для перезагрузки Arduino).

Используя проводной шлейф и Piranha UNO

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



Питание:

- Напряжение питания 5 В постоянного тока подаётся на выводы V (Vcc) и G (GND) модуля.

Подробнее о модуле:

В программе Arduino нужно прописать код который будет сбрасывать Trema-модуль сторожевой таймер, подавая импульсы на его вход S (Signal). Пока таймер постоянно сбрасывается, программа (устройство, Arduino) работает нормально. Если сторожевой таймер перестал получать импульсы сброса (пауза между импульсами превысила Тайм-аут таймера), то он «решает» что Arduino «зависла» и формирует импульс логического «0» на своём выходе RES (Reset) который должен быть подключён к выводу RESET Arduino.

На вход S (Signal) сторожевого таймера можно подавать любые импульсы (положительные или отрицательные). Импульс перезагрузки на выходе RES (Reset) сторожевого таймера формируется при паузе между импульсами на входе S (Signal) больше чем Тайм-аут, не зависимо от логического уровня на входе S (Signal).

Trema-модуль сторожевой таймер аппаратно сбрасывается (вне зависимости от состояний на его входе) при нажатии на кнопку Arduino RESET и при старте загрузки скетчей. Следовательно, Trema сторожевой таймер не требуется отключать от Arduino во время загрузки скетчей, они успеют загрузиться и выполнить сброс таймера до формирования импульса перезагрузки на его выходе.

Примеры:

Код для «зависания» Arduino:

```
const uint8_t pinVD = LED_BUILTIN;                  // Определяем константу с указанием № вывода Arduino к которому подключён
const uint8_t pinWDT = 5;                           // Определяем константу с указанием № вывода Arduino к которому подключён
                                                    //
void setup(void){                                   //
    pinMode     (pinVD,  OUTPUT);                   // Конфигурируем вывод подключённый к светодиоду как выход
    pinMode     (pinWDT, OUTPUT);                   // Конфигурируем вывод подключённый к сторожевому таймеру как выход
    digitalWrite(pinVD,  LOW);                      // Устанавливаем 0 на входе светодиода (выключаем светодиод)
    digitalWrite(pinWDT, LOW);                      // Устанавливаем 0 на входе таймера (далее будем подавать положительные им
}                                                   //
                                                    //
void loop(void){                                    //
    if(millis()%1000<10){                           // Выполняем код оператора if каждые первые 10 мс новой секунды
        digitalWrite(pinWDT, HIGH);                 // Формируем импульс на входе сторожевого таймера
        digitalWrite(pinWDT, LOW);                  // Завершаем импульс на входе сторожевого таймера
        digitalWrite(pinVD, digitalRead(pinVD)^1);  // Меняем состояние светодиода (если он был включён, то выключаем и наобор
```

```
        delay(11);                                  // Ждём 11 мс чтоб не выполнять код оператора if более одного раза в секун
        memset(malloc(100), 5, 10);                 // Пытаемся «зависнуть»
    }
}
```

- В начале кода (до функции setup) определяются выводы к которым подключён светодиод и вход Trema сторожевого таймера.
- В коде setup() определённые ранее выводы переводятся в режим выхода и на них устанавливается уровень логического «0» (LOW).
- В коде loop() все функции находятся в теле оператора if и выполняются только если его условие верно.
  - Условие (millis()%1000<10) будет верно в течении 10 мс в начале каждой секунды (1000 мс) с начала старта скетча, то есть в промежутки времени: 0…10 мс, 1000…1010 мс, 2000…2010 мс, 3000…3010 мс и т.д.
  - Первые две функции digitalWrite() формируют положительный импульс на входе Trema сторожевого таймера (устанавливая высокий логический уровень и сразу сбрасывая его в низкий логический уровень)
  - Следующая строка digitalWrite(pinVD, digitalRead(pinVD)^1) считывает установленный ранее логический уровень вывода светодиода и меняет его на противоположный
  - Далее следует задержка delay(11) чтоб все функции в теле оператора if выполнялись только 1 раз в секунду. Если вход в тело оператора if произошел спустя 1001 мс после старта скетча, то после задержки в 11 мс будет 1012 мс, что не попадает под условие оператора if до начала следующей секунды.
  - И последняя строка memset(malloc(100), 5, 10) пытается «зависнуть», делается это так:
    - Функция malloc(100) выделяет область памяти ОЗУ размером в 100 байт, но так как мы не освобождаем эту область функцией free(), то память в «куче» ОЗУ скоро заполнится.
    - Функция memset(адрес, 5, 10) записывает значение 5 в 10 байт начиная с адреса возвращённого функцией malloc(100).
    - Как только память «кучи» ОЗУ заполнится, функция malloc(100) откажется выделять новую область памяти размером 100 байт и вернёт 0.
    - Но функция memset(адрес, 5, 10) воспримет этот 0 как адрес начиная с которого запишет десять пятёрок, что и приведёт к «зависанию» (вместо значения 5 можно записывать другие значения).
    - В качестве параметра функции malloc() вместо 100 можно указать любое другое число, чем оно будет больше, тем быстрее закончится память «кучи» ОЗУ и быстрее произойдёт «зависание».

В результате светодиод будет мигать несколько секунд пока не «зависнет» Arduino. Так как Arduino «висит», она не отправляет команды ни

светодиоду, ни сторожевому таймеру, следовательно, сторожевой таймер не сбрасывается. По истечении тайм-аута сторожевой таймер сформирует отрицательный импульс на выходе RES который подключён к выводу RESET Arduino, что приведёт к её перезагрузке. После перезагрузки светодиод опять начнёт мигать пока Arduino снова не «зависнет». Если исключить сторожевой таймер из схемы, то Arduino после «зависания» не перезагрузится, пока мы не нажмём на кнопку RESET или не отключим питание.

Примечание:

Так как в рассматриваемой схеме Arduino меняет состояние светодиода каждую секунду (пока она не «зависнет»), что ниже времени тайм-аута сторожевого таймера, то вход S (Signal) сторожевого таймера можно подключить вместе со светодиодом к выводу pinVD, исключив из скетча все строки в которых встречается константа pinWDT. Таким образом мы избавляемся от необходимости выделения отдельного вывода под сторожевой таймер, и от необходимости написания для него дополнительно кода.

Применение:

- В качестве альтернативы WDT интегрированного в Arduino, в отличии от которого, для работы Trema WDT не требуется прошивать новый bootloader (загрузчик). Или в случаях когда тайм-аут WDT должен превышать предельное время в 8 сек. для интегрированного в Arduino.
- Для контроля «зависаний» микроконтроллеров без WDT.
- Для контроля «зависаний» иных модулей подключаемых к Arduino, которым для возобновления работы достаточно вновь получить данные инициализации от Arduino.