Датчик освещенности, люксметр, FLASH-I2C (Trema-модуль)

Общие сведения:

Trema модуль - Датчик освещенности, люксметр, I2C-flash - является цифровым датчиком способным возвращать значение освещённости в люксах и коэффициент пульсаций света в процентах, а так же определять близость препятствий.

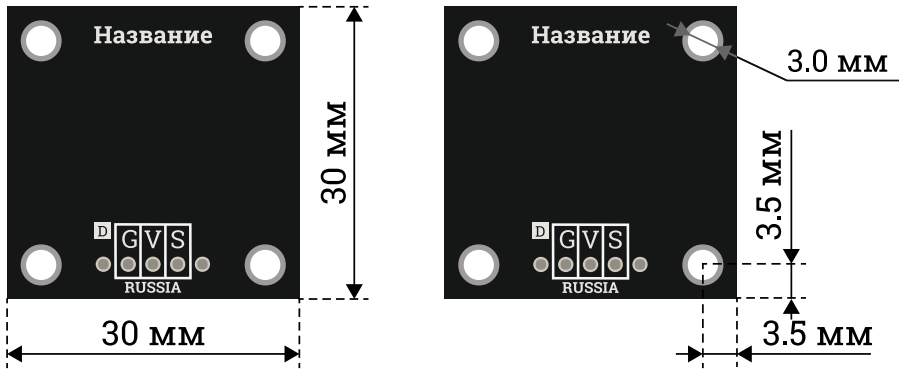
Модуль относится к серии «Flash», а значит к одной шине I2C можно подключить более 100 модулей, так как их адрес на шине I2C (по умолчанию 0x09), хранящийся в энергонезависимой памяти, можно менять программно.

Модуль можно использовать в любых проектах где требуется определять освещённость.

Спецификация:

- Напряжение питания: 3,3 В или 5 В (постоянного тока).
- Потребляемый ток: до 5 мА.
- Диапазон измерений освещённости: от 0 до 8191 лк.
- Диапазон измерений пульсаций света: от 0 до 100%.
- Диапазон обнаружения близости препятствий: от 0 до 1023.
- Интерфейс: I2C.
- Скорость шины I2C: 100 кбит/с.
- Адрес на шине I2C: устанавливается программно (по умолчанию 0x09).
- Уровень логической 1 на линиях шины I2C: 3,3 В (толерантны к 5 В).
- Рабочая температура: от -20 до +70 °C.
- Габариты: 30 x 30 мм.
- Вес: 4 г.

Все модули линейки "Trema" выполнены в одном формате



Подключение:

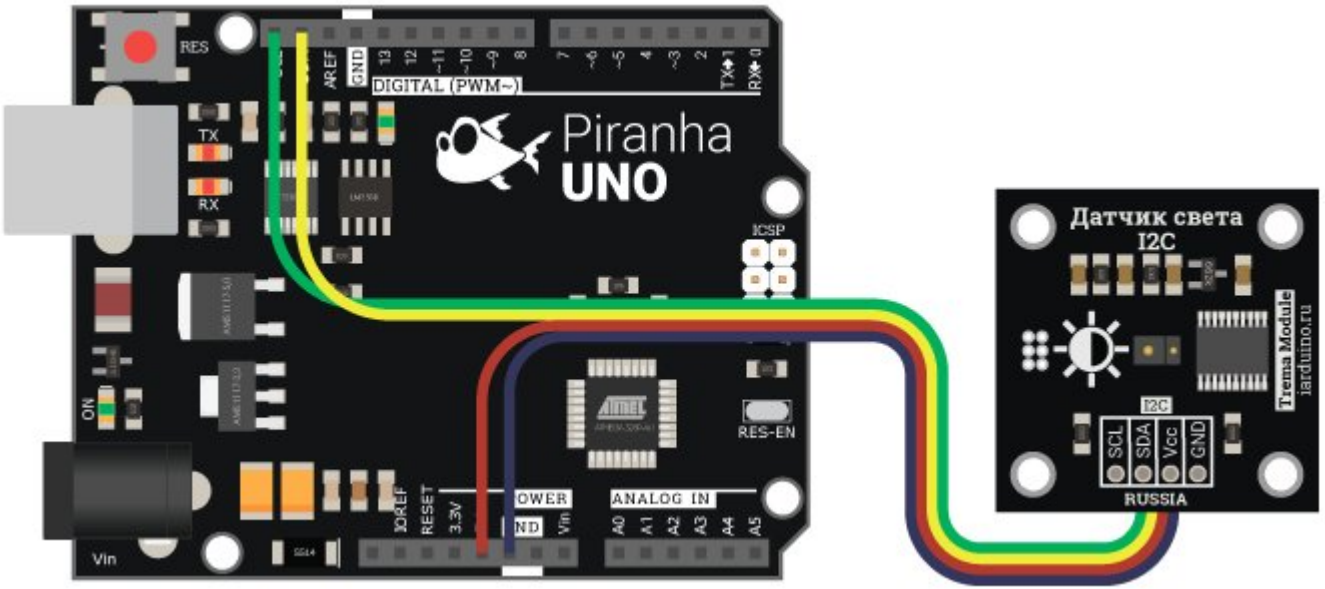
Модуль подключается к аппаратной или программной шине I2C Arduino. Для удобства подключения, предлагаем воспользоваться TremaShield.

> Перед подключением модуля ознакомьтесь с разделом "Смена адреса модуля на шине I2C" в данной статье.

Модуль удобно подключать 3 способами, в зависимости от ситуации:

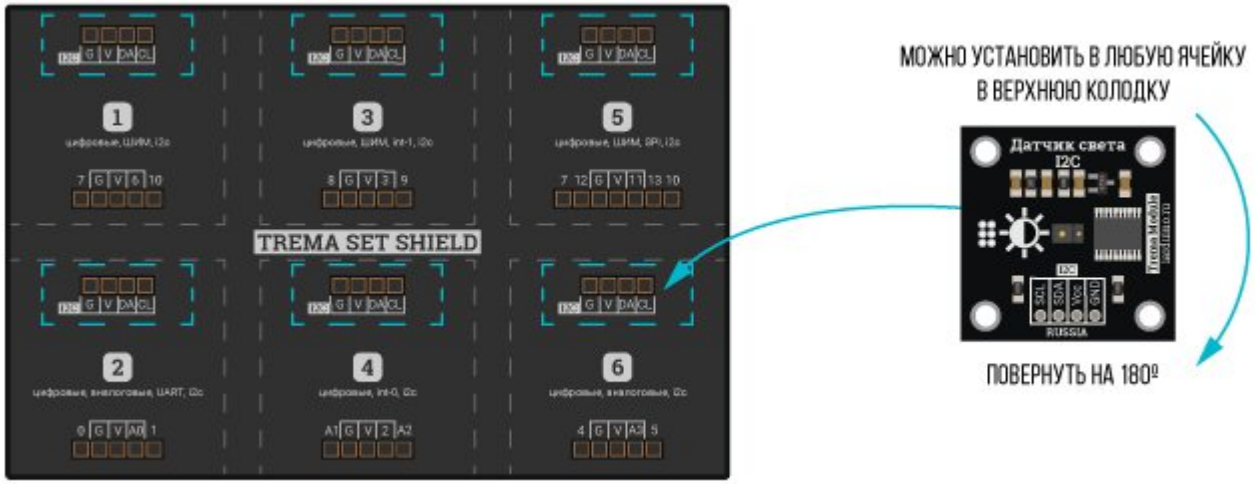
Способ - 1: Используя проводной шлейф и Piranha UNO

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



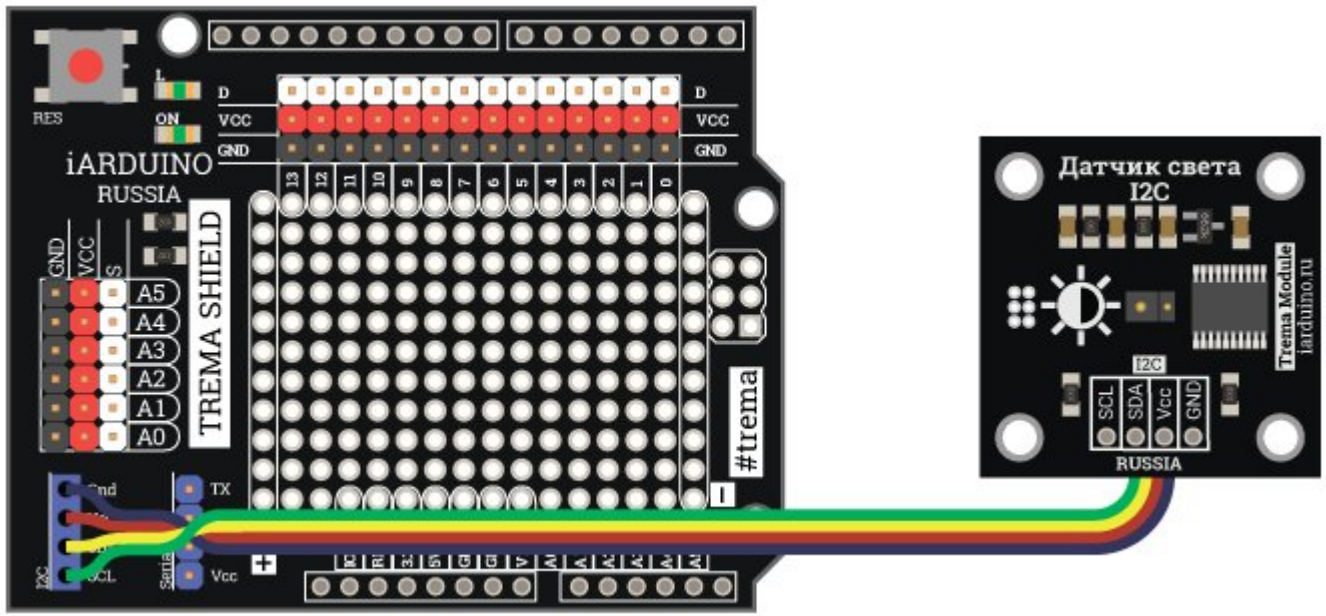
Способ - 2: Используя Trema Set Shield

Модуль можно подключить к любому из I2C входов Trema Set Shield.



Способ - 3: Используя проводной шлейф и Shield

Используя 4-х проводной шлейф, к Trema Shield, Trema-Power Shield, Motor Shield, Trema Shield NANO и тд.



Питание:

Входное напряжение питания модуля 3,3В или 5В постоянного тока (поддерживаются оба напряжения питания), подаётся на выводы Vcc и GND.

Подробнее о модуле:

Модуль построен на базе датчика APDS-9930, микроконтроллера STM32F030F4 и снабжен собственным стабилизатором напряжения. Модуль самостоятельно обрабатывает сигналы поступающие с его датчика, обрабатывает их и возвращает запрошенные результаты.

Модуль позволяет:

- Менять свой адрес на шине I2C.
- Получать уровень освещённости.
- Получать коэффициент пульсаций света.
- Получать близость препятствий.
- Задавать коэффициент сглаживания показаний освещённости и приближения.
- Реагировать на изменение освещённости с заданным порогом чувствительности.

Специально для работы с Trema модулем - Датчик освещенности, люксметр, I2C-flash , нами разработана библиотека iarduino_I2C_DSL которая позволяет реализовать все функции модуля.

Подробнее про установку библиотеки читайте в нашей инструкции.

Смена адреса модуля на шине I2C:

По умолчанию все модули FLASH-I2C имеют установленный адрес 0x09. Если вы планируете подключать более 1 модуля на шину I2C, необходимо изменить адреса модулей таким образом, чтобы каждый из них был уникальным. Более подробно о том, как изменить адрес, а также о многом другом, что касается работы FLASH-I2C модулей, вы можете прочесть в **этой статье**.

В первой строке скетча необходимо записать в переменную **newAddress** адрес, который будет присвоен модулю. После этого подключите

модуль к контроллеру и загрузите скетч. **Адрес может быть от 0x07 до 0x7F**.

```
uint8_t newAddress = 0x09;                          //   Назначаемый модулю адрес (0x07 < адрес < 0x7F).
                                                    //
#include <Wire.h>                                   //   Подключаем библиотеку для работы с аппаратной шиной I2C.
#include <iarduino_I2C_DSL.h>                       //   Подключаем библиотеку для работы с датчиком освещённости I2C-flash.
iarduino_I2C_DSL dsl;                               //   Объявляем объект enc для работы с функциями и методами библиотеки iardu
                                                    //   Если при объявлении объекта указать адрес, например, dsl(0xBB), то прим
void setup(){                                       //
    Serial.begin(9600);                             //
    if( dsl.begin() ){                              //   Инициируем работу с датчиком освещённости.
        Serial.print("Найден люксметр 0x");         //
        Serial.println( dsl.getAddress(), HEX );    //   Выводим текущий адрес модуля.
        if( dsl.changeAddress(newAddress) ){        //   Меняем адрес модуля на newAddress.
            Serial.print("Адрес изменён на 0x");    //
            Serial.println(dsl.getAddress(),HEX );//   Выводим текущий адрес модуля.
        }else{                                      //
            Serial.println("Адрес не изменён!");    //
        }                                           //
    }else{                                          //
        Serial.println("Люксметр не найден!");      //
    }                                               //
}                                                   //
                                                    //
void loop(){}                                       //
```

Примеры:

В данном разделе раскрыты примеры получения данных от модуля по шине I2C с использованием библиотеки iarduino_I2C_DSL. Сама библиотека содержит больше примеров, доступных из меню Arduino IDE: Файл / Примеры / iarduino I2C DSL (датчик освещённости - люксметр).

Чтение всех значений датчика:

```
// ПРИМЕР ЧТЕНИЯ ВСЕХ ЗНАЧЕНИЙ ДАТЧИКА:          // * Строки со звёздочкой являются необязательными.
                                                 //
#include <Wire.h>                                // * Подключаем библиотеку для работы с аппаратной шиной I2C.
#include <iarduino_I2C_DSL.h>                    //   Подключаем библиотеку для работы с датчиком освещённости I2C-flash (Dig
iarduino_I2C_DSL dsl;                            //   Объявляем объект dsl для работы с функциями и методами библиотеки iardu
                                                 //   Если при объявлении объекта указать адрес, например, dsl(0xBB), то прим
void setup(){                                    //
    delay(500);                                  // * Ждём завершение переходных процессов связанных с подачей питания.
    Serial.begin(9600);                          //
    while(!Serial){;}                            // * Ждём завершения инициализации шины UART.
    dsl.begin();                                 //   Инициируем работу с датчиком освещённости.
}                                                //
                                                 //
void loop(){                                     //
    Serial.print("Освещённость = ");             //
    Serial.print( dsl.getLux() );                //   Выводим текущую освещённость, от 0 до 8191 лк.
    Serial.print(" лк.\t");                      //
    Serial.print("Мерцание = ");                 //
    Serial.print( dsl.getPulsation() );          //   Выводим коэффициент пульсаций света, от 0 до 100%.
    Serial.print("%\t");                         //
    Serial.print("Близость = ");                 //
    Serial.print( dsl.getProximity() );          //   Выводим близость препятствий, от 0 (нет  препятствий) до 1023 (датчик п
    Serial.print(".\r\n");                       //
    delay(500);                                  // * Задержка позволяет медленнее заполнять монитор последовательного порта.
}                                                //
```

После загрузки данного примера, в мониторе последовательного порта будут появляться все данные, которые способен вернуть модуль: освещённость, коэффициент пульсаций и близость.

Чтение изменения освещённости:

```
// ПРИМЕР ЧТЕНИЯ ОСВЕЩЁННОСТИ ПРИ ЕЁ ИЗМЕНЕНИИ:

#include <Wire.h>
#include <iarduino_I2C_DSL.h>
iarduino_I2C_DSL dsl;

void setup(){
    delay(500);
    Serial.begin(9600);
    while(!Serial){;}
    dsl.begin();
    dsl.setLuxChange( 10 );
}

void loop(){
    if( dsl.getLuxChanged() ){
        Serial.print("Освещённость = ");
        Serial.print( dsl.getLux() );
        Serial.print(" лк.\r\n");
    }
}
```

```
// * Строки со звёздочкой являются необязательными.
//
// * Подключаем библиотеку для работы с аппаратной шиной I2C.
//   Подключаем библиотеку для работы с датчиком освещённости I2C-flash (Dig
//   Объявляем объект dsl для работы с функциями и методами библиотеки iardu
//   Если при объявлении объекта указать адрес, например, dsl(0xBB), то прим
//
// * Ждём завершение переходных процессов связанных с подачей питания.
//
// * Ждём завершения инициализации шины UART.
//   Инициируем работу с датчиком освещённости.
//   Указываем фиксировать изменение освещённости более чем на 10 лк.
//
//
//
//   Если освещённость изменилась более чем на значение указанное в функции
//
//   Выводим текущую освещённость, от 0 до 8191 лк.
//
//
//
```

После загрузки данного примера, в мониторе последовательного порта будут появляться показания текущей освещённости, но только если она меняется. Если Вам требуется получать освещённость вне зависимости от того изменилась она или нет, то воспользуйтесь предыдущим примером, выводя только показания освещённости.

Описание функций библиотеки:

В данном разделе описаны функции библиотеки iarduino_I2C_DSL для работы с Trema модулем - Датчик освещенности, люксметр, I2C-flash.

Данная библиотека может использовать как аппаратную, так и программную реализацию шины I2C. О том как выбрать тип шины I2C рассказано в статье Wiki - расширенные возможности библиотек iarduino для шины I2C.

Подключение библиотеки:

- Если адрес модуля известен (в примере используется адрес 0x09):

```
#include <iarduino_I2C_DSL.h>       // Подключаем библиотеку для работы с модулем.
iarduino_I2C_DSL dsl(0x09);         // Создаём объект dsl для работы с функциями и методами библиотеки iarduino_I2C_DSL, указы
```

- Если адрес модуля неизвестен (адрес будет найден автоматически):

```
#include <iarduino_I2C_DSL.h>       // Подключаем библиотеку для работы с модулем.
iarduino_I2C_DSL dsl;               // Создаём объект dsl для работы с функциями и методами библиотеки iarduino_I2C_DSL.
```

При создании объекта без указания адреса, на шине должен находиться только один модуль.

Функция begin();

- Назначение: Инициализация работы с модулем.
- Синтаксис: begin();
- Параметры: Нет.
- Возвращаемое значение: bool - результат инициализации (true или false).
- Примечание: По результату инициализации можно определить наличие модуля на шине.
- Пример:

```
if( dsl.begin() ){ Serial.print( "Модуль найден и инициирован!" ); }
else             { Serial.print( "Модуль не найден на шине I2C" ); }
```

Функция reset();

- Назначение: Перезагрузка модуля.
- Синтаксис: reset();
- Параметры: Нет.
- Возвращаемое значение: bool - результат перезагрузки (true или false).
- Пример:

```
if( dsl.reset() ){ Serial.print( "Модуль    перезагружен" ); }
else             { Serial.print( "Модуль не перезагружен" ); }
```

Функция changeAddress();

- Назначение: Смена адреса модуля на шине I2C.
- Синтаксис: changeAddress(АДРЕС);
- Параметры:
  - uint8_t АДРЕС - новый адрес модуля на шине I2C (целое число от 0x08 до 0x7E)
- Возвращаемое значение: bool - результат смены адреса (true или false).
- Примечание: Текущий адрес модуля можно узнать функцией getAddress().
- Пример:

```
if( dsl.changeAddress(0x12) ){ Serial.print( "Адрес модуля изменён на 0x12" ); }
else                         { Serial.print( "Не удалось изменить адрес"    ); }
```

Функция getAddress();

- Назначение: Запрос текущего адреса модуля на шине I2C.
- Синтаксис: getAddress();
- Параметры: Нет.
- Возвращаемое значение: uint8_t АДРЕС - текущий адрес модуля на шине I2C (от 0x08 до 0x7E)
- Примечание: Функция может понадобиться если адрес модуля не указан при создании объекта, а обнаружен библиотекой.
- Пример:

```
Serial.print( "Адрес модуля на шине I2C = 0x" );
Serial.println( dsl.getAddress(), HEX );
```

Функция getVersion();

- Назначение: Запрос версии прошивки модуля.
- Синтаксис: getVersion();
- Параметры: Нет
- Возвращаемое значение: uint8_t ВЕРСИЯ - номер версии прошивки от 0 до 255.
- Пример:

```
Serial.print( "Версия прошивки модуля " );
Serial.println( dsl.getVersion(), HEX );
```

Функция getLux();

- Назначение: Запрос уровня освещённости в люксах.
- Синтаксис: getLux();
- Параметры: Нет.
- Возвращаемое значение: uint16_t - уровень освещённости от 0 до 8191 лк.
- Примечание:
  - При увеличении реальной освещённости более 8191 лк, возвращаемые функцией значения будут резко снижаться.
- Пример:

```
Serial.print( "Освещённость = " );
Serial.print( dsl.getLux() );
Serial.println( "лк." );
```

Функция getPulsation();

- Назначение: Запрос коэффициента пульсаций света.
- Синтаксис: getPulsation();
- Параметры: Нет.
- Возвращаемое значение: uint8_t - коэффициент пульсаций света от 0 до 100%.
- Примечание:
  - Коэффициент пульсаций света определяется модулем в процентах, как разность максимальной и минимальной освещённости среди последних 20 измерений.
- Пример:

```
Serial.print( "Свет пульсирует на " );
Serial.print( dsl.getPulsation() );
Serial.println( "%." );
```

Функция getProximity();

- Назначение: Запрос уровня близости препятствий.
- Синтаксис: getProximity();
- Параметры: Нет.
- Возвращаемое значение: uint16_t - значение близости от 0 (нет препятствий) до 1023.
- Примечание:
  - Значение возвращаемое данной функцией позволяет определить степень близости препятствия, а не расстояние до него.
- Пример:

```
uint16_t i = dsl.getProximity();
if (i>1000) { Serial.println( "Модуль перекрыт"          ); }else
if (i>512 ) { Serial.println( "Препятствие очень близко" ); }else // примерно  1 ...  5 см.
if (i>100 ) { Serial.println( "Препятствие близко"       ); }else // примерно  5 ... 10 см.
if (i>0   ) { Serial.println( "Препятствие далеко"       ); }else // примерно 10 ... 20 см.
            { Serial.println( "Препятствий нет"          ); }
```

Функция getLuxChanged();

- Назначение: Запрос подтверждения изменения освещённости.
- Синтаксис: getLuxChanged();
- Параметр: Нет.
- Возвращаемое значение: bool - подтверждение изменения освещённости (true или false).
- Примечание:
  - Функция возвращает положительный результат `true`, если с момента последнего положительного результата, освещённость изменилась на значение указанное функцией `setLuxChange()`.

Функция setLuxChange();

- Назначение: Установка порога чувствительности фиксации изменения освещённости.
- Синтаксис: setLuxChange(ПОРОГ);
- Параметры: uint8_t ПОРОГ - значение от 1 до 255 лк.
- Возвращаемое значение: bool - результат применения новых настроек (true или false).
- Примечание:
  - Данная функция определяет, как сильно должна измениться освещённость, чтоб функция `getLuxChanged()` вернула `true`.
- Пример:

```
dsl.setLuxChanged(10);                  // Указываем реагировать на изменения в 10 лк.
void loop() {                           //
    if( dsl.getLuxChanged() ){          // Если освещённость изменилась, то ...
        Serial.print("Освещённость = "); //
        Serial.print( dsl.getLux()    ); // Выводим текущую освещённость.
        Serial.print(" лк.\r\n"       ); //
    }                                   //
}                                       //
```

Функция setAveraging();

- Назначение: Установка коэффициента усреднения показаний освещённости и близости.

- Синтаксис: setAveraging(УСРЕДНЕНИЕ);
- Параметр:
  - uint8_t УСРЕДНЕНИЕ - значение от 0 (не усреднять) до 255 (максимальное усреднение).
- Возвращаемое значение: bool - результат сохранения настроек (true или false).
- Примечание:
  - Чем выше значение усреднения, тем плавнее будут меняться показания освещённости и близости. Слишком высокое усреднение приведёт к большой инерционности показаний, а слишком маленькое усреднение приведёт к «скачкам» показаний. Значение по умолчанию 63 (25% от максимального значения).
- Пример:

```
dsl.setAveraging( 50 );
```