# ИК-приемник/передатчик, FLASH-I2C

## Общие сведения:

*Trema модуль ИК-приёмник/передатчик, Flash-I2C* является устройством для беспроводного обмена данными по ИК-каналу. Модуль позволяет передавать данные на устройства с управлением от ИК пультов ДУ, обмениваться данными с подобными себе модулями и получать данные от ИК пультов ДУ, и модулей линейки «Дорожное движение». Управление модулем ИК-приёмопередатчика осуществляется по шине I2C.

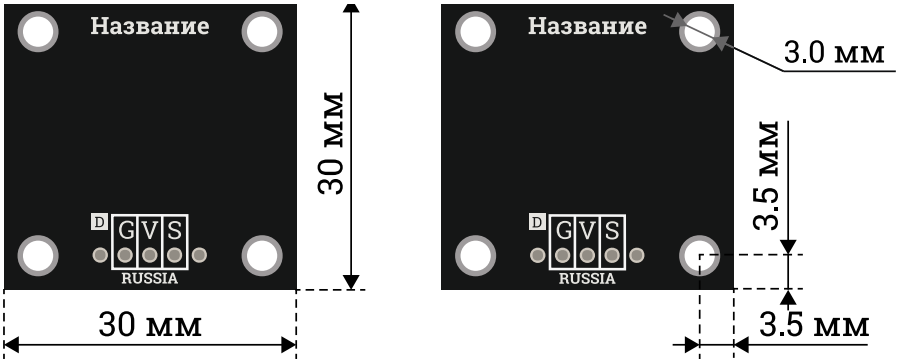
Модуль относится к серии «Flash», а значит к одной шине I2C можно подключить более 100 модулей, так как их адрес на шине I2C (по умолчанию 0x09), хранящийся в энергонезависимой памяти, можно менять программно.

Модуль можно использовать для создания универсального ИК пульта ДУ, управления роботами, машинами, станками бытовыми приборами, а так же модуль можно установить на устройства и управлять ими с ИК пультов ДУ. Модуль можно установить на машину трассы и принимать данные о состоянии светофоров и назначении знаков трассы.

## Спецификация:

- Напряжение питания: 3,3 В или 5 В, поддерживаются оба напряжения.
- Ток потребляемый модулем: до 15 мА (при передаче данных).
- Дальность ИК-связи передатчика: от 20 см до 4 м (регулируется вручную).
- Интерфейс: I2C.
- Скорость шины I2C: 100 кбит/с.
- Адрес на шине I2C: устанавливается программно (по умолчанию 0x09).
- Уровень логической 1 на линиях шины I2C: Vcc (толерантны к 5 В).
- Рабочая температура: от -20 до +70 °C.
- Габариты: 30 х 30 мм..
- Вес: 5 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.

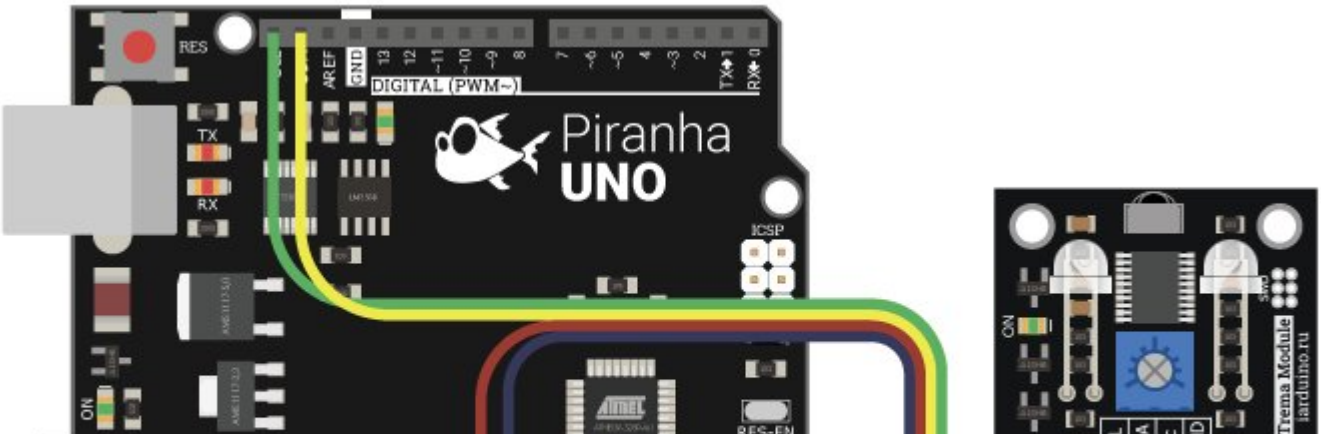
**По умолчанию все модули FLASH-I2C имеют установленный адрес 0x09.**
— Перед подключением 1 модуля к шине I2C **настоятельно рекомендуется** изменить адрес модуля.
— При подключении 2 и более FLASH-I2C модулей к шине необходимо **в обязательном порядке предварительно изменить адрес каждого модуля**, после чего уже подключать их к шине.

Модуль подключается по шине **I2C**, все выводы которой (GND, Vcc, SDA, SCL) размещены на одной колодке модуля.

- **SCL** - вход/выход линии тактирования шины I2C.
- **SDA** - вход/выход линии данных шины I2C.
- **Vcc** - вход питания 3,3 или 5 В.
- **GND** - общий вывод питания.

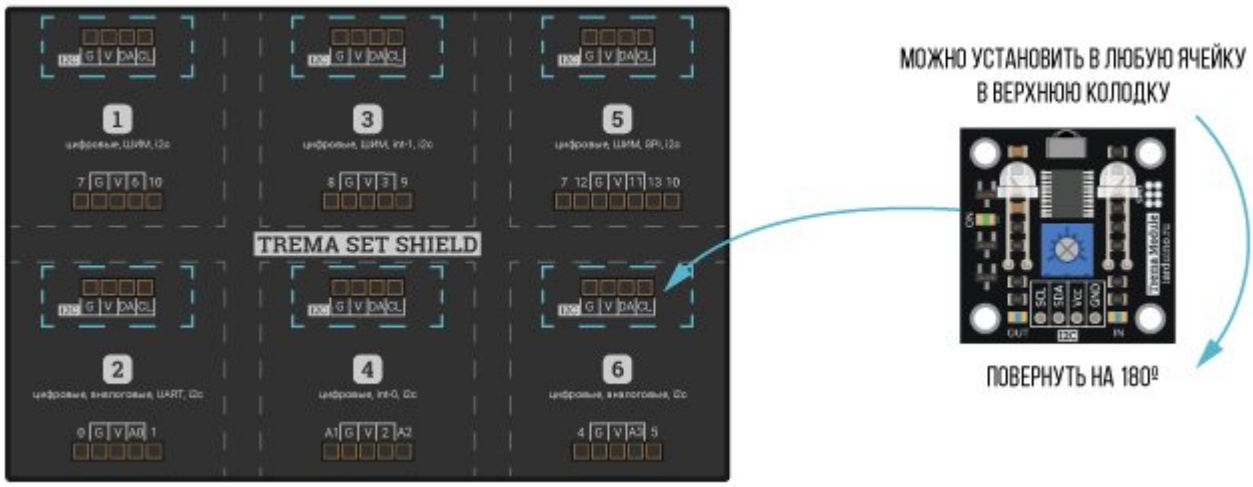
### Способ - 1: Используя проводной шлейф и Piranha UNO

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



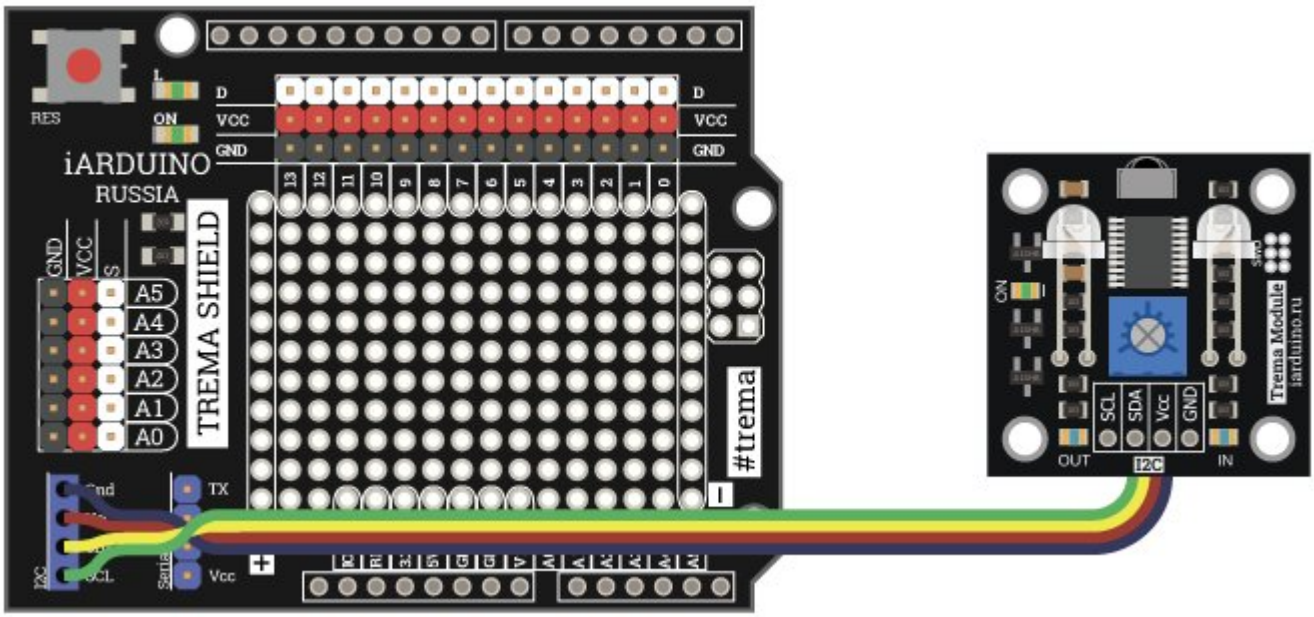

### Способ - 2: Используя Trema Set Shield

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:

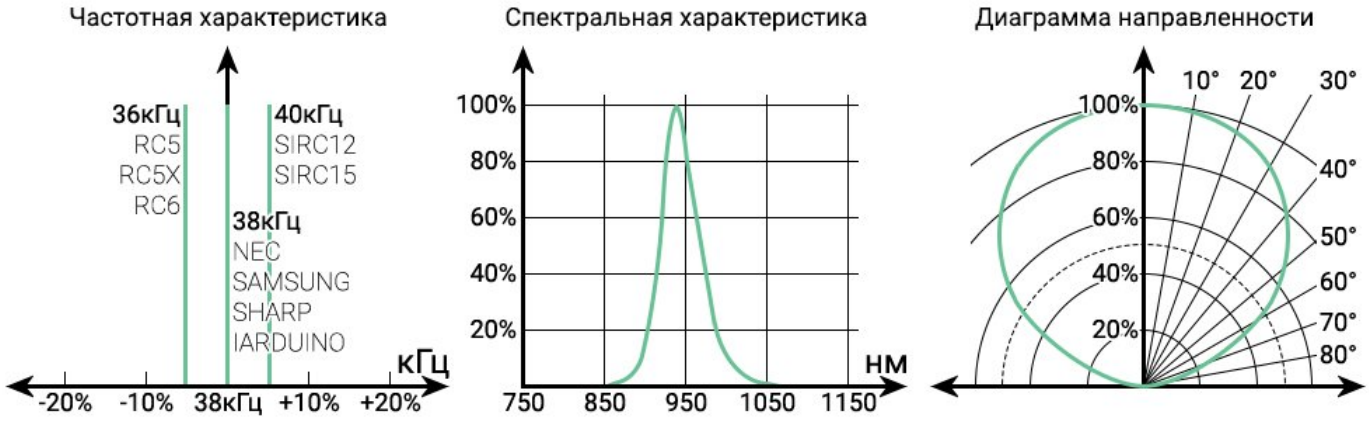
Входное напряжение питания модуля 3,3В или 5В постоянного тока (поддерживаются оба напряжения питания), подаётся на выводы Vcc и GND.

## Подробнее о модуле:

Модуль построен на базе, микроконтроллера STM32F030F4, снабжён ИК-приёмником, ИК-светодиодом, светодиодами информирующими о приёме и передаче данных по ИК-каналу, регулятором дальности ИК-передатчика, и собственным стабилизатором напряжения.

Модуль способен принимать и передавать данные по ИК-каналу в соответствии с протоколами: NEC, SAMSUNG, SHARP, SIRC12, SIRC15, RC5, RC5X, RC6 и IARDUINO. Все перечисленные протоколы используют пакетную передачу данных, данные в пакете представлены в виде адреса ИК-устройства (не путать с адресом устройства на шине I2C) и команды для него. При получении данных по ИК-каналу, модуль способен самостоятельно определить протокол который был использован для их отправки.

## Параметры ИК передатчика:




## Параметры ИК приёмника:


Частотные и спектральные параметры приёмника и передатчика совместимы, а по диаграммам направленности можно определить как снизится уровень ИК сигнала при повороте передатчика или приёмника друг от дура.

Дальность ИК связи зависит от мощности передатчика (настраивается поворотом резистора на плате модуля), и угла поворота приёмника и(или) передатчика друг от друга.

Мощность излучения требуемая приёмнику для уверенного приёма ИК сигнала

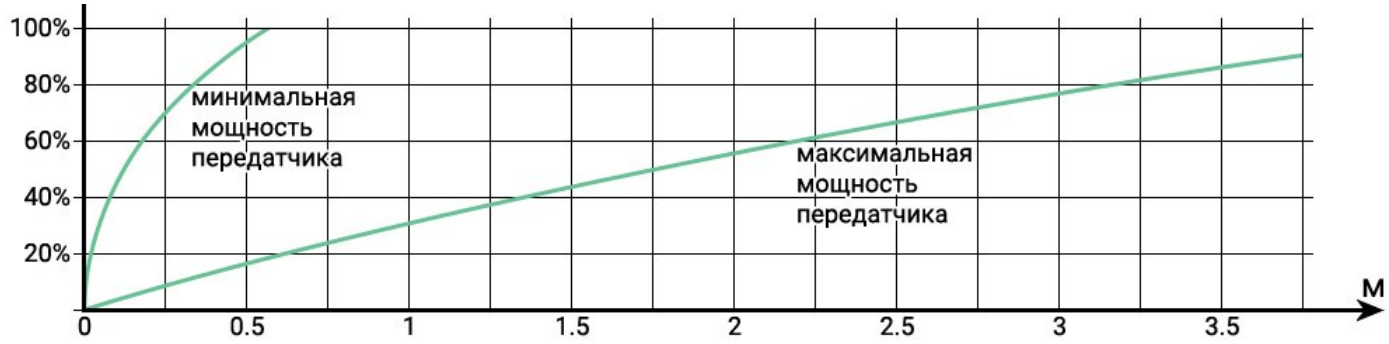



График зависимости требуемой мощности от дистанции, вместе с диаграммами направленности позволяет определить углы и дистанцию уверенного ИК приёма.

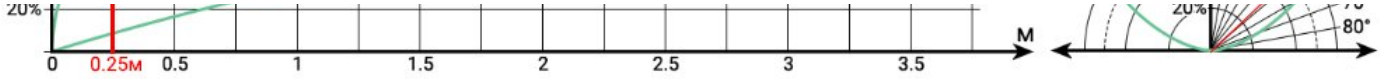
## Пример:

В данном примере поясняется как пользоваться графиком уверенного ИК приема и диаграммами направленности.


Пояснение 1

Предположим что расстояние между приёмником и передатчиком равно 25см (0.25м), а мощность передатчика установлена в минимум. Тогда необходимо провести вертикальную линию от 0.25м до пересечения с графиком минимальной мощности передатчика, точка касания укажет сколько мощности излучения ИК передатчика должно дойти по ИК приёмника для обеспечения уверенного ИК приёма. В примере (Пояснение 1) это не менее 70%. Из диаграммы направленности приёмника (справа) видно, что для обеспечения указанной мощности, его нельзя отклонять от передатчика более чем на 30°, при условии что сам передатчик направлен точно на приёмник.

Если спроецировать мощность на диаграмму направленности передатчика (Пояснение 2), а не приёмника, то будет видно, что для обеспечения той же мощности (70%), его нельзя отклонять от приёмника более чем на 47°, при условии что сам приёмник направлен точно на передатчик.




Пояснение 2

По графику уверенного ИК приёма можно определить как увеличится дистанция при обеспечении того же уровня приёма, с теми же углами отклонения, но на максимальной мощности передатчика (Пояснение 3).


Пояснение 3

Для обеспечения того же уровня ИК приёма, до приёмника должно дойти не менее 70% мощности излучения ИК передатчика. Проводим горизонтальную линию от 70% мощности до пересечения с графиком максимальной мощности ИК передатчика, точка касания укажет на дистанцию. В примере (Пояснение 3) это 2.7м.

Модуль позволяет:

- Менять свой адрес на шине I2C.
- Управлять внутренней подтяжкой линий шины I2C (по умолчанию включена).
- Узнать версию прошивки модуля.
- Передать данные по ИК-каналу однократно (одним пакетом).
- Прочитать принятые по ИК-каналу данные.
- Передавать данные автоматически с заданным интервалом между ИК-пакетами повторов.
- При автоматической отправке данных, можно задать режим хаотичной смены интервалов между пакетами (в диапазоне ±50% от заданного), что позволит передавать данные по ИК-каналу от нескольких модулей, в одном направлении, одновременно.
- Выбрать один из 9 доступных протоколов для передачи данных по ИК-каналу.
- Выбрать один из 9 доступных протоколов для получения данных по ИК-каналу.
- Определить наличие пакетов повторов среди принимаемых данных (определить факт удержания кнопки пульта).
- Определить ИК-протокол принимаемых данных.

Для работы с Trema модулем ИК-приёмник/передатчик Flash-I2C, предлагаем воспользоваться разработанной нами [библиотекой iarduino_I2C_IR](#) позволяющей реализовать все возможности модуля.

## Примеры:

В данном разделе раскрыты примеры управления модулем с использованием библиотеки iarduino_I2C_IR. Сама библиотека содержит больше примеров, доступных из меню Arduino IDE: Файл / Примеры / iarduino I2C IR (модуль ИК-приёмник/передатчик).

### Смена адреса модуля на шине I2C:

Пример меняет текущий адрес модуля шины I2C на указанный в скетче и сохраняет его в энергонезависимую память, значит адрес сохранится и после отключения питания. Для работы скетча не требуется знать какой сейчас адрес у модуля.

```
uint8_t newAddress = 0x09;                          // Назначаемый модулю адрес (0x07 < a
                                                    //
#include <iarduino_I2C_IR.h>                        // Подключаем библиотеку для работы с
iarduino_I2C_IR ir;                                 // Объявляем объект ir для работы с ф
                                                    // Если при объявлении объекта указат
void setup(){                                       //
     Serial.begin(9600);                            //
     if( ir.begin() ){                              // Инициируем работу с модулем.
         Serial.print("Найден модуль 0x");          //
         Serial.println( ir.getAddress(), HEX );    // Выводим текущий адрес модуля.
         if( ir.changeAddress(newAddress) ){        // Меняем адрес модуля на newAddress.
             Serial.print("Адрес изменён на 0x");   //
             Serial.println(ir.getAddress(),HEX);   // Выводим текущий адрес модуля.
         }else{                                     //
             Serial.println("Адрес не изменён!");   //
         }                                          //
     }else{                                         //
         Serial.println("Модуль не найден!");       //
     }                                              //
}                                                   //
                                                    //
void loop(){}                                       //
```

Для работы данного примера, на шине I2C должен быть только один модуль.

Данный скетч демонстрирует не только возможность смены адреса на указанный в переменной `newAddress` , но и обнаружение, и вывод текущего адреса модуля на шине I2C.

### Получение данных с ИК пульта ДУ:

В данном примере принимаются данные отправленные ИК пультом ДУ по протоколу NEC.

```
#include <iarduino_I2C_IR.h>                        // Подключаем библиотеку для работы с
```

```
iarduino_I2C_IR ir(0x09);                       // Объявляем объект ir   для работы с
                                                // Если объявить объект без указания
void setup(){                                   //
     Serial.begin(9600);                        // Инициируем работу с шиной UART для
     ir.begin();                                // Инициируем работу с ИК-приёмником/
     ir.setProtocol(IR_NEC);                    // Указываем протокол для приёма/пере
}                                               //
                                                //
void loop(){                                    //
     if( ir.check(true) ){                      // Если ИК приёмник получил пакет дан
         Serial.print("Адрес=");                // Выводим текст.
         Serial.print(ir.address);              // Выводим принятый адрес ИК-устройст
         Serial.print(", команда=");            // Выводим текст.
         Serial.print(ir.command);              // Выводим принятую команду для ИК-ус
         Serial.print(", кнопка ");             // Выводим текст.
         if( ir.key_press ){                    // Если функция check() среагировала
             Serial.print("нажимается");        // Выводим текст.
         }else{                                 // Если функция check() среагировала
             Serial.print("удерживается");      // Выводим текст.
         }                                      //
         Serial.println(".");                   // Выводим текст.
         delay(100);                            // Ждём 100 мс.
     }                                          //
}                                               //
```

В коде `Setup()` данного скетча происходит инициализация модуля и указание протокола приёма/передачи данных. Библиотека поддерживает 9 ИК протоколов: IR_NEC, IR_SAMSUNG, IR_SHARP, IR_SIRC12, IR_SIRC15, IR_RC5, IR_RC5X, IR_RC6 и IR_IARDUINO.

В коде `loop()` постоянно выполняются проверка получения данных по ИК каналу, обращением к функции `check()`. Данная функция принимает 1 параметр - флаг разрешающий реагировать на пакеты повторов. Если флаг = `true`, то функция будет реагировать и на нажимание и на удержание кнопок ИК пульта ДУ, если флаг = `false` (или отсутствует), то функция будет реагировать только на нажатие кнопок.

Если ИК приёмник получил данные по ИК каналу, то функция `check()` вернёт `true` и скетч выведет следующие данные: `address` - адрес ИК устройства отправленный пультом ДУ, `command` - команда ИК устройству отправленная пультом, и `key_press` - флаг указывающий что кнопка ИК пульта ДУ только что нажата, а не удерживается.

### Отправка данных по ИК каналу:

Пример имитирует нажатие одной и той же кнопки ИК пульта ДУ с её удержанием на 5 секунд в нажатом состоянии и отпусканием на 5 секунд.

```
#include <iarduino_I2C_IR.h>                    // Подключаем библиотеку для работы с
```

```
iarduino_I2C_IR ir(0x09);                       // Объявляем объект ir   для работы с
                                                // Если объявить объект без указания
void setup(){                                   //
     ir.begin();                                // Инициируем работу с ИК-приёмником/
     ir.setProtocol(IR_RC5);                    // Указываем протокол для приёма/пере
}                                               //
                                                //
void loop(){                                    //
//   Указываем модулю отправлять ИК пакеты:     //
     ir.autoSend(0x01, 0x10);                   // Указываем модулю отправить пакет д
     delay(5000);                               // Ждём 5 секунд. Всё это время модул
//   Останавливаем отправку ИК пакетов:         //
     ir.autoStop();                             // Указываем модулю прекратить отправ
     delay(5000);                               // Ждём 5 секунд.
}                                               //
```

В коде `Setup()` данного скетча происходит инициализация модуля и указание протокола приёма/передачи данных. Библиотека поддерживает 9 ИК протоколов: IR_NEC, IR_SAMSUNG, IR_SHARP, IR_SIRC12, IR_SIRC15, IR_RC5, IR_RC5X, IR_RC6 и IR_IARDUINO.

В коде `loop()` постоянно выполняется имитация нажатия и отпускания кнопки ИК пульта ДУ. Протоколы поддерживаемые библиотекой определяют, что передача данных осуществляется пакетами в состав которых входят значения адреса ИК устройства (в примере 0x01) и команды для ИК устройства (в примере 0x10). С задачей передачи этих данных справляется функция `autoSend()` которая принимает значения адреса и команды, и указывает модулю самостоятельно передавать эти данные пока не будет вызвана функция `autoStop()` .

Если задержку после обращения к функции `autoSend()` сократить до 100 мс, то данный скетч будет передавать данные так же как и следующий ниже, так как за 100 мс модуль успеет отправить только один пакет ИК данных по протоколу IR_RC5.

### Отправка данных по ИК каналу:

Пример имитирует кратковременные нажатия одной и той же кнопки ИК пульта ДУ, с паузами между нажатиями в 5 секунд.

```
#include <iarduino_I2C_IR.h>                    // Подключаем библиотеку для работы с
iarduino_I2C_IR ir(0x09);                       // Объявляем объект ir   для работы с
                                                // Если объявить объект без указания
void setup(){                                   //
     ir.begin();                                // Инициируем работу с ИК-приёмником/
     ir.setProtocol(IR_RC5);                    // Указываем протокол для приёма/пере
}                                               //
                                                //
void loop(){                                    //
//   Указываем модулю отправить один ИК пакет:  //
```

```
    ir.send(0x01, 0x10);                    // Указываем модулю однократно отправ
    ir.wait();                              // Ждём завершения отправки ИК пакета
    delay(5000);                            // Ждём 5 секунд.
}                                           //
```

В коде `Setup()` данного скетча происходит инициализация модуля и указание протокола приёма/передачи данных. Библиотека поддерживает 9 ИК протоколов: IR_NEC, IR_SAMSUNG, IR_SHARP, IR_SIRC12, IR_SIRC15, IR_RC5, IR_RC5X, IR_RC6 и IR_IARDUINO.

В коде `loop()` выполняется обращение к функции `send()`, каждые 5 секунд. Данная функция передаёт только один пакет с указанными данными в соответствии с определённым ранее ИК протоколом. После функции `send()` выполняется обращение к функции `wait()`, которая приостанавливает скетч до завершения отправки данных по ИК-каналу. Эту функцию в данном скетче можно не вызывать. Но она может пригодится в тех случаях, когда требуется отправить несколько разных пакетов друг за другом.

### Определение ИК протокола получаемых данных:

Пример ждёт получение любых данных, определяет ИК протокол по которому они были отправлены, выводит название протокола в монитор и далее принимает данные отправленные только в этом протоколе.

```
#include <iarduino_I2C_IR.h>                    // Подключаем библиотеку для работы с
iarduino_I2C_IR ir(0x09);                       // Объявляем объект ir для работы с ф
                                                // Если объявить объект без указания
void setup(){                                   //
    Serial.begin(9600);                         // Инициируем работу с шиной UART для
    ir.begin();                                 // Инициируем работу с ИК-приёмником/
    Serial.println("Нажмите кнопку пульта");    // Выводим текст.
    Serial.println("не менее чем на 0,5 сек."); // Выводим текст.
//  Указываем модулю искать ИК-протокол:        //
    ir.seeProtocol_RX();                        // Переводим ИК приёмник в режим опре
//  Ждём обнаружение ИК-протокола:              //
    while(ir.getProtocol_RX()==IR_UNDEFINED){;} // Выполняем цикл while пока ИК-прото
//  Далее код выполнится если протокол найден:  //
    uint8_t i = ir.getProtocol_RX();            // Сохраняем ИК протокол обнаруженный
    Serial.print("Обнаружен протокол ");        // Выводим текст.
    switch( i ){                                //
     case IR_NEC:      Serial.print("NEC");     break; // Обнаружен протокол NEC.
     case IR_SAMSUNG:  Serial.print("SAMSUNG"); break; // Обнаружен протокол SAMSUNG.
     case IR_SHARP:    Serial.print("SHARP");   break; // Обнаружен протокол SHARP.
     case IR_SIRC12:   Serial.print("SIRC12");  break; // Обнаружен протокол SONY.
     case IR_SIRC15:   Serial.print("SIRC15");  break; // Обнаружен протокол SONY.
     case IR_RC5:      Serial.print("RC5");     break; // Обнаружен протокол PHILIPS.
     case IR_RC5X:     Serial.print("RC5X");    break; // Обнаружен протокол PHILIPS.
     case IR_RC6:      Serial.print("RC6");     break; // Обнаружен протокол PHILIPS.
```

```
      case IR_IARDUINO: Serial.print("IARDUINO"); break; // Обнаружен протокол IARDUINO
    }                   Serial.print(".\r\n");    //
//  ir.setProtocol( i, ir.modeRC6 );              // Указываем использовать обнаруженны
}                                                 //
                                                  //
void loop(){                                      //
    if( ir.check(true) ){                         // Если ИК приёмник получил пакет дан
        Serial.print("Адрес=");                   // Выводим текст.
        Serial.print(ir.address);                 // Выводим принятый адрес ИК-устройст
        Serial.print(", команда=");               // Выводим текст.
        Serial.print(ir.command);                 // Выводим принятую команду для ИК-ус
        Serial.println(".");                      // Выводим текст.
        delay(100);                               // Ждём 100 мс.
    }                                             //
}                                                 //
```

В коде `Setup()` данного скетча происходит инициализация модуля, далее обращением к функции `seeProtocol_RX()` модуль переходит в режим чтения ИК протокола. Функция `getProtocol_RX()` возвращает ИК протокол используемый ИК приёмником, но пока тот не получит данные по ИК-каналу и не определит их протокол, функция `getProtocol_RX()` будет возвращать `IR_UNDEFINED`. Как только протокол будет обнаружен, он будет выведен в монитор. Последняя строка кода `setup()` позволяет применить обнаруженный приёмником протокол и для передатчика, указав его в качестве первого аргумента функции `setProtocol()` или `setProtocol_TX()`, эта строка закомментирована, так как в данном примере передатчик не используется.

В коде `loop()` постоянно выполняются проверка получения данных по ИК каналу, обращением к функции `check()`. Данная функция принимает 1 параметр - флаг разрешающий реагировать на пакеты повторов. Если флаг = `true`, то функция будет реагировать и на нажимание и на удержание кнопок ИК пульта ДУ, если флаг = `false` (или отсутствует), то функция будет реагировать только на нажатие кнопок.

Если ИК приёмник получил данные по ИК каналу, то функция `check()` вернёт `true` и скетч выведет значение `address` - адрес ИК устройства отправленный пультом ДУ и `command` - команда ИК устройству отправленная пультом.

### Получение данных от модулей линейки «Дорожное движение»:

Пример выводит номер знака, команду встречного автомобиля, или разрешённые светофором направления, при обнаружении данных от соответствующих модулей линейки «дорожное движение». Дополнительно в примере реализована постоянная отправка данных, как от автомобиля трассы, для информирования других автомобилей о своём присутствии.

```
#include <iarduino_I2C_IR.h>                      // Подключаем библиотеку для работы с
iarduino_I2C_IR ir(0x09);                         // Объявляем объект ir для работы с ф
```

```
                                                    // Если объявить объект без указания
void setup(){                                       //
     Serial.begin(9600);                            // Инициируем работу с шиной UART для
     ir.begin();                                    // Инициируем работу с ИК-приёмником/
     ir.setProtocol(IR_IARDUINO);                   // Указываем протокол для приёма/пере
//   Отправляем данные другим автомобилям:          //
//   ir.setInterval_TX(200, true);                  // Указываем модулю использовать инте
//   ir.autoSend(MODUL_CAR, 10);                    // Указываем модулю постоянно отправл
}                                                   // В качестве команды автомобиля (вме

void loop(){
//   Если приняты данные:
     if( ir.check(true) ){
     //  Если данные приняты от автомобиля:
         if( ir.device==MODUL_CAR ){
             Serial.print("Автомобиль: принята команда ");
             Serial.print(ir.command); // Выводим номер команды.
             Serial.println(".");
         }
     //  Если данные приняты от дорожного знака:
         if( ir.device==MODUL_SIGN ){
             Serial.print("Дорожный знак: ");
             Serial.print(ir.sign_str); // Выводим строку с номером знака.
             if( ir.sign_tab ){
                 Serial.print(", под знаком есть табличка - ");
                 if( ir.sign_tab==SIGN_CAR_TRUCK    ){ Serial.print("грузовые автомобил
                 if( ir.sign_tab==SIGN_CAR_LIGHT    ){ Serial.print("легковые автомобил
                 if( ir.sign_tab==SIGN_DISTANCE_050 ){ Serial.print("50 метров"); }
                 if( ir.sign_tab==SIGN_DISTANCE_100 ){ Serial.print("100 метров"); }
                 if( ir.sign_tab==SIGN_DISTANCE_150 ){ Serial.print("150 метров"); }
                 if( ir.sign_tab==SIGN_DISTANCE_200 ){ Serial.print("200 метров"); }
                 if( ir.sign_tab==SIGN_DISTANCE_250 ){ Serial.print("250 метров"); }
             }
             Serial.println(".");
         }
     //  Если данные приняты от светофора:
         if( ir.device==MODUL_TLIGHT ){
             Serial.print("Светофор: регулирует перекрёсток с движением ");
             if( ir.track_L ){ Serial.print("налево, "  ); }
             if( ir.track_R ){ Serial.print("направо, " ); }
             if( ir.track_F ){ Serial.print("прямо, "   ); }
             Serial.print("сейчас сигналы светофора указывают");
             Serial.print(": прямо ");
             switch(ir.forvard){
                 case 0: Serial.print("нельзя"              ); break;
                 case 1: Serial.print("можно"               ); break;
                 case 2: Serial.print("можно если нет помех"); break;
             }
```

```
            Serial.print(", направо ");
            switch(ir.right){
                case 0: Serial.print("нельзя"              ); break;
                case 1: Serial.print("можно"               ); break;
                case 2: Serial.print("можно если нет помех"); break;
            }
            Serial.print(", налево ");
            switch(ir.left){
                case 0: Serial.print("нельзя"              ); break;
                case 1: Serial.print("можно"               ); break;
                case 2: Serial.print("можно если нет помех"); break;
            }
            Serial.println(".");
        }
    }
}
```

В коде `Setup()` данного скетча происходит инициализация модуля и указание протокола приёма/передачи данных. Библиотека поддерживает 9 ИК протоколов: IR_NEC, IR_SAMSUNG, IR_SHARP, IR_SIRC12, IR_SIRC15, IR_RC5, IR_RC5X, IR_RC6 и IR_IARDUINO.

Далее следуют две закомментированные строки. Если их раскомментировать, то автомобиль будет постоянно передавать адрес команду 10 другим автомобилям.

Обращением к функции `setInterval_TX()` настраивается интервал между отправляемыми ИК пакетами, а обращением к функции `autoSend()` отправляются данные (адрес MODUL_CAR и команда 10). Так как в примере нет функции `autoStop()`, то данные будут отправляться модулем постоянно. Допускается менять отправляемые данные новыми обращениями к функции `autoSend()` без вызова функции `autoStop()`.

В качестве команд отправляемых автомобилями можно указывать значения от 0 до 255. Вы сами можете придумать какая команда что значит (внимание, уступи дорогу, остановись, уступаю дорогу, поворачиваю влево, еду прямо, еду быстро, еду медленно и т.д.).

В коде `loop()` постоянно выполняется проверка получения данных по ИК каналу `if( ir.check(true) )`. Если данные получены, то выполняются проверки: не отправлены ли эти данные автомобилем `if( ir.device==MODUL_CAR )`, не отправлены ли эти данные модулем «дорожный знак» `if( ir.device==MODUL_SIGN )`, не отправлены ли эти данные модулем «светофор» `if( ir.device==MODUL_TLIGHT )`. Если пройдена любая из указанных проверок, то выводятся соответствующие данные:

```
Автомобиль: принята команда 10.
Дорожный знак: 2.4, под знаком есть табличка - грузовые автомобили.
Светофор: регулирует перекрёсток с движением направо, прямо, сейчас сигналы светофора у
```

Переменные `ir.forvard` , `ir.right` и `ir.left` разрешающие движение на светофоре могут принимать значения: `0` - движение запрещено, `1` - движение разрешено, или `2` - можно если нет помех. Последнее означает что светофор "сломан" (постоянно мигает жёлтый) или разрешается поворот налево по основному зелёному сигналу светофора у которого нет секции поворота налево.

## Описание функций библиотеки:

В данном разделе описаны функции библиотеки iarduino_I2C_IR для работы с Trema модулями ИК-приёмник/передатчик, Flash-I2C.

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_IR.h>   // Подключаем библиотеку для работы с модулем.
iarduino_I2C_IR ir(0x09);      // Создаём объект ir для работы с функциями и методами б
```

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

```
#include <iarduino_I2C_IR.h>   // Подключаем библиотеку для работы с модулем.
iarduino_I2C_IR ir;            // Создаём объект ir для работы с функциями и методами б
```

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( ir.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( ir.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( ir.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( ir.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( ir.getVersion(), HEX );
```

### Функция setPullI2C();

- Назначение: Управление внутрисхемной подтяжкой линий шины I2C.
- Синтаксис: setPullI2C( [ФЛАГ] );
- Параметры:
  - bool ФЛАГ требующий установить внутрисхемную подтяжку линий шины I2C (true или false).
- Возвращаемое значение:
  - bool - результат включения / отключения внутрисхемной подтяжки (true или false).
- Примечание:
  - Вызов функции без параметра равносилен вызову функции с параметром true - установить.
  - Флаг установки внутрисхемной подтяжки сохраняется в энергонезависимую память модуля, а значит будет действовать и после отключения питания.
  - Внутрисхемная подтяжка линий шины I2C осуществляется до уровня 3,3 В, но допускает устанавливать внешние подтягивающие резисторы и иные модули с подтяжкой до уровня 3,3 В или 5 В, вне зависимости от состояния внутрисхемной подтяжки модуля.
- Пример:

```
if( ir.setPullI2C(true ) ){ Serial.print( "Внутрисхемная подтяжка установлена." ); }
if( ir.setPullI2C(false) ){ Serial.print( "Внутрисхемная подтяжка отключена."   ); }
```

### Функция getPullI2C();

- Назначение: Запрос состояния внутрисхемной подтяжки линий шины I2C.
- Синтаксис: getPullI2C();
- Параметры: Нет.
- Возвращаемое значение: bool - ФЛАГ включения внутрисхемной подтяжки (true или false).
- Пример:

```
if( ir.getPullI2C() ){ Serial.print( "Внутрисхемная подтяжка включена."  ); }
else                 { Serial.print( "Внутрисхемная подтяжка отключена." ); }
```

### Функция setProtocol();

- Назначение: Указание ИК-протокола для приёма и передачи данных.
- Синтаксис: setProtocol( ПРОТОКОЛ [,РЕЖИМ] );
- Параметры:
  - uint8_t ПРОТОКОЛ - может принимать одно из следующих значений:
    - IR_NEC - кодирование длинной паузы. Пакет отправляется за 68 мс.
    - IR_SAMSUNG - кодирование длинной паузы. Пакет отправляется за 53...71 мс.
    - IR_IARDUINO - кодирование длинной паузы. Пакет отправляется за 26 мс.
    - IR_SHARP - кодирование длинной паузы. Пакет отправляется за 16...30 мс.
    - IR_SIRC12 - кодирование длинной импульса. Пакет отправляется за 17...25 мс.
    - IR_SIRC15 - кодирование длинной импульса. Пакет отправляется за 21...30 мс.
    - IR_RC5 - бифазное кодирование. Пакет отправляется за 25 мс.

        - IR_RC5X - бифазное кодирование. Пакет отправляется за 25 мс.
        - IR_RC6 - бифазное кодирование. Пакет отправляется за 23 мс.
    - uint8_t РЕЖИМ - необязательный параметр, указывается только для протокола IR_RC6 и определяет режим данного протокола от 0 до 7. Если указать протокол IR_RC6 без параметра РЕЖИМ, то будет использован режим номер 0.
- Возвращаемое значение: bool - результат применения нового протокола (true или false).
- Примечание:
    - Функция устанавливает протокол для приёма и передачи данных по ИК-каналу.
    - Модуль позволяет установить разные протоколы для приёма и передачи данных, для этого в библиотеке есть две функции с теми же параметрами:
        - **setProtocol_TX(** ПРОТОКОЛ [,РЕЖИМ] **)**; - установка протокола для передачи данных.
        - **setProtocol_RX(** ПРОТОКОЛ [,РЕЖИМ] **)**; - установка протокола для приёма данных.
    - ИК-приёмник модуля не будет реагировать на данные отправленные по другому протоколу.
- Пример:

```
ir.setProtocol(IR_IARDUINO); // Указываем ИК-протокол IR_IARDUINO для приёма и передачи
ir.setProtocol(IR_RC6, 2);   // Указываем ИК-протокол IR_RC6 и режим 2 для приёма и пер
```

### Функция getProtocol();

- Назначение: Запрос ИК-протокола используемого для приёма и передачи данных.
- Синтаксис: getProtocol();
- Параметры: Нет.
- Возвращаемое значение: uint8_t ПРОТОКОЛ - может принимать одно из следующих значений:
    - IR_UNDEFINED - протокол не определён.
    - IR_NEC.
    - IR_SAMSUNG.
    - IR_IARDUINO.
    - IR_SHARP.
    - IR_SIRC12.
    - IR_SIRC15.
    - IR_RC5.
    - IR_RC5X.
    - IR_RC6.
- Примечание:
    - Функция возвращает ИК протокол используемый для приёма и передачи данных по ИК-каналу. Если протокол используемый для приёма данных отличается от протокола используемого для передачи данных, то функция вернёт IR_UNDEFINED.
    - Модуль позволяет запросить ИК протокол используемый для приёма или для передачи данных, для этого в библиотеке есть две функции:
        - **getProtocol_TX()**; - запрос протокола используемого для передачи данных.
        - **getProtocol_RX()**; - запрос протокола используемого для приёма данных.

    - Если для приёма или передачи данных используется протокол IR_RC6, то режим данного протокола будет доступен из переменной **modeRC6**, после обращения к любой из функций запроса протокола.
- Пример:

```
uint8_t i = ir.getProtocol();    // Запрос ИК-протокола используемого для приёма и пере
uint8_t j = ir.getProtocol_TX(); // Запрос ИК-протокола используемого для передачи данн
uint8_t k = ir.getProtocol_RX(); // Запрос ИК-протокола используемого для приёма данных
```

### Функция seeProtocol_RX();

- Назначение: Перевод ИК приёмника в режим определения протокола получаемых данных.
- Синтаксис: seeProtocol_RX();
- Параметры: Нет
- Возвращаемое значение: bool - результат перевода в режим определения (true или false).
- Примечание:
    - После обращения к данной функции, ИК приёмник сбросит протокол используемый для приёма данных, самостоятельно определит ИК протокол первых полученных данных и установит его для дальнейшего приёма данных.
    - ИК приёмник может определить протокол только если получит пакет данных и пакет повтора. Значит для определения протокола с ИК пульта ДУ необходимо нажать и удерживать любую кнопку пульта не менее 0,3 - 0,5 секунд.
    - Узнать протокол определённый приёмником можно при помощи функции getProtocol_RX().
- Пример:

```
ir.seeProtocol_RX();                              // Переводим ИК приёмник в режим опр
while( ir.getProtocol_RX()==IR_UNDEFINED ){;}     // Ждём пока запрос протокола исполь
ir.setProtocol( ir.getProtocol_RX(), ir.modeRC6 ); // Указываем использовать обнаруженн
```

### Функция setInterval();

- Назначение: Установка интервала между пакетами повторов. Используется ИК передатчиком при постоянной отправке данных и ИК приёмником для ожидания пакетов повторов.
- Синтаксис: setInterval( ИНТЕРВАЛ [, ФЛАГ ] );
- Параметры:
    - uint8_t ИНТЕРВАЛ - значение от 24 до 255 определяющее время в миллисекундах между пакетами повторов. Если в качестве интервала указать 0, то модуль будет использовать интервал между пакетами повторов в соответствии с используемым ИК протоколом.
    - bool ФЛАГ - разрешающий ИК передатчику хаотично менять интервал между пакетами повторов в диапазоне ±50% от заданного, что позволит передавать данные по ИК-каналу от нескольких модулей, в одном направлении, одновременно. Если ФЛАГ не указан, то интервал меняться не будет.
- Возвращаемое значение: bool - результат применения нового интервала (true или false).

- Примечание:
  - Указанный ИНТЕРВАЛ между пакетами повторов используется ИК передатчиком для автоматической отправки пакетов (см. функцию autoSend()).
  - Указанный ИНТЕРВАЛ между пакетами повторов используется ИК приёмником для отличия нажатия кнопки пульта от её удержания.
  - Модуль позволяет указать разные интервалы для приёмника и передатчика, для этого в библиотеке есть две функции:
    - **setInterval_TX(** ИНТЕРВАЛ [, ФЛАГ ] **)**; - установка интервала для передатчика.
    - **setInterval_RX(** ИНТЕРВАЛ **)**; - установка интервала ожидания для приёмника.
- Пример:

```
ir.setInterval(100);      // Указываем модулю использовать интервал в 100 мс между паке
ir.setInterval(100,true); // Указываем модулю использовать интервал в 100 мс между паке
```

### Функция send();

- Назначение: Однократная отправка данных по ИК-каналу.
- Синтаксис: send( АДРЕС, КОМАНДА );
- Синтаксис: send( КОД );
- Параметры:
  - uint8_t АДРЕС - значение от 0 до 255 определяющее адрес ИК устройства для которого предназначена команда:
  - uint8_t КОМАНДА - значение от 0 до 255 определяющее команду для ИК устройства.
  - uint16_t КОД - двухбайтное число в котором старший байт является адресом ИК устройства, а младший - командой для ИК устройства.
- Возвращаемое значение: bool - результат отправки данных модулю (true или false).
- Примечание:
  - Функция передаёт данные модулю, но не ждёт пока ИК передатчик модуля отправит данные по ИК-каналу. Если после передачи данных модулю требуется дождаться завершения их отправки по ИК-каналу, воспользуйтесь функцией wait().
  - Функция указывает модулю отправить данные однократно, одним пакетом, без отправки пакетов повторов. Аналогичным образом данные отправляет ИК пульт ДУ при однократном и очень кратковременном нажатии на его кнопку.
  - При отправке данных от лица автомобиля трассы, в качестве адреса указывается значение MODUL_CAR, а в качестве команды, любое число от 0 до 255. Вы сами можете придумать какая команда что значит (внимание, уступи дорогу, остановись, уступаю дорогу, поворачиваю влево, еду прямо, еду быстро, еду медленно и т.д.).
  - Для совместимости данной библиотеки с библиотекой iarduino_IR можно воспользоваться функцией **send32(** КОД **)** которая в качестве параметра принимает 4 байтное число.
- Пример:

```
ir.send(0x12, 0x34);       // Однократно отправить по ИК-каналу адрес 0x12 и команду 0x
```

```
ir.send(MODUL_CAR, 0x01);  // Однократно отправить по ИК-каналу команду 0x01 от автомоб
ir.send(0x1234);           // Однократно отправить по ИК-каналу адрес 0x12 и команду 0x
ir.send32(0x48B72CD3);     // Однократно отправить по ИК-каналу адрес 0x12 и команду 0x
```

### Функция wait();

- Назначение: Ожидание завершения однократной отправки данных по ИК-каналу.
- Синтаксис: wait();
- Параметры: Нет.
- Возвращаемое значение: Нет.
- Примечание:
  - Функция ждёт завершение однократной отправки данных, которая была запущена функцией send() или send32().
  - Функция приостанавливает выполнение скетча, постоянно опрашивая модуль по шине I2C, до тех пор пока ИК-передача не будет завершена, но не дольше 500 мс.
  - Время отправки одного пакета данных зависит от используемого ИК протокола (см. функцию setProtocol()).
- Пример:

```
ir.send(0x12, 0x34); // Однократно отправить по ИК-каналу адрес 0x12 и команду 0x34.
ir.wait();           // Дождаться завершения отправки данных по ИК-каналу.
// Эта строка будет выполнена только после отправки данных по ИК-каналу.
```

### Функция autoSend();

- Назначение: Автоматическая отправка пакетов данных по ИК-каналу.
- Синтаксис: autoSend( АДРЕС, КОМАНДА );
- Синтаксис: autoSend( КОД );
- Параметры:
  - uint8_t АДРЕС - значение от 0 до 255 определяющее адрес ИК устройства для которого предназначена команда:
  - uint8_t КОМАНДА - значение от 0 до 255 определяющее команду для ИК устройства.
  - uint16_t КОД - двухбайтное число в котором старший байт является адресом ИК устройства, а младший - командой для ИК устройства.
- Возвращаемое значение: bool - результат отправки данных модулю (true или false).
- Примечание:
  - Функция передаёт данные модулю, для их постоянной отправки по ИК-каналу. Модуль отправляет пакет данных, а за ним постоянно и самостоятельно отправляет пакеты повторов с заранее заданным функцией setInterval() или setInterval_TX() интервалом. Аналогичным образом данные отправляет ИК пульт ДУ при удержании его кнопки.
  - Остановить отправку пакетов повторов можно при помощи функции autoStop().
  - При отправке данных от лица автомобиля трассы, в качестве адреса указывается значение MODUL_CAR, а в качестве команды, любое число от 0 до 255. Вы сами можете придумать

какая команда что значит (внимание, уступи дорогу, остановись, уступаю дорогу, поворачиваю влево, еду прямо, еду быстро, еду медленно и т.д.).
    - Для совместимости данной библиотеки с библиотекой iarduino_IR можно воспользоваться функцией **autoSend32(** КОД **)** которая в качестве параметра принимает 4 байтное число.
- Пример:

```
ir.autoSend(0x12, 0x34);       // Постоянно отправлять по ИК-каналу адрес 0x12 и команд
ir.autoSend(MODUL_CAR, 0x01);  // Постоянно отправлять по ИК-каналу команду 0x01 от авт
ir.autoSend(0x1234);           // Постоянно отправлять по ИК-каналу адрес 0x12 и команд
ir.autoSend32(0x48B72CD3);     // Постоянно отправлять по ИК-каналу адрес 0x12 и команд
```

### Функция autoStop();

- Назначение: Остановка автоматической отправки пакетов по ИК-каналу.
- Синтаксис: autoStop();
- Параметры: Нет.
- Возвращаемое значение: bool - результат остановки пакетов (true или false).
- Примечание:
    - Функция останавливает процесс автоматической отправки данных по ИК-каналу, запущенный функцией autoSend() или autoSend32().
- Пример:

```
ir.autoSend(0x12, 0x34); // Постоянно отправлять по ИК-каналу адрес 0x12 и команду 0x34
delay(1000);             // Ждём 1 секунду.
ir.autoStop();           // Остановить процесс автоматической отправки данных по ИК-кан
```

### Функция check();

- Назначение: Проверка наличия принятых по ИК-каналу данных.
- Синтаксис: check( [ФЛАГ] );
- Параметры:
    - bool ФЛАГ разрешающий реагировать на пакеты повтора (true или false).
- Возвращаемое значение: bool - наличие принятых данных (true или false).
- Примечание:
    - Если функция вызвана с параметром true, то она будет реагировать и на первый пакет данных и на все пакеты повторов (на нажатие и удержание кнопок ИК пульта ДУ).
    - Если функция вызвана с параметром false или без параметра, то она будет реагировать только на первый принятый пакет данных (только на нажатие кнопок ИК пульта ДУ).
    - Функция не реагирует на нулевые данные (если адрес=0 и команда=0).
    - Если функция вернула значение true, значит принятые данные можно прочитать из переменных:
        - uint8_t **address** - переменная хранит принятый байт адреса ИК-устройства.
        - uint8_t **command** - переменная хранит принятый байт команды ИК-устройства.

- uint16_t **code** - переменная хранит двухбайтное число, старший байт которого является адресом ИК-устройства, а младший байт командой ИК-устройства.
- bool **key_press** - флаг указывающий на то, что функция check() среагировала на первый пакет данных, а не на пакет повтора (кнопка пульта нажимается а не удерживается).
- uint32_t **code32** - переменная для совместимости с библиотекой iarduino_IR хранит 4 байтное число состоящее из length бит данных.
- uint8_t **length** - переменная для совместимости с библиотекой iarduino_IR хранит количество информационных бит данных в переменной code32.
- uint8_t **modeRC6** - переменная хранит число от 0 до 7 которое определяет режим используемый протоколом IR_RC6

- При получении данных от модулей линейки «Дорожное движение» доступны данные дополнительных переменных:
  - uint8_t **device** - переменная указывает на наличие данных от модуля линейки «Дорожное движение», может принимать следующие значения:
    - **0** - данные приняты не от модуля линейки «Дорожное движение».
    - **MODUL_TLIGHT** - приняты данные от модуля светофор.
    - **MODUL_SIGN** - приняты данные от модуля знак.
    - **MODUL_CAR** - приняты данные от модуля машина.
  - Информацию переданную автомобилем можно получить из переменной:
    - uint8_t **command** - команда переданная автомобилем (0-255). Вы сами определяете какая команда что значит (внимание, уступи дорогу, остановись, уступаю дорогу, поворачиваю влево, еду прямо, еду быстро, еду медленно и т.д.).
  - Информацию о состоянии светофора можно получить из переменных:
    - bool **track_F** - флаг указывающий о наличии дороги прямо (0-нет, 1-есть).
    - bool **track_R** - флаг указывающий о наличии поворота вправо (0-нет, 1-есть).
    - bool **track_L** - флаг указывающий о наличии поворота влево (0-нет, 1-есть).
    - uint8_t **forvard** - разрешает движения прямо (0-нельзя, 1-можно, 2-если нет помех).
    - uint8_t **right** - разрешает движения вправо (0-нельзя, 1-можно, 2-если нет помех).
    - uint8_t **left** - разрешает движения влево (0-нельзя, 1-можно, 2-если нет помех).
    - uint8_t **gate** - флаг состояния шлагбаума (0-закрыт, 1-открыт).
    - Биты состояний всех цветов светофора и состояния шлагбаума можно прочитать из байта команды ИК-устройства **command**.
    - Если переменные forvard, right или left имеют значение 2 (если нет помех), значит светофор "сломан" (постоянно мигает жёлтый) или разрешается поворот налево по основному зелёному сигналу светофора у которого нет секции поворота налево.
  - Информацию о знаке дорожного движения можно получить из переменных:
    - char* **sign_str** - строка хранящая обозначение знака ПДД ("1.1" ... "7.63.15").
    - uint16_t **sign_int** - переменная хранящая обозначение знака ПДД (10100...76315).
    - uint8_t **sign**[3] - массив хранящий группу, номер и пункт знака ПДД ({1,1,0}... {7,63,15}).
    - uint8_t **sign_tab** - переменная хранящая назначение таблички находящейся под

знаком:

- **0** - под знаком нет таблички уточняющей или ограничивающей его действие
- **SIGN_CAR_TRUCK** - действие знака распространяется только на грузовые ам.
- **SIGN_CAR_LIGHT** - действие знака распространяется только на легковые ам.
- **SIGN_DISTANCE_050** - расстояние до объекта или зона действия знака = 50 м.
- **SIGN_DISTANCE_100** - расстояние до объекта или зона действия знака = 100 м.
- **SIGN_DISTANCE_150** - расстояние до объекта или зона действия знака = 150 м.
- **SIGN_DISTANCE_200** - расстояние до объекта или зона действия знака = 200 м.
- **SIGN_DISTANCE_250** - расстояние до объекта или зона действия знака = 250 м.

- Если приняты данные от модуля машина, значит она появилась в зоне видимости и может стать помехой.

- Пример:

```
if( ir.check(true) ){      // Если ИК приёмник получил пакет данных или пакет повтора.
    Serial.print("Адрес  ="); Serial.println(ir.address);
    Serial.print("Команда="); Serial.println(ir.command);
    Serial.print("Кнопка ");
    if(ir.key_press){         Serial.println("нажимается");  } // Получен пакет данных.
    else{                     Serial.println("удерживается");} // Получен пакет повтора
    Serial.println("----------------");
}
```

### Адреса передаваемые по ИК-каналу зарезервированные для линейки «Дорожное движение»:

- 0x01 - «Автомобиль».
- 0x04 ... 0x1F - «Светофор».
- 0x20 ... 0xFF - «Знак».
- 0x02 ... 0x03 - свободные адреса.