ИК-приёмник (Trema-модуль)

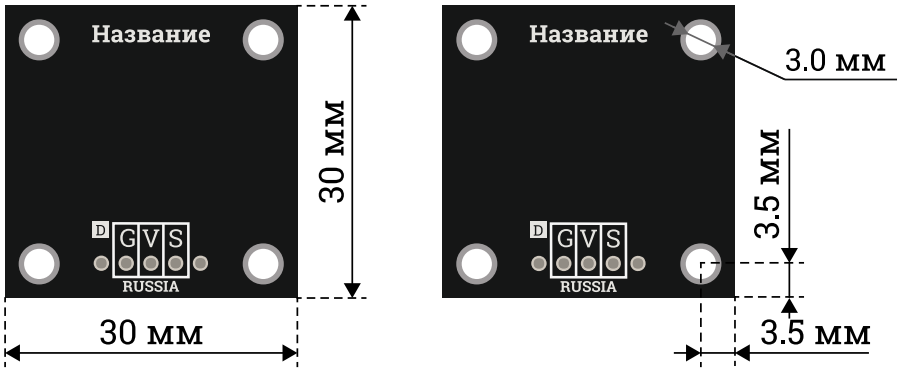
Общие сведения:

Trema-модуль ИК-приёмник - позволяет управлять проектами на расстоянии с помощью обычного ИК-пульта от телевизора или другой техники. Исполнен в линейке Trema-модулей, что позволяет включать модуль в проект, без пайки и макетных плат.

Спецификация:

- Входное напряжение: 2,7 … 5,5 В
- Потребляемый ток: 0,65 … 1,05 мА (при Vcc = 5В) номинально 0,9 мА
- Несущая частота: 38 кГц
- Длинна световой волны: 850 … 1050 нм (пропускаемая фильтром более 80%)
- Чувствительность: 0,17… 30000 мW/м2 (к мощности светового потока)
- Расстояние приёма: до 45 м
- Рабочая температура: -25 … 85 °С
- Угол направленности: ±45°

Все модули линейки "Trema" выполнены в одном формате



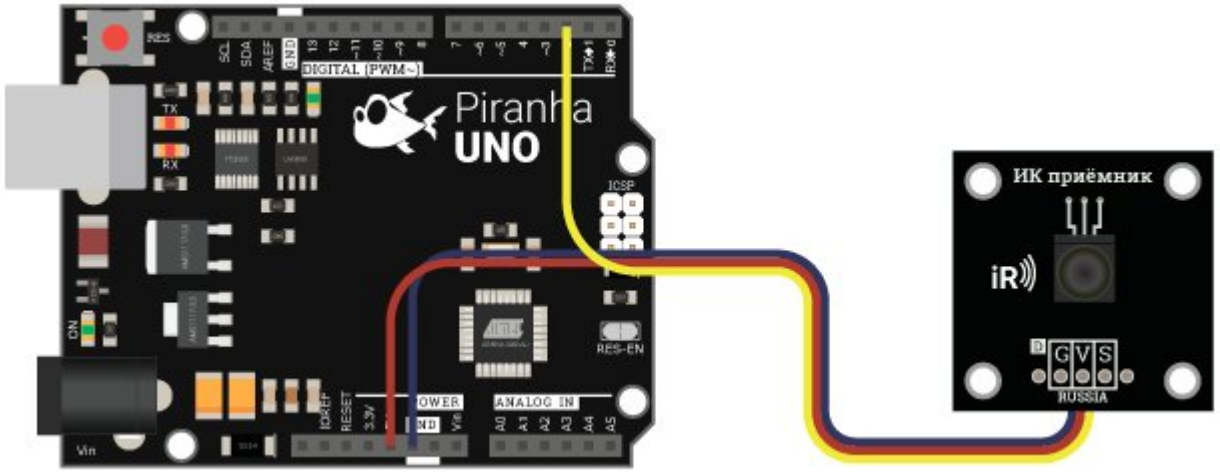
Подключение:

Модуль подключается к любому цифровому выводу arduino. В комплекте имеется кабель для быстрого и удобного подключения к Trema Shield.

Модуль удобно подключать 3 способами, в зависимости от ситуации:

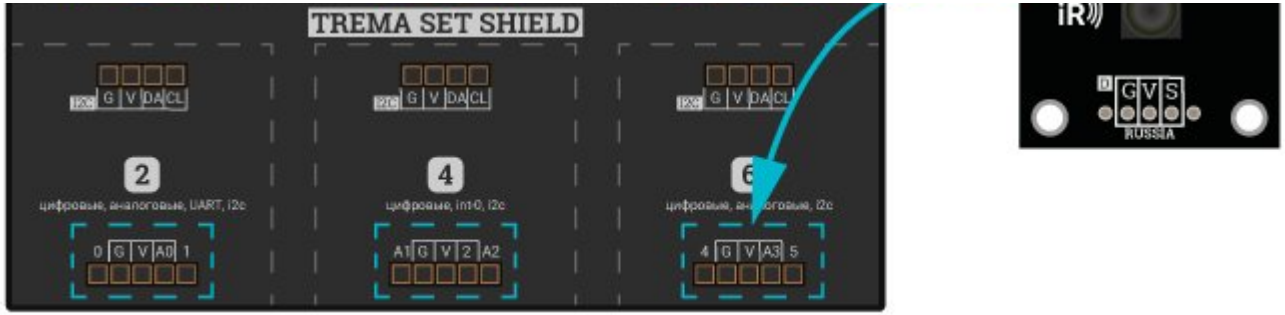
Способ - 1 : Используя проводной шлейф и Piranha UNO

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



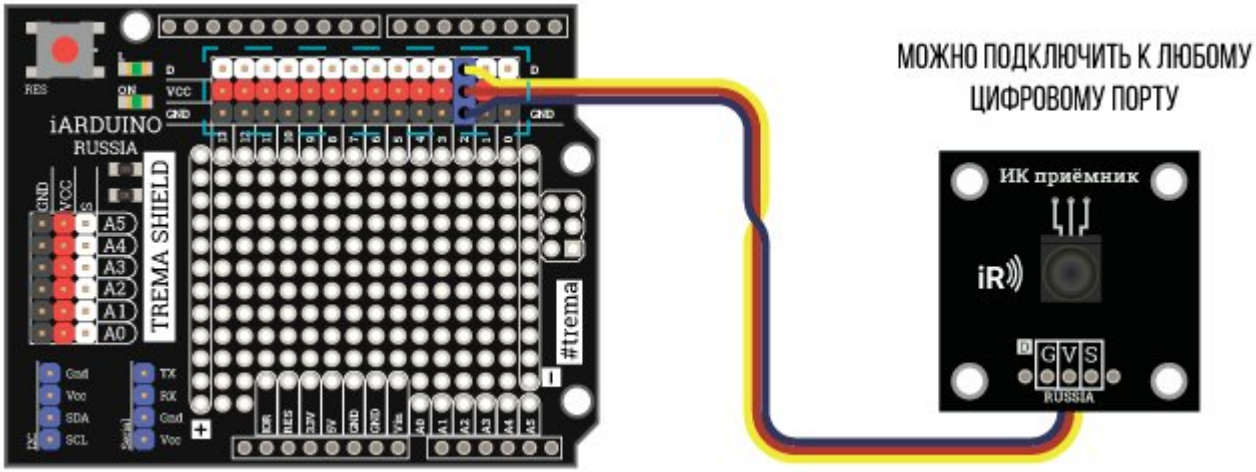
Способ - 2 : Используя Trema Set Shield

Модуль можно подключить к любому из цифровых входов 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.



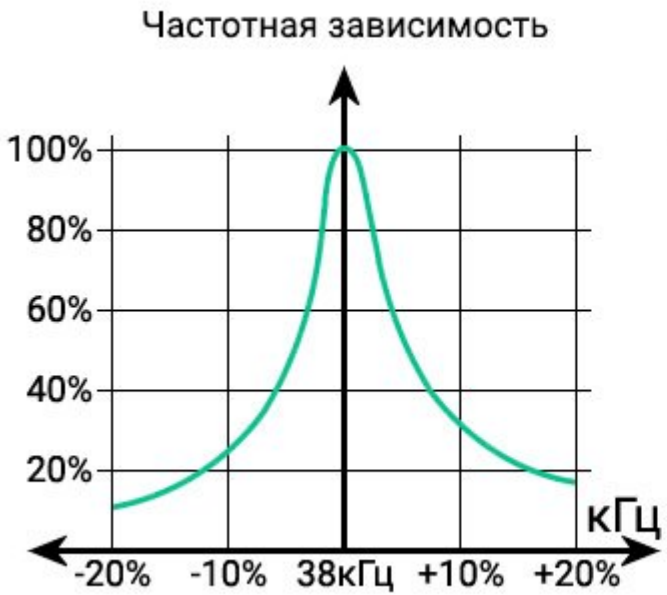
Подробнее о модуле:

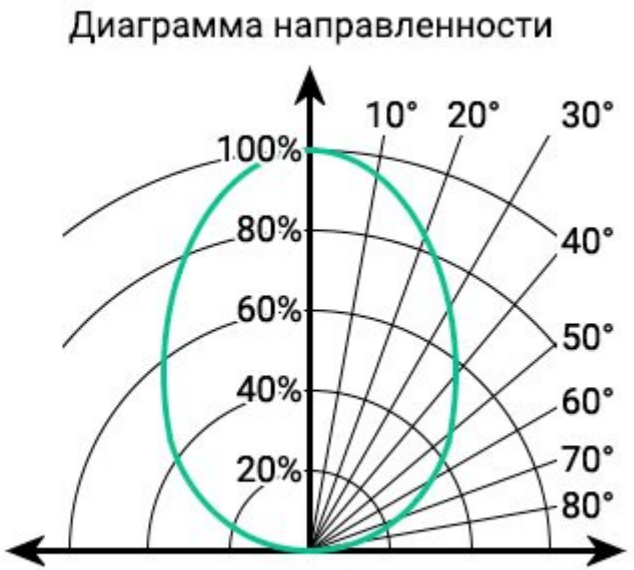
Модуль ИК-приёмника построен на базе модуля TSOP2238, который снабжён:

- фильтром светового потока;
- фотодиодом;
- предусилителем;
- полосовым фильтром несущей частоты - 38 кГц;

- демодулятором;
- операционным усилителем;
- блоками защиты от помех (электромагнитных, световых, пульсаций напряжения)

Благодаря своим характеристикам модуль позволяет работать с большинством ИК-пультов.





Для приёма данных с ИК-пультов, предлагаем воспользоваться библиотекой iarduino_IR, которая позволяет работать с ИК-приёмником и(или) ИК-передатчиком.

Библиотека использует второй аппаратный таймер,
НЕ ВЫВОДИТЕ СИГНАЛЫ ШИМ НА 3 ИЛИ 11 ВЫВОД!

Подробнее про установку библиотеки читайте в нашей инструкции..

Дополнительная информация по работе с модулем:

Пакеты: Практически все пульты отправляют не только информационный пакет (указывающий тип устройства и код нажатой кнопки), но и пакеты повтора, сообщающие устройству об удержании нажатой кнопки. Таким образом принимающее устройство может реагировать на нажатие кнопки однократно или в течении всего времени её удержания.

Например: нажимая и удерживая кнопку с номером телевизионного канала, телевизор переключится на данный канал только один раз. В то время, как нажимая и удерживая кнопку увеличения громкости, телевизор будет её увеличивать в течении всего времени удержания кнопки.

Количество информационных пакетов у большинства пультов равно одному, но некоторые устройства, например кондиционеры, используют 2, 3 и более информационных пакетов.

Состав пакетов: Информационный пакет несёт информацию о коде производителя, типе устройства, коде нажатой кнопки и т.д. Пакеты повтора могут частично или полностью совпадать с информационным пакетом, копировать его биты с инверсией, или не нести никакой информации, представляя последовательность из нескольких одинаковых, для каждого пакета повтора, битов.

Длительность пауз между пакетами: обычно не превышает 200мс.

Протоколы передачи данных: определяют следующие, основные, параметры:

- несущую частоту;
- способ кодирования информации, длительность импульсов и пауз передаваемых битов;
- количество информационных пакетов:
- состав информационного пакета и пакетов повторов;
- длительность пауз между пакетами;
- наличие и форму сигналов Start, Stop и Toggle;

Несущая частота: у большинства пультов равна 38 кГц, именно на эту частоту настроен Trema ИК-приёмник.

Кодирование информации: это принцип передачи битов данных. Выделим три основных вида кодирования, при которых каждый бит передаётся последовательностью из одного импульса и одной паузы:

- кодирование длиной импульсов - сначала передаётся импульс, длина которого зависит от значения передаваемого бита, затем следует пауза, длина которой не зависит от значения бита. Например: в протоколе SIRC (Sony), длина импульса для бита «1» = 1200мкс, а для бита «0» = 600мкс, длина пауз всегда равна 600мкс. Таким образом можно отличить «1» от «0» по длине импульса.
- кодирование длинной пауз - сначала передаётся импульс, длина которого не зависит от значения передаваемого бита, затем следует пауза, длина которой зависит от значения бита. Например: в протоколе NEC, длина паузы для бита «1» = 1687,5мкс, а для бита «0» = 562,5мкс, длина импульсов всегда равна 562,5мкс. Таким образом можно отличить «1» от «0» по длине паузы.
- бифазное кодирование - длина импульса равна длине паузы, а их последовательность определяет тип передаваемого бита. Например: в протоколе RS5 (Philips), для бита «1» импульс следует за паузой, а для бита «0» пауза следует за импульсом. Для протокола NRC (Nokia), наоборот, для бита «1» пауза следует за импульсом, а для бита «0» импульс следует за паузой.

Сигналы Start, Stop и Toggle: по своему названию располагаются в начале, конце или середине пакета.

Stop: При кодировании длинной паузы, нельзя определить значение последнего бита в пакете, так как после пакета следует большая пауза, и последний бит будет всегда определяться как «1», поэтому в пакет добавляется сигнал Stop представляющий из себя импульс не несущий никакой информации.

Start: При бифазном кодировании требуется подать сигнал Start, так как невозможно начать передачу пакета с паузы.

Toggle: Это бит, который меняет своё значение при каждом новом нажатии на кнопку, используется в протоколах RS5, RS5X, RS6 (Philips), где пакеты повторов полностью повторяют данные информационного пакета. Таким образом принимающее устройство может отличить удержание кнопки от её повторного нажатия.

Примеры:

Проверка наличия данных поступивших с ИК-пульта, осуществляется функцией check(). Эта функция реагирует на нажатие кнопок ИК-пульта, но если её вызывать с параметром `true`, то она будет реагировать и на удержание кнопок.

Чтение данных с любого пульта, реагируем только на нажатие кнопок:

```
#include <iarduino_IR_RX.h>                     // Подключаем библиотеку для работы с ИК-приёмником
iarduino_IR_RX IR(7);                           // Объявляем объект IR, с указанием вывода к которому подключён ИК-приёмник
```

```
void setup(){
  Serial.begin(9600);                          // Инициируем передачу данных в монитор последовательного порта, на скорости
  IR.begin();                                  // Инициируем работу с ИК-приёмником
}
void loop(){
  if(IR.check()){                              // Если в буфере имеются данные, принятые с пульта (была нажата кнопка)
    Serial.println(IR.data, HEX);              // Выводим код нажатой кнопки
    Serial.println(IR.length   );              // Выводим количество бит в коде
  }
}
```

В данном скетче функция check() вызывается без аргументов, значит и реагирует она только на нажатия кнопок ИК-пульта.

Чтение данных с любого пульта, реагируем на удержание кнопок:

```
#include <iarduino_IR_RX.h>                    // Подключаем библиотеку для работы с ИК-приёмником
iarduino_IR_RX IR(6);                          // Объявляем объект IR, с указанием вывода к которому подключён ИК-приёмник
void setup(){
  Serial.begin(9600);                          // Инициируем передачу данных в монитор последовательного порта, на скорости
  IR.begin();                                  // Инициируем работу с ИК-приёмником
}
void loop(){
  if(IR.check(true)){                          // Если в буфере имеются данные, принятые с пульта (удерживается кнопка)
    Serial.println(IR.data, HEX);              // Выводим код нажатой кнопки
    Serial.println(IR.length   );              // Выводим количество бит в коде
  }
}
```

В данном скетче функция check() вызывается с параметром `true`, значит и реагирует она как на нажатия, так и на удержания кнопок ИК-пульта.

Чтение данных с любого пульта, с указанием как реагировать на какие кнопки.

```
#include <iarduino_IR_RX.h>                           // Подключаем библиотеку для работы с ИК-приёмником
iarduino_IR_RX IR(6);                                 // Объявляем объект IR, с указанием вывода к которому подключён ИК-приёмн
                                                      //
bool flgKey1 = false; uint32_t codKey1 = 0xFF30CF;    // Определяем флаг нажатия и код кнопки 1
bool flgKey2 = false; uint32_t codKey2 = 0xFF18E7;    // Определяем флаг нажатия и код кнопки 2
bool flgKey3 = false; uint32_t codKey3 = 0xFF7A85;    // Определяем флаг нажатия и код кнопки 3
bool flgKey  = false; uint32_t tmrKey  = 0;           // Определяем флаг разрещающий вывод данных в монитор и время последнего
                                                      //
void setup(){                                         //
  Serial.begin(9600);                                 // Инициируем передачу данных в монитор последовательного порта, на скорс
  IR.begin();                                         // Инициируем работу с ИК-приёмником
}                                                     //
                                                      //
void loop(){                                          //
  if(IR.check(true)){                                 // Если в буфере имеются данные, принятые с пульта (удерживается кнопка),
    if(millis()-200 > tmrKey){                        // Если с последней поступившей команды прошло более 200 мс, то
      flgKey1=false;                                  // Считаем что кнопка 1 не удерживается
      flgKey2=false;                                  // Считаем что кнопка 2 не удерживается
      flgKey3=false;                                  // Считаем что кнопка 3 не удерживается
    } tmrKey = millis();              flgKey=true;    // Сохраняем время последней реакции на пульт и азрешаем вывод данных
    if(IR.data==codKey1){ if(flgKey1){flgKey=false;} flgKey1=true; }else{flgKey1=false;} // Запрещаем вывод данных кнопки 1 пр
    if(IR.data==codKey2){ if(flgKey2){flgKey=false;} flgKey2=true; }else{flgKey2=false;} // Запрещаем вывод данных кнопки 2 пр
    if(IR.data==codKey3){ if(flgKey3){flgKey=false;} flgKey3=true; }else{flgKey3=false;} // Запрещаем вывод данных кнопки 3 пр
    if(flgKey){                                       // Если вывод данных разрешен, то ...
      Serial.println(IR.data, HEX);                   // Выводим код нажатой кнопки
      Serial.println(IR.length   );                   // Выводим количество бит в коде
    }                                                 //
  }                                                   //
}                                                     //
```

В данном скетче функция check() вызывается с параметром `true`, значит она реагирует как на нажатия, так и на удержания кнопок ИК-пульта. Но вывод данных в монитор последовательного порта осуществляется только при установленном флаге `flgKey`, который сбрасывается при удержании кнопок с кодами `0xFF30CF`, `0xFF18E7` и `0xFF7A85`. Получается что на 3 кнопки скетч реагирует только при нажатии, а на остальные кнопки, как на нажатие, так и на удержание.

Чтение данных только с тех пультов, которые работают по указанному протоколу:

```
#include <iarduino_IR_RX.h>                     // Подключаем библиотеку для работы с ИК-приёмником
iarduino_IR_RX IR(5);                           // Объявляем объект IR, с указанием вывода к которому подключён ИК-приёмник
void setup(){
  Serial.begin(9600);                           // Инициируем передачу данных в монитор последовательного порта, на скорости
  IR.begin();                                   // Инициируем работу с ИК-приёмником
  IR.protocol("Ae``QtWL@L`|LJ`G@@@@@BPBp");     // Указываем протокол передачи данных, на который следует реагировать
}
void loop(){
  if(IR.check(true)){                           // Если в буфере имеются данные, принятые с пульта (удерживается кнопка)
    Serial.println(IR.data, HEX);               // Выводим код нажатой кнопки
    Serial.println(IR.length   );               // Выводим количество бит в коде
  }
}
```

В данном скетче, в коде setup(), указан протокол передачи данных, который редко совпадает у разных производителей ИК-пультов. Значит функция check() в коде loop() будет реагировать только на те ИК-пульты, которые поддерживают указанный протокол.

Получение протокола передачи данных и типа кодировки:

```
#include <iarduino_IR_RX.h>                     // Подключаем библиотеку для работы с ИК-приёмником
iarduino_IR_RX IR(4);                           // Объявляем объект IR, с указанием вывода к которому подключён ИК-приёмник
void setup(){
  Serial.begin(9600);                           // Инициируем передачу данных в монитор последовательного порта, на скорости
  IR.begin();                                   // Инициируем работу с ИК-приёмником
```

```
}
void loop(){
  if(IR.check()){                          // Если в буфере имеются данные, принятые с пульта (была нажата кнопка)
    Serial.println(IR.protocol());         // Выводим строку протокола передачи данных
  }
}
```

В данном примере описано как получить протокол передачи данных ИК-пультов. В статье Wiki ИК-передатчик, описано, как передавать коды кнопок по указанному протоколу.

Таким образом, можно создать скетч ИК-передатчика для имитации сигналов различных ИК-пультов. В результате, устройства будут реагировать на ИК-передатчик, как на собственный ИК-пульт.

Описание основных функций библиотеки:

Подключение библиотеки:

```
#include <iarduino_IR_RX.h>                  // Подключаем библиотеку, для работы с ИК-приёмником.
iarduino_IR_RX IR(№_ВЫВОДА [, ИНВЕРСИЯ] ); // Объявляем объект IR, с указанием номера вывода, к которому подключён ИК-приёмник
                                             // Вторым параметром, типа bool, можно указать, что данные с приёмника являются инв
```

Функция begin();

- Назначение: инициализация работы с ИК-приёмником
- Синтаксис: begin();
- Параметры: Нет.
- Возвращаемые значения: Нет.
- Примечание: Вызывается 1 раз в коде setup.
- Пример:

```
```

```
IR.begin(); // Инициируем работу с ИК-приёмником
```

Функция check();

- Назначение: Проверка наличия принятых с пульта данных.
- Синтаксис: check([УДЕРЖАНИЕ]);
- Параметры:
  - УДЕРЖАНИЕ - необязательный параметр, типа bool - указывающий что необходимо реагировать на удержание кнопок пульта.
- Возвращаемые значения: bool - приняты или нет, данные с пульта.
- Примечание: Если функция вызвана без параметра, или он равен false, то функция будет реагировать только на сигналы с пульта при нажатии его кнопок, а если указать true, то функция будет реагировать, как на нажатие, так и на удержание кнопок пульта.
- Пример:

```
if( IR.check()     ){ ... ;} // Если приняты данные с пульта, при нажатии его кнопки
if( IR.check(true) ){ ... ;} // Если принимаются данные с пульта, при удержании кнопки
```

Функция protocol();

- Назначение: Получение, установка или сброс протокола передачи данных.
- Синтаксис: protocol([ПАРАМЕТР]);
- Получение протокола: Если функция вызвана без параметра, то она вернёт строку из 25 символов + символ конца строки. Биты данной строки, несут информацию о типе протокола передачи данных пульта, данные которого были приняты последними. Данную строку можно использовать для установки протокола ИК-передатчику, или ИК-приёмнику (см.ниже).
- Установка протокола: Если функция вызвана с параметром в виде строки из 25 символов протокола + символ конца строки, то после этого, функция chek(), будет реагировать только на пульты, соответствующие указанному протоколу передачи данных.
- Сброс протокола: Если функция вызвана с параметром IR_CLEAN, то функция chek() опять станет реагировать на сигналы с любых пультов.
- Получение параметров протокола: Если функция вызвана с параметром int, от 0 до 17, то она вернёт не строку протокола, а значение типа int с одним из параметров протокола передачи данных пульта, данные которого были приняты последними:
  - 0 - тип кодировки:
    - IR_UNDEFINED - тип кодировки не определён;

        - IR_PAUSE_LENGTH - кодирование длинной паузы;
        - IR_PULSE_LENGTH - кодирование длинной (шириной) импульса (ШИМ);
        - IR_BIPHASIC - бифазное кодирование;
        - IR_BIPHASIC_INV - бифазное кодирование с инверсными битами;
        - IR_NRC - пакеты повтора идентичны, а первый и последний пакеты специальные;
        - IR_RS5 - кодировка PHILIPS с битом toggle;
        - IR_RS5X - кодировка PHILIPS с битом toggle;
        - IR_RS6 - кодировка PHILIPS с битом toggle.
    - 1 - несущая частота передачи данных (в кГц);
    - 2 - заявленное количество информационных бит в 1 пакете;
    - 3 - заявленное количество информационных бит в пакете повтора;
    - 4 - длительность паузы между пакетами (в мс);
    - 5 - длительность импульса в стартовом бите (в мкс);
    - 6 - длительность паузы в стартовом бите (в мкс);
    - 7 - длительность импульса в стоповом бите (в мкс);
    - 8 - длительность паузы в стоповом бите (в мкс);
    - 9 - длительность импульса в бите рестарт или toggle (в мкс);
    - 10 - длительность паузы в бите рестарт или toggle (в мкс);
    - 11 - позиция бита рестарт или toggle в пакете (№ бита);
    - 12 - максимальная длительность импульса в информационных битах (в мкс);
    - 13 - минимальная длительность импульса в информационных битах (в мкс);
    - 14 - максимальная длительность паузы в информационных битах (в мкс);
    - 15 - минимальная длительность паузы в информационных битах (в мкс);
    - 16 - флаг наличия стартового бита (true/false);
    - 17 - флаг наличия стопового бита (true/false);
    - 18 - флаг наличия бита рестарт или toggle (true/false);
    - 19 - тип пакета повтора (0-нет, 1-с инверсными битами, 2-идентичен информационному, 3-уникален);
- Возвращаемые значения: Зависят от наличия и типа параметра.

- Примечание: Если ранее был установлен протокол, то попытка получения протокола, или параметров протокола, вернёт значения установленного ранее протокола, а не протокола передачи данных пульта, данные которого были приняты последними.
- Пример:

```
                              IR.protocol("AeQQV~zK]Kp^KJp[@@@@@@@Bp"); // Устанавливаем протокол. Теперь приёмник будет г
                              IR.protocol(IR_CLEAN);                   // Сбрасываем ранее установленный протокол. Теперь
if( IR.check() ){ Serial.println( IR.protocol()   ); }                 // Получаем протокол. Как только приёмник получит
if( IR.check() ){ Serial.println( IR.protokol(12) ); }                 // Получаем один из параметров протокола. Как толь
```

Переменная data

- Значение: Возвращает код кнопки, принятый с пульта;
- Тип данных: uint32_t.

```
if( IR.check() ){ Serial.println( IR.data ); } // Выводим код нажатой кнопки, если он принят
```

Переменная length

- Значение: Возвращает размер кода кнопки, в битах;
- Тип данных: uint8_t.

```
if( IR.check() ){ Serial.println( IR.length ); } // Выводим размер кода нажатой кнопки, если он принят
```

Переменная key_press

- Значение: Возвращает флаг, указывающий на то, что кнопка пульта нажимается а не удерживается;
- Тип данных: bool.

```
if( IR.check(true) ){
  if( IR.key_press ){Serial.println( "PRESS" );} // Текст будет выведен 1 раз, когда кнопка нажимается
```

```
  else              {Serial.println( "HOLD " );} // Текст будет выводиться постоянно, пока кнопка удерживается
}
```

Применение:

- управление роботами, движущимися, летающими и плавающими моделями, бытовой и специализированной техникой.
- включение/выключение освещения, обогрева, вентиляции, полива и т.д.
- открывание/закрывание дверей, жалюзи, мансардных окон, форточек и т.д.