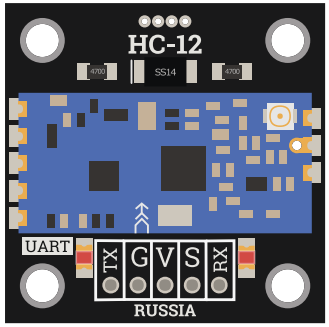
Радио модуль 433.4МГц HC-12 (Trema-модуль V2.0)



Общие сведения:

Trema-модуль HC-12 - это беспроводной полудуплексный модуль UART позволяющий передавать и принимать данные в диапазоне частот от 433,4 МГц до 473МГц на скорости от 1200 до 115200 бод.

Спецификация:

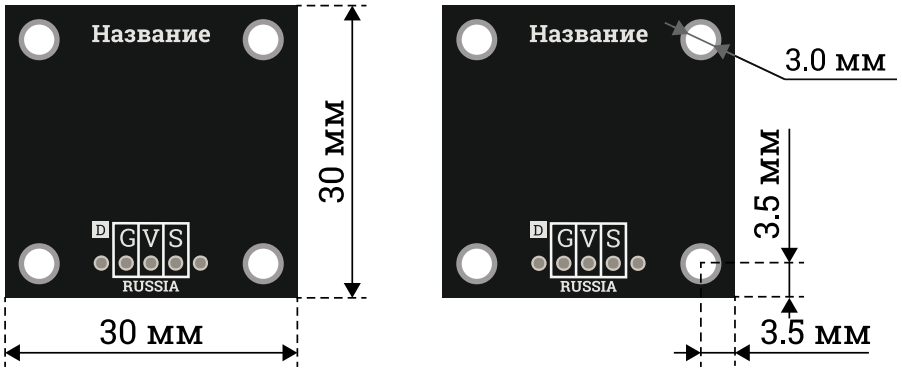
Спецификация модуля:

- Напряжение питания: 3,3 ... 5 В
- Потребляемый ток при передаче данных: 100 мА
- Диапазон частот: 433,4 ... 473 МГц
- Мощность передатчика: до +20 дБм
- Чувствительность приёмника: до -117 дБм
- Дальность связи: до 1 КМ на открытой местности (на скорости 1200 бод)
- Интерфейс: UART
- Поддерживаемые скорости UART: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 бит/сек
- Установки по умолчанию: FU3, 9600 бод, 8-N-1, CH001
- Рабочая температура: -25 ... +75°C

Спецификация антенны:

- Рабочая частота: 433 МГц
- Коэффициент стоячей волны: <=1.5 на 433МГц
- Импеданс: 50Ω
- Длина с коаксиальным шнуром: 10 ... 15 см
- Разъём: IPEX

Все модули линейки "Trema" выполнены в одном формате



Подключение:

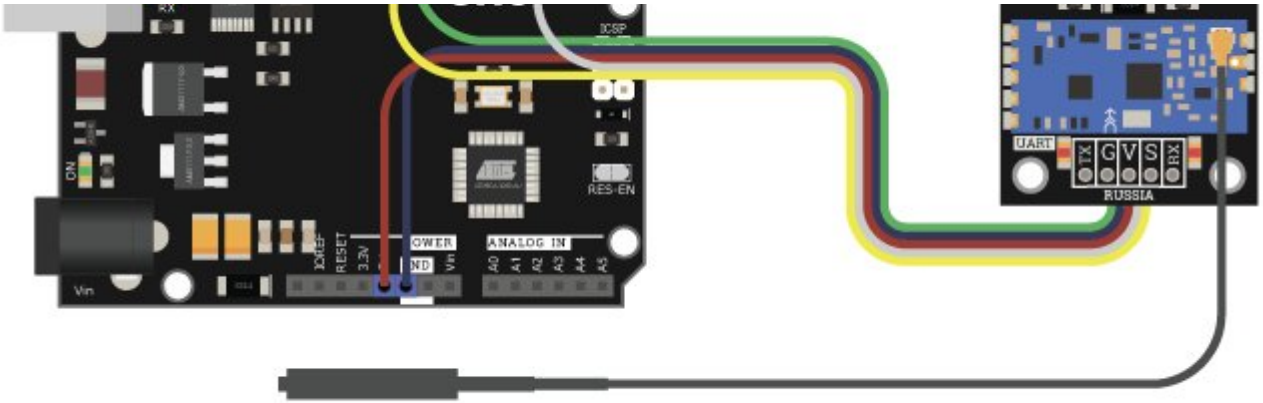
Trema-модуль HC-12 подключается к Arduino по шине UART (можно использовать как аппаратную так и программную шину).

- Вывод модуля **TX** подключается к аппаратному (фиксированному) или программному (назначенному) выводу **RX** Arduino. Это линия шины UART для передачи данных от модуля к Arduino.
- Вывод модуля **RX** подключается к аппаратному (фиксированному) или программному (назначенному) выводу **TX** Arduino. Это линия шины UART для передачи данных в модуль от Arduino.
- Вывод модуля **S** подключается к любому выводу Arduino номер которого указывается в скетче. Это линия перевода модуля в режим AT-команд. Модуль будет воспринимать AT-команды, только во время подачи на этот вывод низкого уровня.

Модуль удобно подключать 4 способами, в зависимости от ситуации:

Способ - 1 : Используя провода, Piranha UNO и программный UART

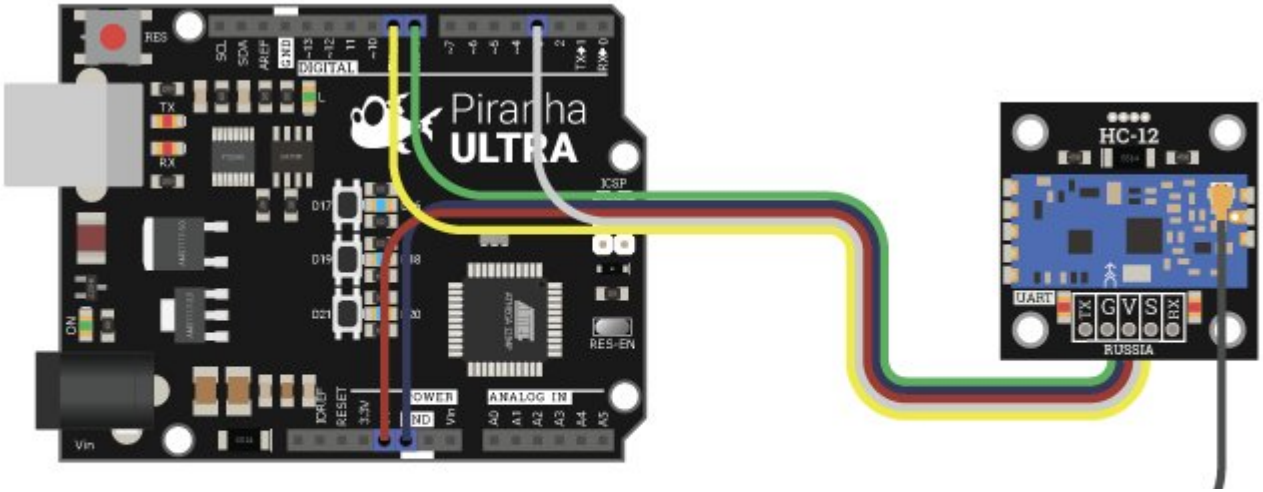
Используя провода «Папа — Мама», подключаем напрямую к контроллеру Piranha UNO.



Способ - 2 : Используя провода, Piranha ULTRA и аппаратный UART

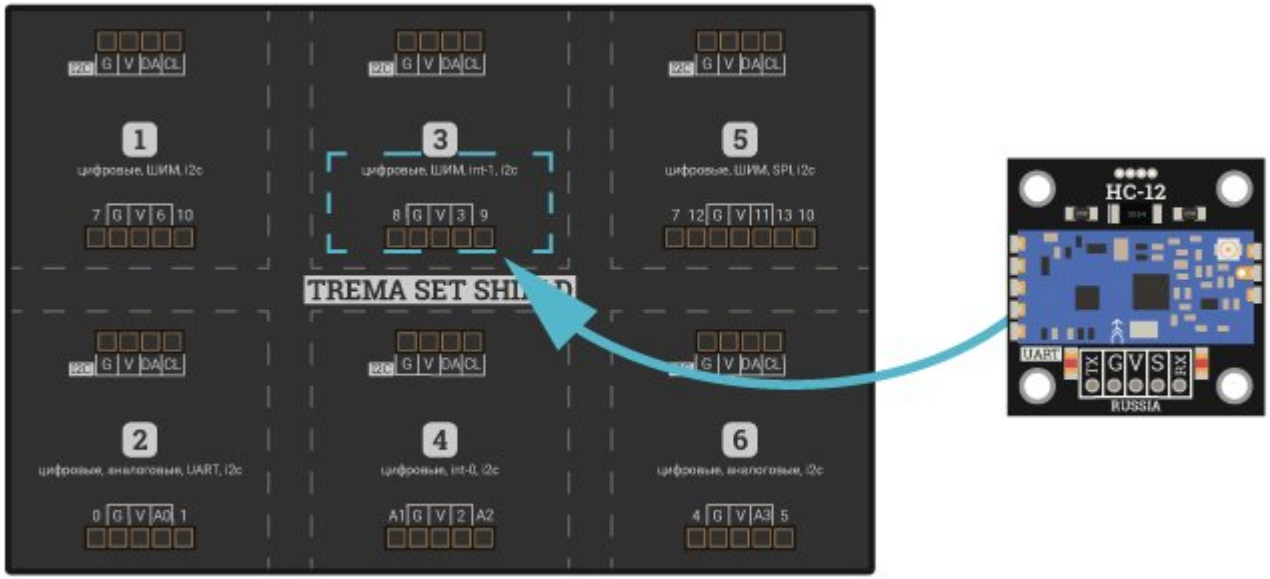
Используя провода «Папа — Мама», подключаем напрямую к контроллеру Piranha ULTRA.

С данным подключением будет использоваться второй аппаратный UART на Piranha ULTRA. Стоит заметить, что программный порт на UNO безошибочно работает на скорости до 57600 бод, в то время как аппаратный без проблем может работать на скорости 115200, вдвое большей.



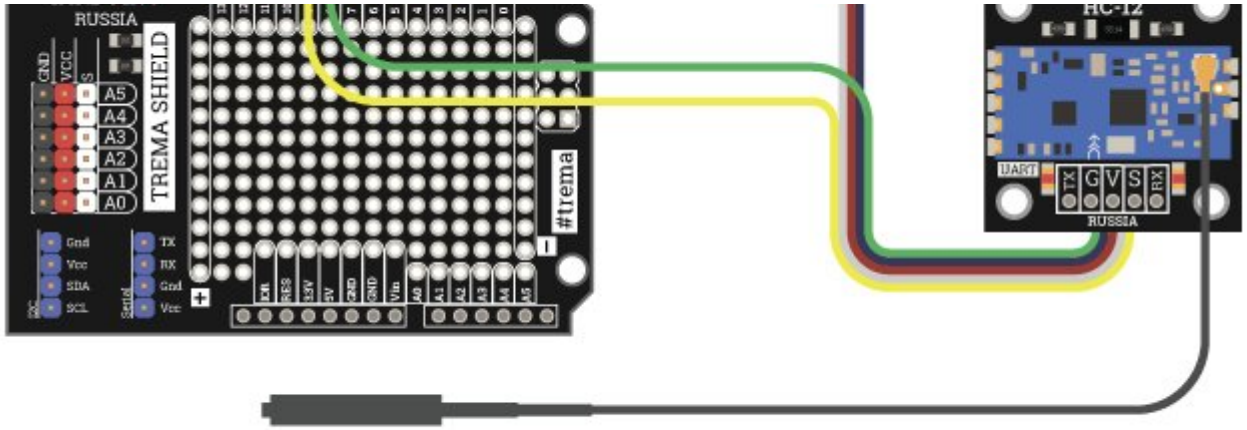
Способ - 3 : Используя Trema Set Shield

В примерах ниже мы будем использовать программный UART на 8 и 9 цифровых выводах, поэтому для удобства подключения можно установить модуль в 3-ю ячейку Trema Set Shield. Так же на этих выводах находиться аппаратный порт Piranha ULTRA, что ещё больше упрощает работу с модулем.



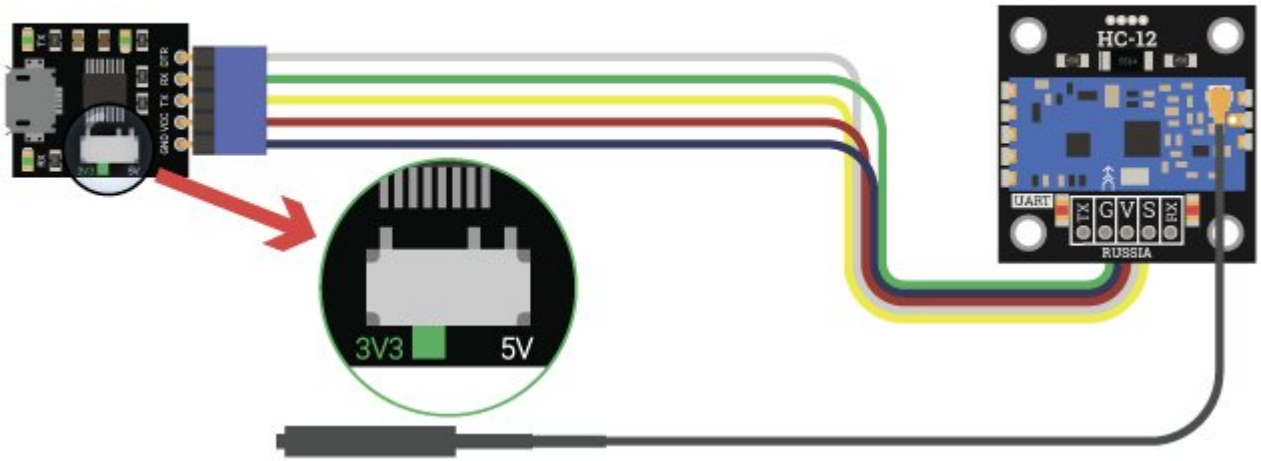
Способ - 4 : Используя проводной шлейф и Shield

Используя 2-х и 3-х проводные шлейфы, к Trema Shield, Trema-Power Shield, Motor Shield, Trema Shield NANO и тд.



Способ - 5 : Напрямую к ПК через USB-UART

Используя USB-UART преобразователь Piranha можно подключить напрямую к ПК. При этом можно получать данные и отправлять команды на Arduino по радиоканалу, как если бы Arduino была подключена по проводу. Пример скетча для данного подключения.



Питание:

Входное напряжение питания 3,3 или 5 В постоянного тока, подаётся на выводы V и G модуля. Для работы от 3,3 В джампер под пайку сзади модуля должен быть запаян.

Подробнее о модуле:

Модуль работает в диапазоне частот от 433,4 МГц до 473МГц, имеет возможность выбора каналов от 1 до 100, возможность выбора восьми вариантов мощности передатчика от -1 дБм до 20 дБм при этом расстояние передачи может достигать 1000 метров на открытой местности при скорости передачи данных в эфире 5000 бит/с.

Модуль является полудуплексным, это означает что он может принимать или отправлять данные.

Режимы работы модуля

Модуль имеет четыре режима работы: FU1, FU2, FU3, FU4 и режим приём АТ команд. Для перехода в режим приёма АТ команд вывод S модуля необходимо прижать к земле (режим логического "0"). Режимы FU отвечают за приём и передачу данных последовательного порта.

Режим FU1 - режим относительно низкого энергопотребления, с током покоя 3,6 мА. В этом режиме можно выбирать скорость передачи

данных по проводу arduino-модуль, но скорость в эфире постоянна и равна 250`000 бит/сек.

Режим FU2 - режим очень низкого энергопотребления, с током покоя 80 мкА. В этом режиме скорости передачи данных могут быть 1200, 2400 и 4800 бит/сек. Скорость в эфире фиксирована и равна 250`000 бит/сек. В этом режиме интервал между пакетами данных должен быть больше 1 секунды, иначе данные будут потеряны.

Режим FU3 - режим по умолчанию, потребляемый ток покоя 16 мА, скорость передачи данных в эфире автоматически переключается в зависимости от скорости проводного последовательного порта по следующей таблице:

Скорость передачи по проводу, бит/сек	1`200/2`400	4`800/9`600	19`200/38`400	57`600/115`200
Скорость передачи в эфире, бит/сек	5`000	15`000	58`000	236`000

Чем ниже скорость передачи по проводу — тем больше расстояние передачи модуля.

Режим FU4 - режим максимальной дальности, ток покоя 16 мА, поддерживает только одну скорость передачи равную 1200 бит/сек при этом скорость передачи в эфире уменьшается до 500 бит/сек, для наибольшей дальности связи. В этом режиме можно передавать не больше 60 байтов за раз, при этом интервал между пакетами должен превышать 2 секунды во избежание потерь данных.

Таблица режимов

Режимы:	**FU1**	**FU2**	**FU3**	**FU4**	**Комментарий**
Ток покоя	3,6 мА	80 мкА	16 мА	16 мА	Среднее значение
Задержка передачи	15-25 мс	500 мс	4-80 мс	1000 мс	Передача одного байта

Режимы:	FU1	FU2	FU3	FU4	Комментарий
Дальность передачи, W = 20 дБм	до 100 м	до 100 м	до 600м на 9600 бит/сек; до 1000 метров на 2400 бит/сек	до 1800 метров на 1200 бит/сек	Прямая видимость и идеальные условия, зависит от антенны.

Для того чтобы модули могли связаться друг с другом у них должен быть выставлен одинаковый режим работы и скорость.

Чувствительность приёмника

Чувствительность приёмника модуля зависит от скорости передачи данных по следующей таблице:

Скорость передачи в эфире	5`000 бит/сек	15`000 бит/сек	58`000 бит/сек	236`000 бит/сек
Чувствительность приёмника	-117 дБм	-112 дБм	-107 дБм	-100 дБм

AT команды модуля:

Для перехода в режим AT команд необходимо притянуть вывод **S** модуля к земле (логическому нулю). Для входа в режим AT команд модулю необходимо 40 миллисекунд. Для выхода из режима AT команд модулю необходимо 80 миллисекунд.

- **AT**

тестовая команда, при удачном выполнении модуль возвращает "OK"

- **AT+Bxxxx**

Команда установки скорости обмена данными с микроконтроллером, где "xxxx" это скорость. Может быть 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200. Пример:

AT+B4800, при удачном выполнении модуль вернёт "OK+B4800"

- **AT+Cxxx**

Команда смены канала, "xxx" - номер канала (001-100). Значение по умолчанию 001. Пример: "AT+C042" сменит канал на 42 и рабочая частота станет 449,8 МГц. Приёмник модуля очень чувствительный и при использовании на скорости больше 9600 бит в секунду или если модули находятся на небольшом расстоянии друг от друга каналы лучше использовать через один, так как возможны перекрёстные помехи на соседних каналах.

- **AT+FUx**

Команда смены режима работы модуля. По умолчанию FU3. Для передачи данных модули должны находится в одном режиме.

- **AT+ Px**

Команда установки уровня мощности передатчика, "x" - от 1 до 8 по следующей таблице:

X	1	2	3	4	5	6	7	8
Мощность (дБм)	-1	2	5	8	11	14	17	20

Чем выше мощность, тем больше расстояние излучения модуля. При уменьшении мощности на 6 дБ рабочее расстояние уменьшается в 2 раза. Значение по умолчанию - 8.

- **AT+Ry**

Запрос одного параметра модуля, где "y": **B** - запросить скорость передачи данных, **C** - запросить канал, **F** - запросить режим работы и **P** - запросить мощность передатчика. Например: при запросе"AT+RB" модуль ответит "OK+B9600".

- **AT+RX**

Запросить все параметры модуля.

- **AT+Uxxx**

  Установка параметров передачи данных. Биты данных, бит чётности, стоп-бит. Например, чтобы установить 8 бит данных, бит чётности и 1 стоп-бит посылаем модулю "AT+U8E1", модуль ответит "OK".

- **AT+V**

  Запросить версию прошивки. При запросе "AT+V" модуль ответит "www.hc01.com HC-12 v2.6"

- **AT+SLEEP**

  Перевод в режим сна. Потребляемый ток в данном режиме 22 микроампера. Передача данных в данном режиме невозможна

- **AT+DEFAULT**

  Сброс параметров модуля на значения по умолчанию

- **AT+UPDATE**

  Установка модуля в режим ожидания обновления прошивки

Примеры:

Пример тестирования модуля и сброса настроек на заводские.

Данный скетч сбрасывает настройки модуля на заводские и включает встроенный светодиод Arduino если получен положительный ответ от модуля.

```
// Строки с * необходимо удалить, если Вы используете Piranha Ultra
#include <SoftwareSerial.h>             // * Подключаем библиотеку программного последовательного порта
                                        //
#define RX       8                      // * Определяем вывод RX (TX на модуле)
#define TX       9                      // * Определяем вывод TX (RX на модуле)
```

```
SoftwareSerial Serial1(RX,TX);          // * Создаём объект программного последовательного порта
                                        //
#define S        3                      // Определяем вывод S
#define RESPONSE "OK+DEFAULT"           // Определяем ожидаемый ответ модуля
#define AT       "AT+DEFAULT"           // Определяем AT команду
String ACK = "";                        // Создаём пустую строку для хранения ответа модуля
                                        //
void setup() {                          //
  Serial1.begin(9600);                  // Инициируем программный последовательный порт
  pinMode(S, OUTPUT);                   // Переводим вывод S модуля в режим выход
  digitalWrite(S, LOW);                 // Назначаем выводу уровень логического нуля
  delay(40);                            // Ждём пока модуль войдёт в режим AT команд
}                                       //
                                        //
void loop() {                           //
  Serial1.println(AT);                  // Передаём модулю команду сброса настроек
  while(Serial1.available()){           // Пока есть непрочитанные данные в буфере ПП
    ACK = Serial1.readStringUntil('\r'); // Записываем ответ до знака возврата каретки в строку
      if (ACK == RESPONSE){             // Если значение строки совпало с предопределённым
        pinMode(LED_BUILTIN, OUTPUT);   // Переводим вывод встроенного светодиода Arduino в режим выхода
        digitalWrite(LED_BUILTIN, HIGH); // Назначаем выводу уровень логической единицы
        delay(200);                     // Даём светодиоду светиться 200 мс
      } else {                          // Если значение строки не совпало с предопределённым
        pinMode(LED_BUILTIN, INPUT);    // Переводим вывод в режим входа
      }                                 //
    }                                   //
  delay(100);                           // Ждём 100 мс в главном цикле скетча
}                                       //
```

Перевод модуля в режим ручного ввода AT команд

Пример входа в режим ввода AT команд в мониторе порта Arduino. После каждой команды необходим символ новой линии или возврата каретки. Скорость передачи должна быть выставлена на 9600 бод.

```
// Строки с * необходимо удалить, если Вы используете Piranha Ultra
#include <SoftwareSerial.h>            // * Подключаем библиотеку программного последовательного порта
                                       //
#define RX 8                           // * Определяем вывод RX (TX на модуле)
#define TX 9                           // * Определяем вывод TX (RX на модуле)
SoftwareSerial Serial1(RX,TX);         // * Создаём объект программного последовательного порта
                                       //
#define S  3                           // Определяем вывод S
                                       //
void setup() {                         //
  Serial.begin(9600);                  // Инициируем аппаратный последовательный порт
  Serial1.begin(9600);                 // Инициируем программный последовательный порт
  pinMode(S, OUTPUT);                  // Переводим вывод S модуля в режим выход
  digitalWrite(S, LOW);                // Назначаем выводу уровень логического нуля
  delay(40);                           // Ждём пока модуль войдёт в режим AT команд
}                                      //
                                       //
void loop() {                          //
  if(Serial1.available()){             // Если в буфере программного последовательного порта есть данные
    Serial.write(Serial1.read());      // Перенаправляем их в аппаратный последовательный порт
    }                                  //
  if(Serial.available()){              // Если в буфере аппаратного последовательного порта есть данные
    Serial1.write(Serial.read());      // Перенаправляем их в программный последовательный порт
  }                                    //
}                                      //
```

Подключение модуля через USB-UART преобразователь

Пример дистанционной отправки команд Arduino, при этом один модуль подключён к ПК через USB-UART преобразователь, второй подключён к Arduino на которую загружен этот скетч. При отправке ключевого слова "toggle" из монитора последовательного порта встроенный светодиод платы будет менять своё состояние.

```
bool led_state = false;                                 // Создаём переменную для хранения состояния светодиода
void setup() {                                          //
  Serial.begin(9600);                                   // Инициируем последовательный порт на скорости 9600 бод
  pinMode(LED_BUILTIN, OUTPUT);                         // Устанавливаем вывод встроенного светодиода в режим выход
}                                                       //
                                                        //
void loop() {                                           //
  if (Serial.available()){                              // Если в буфере последовательного порта есть данные
    String command = Serial.readStringUntil('\n');      // Записываем в строку до символа новой строки
    command.trim();                                     // Удаляем из строки пустые символы
    if (command == "toggle")                            // Если строка равна кодовому слову
      led_state = !led_state;                           // Меняем состояние светодиода на противоположное
      String state;                                     // Создаём строку для вывода в последовательный порт
      if (led_state) state = "on";                      // Записываем состояние светодиода
      else state = "off";                               // в ранее созданную строку
      Serial.println("Led is " + state);                // Выводим в последовательный порт информацию о состоянии светодиода
    }                                                   //
  digitalWrite(LED_BUILTIN, led_state);                 // Записываем состояние светодиода в вывод
}                                                       //
```

Соединение двух Arduino через беспроводной UART

Данный пример передаёт "как есть" всё что он получает от модуля в серийный порт Arduino. Пример можно использовать для проверки связи между модулями и для обмена данными между двумя Arduino через беспроводной UART. Для соединения двух Arduino необходимо загрузить данный скетч в обе.

```
// Строки с * необходимо удалить, если Вы используете Piranha Ultra
#include <SoftwareSerial.h>             // * Подключаем библиотеку программного последовательного порта
                                        //
#define RX 8                            // * Определяем вывод RX (TX на модуле)
#define TX 9                            // * Определяем вывод TX (RX на модуле)
                                        //
```

```
SoftwareSerial Serial1(RX,TX);          // * Создаём объект программного последовательного порта
                                        //
void setup() {                          //
  Serial.begin(9600);                   // Инициируем аппаратный последовательный порт
  Serial1.begin(9600);                  // Инициируем программный последовательный порт
}                                       //
                                        //
void loop() {                           //
  if(Serial1.available()){              // Если в буфере программного последовательного порта есть данные
    Serial.write(Serial1.read());       // Перенаправляем их в аппаратный последовательный порт
    }                                   //
  if(Serial.available()){               // Если в буфере аппаратного последовательного порта есть данные
    Serial1.write(Serial.read());       // Перенаправляем их в программный последовательный порт
  }                                     //
}                                       //
```

Передача массива и включение светодиода на расстоянии

В примере используются два модуля и две платы Arduino. Модули должны быть установленны на один канал и режим работы (см. первый пример).

Пример передачи массива. Включаем светодиод значением первого байта. Arduino с передатчиком посылает заголовок и массив; Arduino с приёмником ждёт заголовок и принимает массив. Тип массива может быть только **byte**, **char**, **int8_t** или **uint8_t,**

Скетч передатчика:

```
/*
 * Скетч для передатчика
 */
// Строки с * необходимо удалить, если Вы используете Piranha Ultra
#include <SoftwareSerial.h>         // * Подключаем библиотеку программного последовательного порта
#define RX 8                        // * Определяем вывод TX программного последовательного порта
```

```
#define TX 9                          // * Определяем вывод RX программного последовательного порта
                                      //
SoftwareSerial Serial1(RX,TX);        // * Создаём программный последовательный порт
                                      //
byte data[5];                         // Объявляем массив для приёма и хранения данных
                                      //
void setup() {                        //
  Serial1.begin(9600);                // Инициируем последовательный порт на скорости 9600 бод
}                                     //
                                      //
void loop() {                         //
  data[0] = 0;                        // Записываем ноль (ВЫКЛ) в первый байт массива
  Serial1.write(0xAA);                // Посылаем заголовок пакета
  Serial1.write(data, sizeof(data));// Посылаем массив данных
  Serial1.write('\n');                // Посылаем символ новой строки
  delay(500);                         // Ждём полсекунды
  data[0] = 1;                        // Записываем единицу (ВКЛ) в первый байт массива
  Serial1.write(0xAA);                // Посылаем заголовок пакета
  Serial1.write(data, sizeof(data));// Посылаем массив данных
  Serial1.write('\n');                // Посылаем символ новой строки
  delay(500);                         // Ждём полсекунды
}                                     //
```

Скетч приёмника:

Скетч принимает массив из скетча выше и включает встроенный светодиод по значению первого байта

```
/*
 * Скетч для приёмника
 */
// Строки с * необходимо удалить, если Вы используете Piranha Ultra
#include <SoftwareSerial.h>                     // * Подключаем библиотеку программного последовательного порта
```

```
#define RX 8                                    // * Определяем вывод TX программного последовательного порта
#define TX 9                                    // * Определяем вывод RX программного последовательного порта
                                                //
SoftwareSerial Serial1(RX,TX);                  // * Создаём программный последовательный порт
                                                //
byte data[5];                                   // Объявляем массив для приёма и хранения данных
                                                //
void setup() {                                  //
  Serial1.begin(9600);                          // Инициируем программный последовательный порт на скорости 9600 бод
}                                               //
                                                //
void loop() {                                   //
// Если в буфере порта есть данные и прочитанный байт равен заголовку пакета
  if (Serial1.available() && Serial1.read() == 0xAA){
    // Читаем строку из последовательного порта и конвертируем её в массив байт data
    Serial1.readStringUntil('\n').toCharArray(data, sizeof(data));
    digitalWrite(LED_BUILTIN, data[0]);         // Включаем или выключаем встроенный светодиод, в зависимости от полученных д
  }                                             //
}                                               //
```

Пример переключения каналов модуля.

Скетч передатчика:

Пример смены канала передатчика. Передаётся байт на 1-м канале, затем канал переключается и передаётся байт на 3-м канале.

```
/*
 * Скетч для передатчика
 */

// Строки с * необходимо удалить, если Вы используете Piranha Ultra
#include <SoftwareSerial.h>             // * Подключаем библиотеку программного последовательного порта
```

```
                                        //
#define RX 8                            // * Определяем вывод RX (TX на модуле)
#define TX 9                            // * Определяем вывод TX (RX на модуле)
SoftwareSerial Serial1(RX,TX);          // * Создаём объект программного последовательного порта
                                        //
#define S  3                            // Определяем вывод S
                                        //
// Функция смены канала                 //
void channel_change(int _channel){      // Функция принимает целое число - номер канала на который переключить модуль
  String ch = "";                       // Создаём пустую строку для форматирования номера канала
  if (_channel < 100) ch += "0";        // Если значение меньше 100, добавляем ноль в начало
  if (_channel < 10) ch += "0";         // Если меньше 10, добавляем ещё один ноль
  ch += String(_channel);               // Добавляем к строке номер канала
  String command = "AT+C" + ch;         // Создаём строку команды и записываем в неё команду и номер канала
  pinMode(S, OUTPUT);                   // Переводим вывод S модуля в режим выход
  digitalWrite(S, LOW);                 // Назначаем выводу уровень логического нуля
  delay(40);                            // Ждём пока модуль войдёт в режим AT команд
  Serial1.println(command);             // Посылаем AT команду модулю
  pinMode(S, INPUT);                    // Выходим из режима AT команд, переведя вывод S в режим входа
  delay(80);                            // Ждём пока модуль выйдет из режима AT команд
}                                       //
                                        //
void setup() {                          //
  Serial1.begin(9600);                  // Инициируем последовательный порт
}                                       //
                                        //
void loop() {                           //
  channel_change(1);                    // Меняем канал на первый
  Serial1.write(0xAA);                  // Посылаем байт со значением 170
  delay(400);                           // Ждём 400 мс
  channel_change(3);                    // меняем канал на третий
  Serial1.write(0xAA);                  // Посылаем байт со значением 170
  delay(400);                           // Ждём 400 мс
```

```
  channel_change(1);                    // Меняем канал на первый
  Serial1.write(0xAD);                  // Посылаем байт со значением 173
  delay(400);                           // Ждём 400 мс
  channel_change(3);                    // меняем канал на третий
  Serial1.write(0xAD);                  // Посылаем байт со значением 173
  delay(400);                           // Ждём 400 мс
}                                       //
```

Одинаковый скетч для двух приёмников. Приёмники должны быть установленны на 1-й и 3-й каналы.

Если в последовательный порт приходит байт 0xAA, встроенный светодиод включается. Если приходит любой другой байт, светодиод выключается. Канал модуля должен быть 001 или 003. Выставить канал можно при помощи скетча выше "Перевод модуля в режим ручного ввода АТ команд". Для этого из монитора порта необходимо ввести `AT+C001` и enter для одного модуля и `AT+C003` и enter для другого. Так же это можно сделать автоматически, при помощи скетча "Пример тестирования модуля и сброса настроек на заводские" заменив при этом команду **AT+DEFAULT** на **AT+C003** для одного из модулей.

```
/*
 * Скетч для приёмников
 */

// Строки с * необходимо удалить, если Вы используете Piranha Ultra
#include <SoftwareSerial.h>             // * Подключаем библиотеку программного последовательного порта
                                        //
#define RX 8                            // * Определяем вывод RX (TX на модуле)
#define TX 9                            // * Определяем вывод TX (RX на модуле)
                                        //
SoftwareSerial Serial1(RX,TX);          // * Создаём объект программного последовательного порта
                                        //
void setup() {                          //
  Serial1.begin(9600);                  // Инициируем последовательный порт
  pinMode(LED_BUILTIN, OUTPUT);         // Устанавливаем режим работы вывода встроенного светодиода
```

```
}                                       //
                                        //
void loop() {                           //
  if(Serial1.available()){              // Если в буфере последовательного порта есть данные,
    if (Serial1.read() == 0xAA){        // если прочитанный байт равен 170
      pinMode(LED_BUILTIN, OUTPUT);     // переводим вывод встроенного светодиода в режим выхода.
      digitalWrite(LED_BUILTIN, HIGH);  // Выводим на светодиод логическую единицу.
    } else {                            // Если прочитанный байт равен любому другому значению
      pinMode(LED_BUILTIN, INPUT);      // переводим вывод встроенного светодиода в режим входа
    }                                   //
  }                                     //
}                                       //
```

Применение:

- Создание беспроводной связи между двумя Arduino
- Дистанционное управление роботами
- Дистанционное получение данных от датчиков, детекторов, сигнализаций и т.д.