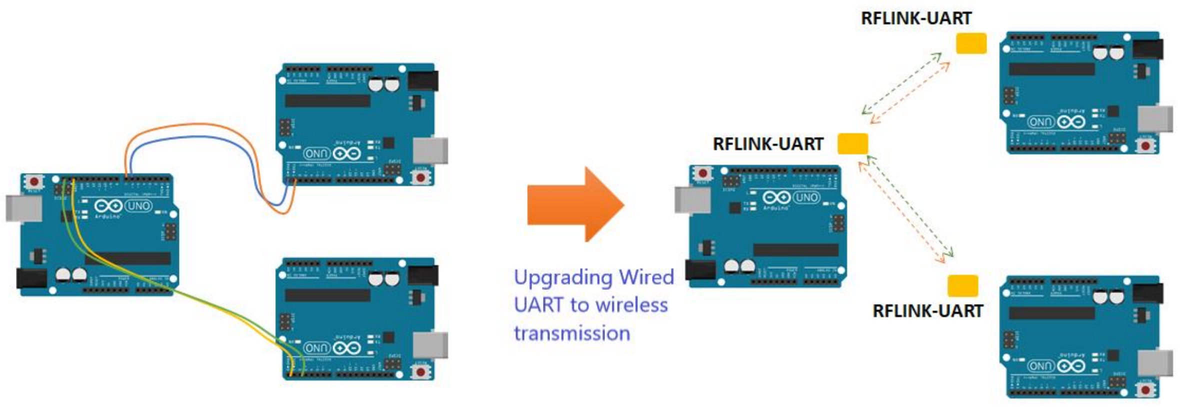
RFLINK-UART

Removing the Cable of UART immediately and Upgrading Wired UART to Wireless Transmission painlessly



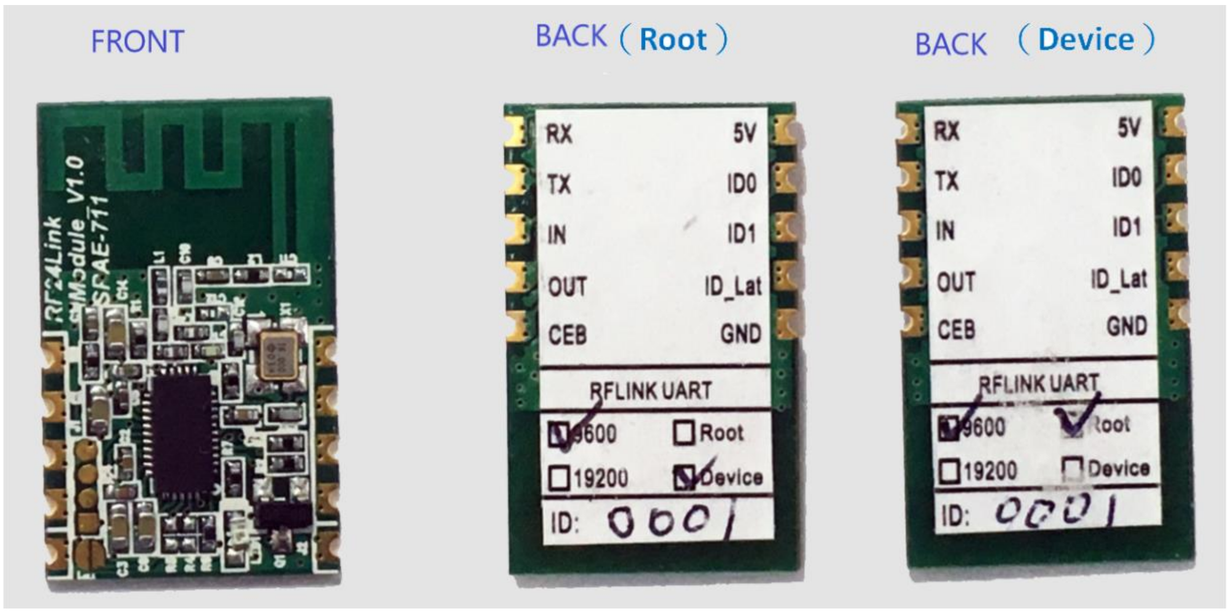
Directory

The RFLINK-UART Wireless UART Transmission Module is an easy-to-use module that instantly and painlessly upgrades wired UART to wireless UART transmission. More than that , there is a set of I/O port there , thus you don't need any coding effort and hardware to make IO switches well controlled remotely.

Module Appearance and Dimension

The RFLINK-UART module contains one root terminal (left) and up to four Device end (on the right side of the figure below, might be numbered from 1 to 4), the two are outward-looking the same, it can be identified by the label on the back .

As shown below, the Group ID of the RFLINK-UART module is 0001 and the BAUD is 9600.

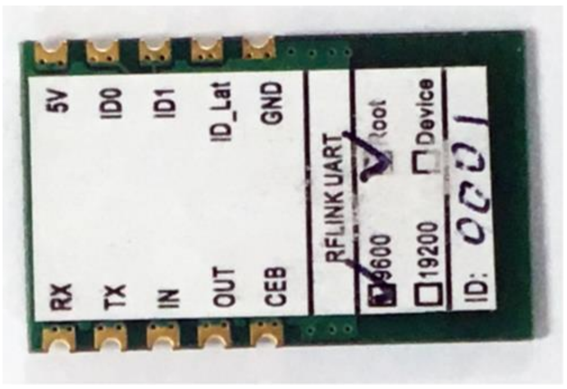
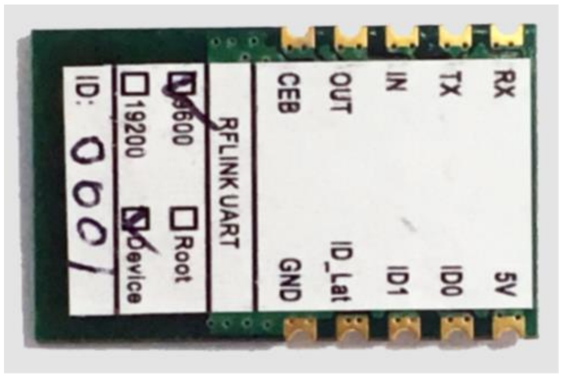


Module characteristics

1. **Operating voltage:** 3.3~5.5V
2. **RF Frequency**:2400MHz~2480MHz。
3. **Power** consumption: 24 mA@ +5dBm at TX mode and 23mA at RX mode.
4. **Transmit power:** +5dBm
5. **Transmission rate:** 250Kbps
6. **Transmission distance: around** 80 to 100m in the open space
7. **Baud rate**： 9,600bps or19,200bps

8. **Supports 1-to-1 or 1-to-multiple (up to four) transmission.**

Pin definition

Root	**Device**
 5V ID0 ID1 ID_Lat GND RX TX IN OUT CEB RFLINK UART Root Device 9600 19200 ID:	 CEB OUT IN TX RX GND ID_Lat ID1 ID0 5V RFLINK UART 9600 19200 Root Device ID:
GND→ Ground **+5V**→ 5V voltage input **The TX**→ corresponds to the RX of the development board UART **The RX**→ corresponds to the TX of the development board UART **THE CEB**→ This CEB should connect to the ground (GND),then the module will be power-on and can be used as a power-saving control function. **OUT**→ Output pin of IO Port (On/Off export) **IN**→Input pin of the IO Port (On/Off receive). **ID1, ID0** →selects which device to connect to via the HIGH/LOW combination of these two pins. **ID_Lat**→ Device ID Latch pins. When Root sets the target device via ID0, ID1, you need to set this pin LOW then the connection will officially be switched to the specified device.	**GND**→ Ground **+5V**→ 5V voltage input **The TX**→ corresponds to the RX of the development board UART **The RX**→ corresponds to the TX of the development board UART **THE CEB**→ This CEB should connect to the ground (GND),then the module will be power-on and can be used as a power-saving control function. **OUT**→ Output pin of IO Port (On/Off export)I **IN**→ Input pin of the IO Port (On/Off receive). **ID1, ID0**→ Through the HIGH/LOW combination of these two pins, the Device can be set to different device numbers. **ID_Lat**→ This Pin foot has no effect on Device.

How to use

All types of development boards and MCUs that support the UART communication interface can use this module directly, and there is no need to install additional drivers or API programs.

Setup Root and Devices

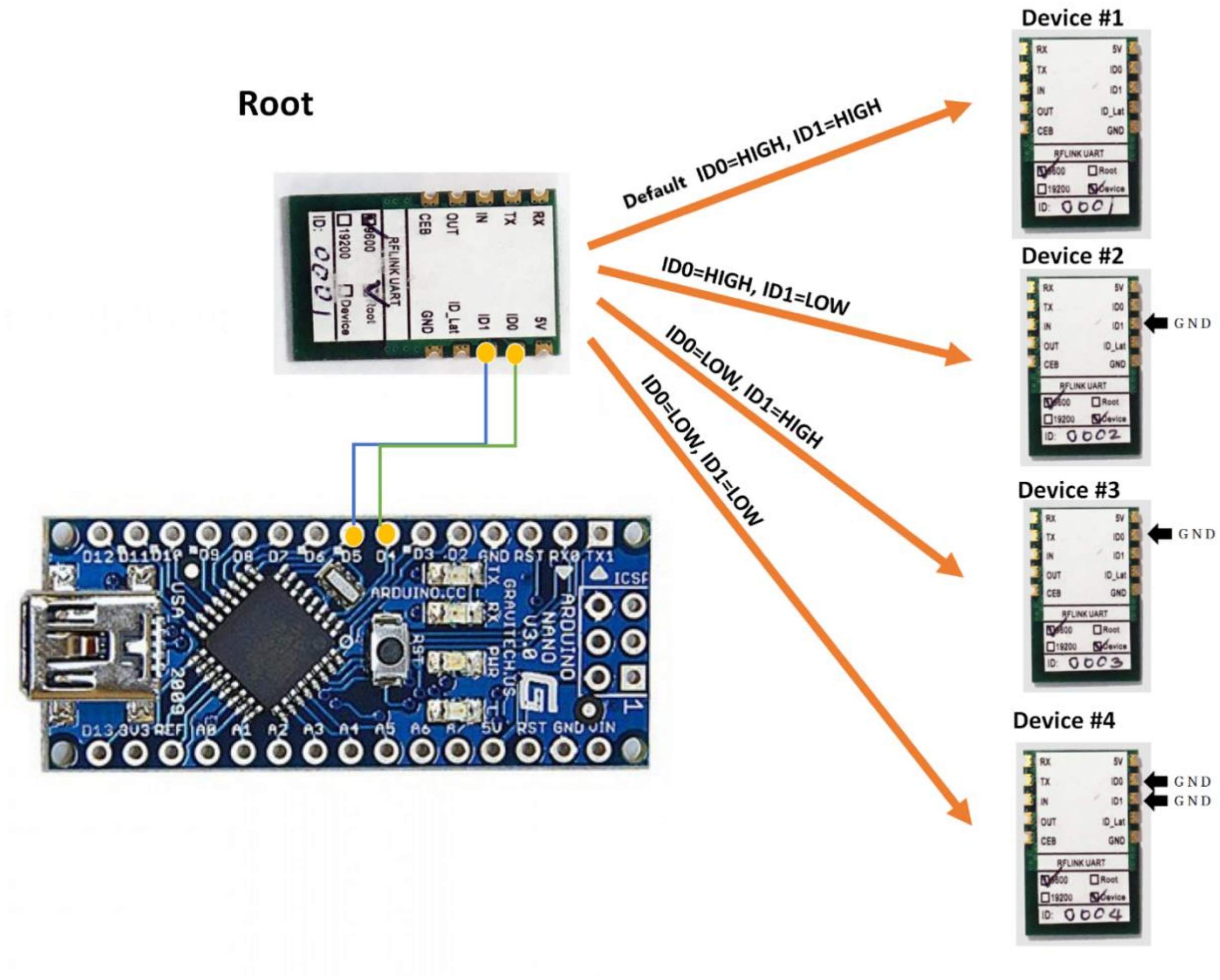
The traditional wired TTL is 1 to 1 transmission, the RFLINK-UART wireless UART transmission module will support 1-to-multiple type, default Root terminal (#0) after power-on with device (#1) is connected if you have another numbered Device (#2~#4). You can select different device side you want to connect to via the ID0 and ID1 pins on the root side. For the ID0/ID1 combination of device selection, please refer to the table below.

	Device 1 (#1)	*Device 2 (#2)*	*Device 3 (#3)*	*Device 4 (#4)*
ID0 pin	HIGH	HIGH	LOW	LOW
ID1 pin	HIGH	LOW	HIGH	LOW

ID0, ID1 pin are default HIGH, they will be LOW via connecting to the ground.
Note: Device side should be set to the required device number according first, the root will choose the target device via the same table.

You can choose the different device to transfer message via the ID0 and ID1 of root, usually tie ID0 or/and ID1 to the GND. More than that, the root side can also send Low/High signal through the IO pin to choose the target device on the fly.

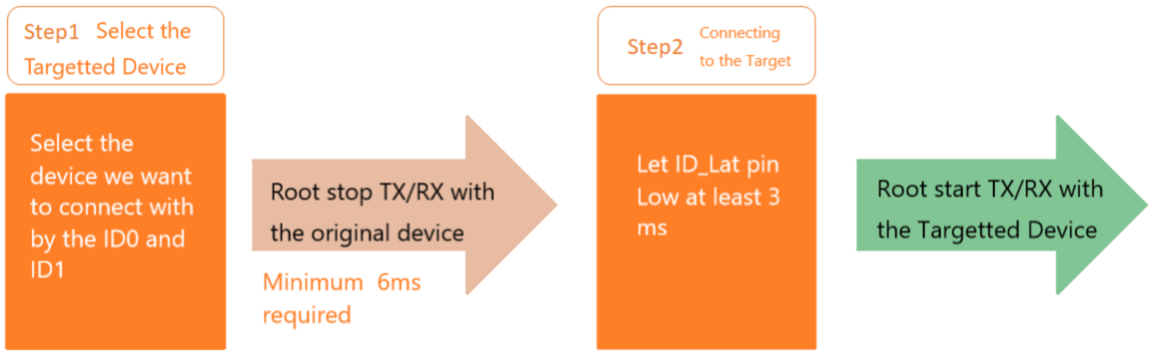
For example, in the figure below, Arduino Nano selects the Device to connect via the D4 and D5 pins 。



After sending the corresponding High/Low signal to the ID0 and ID1 pins, the Root terminal will interrupt the transmission with the old connection end (that is, stop the transmission and receiving with the old connection end). And wait for a Low signal from the ID_Lat pin to switch to the new connection.

Start transmitting/receiving messages with the new connection

After you send the target device number signal via ID0, ID1, all transection between the root and the current connected device will be halted. The new transection won't start until you send a LOW signal of ID_Lat at least 3ms.

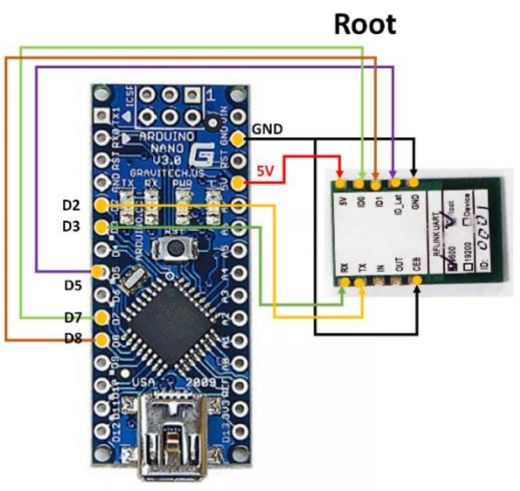

Step1 Select the Targetted Device
Select the device we want to connect with by the ID0 and ID1
Root stop TX/RX with the original device
Minimum 6ms required
Step2 Connecting to the Target
Let ID_Lat pin Low at least 3 ms
Root start TX/RX with the Targetted Device

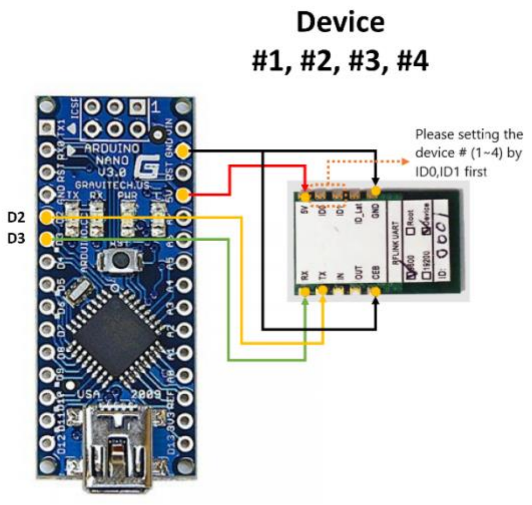
There are three use cases for Arduino , Raspberry Pi and sensors.

Working with Arduino

In addition to using Arduino's hardware TX/RX ports directly, this module also supports software serials, so it can Use in a software emulated UART to avoid occupying the physical UART interface.

The following example is connecting D2 and D3 to TX and the Root side of the RFLINK-UART module through the software serial RX, D7, D8 are the pins that set the connection to the device, and D5 is used as the ok toggle pin. Through Arduino's instructions digitalWrite outputs LOW or HIGH for the D7, D8 and D5 pins We can achieve the ability to dynamically connect to different devices.





Arduino (Italy)	D2	D3	D5	D7	D8	5V	GND

RFLINK-UART	RX	TX	ID_Lat (Root)	ID0 (Root)	ID1 (Root)	5V	GND CEB

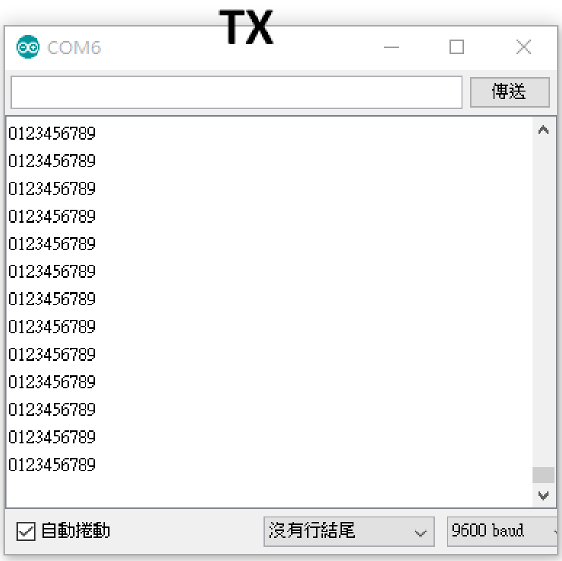
Example of a root-side transport program:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3); // RX, TX

#define pinLat 5
#define pinID0 7
#define pinID1 8

void setup() {
   pinMode(pinLat, OUTPUT);
   pinMode(pinID0, OUTPUT);
   pinMode(pinID1, OUTPUT);

   Serial.begin(9600);
   mySerial.begin(9600);
}

void loop() {
   #ID0=LOW, ID=HIGH means to connect device #3
   digitalWrite(pinID0, LOW);
   digitalWrite(pinID1, HIGH);
   # Wait 3ms seconds
   delay(3)
   # Determine the connection
   digitalWrite(pinLat, LOW);
   # Start teleportation
   mySerial.print("0123456789");
   Serial.println("0123456789");
   delay(1000);

   #ID0=LOW, ID=LOW, indicates that you want to connect device #1
   digitalWrite(pinID0, LOW);
   digitalWrite(pinID1, LOW);
   # Wait 3ms seconds
   delay(3)
   # Determine the connection
   digitalWrite(pinLat, LOW);
   # Start teleportation
   mySerial.print("abcdefghij");
   Serial.println("abcdefghij ");
   delay(1000);

}
```

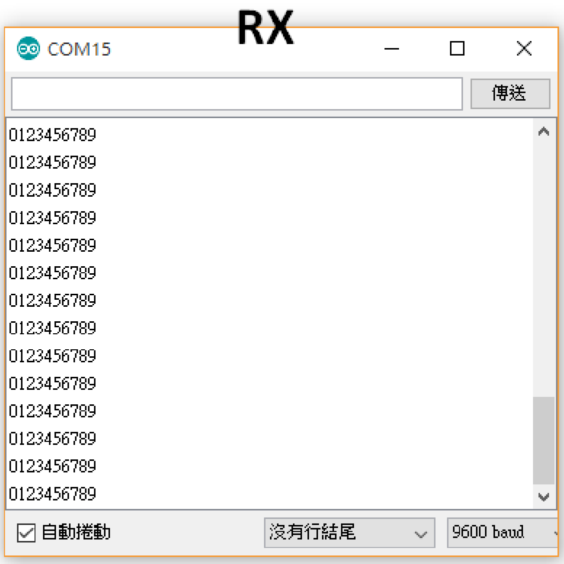
Example of RX receiver-side program :

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3); // RX, TX

void setup() {
   Serial.begin(9600);
   mySerial.begin(9600);
}

void loop() { // run over and over

   if (mySerial.available()) {
      Serial.println("");
      while (mySerial.available()) {
         Serial.print(char(mySerial.read()));
      }
   }

   delay(1000);
}
```

execute



```
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
```

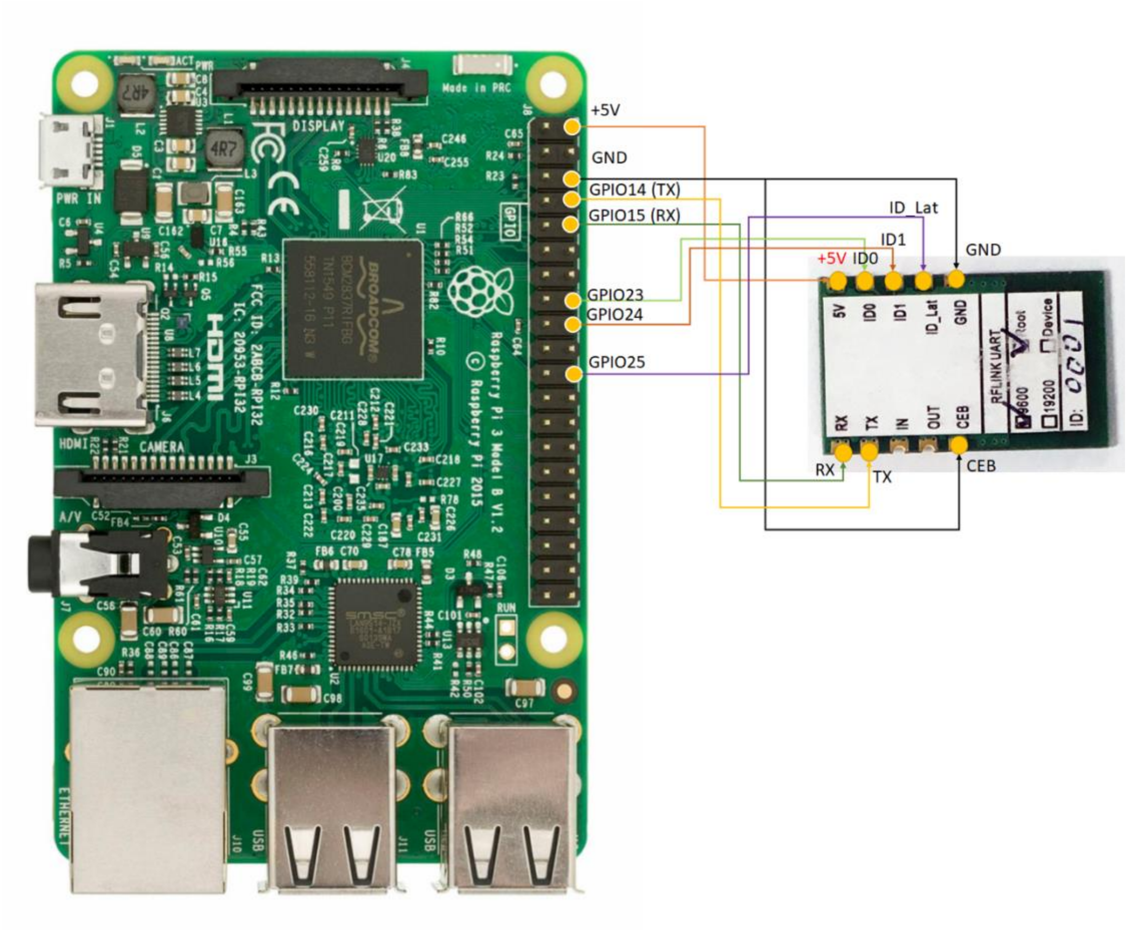


```
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
```

Working with the Raspberry Pi

Using this mod on the Raspberry Pi is also fairly easy! The pins of the RFLINK-UART module are connected to the corresponding ones of the Raspberry Pi as in the example of the Arduino above. In other words, you can read and write directly to the RX/TX pin and specify the device to connect, just like a traditional UART.

The following figure shows the connection method between the Root-side Raspberry Pi and the RFLINK-UART module, and the connection method of the Device end is basically the same, but it ID_ The Lat pin pin does not need to be connected, and ID0 and ID1 are set to different ID numbers depending on the requirements.



Example of program:

The transmitter repeatedly transmits information to device #3 and device #1

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import time
import serial
import RPi.GPIO as GPIO
```

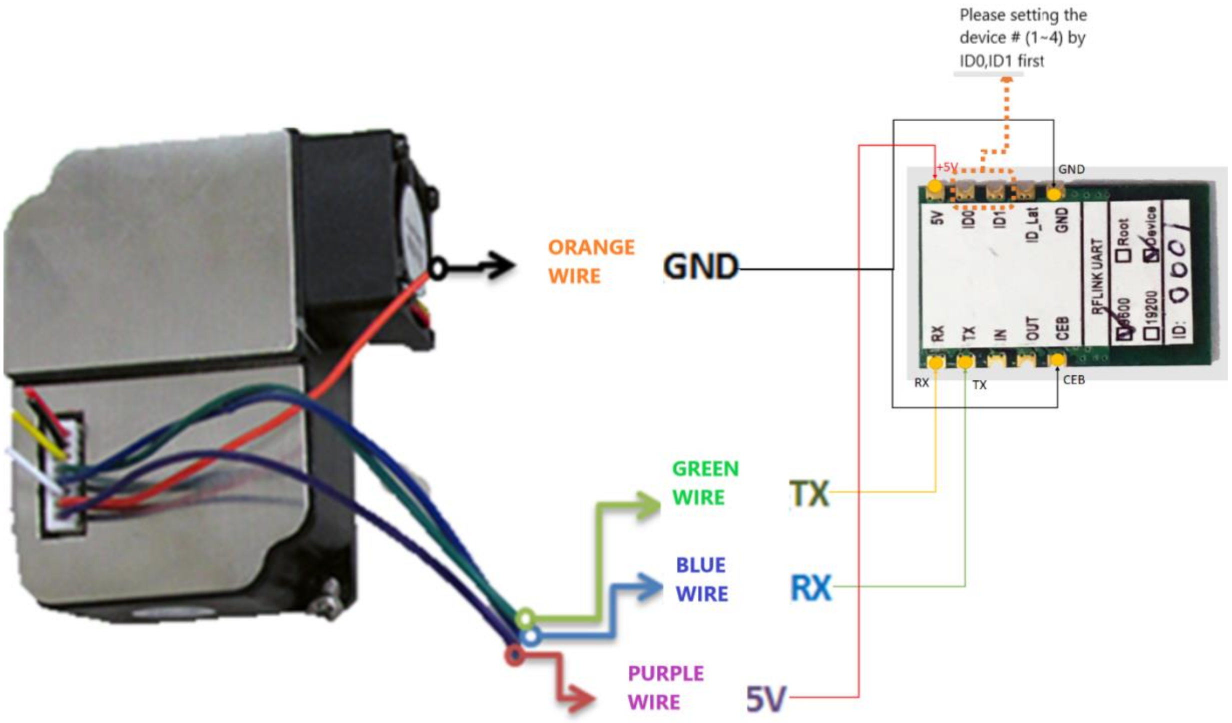
```
GPIO.setmode(GPIO. BCM)

GPIO.setup(23, GPIO. OUT)
GPIO.setup(24, GPIO. OUT)
GPIO.setup(25, GPIO. OUT)

ser = serial. Serial(
        port='/dev/ttyS0',
        baudrate = 9600,
        parity=serial. PARITY_NONE,
        stopbits=serial. STOPBITS_ONE,
        bytesize=serial. EIGHTBITS,
        timeout=1
)
counter=0

while 1:
        #Specifies to connect to device #3
        GPIO.output(23, GPIO. LOW)
        GPIO.output(24, GPIO. HIGH)
        # Wait 3ms
        time.sleep(0.03)
        #OK switch to device #3
        GPIO.output(25, GPIO. LOW)

        #Read the data from UART
        x=ser.readline()
        print x
#Writing data into UART, device #3 will receive this message
        ser.write('Write counter: %d \n'%(counter))

        #Specifies to connect to device #1
        GPIO.output(23, GPIO. HIGH)
        GPIO.output(24, GPIO. HIGH)
        # Wait 3ms
        time.sleep(0.03)
        #OK to switch to device #1
        GPIO.output(25, GPIO. LOW)

        #Read the data from UART
        x=ser.readline()
        print x
#Writing data into UART , device #1 will receive this message
        ser.write('Write counter: %d \n'%(counter))

        time.sleep(1)
```

Receiver: This example is a simple receive

```
#!/usr/bin/env python
import time
import serial

ser = serial. Serial(
        port='/dev/ttyS0',
        baudrate = 9600,
        parity=serial. PARITY_NONE,
        stopbits=serial. STOPBITS_ONE,
        bytesize=serial. EIGHTBITS,
        timeout=1
)
counter=0

while 1:
        #Read the data from UART
        x=ser.readline()
        print x
        #Writing data to UART
        ser.write('Write counter: %d \n'%(counter))
```

Direct connecting with sensor

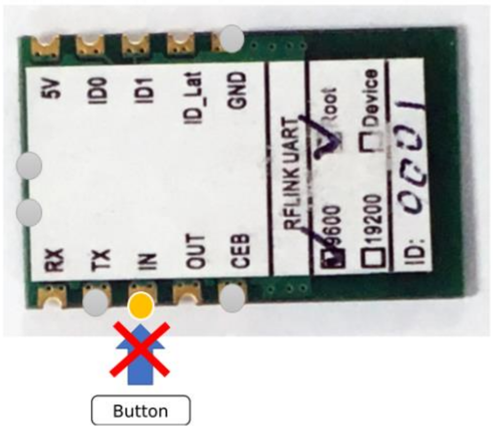
If your sensor supports the UART interface and the Baud rate supports 9,600 or 19,200 , then you can directly connect it to the device side of the RFLINK-UART module, and you can quickly and painlessly upgrade it Wireless function sensor too. The following G3 PM2.5 sensor is taken as an example, refer to the following connection method



Next, please prepare a development board (either Arduino or Raspberry Pi) to connect the RO of the RFLINK-UART module　On the ot side, you can read the G3 transmission in the general UART way PM2.5 data, congratulations, the G3 has been upgraded to a PM2.5 sensing module with wireless transmission capabilities.

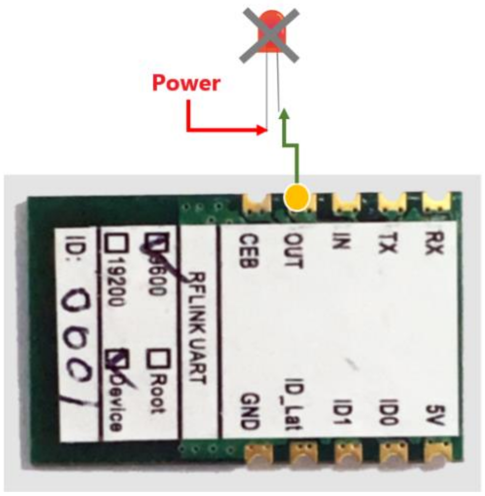
Use IO Ports

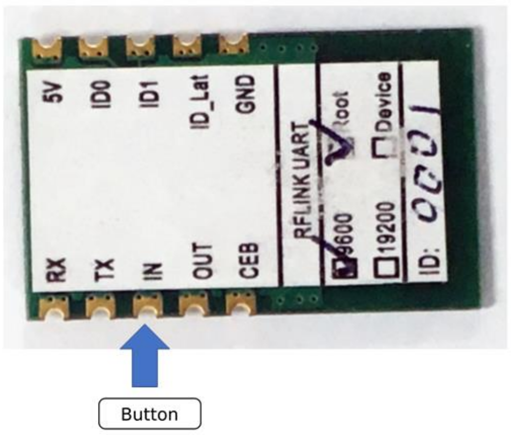
The RFLINK-UART module provides a set of IO ports that allow you to transmit on/off commands wirelessly, and this set Io Ports are not limited to the transmission or receiving end of the module, and both ends can control each other. As long as you change the voltage of the IN port at either end, you will change the output voltage of the Out port at the other end synchronously. Please refer to the following usage example to explain how to use IO Port to remotely control the switch LED bulb.



While the IN pin of Root/Device board is not connected to the GND , it is defult HIGH；

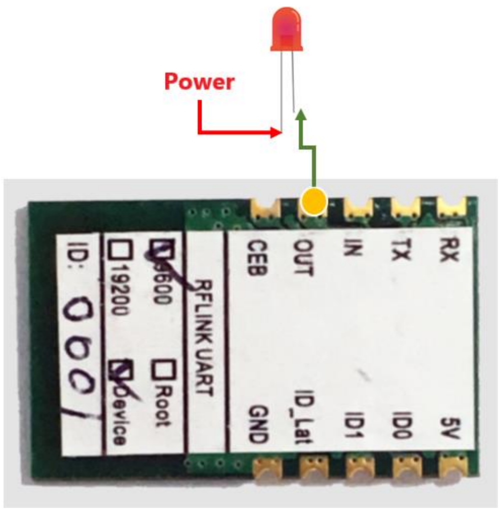
then the OUT pin of Root/Device board on the other side will be HIGH.-->LED is OFF





While the INpin of Root/Device board is connected to theGND (Low Level);

then the OUT pin of Root/Device board on the other side should be Low.--> LED is on


Power
CEB
OUT
IN
TX
RX
GND
ID_Lat
ID1
ID0
5V
RFLINK UART
Root
Device
9600
19200
ID: 0001