Bluetooth Transceiver RF Module Wireless Serial TTL V1.05Manual

Prepared to you by

عِلمٌ يُنتَفَعُ بِهِ

A product from

RAM® Electronics

INTEGRATED SOLUTIONS AT ONE PLACE

www.ram.com.eg

5/13/2013

Contents

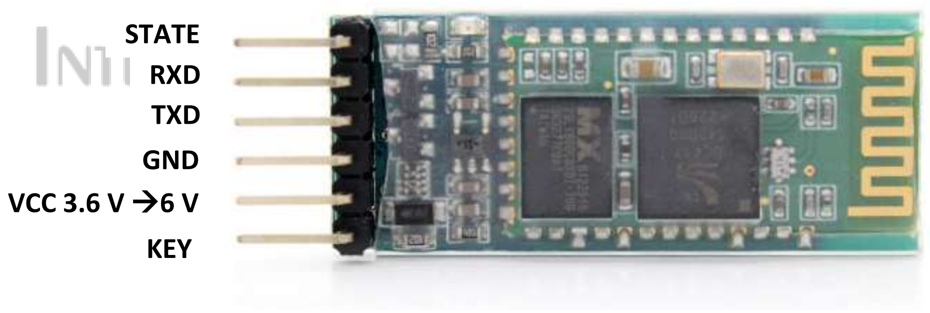
1. Introduction

This serial Bluetooth module can work as either **master or slave.** It is composed of the BC417 Bluetooth chip put on a PCB optimized for working with Arduino boards. However, it can be used with any microcontroller. The communication between the module and any microcontroller is done via serial communication. The data rate of the communication is configurable via special commands sent to the module.

Features:

- Use BC417 popular Bluetooth chip, Bluetooth V2.0 Protocol standard and Mac layer IC with Serial Communications.
- Serial port operation voltage 3.3V
- Baud rate set at 9600, but you can change it with AT command
- Operation current, pairing at 30mA, after pairing is 8mA during communication
- Also can pair and communicate with laptop, laptop Bluetooth adapter, Bluetooth Shield on Arduino

2. Circuit Connection:



As shown in figure, the module contains six pins, explained as follows:

- **VCC:** input supply range from 3.6 V to 6 V (although the BC417 chip itself runs on 3.3V, but a regulator is used on the PCB to make its range from 3.6V to 6V).
- **GND:** ground.
- **RXD:** Serial RX, connected to the RX pin in the Arduino. The RX pin on Arduino can be the Hardware RX pin of the Arduino port or any port which is configured as a receiver using the Arduino serial software library.

- **TXD:** Serial TX, connected to the TX pin in the Arduino. The TX pin on Arduino can be the Hardware TX pin of the Arduino port or any port which is configured as a Transmitter using the Arduino serial software library.
- **KEY:** Connected to VCC before switching the module up (before connection of the power to the VCC).
- **STATE:** Not used.

3. Modes of Operation

The Module has two modes of operation namely,

- AT Command Mode.
- Connection Mode.

The Modes could be distinguished by the LED blinking rate of the LED integrated on the Module board. This will be explained in detail in the discussion of each mode.

3.1. AT Command Mode

It is a mode of the module where a set of commands (AT Commands) are used to setup and configure the module.

In this mode, the module can't be detected by other Bluetooth Devices. All the Commands are sent to the module serially as a string. This string must be capital letters (case sensitive) and each command should be ended by "\r\n".

If the command is successfully understood by the module, the module will reply back with a response string usually contains the string "OK". Otherwise, the module returns back the error string “ERROR()” with a specific code that defines the type of this error (for more details see the Appendix).

Some commands may take parameters and others don't. Some Commands return parameters in the response string and others don't.

Commands could be classified as follows:

- Test Commands.
- Nearby Devices data inquiring Commands.

- Module data inquiring Commands.
- Module Control and Configuration commands.

These are explained in detail below.

❖ Test Commands

1. Test

This command is used to test if the Module is responding.

Command	Response	Parameter
AT	OK	None

Example:

Sent String:

"AT\r\n"

Response String:

"OK"

2. Reset

This command resets the module.

Command	Response	Parameter
AT+RESET	OK	None

Example:

Sent String:

"AT+RESET\r\n"

Response String:

"OK"

❖ Nearby Devices data inquiring Commands

3. Get the remote Bluetooth device's name

The command returns back the name of any Bluetooth device that is paired with the module.

Command	Response	Parameter
AT+RNAME?<Param1>	1. +NAME:<Param2> OK----success 2. FAIL----failure	Param1: Remote Bluetooth device address Param2: Remote Bluetooth device address

Example:

Assume the address of the paired device is aa:bb:cc:dd:ee:ff and its name is "RAM"

Sent String:

"AT+ RNAME?aabb,cc,ddeeff\r\n"

Response String:

"+RNAME:RAM
OK"

4. Get the Bluetooth address of most recently used paired device

Command	Response	Parameter
AT+MRAD?	+ MRAD : <Param> OK	Param: the Bluetooth address of the most Recently authenticated device

Example:

Sent String

" AT+MRAD?\r\n"

Response String:

"+MRAD:aa:bb:cc:dd:ee:ff
OK"

5. Inquire Bluetooth device

This command scans the area and returns back the addresses for all the detected devices.
Note that, the module should be in the "PAIRABLE" state. It is worth mentioning that this command is used also to set the module to the "PAIRABLE" state (when issued for the first time).

Command	Response	Parameter
AT+INQ	+INQ: <Param1>,<Param2>,<Param3> OK	Param1: Bluetooth address Param2: device type Param3: RSSI signal intensity

Example:

Sent String

" AT+INQ \r\n"

Response String:

```
"+INQ:aa:bb:cc:dd:ee:ff
+INQ:11:22:33:44:55:66 +INQ:zz:xx:yy:rr:ss:ww
 OK"
```

❖ Module data inquiring Commands

6. Get the Module version

Command	Response	Parameter
AT+VERSION?	+VERSION: <Param> OK	Param: Version number

<u>**Example:**</u>

Sent String

" AT+VERSION?\r\n"

Response String:

"+VERSION:2.0-20100601
OK"

7. Set/ inquire device’s name

Command	Response	Parameter
AT+NAME=<Param>	OK	Param: Bluetooth device name Default: “HC-05”
AT+NAME?	1. +NAME:<Param> OK----success 2. FAIL----failure	

Example:

To set the name:

Sent String:

" AT+NAME=EMAR\r\n"

Response String:

"OK"

Ask for the name:

Sent String:

" AT+NAME?\r\n"

Response String:

" +NAME: EMAR

OK"

8. Set/Inquire- passkey

Command	Response	Parameter
AT+PSWD=<Param>	OK	Param: passkey Default: "1234"
AT+ PSWD?	+ PSWD : <Param> OK	

9. Delete all paired devices in the pair list

Command	Response	Parameter
AT+RMAAD	OK	None

10. Get the work status of Bluetooth module

Command	Response	Parameter
AT+STATE?	+ STATE: <Param> OK	Param: work status of module Return value： “INITIALIZED” ----initialized status “READY” ---- ready status “PAIRABLE” ----pairable status “PAIRED” ----paired status “INQUIRING” ----inquiring status “CONNECTING” ----connecting status “CONNECTED” ----connected status “DISCONNECTED” ----disconnected status "NUKNOW" ----unknown status

11. Get module Bluetooth address

Command	Response	Parameter
AT+ADDR?	+ADDR: <Param> OK	Param: Bluetooth address

Example:

Module Bluetooth address: 12: 34: 56: ab: cd: ef

Sent String:

"AT+ADDR?\r\n"

Response String:

"+ADDR:1234:56:abcdef

OK"

12. Set/inquire device type

A class of a Bluetooth module determines whether this module is part of a laptop, cellular phone, headset…etc.
Each class has a defined value to identify it (refer to the appendix no. 2 in this datasheet for more details).

Command	Response	Parameter
AT+CLASS=<Param>	OK	Param: device type. Bluetooth device type is a 32 byte parameter indicates the device type and what type can be supported. Default: 0 More information is provided in this datasheet.
AT+ CLASS?	1. + CLASS:<Param> OK----success 2. FAIL----failure	

13. Delete paired device in the Bluetooth pair list

Command	Response	Parameter
AT+RMSAD=<Param>	OK	Param: Bluetooth device address

14. Seek the paired device in the Bluetooth pair list

Command	Response	Parameter
AT+FSAD=<Param>	1. OK----success 2. FAIL----failure	Param: Bluetooth device address

❖ Module Control and Configuration commands

15. Restore default status

Command	Response	Parameter
AT+ORGL	OK	None

The parameters of default status:

- Device type: 0
- Inquire code: 0x009e8b33
- Module work mode: Slave Mode
- Connection mode: Connect to the Bluetooth device specified
- Serial parameter: Baud rate: 38400 bits/s; Stop bit: 1 bit; Parity bit: None.
- Passkey: “1234”
- Device name: “H-C-2010-06-01”

16. Set/ inquire module role

Command	Response	Parameter
AT+ROLE=<Param>	OK	Param: 0---- Slave role 1---- Master role 2---- Slave-Loop role Default: 0
AT+ ROLE?	+ ROLE:<Param> OK	

17. Set/ Inquire- serial parameter

Command	Response	Parameter
AT+UART=<Param>,< Param2>,<Param3>	OK	Param1: baud rate(bits/s) The value (Decimal) should be one of the following: 4800 9600 19200 38400 57600 115200 23400 460800 921600 1382400 Param2:stop bit: 0----1 bit 1----2 bits Param3: parity bit: 0----None 1----Odd parity 2----Even parity Delete: 9600, 0, 0

18. Set/ Inquire - connection mode

Command	Response	Parameter
AT+CMODE=<Param>	OK	Param: 0----connect the module to the specified Bluetooth address. (Bluetooth address can 1----connect the module to any address (The specifying address has no effect for this mode.) 2----Slave-Loop Default connection mode: 0

19. Initialize the SPP profile lib

Command	Response	Parameter
AT+INIT	1. OK----success 2. FAIL----failure	None

20. Inquire Bluetooth device

Used for changing the state of the module from "INTIALIZED" to "PAIRABLE". Entering this state enables the module to scan for the addresses of the nearby Bluetooth devices using the same command again as we stated in the command number 5 above.

Command	Response	Parameter
AT+INQ	+INQ: <Param1>,<Param2>,<Param3> OK	Param1: Bluetooth address Param2: device type Param3: RSSI signal intensity

Example:

Sent string

" AT+INQ \r\n"

Response string

"OK"

21. Set pair

Allows the module enter the "PAIRED" state with another Bluetooth device.

Command	Response	Parameter
AT+PAIR=<Param1>,<Param2>	1. OK----success 2. FAIL----failure	Param1: Bluetooth address of remote device Param2: limited time connection (second)

Example

Make pair with the remote Bluetooth device(address:12:34:56:ab:cd:ef), the limited time is 20s.

Sent string

"AT+PAIR=1234,56,abcdef,20\r\n"

Response string

"OK"

22. Connect device

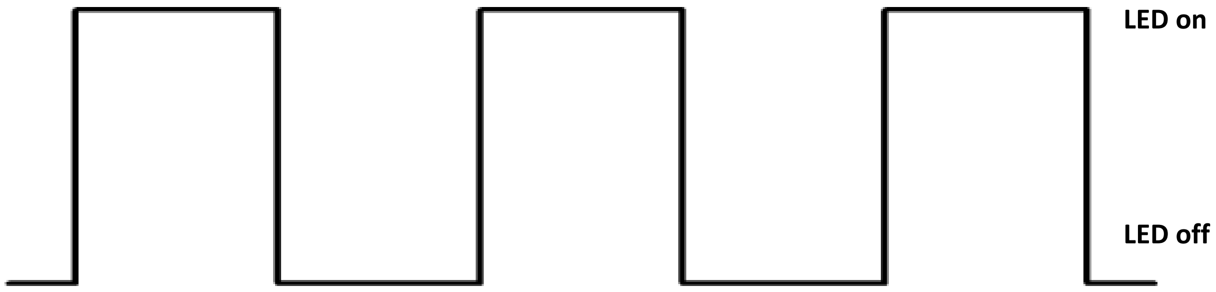
This makes the device enter the "CONNECTED" state, where any serial data imposed on the transmitter pin is transmitted immediately.

Command	Response	Parameter
AT+LINK=<Param>	1. OK----success 2. FAIL----failure	Param: Bluetooth address remote device

The previous commands are the most important and frequently used. For more commands refer to the datasheet.

The detection of the AT mode according to the LED blinking rate

You can detect the device in AT mode by noticing that the LED is blinking slowly and it keeps on for long time intervals as shown in figure.



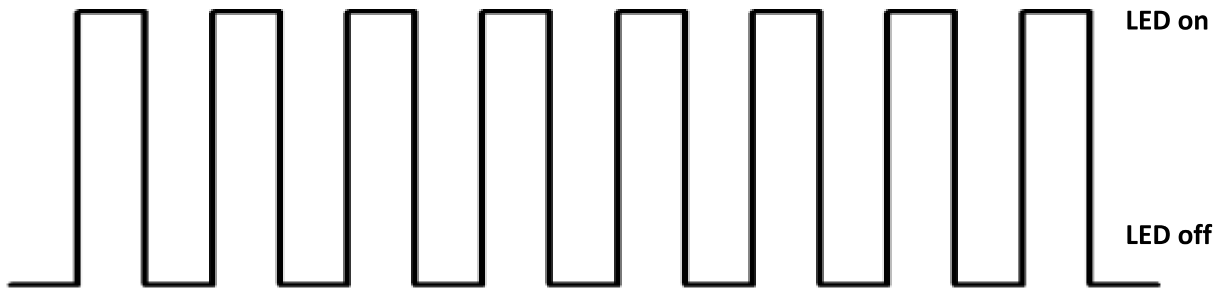
LED blinking pulses of AT mode

3.2. Connection Mode

The connection mode passes by three stages.

1. "PAIRABLE" state

This state prepares for the real connection between the Module and any Bluetooth devices and detected by noticing the fast blinking rate of the LED as shown in figure below.



LED blinking pulses of "PAIRABLE" state

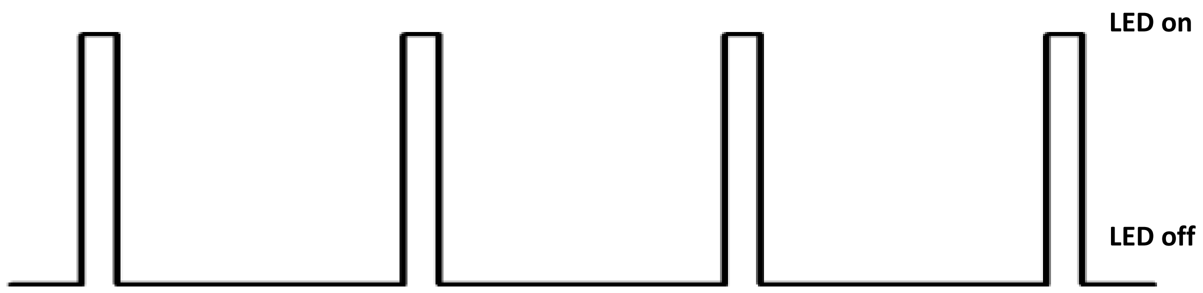
In this state, the Module will be visible for any nearby Bluetooth devices.

2. "PAIRED" state

At this state, the module and the other Bluetooth device are logically linked together that they can access each other setting such as name, address…etc.

The Module and the other device can't however have a real communication (Send/Receive data to each other).

The state could be detected by the LED blinking rate as the figure shown below.

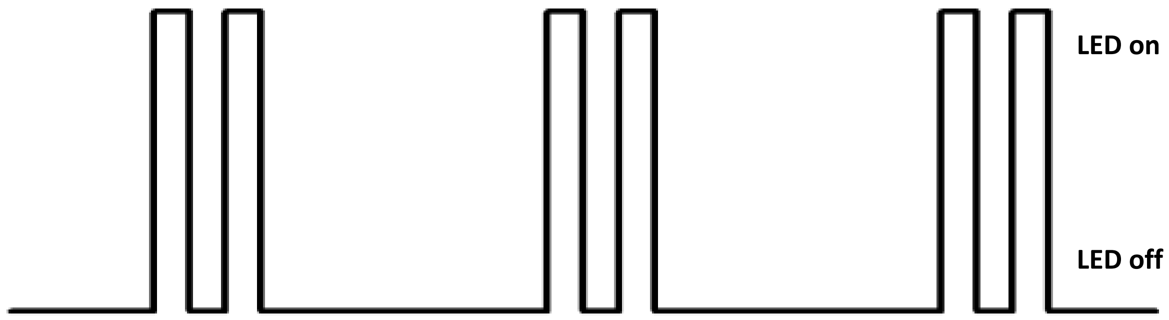


LED blinking pulses of "PAIRED" state

3. "CONNECTED" state

In this state, the Module and the other device can have a real communication (Send/Receive data to each other).

The state could be detected by the LED blinking rate as the figure shown below.



LED blinking pulses of "CONNECTED" state

4. General procedures to setup the Bluetooth module for communication:

The following steps are used as a guide to establish your own communication between two Arduino boards or an Arduino board and another Bluetooth device.

1) Connect the Bluetooth module to the Arduino as follows:

- Connect the “KEY” pin to the “VCC” pin and connect them together to the 5V pin on the Arduino. Connect the ground pin also.
- Connect the “RX” pin to the Rx pin on the Arduino (or any other I/O pin configured as a software serial Rx pin).
- Connect the “TX” pin to the Tx pin on the Arduino (or any other I/O pin configured as a software serial Tx pin).

2) After the module is connected and powered up, it is working in the AT command mode. You can change the settings of the module like its name, password key, and the communication baud rate.

3) Set the module to the “PAIRABLE” state to enable it to start to communicate with other Bluetooth devices. This is done by calling the two commands “AT+INIT” and “AT+INQ” respectively.

4) The device is then in “PAIRABLE” state. This enables any other device to pair or to connect to the module. It also enables the module to pair or connect to other Bluetooth modules. Use the “AT+INQ” command again to search for the nearby devices addresses.

5) After you pick up the address of the device you want your module to pair or connect with, use the “AT+PAIR” or “AT+LINK” command along with this address to pair or to connect to the device respectively.

6) After using the “AT+LINK” command, the module is now connected and any serial data imposed on the transmitter will be sent directly via Bluetooth. Develop your program on the Arduino that will send and/or receive the data. The communication is now seen like an ordinary serial communication.

5. Applications

5.1. Communication between two Arduinos through two Bluetooth Modules

The following code implements the communication between two Arduino boards using two Bluetooth modules, one configured as master and the other is configured as a slave. The initialization made for both the master and the slave is similar except for the command which determine the role of the Bluetooth module (master or slave).

The code main body (loop function) contains two sections, one is used in the master device and the other is used for the slave device. When you are uploading the program on the master comment the section specified for the slave and vice versa.

The following tips are to be followed:

1) Initialization cannot be done unless the device is in the AT mode, with the LED blinking rate is slow.

2) In the initialization phase as you will see in the following code, you must issue the "AT+INIT" command before issuing either "AT+INQ" command (which sets the module to "PAIRABLE" state) or "AT+ROLE" command (which sets the device is master or slave), or the "AT+CMODE" command.

3) The "AT+INQ" is usually used twice. The first one is to set the device to the "PAIRABLE" state where it can pair or connect with other Bluetooth devices. The second one is according to the user's need, where it gives the user back the available nearby Bluetooth devices' addresses. The user chooses the appropriate one to communicate with or to pair with.

4) The pairing is totally different from connection. Pairing binds the two Bluetooth devices so as to be seen to each other and exchange their settings data such as names and addresses. But full connection enables the two devices to transmit and receive the user's data (files, images…etc.).

```
#include<SoftwareSerial.h>

#define RxD 6

#define TxD 7


SoftwareSerial BT(RxD,TxD);


void setup()

{

  Serial.begin(9600);

  BT.begin(38400);

  delay(1000);

  initialize_bluetooth();


  //connecting to the bluetooth device with the address "00:12:11:17:02:10"

  /*Note that this command cannot be issued unless the module is in pairable state,

  in other words, the "AT+INIT" then the "AT+INQ" commands should be first issued in order.

  It worth mentioning that this is done in the initialization function*/

  BT.print("AT+LINK=0012,11,170210\r\n"); //This command used only in one of the two devices


  /*Note that the address of the device which we want to connect to can be got using the
"AT+INQ" command. Then the response string will be on the form: "+INQ:00:12:11:17:02:10,xx,yy".
The address string can be then extracted from the response string by some simple code
manipulation and then used in the "AT+LINK" command to establish the connection. For more
information about string operations in Arduino, please consult the Arduino reference.*/


  delay(1000);

}
```

```
void loop()
{
  //----------------------------------------------------------
  //Simple code for the master sending a message to the slave
  //----------------------------------------------------------
  //transmitting data to the slave
  BT.print("Hello from Emar to RAM!\r\n");
  BT.flush();         //this makes the code stops untill all serial data finish transmission
  delay(500);

  //waiting for the slave response and print the response on the serial monitor
  while(1)
  {
    if(BT.available()>0)
    {
      print_bt_response();
      break;
    }
  }



  /*
  //--------------------------------------------------------------------------------
  //Simple code for the slave receiving a message from the master and then responding back
  //--------------------------------------------------------------------------------
  //receiving data from the master by the slave
  while(1)
  {
    if(BT.available()>0)
    {
      print_bt_response();
      break;
    }
  }

  //responding back to the master
  BT.print("Hello from RAM to Emar!\n");
  BT.flush();     //this makes the code stops untill all serial data finish transmission
  delay(500);
  */

}



//*********************************************************************************
//Function that reads BT received data and returns 1 if successfull and 0 otherwise
//*********************************************************************************
int print_bt_response()
{
  int response;
  char out,outprev = '$';
  while(BT.available()>0)
  {
    out = (char)BT.read();
    Serial.print(out);
    if ((outprev == 'O')&&(out == 'K'))
    {
      Serial.print("\n");
      response = 1;
      return response;
    }
    outprev = out;
  }
  response = 0;

  return response;

}

//--------------------------------------------------------------------------------

//--------------------------------------------------------------------------------



//*********************************************************************************

//Initializing the bluetooth and put it into the pairable state to be ready for connection

//*********************************************************************************

void initialize_bluetooth()

{

  int flag=2;


  delay(500);


  //restore default status

  BT.print("AT+ORGL\r\n");

  flag = print_bt_response();

  delay(500);

  if(flag == 1)

  {

    Serial.print("Module default settings restored ... Success(1)\n\n");

  }

  else if(flag == 0)

  {

    Serial.print("Failed(1)\n\n");

  }

  flag = 2;


  //set bt name as "Emar"

  BT.print("AT+NAME=Emar\r\n");
  flag = print_bt_response();

  delay(500);

  if(flag == 1)

  {

    Serial.print("Module took new name ... Success(2)\n\n");

  }

  else if(flag == 0)

  {

    Serial.print("Failed(2)\n\n");

  }

  flag = 2;


//set pin code (should be the same for the master and slave for establishing //connection)

  BT.print("AT+PSWD=1234\r\n");

  flag = print_bt_response();

  delay(500);

  if(flag == 1)

  {

    Serial.print("Module took new password ... Success(3)\n\n");

  }

  else if(flag == 0)

  {

    Serial.print("Failed(3)\n\n");

  }

  flag = 2;



  delay(1000);


  //initialize spp profile lib (should be issued before issuing the next command)

  BT.print("AT+INIT\r\n");

  flag = print_bt_response();
  delay(1000);

  BT.print("AT+INIT\r\n");

  flag = print_bt_response();

  delay(1000);

  if(flag == 1)

  {

    Serial.print("Module spp profile lib initialized ... Success(4)\n\n");

  }

  else if(flag == 0)

  {

    Serial.print("Failed(4)\n\n");

  }

  flag = 2;


//set to pairable state (to be seen by other bluetooth devices and to be ready for
//connection)

  BT.print("AT+INQ\r\n");

  flag = print_bt_response();

  delay(500);

  if(flag == 1)

  {

    Serial.print("Module inquirable (set to pairable state) ... Success(5)\n\n");

  }

  else if(flag == 0)

  {

    Serial.print("Failed(5)\n\n");

  }

  flag = 2;


  //if you want to set bt to be master, then "AT+ROLE=1"

  //if you want to set bt to be slave, then "AT+ROLE=0"

  BT.print("AT+ROLE=1\r\n");       //set as master here
  flag = print_bt_response();

  delay(500);

  if(flag == 1)

  {

    Serial.print("Module role set to master ... Success(6)\n\n");

  }

  else if(flag == 0)

  {

    Serial.print("Failed(6)\n\n");

  }

  flag = 2;


}
```

5.2. Communication between Arduino and Laptop through Bluetooth Module and Laptop Bluetooth

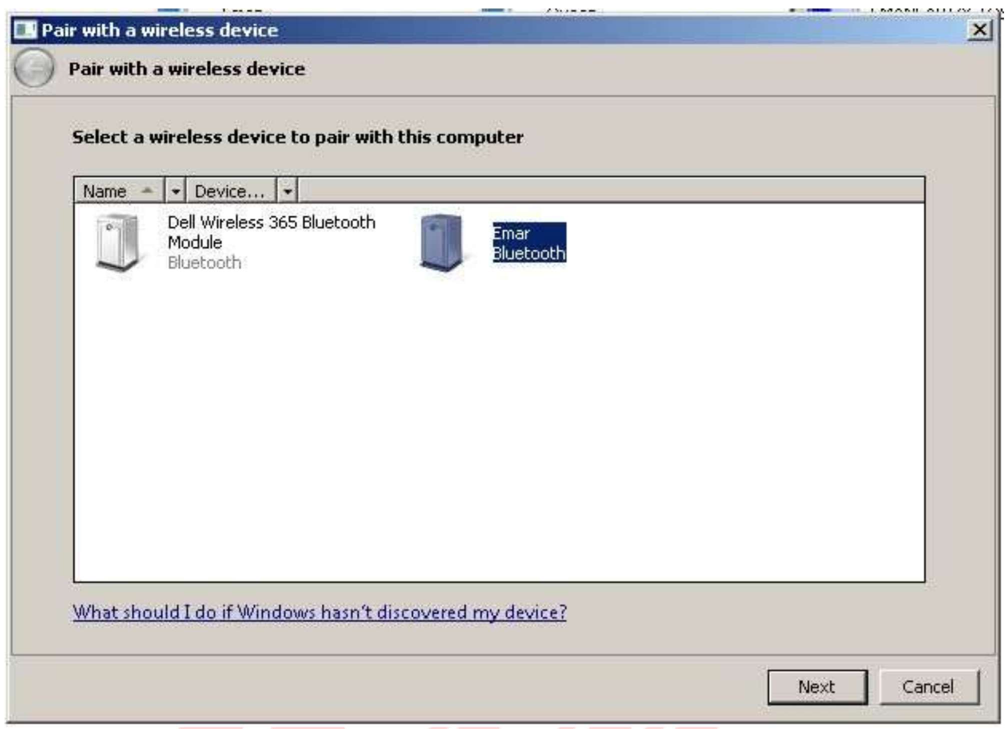
The connection between the Module and the Laptop Bluetooth could be done easily as the connection between any two Bluetooth devices. Note that if you are going to setup the communication between the Bluetooth module and your laptop Bluetooth, you must first put the module into the “PAIRABLE” state using the procedures and codes used in the previous section. This enables the module to be seen and paired or connected by the laptop Bluetooth.

After setting up your Bluetooth module and making your program on the Arduino that will send/receive the data, the following steps are to be done on your laptop to connect to the Bluetooth module.

Do not forget that you also must make first the program that will send/receive the data on your laptop. This program can be made using any programming language or MATLAB. The program will send/receive the data to be sent/received by the laptop Bluetooth via the COM port specified for the Bluetooth on your laptop. In the following example we send/receive those data using a simple application which you can download from many sites called “serial port monitor”. The serial port monitor acts as a hyperterminal that enables the user to see the received data on any COM port and send any characters to any COM port.

All steps are shown below:

1. **search for the Bluetooth Devices**


Pair with a wireless device

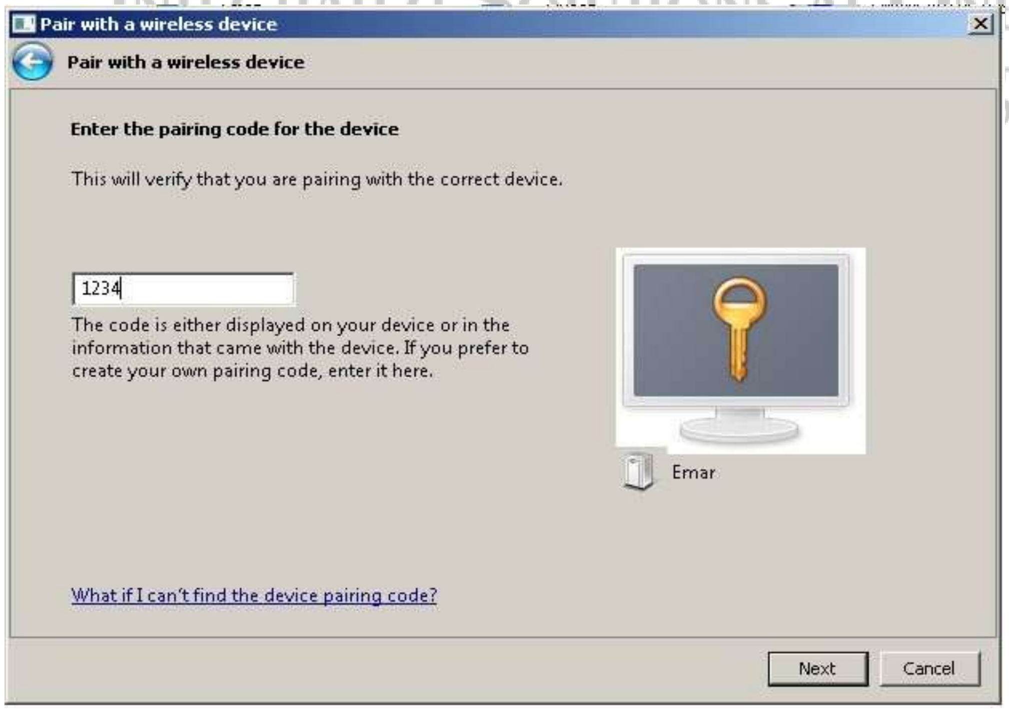
Pair with a wireless device

Select a wireless device to pair with this computer

Name	Device...
Dell Wireless 365 Bluetooth Module	Bluetooth
Emar Bluetooth	

What should I do if Windows hasn't discovered my device?

2. **Enter the pairing code**


Pair with a wireless device

Pair with a wireless device

Enter the pairing code for the device

This will verify that you are pairing with the correct device.

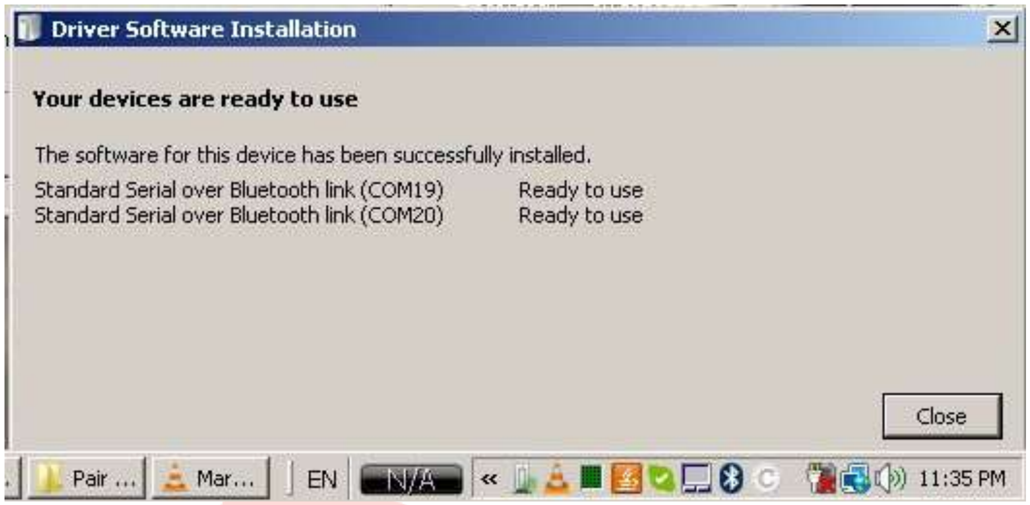
1234

The code is either displayed on your device or in the information that came with the device. If you prefer to create your own pairing code, enter it here.

Emar

What if I can't find the device pairing code?

3. the Bluetooth COM ports will be viewed



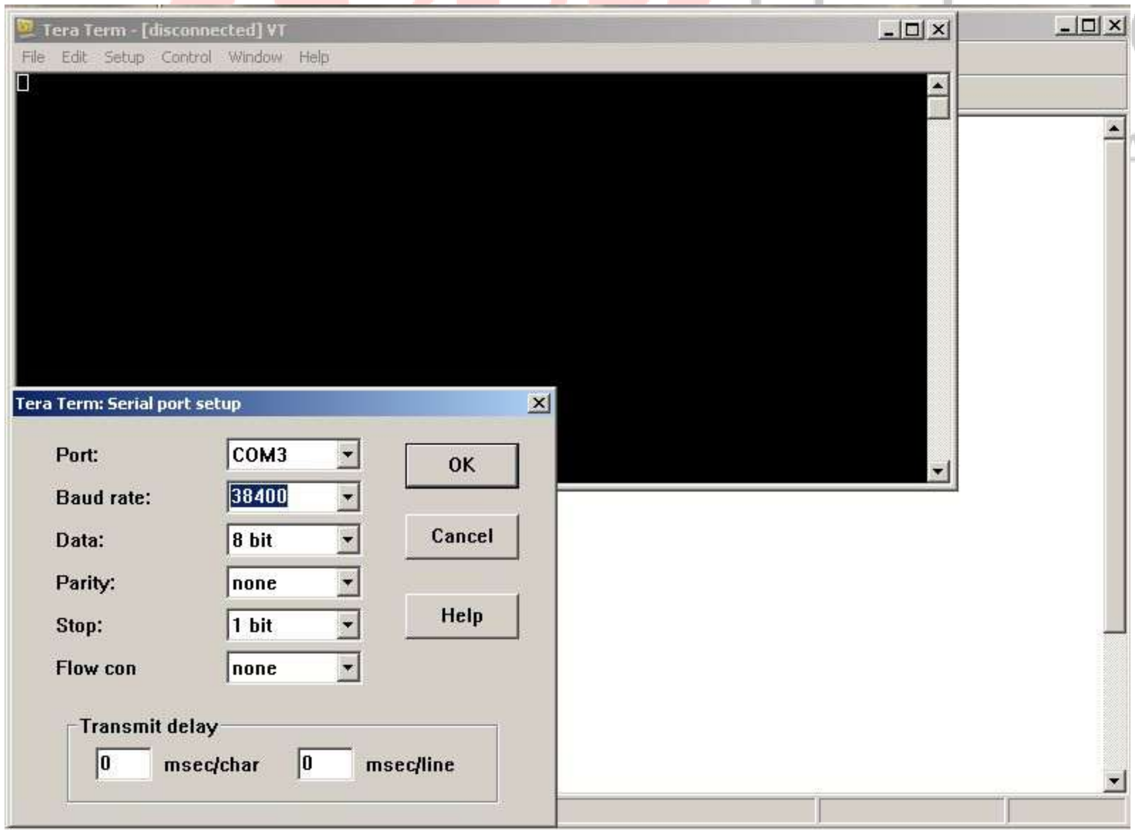
Driver Software Installation

Your devices are ready to use

The software for this device has been successfully installed.

Standard Serial over Bluetooth link (COM19) Ready to use
Standard Serial over Bluetooth link (COM20) Ready to use

4. Use any Serial Port Monitor software (we use the program called tera term), adust it's Baud rate from 9600 to 38400



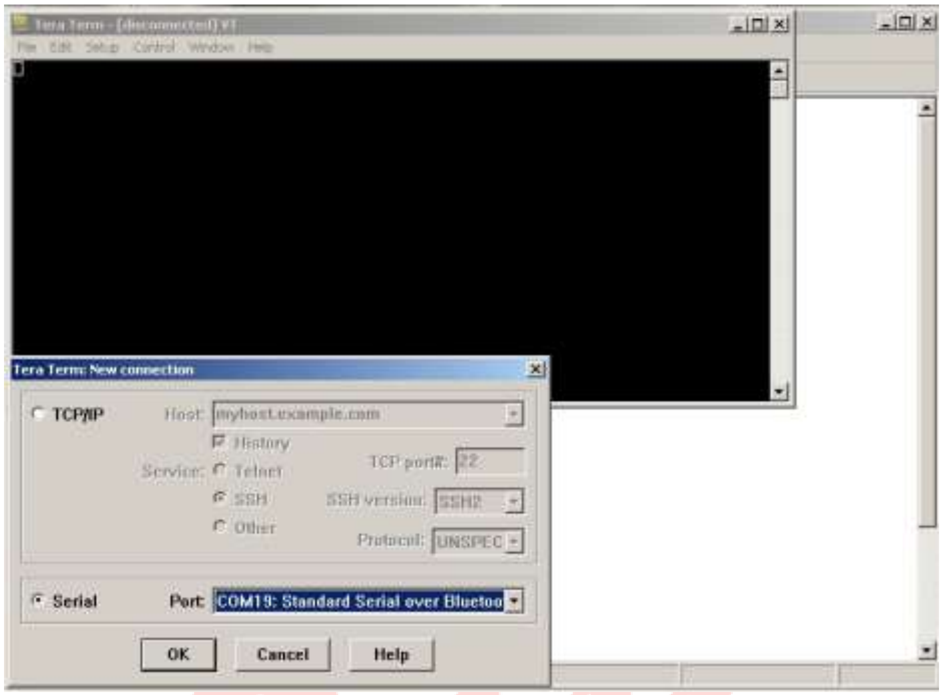
Tera Term - [disconnected] VT

Tera Term: Serial port setup

Port: COM3
Baud rate: 38400
Data: 8 bit
Parity: none
Stop: 1 bit
Flow con: none

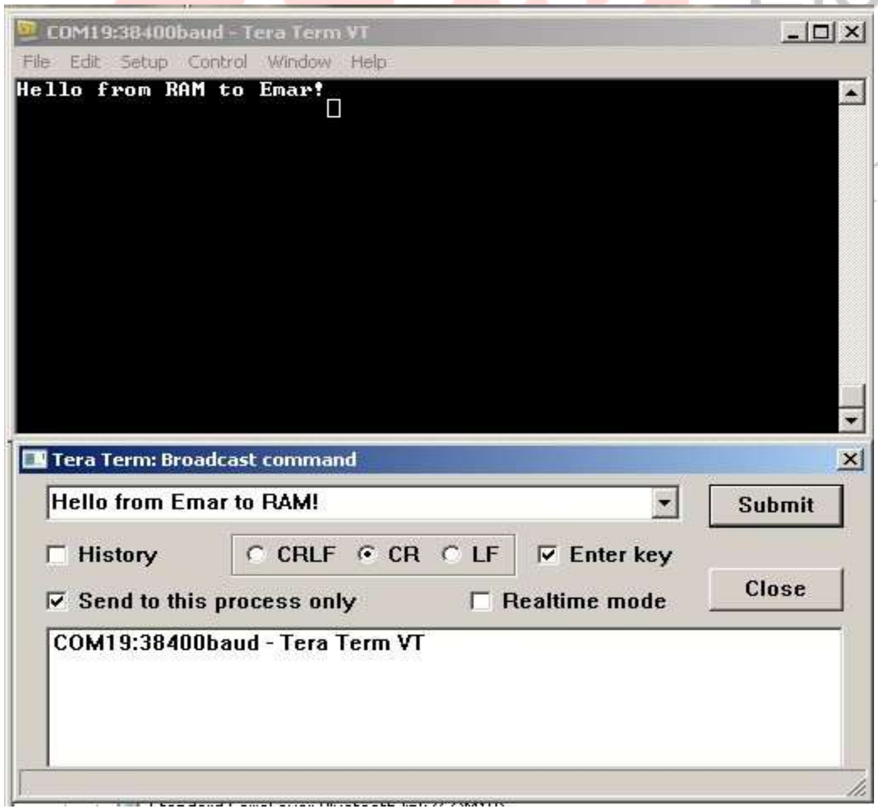
Transmit delay
0 msec/char 0 msec/line

5. **connect the Serial Monitor to the Bluetooth COM port (COM port 19 in our case)**


Tera Term - [disconnected] VT
File Edit Setup Control Window Help
Tera Term: New connection
OK
Cancel
Help

TCP/IP Host: myhost.example.com
History
Service: Telnet TCP port#: 22
SSH SSH version: SSH2
Other Protocol: UNSPEC
Serial Port: COM19: Standard Serial over Bluetoo

6. **The module will sent replies on the monitor terminal**


COM19:38400baud - Tera Term VT
File Edit Setup Control Window Help
Tera Term: Broadcast command
Submit
Close

```
Hello from RAM to Emar!
```

Hello from Emar to RAM!

History CRLF [✓ CR] LF [✓ Enter key]

[✓ Send to this process only] Realtime mode

COM19:38400baud - Tera Term VT

Appendix

Introduction of AT command error code:

The Bluetooth module provides a very useful feature that eases programming it. This feature is the AT command error codes. The module returns those codes to the user in cases of wrong AT commands are sent to the module or when an AT command is incorrectly used. Each code represents a specific error. This makes debugging the code using to control the module much easier. The following table shows each error and its corresponding error code.

The form of error returned by the module is:
ERROR:(error_code)

error code(Hexadecimal)	Note
0	AT command error (i.e. the command used is not an AT command)
1	Default result
2	PSKEY write error
3	Too long length of device name (more than 32 bytes).
4	No device name
5	Bluetooth address: NAP is too long.
6	Bluetooth address: UAP is too long.
7	Bluetooth address: LAP is too long.
8	No PIO number's mask
9	No PIO number
A	No Bluetooth devices.
B	Too length of devices
C	No inquire access code
D	Too long length of inquire access code
E	Invalid inquire access code
F	The length of passkey is 0.
10	Too long length of passkey (more than 16 bytes)
11	Invalid module role

12	Invalid baud rate
13	Invalid stop bit
14	Invalid parity bit
15	Authentication device is not at the pair list.
16	SPP lib hasn't been initialized.
17	SPP lib has been repeated initialization.
18	Invalid inquire mode
19	Too long inquire time
1A	No Bluetooth address
1B	Invalid safe mode
1C	Invalid encryption mode