Arduino with HC-05 (ZS-040) Bluetooth module – AT MODE

Posted on October 28, 2014

Updated 19.07.2015
Updated 26.07.2015
Updated 30.10.2015

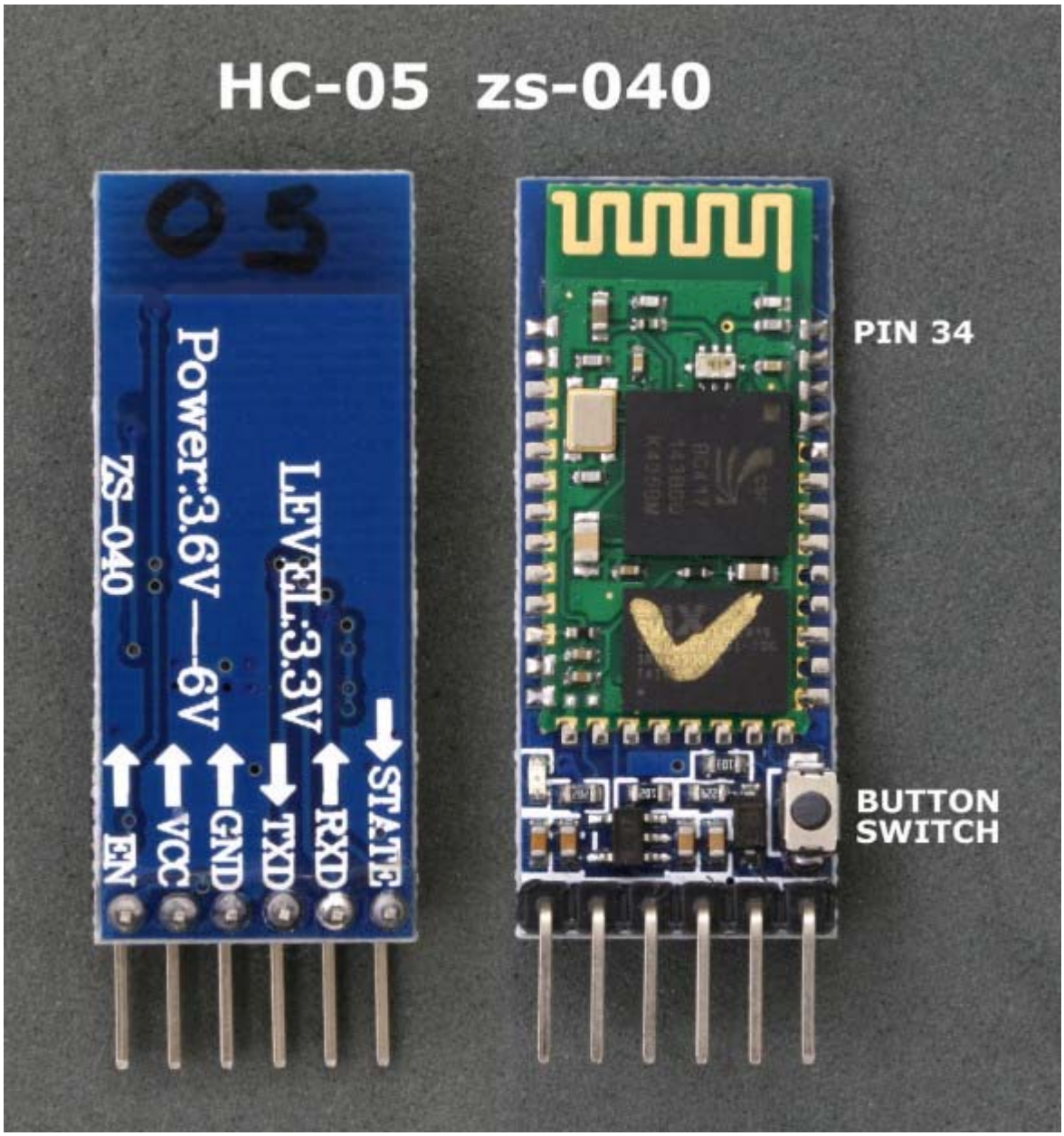
AT mode allows you to interrogate the BT module and to change some of the settings; things like the name, the baud rate, whether or not it operates in slave mode or master mode. When used as a master device AT commands allow you to connect to other Bluetooth slave devices.

There are many slightly different HC-05 modules, the modules I have are marked ZS-040 and have an EN pin rather than a KEY pin. They also have a small button switch just above the EN pin. They are based on the EGBT-045MS Bluetooth module.

Update: I now also have boards marked fc-114. See:
HC-05 FC-114 and HC-06 FC-114. First Look
HC-05 FC-114 and HC-06 FC-114. Part 2 – Basic AT commands
HC-05 FC-114 and HC-06 FC-114. Part 3 – Master Mode and Auto Connect

On the zs-040 modules there are 2 AT modes. I do not know if this is intentional but some commands only work when pin34 is HIGH. Other commands work when pin 34 is either HIGH or LOW. This fooled me for quite a while. For this post I have called the different modes “mini” AT mode and “full” AT mode.

- STATE - HIGH when module is connected;
- RXD - data input pin;
- TXD - data output pin;
- GND - ground pin;
- VCC - power input pin;
- EN - Unknown pin. It should be connected to pin 34 of EGBT-045MS Bluetooth module, but it seems it doesn't work in preferred way;



To activate AT mode on the HC-05 zs-040 modules we can:
– 1. Hold the small button switch closed while powering on the module.
– 2. Set pin 34 HIGH (3.3v) when power on.
– 3. Close the small push button switch after the HC-05 is powered.
– 4. Pull pin 34 HIGH after powering the HC-05.

Method 1.
Enters AT mode with the built in AT mode baud rate of 38400. The baud rate cannot be changed by the user.
This method allows the module to enter AT mode on start but but does not keep pin 34 HIGH and uses the “mini” AT mode.

Method 2.
Enters AT mode with the built in AT mode baud rate of 38400. The baud rate cannot be changed by the user.

If you keep pin 34 HIGH you will enable the “full” AT mode which allows all AT commands to be used.
If you let pin 34 return LOW after power on then “mini” AT mode will be enabled.

Method 3.*
Enters “mini” AT mode using the user defined communication mode baud rate.

Method 4.*
Enters “full” AT mode using the user defined communication mode baud rate.

If pin 34 is kept HIGH then the HC-05 enters the “full” AT mode. If pin 34 is brought HIGH and returned to LOW it will put the module in to “mini” AT mode.

* added 21.07.2015

Method 1 and 2 are good in that you know the baud rate – it will always be 38400. This could be useful if you have modules other people have used or if you forget what communication mode baud rate you have previously set.

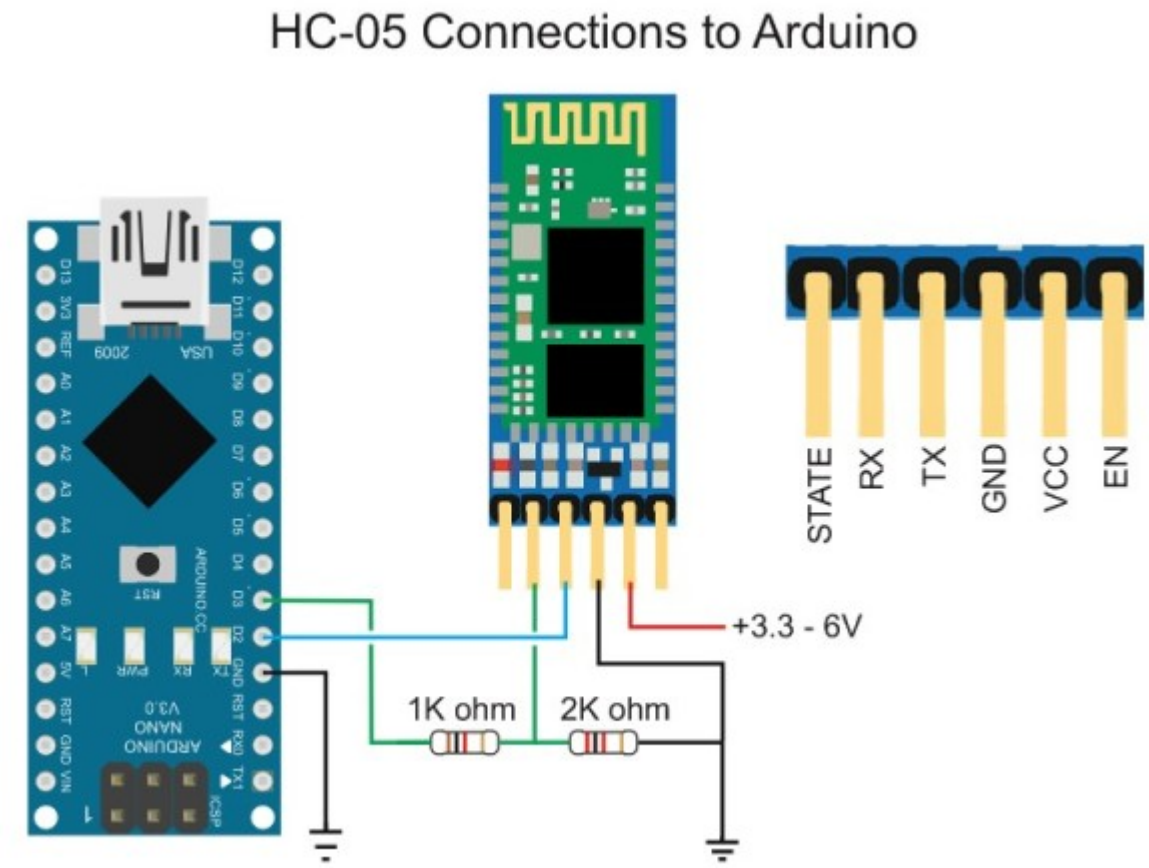
Method 3 and 4 adds convenience. You can enter AT mode, make changes and return back to communication mode without switching sketches and messing around with different baud rates.

I use software serial on Arduino pins 2 and 3 to talk to the HC-05. This means I can still use the hardware serial to talk to the serial monitor on a host computer.

Entering AT Mode Method 1. Use the button switch

The small push button switch, when closed, connects pin 34 to vcc which allows you to enter AT mode. Close the button switch when powering on the module and you enter AT mode using 38400 baud rate. Once on you can release the button switch, however, releasing the button switch puts the module in to a mini AT mode and some commands will not work. Commands such as AT+NAME?, AT+INQ, AT+RNAME? only work when pin 34 is HIGH. As soon as you release the button switch pin 34 returns LOW. If you want access to the extended commands you can simply close the switch just before issuing the AT command and release the button switch after the command has been sent.

Make the following connections
– BT VCC to Arduiono 5V
– BT GND to Arduino GND
– BT TX to Arduino D2
– BT RX to Arduino D3 through a voltage divider (3.3V)



Connect the Arduino to the host computer. The LED on the HC-05 should be blinking quickly at about 5 times a second.

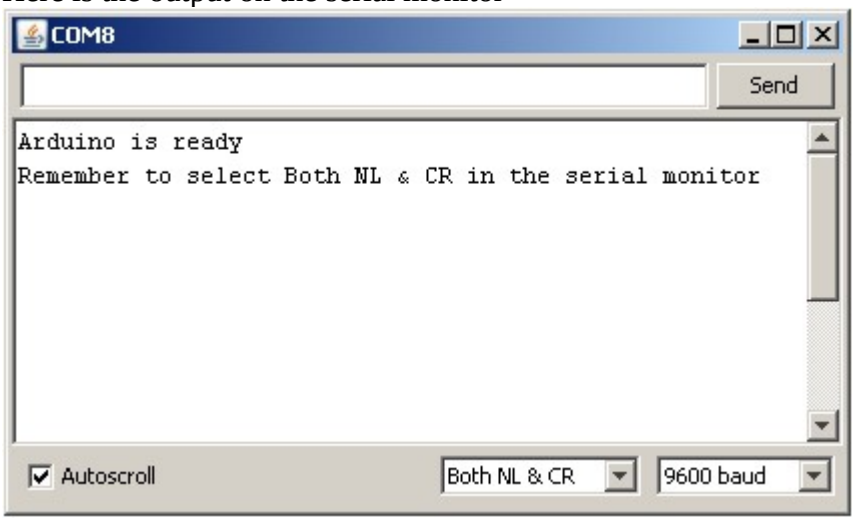
With the Arduino on, do the following
– Remove the 5V connection to BT VCC
– Press and hold the button switch on the BT module
– Re-connect BT VCC to 5V (while still pressing the button switch), the LED should come on.
– Release the button switch and the LED should be blinking slowly on/off once every couple of seconds. This indicates AT mode.

The following sketch is used to talk to the BT module. Run the sketch and put the HC-05 in to AT mode.

```
// Basic Bluetooth sketch HC-05_AT_MODE_01
// Communicate with a HC-05 using the serial monitor
//
// The HC-05 defaults to communication mode when first powered on you will need to manually enter AT mode
// The default baud rate for AT mode is 38400
// See www.martyncurrey.com for details
//


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

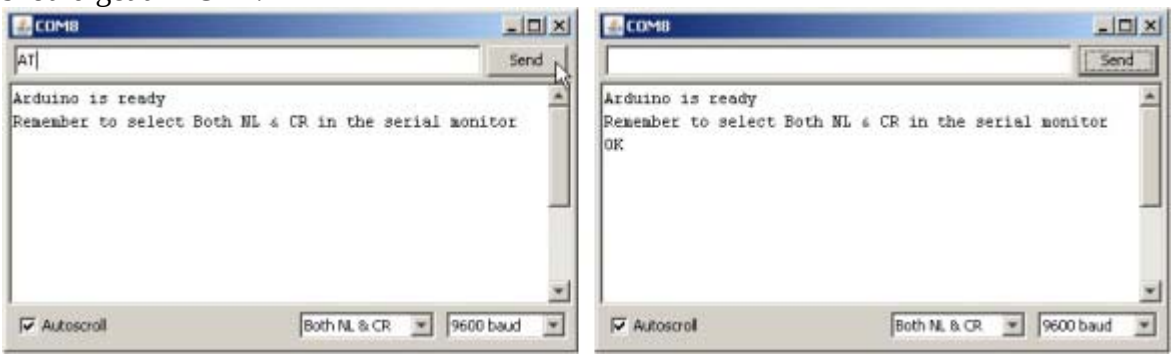
```
// Connect the HC-05 TX to Arduino pin 2 RX.
// Connect the HC-05 RX to Arduino pin 3 TX through a voltage divider.
//

char c = ' ';

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino is ready");
    Serial.println("Remember to select Both NL & CR in the serial
monitor");

    // HC-05 default serial speed for AT mode is 38400
    BTserial.begin(38400);
}

void loop()
{

    // Keep reading from HC-05 and send to Arduino Serial Monitor
    if (BTserial.available())
    {
        c = BTserial.read();
        Serial.write(c);
    }

    // Keep reading from Arduino Serial Monitor and send to HC-05
    if (Serial.available())
    {
        c =  Serial.read();
        BTserial.write(c);
    }

}
```

Here is the output on the serial monitor



The HC-05 expects a new line and a carriage return character at the end of each command so make sure “Both NL & CR” is selected at the bottom of the serial monitor. To confirm you are actually in AT mode, in the serial monitor type “AT” (no quotes) and hit Send. You

should get an “OK”.



Entering AT Mode Method 2. Using the Arduino to Control the HC-05

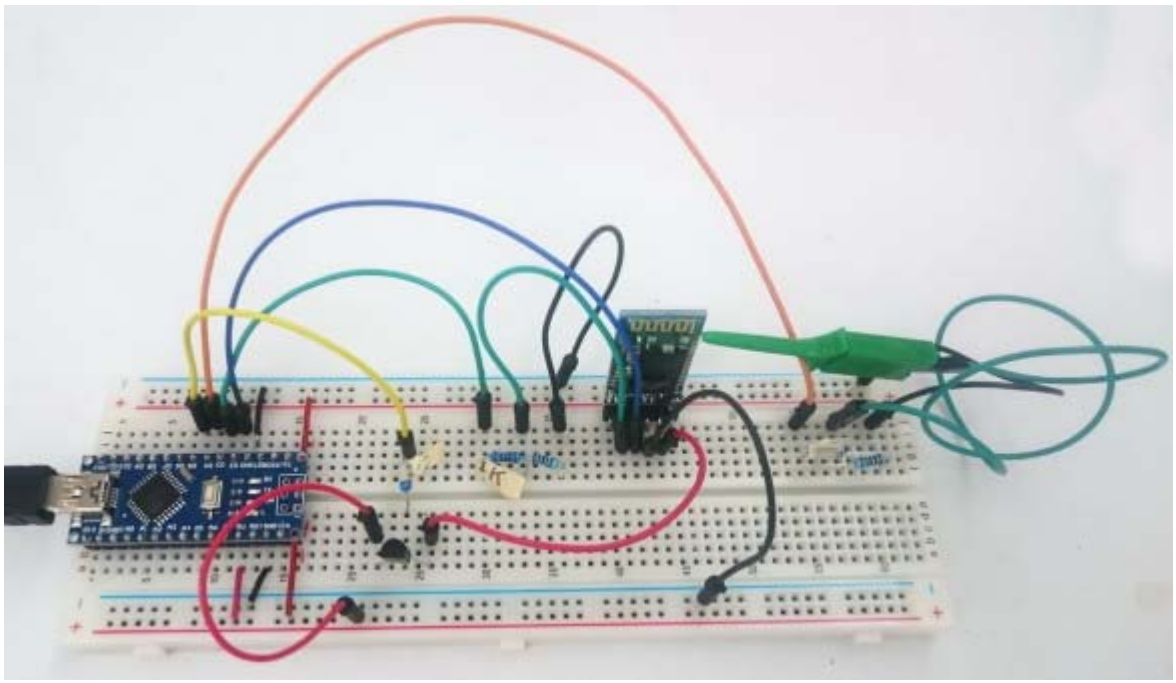
In this example the Arduino fully controls the HC-05. The Arduino pin D4 connects to a PNP transistor which is used as a switch to control the power and D5 is connected to the HC-05 pin 34 to control AT mode. Of course you can control the HC-05 manually if you wish.

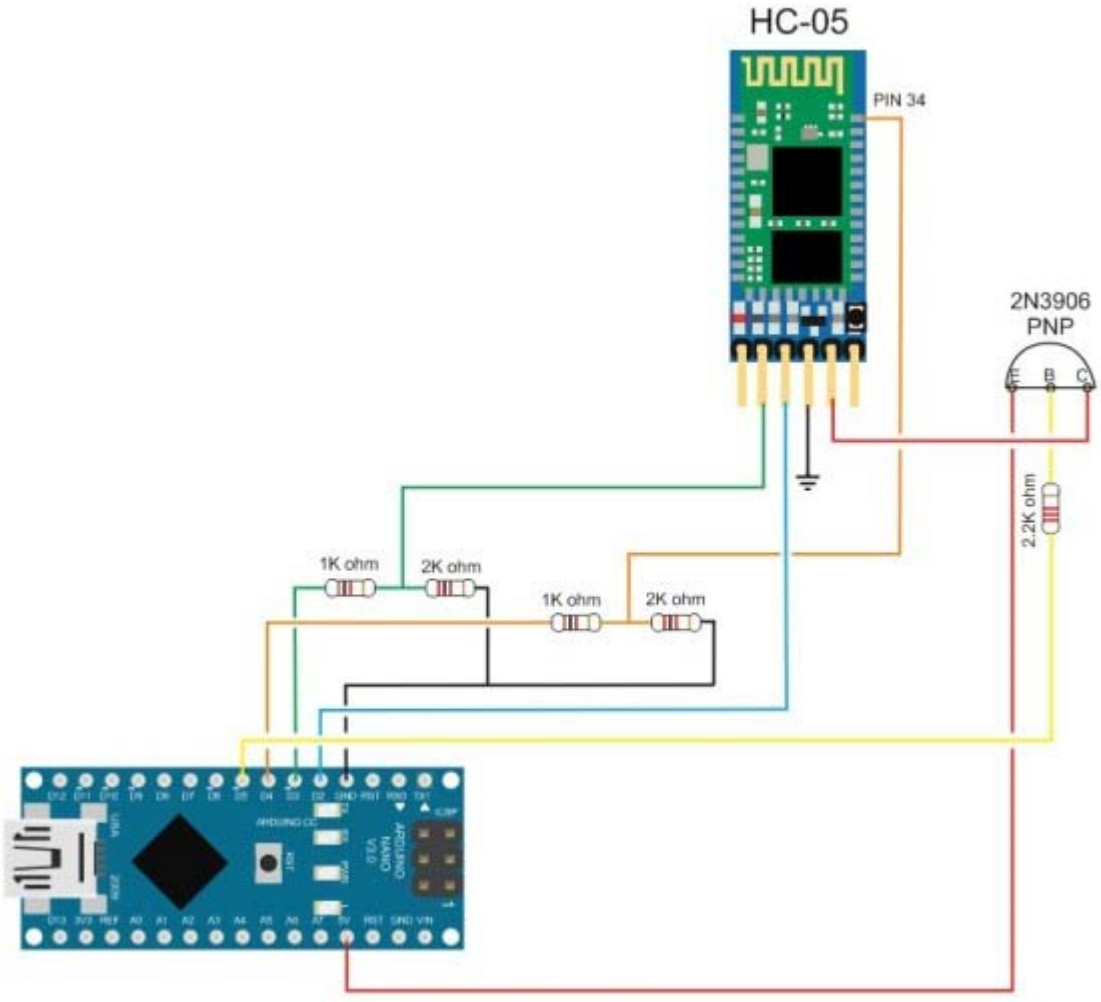
This used to be my preferred method but I have found that once I have set up a module I seldom change it and it has become more convenient to use the button switch or a temporary connection to pin 34. The benefit of this example is that the Arduino can control the process. Also, certain commands only work when pin 34 is HIGH. Using this method allows you to keep pin 34 HIGH.

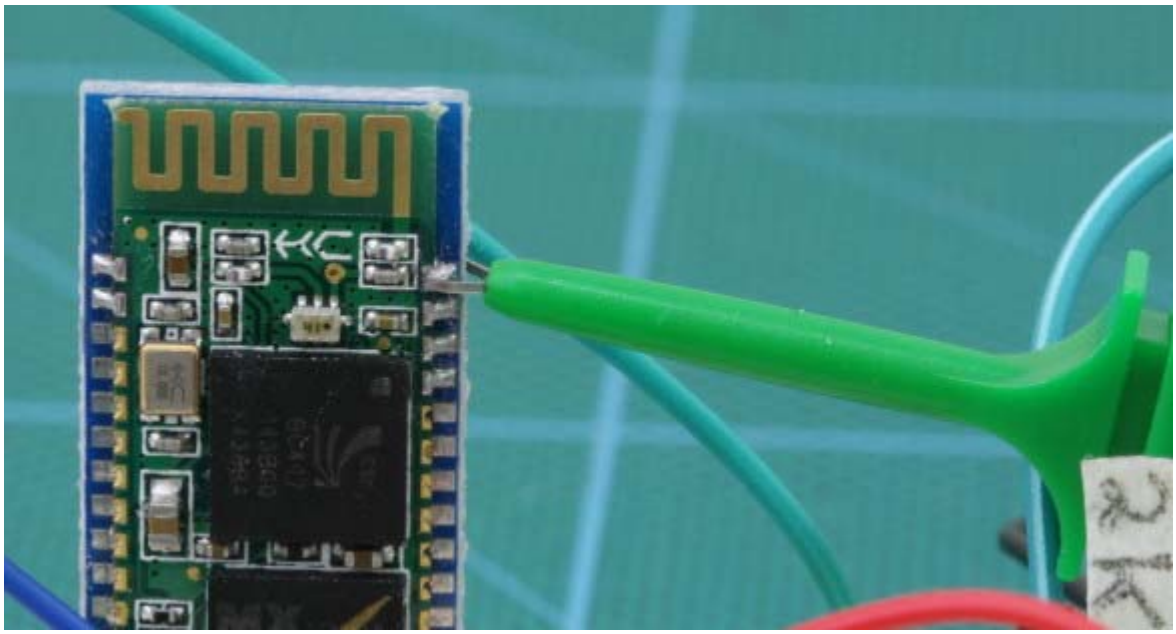
The HC-05 can draw a maximum of 40mA and although Arduino pins can supply 40mA this is the maximum and not recommended. The Arduinos 5V out pin can supply up to 200mA and is a much safer option and we use this to power the HC-05. We still need a regular pin to control the transistor which we are using to switch the power on and off.

With the Arduino turned off make the following connections:
– Arduino D2 to HC-05 TX
– Arduino D3 to HC-05 RX through a voltage divider
– Arduino D4 to HC-05 pin 34 through a voltage divider
– Arduino D5 to PNP transistor base via a 2.2k resistor
– HC-05 GND to common GND
– PNP emitter to +5V
– PNP collector to HC-05 vcc

I am using a 2N3906 PNP transistor because it is what I have. Similar PNP transistors can also be used.




HC-05
PIN 34
2N3906
PNP
E
B
C
2.2K ohm
1K ohm
2K ohm
1K ohm
2K ohm



HC-05 with pin 34 connected by a probe clip. You can also simply hold a piece of wire to the pin. For a long term solution you would need to solder a wire to pin 34.

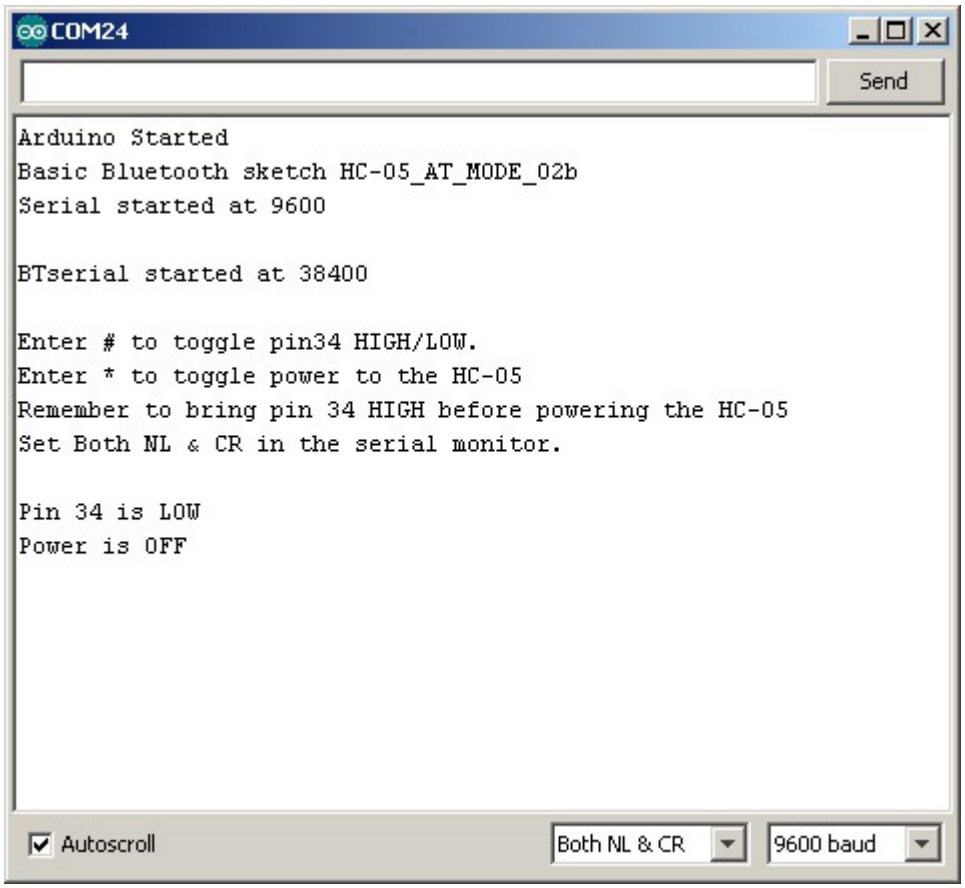
Compile and upload the following sketch

```
// Basic Bluetooth sketch HC-05_AT_MODE_02b
// Connect the HC-05 module and communicate using the serial monitor
// Arduino automates entering AT mode
//
// The default baud rate for AT mode when pin 34 is HIGH on power on is
38400
// See www.martyncurrey.com for details
//
//
//  Pins
//  Arduino D2 to HC-05 TX
//  Arduino D3 to HC-05 RX via a voltage divider
//  Arduino D4 to HX-05 pin 34 via a voltage divider
//  Arduino D5 to PNP Base via a 2.2k resistor
//  BT VCC to PNP Collector
//  BT GND to GND
//  PNP Emitter to vcc
//
// When a command is entered in to the serial monitor on the computer
// the Arduino will relay it to the Bluetooth module and display the
result.
//

const byte BT_POWERPIN = 5;
const byte BT_PIN34_PIN = 4;

const boolean ON = LOW;
const boolean OFF = HIGH;

boolean BT_POWER = HIGH;
boolean PIN34_STATE = LOW;

boolean NL = true;
char c = ' ';
```

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

void setup()
{
    pinMode(BT_POWERPIN, OUTPUT);
    pinMode(BT_PIN34_PIN, OUTPUT);


    digitalWrite(BT_POWERPIN, OFF);
    digitalWrite(BT_PIN34_PIN, LOW);


    Serial.begin(9600);     // communication with the host computer
    //while (!Serial)   { ; }

    Serial.println("Arduino Started");
    Serial.println("Basic Bluetooth sketch HC-05_AT_MODE_02b");


    Serial.println("Serial started at 9600");
    Serial.println(" ");

    // Start the software serial - baud rate for AT mode is 38400
    BTserial.begin(38400);
    Serial.println("BTserial started at 38400");
    Serial.println(" ");

    Serial.println("Enter # to toggle pin34 HIGH/LOW.");
    Serial.println("Enter * to toggle power to the HC-05");
    Serial.println("Remember to bring pin 34 HIGH before powering the HC-
05");
    Serial.println("Set Both NL & CR in the serial monitor.");
    Serial.println(" ");
    Serial.println("Pin 34 is LOW");
    Serial.println("Power is OFF");
    Serial.println(" ");

}



void loop()
{

    // Keep reading from Arduino Serial Monitor and send to HC-05
    if (Serial.available())
    {
          c =  Serial.read();

          // Echo the user input to the main window. The ">" character
indicates the user entered text.
          if (NL) { Serial.print(">");  NL = false; }
          Serial.write(c);
          if (c==10) { NL = true; }


          if (c=='*')
          {
```

```
                Serial.println(""); Serial.print("BT power is ");
                if (BT_POWER == ON)         { BT_POWER = OFF;
digitalWrite(BT_POWERPIN, OFF);   Serial.println("OFF");    }
                else if  (BT_POWER == OFF)  { BT_POWER = ON;
digitalWrite(BT_POWERPIN, ON);    Serial.println("ON");   }

                if (  (PIN34_STATE == HIGH) && (BT_POWER == ON)  ) {
Serial.println("AT mode");  }
          }

          else if (c=='#')
          {
                Serial.println(""); Serial.print("BT pin 34 is ");
                if (PIN34_STATE == LOW)           { PIN34_STATE = HIGH;
digitalWrite(BT_PIN34_PIN, HIGH);  Serial.println("HIGH");   }
                else  if (PIN34_STATE == HIGH)    { PIN34_STATE = LOW;
digitalWrite(BT_PIN34_PIN, LOW);   Serial.println("LOW");    }
          }

          else { BTserial.write(c);  }

    }



  // Keep reading from the HC-05 and send to the Arduino Serial Monitor
    if (BTserial.available())
    {
        c = BTserial.read();
        Serial.write(c);
    }


}
```

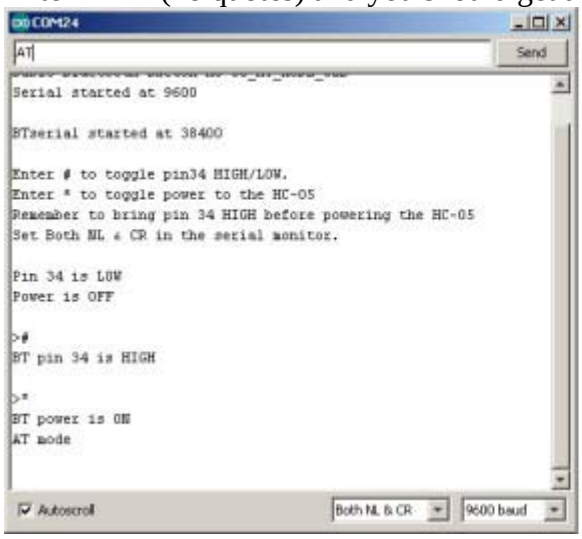
After the sketch has been uploaded, started the Arduino and open the serial monitor at 9600 baud rate. The HC-05 should be off.

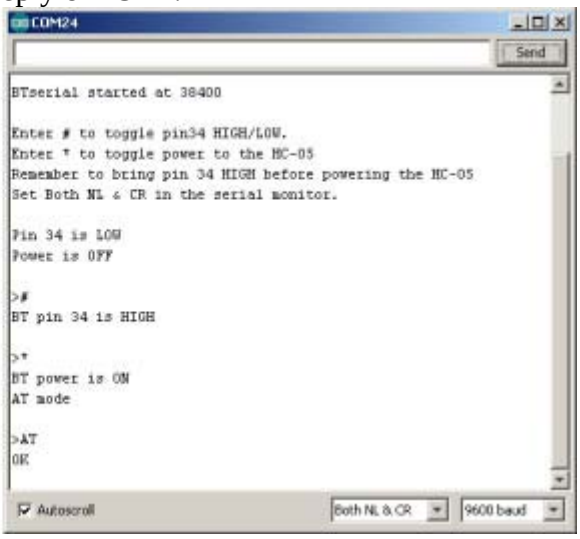


To start the HC-05 in AT mode, bring pin 34 HIGH and then start the BT module.
– Enter “#” (no quotes) to bring pin 34 HIGH
– Enter “*” (no quotes) to turn on the HC-05

The HC-05 should now be on and in AT mode with the on board LED blinking slowly on/off every couple of seconds.

Enter “AT” (no quotes) and you should get a reply of “OK”.



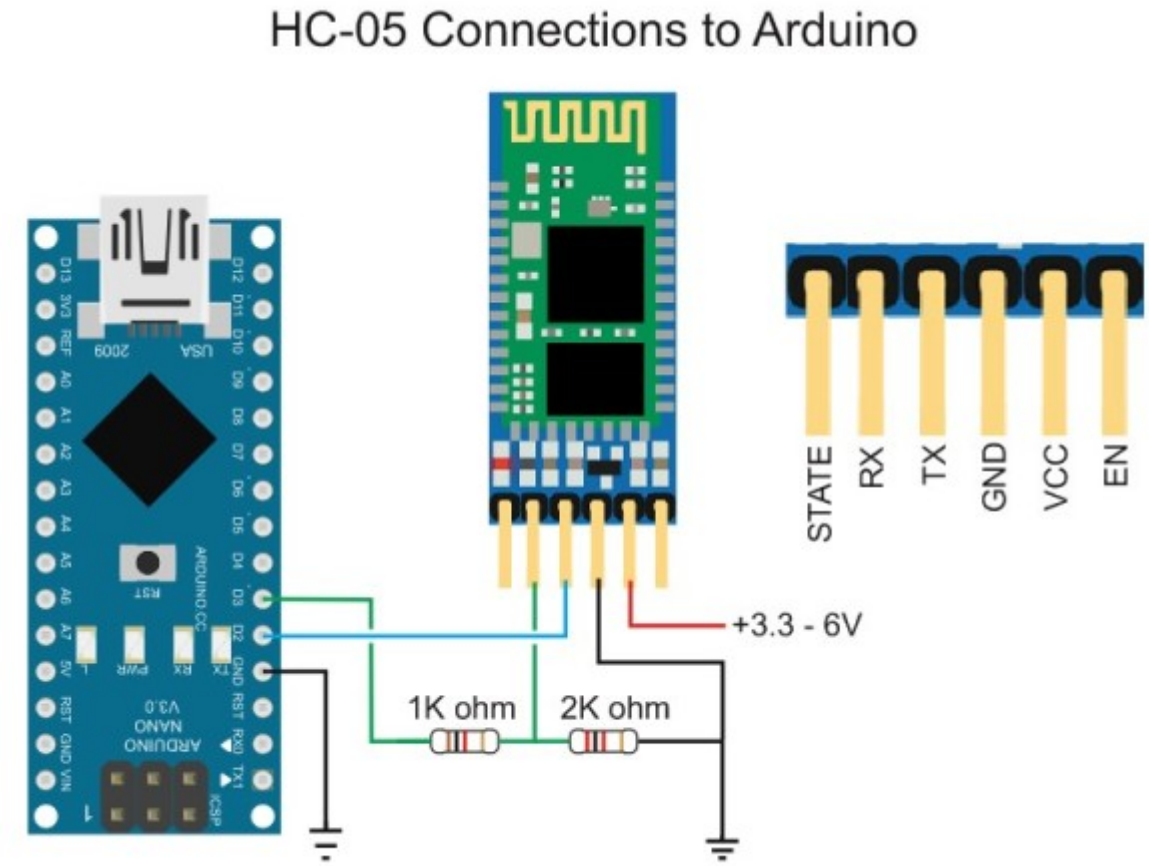


It should be fairly simple to extend this example and fully control the HC-05 from your own sketch.

Entering AT Mode Method 3. Close the small push button switch after the HC-05 is powered.

This method enters the “mini” AT mode using the baud rate defined for communication mode. This means you can use communication mode and enter AT mode without uploading a new sketch but not all commands will work.

Set up the HC-05 as below:



Upload the following sketch.

```
// Basic Bluetooth sketch HC-05_02_9600+ECHO
// Connect the HC-05 module and communicate using the serial monitor
//
// The HC-05 defaults to commincation mode when first powered on.
// The default baud rate for communication mode is 9600. Your module may
have a different speed.
//

#include <SoftwareSerial.h>
SoftwareSerial BTserial(2, 3); // RX | TX
// Connect the HC-05 TX to Arduino pin 2 RX.
// Connect the HC-05 RX to Arduino pin 3 TX through a voltage divider.

char c = ' ';

void setup()
{
    Serial.begin(9600);
    Serial.println("Arduino is ready");

    // HC-05 default serial speed for communication mode is 9600
    BTserial.begin(9600);
    Serial.println("BTserial started at 9600");
}

void loop()
```

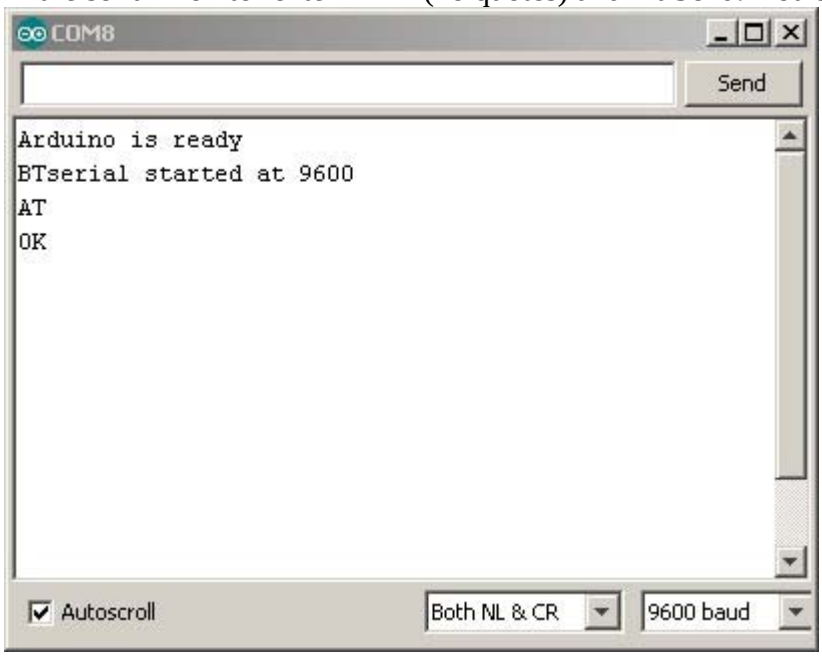
```
{
    // Keep reading from HC-05 and send to Arduino Serial Monitor
    if (BTserial.available())
    {
        c = BTserial.read();
        Serial.write(c);
    }

    // Keep reading from Arduino Serial Monitor and send to HC-05
    if (Serial.available())
    {
        c =  Serial.read();

        // Copy the serial data back to to the serial monitor.
        // This makes it easy to follow the commands and replies
        Serial.write(c);
        BTserial.write(c);
    }

}
```

The HC-05 should be in communication mode with the LED on the HC-05 blinking about 5 times a second. This indicates the module is waiting for a connection or to be paired.

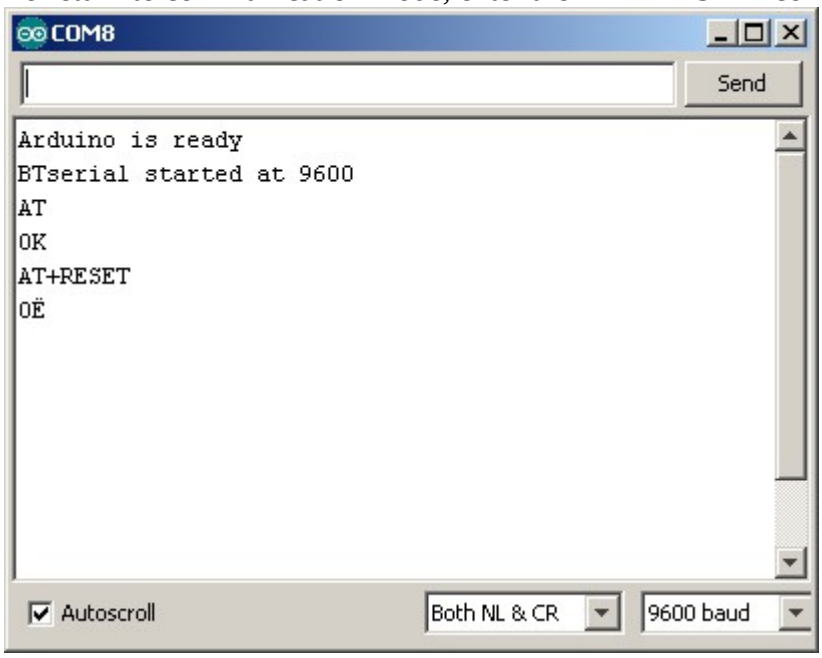
Open the serial monitor.
Press the small button switch.
Release the small button switch.

That's it. You are now in "mini" AT mode. The LED does not change. It still blinks quickly at 5 times a second.

In the serial monitor enter "AT" (no quotes) and hit Send. You should get an "OK"



To return to communication mode, enter the “AT+RESET” command.

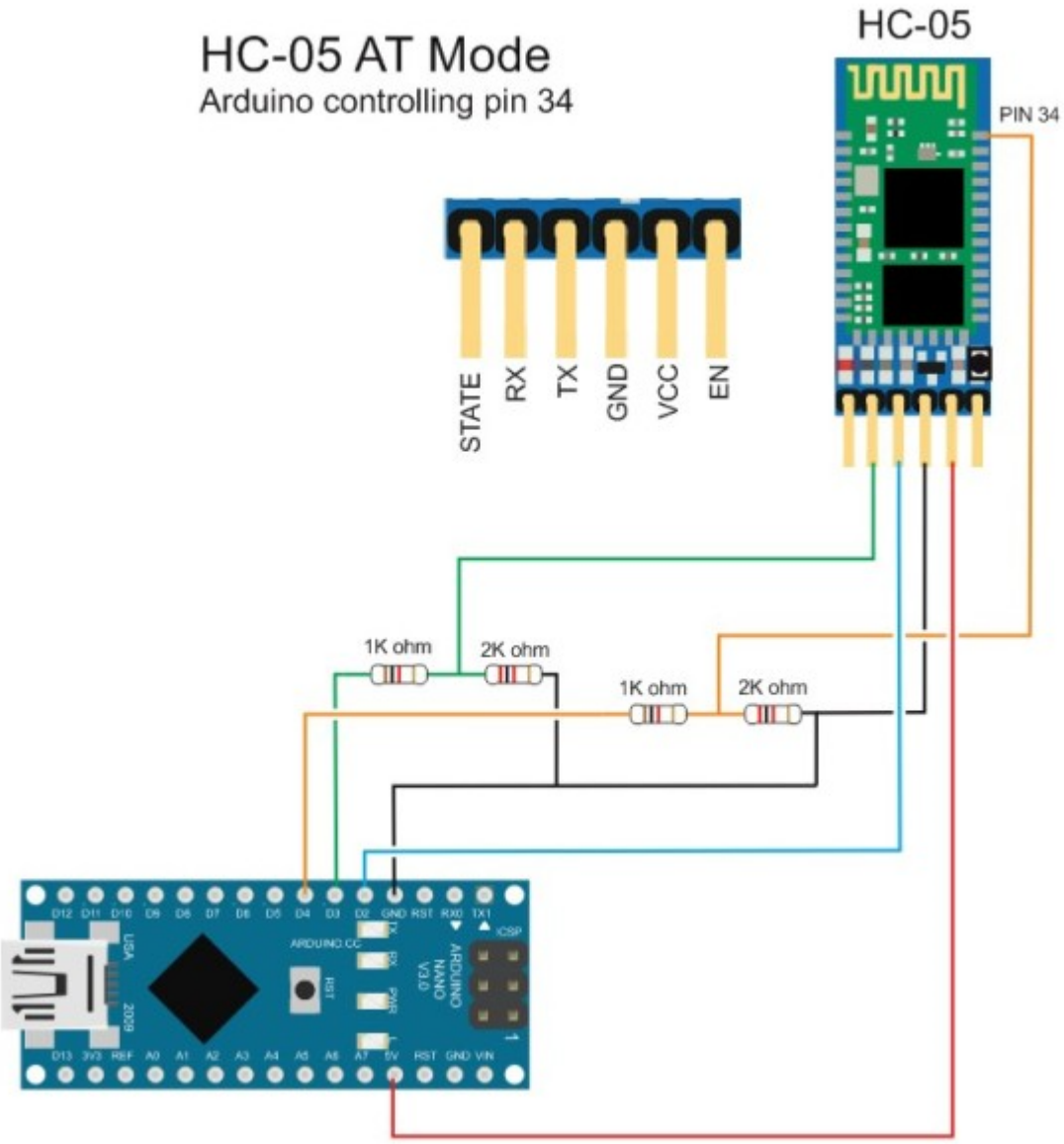


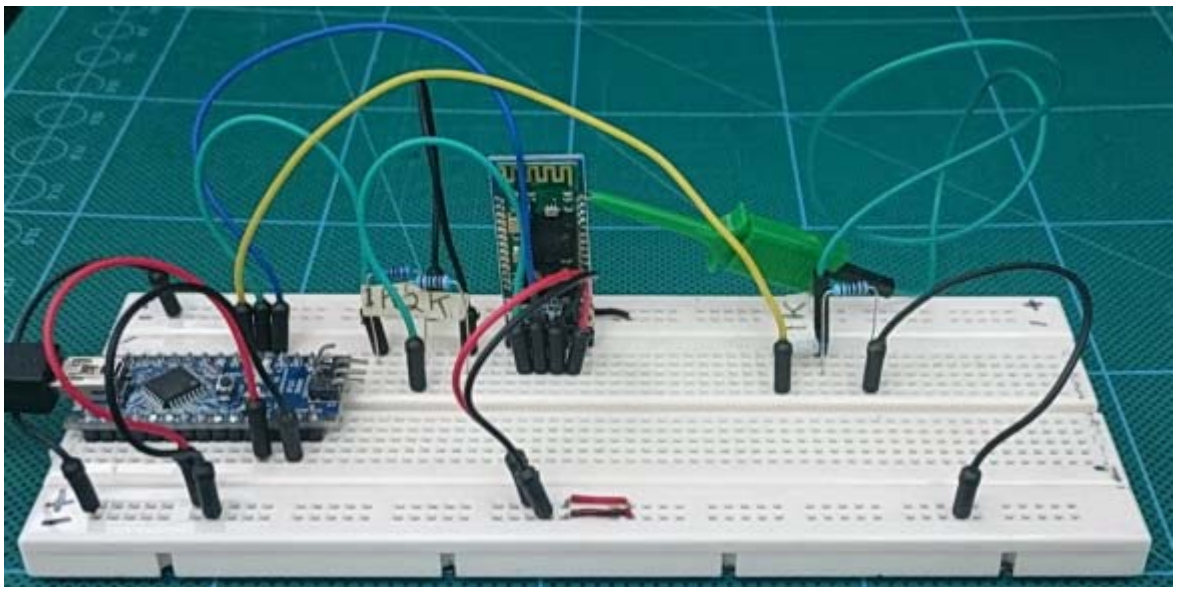
Entering AT Mode Method 4. Pull pin 34 HIGH after powering the HC-05.

This method allows you to enter and exit AT mode from within a sketch. This can be handy when you run a set up sketch or when you want the user to be able to make changes. For example change the name of the HC-05. The sketch could ask the user for the new name and then set it without the need to change sketches or any manual operation from the user.

Because pin 34 is being pulled HIGH the HC-05 will enter “full” AT mode and all commands will work.

Make the following connections
– D2 (RX) to HC-05 TX
– D3 (TX) to voltage divider and then HC-05 RX
– D4 to voltage divider and then to pin 34
– HC-05 vcc to 5v
– HC-05 GND to common GND


HC-05 AT Mode
Arduino controlling pin 34
HC-05
PIN 34
STATE
RX
TX
GND
VCC
EN
1K ohm
2K ohm
1K ohm
2K ohm



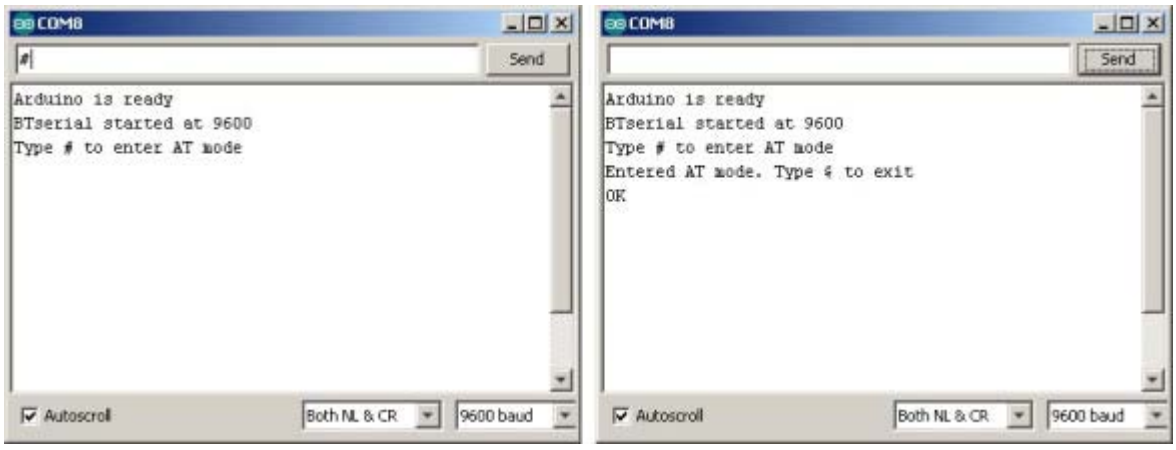
Upload the following sketch. Once the sketch is uploaded open the serial monitor.

```
// Basic Bluetooth sketch HC-05_03_AT_MODE_Controlled
// Connect the HC-05 module and communicate using the serial monitor
//
// The HC-05 defaults to communication mode when first powered on.
// The default baud rate for communication mode is 9600. Your module may
have a different speed.
// This sketch allows the user to enter AT mode on the HC-05
//
//
// Pins
// D2 (RX) to HC-05 TX
// D3 (TX) to voltage divider and then HC-05 RX
// D4 to voltage divider and then to pin 34
// HC-05 vcc to 5v
// HC-05 GND to common GND
//

#include <SoftwareSerial.h>
SoftwareSerial BTserial(2, 3); // RX | TX
// Connect the HC-05 TX to Arduino pin 2 RX.
// Connect the HC-05 RX to Arduino pin 3 TX through a voltage divider.

char c = ' ';
byte ATmodePin=4;

void setup()
{
    // set up the pin used to turn on AT mode
    pinMode(ATmodePin, OUTPUT);
    digitalWrite(ATmodePin, LOW);

    // Start the serial monitor
    Serial.begin(9600);
    Serial.println("Arduino is ready");

    // HC-05 default serial speed for communication mode is 9600
    BTserial.begin(9600);
    Serial.println("BTserial started at 9600");
    Serial.println("Type # to enter AT mode");
}

void loop()
{

    // Keep reading from HC-05 and send to Arduino Serial Monitor
    if (BTserial.available())
    {
        c = BTserial.read();
        Serial.write(c);
    }

    // Keep reading from Arduino Serial Monitor and send to HC-05
    if (Serial.available())
    {
        c =  Serial.read();

        if (c=='#')  // enter AT mode
        {
```

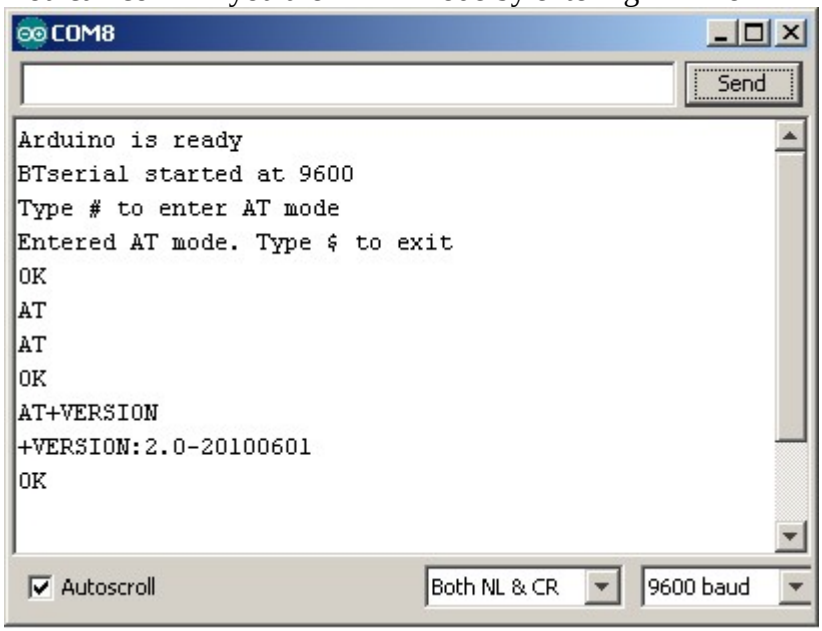
```
            digitalWrite(ATmodePin, HIGH);
            Serial.print("Entered AT mode. Type $ to exit");
        }

        else if (c=='$') // exit AT mode by reseting the HC-05
        {
            digitalWrite(ATmodePin, LOW);
            BTserial.print("AT+RESET\n\r");
            Serial.print("AT+RESET\n\r");
        }

        else
        {
              // Copy the serial data back to to the serial monitor.
              // This makes it easy to follow the commands and replies
              Serial.write(c);
              BTserial.write(c);
        }

    }

}
```

The sketch is similar to the other HC-05 sketches, it copies what is entered in to the serial monitor to the HC-05 and what ever it receives from the HC-05 it sends to the serial monitor. The extra bit is # and $. Entering a “#” (no quotes) puts the HC-05 in to AT mode and entering a “$” (no quotes) returns to communication mode. It returns the HC-05 to communication mode by resetting the module.

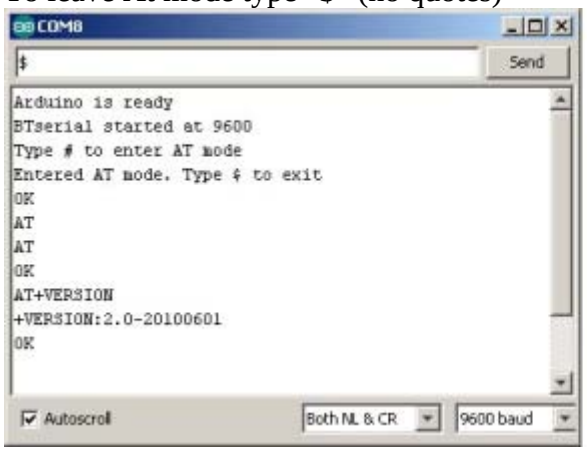
To enter AT mode type “#”
To exit AT mode type “$”

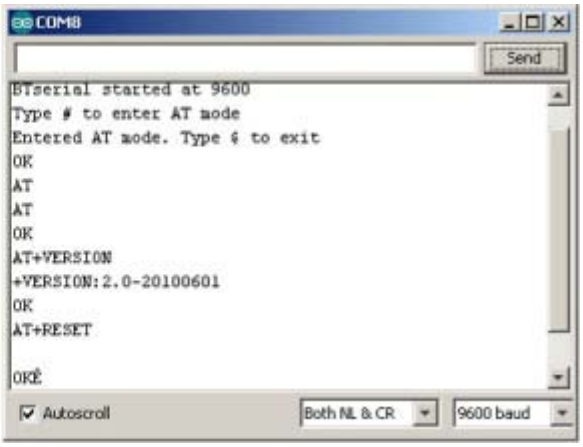


You can confirm you are in AT mode by entering “AT” or “AT+VERSION”



To leave At mode type “$” (no quotes)





AT commands

The HC-05 expects commands to include a carriage return and newline characters (\r\n). You can add these automatically in the serial monitor by selecting Both NL & CR at the bottom of the window.

You can also enter them manually in the form AT\r\n. If you forget to add carriage return and newline characters the HC-05 will not respond.

Example commands

AT – simple feedback request. Will return
OK

AT+VERSION – returns the firmware version. In my case it returns
+VERSION:2.0-20100601
OK

AT+STATE – returns the current state of the module
+STATE:INITIALIZED
OK

AT+ROLE – the possible values are ; 0 – Slave, 1 – Master, 2 – Slave-Loop
Returns
+ROLE:0
OK

To change to Master Mode, enter AT+ROLE=1, returns:
OK

AT+UART – returns the baud rate used by the HC-05 in communication mode. The default for the modules I have is 9600.
Returns:
+UART:9600,0,0
OK

To change the baud rate to 38400 – AT+UART=38400,0,0
Returns:
OK

Windows does not support baud rates above 115200. If you accidentally set the baud rate higher than 115200 you will not be able to use communication mode. You should still be able to enter AT mode at 38400 using method 1 or method 2 above and change the communication mode baud rate to something Windows can handle.

AT+NAME
Querying the modules name with AT+NAME? only works in “full” At mode. If you cannot get AT+NAME? to work you need to bring pin34 HIGH.
Changing the modules name with AT+NAME=newname works in “full” AT mode and “mini” AT mode.

What you should get is:
AT+NAME?, returns
+NAME:HC-05
OK
(or something similar depending what your module is called)

Other commands that require pin 34 to be HIGH are AT+INQ and AT+RNAME. This is not a complete list through.

Full list of AT commands

AT COMMAND LISTING

	COMMAND	FUNCTION
1	AT	Test UART Connection
2	AT+RESET	Reset Device
3	AT+VERSION	Query firmware version
4	AT+ORGL	Restore settings to Factory Defaults
5	AT+ADDR	Query Device Bluetooth Address
6	AT+NAME	Query/Set Device Name
7	AT+RNAME	Query Remote Bluetooth Device's Name
8	AT+ROLE	Query/Set Device Role
9	AT+CLASS	Query/Set Class of Device (CoD
10	AT+IAC	Query/Set Inquire Access Code
11	AT+IMOD	Query/Set Inquire Access Mode
12	AT+PSWD	Query/Set Pairing Passkey
13	AT+UART	Query/Set UART parameter
14	AT+CMODE	Query/Set Connection Mode
15	AT+BIND	Query/Set Binding Bluetooth Address
16	AT+POLAR	Query/Set LED Output Polarity
17	AT+PIO	Set/Reset a User I/O pin
18	AT+MPIO	Set/Reset multiple User I/O pin
19	AT+MPIOY	Query User I/O pin
20	AT+PSCAN	Query/Set Scanning Parameters
21	AT+SNIFF	Query/Set SNIFF Energy Savings Parameters
22	AT+SENM	Query/Set Security & Encryption Modes
23	AT+RMAAD	Delete Authenticated Device from List
24	AT+FSAD	Find Device from Authenticated Device List
25	AT+ADCN	Query Total Number of Device from Authenticated Device List
26	AT+MRAD	Query Most Recently Used Authenticated Device
27	AT+STATE	Query Current Status of the Device
28	AT+INIT	Initialize SPP Profile
29	AT+INQ	Query Nearby Discoverable Devices
30	AT+INQC	Cancel Search for Discoverable Devices
31	AT+PAIR	Device Pairing
32	AT+LINK	Connect to a Remote Device
33	AT+DISC	Disconnect from a Remote Device
34	AT+ENSNIFF	Enter Energy Saving mode
35	AT+EXSNIFF	Exit Energy Saving mode

ERROR CODES

ERROR CODE	VERBOSE
0	Command Error/Invalid Command
1	Results in default value
2	PSKEY write error
3	Device name is too long (>32 characters)
4	No device name specified (0 length)
5	Bluetooth address NAP is too long
6	Bluetooth address SAP is too long
7	Bluetooth address UAP is too long
8	PIO mask not specified (0 length)
9	Invalid PIO port Number entered
A	Device Class not specified (0 length)
B	Device Class too long
C	Inquire Access Code not Specified (0 length)
D	Inquire Access Code too long
E	Invalid Inquire Access Code entered
F	Pairing Password not specified (0 length)
10	Pairing Password too long (> 16 characters)
11	Invalid Role entered
12	Invalid Baud Rate entered
13	Invalid Stop Bit entered
14	Invalid Parity Bit entered
15	No device in the Pairing List
16	SPP not initialized
17	SPP already initialized
18	Invalid Inquiry Mode
19	Inquiry Timeout occurred
1A	Invalid/non-existent address entered
1B	Invalid Security Mode entered
1C	Invalid Encryption Mode entered

This list is taken from the EGBT-045MS bluetooth module user guide and not all commands may be supported or work straight away. For example AT+NAME? only works when pin 34 is HIGH.

For more information look at the HC-05 user guide or the EGBT-046S/EGBT-045MS user guide