**A CHILD'S GUIDE to BASIC TWO WAY BLUETOOTH COMMUNICATION BETWEEN ARDUINO AND ANDROID.**

version 10 (All brickbats and bouquets gladly received)

I put this up as a discussion on getting a simple bluetooth connection. I use it for simple datalogging purposes but the notes below provide for two way communication.

I had a bad time getting started with Bluetooth and had to make notes for myself, from which the following is derived. Bluetooth with Arduino is a good combination that is chronically misunderstood, and plagued with misinformation. The following is an attempt to get past those misunderstandings and misinformation, and instill some badly needed confidence.

While I wrote this with Android in mind, virtually everything below applies equally to Windows. Indeed all my learning was done on a Dell XP laptop with the standard Toshiba bluetooth sniffer and running RealTerm. The Android came later, and I only got it because of the success with the Dell.

## 1. THE OPERATING ENVIRONMENT.

The objective is to be able to stand near the Arduino and casually aquire live data. The equipment is claimed to work over 10m. I have used it over 15m with clear line of sight. One wall of lightweight domestic construction will cut the range to about 5m maximum, and a single layer of foil building insulation can kill it stone dead. This last can mean that indoor to outdoor communication could be pretty risky.

## 2. EQUIPMENT USED

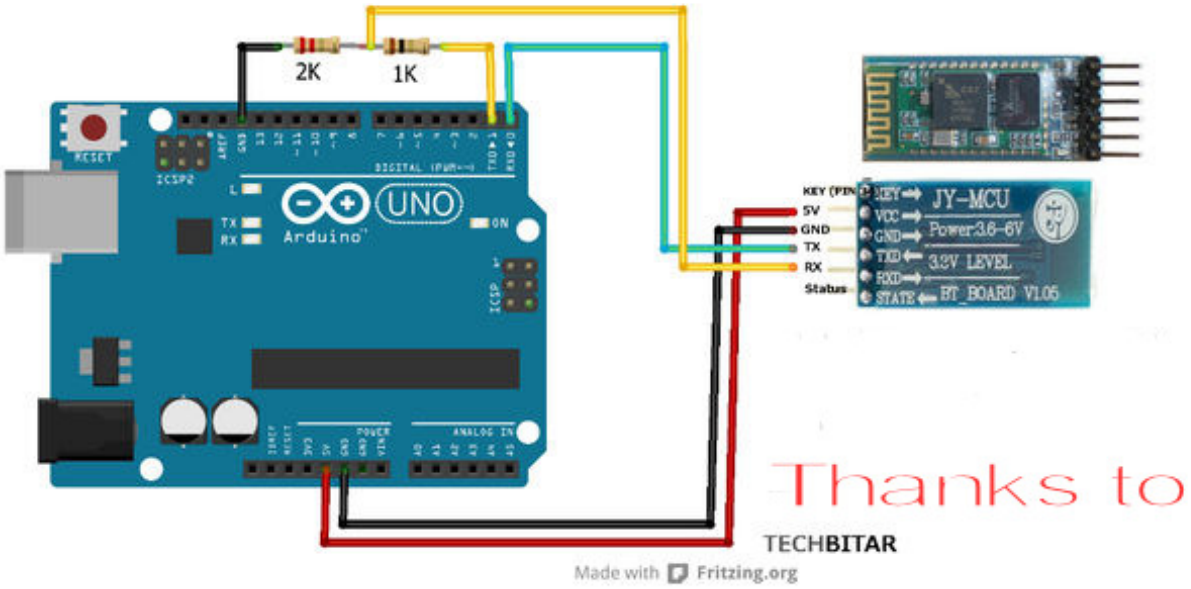
1.
A standard Arduino Uno or Mega. Any 5volt Arduino should suffice.

2.
An HC-05 or HC-06 bluetooth module on a JY-MCU backboard. These devices are available for about $7 on eBay and you need some pretty persuasive reasons to pay any more money than that.

The two types are visually indistinguishable. The HC-06 operates as a slave only but is entirely suitable for this exercise. The HC-05 can operate as a master and thus has more commands. I don't think there is much difference in the price, and its extra versatility may be of value in the future.

3.
A means of connection. I use a four-conductor cable to a header on a proto shield. A breadboard lashup would suffice, or female-male leads direct into the Arduino headers. You could solder the module directly into a proto shield. In this event, it would be wise to have a jumper in the 5v line so that Bluetooth can be isolated while the code is uploaded.

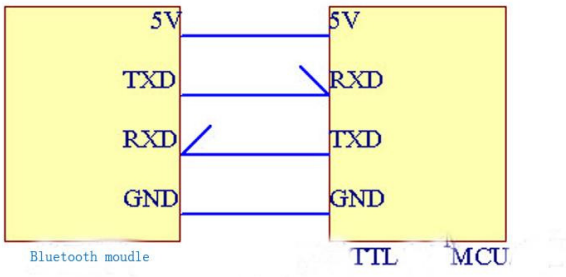
Note also that the HC-0x is a 3.3v device and, while it is clear that the JY-MCU handles the 5v power supply, some level shifting on the Tx line also is a wise precaution, even though I have not done it myself – yet. This may be as simple as running 1k and 2k between Tx and ground. The picture shows an example of this




Finally,
4.
An Android device. I use a Huawei  Y210. This is cheap, cheerful, and just fine for the job.

### 3. THE COMMUNICATION METHOD AND THE CONNECTIONS  REQUIRED

This is all about using the standard serial protocol using hardware, using pins D0 and D1 on the Arduino, which are clearly marked for the purpose.  Pin D0, Rx, is the receiver and therefore connected to the Tx pin on Bluetooth. This means D1 is connected to Rx on Bluetooth.




Correct connection

This is not about the alternative procedure, known as “software serial” so, if in the unlikely event that you really do have a need to use software serial for Bluetooth, read no further.

The only other connections required are the standard 5v and ground. I believe all HC-05s come with six pins. You don’t need the other two for this exercise. And yes, the HC-0x modules are 3.3v devices but note that the JY-MCU package level shifters on board for the power supply and is clearly labelled for 3.6 to 6v  operation.  If your Arduino is made for 3.3v operation, note that they have been known to work, but I wouldn’t trust it.

Note particularly that, while Bluetooth modules come in two types, master and slave, these characteristics are entirely irrelevant to this exercise, and the words will not be mentioned again until the appendix at the end – a section that you don’t need to read.

## 4. NOTES ON WHAT THE BLUETOOTH MODULE IS ABOUT

The bluetooth is a separate device between Arduino and Android. To the Arduino, it is just another serial device, indeed it is indistinguishable from the serial monitor and is used in the same way. To the Android, it is just another Bluetooth device to be paired with, and the fact that there is an Arduino connected to it is immaterial. What this particularly means is:

1.
Arduino is not involved with the pairing. It is just providing the power and, if there was another source of power, it needn't be connected.

2.
Consequently, a successful pairing is just between Bluetooth and Android, and does not guarantee successful communication with the Arduino.

3.
Similarly, the serial communication between Bluetooth and Arduino does not guarantee successful communication with Android. Note that there is no way of checking communication from Arduino to Bluetooth other than getting the signal all the way to Android.

4.
All the procedure for pairing and establishing connection is done at the Android end.

## 5. VITAL KNOWLEDGE

1.
The **big secret** about Arduino to Android communication via Bluetooth is:

**there is no big secret**.

2.
You can prove the code is kosher without the Bluetooth being connected. Remember this. It can do wonders for your confidence when the connection doesn't work, but the really important bit you need to realise is that you can do this because

**there are no software requirements or special commands for running bluetooth.**

This means **no libraries**. It is vital that you understand that it is just another serial device and the serial facility you already have is all you need.

3.
Bluetooth will work as it comes out of the box. Don't be tempted to fiddle with the configuration, but note that you do need to know the baud rate of Bluetooth, and set Arduino accordingly. It is normally 9600.

4.
In the light of item 3, you don't need to be able to read or understand the data sheets (other than know the baud rate!). This should be quite comforting to know, as every data sheet I have on these devices has been hard to find, hard to read, and hard to understand.

5.
The serial connection settings are for just that – the serial connection between Arduino and Bluetooth. The connection between Android and Bluetooth is strictly Bluetooth

communication that has its own speed completely independent of the serial speed, hence these setting are made in Arduino to match Bluetooth, not Android.

6.
A popular mistake is to try loading and testing your programme and having the bluetooth connection with the same computer. This doesn't work because bluetooth and serial monitor share the same port on Arduino, and the same code. Note item 2. above

7.
Note, as final boost to confidence, that it is simpler to use bluetooth with an Android than a PC. No configuration means a more easily assured connection – no grey areas.

8.
While I have tried to make this all look painless, I have to admit to experiencing mysterious problems but I'm disinclined to blame Bluetooth or Arduino. There is further discussion in the appendix. Having routine operations with both the Android phone and XP laptop proves this project is essentially kosher.

## 6. THE LED

The only relationship between the LED on Bluetooth and your Arduino is that, if the LED shows nothing, there is no power. If the LED is flashing about 2Hz, the device is ready, awaiting connection. When a connection is made by the Android, the LED goes permanently on. Note, again, that this only means Android is connected to Bluetooth.

## 7. THE BASIC TEST CODE

```
void setup()
{
   Serial.begin(9600);   //note this may need to be changed to match your module
   Serial.println("OK then, you first, say something.....");
   Serial.println("Go on, type something in the space above and hit Send,");
   Serial.println("or just hit the Enter key");
}

void loop()
{
  while(Serial.available()==0)
  {}
   Serial.println("");
  Serial.println("I heard you say:");
  while(Serial.available()>0)
  {
    Serial.write(Serial.read());//         note it is Serial.WRITE
  }
  Serial.println("");
}
```

Yes, folks, that is all there is to it, and all you need

http://homepages.ihug.com.au/~npyner/Arduino/BT2way.ino

This code is as basic as possible and serves no purpose other than to prove you have what you really need – a two way connection via bluetooth between your Arduino and a bluetooth-enabled device. I didn't write it, and no claims are made for sophistication. The format varies with the terminal programme at the other end and doesn't merit manipulation since, once you have it all working properly, it's time to move on to something more productive. A single letter entered is enough to prove the point.

The setup calls the serial facility and prints two lines on the screen. You only see them sent when you call up the serial monitor or press the reset button on Arduino.

Arduino then simply runs round the loop trying to pick up something from Bluetooth. If something is received, it says "I heard you say", and repeats the information back.

Upload this with **Bluetooth disconnected**, and test with serial monitor.

Hopefully, you can upload this and have it work first time up on your serial monitor.

## 8. UPLOADING PROBLEMS "COM PORT IN USE"

This is invariably caused by trying to upload when Bluetooth is connected, and usually when making a small adjustment after the programme has been working. It can get **very tedious**, but not tedious enough to start using software serial. If it happens, you need to shut down the IDE and Arduino. Reconnect Arduino before opening the IDE so that the COM port is ready to be found. Try to do as much testing as you can using the serial monitor, even if this means some temporary formatting.

## 9. ESTABLISHING BLUETOOTH CONNECTION

Here is the moment of truth. **Connect Bluetooth to Arduino**.

You should see the LED **flashing**.

You now need to **set up Bluetooth in the Android**, using the bluetooth settings app.

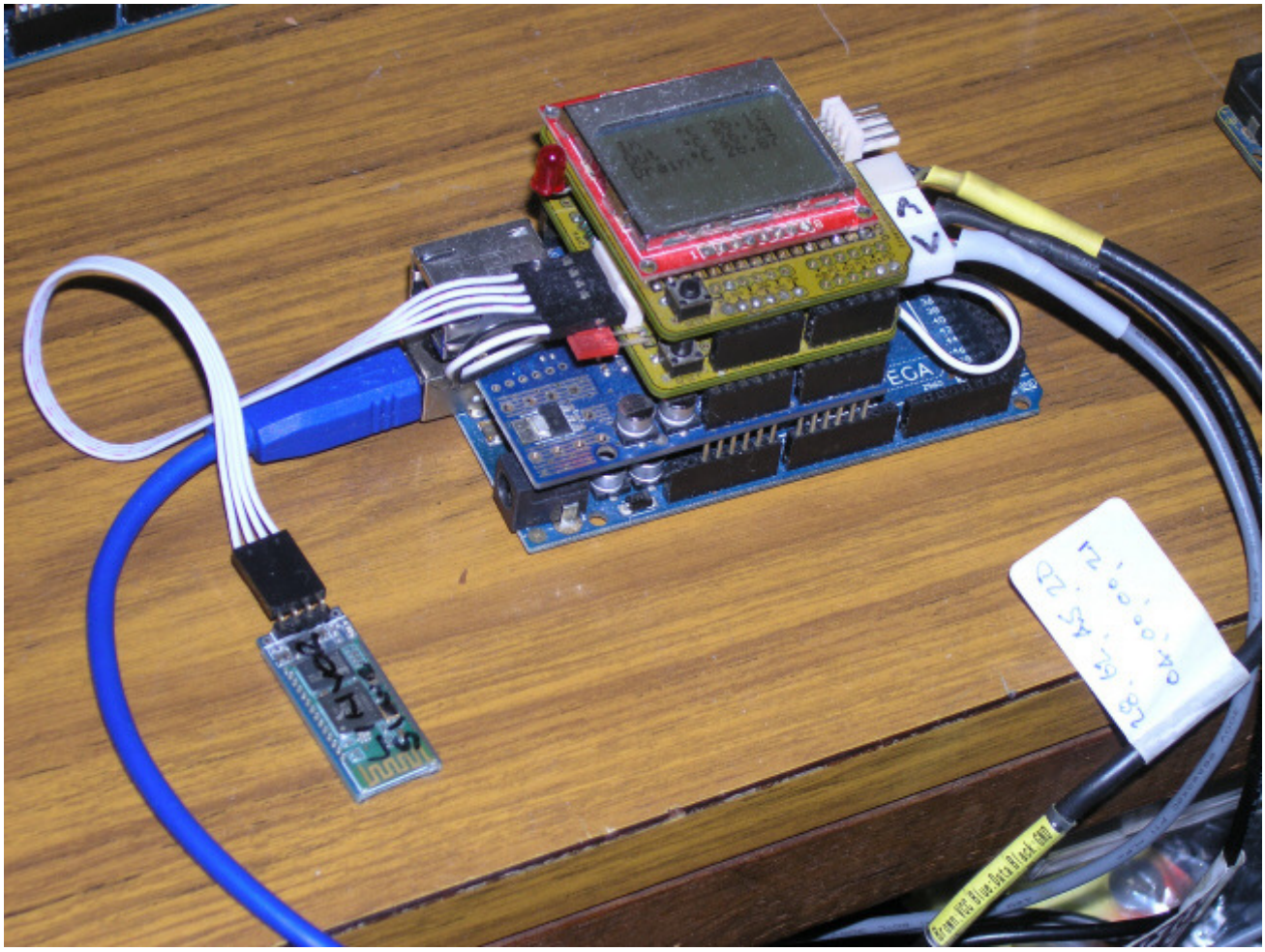
1.
Ensure Bluetooth is ON

2.
You may see a prompt about making Android visible to other devices. Say NO. This is because you are doing the looking, not Arduino.

3.
Select the "Search For Devices" bar.

4. After a short wait, Bluetooth should show up on Android.. HC-05s will identify as HC-05, or perhaps just a COM port number. HC-06 will always identify itself as "LINVOR". Select it to make contact. A password may be required – "0000" or "1234". Android should confirm the contact and the LED on Bluetooth should go **steady**. Note that this is the pairing process and it always take two or three seconds, which seems like forever.

Exiting this setup will mean a loss of connection, LED resumes flashing, but you now have Bluetooth in the list of Bluetooth contacts in the Android.




## 10. THE ANDROID APP

There is a swag of apps available, all free. BlueTerm seems to be the most popular. I prefer Bluetooth Terminal and Bluetooth Graphics, the latter is quite a gem and is just as useful in its text mode. Bluetooth Terminal can save data to a file. This defaults to SD card and is datestamped yyyymmdd.log. All allow the connection to be made from within the app. Some do not have a disconnect facility. This is not really a problem, there are other means. I don't know of any way that you can transfer a file via Bluetooth by using a file transfer app. The SD dump programme in the Arduino examples dumps the contents of the file to something like Bluetooth Terminal. This means the app offers two ways to the same end:

1. Live acquisition of data to a log file
2. Collect the same data later off an SD card.

Hence you can view data live in graph form with Bluetooth Graphics, which uses a string input from Arduino, and then pick up proper comma separated data off an SD card later..

## 11. BLUETOOTH OPERATION

**Now for the big bit.**

1.
Run your Bluetooth terminal programme.

2.
Go to the settings and press the connect button.

3.
The list of bluetooth connections should come up.

4.
Select Bluetooth and, see the confirmation, and the LED go continuous, now exit.

**Now for the moment of real triumph.**

5.
Press reset button on Arduino

6.
See the message on Android.

7.
Press a key, any key.

8.
Read the Android and feel the sweet sensation of success.

That is all you need to do. It really is quite easy and you only need to do it once to be aware of that. The above might sound a bit glib but it is true to the point that, if it fails to deliver first time, I'm afraid there is little option other than go through it again and check for some silly procedural error, an error that will only happen once. I remind you that some Bluetooth modules and shields are not supplied to run at the default speed 9600. Change Arduino code to match, not Bluetooth.

## 12. SUMMARY OF PROOFS

Looking at it more positively than "Troubleshooting":

1.
If you can communicate two way with the serial monitor, your code is OK for bluetooth. This is a good reason for using hardware serial D0,D1.

2.
If you can see any sort of LED on Bluetooth, the power is OK

3.
All fiddling is done at the Android end.

4.
Most, and probably all, Android apps give adequate on-screen confirmation of successful connection.

5.
A solid LED on Bluetooth indicates the connection is good. Note that it does not flash during data transfer, it stays solid until it is disconnected.

6.
All Android bluetooth apps work in much the same way, and the choice between them is just personal.

Now, after clearly understanding the above, about the only human error that is not made obvious by Arduino is the wiring of the communiucation lines therefrom to Bluetooth. So, about the last card in the pack is to ensure that Tx on Arduino is connected to Rx on Bluetooth and Rx on Arduino to Tx. Note that, if you got this wrong, you probably haven't done any damage.

## 13. KNOWN FAULTS

About the only problem that has arisen that is not user error has been a faulty board. There have been a couple of occasions where the Rx and Tx connections have been bridged by excess solder, leading to much consternation and seriously weird results. Once the problem has been identified it is easily fixed with a soldering iron.

## 14. NOW WHAT?

I have said this is just the basic stuff but, for many users, it may be more than you ever need to know. For straight datalogging, you only need one-way traffic anyway and the extra work required is not about bluetooth, it is about collecting and formatting the information that needs to be fed to bluetooth.

Be clear on this. Once you have proven your procedures are OK, you just send the data to serial just like you send it to serial monitor – nothing special, no libraries, just get the data together and send it.

Similarly, if you want to control some Arduino function by Android, it is just a matter of coding Arduino to do more than just say "I heard you say…", and that is not bluetooth related, it is all about what comes from Android, and what Arduino does with it. It shouldn't be too hard to get it together. There are several Android apps for controlling Arduino via bluetooth in the Google Store, and I imagine they include Arduino code to go with them.

## 15. CONCLUSION

1.
Read Section 5 Item 1. again.

2.
For cheap, simple, short-range communication, Arduino / Bluetooth / Android really is a ménage à trois made in heaven.

Thanks for comments, prodding, "mine doesn't work complaints", and feedback of silly events to

Wabbitguy
Retronet
Lar3y
Stoopkid
Dispplacer and several others

This is ongoing, and I would be glad of further comment on the Arduino forum

## 16. APPENDIX

### 1 WHO'S THE MASTER , WHO'S THE SLAVE, AND WHO NEEDS TO KNOW?

There has been a lot of confusion about this. The first thing to know is that the person who doesn't need to know is the person doing the exercise above.

Any Bluetooth connection has to be initialized by one of the parties, which is the master. That is where the pairing is done. In this exercise it is done by the Android, and Arduino is the slave. For the lack of a better term, pairing is the initial initialization – getting on the list. Once done, re-connection is a one-button affair and the slave simply needs to be present.

Arduino may be a master too, but in this exercise it doesn't act like one.. All the fiddling is done at the Android end – no exceptions.

The HC-05 is a slave by default but can be configured as a master. The HC-06 cannot, it is slave only. Both work fine in the above exercise because Arduino is the slave.

You may want to do other things in the future, like work with a device that is a slave e.g. an X-Box controller. In this case, Arduino must be a master, and programmed accordingly.

A master device is usuallydistinguishable because it enables you to see what you are trying to pair with and get confirmation that the pairing has succeeded e.g. the screens on phones and laptops. An Arduino as master need not have a screen and therefore cannot sniff about, but it is pre-programmed to pair with a device, no sniffing needed, and gives visual confirmation by flashing a LED.

A master can perform both ways without user intervention. This is common in cars. In our Getz, the radio pairs with the phone but the phone pairs with the handsfree in the Fairlane.

Note that "master" simply means that is where the pairing is done and has nothing to do with control. An Xbox gamepad is a slave device, no screen, but once the pairing is done, it can be used to control the Arduino.

### 2 ADDITIONAL SERIAL PORTS ON THE MEGA 2560

The Mega has extra ports RX/TX1, 2, and 3 on pins 19 to 14 in that order, all clearly marked. If you have the original exercise working, you can confidently relocate Bluetooth to another port if you need to. For instance, if you needed to run it on Serial2, pins 17 & 16, the only software change required in the above code is to rename every serial command to serial2 e.g.

```
Serial2.begin(9600);      etc.
```

Note that this deprives you of the comfort of checking your code with the PC's serial monitor, as Bluetooth is no longer sharing the same facility, but I guess you now have sufficient confidence not to be worried by that…

### 3 SOME MYSTERIOUS CONNECTION ISSUES

I have three JY-MCU devices, one HC-06 and two HC-05s. All were bought by mistake. The HC-06 should have been an HC-05, and the HC-05s were bought as bare modules and had to be soldered to JY-MCU boards. Once everything was sorted, all worked OK off a

Uno shield and connected to a Dell laptop. After that, all operations were with the HC-06, simply because I had not connected the LED on the HC-05s, and I had no immediate need for them..

The HC-06 has worked perfectly with laptop and the phone, hence the above notes.

While my cheapo tablet can usually see the HC-06, I have never gotten it to connect and work with it. The main reason why I got the phone was that I had despaired of ever getting the tablet to work.

I recently found that the tablet worked just fine with a bluetooth keyboard I had bought for the phone.

When experimenting with another Mega and an HC-05 on serial2, I found that the tablet connected fine and worked OK. I then swapped the HC-06 and HC-05, and found that the tablet worked with HC-05 on that Mega too. It still won't work with the HC-06.

I am still inclined to blame the tablet, but things are clearly not all they might be in the bluetooth world. No, I don't regret getting the Huawei.

## 4 SOMETHING OTHER THAN A JY-MCU?

The HC-0x family are available on proprietary shields and should perform thereon in exactly the same manner. If a shield is configurable, it is likely to work in the kosher hardware mode by default.

I would be glad of any feedback about these notes being useful with any other Bluetooth devices. And also about level shifting on foreign shields.

## 5. HC-05 OR HC-06?

When Android searches, HC-05s identify as HC-05 and HC-06s identify as LINVOR – unless you have changed the names. Both can come on JY-MCU boards. I believe all HC-05s thereon come with a 6-pin connector, and HC-06s with four. Blank JY-MCU boards all have six pins

## 6. USING WINDOWS INSTEAD

It really isn't very different. You are already aware that you cannot use the serial monitor for Bluetooth communication. There is a swag of free terminal programmes that you can use. I use RealTerm. The operation is largely intuitive. You use the COM port that is advised by the sniffer in the pairing process. This is often around COM40 so, if you see that, it is probably not a mistake.