A CHILD’S GUIDE to BASIC TWO WAY BLUETOOTH COMMUNICATION via HC-05 or HC-06 BETWEEN ARDUINO AND ANDROID.

version 52 for print on A4 October 2018 Extra links at the end.

EXECUTIVE OVERVIEW
Got an Arduino currently collecting data and sending a stream to serial monitor, and you would now like to send it to a device via bluetooth?

1. Ensure current serial.print in your code is the default 9600
2. Connect bluetooth as shown in diagram in section 4. LED flashes 2Hz.
3. Run programme
4. Power up device and get bluetooth to scan for Arduino’s bluetooth
5. Default password is 1234, or maybe 0000 to pair it. LED still flashes 2Hz
6. Run terminal programme in device and make Bluetooth connection from there
7. LED flash rate changes, may even be solid.
8. See data stream in terminal

The details of these steps will follow, but the above is an outline of all you need. There is no change to the code you already use, no messing about, no technobabble. The only thing specific to bluetooth you need to know is that, if you need to upload new code later, you need to have Bluetooth disconnected while doing so. That is all.

AND NOW FOR THE DETAILS
I put this up as a discussion on getting a simple TTL serial bluetooth connection, with HC-05 or HC-06 modules. I use it for simple datalogging purposes and accessing recorded data, using the plain vanilla SPP protocol. This also involves transfer of files containing the same data.

If you intend to stream or transfer audio/video files, none of this applies to you, so read no further.

These notes do not address using Bluetooth 4 (BLE) devices, they being unsuitable or unable to work as SPP data device and this thereby also excludes all IOS devices, and the Arduino Tian. So, again, read no further.

There are other “classic bluetooth” devices that are just fine with Arduino and Android, but I am still trying to keep it simple and cheap and don’t want to address them either. Similarly, I don’t address those versions of HC-0x on full size shields, for the same reasons, but one HC-0x is much the same as another, regardless of how it is presented.

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 instil some badly needed confidence. I’m afraid it has become a bit of a magnum opus as the information accumulates, also as the bluetooth scene changes.

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. I think I got a bit lucky with that and it doesn’t follow that a desktop with a bluetooth dongle is as easy a way to go. My current Windows 7 desktop is
1. THE OBJECTIVE

The objective is to be able to stand near the Arduino and casually acquire data; firstly to tap into a regular stream – one-way traffic – and secondly, to download stored data on command – two-way traffic. 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 can 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 - ARDUINO

A standard Arduino Uno or Mega, and powered by a standard 9v wall wart. If you want to use batteries, do it later, not now.

An “Arduino” I see as a dead loss is the ATtiny 85. I don’t know anything about it, but I understand it has no UART, hence no hardware serial, so, if you are using one, don’t bother to read any further. This problem is probably fixable, but not here. Also, any Arduino that has a Bluetooth-4 (BLE) incorporated, starting with the 101, is out of this realm.

The following is all I can say about exotic Arduinos, by which I mean Arduinos that do not take a standard Arduino shield in the manner of Uno and Mega, and may use different pins for serial connection, and maybe different code for serial commands. I’m sure any problem is fixable, as we are only talking about power and the most basic serial connection. Note particularly that, while I refer to exotic Arduinos as those that do not take standard Arduino shields, this is not a liability per se, as the only bluetooth devices I will address are not on standard shields anyway(!) Having said that, here is a quote pertinent to Leonardo, which does take a shield but is still exotic:

"-Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data using the ATmega32U4 hardware serial capability. Note that on the Leonardo, Serial refers only to USB communication; so, for TTL serial on pins 0 and 1, use Serial1."

I understand the same applies to the Micro and Micro Pro, which are a stripped down Leonards, and also the Zero and MKR1000. What this means is that, with these devices, wherever you see a command in a programme for Serial, you simply replace it with Serial1. This procedure may apply to the rest of the MKR family.

Nanos and 5v Pro Minis are stripped-down Unos and perform the same way. Same applies to the smaller form Megas.

And there is comment later on the 3.3v Arduinos in the matter of wiring connections, and pictures of small Arduinos in section 6. Most notable of the 3.3v Arduinos is the Teensy 3.2. This has three hardware serial ports, which should appeal to Bluetooth users.

3. EQUIPMENT USED – BLUETOOTH

The situation here is changing - constantly. I originally dealt specifically with an HC-05 or HC-06 bluetooth module on a standard JY-MCU backboard. This backboard has become a
generic term – to the point where some users think JY-MCU refers to the bluetooth device. The two types are only distinguishable by the number of pins. The four-pin HC-06 operates as a slave only but is quite suitable for this exercise. The six-pin 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. These devices are available for about $3 on eBay etc., and you need some pretty persuasive reasons to pay any more money than that for a BT-2 device. Note that there are other devices in the HC-0x family but these notes address the HC-05 and HC-06 only. This to the point that, in the unlikely event that you actually find any of the others, I submit you should avoid them like the plague. This particularly applies to the HC-09, which was apparently brought out as a drop-in replacement for the HC-06, but probably isn’t. Also note Section 9 at the end of the appendix – hardware to avoid!

I get the impression that, at least as far as the HC-05 is concerned, the JY-MCU backboards are being supplanted by others that are essentially similar but have a button for use when configuring the module. Known examples of this are the ZS-040 and FC-114. About the only clear difference for routine operations may be the way the LED works. The ZS-040 board has the LED giving double flashes at approximately two second intervals after a connection is made. This is in contrast to the steady on state used by the JY-MCU.

Another board is on eBay. It has pins pointing vertically down, thereby making it better for installing on a proto shield. It is dual voltage and has no button. The CZ-HC-05 board is probably just like the JY-MCU in that it does not have a button, but the pin arrangement is different. LED and STATE are the same thing. The Grove Bluetooth V3.0 is just an HC-05 with an eye-watering price tag.

The HC-06 has four pins and no button, irrespective of the board it is on.

As far as the HC-0x family is concerned, I understand that variations in firmware, backboard etc. really only become manifest when entering AT mode. You may never need to do this, and certainly don’t need to worry about it now..

So, while there are little nuances, and other than the blacklist, all these devices have essentially the same look and feel, and serve the same purpose. In the immortal words of some guru on the Arduino forum, bluetooth is just

**serial communication without wires**

and, as far as Arduino is concerned, the job it has to do is the same as serial **with wires**.
Also note that, in all cases, the device should work fine as it comes out of the box, 

no configuration required.

The only thing you need to know is the baud rate, which, amongst **HC-05**, has never been known to be anything other than 9600 baud. In the unlikely event that it is different, there is no need to change it, it is easier to alter the programme to match it. This will also mean that you will need to alter the baud rate of the serial monitor as well. This is done in the settings in the lower right hand corner. In the case of the **HC-06**, it now seems that there are versions on the market that are 38400 baud. Fortunately, it seems that these are easily identifiable as your phone will identify them as HC-06, while the standard signal is LINVOR. To check this, you just have Bluetooth connected to power and sniff it with the phone. No Arduino code is need for this.

The only operational variation I am aware of is the JY-MCU v1.06, which I understand has an extra diode included to prevent the module futilely attempting to power Arduino when it has power and Arduino doesn’t. This requires the line

pinMode(0, INPUT_PULLUP);  // just a precaution

which I imagine does no harm when used with older versions. This is likely to come to all these modules, and thus using this line as a standard procedure for all is probably a good idea. In short, it doesn’t matter if you use HC-05 or HC-06 for this, and it doesn’t matter which version, or which board it is on.

4. THE MEANS OF CONNECTION

A four-conductor ribbon cable, or a breadboard lashup should suffice, or female-male leads direct into the Arduino headers. You need to ensure that Bluetooth can be isolated while the code is uploaded. With a Mega, it makes sense to routinely use one of the other serial ports and change the code accordingly, i.e. change all Serial commands to Serial1, and connect to that. This takes advantage of the Mega having several ports, and you thus don’t have to share with the serial monitor, **BUT** only make this change after you have this little project running!

Note also that the HC-0x is a 3.3v device and, while the JY-MCU handles the 5v power supply,.

*SIGNALS ARE*

![Diagram of signals]

Some level shifting on Arduino’s **Tx** line is a wise precaution when using 5v Arduino by using a 1k and 2k between Tx and ground. The following picture shows an example of this
Plain vanilla HC-05 connected to Uno. Connection to Mega may be the same but it is better to use Serial1 etc, pins 18,19. HC-06 looks the same but usually has only four pins.

JY-MCU refers to the breakout board, not the bluetooth module.

Note particularly:

1. Never be tempted to run these devices off the 3.3v pin, not even on a Due. I understand that most 3.3v Arduinos have a 5v supply available.

2. The only exceptions to this that I know of are an HC-06 that is specifically made to run on 3.3v, and a dual voltage HC-05, in which case never run the former on anything other than a Due as all the other Arduinos have insufficient power available at 3.3v. Same goes for the dual voltage at 3.3v.
The picture shows an HC-05 connected to a sensor shield via a ribbon cable. This is all that is needed for communications. Yes, it still uses pins D0, D1 for serial.

Here is a later version - an HC-05 on JY-MCU plugged directly into a little clock shield on a Mega, and using pins 18, 19 for Serial1. Using Serial1 means it could be soldered in without complications, but I’m glad I didn’t. There is a voltage divider under the clock. While this fits neatly under the Ethernet shield, I’m not sure it works very well and I believe placing Bluetooth in an obscure position hinders the signal. This might not have been helped by the
foil insulation on the wall behind. I have never had a problem before, but I always had Bluetooth on flying leads and positioned in clear air.

The following shows the same gear, ready to go. Bluetooth has been relocated to the side of the box, using a ribbon cable, and is located with sticky Velcro tape.

Finally,

5. WHAT’S AT THE OTHER END?

Any Android device or any PC, Windows, Linux, or mac OS, is OK. I use a $50 Huawei phone. With BLE becoming commonplace, I guess it is now less appropriate to say that, if you use an iPhone, you are on the wrong tram, but you are if you are using HC-0x, so read no further.

Note particularly that these notes do not address Android programming. There is no need. There are several apps available that are fine for this exercise, and anything else is just an attempt to re-invent the wheel. Further notes on Android apps follow.

6. THE COMMUNICATION METHOD AND THE CONNECTIONS REQUIRED

This is all about using the standard hardware serial protocol, usually 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. Needless to say, this wiring has nothing to do with the wireless transmission, so the setup is Arduino transmits along a wire from its Tx connector, and Bluetooth receives the signal at its Rx connector – hence the arrows! Read what is written by the pins!
The pin configuration in the various exotic Arduinos can vary with different manufacturers, but I’m sure that the hardware serial pins are always clearly marked as to function, even if the arrows are missing.

These notes are 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, although you might like to read item 6 in the appendix below.

Note particularly that, while Bluetooth modules can be configured in two modes, master and slave, these characteristics are 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.

If you are using a wall wart and HC-0x in the normal manner, power consumption is not an issue. There is a note on power in the appendix.

7. NOTES ON WHAT THE BLUETOOTH MODULE IS ABOUT

The bluetooth is a separate device between Arduino and Android – a go-between, and neither Arduino nor Android know nor care what is on the other side. 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 or connection. It is just providing the power and, if there was another source of power, that could be used instead. No code required.

2. Consequently, a successful pairing or connection 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.

**BUT NOTE**

> if you got comms with the serial monitor, your only problem can be wiring between Arduino and Bluetooth, and it’s time to check the Rx/Tx wiring again.

> you could check comms are kosher by sending an AT command and getting an OK. This should not be necessary, and is really a last resort to test if a wire is broken, rather than that it is connected properly.

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

In summary, Arduino<>Android comms are established by two operations:

1. Arduino>Bluetooth. This is done by **programming**, which may be as simple as
   Serial.begin(9600);
   Serial.println("lahdedah");

2. Bluetooth>Android. This is done by **procedure**, all at the Android end – the pairing in the device settings, and the connection in the Android app.

**8. 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

   **no extra software requirements or special commands are required for bluetooth.**

This means **no libraries**. It is vital that you understand that it is just another serial device, no difference from the serial monitor, 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, no AT+ commands, but note that you do need to know the default baud rate of Bluetooth, and code Arduino accordingly. It is normally 9600, indeed I have never actually known it to be anything else with these HC-0x modules, but there is now suspicion of a supplier fiddling with the default and setting it to 38400.

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. Bluetooth is essentially two communication devices in one. It has a serial facility to connect with Arduino with wires, and a bluetooth facility to connect with the phone without wires. The serial connection settings are for just that – the serial connection between Arduino and Bluetooth. The connection between Android and Bluetooth is strictly by Bluetooth protocol that you cannot do anything about, and has its own speed completely independent of the serial speed, hence the serial setting is 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 a final boost to confidence, I submit that it is actually 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. More likely my cheapo tablet. There is further discussion in the appendix. Having routine operations with both the Android phone and XP laptop proves this project is essentially kosher.

9. THE LED

Assuming Bluetooth is powered by Arduino, the only relationship between the LED on Bluetooth and your Arduino is that, if the LED shows nothing, there is no power. Things are changing with LED actions as different versions are produced.

HC-05 and HC-06 both flash fast @ 2Hz when power is on and ready to connect irrespective of the board they are on.

Most have solid LED when connected. HC-05 on ZS-040 board gives a pair of brief flashes once every two seconds or so when connected.

I believe other HC-05s go steady off when connected, and then give a brief flash on receipt of signal. The thing to look for is a change from the regular 2Hz flash signifying power on. Whatever it is, it will be clear enough.

I believe the LED on the HC-05 is configurable, which could be useful.
10. THE BASIC TEST CODE

The code can be downloaded from


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.

So here it comes

String readString;
char c;

void setup()
{
   pinMode(0, INPUT_PULLUP);// only needed for JY-MCU v1.06?
   Serial.begin(9600);
   Serial.println("OK then, you first, say something.....");
   Serial.println("Go on, type something and hit Send,");
   Serial.println("or just hit the Enter key,");
   Serial.println("then I will repeat it!");
   Serial.println("\n");
}

void loop() {
   while (Serial.available())
   {
      delay(3);
      c = Serial.read();
      readString += c;
   } // end while
   if (readString.length() >0)
   {
      Serial.write(c);
      readString="";
   } // end if
}

Yes, folks, that is all there is to it, and all you need. Actually, you can get by with even simpler programmes than this but the few extra lines gives a programme that can be made better use of in future projects. This is what I use for downloading off SD card, as shown in part 16.

The setup calls the serial facility and prints a message. 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 repeats the information back.

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

### 11. 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.

### 12. ESTABLISHING BLUETOOTH CONNECTION

Here is the moment of truth. **Connect Bluetooth to Arduino.** Note that you don’t actually have to have the programme loaded or running at this stage, this is just for the pairing and all Arduino does is provide power to Bluetooth.

You should see the LED **flashing 2Hz**. See above

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

1. Ensure Bluetooth is ON – LED flashes 2Hz or faster

2. In Android settings, 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” or “Scan” bar. Some phones scan as soon as you open bluetooth settings.

4. After a short wait, Bluetooth should show up on Android. HC-05s will identify as HC-05, HC-06 will usually identify itself as “LINVOR”, or possibly HC-06. Select it to make contact. A password may be required – “0000” or “1234”. Android should confirm the contact. The LED on Bluetooth should change. Main confirmation is on phone. Note that this is the pairing process and it always takes several seconds, which seems like forever. Also note particularly that may pair automatically without you realising it, no password required, so check the paired devices list if there is no scan activity.

Exiting this setup will mean a loss of connection, HC-05 LED resumes flashing, but you now have your Bluetooth in the list of Bluetooth contacts in the Android. LED must resume 2Hz flash. If it won’t, reset or disconnect power at Arduino end. Note that, once paired, Bluetooth and Android stay paired until you specifically “unpair” them.

The **pairing** is done in Android’s setup but the actual **connection** is done in the app you open and only applies to that app. This means that, if you want to move to a different app, you need to disconnect in the first app before you do. **Pair once - connect many.**
Note particularly that, if your HC-06 identifies as HC-06, not LINVOR, it may mean that Bluetooth runs at 38400 baud, not the default 9600. This means you should change the Serial.begin command in your programme to 38400 and also change the serial monitor. This baud rate has no bearing on the pairing, the pairing process is just an easy way of finding this out before you get any grief.

13. THE ANDROID APP

There is a swag of apps available, all free. BlueTerm seems to be the most popular. I prefer Bluetooth Terminal by Qwerty Tools and Bluetooth Graphics Terminal. I use both rather than try to tweak the latter to serve as the former. Blue Graph is another with great promise but I have not used it yet. As an Arduino user, these graphic apps are just about the best reason you will ever have for getting an Android tablet or phone but Bluetooth Terminal might be better for these basic exercises as it seems a bit easier to use for saving downloaded data to a file. Tick the “Append newline(\r\n)” box in the setup. This defaults to device and is datestamped yyyymmdd.log. This file is transferred to PC and opened with Notepad or Excel. Be aware that there are several apps called “Bluetooth Terminal” and you need to ensure you have one that can log data. I downloaded one that doesn’t, which makes it pretty useless.

All terminals have 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, i.e. they all just read the contents of the file, and one Android log file may contain the contents of several Arduino files.

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.

14. BLUETOOTH OPERATION

Now for the big bit.

1. Run your Android Bluetooth terminal programme.

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

3. The list of paired bluetooth connections should come up.

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

Now for the moment of real triumph.

5. Press reset button on Arduino. This is just to give you the start message on Android.
6. See the message on Android. (That’s right, Arduino’s serial monitor is not a player, but it may be a spectator)

7. Press a key in your Android terminal app, 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 believe all HC-0x modules are 9600 by default, but I remind you that some other Bluetooth modules and shields use other speeds. Change Arduino code to match, i.e. Serial.begin(9600);, not Bluetooth, and change the speed of the monitor to match.

15. 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. Any change from the initial 2Hz flashing of the LED indicates the connection is good.

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 errors that are not made obvious by Arduino are

1. The wiring of the communication 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.

2. The baud rate. Most are 9600 by default. If yours is different, change the Serial.begin(9600); command to match, and don’t forget to change the serial monitor as well.
I guess THE last card in the pack is the power supply. The Bluetooth modules are not famous for the low power consumption. Testing with just Bluetooth and using the USB for power is surely OK. If you are using any battery power, it would be better to use a proper wall wart until you have proven the system. If you are using a 9v battery, get rid of it immediately. They are useless for this purpose and you will probably regret it if you don’t.

16. 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 echo what is sent, 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.

17. PRACTICAL APPLICATION

This is a stripped-down version of the main programme I am currently using. Essentially, the LCD and Internet sections have been removed.

Temperatures from three DS18B20 sensors are read at one second intervals for transmission over bluetooth - no timestamping. It is only for real time reading anyway.

Every tenth reading is recorded on SD card in real numbers to two decimal places. Each line is timestamped.

A new file is created at midnight using the date as the filename and they can be read from SD via bluetooth by sending MMDD from the Android terminal.

Note that baud rate is **115200**. This is only needed to lessen the time for dumping the daily files, and it is OK to use 9600 instead.

No great claims are made for the elegance of this programme.

```cpp
#include <OneWire.h>
#include <DallasTemperature.h>
#include "Wire.h"
#include <SD.h>
#include <SPI.h>

#define DS1307_ADDRESS 0x68

char filename[] = "00000000.CSV";
File myFile;
```
char dumpName[] = "00000000.CSV";
File dumpFile;

// Green group Dee Why
byte InThermo[8] = {0x28, 0x69, 0xC2, 0xB0, 0x03, 0x00, 0x00, 0x9F};
byte OutThermo[8] = {0x28, 0x7A, 0x8B, 0xC0, 0x03, 0x00, 0x00, 0x2F};
byte DrainThermo[8] = {0x28, 0x54, 0xF7, 0x2D, 0x04, 0x00, 0x00, 0x68};
#define ONE_WIRE_BUS 3
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

byte second, minute, hour, weekDay, day, month, year;
int k=0;
const int chipSelect = 4;

float tempC, InTemp, OutTemp, DrainTemp;

// Define the strings for our datastream IDs
char sensorId0[] = "InThermo";
char sensorId1[] = "OutThermo";
char sensorId2[] = "DrainThermo";
char charBuf [15];

String readString;
String StringOne;

void setup() {
  Serial.begin(115200);

delay(300);//Wait for newly restarted system to stabilize
pinMode(0, INPUT_PULLUP);// only needed for JY-MCUY v1.06
pinMode(53, OUTPUT);//MEGA use pin 10 if Uno

// See if the card is present and can be initialized:
if (!SD.begin(chipSelect))
{
  Serial.println("Card failed");
  // don't do anything more:
  return;
}
Serial.println("CARD OK");
GetClock();
getFileName();
}

void loop() {
  if (today != day)
  {
    today = day;
    getFileName();
  }
  while (Serial.available())
{  delay(3);  char c = Serial.read();  readString += c;}// end while  if (readString.length() >0)  {  getDump();  readString="";  } // end if

// get the values from the DS8B20's
sensors.requestTemperatures();

InTemp = sensorValue(InThermo);
OutTemp = sensorValue(OutThermo);
DrainTemp = sensorValue(DrainThermo);

k=k+1;
if (k>9 )  {
    myFile = SD.open(filename, FILE_WRITE);//<<<<<<<<<<< OPEN
    myFile.print(hour);
    myFile.print(":");
    myFile.print(minute);
    myFile.print(":");
    myFile.print(second);
    myFile.print(".");
    myFile.print(InTemp);
    myFile.print("");
    myFile.print(OutTemp);
    myFile.print("");
    myFile.println(DrainTemp);
    myFile.close();//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>CLOSE
    k=0;
}  delay(1000);
} // loop ends here

//sensorValue function
float sensorValue (byte deviceAddress[])
{
    tempC = sensors.getTempC (deviceAddress);
    return tempC;
}

void getFileName()
{
    sprintf(filename, "%02d%02d%02d.csv", year, month, day);
}

void GetClock() {
    // Reset the register pointer
Wire.beginTransmission(DS1307_ADDRESS);
byte zero = 0x00;
Wire.write(zero);
Wire.endTransmission();
Wire.requestFrom(DS1307_ADDRESS, 7);

second = bcdToDec(Wire.read());
minute = bcdToDec(Wire.read());
hour = bcdToDec(Wire.read() & 0b111111); //24 hour time
weekDay = bcdToDec(Wire.read()); //0-6 -> Sunday - Saturday
day = bcdToDec(Wire.read());
month = bcdToDec(Wire.read());
year = bcdToDec(Wire.read());
}

byte bcdToDec(byte val)  {
    // Convert binary coded decimal to normal decimal
    return ( (val/16*10) + (val%16) );
}

void getDump() { 
    StringOne = "2015" + readString + ".csv";
    StringOne.toCharArray(charBuf, 15);
    Serial.println("");
    Serial.println(charBuf); // identify filename on phone
    File dumpFile = SD.open(charBuf);
    if (dumpFile)
    {
        while (dumpFile.available())
        {
            Serial.write(dumpFile.read());
        }
        dumpFile.close();
    }
    else {
        Serial.println("error opening file");
    }

18. KNOWN FAULTS

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.

There has been a pretty clear instance where HC-05s have been sold as HC-06s as a result of some idiotic manufacturing stuff-up. The GW-040 breakout boards had four pins, thereby identifying the module as HC-06, but they could not be configured as HC-06. They identified as HC-06, which immediately makes them suss, as kosher HC-06s identify as LINVOR.. Further, there was voltage on a pin that is not connected on a genuine HC-06, which rather proves they were not. I imagine these devices would be fine if you did not need to configure them, which was not the case for the unfortunate buyer.


18
19. Conclusion

Read Section 8 Item 1. - again.

For cheap, simple, short-range communication, Arduino / Bluetooth / Android really is a ménage à trois made in heaven. For me, this is largely because of Bluetooth Graphics Terminal, which I found to be a game changer.

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

Wabbitguy - on BLE and IOS etc.
Retronet
Lar3y - on input from Android
Stoopkid- on PCB problems
Displacer – on silly mistakes
NJM on matters of mystery re HC-06
EGB – Arduino<>Arduino communications
manatom and several others
Note again, as a final confidence booster, that you can watch proceedings on the PC serial monitor while you do the deeds on your Android device, but only watch. You cannot send from the serial monitor.

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

Appendix follows, more anon.
20. 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 and connecting 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. perhaps, an X-Box controller. In this case, the Bluetooth module connected to Arduino must be a master, and configured accordingly.

A master device is usually distinguishable 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. A master bluetooth module on an Arduino need not have a screen and therefore cannot sniff about, but it is instead pre-programmed to pair with another 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 where the radio sniffs and connects with the phone automatically.

Note that “master” simply means that is where the pairing or connection is done and has nothing to do with control. Once the connection is made, it doesn’t matter who is who.

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 changes required in the above code are to rename every serial command to serial2 e.g.

Serial2.begin(9600);   etc.

And change the pullup

pinMode(17, INPUT_PULLUP);

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 first cheapo tablet could see the HC-06, I never got 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. The tablet finally died and I now have a new one. It connects OK with HC-06 but it must have taken ten seconds to pair and I nearly gave up.

It turned out that the original 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 wouldn’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.

It now seems that I am not alone in encountering connection problems with the HC-06, but there is still no explanation, and the problems seem as random as ever.

4     SOMETHING OTHER THAN A JY-MCU OR THE LIKE?

All the little breakout boards for HC-0x are essentially the same. The modules are available on proprietary shields and should perform thereon in exactly the same manner as the common modules. 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.

5     SOFTWARE SERIAL

I am trying not to be too bombastic about the errors and futility of using Software Serial. The following is by “Pylon”, a notable contributor to the Arduino Forum. I have based this project on the standard hardware serial being an advantage for beginners. This explains the disadvantage of software serial for everybody.

The only good enough reason (for using it) is that you have absolutely no other option. SoftwareSerial blocks the Arduino for long periods with interrupts disabled which poses lots of problems if you have to deal with hardware that needs a reaction within some time. And it works reliably only up to about 9600 baud, above that you need a very tolerant
communication partner because the timing gets quite lousy and I never reached anything above 38400 baud, not even with adapting hardware.

The “absolutely no other option” comes about where you need more than one serial device and have bought the wrong Arduino, namely a Uno, and you should have bought a Mega, which has four serial ports. Or you have more than four serial devices, in which case maybe you should not be using Arduinos at all.

Now the two common cases of “absolutely no other option” are:

1. where you are not merely communicating via bluetooth between Arduino and another device, but are actually communicating between two other devices, and Arduino is just an interface. An example is where you are using a PC to send data to a phone, i.e. use the keyboard, and the only reason you are doing that is because the PC has a keyboard and Arduino doesn’t. This only happens when you are playing around. If you are serious about communicating between PC and your phone, there are better ways of doing that than using an Arduino.

2. where you need to configure a bluetooth using an Arduino with only one hardware serial port, and you need that for the serial monitor in order to send the commands, e.g. a Uno. I’m not sure it is entirely necessary to use software serial even for that, but it makes sense to do so, and it is just a temporary setup anyway.

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. It does a fine job and 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 something like that, it is probably not the mistake it might at first appear to be.

All the above applies to a Windows laptop with bluetooth Installed and running. In my case they are Dells with the Toshiba bluetooth stack. I have never succeeded in communicating via a bluetooth dongle in my desktop, and there is very little motivation anyway.

There is no evidence that suggests that a windows phone has anything to offer.

7 PROGRAMMING BLUETOOTH

I have studiously resisted making any changes to the defaults but I have been faced with downloading daily 250k files. This is pretty tedious at 9600 baud and I am having difficulty making the files smaller, so I am faced with boosting the speed.

Note that I have heard rumours that slower 3.3v Arduinos are not good for running at 115200.

Other than that, Arduinos can run bluetooth modules just fine at 115200 with hardware serial, so, if you need to change the speed at all, I fail to see why it should be to anything less than 115200. Note that 115200 seems to be a maximum only by convention, which is inherited from Windows, and I believe really is a maximum for HC-06. The HC-05 can be
set to 1382400, twelve times as fast and, while I have never heard of it being used, should move a serious amount of data in a hurry.

The first programmes are for **HC-06** only. The HC-06 is in AT mode 9600 baud, ready for configuration as soon as power is applied, and remains so until a connection is made.

The HC-06 is quite straightforward. If it has four pins, it is an HC-06 and there is only one way to do configure it. The HC-05 is not so obvious, and there are even some myths and confusion. The guru who has it together with the HC-05, including its backboard variations, is Martyn Currey. He demonstrates using oddball Arduinos too. I don’t think he covers the Mega per se, but Uno code and procedure will work on a Mega anyway. Check his website listed below. He does HC-06 as well!

The first HC-06 programme enables you to change the settings with a **Mega**, and using Tx1, Rx1 for Bluetooth, thereby using the serial monitor at the same time.

It is a one-shot programme, all the work is done in the setup, and you write the changes you want therein. I fail to see the need for anything more sophisticated. It’s not the sort of thing one needs to do often. Changing the speed can’t be a bad idea but it took me a couple of years to feel the need justified the effort, and about the only need to change the name is to distinguish one module from another when you have several.

/*
This is for MEGA

HC-06 only responds to 9600 internally. Even if you set the baud rate to 115200, you don’t need to know that, and you can reprogramme the module again with 9600.

JY-MCU board pins
RX    - 18   Tx1     orange
TX    - 19   Rx1     white
GND   - GND        black
VCC   - 5v             red

Kudos to marguskohv - he sowed the seed....
Serial monitor is just aide memoire
*/

**String command = ""; // Stores response from HC-06**

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);       //monitor
  Serial1.begin(9600);      //bluetooth

  Serial.print("AT\r\n");
  Serial1.print("AT");      //PING confirmation of kosher connection!!
  if (Serial1.available()) {
    while(Serial1.available()) { // While there is more to be read, keep reading.
      delay(3);
      char c = Serial1.read();
      command += c;
    }
  }
}
```
delay(2000);
Serial.println(command);
command = ""; // No repeats

Serial.print("AT+NAMEFosters ");
Serial1.print("AT+NAMEFosters"); //CHANGE NAME
if (Serial1.available()) {
    while(Serial1.available()) { // While there is more to be read, keep reading.
        delay(3);
        command += (char)Serial1.read();
    }
} else {
    delay(2000);
}
Serial.println(command);
command = ""; // No repeats

Serial.println("AT+PIN1234");
Serial1.print("AT+PIN1234"); //CHANGE PASSWORD
if (Serial1.available()) {
    while(Serial1.available()) { // While there is more to be read, keep reading.
        delay(3);
        command += (char)Serial1.read();
    }
} else {
    delay(2000);
}
Serial.println(command);
command = ""; // No repeats

Serial.print("AT+BAUD8 ");
Serial1.print("AT+BAUD8"); //CHANGE SPEED TO 115K
if (Serial1.available()) {
    while(Serial1.available()) { // While there is more to be read, keep reading.
        command += (char)Serial1.read();
    }
} else {
    delay(2000);
}
Serial.println(command);
}

void loop()
{
} //one-shot - nothing here

The following is for **Arduino with a single serial port** i.e. Unos and others of that ilk. This uses software serial. Despite my contemptuous dismissal of software serial, there is a place for it, and a temporary hookup for configuration is the best example. It is indeed almost exactly the same as the previous, but simply uses software serial for Serial1 and different pins. Never use software serial on hardware serial pins. You will be amazed how many do it, and they deserve all the grief they get.
This is for any Arduino

HC-06 only responds to 9600 internally. Even if you set the baud rate to 115200, you don’t need to know that, and you can reprogramme the module again with 9600.

One Shot
Kudos to marguskohv - he sowed the seed....
Serial monitor is just aide memoire

#include <SoftwareSerial.h>
SoftwareSerial Serial1(2, 4); // RX | TX
String command = ""; // Stores response from HC-06

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);       //monitor
    Serial1.begin(9600);      //bluetooth
    Serial.print("AT      ");
    Serial1.print("AT");      //PING
    if (Serial1.available()) {
        while(Serial1.available()) { // While there is more to be read, keep reading.
            delay(3);
            char c = Serial1.read();
            command += c;
        }
    }
    delay(2000);
    Serial.println(command);
    command = ""; // No repeats
    Serial.print("AT+NAMEFosters      ");
    Serial1.print("AT+NAMEFosters");        //CHANGE NAME
    if (Serial1.available()) {
        while(Serial1.available()) { // While there is more to be read, keep reading.
            delay(3);
            command += (char)Serial1.read();
        }
    }
    delay(2000);
    Serial.println(command);
    command = ""; // No repeats
    Serial.println("AT+PIN1234");
    Serial1.print("AT+PIN1234");        //CHANGE PASSWORD
    if (Serial1.available()) {
        while(Serial1.available()) { // While there is more to be read, keep reading.
            delay(3);
            command += (char)Serial1.read();
        }
    }
    delay(2000);
    Serial.println(command);
}
command = ""; // No repeats

Serial.print("AT+BAUD8 ");
Serial1.print("AT+BAUD8"); //CHANGE SPEED TO 115K
if (Serial1.available()) {
    while(Serial1.available()) { // While there is more to be read, keep reading.
        command += (char)Serial1.read();
    }
} delay(2000);
Serial.println(command);
}

void loop(){
} //one-shot - nothing here

The following is for **HC-05** with **button switch** e.g. ZS-040 on **Mega**, using **Serial1**. This approach uses keyboard entry, rather than the simple one-shot above. Hold button down as you connect the power - easy. The LED flashes much more slowly, about two seconds on and two seconds off.

    /*HC-05 ex Currey
Serial monitor 9600 Both NL & CR
The "->" character indicates the user entered text.
AT+NAME=name AT+UART=115200,0,0 */

char c=' '; boolean NL = true;

void setup() {
    Serial.begin(9600);
    Serial1.begin(38400); // this is for Bluetooth in AT mode, irrespective of comms speed.
    Serial.println("Serial active...enter AT command");
}

void loop() {
    if (Serial1.available()) \Bluetooth in
    {
        c = Serial1.read();
        Serial.write(c);
    }

    if (Serial.available()) \keyboard
    {
        c = Serial.read();
        Serial1.write(c);
        if (NL) { Serial.print("->"); NL = false; } // Echo
        Serial.write(c);
        if (c==10) { NL = true; }
    }
}
8  POWER CONSUMPTION

During the pairing, the current is in the range of 30-40mA. The mean current is about 25mA. After pairing, no matter if it is communicating or not, the current is 8mA. There is no sleep mode. This parameter is same for all HC-0x. Note that the above only applies to the bare module, LED and regulators are extra, and HC-0x on its breakout board is not famous for low power consumption.

I understand the state LED on HC-05 can be programmed. I don’t know how this is done, but this may be useful for the battery-powered brigade. You may even be able to programme it to stay off. Another option is to solder the bare module onto a blank JY-MCU board, and leave LED disconnected.

9  HARDWARE TO AVOID

1. This is not a bluetooth module, it is just a bare backboard. I believe the most obvious reason why they are on sale is that some other sucker might buy one.

Note that you can get a bare HC-05, like that shown below, to go on this. It is not such a good idea, but see the above note on power.

2. Further to item 18 on p18, There may well be nothing wrong with GW-040 breakout boards per se, but these are the ones that have had suss HC-06s on them. There is no good reason to buy an HC-06 on a GW-040 when there are plenty of others around on a JY-MCU. If you are already stuck with one of these and need to configure it, you might be able to add a wire, or apply a 5v signal to the vacant EN pin, or a 3.3v probe direct to pin 34 on the module, and the programme it as one would an HC-05.

3. Here is another suss HC-06. Guilty until proven innocent. I know of three instances where people have had problems with HC-06 that I believe can only be explained by an HC-05 masquerading as an HC-06.

4. If you can work out what this next one is all about, you are a better man than I. So, until proven otherwise, I believe this module has been put out by some idiot who has no idea of what he is doing. Don’t buy it unless you do know what you are doing – which is unlikely. I don’t know where it comes from, it turned up on the Arduino Forum.
10 OLD or NEW BREAKOUT BOARDS?

I have previously thought that there may be an advantage with the older JY-MCU board in that the absence of a button means that it would be configurable on the fly. I’m pretty sure this isn’t right. The new boards, like ZS-040 etc., still have the EN pin and thus can be programmed in the old fashioned way. There is no advantage in the JY-MCU board.

11 USEFUL LINKS

1 ARDUINO <> ARDUINO COMMUNICATION

Martyn Currey and Phillipe Cantin are the gurus who have comprehensively documented this. Note that you don’t actually need two HC-05s. An HC-06 will do for a slave.


http://phillipecantin.blogspot.ca/2014/01/hc-05-bluetooth-link-of-2-arduino.html

2 ANDROID BLUETOOTH JOYSTICK

This is one of the biggest threads on the Arduino Forum. I have never read it but it must have everything you need on this subject by now.


3 BASIC SERIAL COMMS

With plain-vanilla datalogging, you just print the variables to serial, SD card, and display in much the same manner. For more fancy stuff, you need to know more about Serial.print, Serial.write, and the nature of variables.


4 CONFIGURE HC-05 ON ZS-040 BOARD

This covers the button switch for getting into AT mode. It probably covers the FC-114 as well.

http://www.martyncurrey.com/?p=1348

5 USING THE LATER VERSIONS OF THE JY-MCU BOARD (V1.06)

11 NONE OF THIS APPLIES TO ME, AND I’M BUSTING A GUT TO GET INTO BLUETOOTH 4

http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/

Note that, while this link is the best I know of, the advice therein is that

1. Bluetooth 4 is not an upgrade for “classic” Bluetooth like HC-0x, it is a different system for different purposes.
2. You can use BT-4 devices like HM-10 for Arduino<>Arduino connections but they were not designed for this and HC-0x is a better option.
3. BT-4 achieves its low energy use by using infrequent small packets of data, and is not for continuous streams with large amounts of data.

Data logging to the outside world of Android and PC. Is not covered.