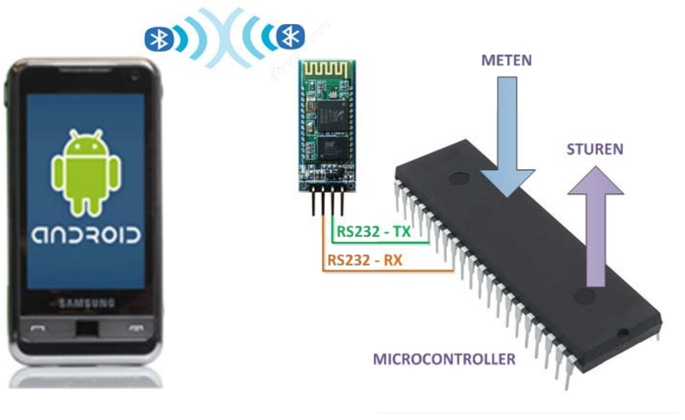
RS232 HC06 BLUETOOTH MODULE ZONDER INTERRUPT

Required knowledge RS232 comms

JUST Google™ It!	**RS232 protocol – HC06 Bluetooth module**

Components:

HC06 Bluetooth module	Aliexpress, DX.COM, … (+/-3€)



Smartdevice program:	Microcontroller program:
App – developed in APPINVENTOR - Source code available via www.E2CRE8.be - Installer available via www.E2CRE8.be	Source code in various programming languages available via www.E2CRE8.be

Most SmartDevices (Tablets, Smartphones) have a built-in Bluetooth module. APPINVENTOR (and other app development software) make it possible to send and receive data via Bluetooth in a relatively easy way.

Microcontrollers do not have a built in Bluetooth module. For this we need to connect the uC to a external BT device. We use the cheap HC06 module.

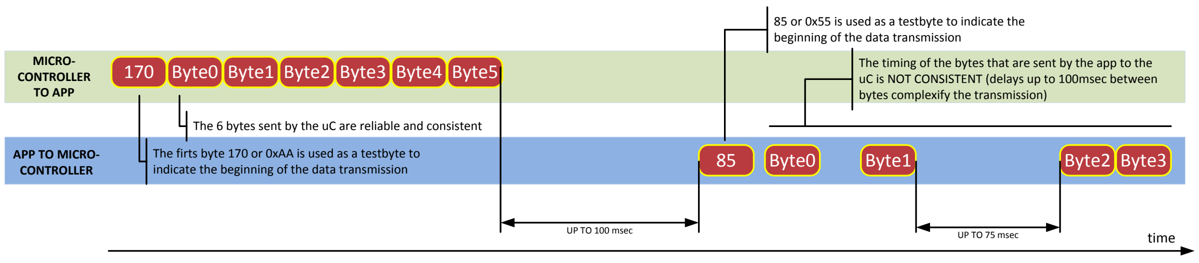
The blue motherboard of the HC06 converts the voltages of the green PCB from 3.3 to 5V and makes it compatible with our uC.

The HC06 can communicate Full Duplex (in two directions at the same time) via RS232 signals. All it does is translate RS232 signals into wireless Bluetooth signals. Our uC has a RS232 module that makes this RS232 communication fairly easy.

There are two ways to communicate over RS232:

Interrupt based	AT any incoming byte of data the program is automatically redirected to handle this data before returning back to the program. No incoming data is ever lost this way, but interrupts are not always supported in high level programming languages like Flowcode and Arduino IDE.
Not Interrupt based	To make sure that the uC reacts fast to new incoming data we need to take some precautions: • One device should always be the master to avoid data being transmitted while the other device is not ready to receive data. (The uC is the master – the app is the slave) • We wait long enough for new data to arrive • We use a flow-control byte to indicate the start of the transmission • We try to use as little bytes as possible. The more bytes – the more chance that something goes wrong.

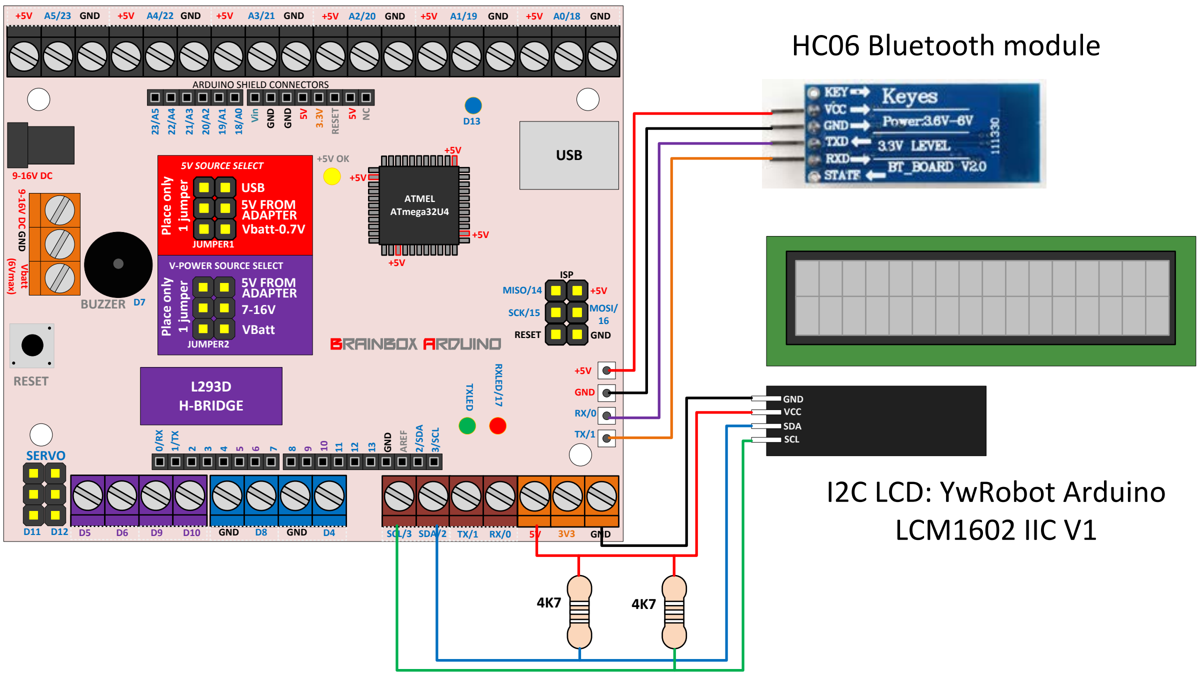
This is a diagram that illustrates how data is transmitted between the uC and the APP. We have measured this data with a logic analyser at the TXD and RXD pins of our HC-06 module.


85 or 0x55 is used as a testbyte to indicate the beginning of the data transmission
MICRO-CONTROLLER TO APP
170
Byte0
Byte1
Byte2
Byte3
Byte4
Byte5
The timing of the bytes that are sent by the app to the uC is NOT CONSISTENT (delays up to 100msec between bytes complexify the transmission)
The 6 bytes sent by the uC are reliable and consistent
APP TO MICRO-CONTROLLER
The firts byte 170 or 0xAA is used as a testbyte to indicate the beginning of the data transmission
85
Byte0
Byte1
Byte2
Byte3
UP TO 100 msec
UP TO 75 msec
time

1. The uC is Master and will start the communication by sending '170' (0b10101010) to the APP. The app uses this '170' to test if the first byte received is really the first byte.
2. The App is the slave and will only start sending data to the uC after all 7 bytes from the uC are received.
   a. Our measurements clearly show that that the sending of data by our Smartdevice isn't so real-time as we would like it to be.
   b. It may take up to 100msec before the app sends its first byte back to the uC
   c. The delays between the next 4 bytes may rise up to 75msec.
3. The first byte sent by the app is always '85' (0b01010101). This flow control byte is used by the uC to check if the first received byte is indeed the first byte.

Using interrupts will be a better – more professional way to handle this but our demo program are good enough to handle some bidirectional data. However, keep in mind that it are only demo programs and that these programs have no guarantee of working over long periods.

Aansluiting HC06 en I2C LCD



CODE EXAMPLE: **‘RS232- HC06- NOINT’**