RS232 HC06 BLUETOOTH MODULE ZONDER INTERRUPT

Vereiste voorkennis RS232 communicatie

Videolessen Deel 5 – Sturen en meten met APPS

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

Componenten:

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

Programma langs de kant van het Smartdevice:	Programma langs kant van microcontroller
App – geschreven in APPINVENTOR - Broncode beschikbaar via www.E2CRE8.be - Installer beschikbaar via www.E2CRE8.be	Broncode beschikbaar in verschillende programmeertalen via www.E2CRE8.be

De meeste SMARTDEVICES hebben een ingebouwde BLUETOOTH module. APPINVENTOR (en andere APP development software) geeft ons de kans om op een eenvoudige manier data te verzenden en te ontvangen via die ingebouwde BLUETOOTH module van onze SMARTDEVICES.

Microcontrollers hebben geen ingebouwde BLUETOOTH module aan boord. We moeten hiervoor een connectie maken met een externe BLUETOOTH module. Hiervoor hebben wij de keuze gemaakt voor de goedkope HC-06 module.

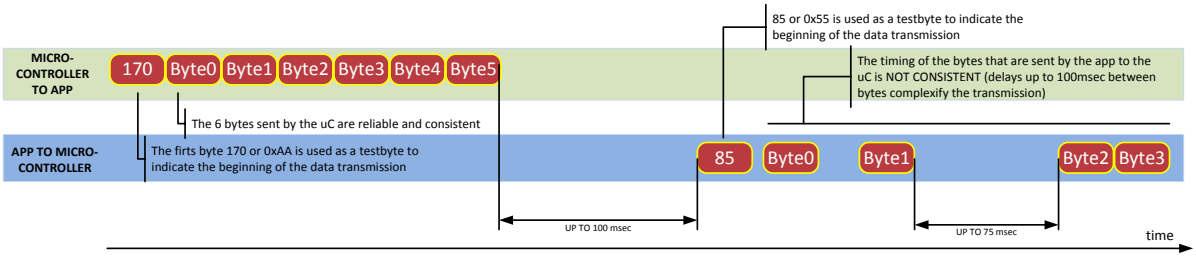
Het blauwe moederbordje past de spanningsniveaus van het groene printje aan van 3.3V naar 5.0V en maakt de module hierdoor bruikbaar voor onze microcontroller.

Deze HC06 module communiceert met de microcontroller in 2 richtingen via het populaire RS232 protocol. Onze microcontroller heeft een RS232 module aan boord die RS232 signalen perfect kan verzenden en ontvangen.

Er zijn twee manieren om met RS232 signalen te communiceren:

Interrupt gebaseerd	Bij elke inkomende byte data wordt het programma van de ontvanger automatisch onderbroken om de byte binnen te halen.
Niet interrupt gebaseerd	Om er nu zeker van te zijn dat inkomende data tijdig uit de ontvangst-buffer wordt gehaald voordat de volgende data binnenkomt moeten we een aantal maatregelen nemen: • Één van beide devices moet master zijn – de andere slave, op deze manier weet de master ongeveer wanneer de slave data zal terugsturen en moet het device niet onnodig lang wachten op data. (wij maken de uC master) • Door wachttijden in te bouwen kunnen we met grotere zekerheid garanderen dat de inkomende data correct binnenkomt. • D.m.v. controlebytes kan je vermijden dat een gemiste byte de ontvangst door mekaar haalt. • Hoe meer databytes ontvangen worden, hoe groter de kans dat er iets misloopt

Hieronder ziet u een mooie voorstelling van de signalen die we hebben opgemeten met een logic analyzer op onze programma-code.

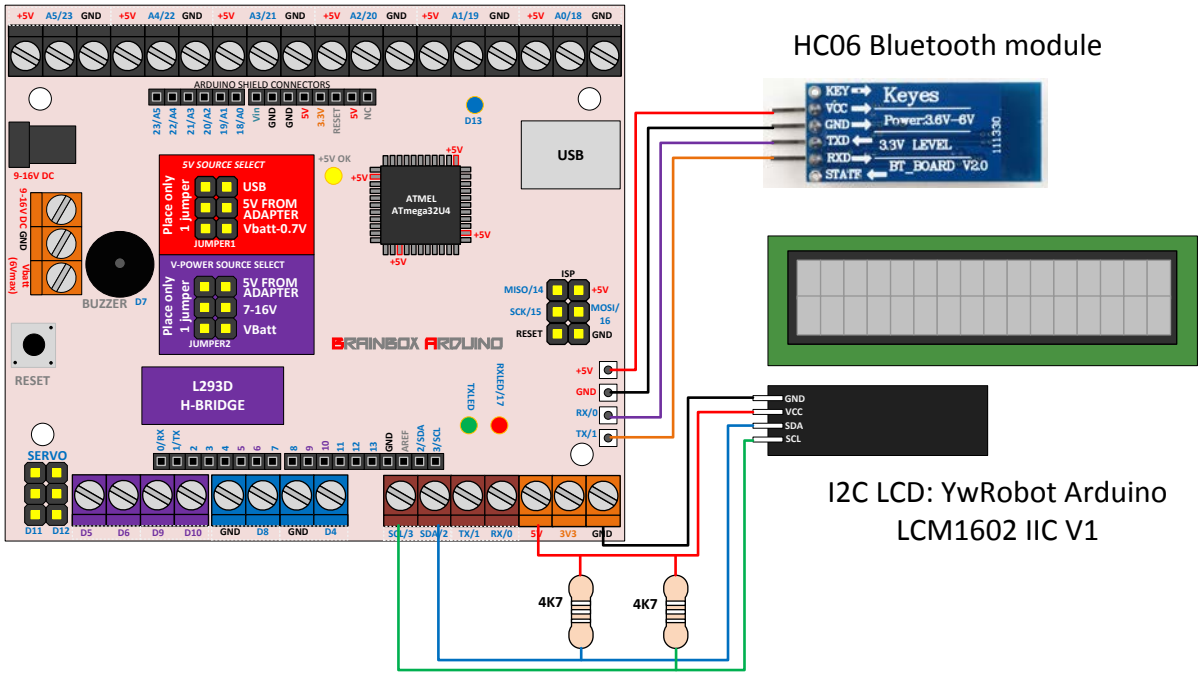


1. De microcontroller is ‘master’ en start de communicatie door ‘170’ door te sturen naar het Smartdevice (Tablet, Smartphone). De app test steeds of de eerste ontvangen byte wel ‘170’ is en beschouwt de 6 volgende bytes als corrupt als dat niet zo is.
2. De App is zo geschreven dat de APP steeds pas 5 bytes data naar de uC zal verzenden meteen nadat de APP de 7 bytes van de uC heeft ontvangen.
    a. Uit de meetresultaten blijkt duidelijk dat de APP op ons Smartdevice niet zo ‘realtime’ werkt als we graag zouden zien.
    b. Soms duurt het tot 100msec voordat de app de eerste byte verstuurt
    c. De tijden tussen de volgende 4 bytes verschillen ook elke keer en kunnen oplopen tot 75msec.
3. De eerste byte die door de APP gestuurd wordt is ‘85’ en deze wordt door de uC gebruikt om te controleren of de eerste byte wel degelijk de eerste byte is.

Werken met interrupts zou hier de beste oplossing zijn, maar we hebben het in onze voorbeeldprogramma’s ook klaargekregen zonder interrupts.

!! Houd in het achterhoofd dat deze programma's geen perfecte communicatie garanderen gedurende uren aan één stuk. Daarvoor zijn ze onvoldoende uitgetest.

Aansluiting HC06 en I2C LCD



VOORBEELDCODE AAN/UIT: **'RS232- HC06- NOINT'**