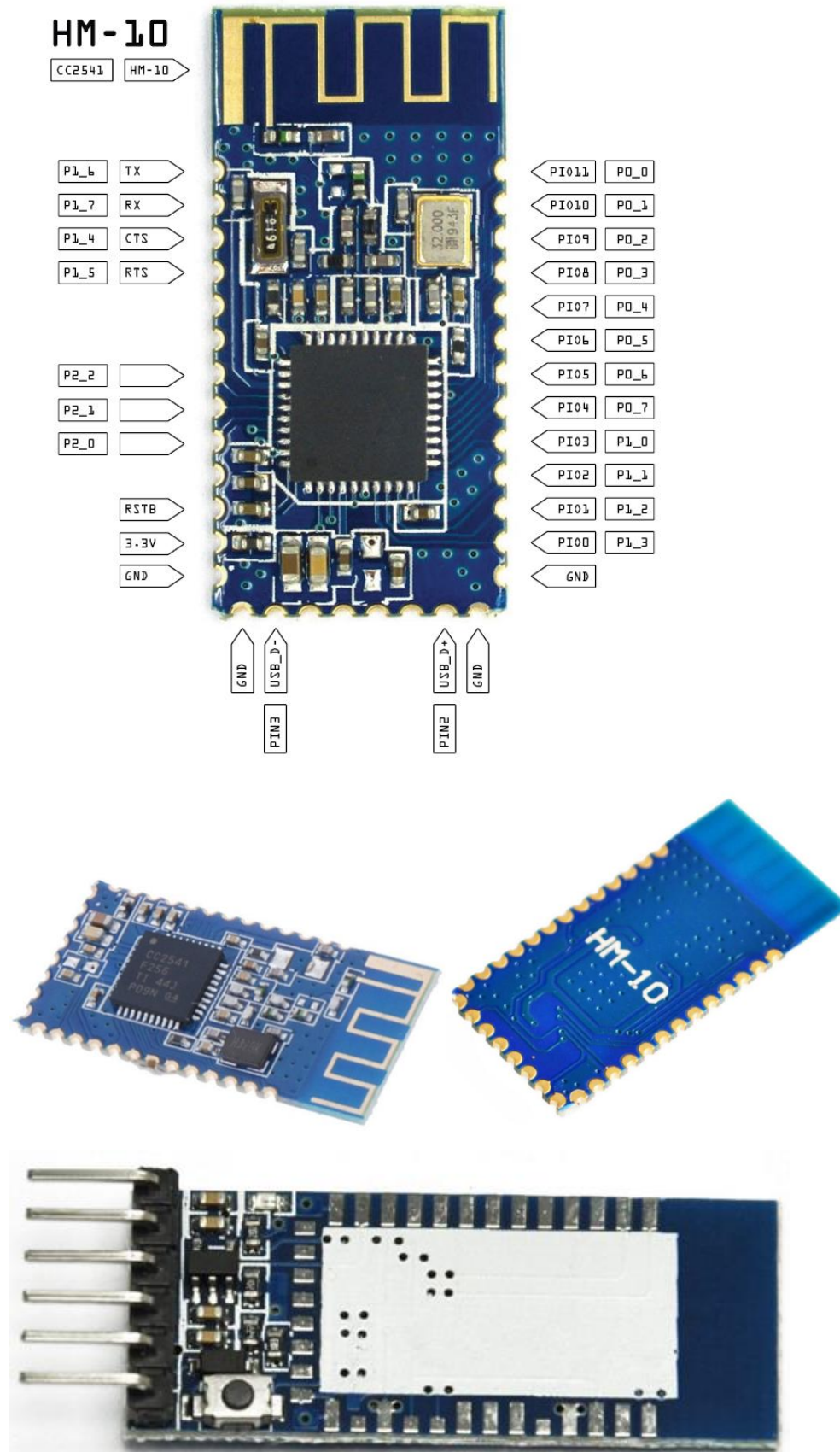
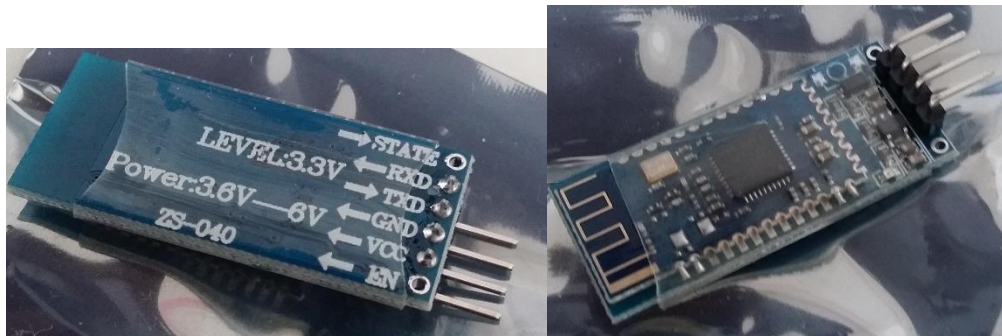


HM-10 Bluetooth BLE





A fenti képeken, a csatlakozás megfigyelése mellett az is észrevehető, hogy a modul körbe van véve egy vékony műanyag réteggel, ezzel is egy kicsit megóvva a bluetooth-t a külső feszültségektől. Erre azért van szükség, mert a bluetooth modul könnyen károsodhat, ha valaki a testében felhalmozódott statikus töltésekkel hozzáér az eszközhöz. Ez nem teljes körű védelem, így mindig legyünk meggyőződve arról, hogy mielőtt hozzáérünk a modulhoz, levezettük a testünkben lévő töltéseket egy fém tárgyon, például egy radiátoron.

Models	VDD	Size(mm)	Flash	Chip	BT Version
HM-10	2-3.7V	26.9*13*2.2	256Kb	CC2540/1	V4.0 BLE

A régi 530 Firmware verzióhoz tartozó tulajdonságok.

BT Version: Bluetooth Specification V4.0 BLE

Send and receive no bytes limit.

Working frequency: 2.4GHz ISM band

Modulation method: GFSK(Gaussian Frequency Shift Keying)

RF Power: -23dbm, -6dbm, 0dbm, 6dbm, can modify through AT Command AT+POWE.

Speed: Asynchronous: 6K Bytes

Synchronous: 6K Bytes

Security: Authentication and encryption

Service: Central & Peripheral UUID FFE0,FFE1

Power: +3.3VDC 50mA

Long range: Open space have 100 Meters with iphone4s

Power: In sleep mode 400uA~1.5mA, Active mode 8.5mA.

Working temperature:-5 ~ +65 Centigrade

Size: HM- 10 26.9mm x 13mm x 2.2 mm; HM-11 18*13.5*2.2mm

Az új 621-es verziószámú firmware dokumentációjához már az alábbi tulajdonságok tartoznak:

BT Version: Bluetooth Specification V2.1+EDR

Working frequency: 2.4GHz ISM band

Modulation method: GFSK(Gaussian Frequency Shift Keying)

RF Power: $\leq 4\text{dBm}$, Class 2

Sensitivity: $\leq -84\text{dBm}$ at 0.1% BER

Speed: Asynchronous: 2.1Mbps(Max) / 160 kbps

Synchronous: 1Mbps/1Mbps(Max)

Security: Authentication and encryption

Service: Bluetooth SPP(Master & Slave)

Power: +3.3VDC 50mA

Working temperature: $-5 \sim +65$ Centigrade

Size: 26.9mm x 13mm x 2.2 mm、27.4*12.5*4.3mm、 etc.

Az általunk megvásárolt szerkezet, mint utólag kiderült egy klón, ami igazából a **BLE-CC41-A** névre hallgató eszköz. Jóval korlátozottabb, mint HM-10-es eredeti változat. Legfőbb hiányossága, hogy a Firmware-e nem upgrade-elhető az AT+SBLUP paranccsal. A parancslista és a parancsok szerkezete is kicsit eltér, erre a későbbiekben kitérünk. A klónhoz és a hozzá tartozó 306-os Firmware-hez az alábbi tulajdonságok tartoznak.

Bluetooth protokoll: Bluetooth V4.0 BLE, nincs byte korlát,

110m-es kommunikáció nyílt környezetben iPhone4S-el

Működési frekvencia: 2.4GHz ISM band

Moduláció: GFSK(Gaussian Frequency Shift Keying)

Érzékenység: $\leq -84\text{dBm}$ at 0.1% BER

Átviteli sebesség: Asynchronous: 6 kbps Synchronous: 6 kbps

Biztonsági funkciók: Authentication and encryption

Support Services: Central & Peripheral UUID FFE0, FFE1

Fogyasztás: alvó üzemmód automatikus, készenléti áram $400\mu\text{A} \sim 1.5\text{mA}$, átvitel 8.5mA

Áramellátás: +3.3VDC 50mA

Méret: 26.9mm x 13mm x 2.2 mm

Bluetooth minősítés: ROHS REACH BQB

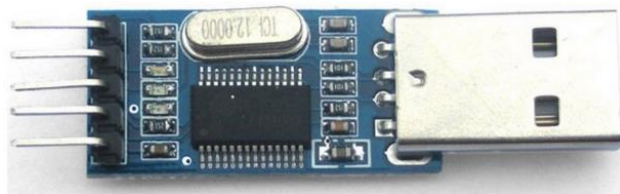
A bluetooth meghajtásához a 1,2V feszültségű nikkel-metán-hibrid (NiMH) akkumulátorból 3-at összekötve, $1,2 \cdot 3 = 3,6\text{V}$ -ot kapunk. Ezzel könnyedén meghajthatjuk a modult. Ezeknek az akkumulátoroknak a névleges töltő kapacitása 1100-1600mAh között lehet. Ha a legalacsonyabb kapacitást feltételezzük, azaz azt, hogy az akkumulátor 1 órán keresztül képes leadni a maximum 1100 mA áramát, akkor az aktív módban 8,5mA fogyasztó bluetooth modult $1100/8,5 = 129,41$, azaz kicsit több mint 129 órán keresztül képes folyamatos működés mellett üzemeltetni. Ugyan kevesebb ideig üzemel, mint egy alkáli elem, de nagy előnye, hogy újratölthető.

Alkáli elemek esetén a 1,5V feszültségű elemből 2-őt összekötve, $1,5 \cdot 2 = 3\text{V}$ -ot kapunk. Ezeknek az elemeknek 2700mAh körül van a kapacitásuk, ami $2700/85 = 317,64$ órán keresztül képes táplálni a bluetooth eszközünket.

Amennyiben a RC2032-es 3V-os lítiumos gomelemet használjuk, melynek 220-240mAh között van a kapacitása, akkor $220/8,5 = 25,88$, akkor körülbelül 26 órán keresztül tudjuk vele egyfolytában üzemeltetni bluetooth modulunkat.

A breadboard-nak köszönhetően azonban a 2-3,7V-os tápfeszültség tartomány helyett, használhatunk 3,6-6V közöttit is a bluetooth táplálására. Ez azt jelenti, hogy könnyedén meghajthatjuk a bluetooth-t egy 4,5V-os zsebleppel, vagy 4db 1,5V-os alkáli elemmel is.

Természetesen ahhoz, hogy egy ilyen modult USB-n keresztül soros kommunikációra tudjunk bírni szükségünk lesz egy USB to TTL logikai szint illesztő adapterre. Erre azért van szükség, mert az USB port 5V-os feszültségű jelekkel kommunikál, a bluetooth modulunk viszont csak 3,3V-al. A szint illesztőt úgy kell használnunk, hogy RX portját rákötjük a bluetooth modul TX portjára, TX portját pedig az RX portjára. A GND portját rákötjük a bluetooth modul földjére, vagy egy azzal azonos szinten lévő föld pontra és már csak rá kell dugnunk a számítógép USB portjára és használható is.



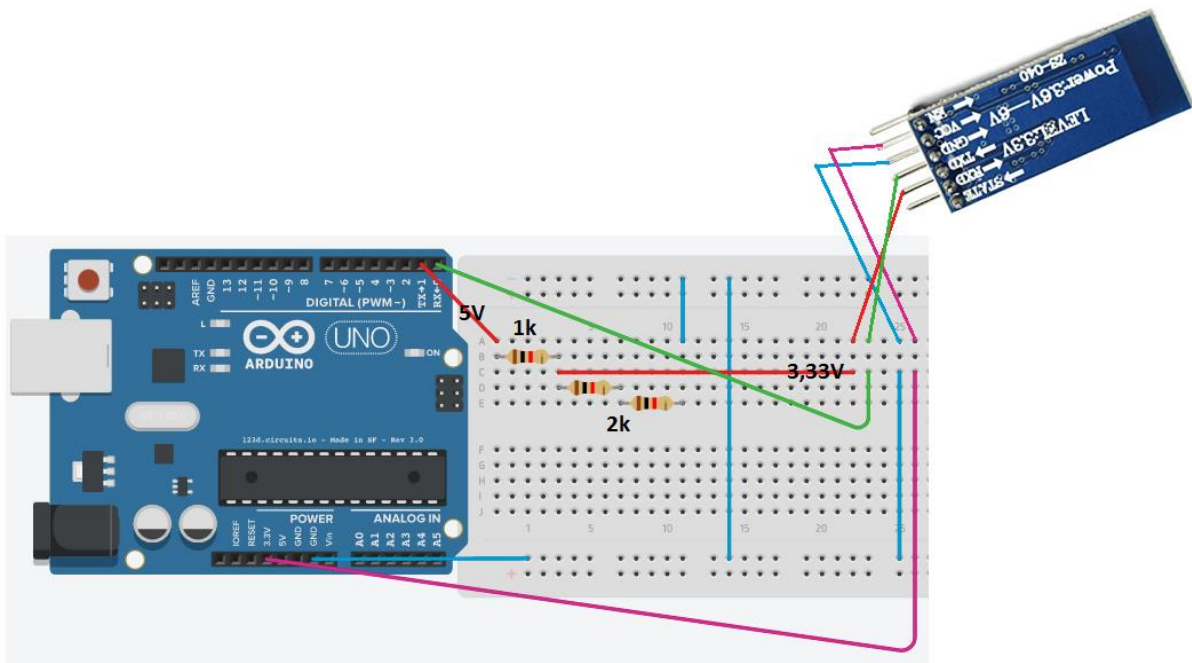
Ha nincs nálunk egy ilyen, akkor erre a célra használhatjuk az Arduino-nkat is, ugyanis ezekben az eszközökben alapjáraton megtalálható egy ilyen átalakító beépítve. Az egyetlen hátrány amit ilyenkor figyelembe kell venni, hogy az Arduino 5V-on üzemel, a bluetooth-nak pedig 3,3V szükséges. Ezért az Arduino TX, adatot küldő lábára egy feszültségosztót kell kössünk, ami az 5V-értékű magas jelet leosztja 3,3V-al kompatibilis magas jellé. Ezzel a működéssel elvileg elérhetünk 2V-nál magasabb, de 3,3V-ot nem meghaladó magas jelet, amelyet a bluetooth modulunk vár. A bluetooth azonban csak 3,3V-nak megfelelő magas jelet küld a kommunikáció másik oldalán, az Arduino pedig 5V-nak megfelelő magas jelet vár. 3,3V-nál olyan jel számít magasnak, ami 2V-nál nagyobb. 5V-nál pedig csak az olyan jelek számítanak magasnak, melyek 3V-nál nagyobb feszültségűek. Így az Arduino csak azokat a jeleket fogja magas jelnek venni, melyek 3 és 3,3V között vannak, de a 3,3V-ál még szintén magasnak számító 2V és 3V közötti feszültségszinteket nem. Mivel most a bluetooth-unk az Arduino-ról fogja kapni a fixen 3,3V feszültséget, ezért remélhetőleg a kimenetén lévő TX lábon is csak 3V feletti magas jelet fog küldeni. Gyengébb tápellátásnál meglehet, hogy a küldött adat hibásan fog megérkezni, mert a bluetooth-ról küldött logikai magas érték lehet, hogy az Arduino-n nem fog logikai magas értéknek megfelelni. Tehát egy ilyen megoldás mellett ha a bluetooth kimeneti áramköre nincs nagyon terhelve, akkor a 3,3V már logikai magas szintnek számít, ellenkező esetben, vagy ha zajos környezetben alkalmazzuk az eszközöket, például egy fizikai motor közvetlen

közelében, akkor meg van az adatvesztés lehetősége. Ilyen környezetekben használhatunk tranzisztort, vagy fet-et a jelszint megemelésére.

Feszültségosztás:

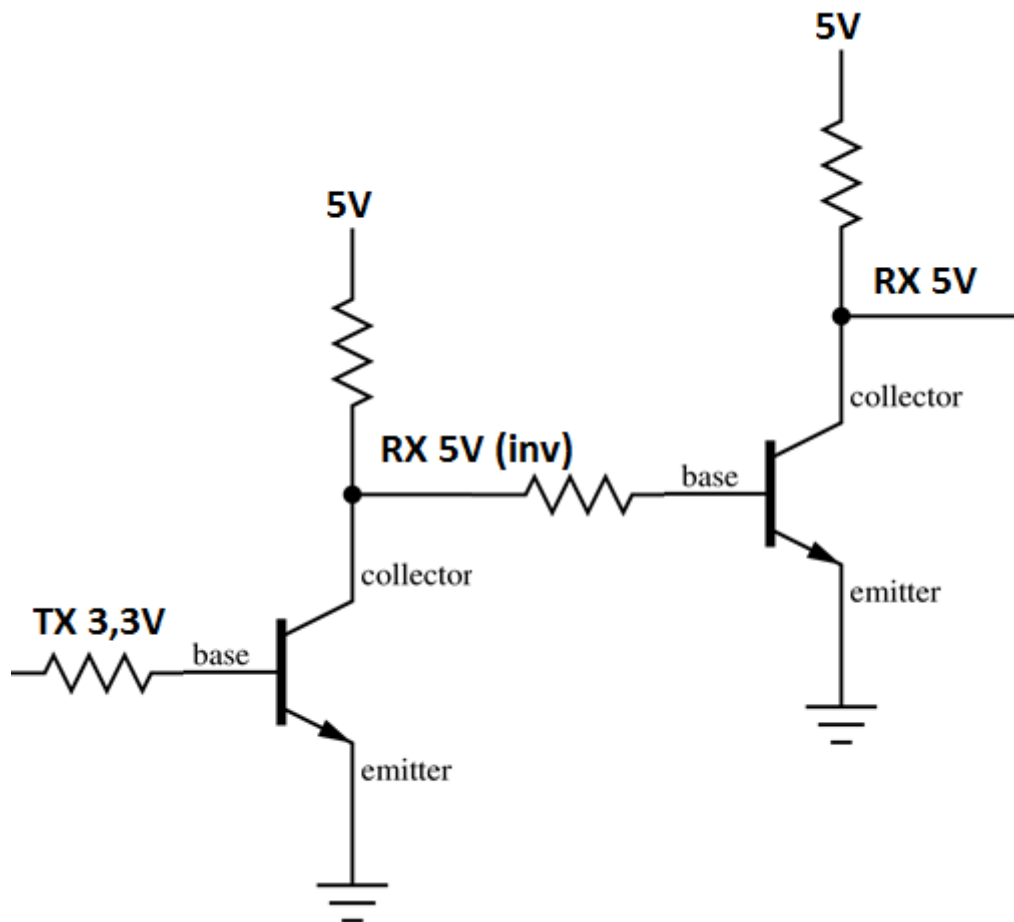
$$U_{ki} = U_{be} * (R_{esik} / R_{összes})$$

Ki kell tehát találnunk, hogyha 3,3V-ot szeretnénk kapni U_{ki} -re és 5V az U_{be} , akkor milyen legyen a két ellenállás értékünk. Természetesen nem fontos pontosan 3,3V-os jelet generálnunk, jó nekünk 2V és 3,3V között bármi magas jelnek. Egy egyszerű megoldás, ha egy 1kOhm-os és egy 2kOhm-os ellenállást választunk (a 2kOhm-osat helyettesíthetjük 2db sorba kötött 1kOhm-ossal is). Ekkor $5V * (1kOhm / (1+2)kOhm) = 5 * (1/3) = 1,67V$ feszültség esik az 1kOhm-os ellenálláson, illetve $5 / (2/3) = 3,33V$ feszültség esik a 2kOhm-os ellenálláson. Jól látható, hogy nekünk utóbbira volt szükségünk, amikor az Arduino magas 5V-os feszültségű jele helyett, a bluetooth-nak megfelelő magas 3,3V-os feszültségű jelet szerettünk volna küldeni a bluetooth modulunknak, az Arduino felől az Arduino TX csatornáján.



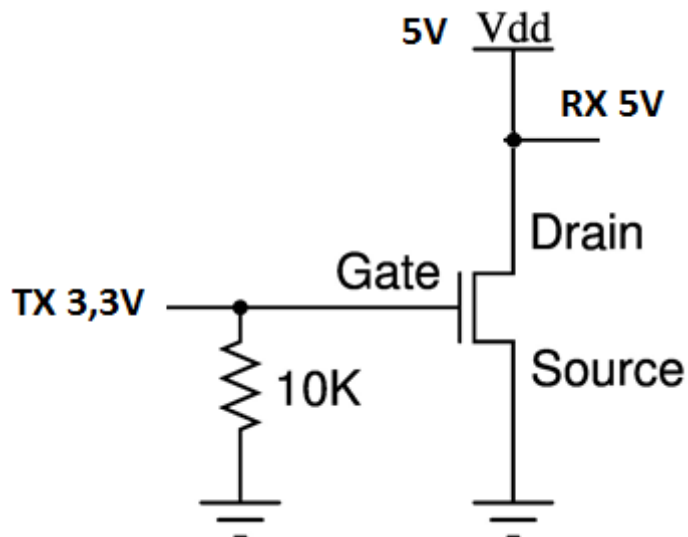
Tranziszor:

Tranzisztoros szintillesztésnél nincs más dolgunk, mint a fenti kapcsolásban a zöld vezetéket megszakítani és közbe ékelni az alábbi kapcsolást, melyben az n csatornás tranzisztor egy kapcsolóként szolgál. Amennyiben a tranzisztor bázisára megfelelő feszültséget juttatunk, akkor a kollektor és emitter között a tranzisztor kinyit és a kollektor felől a földelt emitter felé feszültség fog esni. Ha jól választjuk meg a tranzisztort, akkor 2-3,3V-os nyitófeszültség között fog nyitni. Ha a kollektor feszültségét 5V-os tápra kötjük, akkor ezt a feszültséget fogjuk elküldeni a 3,3V-os feszültség helyett az Arduino-nak, így biztosan nem lesz jelvesztés, mert a szintillesztés megoldja a problémánkat. Egy tranzisztor után a jel még invertált, ezért két tranzisztoron keresztül kell átküldenünk. A bal szélső és köztes ellenállás azért szükséges, hogy a kívülről érkező jelet nyitófeszültségűre hozzuk. A kollektornál lévő ellenállás pedig azért van, hogy a tranzisztor vezetése esetén beállítsa az áramot. $I = U/R$



Fet:

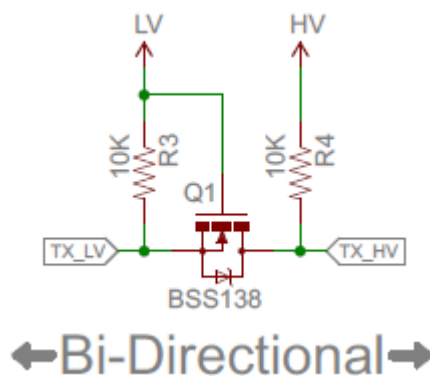
A fet-el való megoldás hasonlít a tranzisztoros-ra, hiszen MOSFET gyakorlatilag metal-oxide-semiconductor field-effect transistor, tehát egy tranzisztor, ami feszültség vezérelt. Ahogyan nő a feszültsége, annál több áramot kapcsol. A sima tranzisztornál egy terhelő ellenállást kapcsoltunk a meghajtó áramkörre, ami terheli azt. Azért terhelő, mert sorosan van vele kötve. Ezen az ellenálláson esik a feszültség és ennek a feszültségesésnek a hatására keletkezik áram. A fet azért jobb, mert nem terheli a meghajtó áramkört, hiszen nincs terhelő ellenállás a kapcsolásban. A kapcsolásban csak egy lehúzó ellenállás található. A lehúzó vagy felhúzó ellenállások párhuzamosan vannak bekötve az áramkörrel és szerepük, hogy a lebegő logikai szinteket lehúzzák logikai alacsony, vagy felhúzzák logikai magas szintre. Erre azért van szükség, mert amikor a TX porton, ha éppen nem küldünk jelet, és felszedi a környezetéből a zajokat / a külső feszültséget, akkor tévesen magas jelet jelezhetne, ekkor a nagy értékű ellenállás lehúzza a lebegő jelet alacsony szintre, így nem tud tévesen magasat jelezni. (elvileg ide is kellene terhelő ellenállás az RX elé) illetve (elvileg ez is invertál, szóval itt is kétszer kell egymás után kötni a jelet)



Kétirányú szintillesztővel is megoldható a 3,3V-os szintről az 5V-os szintre való illesztés, illetve a bidirectional miatt fordítva is kihasználható.

$$RX_LV\ Max = HV \times (0.67)$$

HV	LV	RX_LV Max
5V	3.3V	3.3V
3.3V	2.8V	2.2 V
5V	2.8V	3.3V
5V	1.8V	3.3V
3.3V	1.8V	2.2V



Miután minden vezetékezést megoldottunk, csatlakoztassuk az Arduino-t a számítógépre, amelyre előzőleg rácsatlakoztattuk a bluetooth modult. A bluetooth modul alapjáraton 9600-as baud rátával dolgozik, ezért amikor megnyitjuk az Arduino Software IDE, Serial Monitor nevű soros porti terminálját, akkor ezt válasszuk ki baud rátának a terminálban is. Most teszteljük le, hogy a modulunk működik-e. Töltsük fel az Arduino-ra az alábbi kódot. Én végül Arduino Mega-ra csatlakoztattam a bluetooth modult, így a bluetooth TX portját az Arduino RX3 lábára kötöttem. Ennek megfelelően, aki máshogy csinálta, annak kicsit módosítania kell a kódon.

```
void setup()
{
  Serial.begin(9600);
  Serial3.begin(9600);
}

void loop(){
  while(Serial3.available()>0){
    String s =Serial3.readString();
    Serial.println("Revceived from RX3: " + s);
  }
}
```



```

while(Serial.available()>0){
  String s =Serial.readString();
  Serial.println("Sent from TX0: " + s);
  Serial3.println(s);
}
}

```

Arduino Uno-n a Bluetooth-nak egy szoftveres soros megoldást kell alkalmaznia, hiszen ott csak egy hardveres RX, TX lehetőség van, amit pedig a terminál fog használni. Ekkor egy SoftwareSerial objektumot kell létrehozni, például a D2-es és D3-as port-tal, amelyre a bluetooth RX és TX lábait kötjük. És a továbbiakban ezt az objektumot kell használni az olvasásra és írásra. Az alábbi példában azért a D10-et és D11-et használjuk, mert a Mega esetében nem minden port használható RX-nek, csak a D10-D15-ig, a D50-53-ig, valamint az A8-A15-ig, amelyek számmal a 62-69-ig érhetőek el. A software-es soros portoknál fontos tudni, hogyha többet használunk egyszerre, akkor egy időben csak egy képes olvasni.

SoftwareSerial bluetooth(10,11); //D10,D11 (Bluetooth RX, Bluetooth TX)

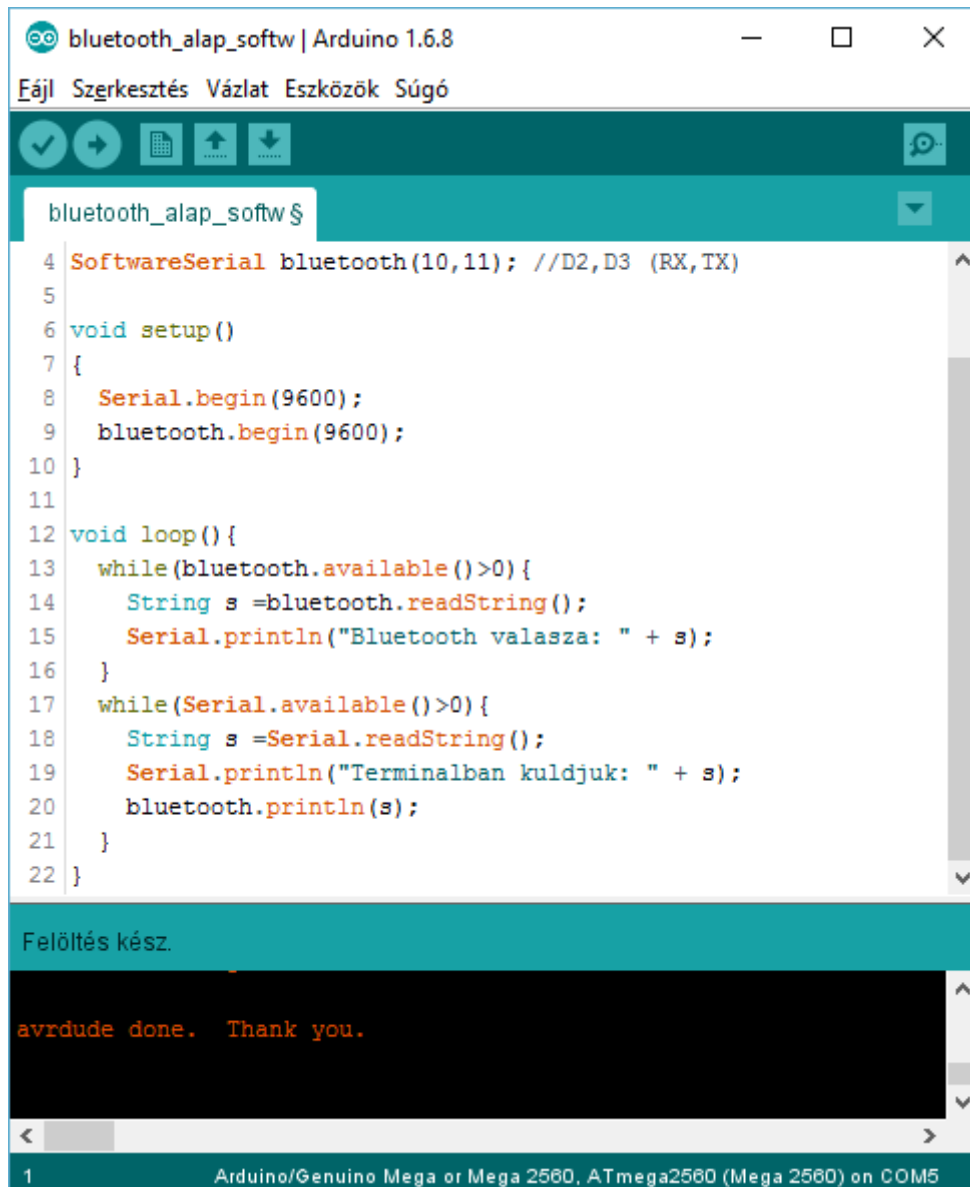
```

void setup()
{
  Serial.begin(9600);
  bluetooth.begin(9600);
}

void loop(){
  while(bluetooth.available()>0){
    String s =bluetooth.readString();
    Serial.println("Revceived from RX10: " + s);
  }

  while(Serial.available()>0){
    String s =Serial.readString();
    Serial.println("Sent from TX0: " + s);
    bluetooth.println(s);
  }
}
}

```



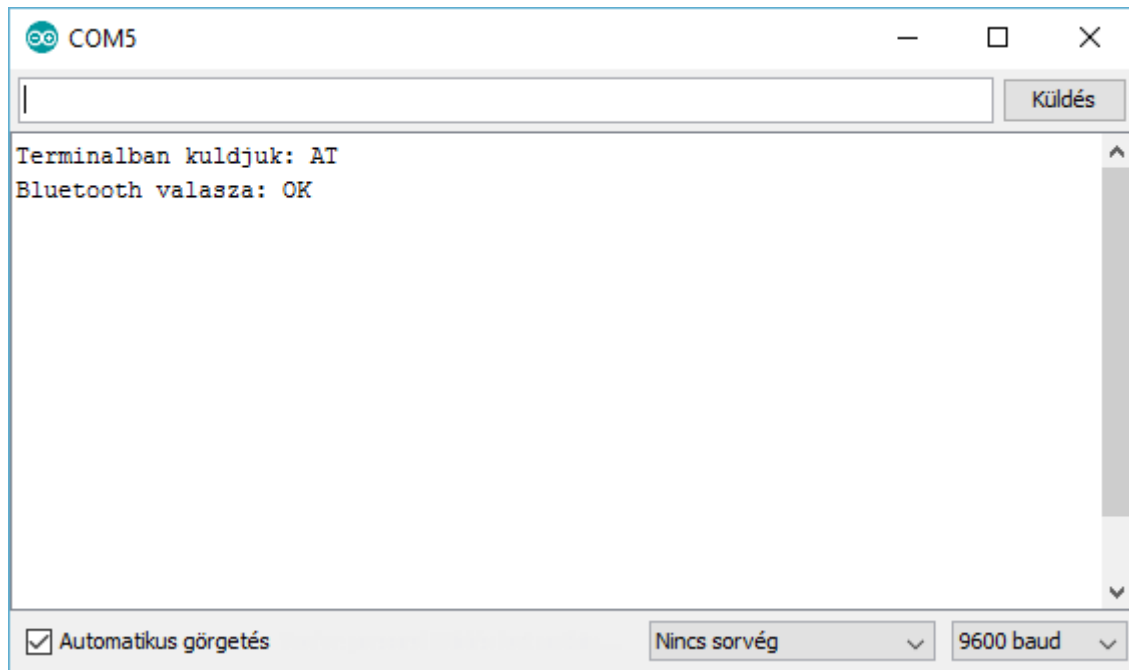
```
bluetooth_alap_softw $  
4 SoftwareSerial bluetooth(10,11); //D2,D3 (RX,TX)  
5  
6 void setup()  
7 {  
8   Serial.begin(9600);  
9   bluetooth.begin(9600);  
10 }  
11  
12 void loop() {  
13   while(bluetooth.available()>0) {  
14     String s =bluetooth.readString();  
15     Serial.println("Bluetooth valasza: " + s);  
16   }  
17   while(Serial.available()>0) {  
18     String s =Serial.readString();  
19     Serial.println("Terminalban kuldjuk: " + s);  
20     bluetooth.println(s);  
21   }  
22 }
```

Felöltés kész.

avrdude done. Thank you.

1 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM5

Ha az AT parancsot kiadjuk a terminalon és az OK-ot kapjuk válaszul, akkor elértük a bluetooth modult:



Eddig minden ugyanúgy működik HM-10-en és a klón BLE-CC41-A modulon is. Ám a parancsok innentől kezdve kicsit eltérhetnek. A parancsok listáját a klónon az AT+HELP, az eredeti modulon pedig az AT+HELP? paranccsal kérhetjük le. A parancsok nagyjából hasonlítanak egymáshoz, de innentől kezdve mindenki alkalmazza a modulnak megfelelőt. A dokumentum további része a klón parancsaival foglalkozik.

Command	Description	

* AT	Check if the command terminal work normally	*
* AT+RESET	Software reboot	*
* AT+VERSION	Get firmware, bluetooth, HCI and LMP version	*
* AT+HELP	List all the commands	*
* AT+NAME	Get/Set local device name	*
* AT+PIN	Get/Set pin code for pairing	*
* AT+PASS	Get/Set pin code for pairing	*
* AT+BAUD	Get/Set baud rate	*
* AT+LADDR	Get local bluetooth address	*
* AT+ADDR	Get local bluetooth address	*
* AT+DEFAULT	Restore factory default	*
* AT+RENEW	Restore factory default	*
* AT+STATE	Get current state	*
* AT+PWRM	Get/Set power on mode (low power)	*
* AT+POWE	Get/Set RF transmit power	*
* AT+SLEEP	Sleep mode	*
* AT+ROLE	Get/Set current role.	*
* AT+PARI	Get/Set UART parity bit.	*
* AT+STOP	Get/Set UART stop bit.	*
* AT+START	System start working.	*
* AT+IMME	System wait for command when power on.	*
* AT+IBEA	Switch iBeacon mode.	*
* AT+IBEO	Set iBeacon UUID 0.	*
* AT+IBE1	Set iBeacon UUID 1.	*
* AT+IBE2	Set iBeacon UUID 2.	*
* AT+IBE3	Set iBeacon UUID 3.	*
* AT+MARJ	Set iBeacon MARJ .	*
* AT+MINO	Set iBeacon MINO .	*
* AT+MEA	Set iBeacon MEA .	*
* AT+NOTI	Notify connection event .	*
* AT+UUID	Get/Set system SERVER_UUID .	*
* AT+CHAR	Get/Set system CHAR_UUID .	*

* Note: (M) = The command support slave mode only.		*
* For more information, please visit http://www.bolutek.com		*
* Copyright©2013 www.bolutek.com. All rights reserved.		*

Az AT+VERSION paranccsal lekérhetjük a bluetooth firmware-ének verzióját. Ez a parancs az eredeti modulnál például AT+VERR? .

```
Terminalban küldjük: AT+VERSION
Bluetooth válasza: +VERSION=Firmware V3.0.6,Bluetooth V4.0 LE
```

Az AT+NAMECIWSduino paranccsal az eszközt elnevezhetjük CIWSduino-nak, majd a sima AT+NAME paranccsal lekérdezzhetjük hogy átállt-e a név, illetve, hogy jelenleg mi a bluetooth eszköz neve.

```
Terminalban küldjük: AT+NAMECIWSduino
Bluetooth válasza: +NAME=CIWSduino
OK
```

```
Terminalban küldjük: AT+NAME
Bluetooth válasza: +NAME=CIWSduino
```

Az AT+PIN, vagy AT+PASS parancs valamelyikével lekérdezhethetjük az eszköz párosításához használatos aktuális PIN kódot, amely egy 000000 és 999999 közötti szám lehet. Az alapbeállítás 000000.

```
Terminalban küldjük: AT+PIN
Bluetooth válasza: +PIN=000000
```

```
Terminalban küldjük: AT+PASS
Bluetooth válasza: +PASS=000000
```

Én átállítom 198709 az AT+PASS198709 segítségével, de ugyanúgy használhatnánk az AT+PIN-t erre a célra is.

```
Terminalban küldjük: AT+PASS198709
Bluetooth válasza: +PASS=198709
OK
```

Az AT+BAUD parancssal lekérdezhethetjük a bluetooth modulunk baud rátáját. Alapbeállításon ez 4-et ad értékül, amely a 9600-as értéket jelenti.

```
1---1200
2---2400
3---4800
4---9600
5---19200
6---38400
7---57600
8---115200
9---230400
```

```
Terminalban küldjük: AT+BAUD
Bluetooth válasza: +BAUD=4
```

Ezt az értéket átállíthatjuk például 115200-ra az AT+BAUD8 parancssal.

```
Terminalban küldjük: AT+BAUD8
Bluetooth válasza: +BAUD=8
OK
```

De ekkor ne felejtjük el átállítani a kódunkat is azonos értékűre és újra feltölteni az Arduino-ra a módosított kódunkat, mert különben nem kapunk választ, hiszen a bluetooth-nak a jeleket mi csak 9600baud-al küldjük, de az 115200-al próbálja a parancsokat értelmezni, ami egy értelmezhetetlen parancsot fog számára eredményezni, így választ sem küld rá.

```
void setup()
{
  Serial.begin(9600);
  bluetooth.begin(115200);
}
```

Ekkor előfordulhat adatvesztésből fakadó torzulások a szövegben. Az alábbi példa például az AT+HELP parancsra adott válasz.


```

* AT+BAUD          Get/SÚ7&Q«#Q&QŁ+K          *
* AT+LADDR         Get local bluetooth address    *
* AT+ADDR          Get local bluetooth address    *
* AT+DEFAULT       Restore factory default        *
* AT+RENEW         Restore factory default        *
* AT+STATE         Get current state              *
* AT+PWRM         Get/Set power on mode(low power) *
* AT+POWE         Get/Set RF transmit power      *
* AT+SLEEP        Sleep mode                     *
* AT+R0Z400000000000v+L{6-676«""+sŁQ&{c+s-000000000000000000!Y0á* AT+PAII

```

Terminalban küldjük: AT

Terminalban küldjük: AT+HELP

☒ Automatikus görgetés

Nincs sorvég

9600 baud

Én most visszaállítom a 9600-as baud rátát az AT+BAUD4 paranccsal és a forráskódban is visszaírom a 115200-at 9600-ra. Fontos a sorrend, különben rossz baud rátával küldjük vissza a parancsot és jelenleg a 115200-al kell küldenünk, csak ezután állíthatjuk át és tölthetjük fel az Arduinora az új programunkat.

Ezek után az AT+LADDR vagy AT+ADDR paranccsal lekérdezhethetjük a modul helyi MAC címét.

Terminalban küldjük: AT+ADDR

Bluetooth válasza: +ADDR=00:15:83:00:6F:76

Terminalban küldjük: AT+LADDR

Bluetooth válasza: +LADDR=00:15:83:00:6F:76

Az AT+STATE paranccsal lekérdezhető, hogy a bluetooth milyen módban van. Alapjáraton 2-es állapotban van.

0: Transmission Mode

1: PIO collection Mode +

Mode 0

2: Remote Control Mode

+ Mode 0

A 0-s állapot, amikor UART-on AT parancsokkal beállítjuk az eszközt és utána ha kialakult a kapcsolat, akkor adatokat tudunk küldeni.

Az 1-es állapot ugyanez, de ekkor a távoli eszköz tud távolról AT parancsokat küldeni bluetooth-on keresztül. Valamint HM-10 esetén a PIO04 és PIO11 közötti lábak bemeneti jelét be tudjuk gyűjteni és távolról tudjuk vezérelni a PIO2 és PIO 3 kimeneti állapotát. UART-on adatot tudunk küldeni (nem AT parancsokat, maximum 20byte-ot).

A 2-es állapot ugyanaz mint az 1-es, de ekkor szintén tud a távoli eszköz AT parancsokat küldeni bluetooth-on keresztül. HM-10 esetén ekkor a PIO2 és PIO11 közötti lábak kimeneti állapotát tudjuk távolról vezérelni. UART-on adatot tudunk küldeni (nem AT parancsokat, maximum 20byte-ot).

```
Terminalban kuldjuk: AT+STATE
Bluetooth valasza: +STATE=2
OK
```

Az AT+PWRM paranccsal lekérhetjük a power módot. Alapjáraton ez 1-et ad vissza, azaz nem megy el automatikusan alvó módba a bluetooth, csak ha megkapta az AT+SLEEP parancsot. Ha ezt átállítjuk 0-ra, akkor az eszköz automatikusan alvó módba lép, ha nincs használatban. Alvó módban az eszköz jóval kevesebbet fogyaszt, akkor célszerű a használata, ha spórolni szeretnénk az akkumulátorral. Az adatlap szerint ilyenkor 8,5mA helyett 400uA-1,5mA között fogyaszt a bluetooth.

```
Terminalban kuldjuk: AT+PWRM
Bluetooth valasza: +PWRM=1
```

Ha szeretnénk, akkor átállíthatjuk az AT+PWRM0 paranccsal, hogy automatikusan mennyen aludni, de én ezt most nem fogom megtenni.

Ahogy korábban is említettem manuálisan is elküldhetjük aludni az eszközünket az AT+SLEEP paranccsal.

```
Terminalban kuldjuk: AT+SLEEP
Bluetooth valasza: +SLEEP
OK
```

Alvó állapotból úgy tudjuk visszahozni az eszközt, hogy egy 80 karakternél hosszabb szöveget küldünk a számára. HM-10-nél még rendelkezésre áll erre a feladatra egy gomb melyet, ha 1 másodpercnél tovább nyomunk, akkor feléled az eszköz. Az alábbi képen jól látható, hogy gyakorlatilag egyetlen a karakter elküldésével feléleszthetjük a bluetooth-t.

```
Terminalban kuldjuk: a
Bluetooth valasza: +WAKE
OK
```

Az AT+POWE parancs visszaadja a rádió frekvenciás átvitel erejét. Alapjáraton 2-vel tér vissza, ami a 0dB-es erősítetlen jelet jelenti.

0: -23dbm

1: -6dbm

2: 0dbm

3: 6dbm

```
Terminalban kuldjuk: AT+POWE
Bluetooth valasza: +POWE=2
```

Ha akarjuk, változtathatjuk az erejét. Erősíthetjük 6db-re vagy visszagyengíthetjük -6, vagy -23db-re. Például az ATPOWE3 paranccsal felerősíthetjük a jelet 6db-re. Ez egyben azt is eredményezi, hogy megnő a bluetooth fogyasztása. Jelenleg erre nincs szükség, így nem állítom át.

Az AT+ROLE parancs visszaadja a master-slave szabályt. Alapértelmezettként 0, azaz periféria szerepét tölti be a bluetooth. 1 esetén pedig központi szerepet. Ezt is át lehet állítani az AT+ROLE1 paranccsal központira, de nekünk most nincs erre szükségünk.

Terminalban küldjük: AT+ROLE
Bluetooth válasza: +ROLE=0

Az AT+PARI paranccsal az UART kommunikáció során használt paritás bit beállításait kérdezhetjük le. Alapértelmezettként ez 0, azaz nincs. 1 esetén páros, 2-esetén páratlan.

Terminalban küldjük: AT+PARI
Bluetooth válasza: +PARI=0

Az AT+STOP paranccsal az UART kommunikáció során használt stop bit beállításait kérdezhetjük le. Alapértelmezettként ez 0, azaz 1 stop bit van. 1 esetén pedig 2 stop bit.

Terminalban küldjük: AT+STOP
Bluetooth válasza: +STOP=0

Az Arduino alapjáraton 8N1-el kommunikál. Azaz 8 adatbit van, nem használ paritásbitet és 1 db stop bitet használ. Így a fentebbi beállításokat úgy hagyhatjuk, ahogy vannak. Szükség esetén átállítható mind a paritás, mind a stop bit beállításai. Például, ha 2 stop bitet szeretnénk, használhatjuk az AT+STOP1 parancsot. Ha ilyen kommunikációt használunk, akkor át kell állítanunk a Serial beállításait is. Ebben az esetben például 8N2-re az alábbi módon a bluetooth-hoz.

bluetooth.begin(9600,SERIAL_8N2);

A AT+NOTI paranccsal lekérdezhetjük az értesítések módját. 0 esetén nem kapunk, 1 esetén kapunk értesítést ha egy bluetooth eszköz sikeresen kapcsolódott, vagy ha elveszett a kapcsolat. Alapjáraton ez az érték 0, azaz nem kapunk értesítést. Az AT+NOTI1-el átállíthatjuk, de erre most nincs szükség.

Terminalban küldjük: AT+NOTI
Bluetooth válasza: +NOTI=0

Az AT+UUID paranccsal lekérdezhető a szolgáltatás UUID értéke. ez 0x0001 és 0xFFFE között bármilyen hexadecimális érték lehet. Azaz 1 és 65534 közötti érték, hexadecimális megfelelője. Alapjáraton ez az érték 0xFFE0, ami a 65504-nek felel meg decimálisan.

Az AT+CHAR paranccsal lekérdezhető a karakterisztika értéke. Ez szintén az előbbi 0001 és FFFE közötti tartományban mozoghat, de alapértelmezett értéke 0xFFE1, ami 65505.

Terminalban küldjük: AT+UUID
Bluetooth válasza: +UUID=0xFFE0

Terminalban küldjük: AT+CHAR
Bluetooth válasza: +CHAR=0xFFE1

Szükség esetén a parancs után közvetlenül beírt hexadecimális számjeggyel átállíthatóak, azaz 0001-es karakterisztikához az AT+CHAR0x0001 parancsot kell kiadunk. Ugyanígy működik az AT+UUID is. Ezekre most nincs szükségünk.

Az AT+IMME parancs használatával megtudhatjuk a modul működési típusát. Alapértelmezettként 0-át ad vissza, ami azt jelenti, hogy amint feszültséget kap a modul azonnal működésbe kezd. 1-es esetén csak az AT parancsokra reagál, egyébként nem csinál semmit, amíg az AT+START parancsot ki nem adjuk neki. Illetve amíg az AT+CON, vagy AT+CONNL parancsot nem használjuk. Ez a parancs csak központi szabály alkalmazásánál fontos. Az AT+IMME1 paranccsal szükség esetén átállítható.

Terminalban küldjük: AT+IMME
Bluetooth válasza: +IMME=0

Az AT+START parancs kiadását csak AT+IMME1 esetén használjuk. Ez a parancs azonnal munkába állítja az eszközt, ha az eszköz úgy van beállítva, hogy addig ne csináljon semmit, amíg ezt a parancsot meg nem kapta.

Az AT+TYPE parancs segítségével megtudhatjuk a modul párosításának módját. Alapértelmezett értéke 0, azaz nincs szükség pin-kódra a párosításhoz.

0:Not need PIN Code

1:Auth not need PIN

2:Auth with PIN

3:Auth and bond

Terminalban küldjük: AT+TYPE
Bluetooth válasza: +TYPE=0

Fontos, hogy 515-ös firmware alatt nem szabad használni ezt a parancsot. A 3-as opció csak 524-es firmware-től elérhető. 4.3-as Android alatt a 1-es és 2-es opció hasonló. Ha olyan eszközről szeretnénk beállítani, ahol szabad, ott az AT+TYPEszám paranccsal átállíthatjuk. Fontos, hogy ahhoz, hogy Android-ról rá tudjunk csatlakozni, be kell állítanunk egy nevet, egy pin kódot és ezt a type értéket 3-ra.

Ha az eszköz újraindítására van szükség, akkor az AT+RESET parancs kiadásával ezt megtehetjük. Az előző autentikációs beállítás után pedig ajánlott az újraindítás.

Az AT+ADVI paranccsal lekérhetjük a hirdetési intervallumot. Az IOS rendszerekhez a maximum ajánlás 1285ms. Ez azt jelenti, hogy az apple a 1285ms-ot engedélyezi, de a válaszban hogy scan-nel és kapcsolódott az hosszú idő lesz. Az értéket a szokásos módon változtathatjuk meg.

0: 100ms

1: 152.5 ms

2: 211.25 ms

3: 318.75 ms

4: 417.5 ms A: 2000ms

5: 546.25 ms B: 3000ms

6: 760 ms C: 4000ms

7: 852.5 ms D: 5000ms

8: 1022.5 ms E: 6000ms

9: 1285 ms F: 7000ms

Terminalban küldjük: AT+RESET
Bluetooth válasza: +RESET
OK

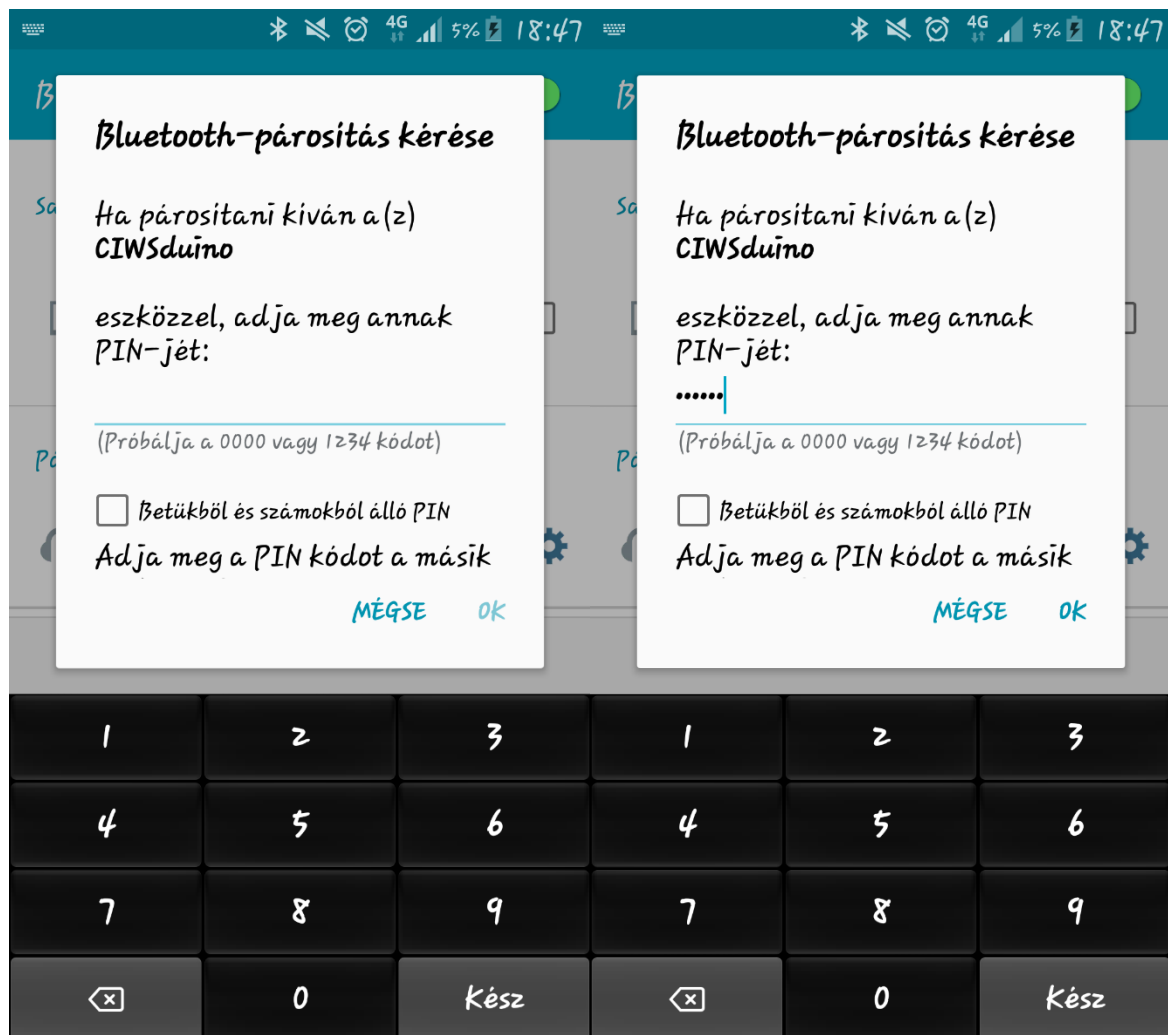
Ha az eszközt vissza szeretnénk állítani az alapértelmezett gyári beállításaira, akkor az AT+DEFAULT, vagy AT+RENEW parancsokat használhatjuk.

Terminalban küldjük: AT+DEFAULT
Bluetooth válasza: +DEFAULT
OK

Ha ilyenkor megnézzük az eszköz nevét, akkor láthatjuk, hogy visszaállt az alapbeállításra, a CC41-A névre.

Terminalban küldjük: AT+NAME
Bluetooth válasza: +NAME=CC41-A

Ha beállítottuk az eszköz nevét, pin kódját és az autentikáció típusát átállítottuk párosításra, majd újraindítottuk a bluetooth-t, akkor már rá is csatlakozhatunk egy külső eszközhöz, mondjuk egy Androidos mobiltelefonról.



Bluetooth



Saját eszköz



Cyber Zero (GT-I9506)

Csak párosított eszköz láthatja. Ér.
meg, hogy láthassák más eszközök.



Párosított eszközök



CIWSduino

Párosítva



Elérhető eszközök

KERESÉS