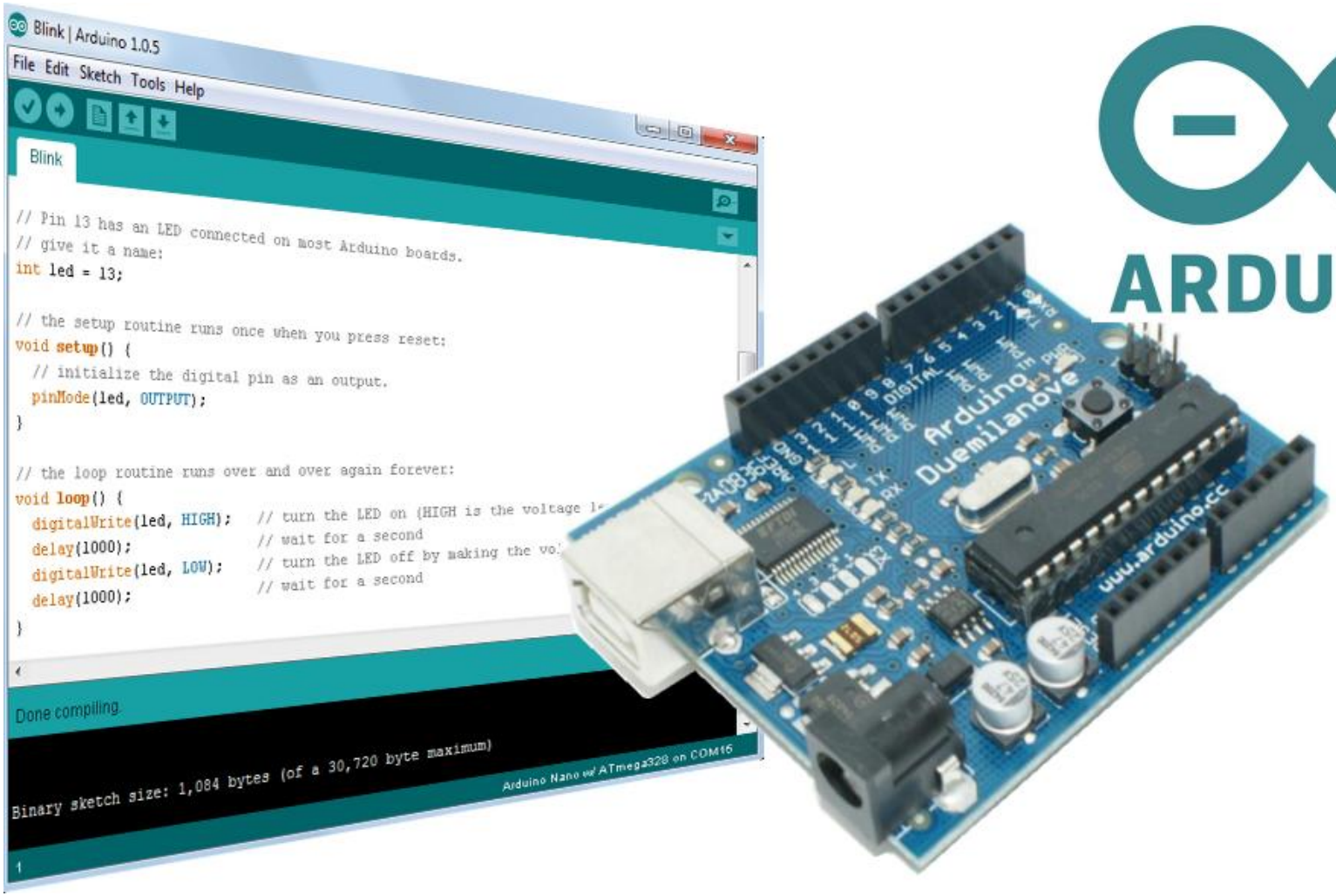


Bevezetés a mikrovezérlők programozásába:

Kommunikáció

Ajánlott irodalom

- Aduino LLC.: Arduino Language Reference
- ATMEL: ATmega328p mikrovezérlő adatlapja
- Brian W. Kernighan, Dennis Ritchie: A C programozási nyelv
- Cseh Róbert: Arduino programozási kézikönyv
- Ruzsinszki Gábor: Mikrovezérlős rendszerfejlesztés C/C++ nyelven I. – PIC mikrovezérlők
- Ruzsinszki Gábor: Mikrovezérlős rendszerfejlesztés C/C++ nyelven II. – Arduino

Lab 15 projektek

ASCIITable – az ASCII karaktertábla kiíratása

ReadASCIIString – RGB LED vezérlése számítógépről, soros kommunikációval

Adafruit_webclient – webkliens próba: letölt egy weblapot a megadott helyről

Ledblink – LED villogtatás az ESP8266EX mikrovezérlő segítségével.

ThingSpeak_DHT22 – IOT kliens, DHT22 szenzorral, a ThingSpeak szerverhez

A soros port kezelése

A mikrovezérlő soros porti kommunikációjának kezelése a **Serial** osztálykönyvtár használatával történhet. Ennek segítségével kommunikálhat a mikrovezérlő a számítógéppel, vagy más külső eszközzel. Ha a számítógéppel kommunikálunk, akkor az **Arduino** (vagy az **Energia**) fejlesztői környezet beépített monitorablakát használhatjuk az ellenoldali kommunikációra. Fontos, hogy ugyanarra a sebességre állítsuk be, mint amit a programunkban a `Serial.begin()` metódus hívásánál megadtunk.

Az általunk leggyakrabban használt függvények:

`begin()` – A soros port inicializálása, sebesség megadása pl: `Serial.begin(9600);`

`available()` – A soros portra beérkezett, még kiolvasatlan karakterek számát adja meg

`read()` – A soron következőt veszi elő a beérkezett karakterek tárolójából (-1, ha nincs ilyen)

`parseInt()` – Egy egész szám beolvasása (a bevezető nem szám/előjel karaktereket átlépi)

`print()` - Adatok kiíratása olvasható formában (a számokat karaktersorozattá alakítja)

`println()` – ugyanaz, mint a print(), csak egy sorvége jelzést hozzáfűz a kiíratás végén

`write()` – egy karakter kiírása. Példa: `Serial.write(65);` egy „A” betűt ír ki.

Megjegyzés: A többi függvény leírása itt található: [Arduino Reference - Serial]

Serial.print() részletesebben

A számokat karaktersorozattá alakítja, a karaktereket és karakterfüzéreket változatlanul hagyja. Például:

```
Serial.print(78)                ===> 78      (alapértelmezett: DEC)
Serial.print(1.23456)           ===> 1.23    (alapértelmezett: 2 tizedes)
Serial.print('N')               ===> N
Serial.print("Hello world.") ===> Hello world
```

Egész számok kiíratásánál opcionálisan a kiíratási számrendszer is megadható:

```
Serial.print(78,BIN)        ===> 1001110
Serial.print(78,OCT)        ===> 116    (64 + 8 + 6 = 78
Serial.print(78,DEC)        ===> 78
Serial.print(78,HEX)        ===> 4E     (0100 1110 = 0x4E)
```

Lebegőpontos számok (float) kiíratásánál opcionálisan a kiírandó tizedesjegyek száma is megadható:

```
Serial.print(1.23456,0)    ===> 1
Serial.print(1.23456,2)    ===> 1.23
Serial.print(1.23456,4)    ===> 1.2346 (utolsó jegy kerekítve…)
```

ASCIITable.ino

A kiíratást gyakoroljuk. Beépített gyári demo: **File/Examples/04. Communication/ASCIITable**

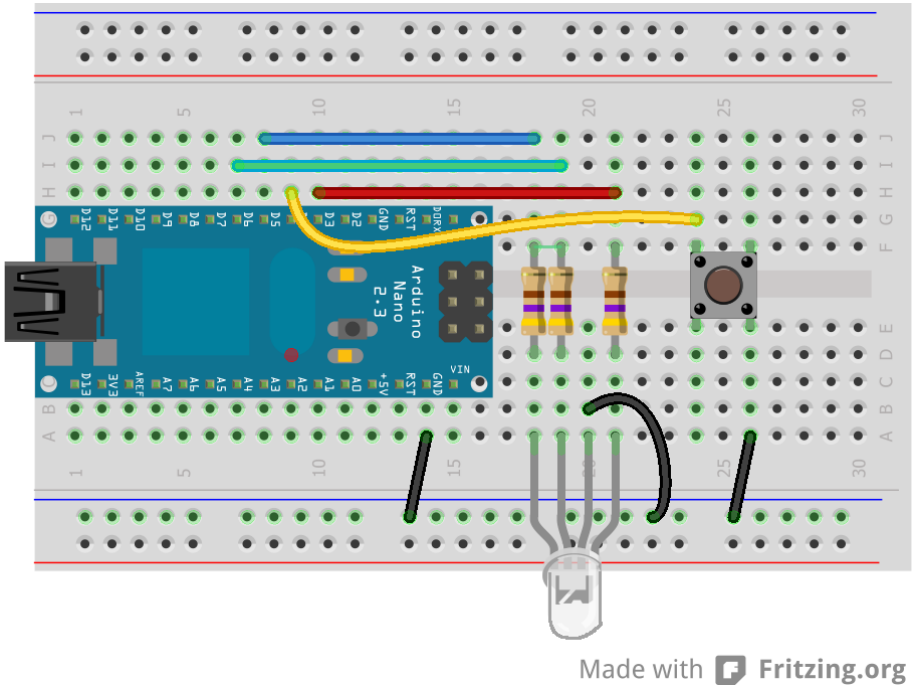
```
void setup() {
  Serial.begin(9600);
  Serial.println("ASCII Table ~ Character Map");
}

int thisByte = 33;              //Az első nyomtatható karakter ('!')

void loop() {
  Serial.write(thisByte);       //A karakter kiírása
  Serial.print(", dec: ");      //Karakterkód decimálisan
  Serial.print(thisByte, DEC);
  Serial.print(", hex: ");      //Karakterkód hexadecimálisan
  Serial.print(thisByte, HEX);
  Serial.print(", oct: ");      //Karakterkód oktálisan
  Serial.print(thisByte, OCT);
  Serial.print(", bin: ");      //Karakterkód binárisan
  Serial.println(thisByte, BIN);
  if(thisByte == 126) {         // Az utolsó nyomtatható karakter ('~')
    while(true) {
      continue;                 //Végtelen ciklusban RESET-re vár…
    }
  }
  thisByte++;
}
```

RGB LED vezérlése

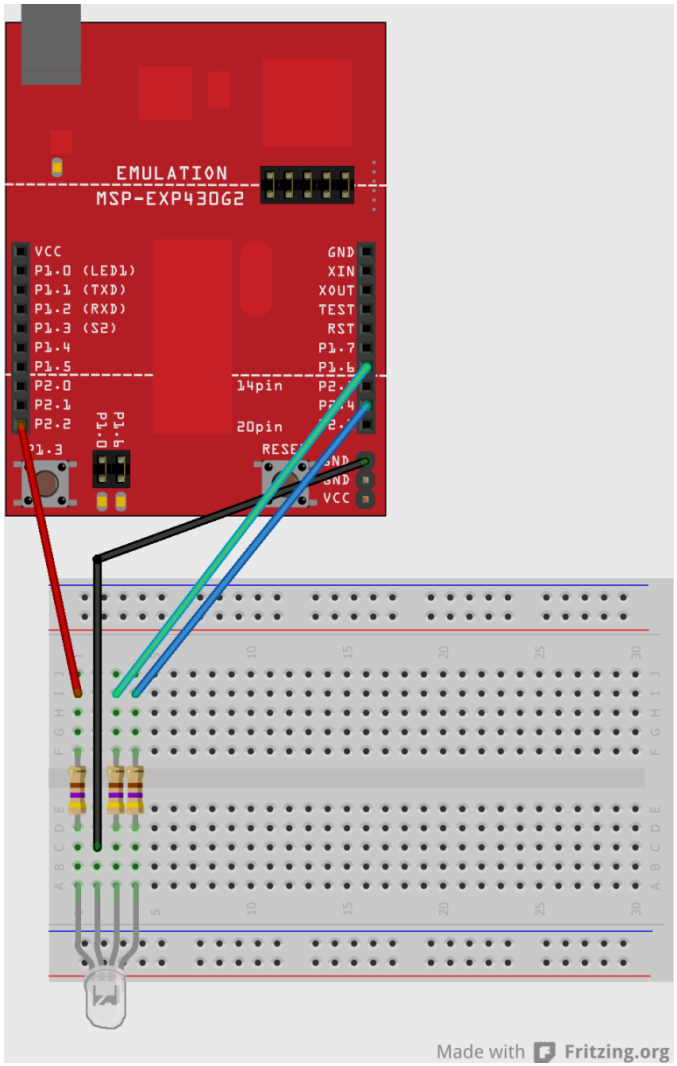
Vezéreljük egy RGB LED színbeállítását a PC-n beírt számhármassal!

Megjegyzés: A kapcsolás közös katódú RGB LED-et feltételez!

R LED = D3
G LED = D6
B LED = D5
PUSH2 = D4



R LED = P2_2
G LED = P1_6
B LED = P2_4
PUSH2 = P1_3



RGB LED vezérlése

ReadASCIIString.ino

A terminálablakban beírt három számmal vezéreljük az RGB színkomponenseket. Ehhez értelmeznünk kell a befutó karaktererek sorozatát (fel kell ismerni a többjegyű számokat). Erre a **`parseInt()`** függvényt fogjuk használni.

```
const int redPin = 3    //P2_2;
const int greenPin = 6 //P1_6;
const int bluePin = 5   //P2_4;
```

Hardverfüggő rész

```
void setup() {
  Serial.begin(9600);
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
```

```
void loop() {
  while (Serial.available() > 0) {
    int red = Serial.parseInt();
    int green = Serial.parseInt();
    int blue = Serial.parseInt();
    if (Serial.read() == '\n') {
      red = constrain(red, 0, 255);
      green = constrain(green, 0, 255);
      blue = constrain(blue, 0, 255);
      analogWrite(redPin, red);
      analogWrite(greenPin, green);
      analogWrite(bluePin, blue);
      Serial.print(red, HEX);
      Serial.print(green, HEX);
      Serial.println(blue, HEX);
    }
  }
}
```

Adatbevitel

Korlátozás

Vezérlés

Kiírás

ESP8266 WiFi modul

Az **ESP-01** modul egyszerű megoldást kínál WiFi kapcsolatok létesítésére. A mikrokontroller irányába soros kommunikációval kapcsolódik (TTL, soros). A könnyű kezelhetőséget a firmware által értelmezett egyszerű AT parancsok biztosítják.

- Tápfeszültség, I/O jelszint: 3.3V,
- Áramfelvétel: min. 300mA.
- CPU: Tensilica Xtensa LX3 32 bit 80 MHz
- SOC: Expressif ESP8266EX
- Frekvencia: 2.4 GHz sáv (802.11 b/g/n),
- Antenna: integrált,
- Adatsebesség: 9600 - 115200/8/n/1 (konfigurálható),
- Biztonság: WiFi (OPEN / WEP / WPA_PSK / WPA2_PSK / WPA_WPA2_PSK)
- Működési mód: Kliens / Access Point / mindkettő
- Kommunikáció: TCP és UDP
- SSL támogatás: nincs
- Egyidejű kapcsolatok száma: max. 5

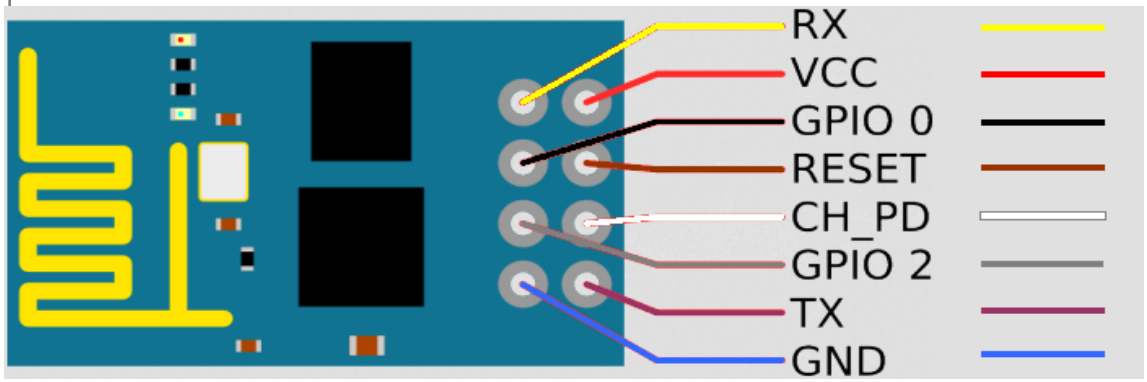
Link: [Az AT parancsok leírása]

Egyéb lehetőségek: A mikrovezérlőbe saját program is tölthető, így az AT parancsokkal történő vezérlés helyett alkalmazásprocesszorként is programozható. Elterjedt megoldások:

1. Lua interpreter betöltése és a modul Lua nyelvű programozása (NodeMCU)

2. Arduino 1.6 környezet kibővítése **ESP8266** támogatással, így közvetlenül is programozható.

A kivezetések szerepe

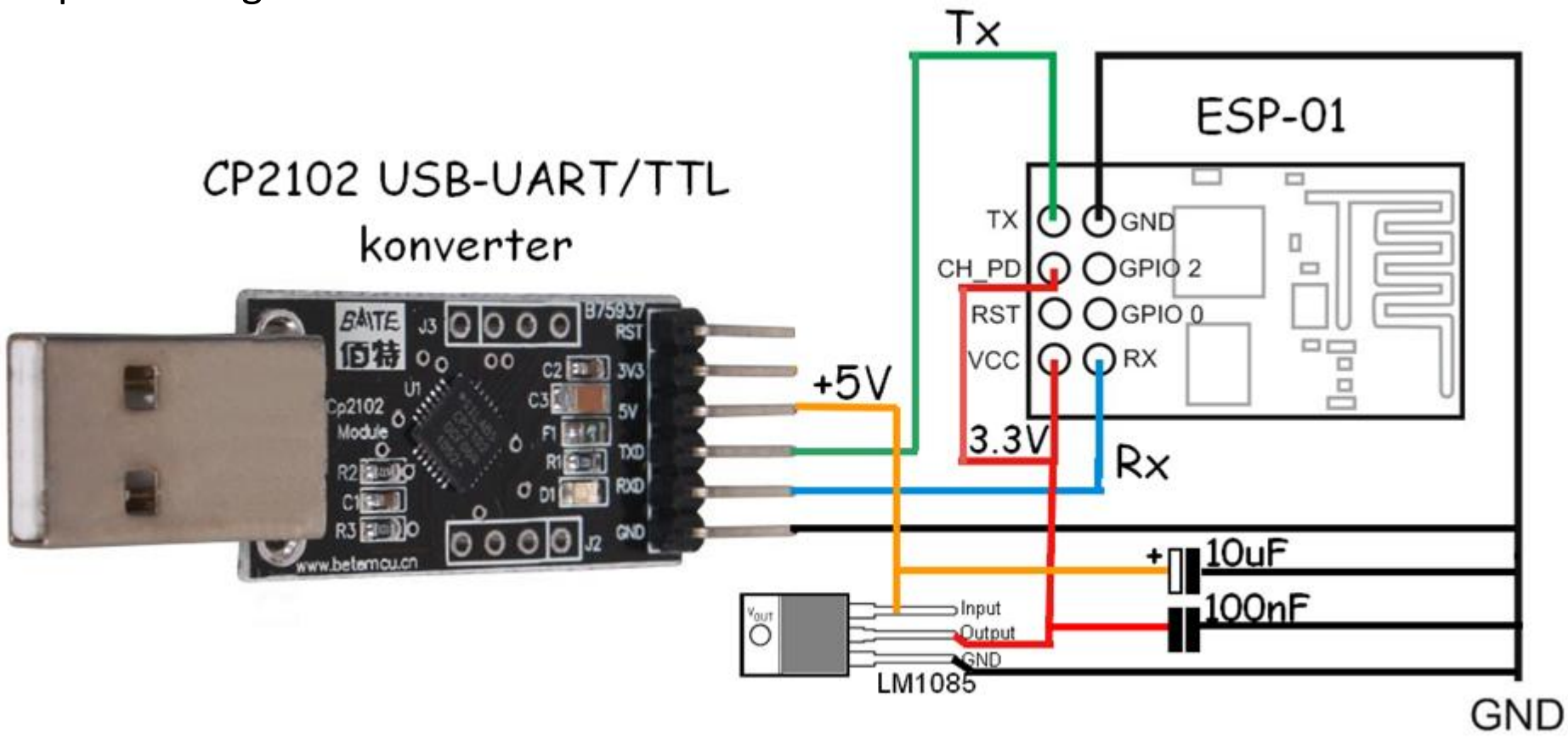


Felülnézeti rajz

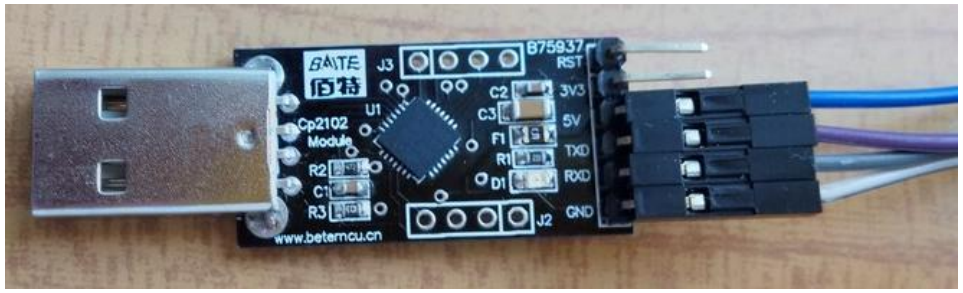
Kivezetés	Funkció
GND	A tápegység közös pontja
TX	UART (TTL) kimenet
GPIO 2	szabadon felhasználható I/O kivezetés
CH_PD	A modul működéséhez magas szintre kell húzni
GPIO 0	Ha induláskor LOW, akkor a bootloader indul
RESET	Lehúzásra újraindítja a mikrovezérlőt
RX	UART (TTL) bemenet
VCC	Tápfeszültség (3,3 V)

Teszt kapcsolás

A modult egy 3,3 V kompatibilis **USB-TTL átalakítóval** kapcsolhatjuk össze a PC-vel. Sajnos, a konverter nem biztosítja az **ESP-01** áramigényét, ezért az USB busz 5 V-os tápellátásából (ami 0,5 A terhelhetőségű) kell előállítani a 3,3 V-os tápfeszültséget.

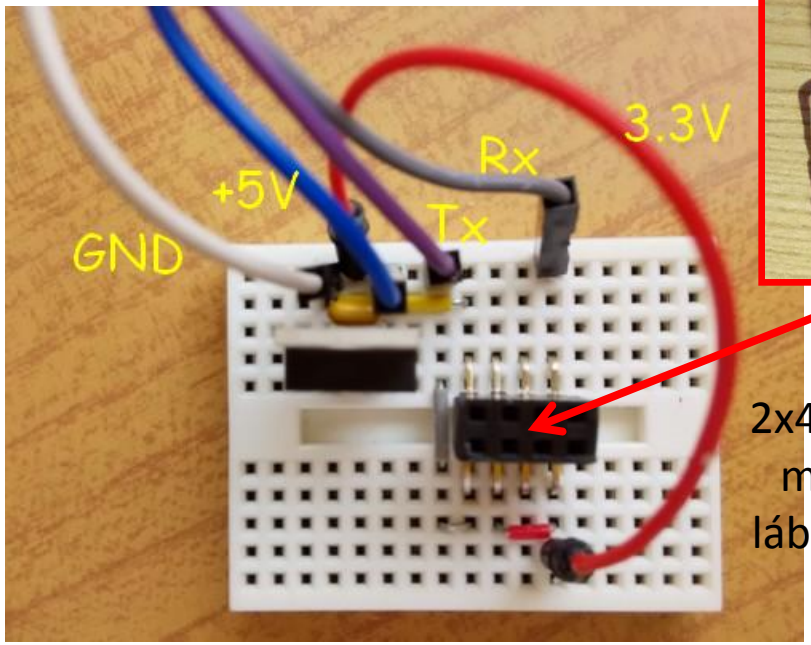


Teszt kapcsolás

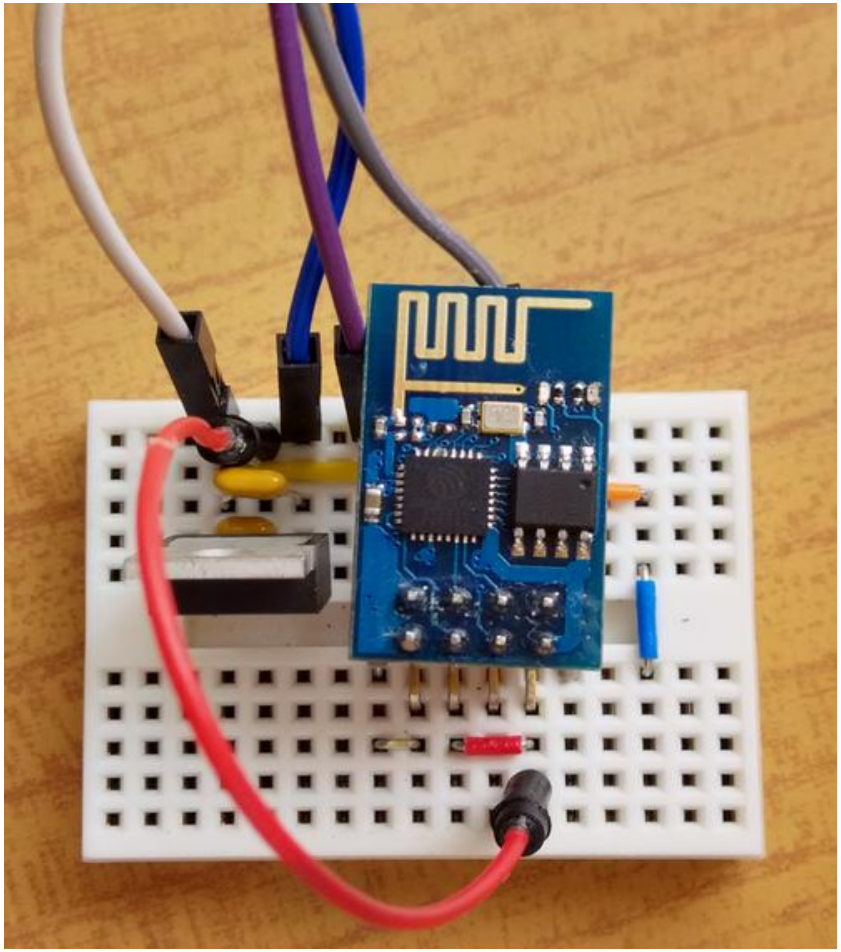


CP2102 USB-TTL átalakító

A huzalozás felülnézetben

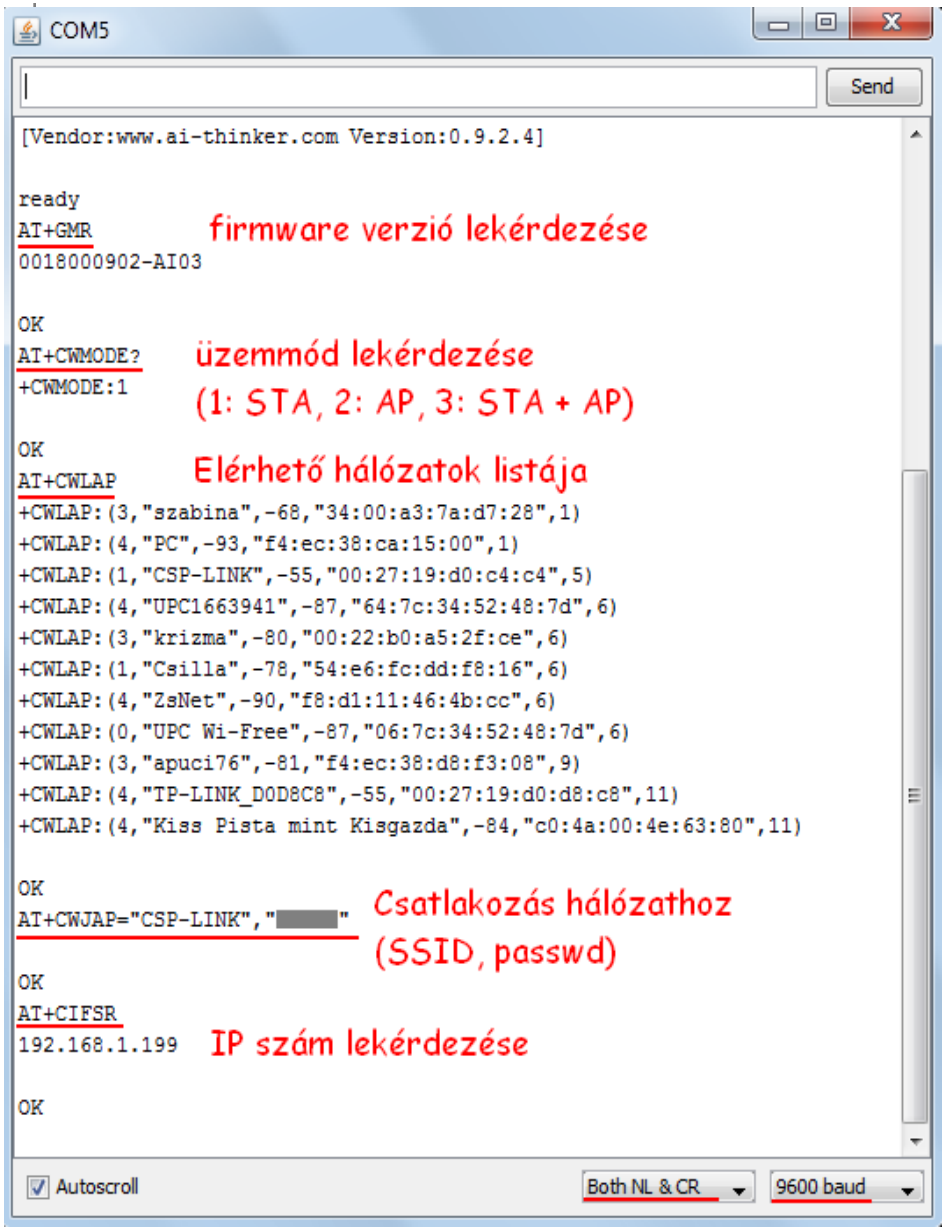


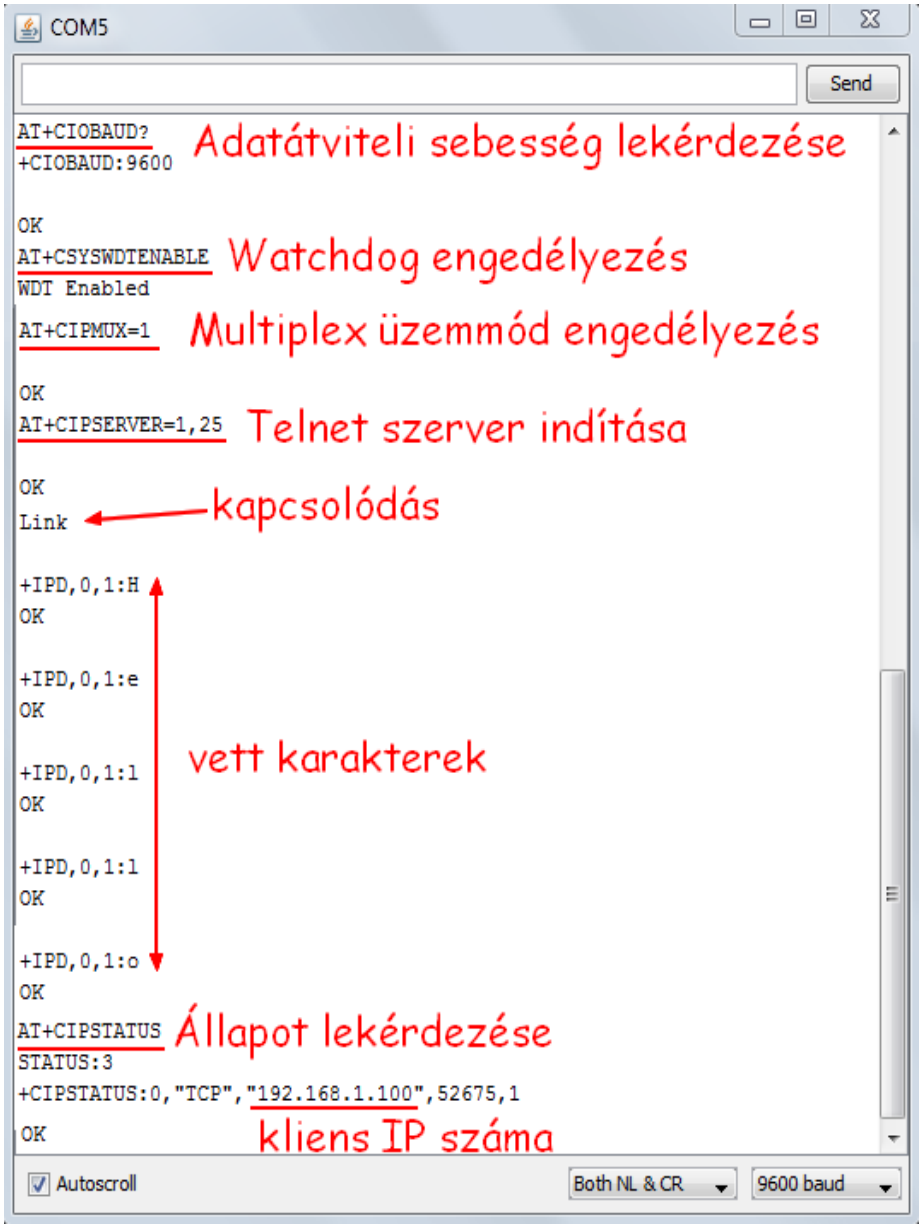
2x4 kivezetéses meghajlított lábú csatlakozó



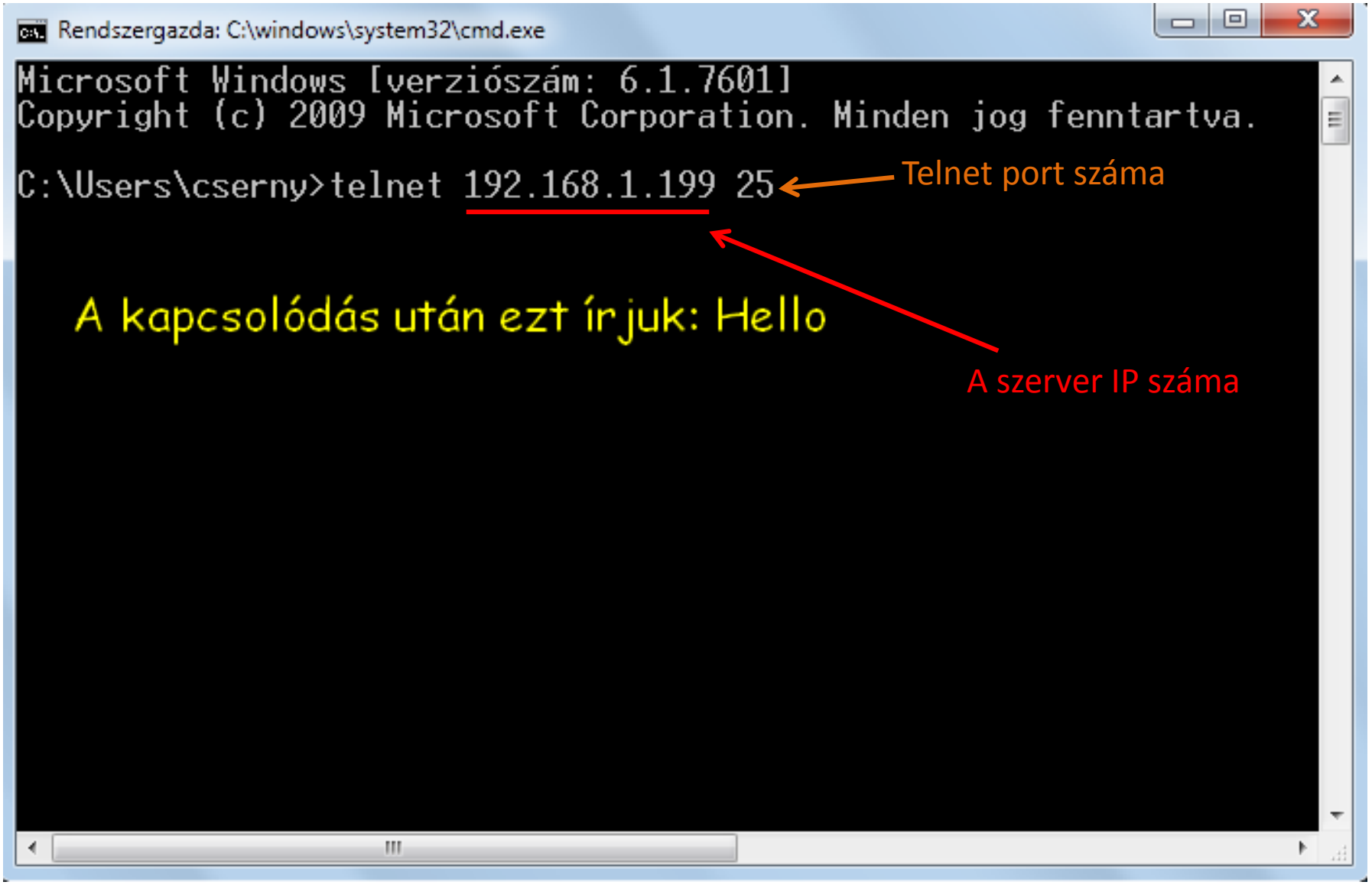
A kész kapcsolás (DHT22-höz előkészítve)

1. Az AT parancsok használata



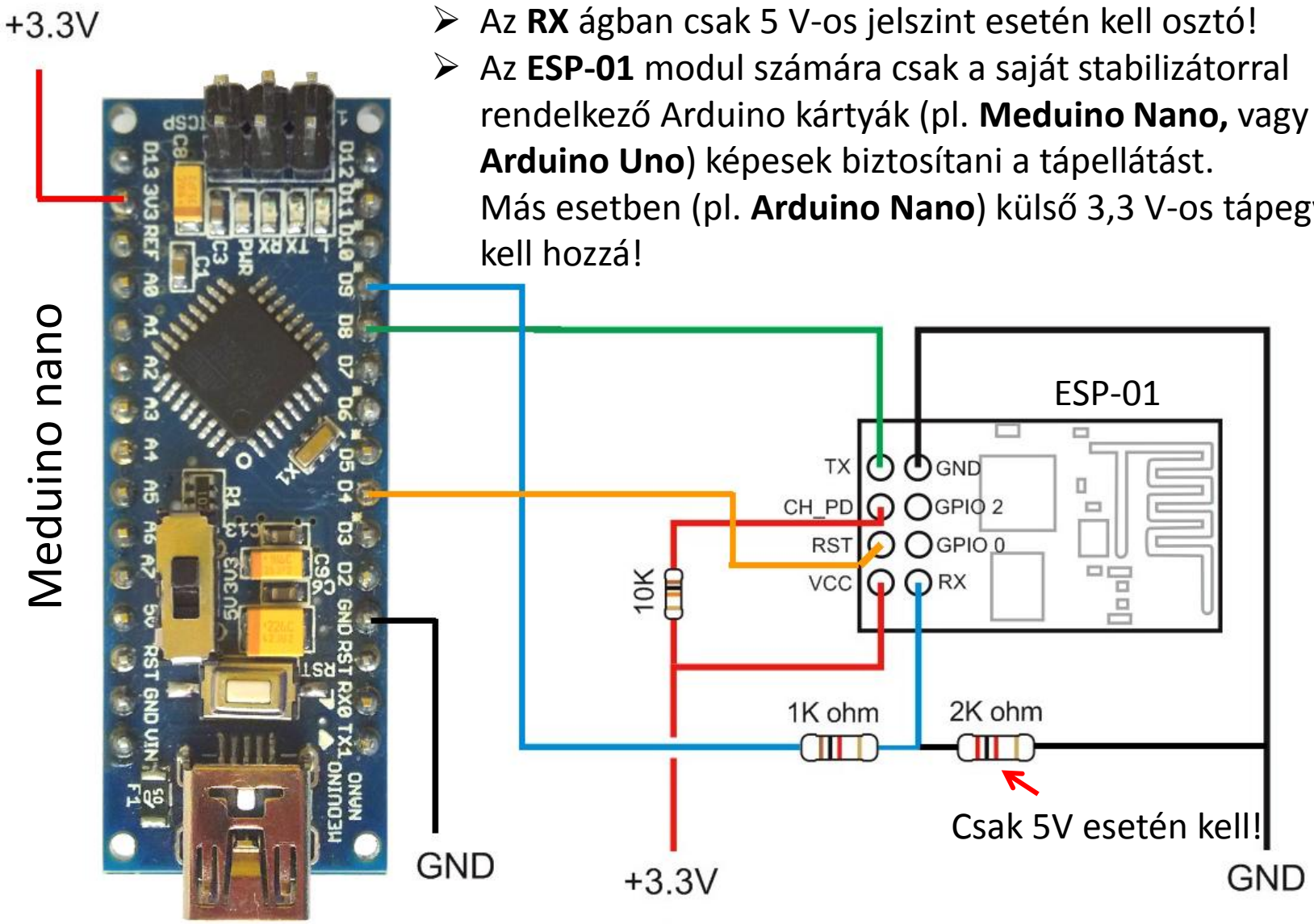


Windows Telnet kliens indítása



II. Vezérlés Arduino-val

- Az **RX** ágban csak 5 V-os jelszint esetén kell osztó!
- Az **ESP-01** modul számára csak a saját stabilizátorral rendelkező Arduino kártyák (pl. **Meduino Nano,** vagy **Arduino Uno**) képesek biztosítani a tápellátást.
  Más esetben (pl. **Arduino Nano**) külső 3,3 V-os tápegység kell hozzá!



Vezérlés Arduino-val

A 2x4 kivezetéses meghajlított lábú csatlakozó breadboard-képessé teszi az ESP-01 modult…

Adafruit_webclient.ino

Az alábbi program megpróbál a www.adafruit.com webszerveréhez kapcsolódni és letölteni a /testwifi/index.html oldalt. A mintaprogam az Adafruit_ESP8266.h programkönyvtárat használja (Link: github.com/adafruit/Adafruit_ESP8266).
Ez a program csak **Arduino környezetben** fordul le!

1. rész: Beállítási paraméterek megadása

```
#include <Adafruit_ESP8266.h>
#include <SoftwareSerial.h>
#define ESP_RX   8                          //connect to ESP-01 TX pin
#define ESP_TX   9                          //connect to ESP-01 RX pin
#define ESP_RST  4                          //connect to ESP-01 RST pin
SoftwareSerial softser(ESP_RX, ESP_TX);
// Must declare output stream before Adafruit_ESP8266 constructor; can be
// a SoftwareSerial stream, or Serial/Serial1/etc. for UART.
Adafruit_ESP8266 wifi(&softser, &Serial, ESP_RST);
// Must call begin() on the stream(s) before using Adafruit_ESP8266 object.
#define ESP_SSID "SSID"                     // Your network name here
#define ESP_PASS "password"                 // Your network password here
#define HOST     "www.adafruit.com"         // Host to contact
#define PAGE     "/testwifi/index.html"     // Web page to request
#define PORT     80                         // 80 = HTTP default port
#define LED_PIN  13
```

Otthoni WiFi hálózat belépési adatai (név, jelszó)

2. rész: a WiFi modul újraindításának és a firmware verziójának ellenőrzése

```
void setup() {
  char buffer[50];
  pinMode(LED_PIN, OUTPUT);
  for(uint8_t i=0; i<3; i++) {
    digitalWrite(13, HIGH); delay(50);
    digitalWrite(13, LOW);  delay(100);
  }
  wifi.setBootMarker(F("Version:0.9.2.4]\r\n\r\nready"));
  softser.begin(9600); // Soft serial connection to ESP8266
  Serial.begin(115200); UART serial debug
  if(!wifi.hardReset()) {
    Serial.println(F("no response from module."));
    for(;;);
  }
  if(!wifi.softReset()) {
    Serial.println(F("no response from module."));
    for(;;);
  }
  wifi.println(F("AT+GMR"));
  if(wifi.readLine(buffer, sizeof(buffer))) {
    Serial.println(buffer);
    wifi.find();        // Discard the 'OK' that follows
  } else {
    Serial.println(F("error"));
  }
```

3. rész: Kapcsolódás a hálózatra és a kért oldal letöltése

```
if(wifi.connectToAP(F(ESP_SSID), F(ESP_PASS))) {
  wifi.println(F("AT+CIFSR"));
  if(wifi.readLine(buffer, sizeof(buffer))) {
    Serial.println(buffer);
    wifi.find(); // Discard the 'OK' that follows
    if(wifi.connectTCP(F(HOST), PORT)) {
      Serial.print(F("OK\nRequesting page..."));
      if(wifi.requestURL(F(PAGE))) {
        Serial.println("OK\nSearching for string...");
        if(wifi.find(F("working"), true)) {
          Serial.println(F("found!"));
        } else {
          Serial.println(F("not found."));
        }
      } else { // URL request failed
        Serial.println(F("error"));
      }
      wifi.closeTCP();
    } else { // TCP connect failed
      Serial.println(F("D'oh!"));
    }
  } else { // IP addr check failed
    Serial.println(F("error"));
  }
  wifi.closeAP();
} else { // WiFi connection failed
  Serial.println(F("FAIL"));   }}
```

Kapcsolódás a WiFi-re

IP szám lekérdezése

Kapcsolódás a szerverhez

Weblap lekérése

Megtalálta,
Vagy nem?

Szerverkapcsolat bontása

WiFi kapcsolat bontása

```
Adafruit ESP8266 Demo
Hard reset...
[Vendor:www.ai-thinker.com Version:0.9.2.4]
ready'
OK.
Soft reset...---> AT+RST
<--- '
AT+RST
OK
Checking firmware version...---> AT+GMR
<--- '0018000902-AI03'
0018000902-AI03

OK
'
Connecting to WiFi...---> AT+CWMODE=1
<--- 'no change'
---> AT+CWJAP="CSP-LINK", "xxxxx"
<--- '
OK
'
---> AT+CIPMUX=0
<--- '
OK
'
OK
Checking IP addr...---> AT+CIFSR
<--- '192.168.1.199'
192.168.1.199

OK
'
```

```
Connecting to host...--->
AT+CIPSTART="TCP","www.adafruit.com",80
<--- '
OK
Linked'
OK
Requesting page...---> AT+CIPSEND=61
<--- '> '
---> GET /testwifi/index.html HTTP/1.1
Host: www.adafruit.com

<--- 'SEND OK,
Searching for string...
+IPD,'
536:HTTP/1.1 200 OK
Date: Tue, 28 Apr 2015 15:14:31 GMT
Server: Apache
Access-Control-Allow-Headers: Origin...
Content-Length: 74
Content-Type: text/html

This is a test of the CC3000 module!
If you can read this, its working'
found!
---> AT+CIPCLOSE
<--- ' :)
OK
OK
Unlink
'
---> AT+CWQAP
<--- ,
OK'
```

III. Arduino – ESP8266 támogatással

Egy **ESP8266** támogatással bővített **Arduino 1.6** fejlesztői környezet található a github.com/esp8266/Arduino címen. Ennek segítségével az **ESP8266EX** mikrovezérlő közvetlenül és kényelmesen (**Arduino/Wiring** nyelven) programozható.

Beállítások:

Telepítés után a kártya kiválasztásához a **Tools/Board** menüben a **„Generic ESP8266 board”** opciót kell választani. A soros port kiválasztása előtt csatlakoztatni kell a kommunikációhoz használt **USB-UART/TTL** átalakítót.

Programletöltés:

- A **Tools/Programme**r menüben az **„esptool”** eszközt kell kiválasztani
- Programletöltés előtt a modult bootloader módba kell állítani: az **ESP-01** modul **GPIO 0** lábát földre kell húzni és így kell resetelni (RST láb lehúzása, vagy tápfeszültség pillanatnyi megszakítása). Ekkor a modul bootloader módba kapcsol.

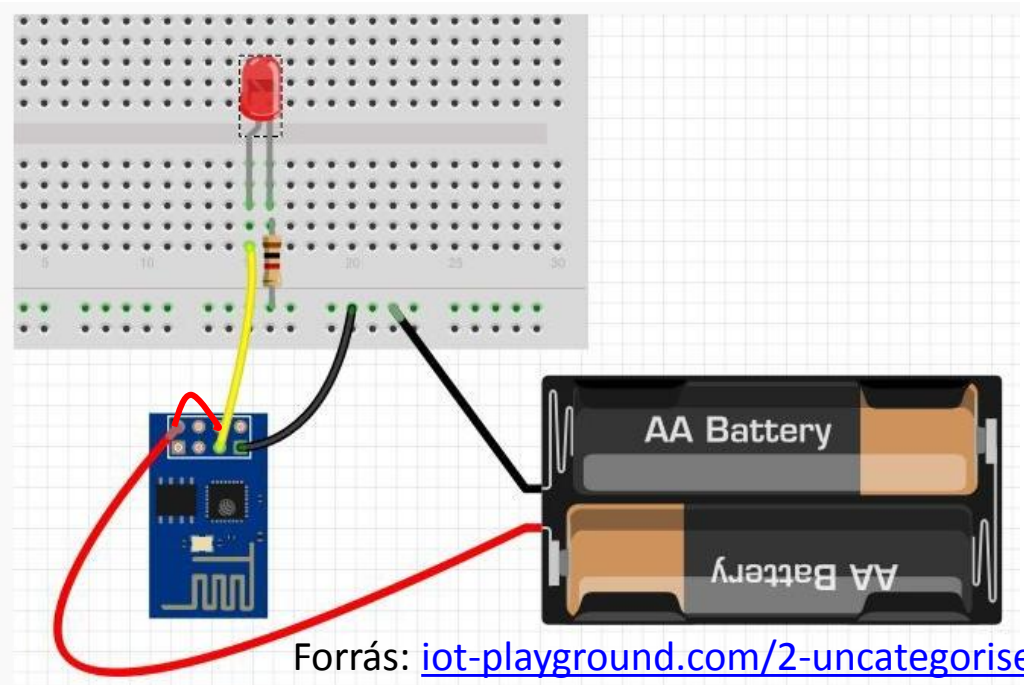
Kapcsolás:

A kapcsolás lehet ugyanaz, amit az **AT parancsok** kipróbálásánál használtunk, illetve az igényeinknek megfelelően bővíthetjük a kapcsolást. Sajnos, az **ESP-01** modulnál csak a **GPIO 2** láb használható szabadon.

Ledblink.ino

```
void setup() {
  pinMode(2, OUTPUT);       // Csak GPIO 2 használható!
}

void loop() {
  digitalWrite(2, HIGH);    // bekapcsoljuk a LED-et (kimenet magas szintre)
  delay(1000);              // várunk egy másodpercet
  digitalWrite(2, LOW);     // kikapcsoljuk a LED-et (kimenet alacsony szintre)
  delay(1000);              // várunk egy másodpercet
}
```

A szokásos LED villogtató program, ESP-01 kártyán futtatva.

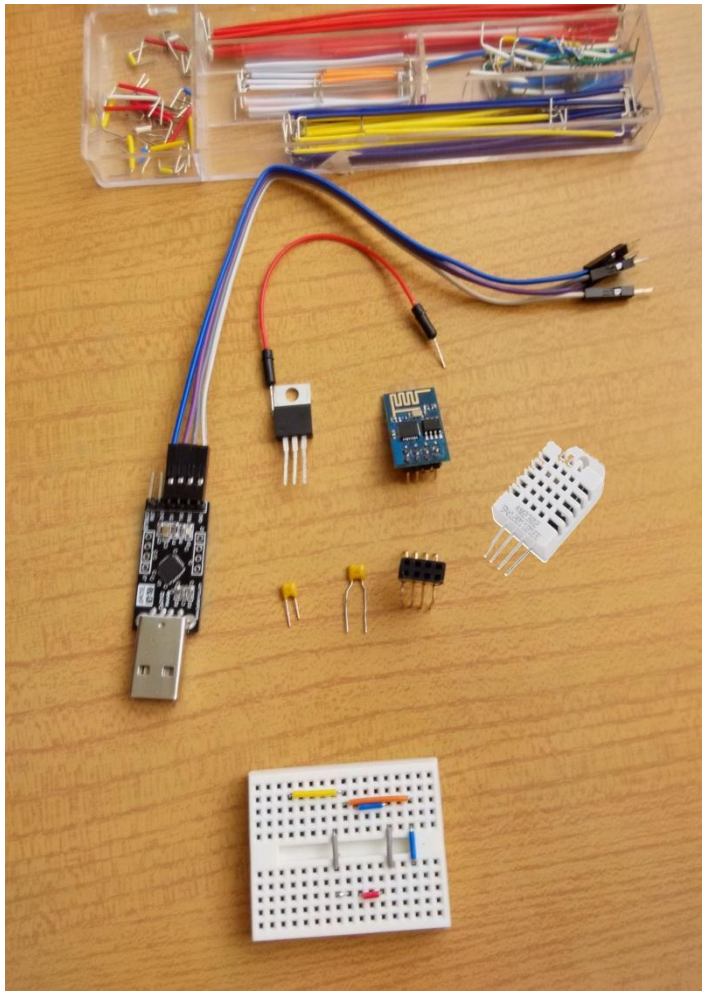


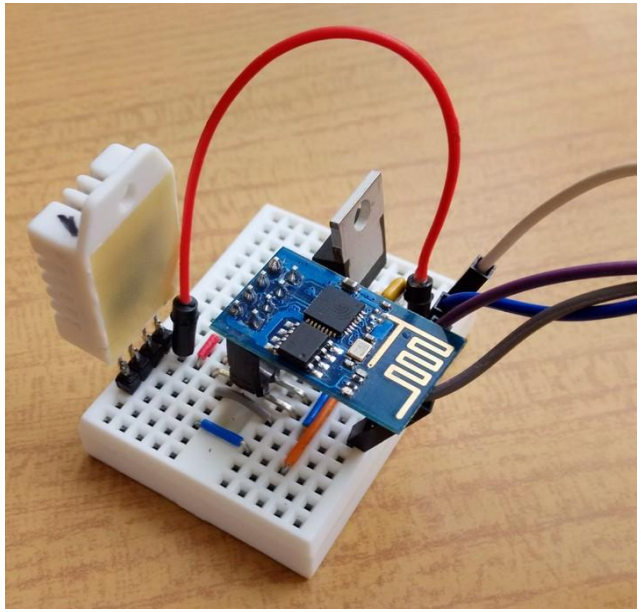
Forrás: iot-playground.com/2-uncategorised/38-esp8266-and-arduino-ide-blink-example

ThingSpeak kliens DHT22 szenzorral

Cél: Hőmérséklet és relatív páratartalom mérése DHT22 szenzorral, s az adatok továbbítása az **api.thingspeak.com** szerverre, ahol az adatok automatikusan megjelenítésre kerülnek.

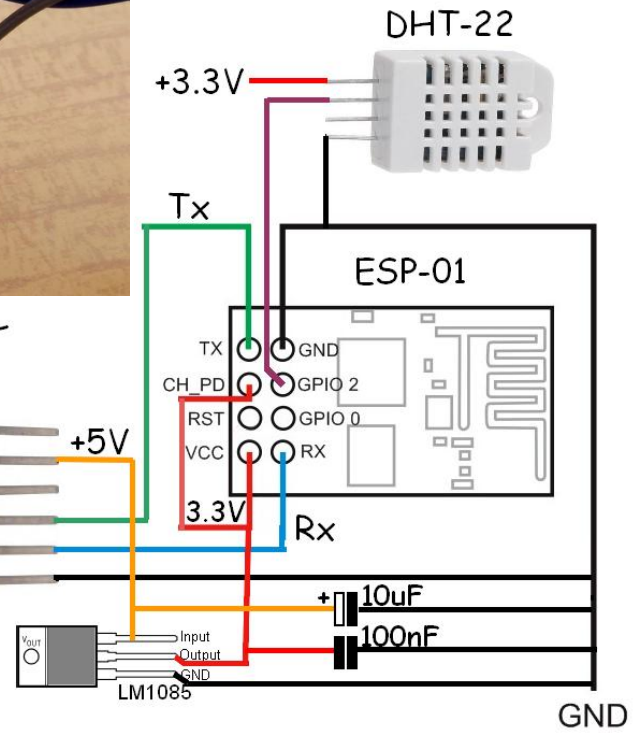
A hozzávalók:





CP2102 USB-UART/TTL konverter

A kapcsolás



ThingSpeak_DHT22.ino

Használnunk kell a beépített **ESP8266WiFi** könyvtárat és az **Adafruit DHT22** könyvtárat.

1. rész: paraméterek definiálása, objektumok példányosítása

```
#include <ESP8266WiFi.h>
const char* ssid     = "your-SSID";
const char* password = "your-passwd";
const char* host = "api.thingspeak.com";
const char* writeAPIkey = "your-API KEY";

#include "DHT.h"
#define DHTPIN 2                    //Digital input pin for the sensor
#define DHTCOUNTS 15                //Arduino=5, MSP430=11, ESP8266=15
DHT dht(DHTPIN, DHT22,DHTCOUNTS);

String T = String("20.0");          //Default values
String H = String("25.0");
```

ThingSpeak_DHT22.ino

2. rész: kapcsolódás a hálózatra, IP szám lekérdezése

```
void setup() {
  Serial.begin(115200);
  dht.begin();                              //DHT szenzor inicializálás
  delay(100);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);               //Kapcsolódás a WiFi hálózatra
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");                      //Várakozás a kapcsolódásra…
  }
  Serial.println("");
  Serial.println("WiFi connected");         //Kapcsolódás megtörtént
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());           //IP szám lekérdezése
}
```

3. rész: kapcsolódás a hálózatra, IP szám lekérdezése

```
void loop() {
  delay(5000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  // check if returns are valid, if they are NaN (not a number)
  // then something went wrong!
  if (isnan(t) || isnan(h)) {
    Serial.println("Failed to read from DHT");
  }
  else {
    T = String(t);              //Convert temperature to String
    H = String(h);              //Convert humidity to String
  }
  Serial.print("connecting to ");
  Serial.println(host);
```

Folytatás a következő oldalon...

4. rész: kapcsolódás a szerverhez, adatküldés és a válasz kilistázása

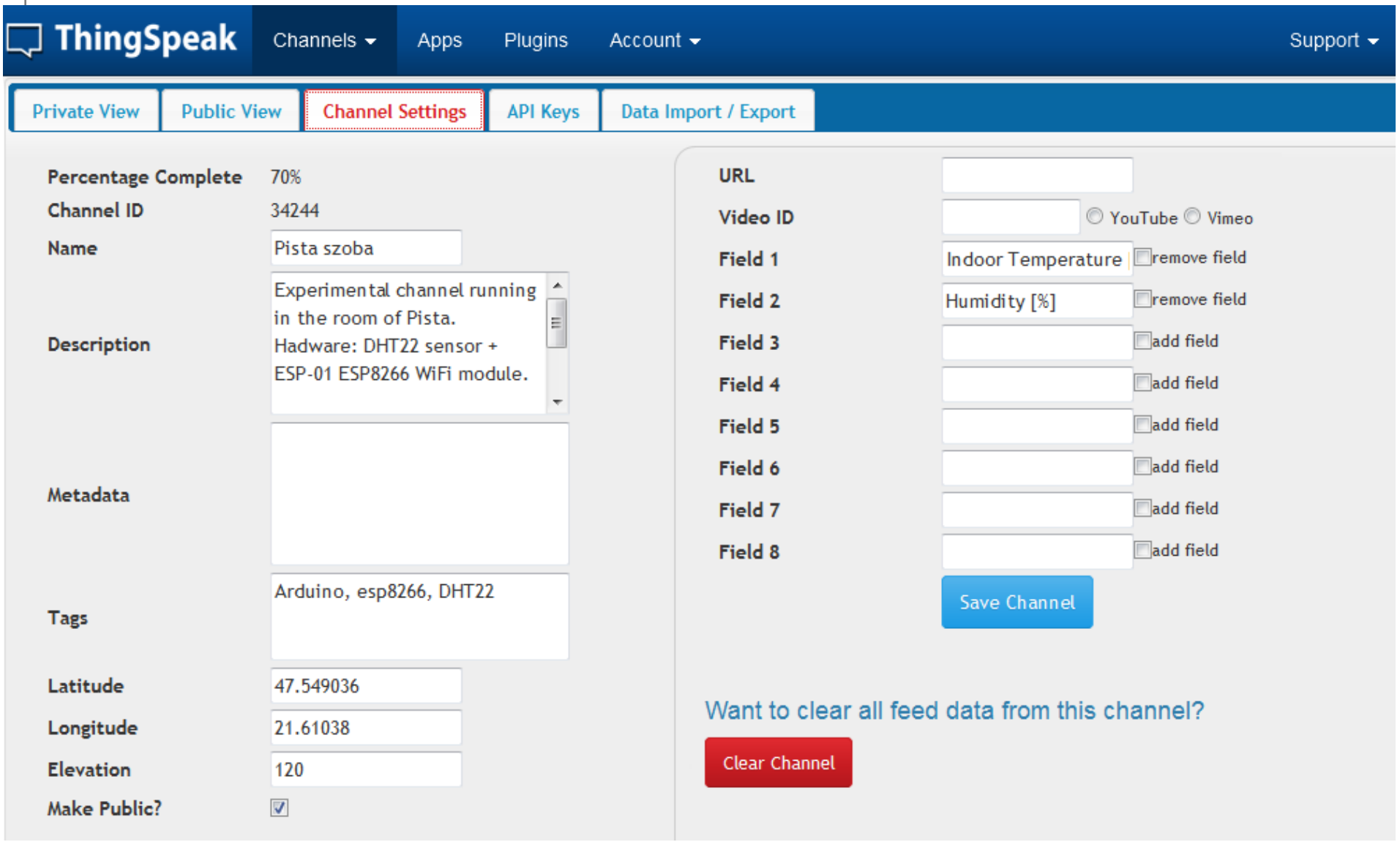
```
  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }
  String url = "/update?key=" + String(writeAPIkey);
  url += "&field1=" + T;
  url += "&field2=" + H;
  Serial.print("Requesting URL: ");  Serial.println(url);
  client.print(String("GET ") + url + "\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  delay(10);
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
  Serial.println();
  Serial.println("closing connection");
}
```

TCP kapcsolódás

Adatok kiküldése

Válasz üzenet vétele

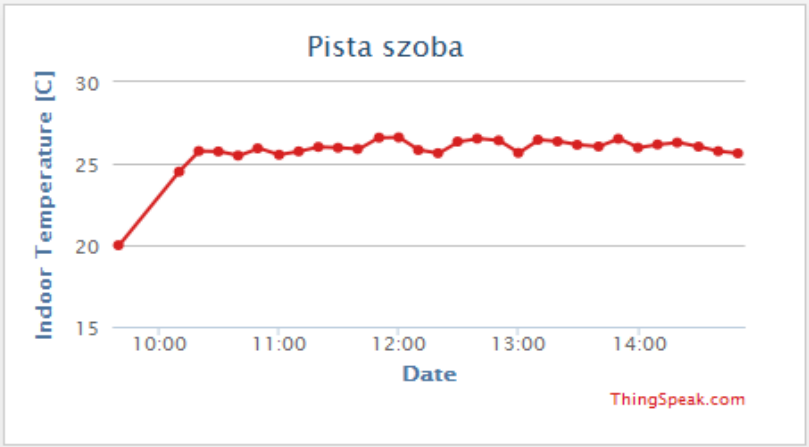
Csatorna definiálása



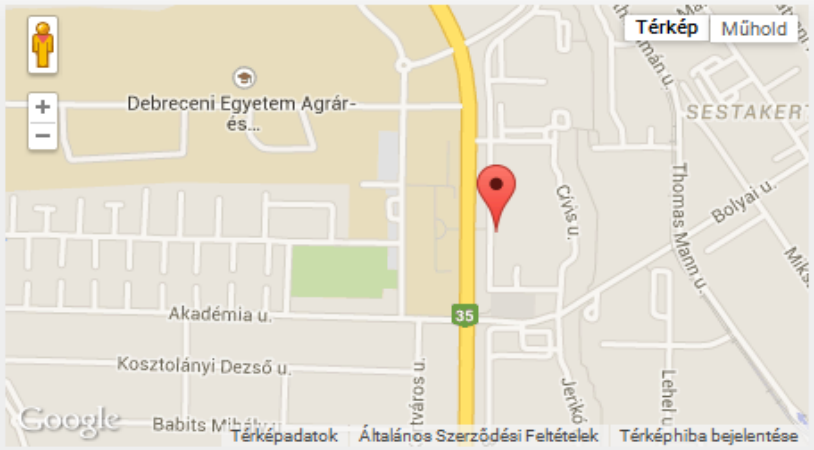
Az eredmény

ThingSpeak | Channels | Apps | Support | Blog | Sign In

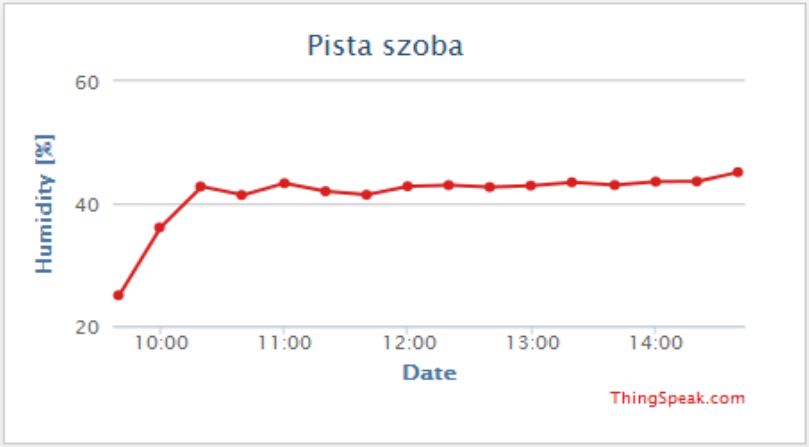
Field 1 Chart



Channel Location

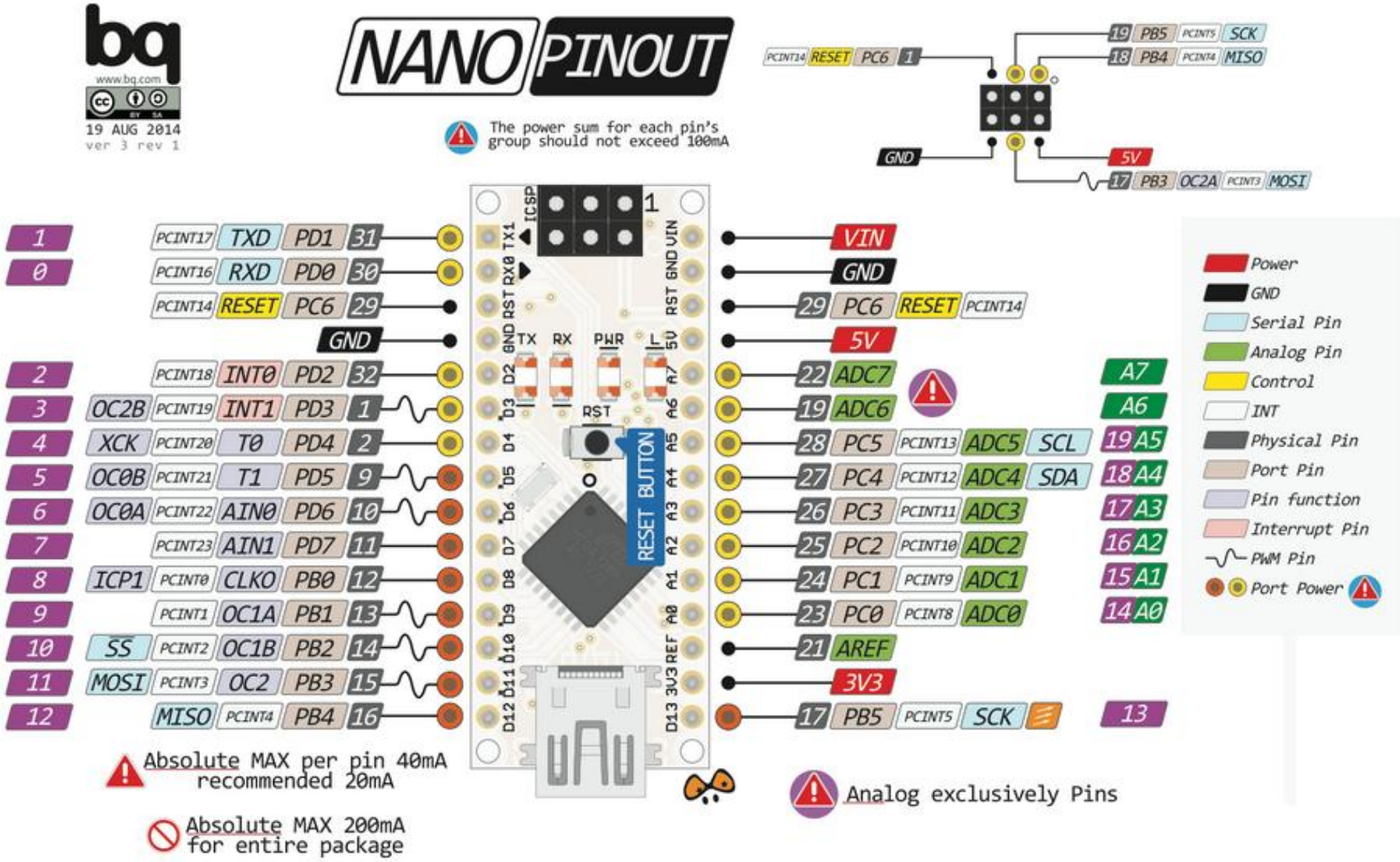


Field 2 Chart



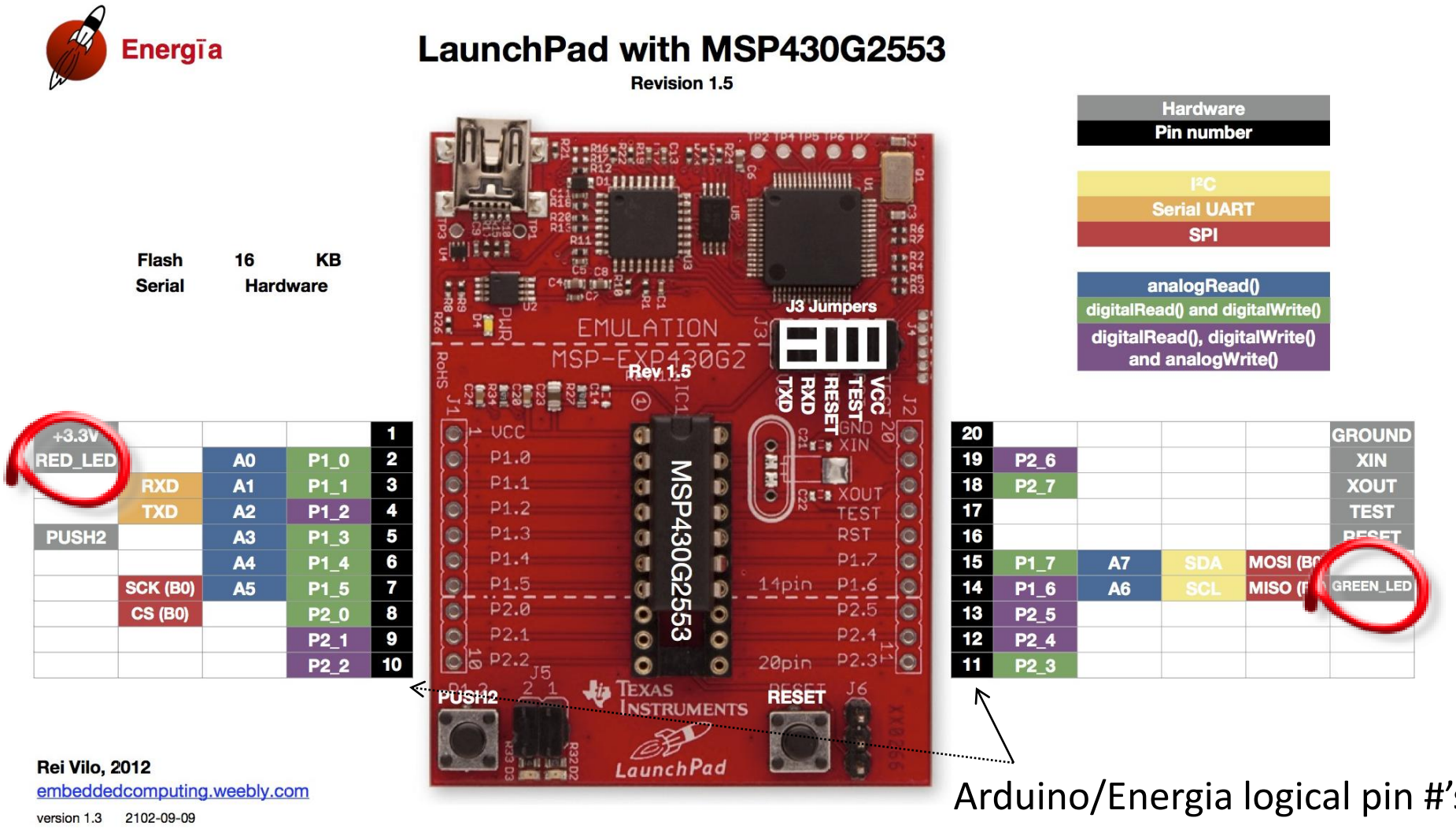
https://thingspeak.com/channels/34244

Emlékeztető: Arduino nano v3.0



MSP430 Launchpad : Energia Pinout

http://github.com/energia/Energia/wiki/Hardware



Rei Vilo, 2012
embeddedcomputing.weebly.com
version 1.3 2102-09-09

Arduino/Energia logical pin #'s