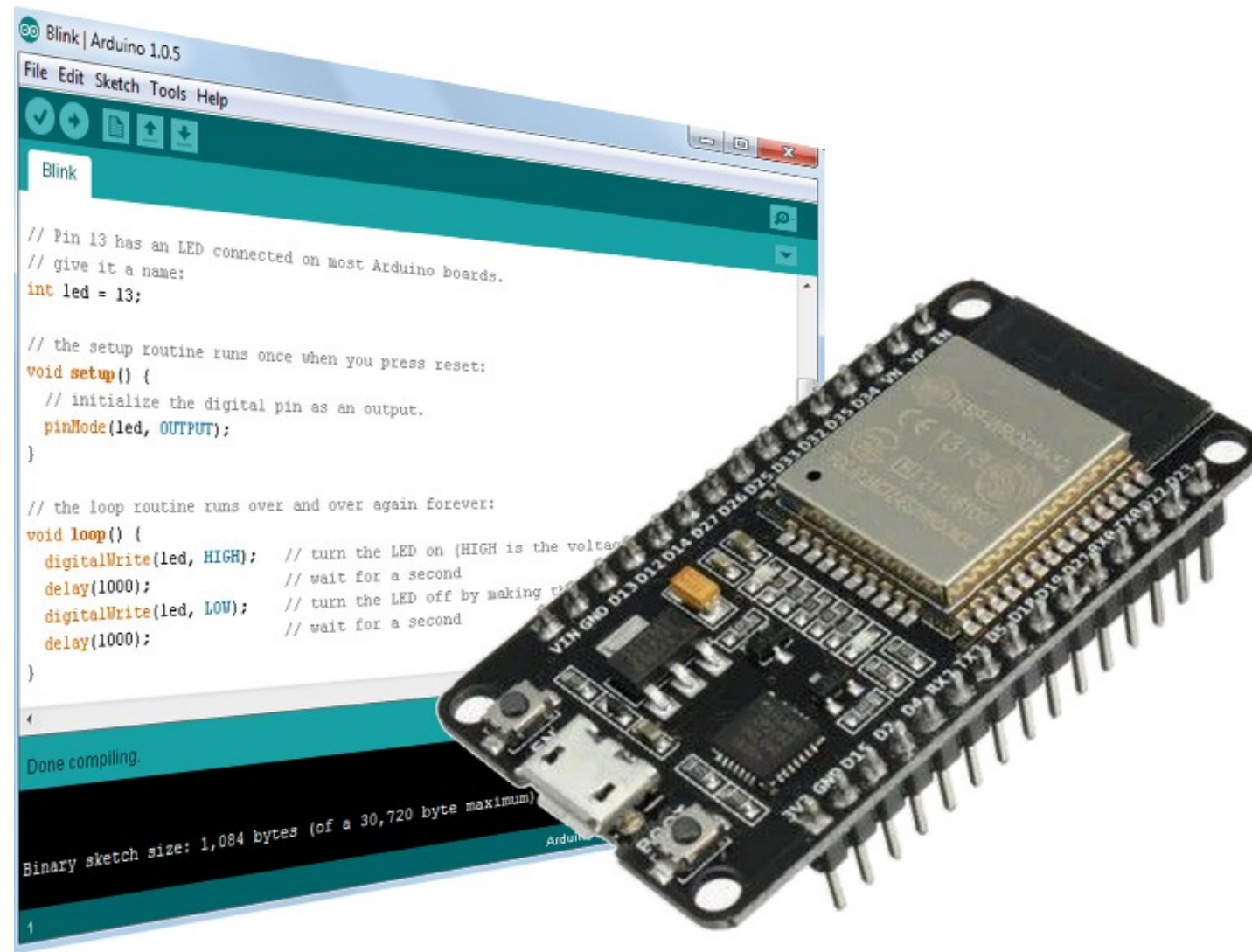


# ESP32 mikrovezérlők programozása Arduino környezetben



## 1. Ismerkedés az ESP32 kártyával és az Arduino környezettel

# Felhasznált és ajánlott irodalom

---

- Expressif: [ESP32 datasheet](#)
- Rui Santos & Sara Santos: [Random nerd tutorials](#)
- Rui Santos & Sara Santos: [ESP32 Web Server with Arduino IDE](#)
- Ravi Teja: [Getting Started with ESP32](#)
- Ravi Teja: [ESP32 Pinout](#)
- Microcontrollerslab: [ESP32 ADC with Arduino IDE](#)
- Circuits4you: [ESP32 DAC example](#)
- Brian W. Kernighan – Dennis M. Ritchie: [A C programozási nyelv](#)

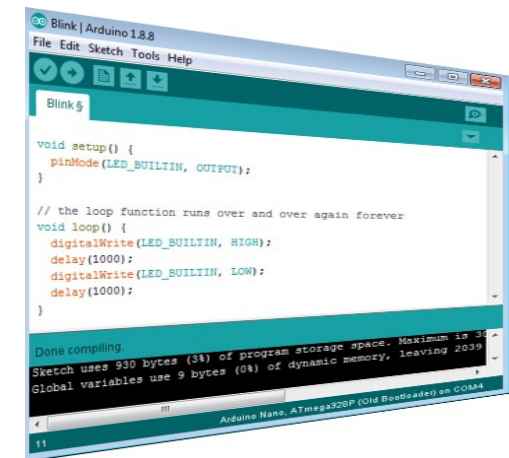
# Mi az Arduino?

- Az **Arduino** egy szabad szoftveres, nyílt forráskódú elektronikai fejlesztőplatform, vagy ökoszisztéma az elektronikus eszközök könnyen elsajátítható kezeléséhez

- ❖ **Arduino IDE** (integrált fejlesztői környezet): Java alapú, keresztplatformos fejlesztői környezet (szerkesztő, fordító, programletöltő stb.)
- ❖ **Arduino kártya**: eredetileg **ATmega** mikrovezérlőn alapuló hardver, amely önállóan vagy a számítógéppel összekapcsolva is működhet, de a támogatott kártyák száma rohamosan bővül, és a keretrendszer bővítőcsomagok telepítésével könnyen kiegészíthető

Ebben az évben a **DOIT ESP32 Devkit-1** kártyára mutatunk be példaprogramokat

- ❖ **Arduino programnyelv és programkönyvtár-gyűjtemény**: amely lehetővé teszi, hogy a mikrovezérlő részleteinek pontos ismerete nélkül, egyszerűen írassunk programot



```
void setup() {  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH);  
  delay(1000);  
  digitalWrite(LED_BUILTIN, LOW);  
  delay(1000);  
}
```

# Az Arduino IDE telepítése

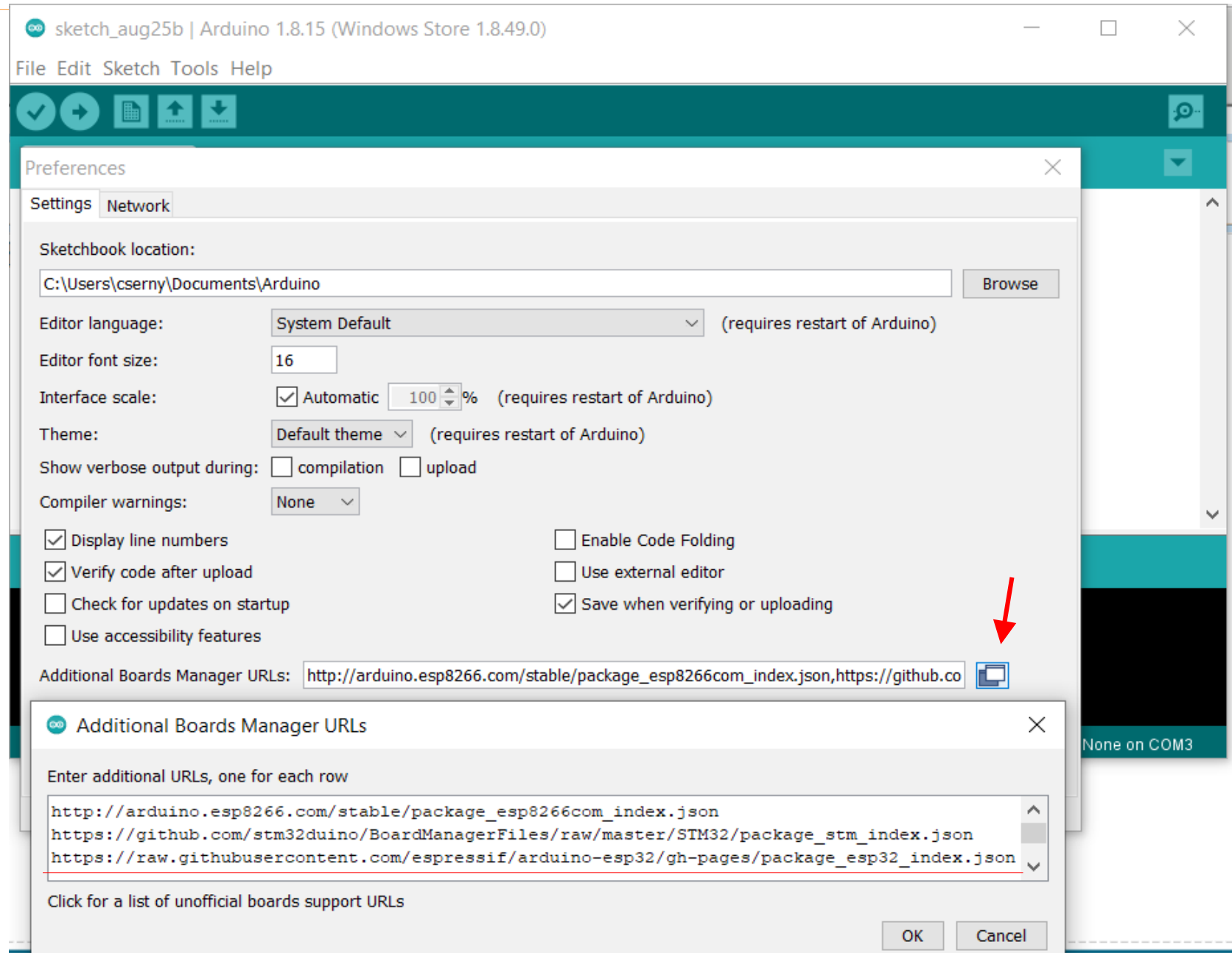
- **Windows 10** esetén a **Microsoft Store**-ban is rendelkezésre áll az **Arduino IDE**
- Más esetben az **arduino.cc/en/Main/Software** oldalról töltsük le a legfrissebb **Arduino** kiadást – „**JUST download**” (én általában a ZIP változatot töltöm le)

The screenshot shows the Arduino IDE 1.8.15 download page. On the left, there is a teal square icon with a white infinity symbol containing a minus and a plus sign. To its right, the text 'Arduino IDE 1.8.15' is displayed. Below this, a paragraph describes the software as open-source and easy to use with any Arduino board. A link to the 'Getting Started' page is provided. On the right side, a teal box titled 'DOWNLOAD OPTIONS' lists various download links: 'Windows Win 7 and newer', 'Windows ZIP file' (underlined in red), 'Windows app Win 8.1 or 10' with a 'Get' button, and four Linux options: 'Linux 32 bits', 'Linux 64 bits', 'Linux ARM 32 bits', and 'Linux ARM 64 bits'.

- A letöltés és telepítés után a kártyához való meghajtó programot is telepíteni kell: a soros illesztő IC típusától függően vagy a Silicon Labs **CP2102** meghajtóprogramját, vagy a WCH honlapjáról a **CH341SER.EXE** programot kell letölteni és telepíteni

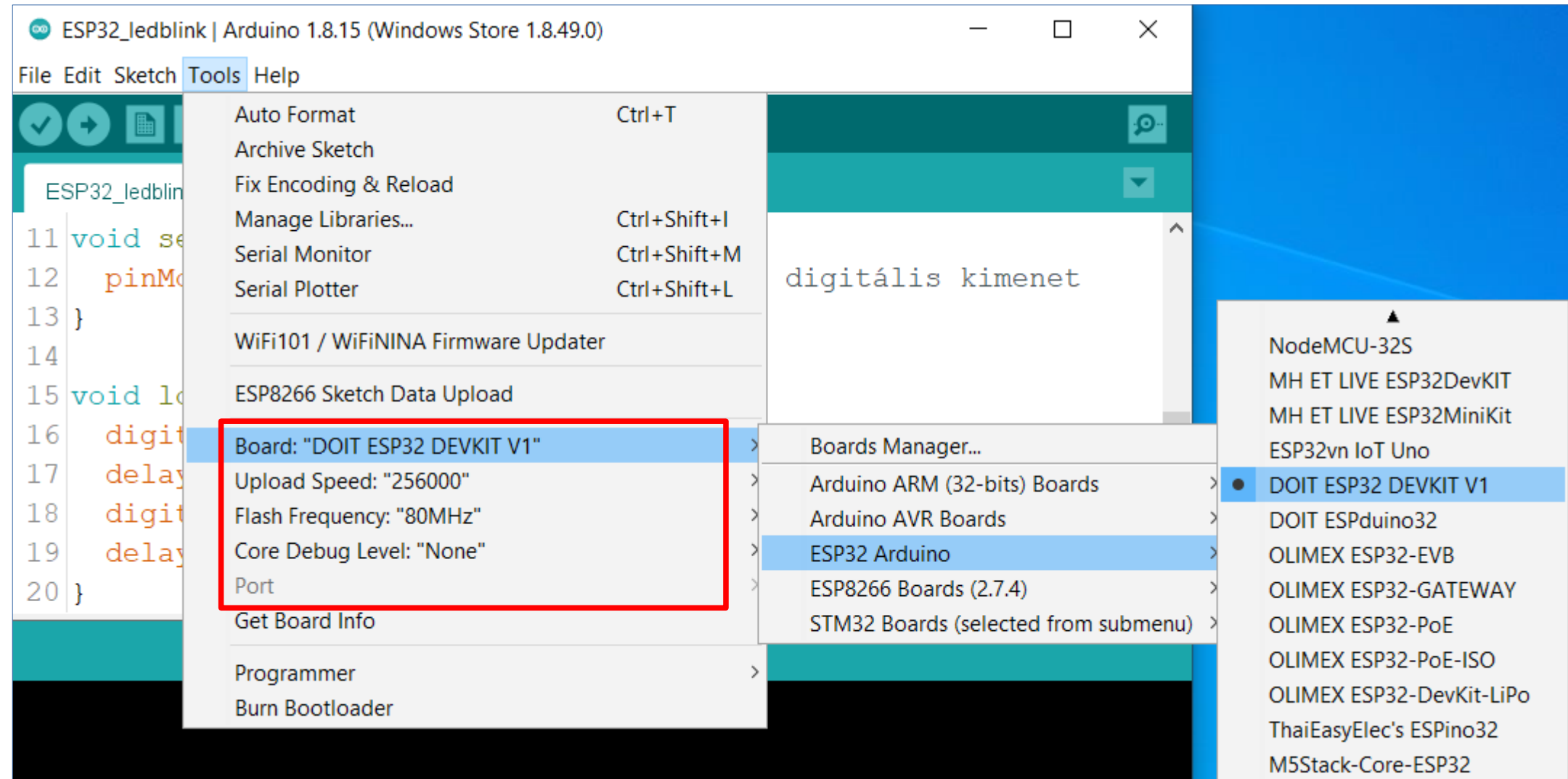
# Az ESP32 hardver támogatás telepítése

- Az Arduino **File/Preferences** menüpontjára kattintunk
- A felbukkanó lapon az Additional Boards Manager URLs rovatba másoljuk be (vagy a mellette levő ikonra kattintva szerkesszük bele a listába) az alábbi sort:  
[https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json)
- Ezután a **Tools** menü **Boards Manager** pontjában a felbukkanó listában választható és telepíthető az **esp32** kártyát támogató programcsomag



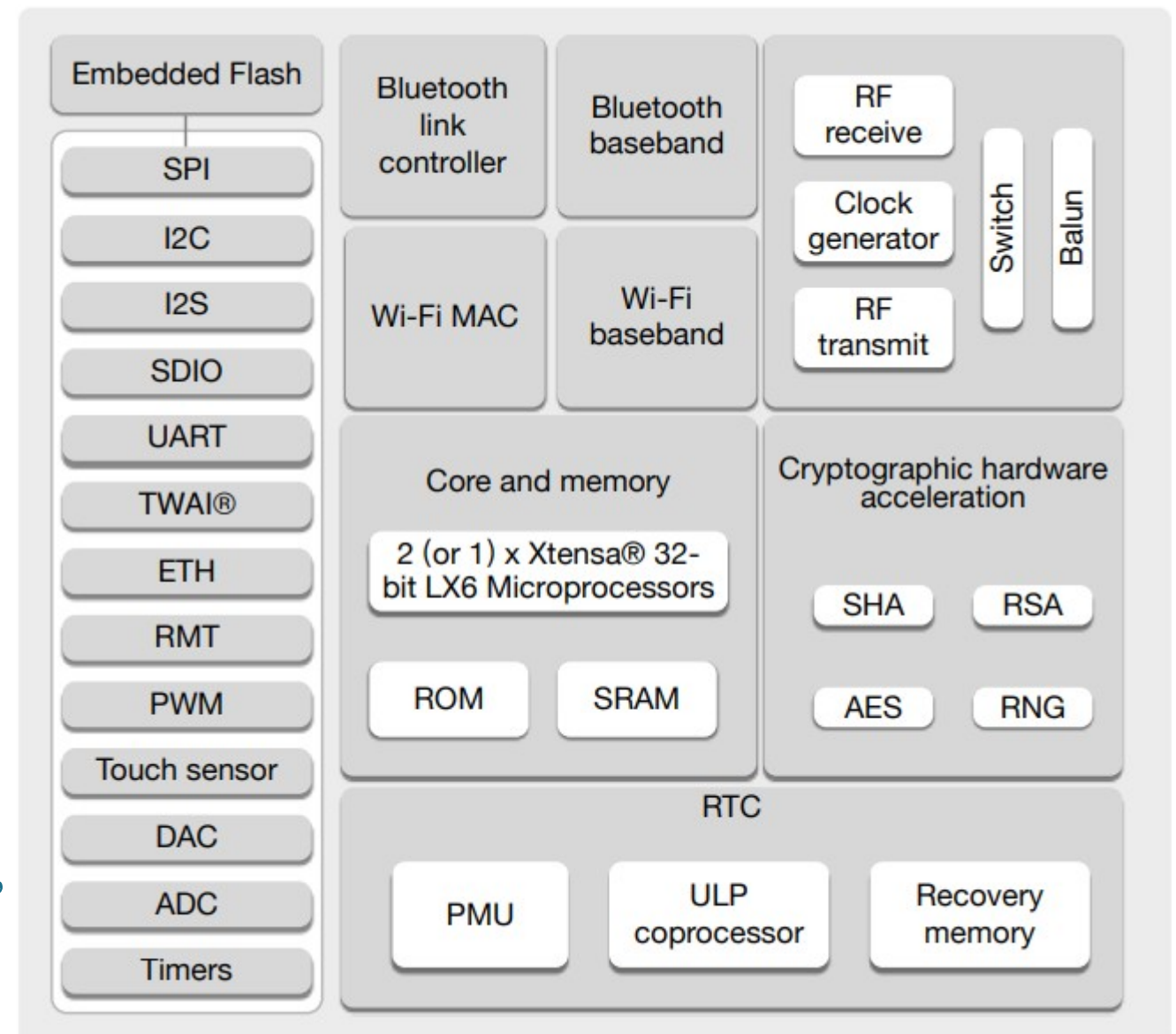
# Az Arduino IDE beállítása

- A **Tools** menüben az **ESP32** kártyák közül válasszuk a **DOIT ESP32 DEVKIT V1**-et és konfiguráljuk az ábra szerint!
- Csatlakoztatás után válasszuk ki a kártyához csatlakozó soros portot! (pl. COM4)



# Az ESP32 mikrovezérlő bemutatása

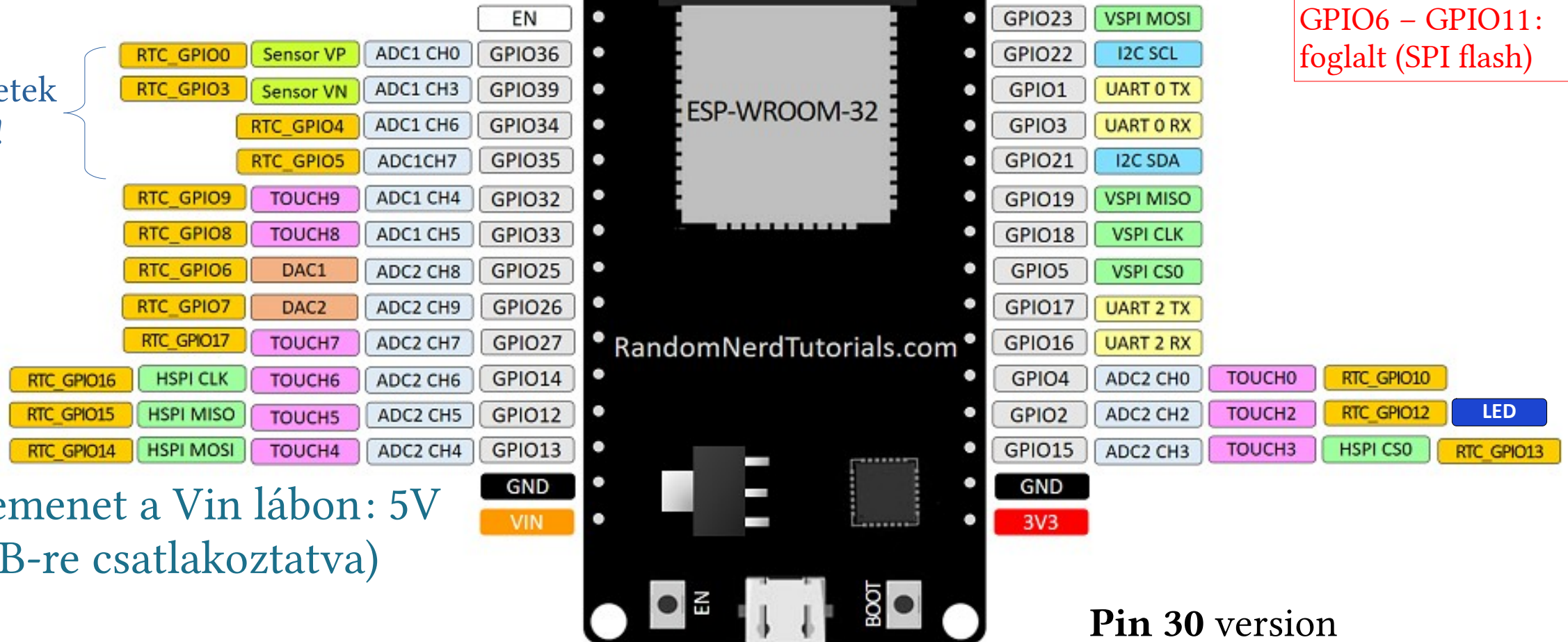
- Az **ESP32** az **Expressif** kétmagos mikrovezérlője, beépített WiFi és Bluetooth/BLE kommunikációs képességgel
- CPU: 2x 32bites Xtensa LX6 mag
- ROM: 448 kB (firmware)
- RAM: 520 kB + 16 kB RTC RAM
- Flash: external QSPI
- 34 GPIO, 18 analog (ADC 12 bit), 10 x touch sensor, 16 x PWM, 2 x 8bit DAC, 3 x SPI, 2 x I2C, 2 x I2S, 3 x UART, IR (Tx/Rx), TWAI, Hall sensor
- 1 host (SD/eMMC/SDIO), 1 slave (SDIO/SPI), 1024-bit OTP, up to 768-bit for customers, Cryptographic hardware acceleration



# Az ESP32 Devkit-1 (DOIT) kártya kivezetései

- A Doit ESP32 Devkit-1 kártya egy ESP WROOM-32 modult (ESP32 + 4MB flash) és az alapkártyán egy CP2102 USB-UART átalakítót, egy 3,3 V-os stabilizátort, egy Reset és egy Boot nyomógombot tartalmaz

Csak bemenetek lehetnek!



GPIO6 – GPIO11: foglalt (SPI flash)

- Tápellátás bemenet a Vin lábon: 5V (ha nincs USB-re csatlakoztatva)

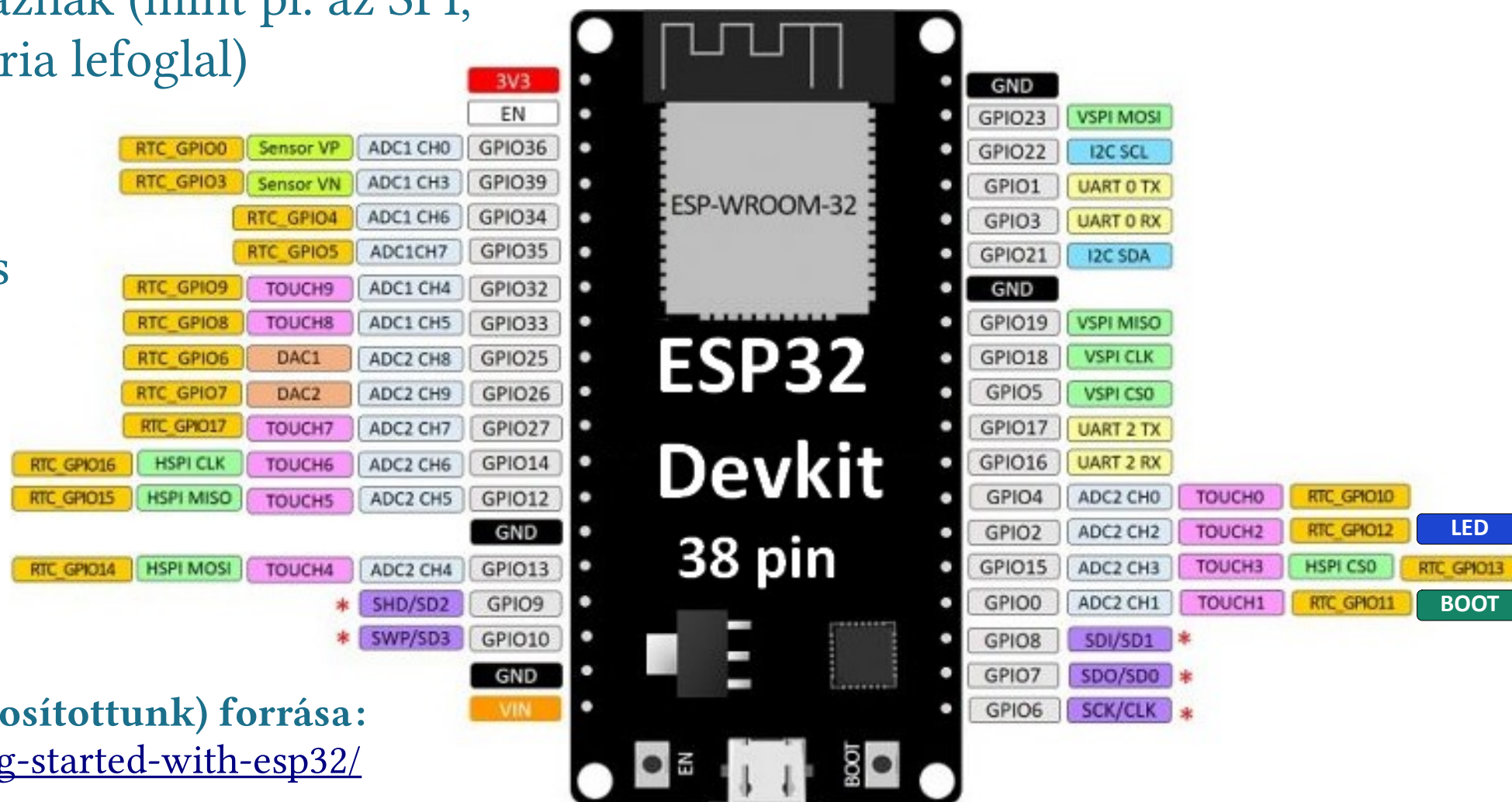
Forrás: [randomnerdtutorials.com/getting-started-with-esp32/](https://randomnerdtutorials.com/getting-started-with-esp32/)

Pin 30 version



# ESP32 fejlesztőkártya, 38 kivezetéssel

- Az ESP32 36, vagy 38 kivezetéses kártyák bekötése gyártófüggő, és többnyire szükségtelen kivezetéseket tartalmaznak (mint pl. az SPI, amelyet a flash memória lefoglal)
- Az általam beszerzett kártya bekötése az ábrán látható, hasznos többletnek csak a **GPIO0** kivezetése mondható, a 30 kivezetéses kártyához képest



Az eredeti ábra (amelyet módosítottunk) forrása:  
[randomnerdtutorials.com/getting-started-with-esp32/](https://randomnerdtutorials.com/getting-started-with-esp32/)

\* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on ESP-WROOM-32 and are not recommended for other uses.

# Digitális I/O

- **pinMode(*pin*, *mode*)** - beállítja a megnevezett kivezetés üzemmódját  
**pin** – a kiválasztott GPIO kivezetés sorszáma (0 – 39, de nem mindegyik elérhető)  
**mode** – üzemmód: **INPUT**, **INPUT\_PULLUP**, **INPUT\_PULLDOWN**, vagy **OUTPUT**  
az **INPUT\_PULLUP** belső felhúzást, az **INPUT\_PULLDOWN** belső lehúzást jelent  
  
Ezt a függvényt többnyire a program **setup()** szekciójában használjuk, a kezdeti beállításoknál
- **digitalRead(*pin*)** – beolvassa a megadott sorszámú kivezetésen a pillanatnyi jelszintet  
**pin** – a kiválasztott GPIO kivezetés sorszáma  
a visszatérési érték a pillanatnyi jelszint, ami 0 (alacsony), vagy 1 (magas) értékű lehet
- **digitalWrite(*pin*, *level*)** – beállítja a korábban kimenetnek állított kivezetésen a jelszintet  
**pin** – a kiválasztott GPIO kivezetés sorszáma  
**level** – a kimeneti szint, ami **LOW** (= 0, alacsony), vagy **HIGH** (=1, magas) értékű lehet

# Digitális I/O

Az általános célú ki- és bemenetek egyedi sajátosságait az alábbi táblázatokban foglaltuk össze

Név	Felirat	Jellemző
GPIO0	–	Boot nyomógomb, felhúzás
GPIO1	TX0	Soros port kimenet
GPIO2	D2	Lehúzás, beépített LED
GPIO3	RX0	Soros port bemenet
GPIO4	D4	lehúzás
GPIO5	D5	felhúzás
GPIO6	–	Az SPI flash memóriához kötve
GPIO7	–	Az SPI flash memóriához kötve
GPIO8	–	Az SPI flash memóriához kötve
GPIO9	–	Az SPI flash memóriához kötve
GPIO10	–	Az SPI flash memóriához kötve
GPIO11	–	Az SPI flash memóriához kötve
GPIO12	D12	lehúzás
GPIO13	D13	
GPIO14	D14	Induláskor PWM jelet ad ki
GPIO15	D15	Felhúzás, induláskor PWM jelet ad
GPIO16	RX2	UART2 RX
GPIO17	TX2	UART2 TX

**Boot feltételek**  
indításhoz, (illetve program-letöltéshez)

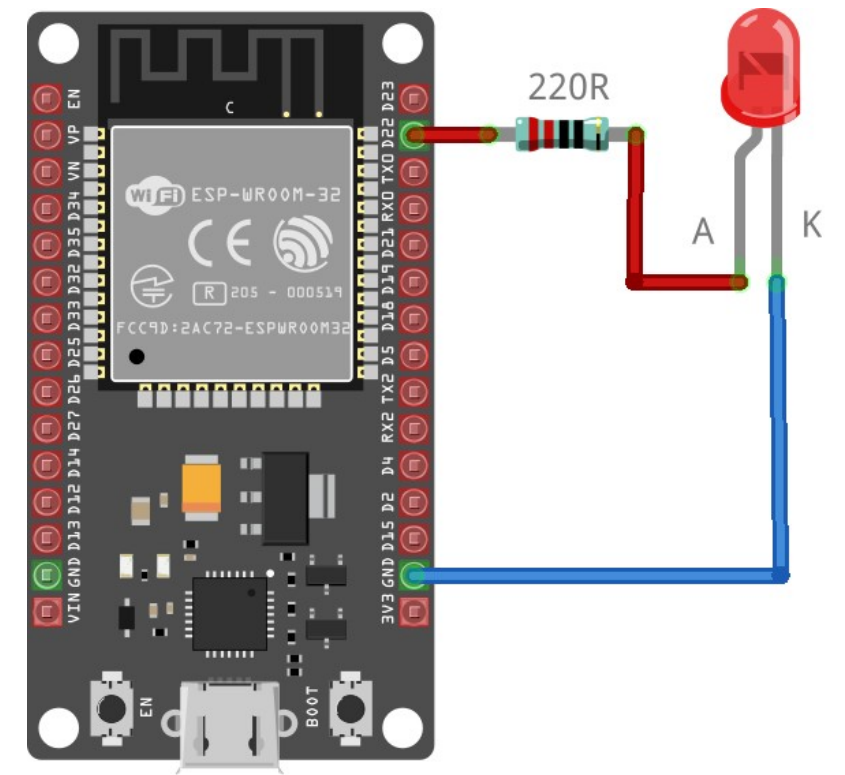
PIN	Level
GPIO0	High (Low)
GPIO2	Low
GPIO5	High
GPIO12	Low
GPIO15	High

Név	Felirat	Jellemző
GPIO18	D18	
GPIO19	D19	
GPIO21	D21	I2C SDA
GPIO22	D22	I2C SCL
GPIO23	D23	
GPIO25	D25	
GPIO26	D26	
GPIO27	D27	
GPIO32	D32	
GPIO33	D33	
GPIO34	D34	Nem lehet digitális kimenet
GPIO35	D35	Nem lehet digitális kimenet
GPIO36	VP	Nem lehet digitális kimenet
GPIO39	VN	Nem lehet digitális kimenet

# ESP32\_ledblink.ino – egyszerű LED villogtatás

- Villogtassuk a **GPIO22** kimenetre kötött LED-et! A LED áramát egy soros ellenállással (pl. 220 Ω) korlátozhatjuk
- Ha a LED katódját a GND-re kötjük, az anódját pedig az áramkorlátozó ellenálláson keresztül a **GPIO22** kivezetésre, akkor a kimenet magas szintje gyújtja ki a LED-et
- A késleltetéshez a beépített **delay()** függvényt használjuk, a késleltetés idejét milliszekundumokban kell megadni
- **Megjegyzés: GPIO2** használatával a beépített LED villog (lásd: **ESP32\_ledblink2.ino**)

```
void setup() {  
    pinMode(22, OUTPUT);    // GPIO22 legyen digitális kimenet  
}  
  
void loop() {  
    digitalWrite(22, HIGH); // GPIO22 aktív magas  
    delay(1000);           // egy másodperc késleltetés  
    digitalWrite(22, LOW); // GPIO22 aktív alacsony  
    delay(1000);           // egy másodperc késleltetés  
}
```



fritzing

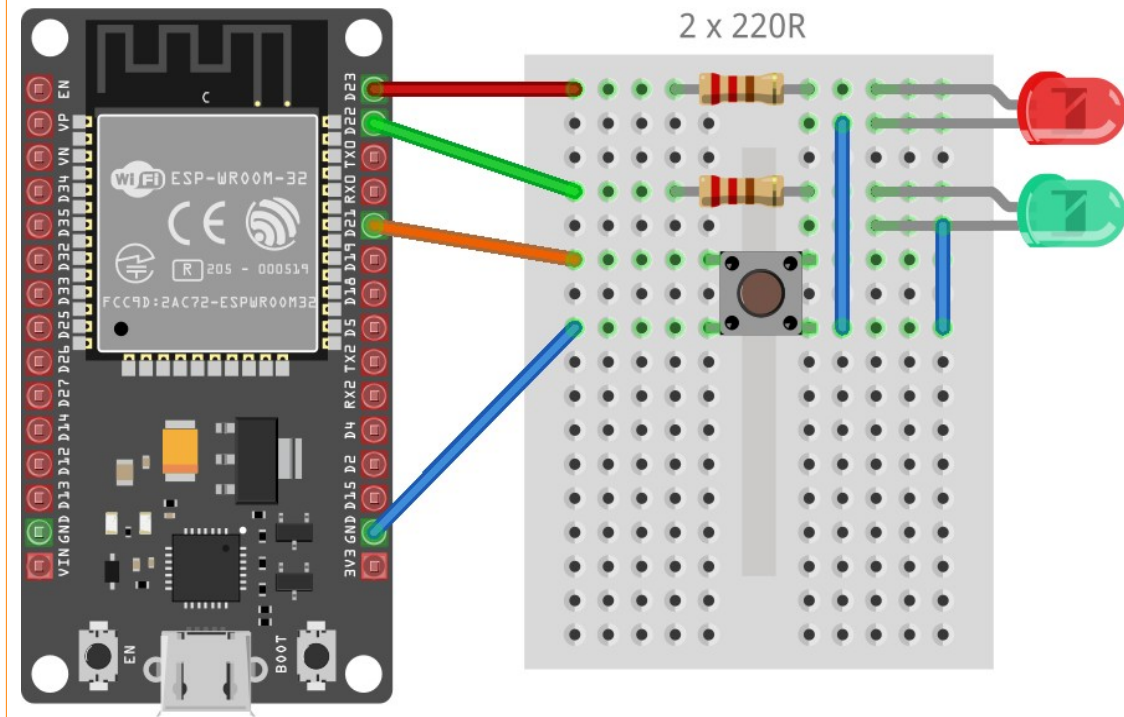
# ESP32\_button2led.ino – nyomógomb állapotának beolvasása

- **Feladat:** A két LED a kapcsoló állásától függően világítson:
  - ❖ Ha a kapcsoló nyitva van, a piros LED világítson!
  - ❖ Ha a kapcsoló zárva van, a zöld LED világítson!
- A nyomógomb állapotát a **digitalRead()** függvénnyel vizsgáljuk!

```
#define RED_LED    23
#define GREEN_LED 22
#define BUTTON    21

void setup() {
  pinMode(RED_LED,OUTPUT);    // legyen kimenet
  pinMode(GREEN_LED,OUTPUT); // legyen kimenet
  pinMode(BUTTON,INPUT_PULLUP); // Bemenet belső felhúzással
}

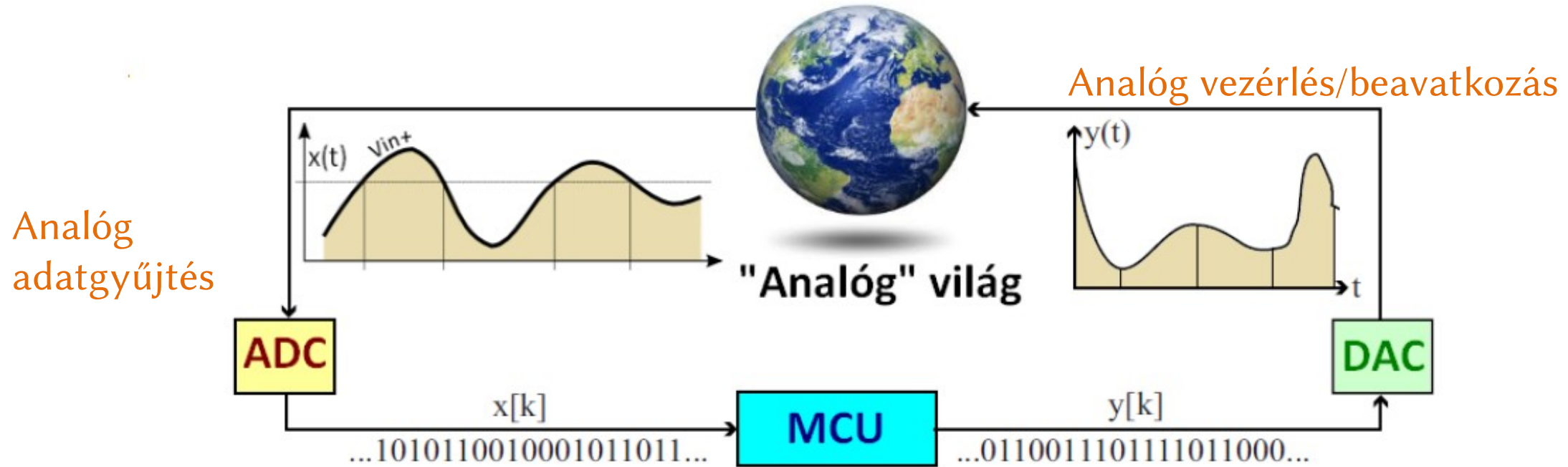
void loop()
  int state = digitalRead(BUTTON)
  digitalWrite(RED_LED, state); // világít, ha state = HIGH
  digitalWrite(GREEN_LED,!state); // világít, ha state = LOW
  delay(20); // pergesmentesítő késleltetés
}
```



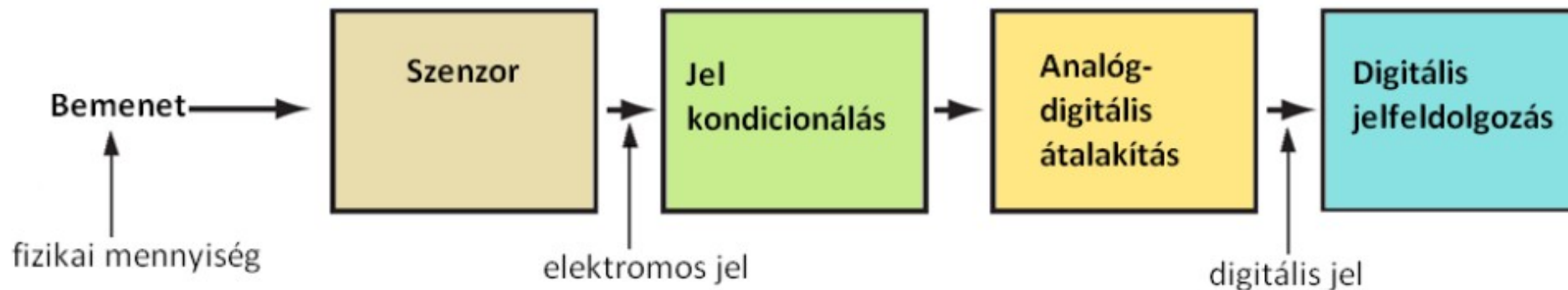
fritzing

# Analóg jelfeldolgozás

- Analóg világban élünk, de digitális mikrovezérlővel dolgozunk...



- Szenzor:** a folytonos fizikai mennyiséget elektromos jellé alakítja



- Jel kondicionálása:** szűrés, erősítés, impedancia illesztés, stb.

# ADC – Analóg-digitális átalakító

- Az ADC feladata az, hogy diszkrét kódokká alakítsa a bejövő analóg jelet
- A konverzió digitális értéke ( $N_{ADC}$ ):

- ❖ Végkitérés:  $N_{ADC} = 4095$ , ha a felbontás 12 bites és a bemenő jel  $\geq V_{R+} - 1.5 * LSB$

- ❖ Nulla:  $N_{ADC} = 0$ , ha a bemenő jel  $\leq V_{R-} + 0.5 * LSB$

- ❖ Közbeeső értékekre:

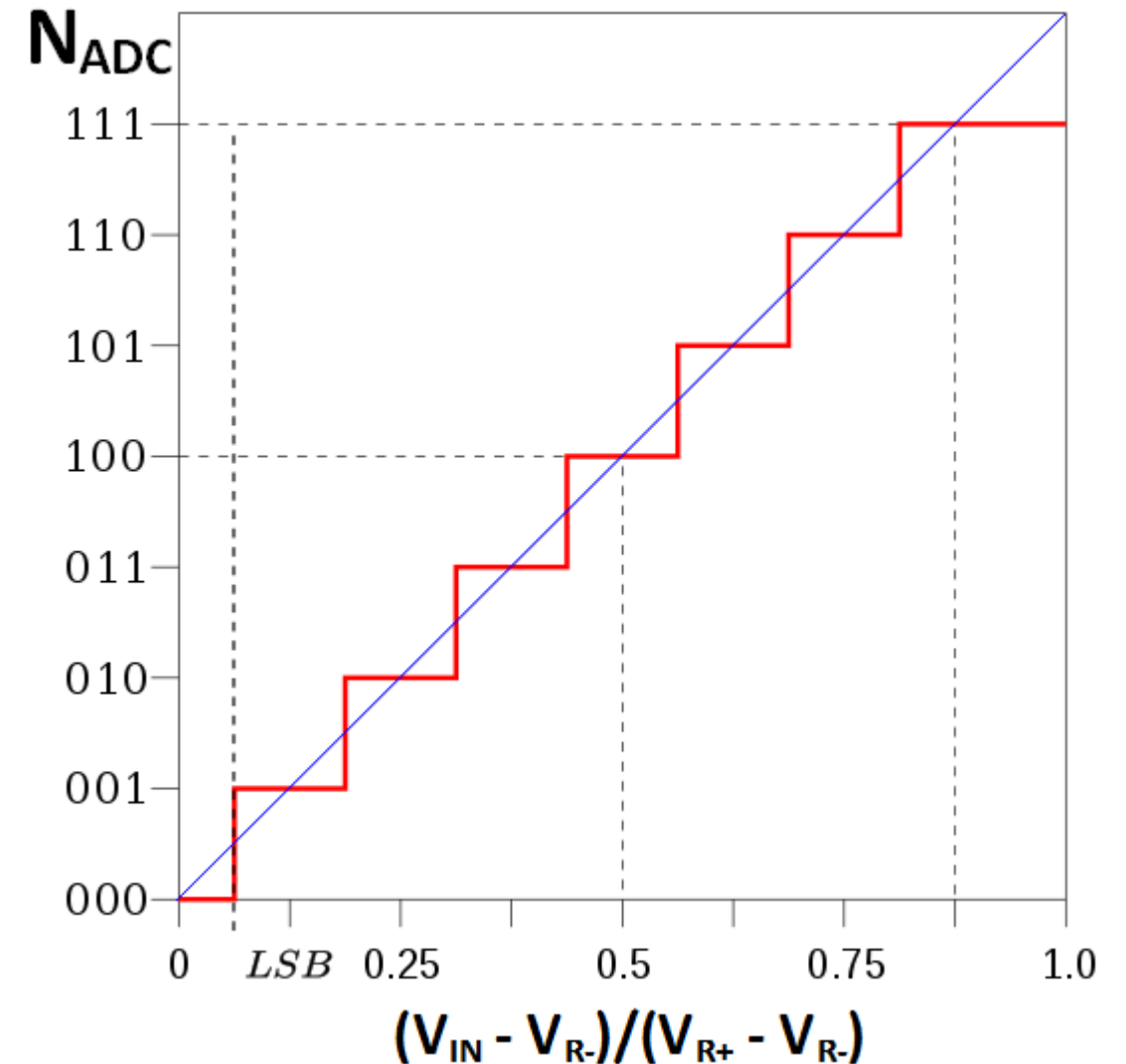
$$N_{ADC} = 4096 * (V_{IN} - V_{R-}) / (V_{R+} - V_{R-})$$

- A fenti képletből  $V_{IN}$ -t kifejezve, ezt kapjuk:

$$V_{IN} = (V_{R+} - V_{R-}) * N_{ADC} / 4096 + V_{R-}$$

- $V_{R-}$  általában = 0

$V_{R+}$  és  $V_{R-}$  a referenciaforrás két sarka



# Analóg bemenetek

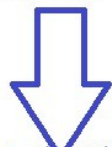
- Két ADC van, de nem mindegyik bemenet érhető el, s ADC2 csak a WiFi letiltott állapotában használható

DOIT ESP32 DEVKIT V1

PINOUT

Microcontrollerslab.com

ADC 1 CHANNELS



	Chip-enable signal, Active High.	EN	pin15						
ADC PA	RTC_GPIO0	ADC1_CH0	SENSOR_VP	GPI036	pin14				
ADC PA	RTC_GPIO3	ADC1_CH3	SENSOR_VN	GPI039	pin13				
	RTC_GPIO4	ADC1_CH6	VDET1	GPI034	pin12				
	RTC_GPIO5	ADC1_CH7	VDET2	GPI035	pin11				
XTAL_32kHz	Touch9	RTC_GPIO9	ADC1_CH4	GPI032	pin10				
XTAL_32kHz	Touch8	RTC_GPIO8	ADC1_CH5	GPI033	pin9				
DAC_1	RTC_GPIO6	ADC2_CH8	EMAC_RXD0	GPI025	pin8				
DAC_2	RTC_GPIO7	ADC2_CH9	EMAC_RXD1	GPI026	pin7				
Touch7	RTC_GPIO17	ADC2_CH7	EMAC_RX_DV	GPI027	pin6				
HS2_CLK	SD_CLK	HSPI_CLK	MTMS	Touch6	RTC_GPIO16	ADC2_CH6	EMAC_TXD2	GPI014	pin5
HS2_DATA2	SD_DATA2	HSPI_MISO	MTDI	Touch5	RTC_GPIO15	ADC2_CH5	EMAC_TXD3	GPI012	pin4
HS2_DATA3	SD_DATA3	HSPI_MOSI	MTCK	Touch4	RTC_GPIO14	ADC2_CH4	EMAC_RX_ER	GPI013	pin3
									pin2
									pin1

ADC2 CHANNELS



pin15	GPI023	SPI_MOSI	HS1_STROBE						
pin14	GPI022	EMAC_TXD1	U0RTS	I2C_SCL					
pin13	GPI01	EMAC_RXD2	U0TXD	CLK_OUT3					
pin12	GPI03		U0RXD	CLK_OUT2					
pin11	GPI021	EMAC_TX_EN		I2C_SDA					
pin10	GPI019	EMAC_TXD0	U0CTS	SPI_MISO					
pin9	GPI018		SPI_CLK	HS1_DATA7					
pin8	GPI05	EMAC_RX_CLK	SPI_CS0	HS1_DATA6					
pin7	GPI017	EMAC_CLKOUT180	U2_TXD	HS1_DATA5					
pin6	GPI016	EMAC_CLKOUT	U2_RXD	HS1_DATA4					
pin5	GPI04	EMAC_TX_ER	ADC2_CH0	RTCI010	Touch0	HSPHID	SD_DATA1	HS2_DATA1	
pin4	GPI02		ADC2_CH2	RTCI012	Touch2	HSPHWP			
pin3	GPI015	EMAC_RXD3	ADC2_CH3	RTCI013	Touch3	MTDO	HSPH_CS0	SD_CMD	HS2_CMD
pin2	GND								
pin1	VDD 3V3								

ADC2 CHANNELS



POWER
GND
Serial Pin
Analog Pin
Control
Physical Pin
Port Pin
Touch Pin
DAC Pin



playelek.com



22-AUG-2016  
VER 1



# Az analóg-digitális átalakítót kezelő függvények

ADC-vel analóg jeleket mérhetünk meg, ami hasznos egy potméterrel leosztott feszültség vagy egy analóg szenzor jele számszerű értékének meghatározására

- **analogRead(*pin*)** – megméri a megnevezett bemeneten a feszültséget és visszaad egy számot (alapértelmezetten 12 bit a felbontás és kb. 3,3 V a méréshatár)
- **analogReadResolution(*resolution*)** – beállítja a felbontást (9 –12 bit, default: 12)
- **analogSetAttenuation(*attenuation*)** – méréshatár beállítása az összes bemenetre vonatkozóan (**ADC\_0db**: 1 V, **ADC\_2\_5db**: 1,5 V, **ADC\_6db**: 2 V, **ADC\_11db**: 3,3 V)
- **analogSetPinAttenuation(*pin, attenuation*)** – a méréshatár beállítása egy adott lábra
- További ADC kezelő függvények (haladóknak):
  - adcAttachPin(*pin*)** – kivezetés kiválasztása és ADC-hez rendelése
  - analogSetClockDiv(*attenuation*)** – ADC órajel leosztás beállítása (1 – 255, default: 1)

# ESP32\_check\_ADC\_linearity

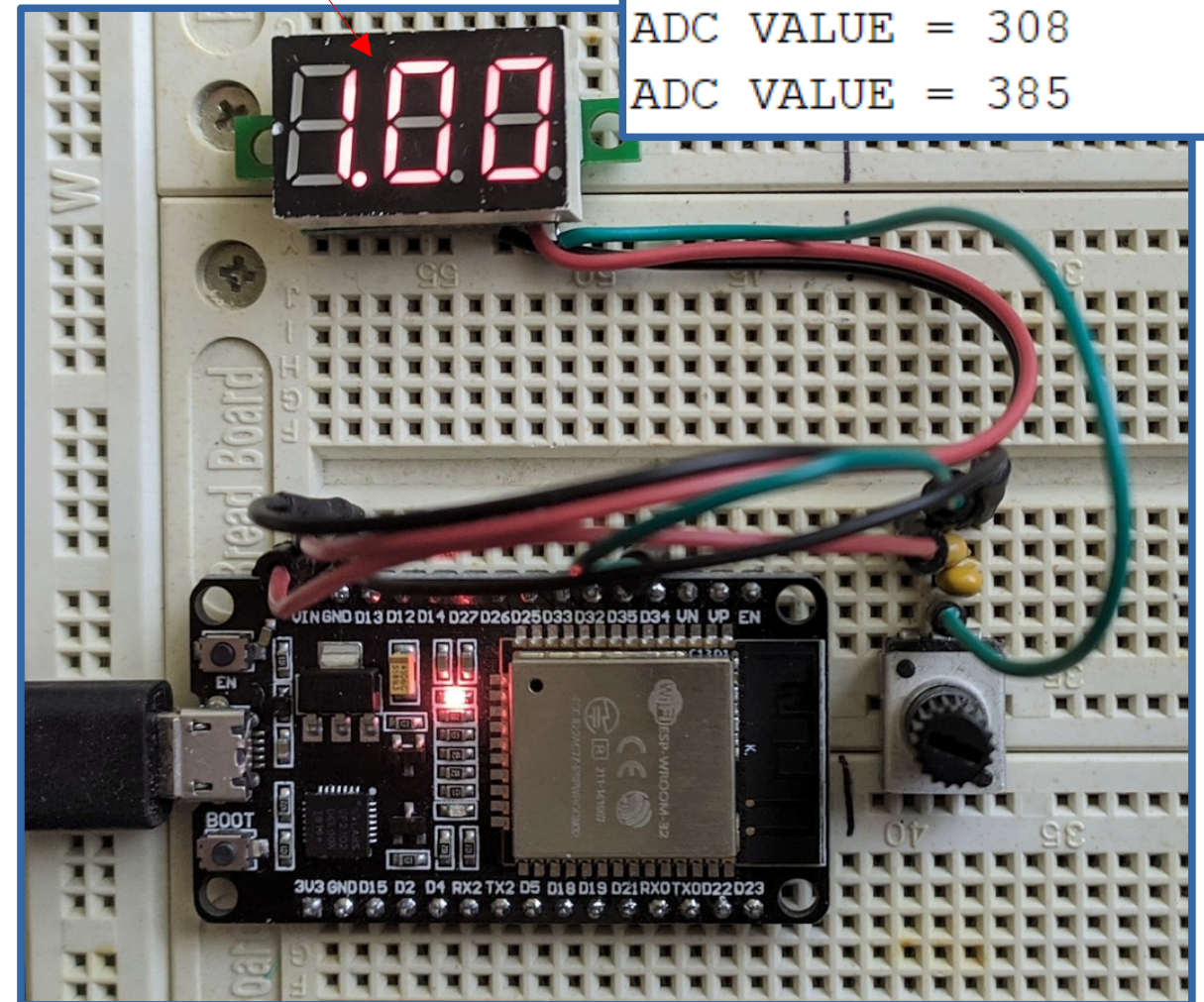
- A GPIO34 bemeneten mérjük a potméterrel leosztott feszültséget

```
const int Analog_channel_pin = 34;
uint32_t ADC_VALUE = 0;
char c;

void setup() {
  Serial.begin(115200);
  analogReadResolution(12); // 12-bit felbontás
  analogSetAttenuation(ADC_11db); // Méréshatár 0-3,3 V
}

void loop() {
  if (Serial.available()) {
    while (Serial.available()) {
      c = Serial.read();
    }
    ADC_VALUE = 0;
    for (int i = 0; i < 1000; i++) {
      ADC_VALUE += analogRead(Analog_channel_pin);
    }
    Serial.print("ADC VALUE = ");
    Serial.println((ADC_VALUE + 500) / 1000);
  }
}
```

DSN-DVM-368



COM4

```
ADC VALUE = 0
ADC VALUE = 62
ADC VALUE = 176
ADC VALUE = 249
ADC VALUE = 308
ADC VALUE = 385
```

# ESP32\_check\_ADC\_linearity

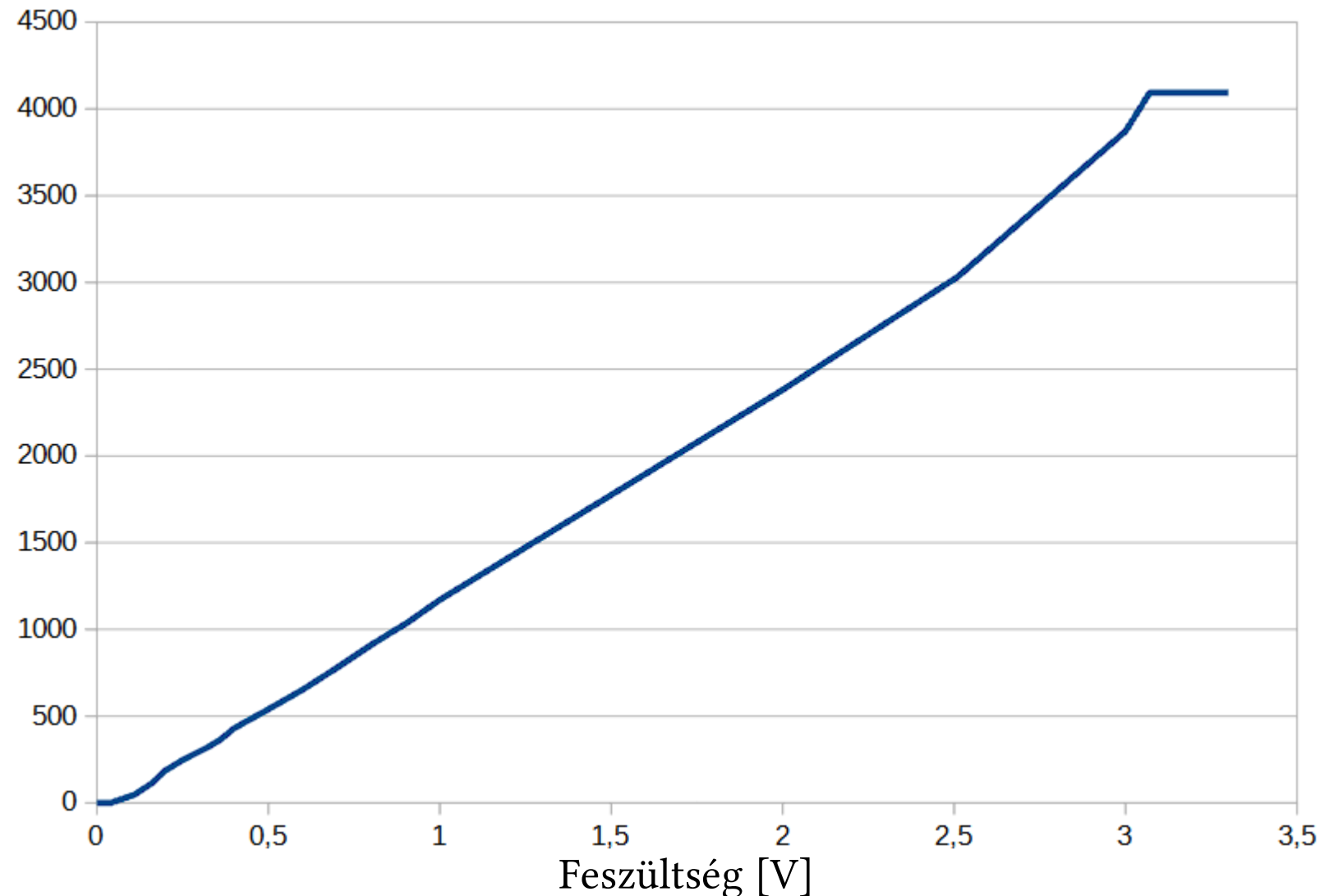
- Az általam kapott mérési adatok alapján a (0,2 – 2,5) V bemeneti tartományban az alábbi közelítő képletet használhatjuk:  $N_{ADC}$

$$N_{ADC} = (1239 \cdot U - 80)$$

- Megfordítva, az ADC átalakítóból kapott számból a feszültséget így határozhatjuk meg:

$$U = (N_{ADC} + 80) / 1239$$

- **Megjegyzés:** a fenti képletekben a feszültség V egységben értendő

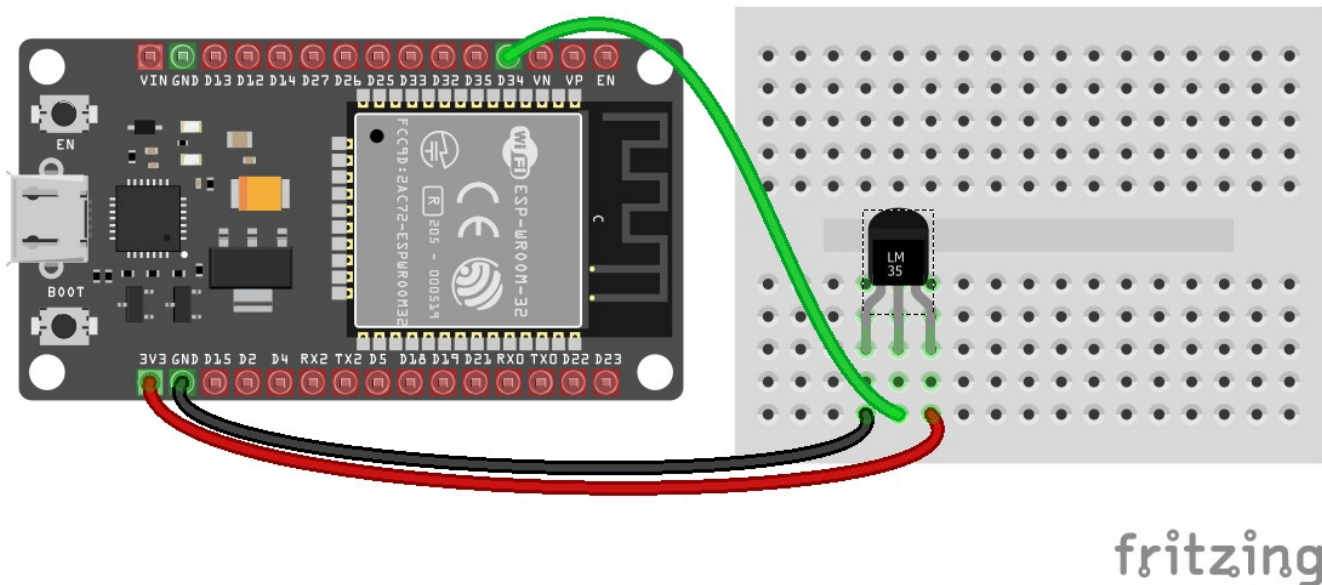
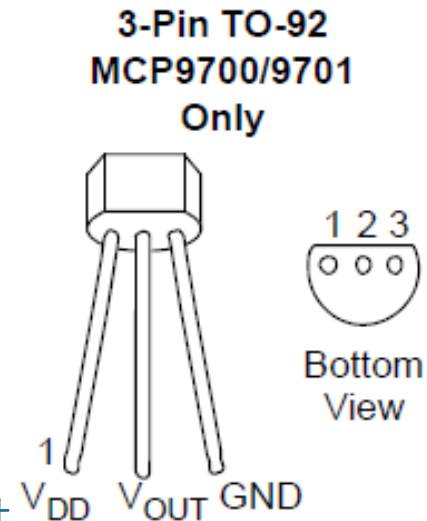


# ESP32\_MCP9700.ino – analóg hőmérő használata

## Microchip MCP9700

- ❖ VDD = 2,5 – 5,5 V
- ❖ Mérési tart.: -40 – 150 °C
- ❖ Érzékenység: 10 mV / °C
- ❖ Nullapont: 500 mV @ 0 °C

- Kössük a hőmérő kimenetét az ESP32 GPIO34 bemenetére!



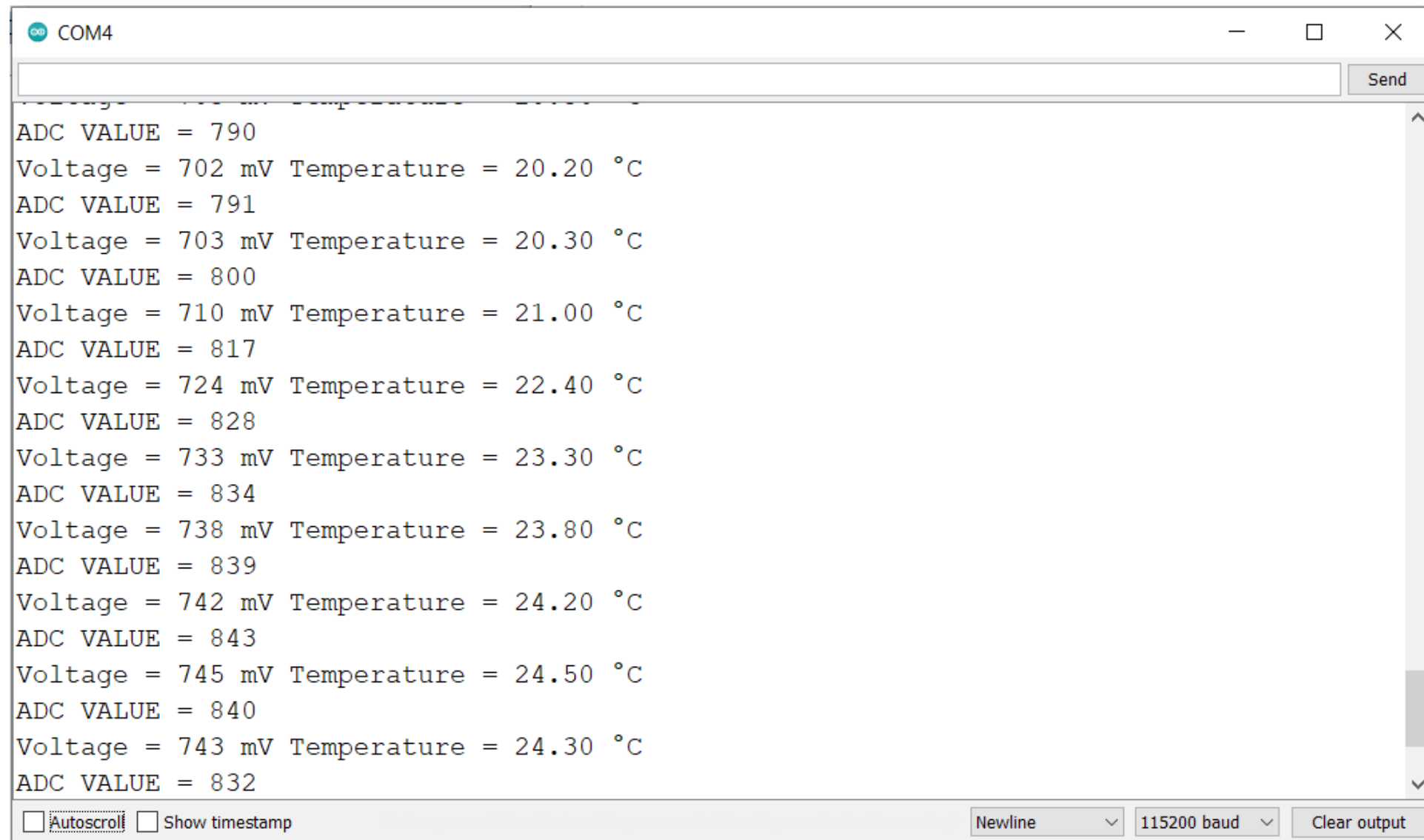
```
const int analog_pin = 34;
uint32_t ADC_VALUE = 0;
uint32_t millivolts = 0;
float tempC;

void setup() {
  Serial.begin(115200);
  analogReadResolution(12); // 12-bit felbontás
  analogSetAttenuation(ADC_11db); // Mérés határ 0-3,3 V
}

void loop() {
  ADC_VALUE = 0;
  for(int i=0; i<1000; i++) {
    ADC_VALUE += analogRead(analog_pin);
  }
  Serial.print("ADC VALUE = ");
  Serial.println(ADC_VALUE/1000);
  millivolts = (ADC_VALUE+80000) / 1239;
  tempC = (millivolts - 500) / 10.0;
  Serial.print("Voltage = ");
  Serial.print(millivolts);
  Serial.print(" mV Temperature = ");
  Serial.print(tempC);
  Serial.println(" °C");
  delay(2000);
}
```

# ESP32\_MCP9700.ino eredmény

- Az ábrán a program futási eredménye látható (kézzel melegítettük a hőmérőt)



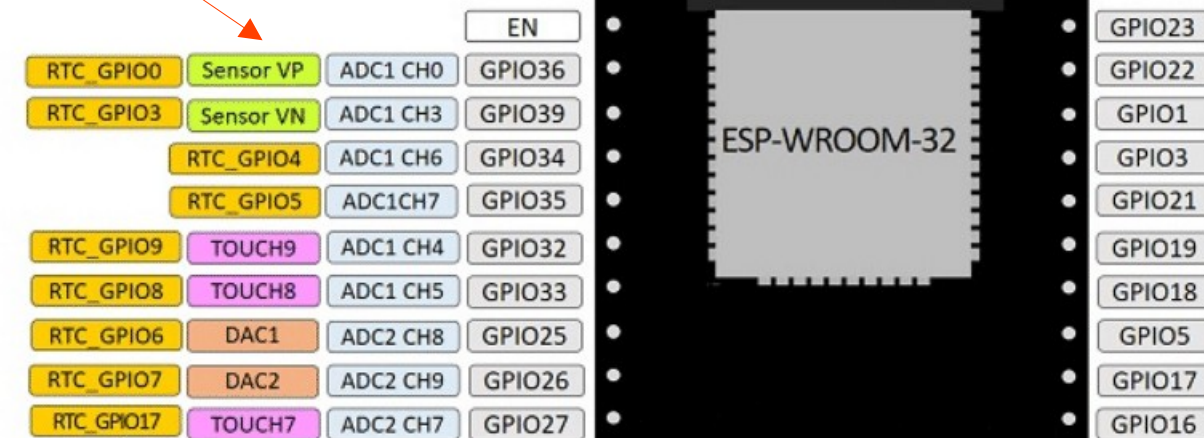
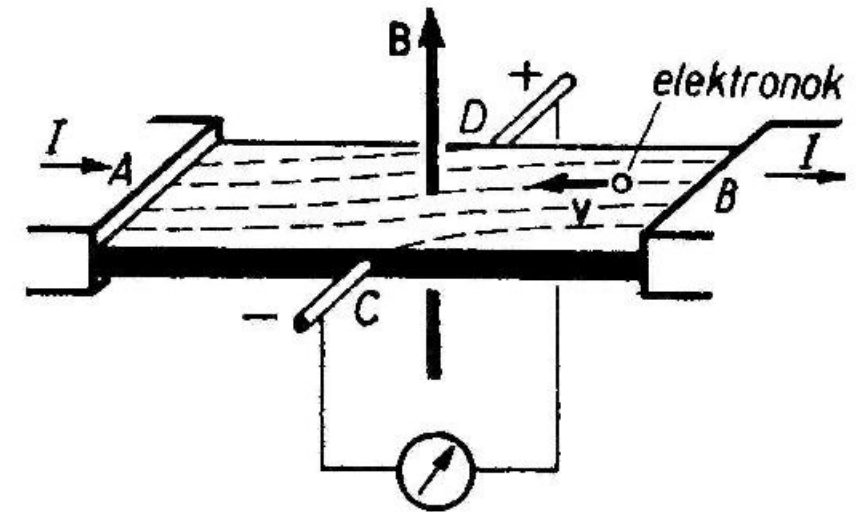
The screenshot shows a serial terminal window titled 'COM4'. The window contains the following text output from the program:

```
ADC VALUE = 790
Voltage = 702 mV Temperature = 20.20 °C
ADC VALUE = 791
Voltage = 703 mV Temperature = 20.30 °C
ADC VALUE = 800
Voltage = 710 mV Temperature = 21.00 °C
ADC VALUE = 817
Voltage = 724 mV Temperature = 22.40 °C
ADC VALUE = 828
Voltage = 733 mV Temperature = 23.30 °C
ADC VALUE = 834
Voltage = 738 mV Temperature = 23.80 °C
ADC VALUE = 839
Voltage = 742 mV Temperature = 24.20 °C
ADC VALUE = 843
Voltage = 745 mV Temperature = 24.50 °C
ADC VALUE = 840
Voltage = 743 mV Temperature = 24.30 °C
ADC VALUE = 832
```

At the bottom of the window, there are several controls: a checkbox for 'Autoscroll' (unchecked), a checkbox for 'Show timestamp' (unchecked), a dropdown menu for 'Newline' (set to '\n'), a dropdown menu for '115200 baud', and a 'Clear output' button.

# ESP32\_HallSensor.ino – a Hall-szenzor használata

- **Hall-effektus:** ha egy félvezető lapka hosszú élével párhuzamosan áram folyik, és a lapkára merőleges irányban mágneses teret hozunk létre, akkor a mágneses térre és az áramra merőleges irányban a mintán elektromos feszültség jön létre.
- A beépített Hall-szenzor a **VP** és **VN (GPIO36 és GPIO39)** bemeneteket használja, ezeket hagyjuk szabadon!
- **HallRead()** – kiolvassa a szenzor értékét
- Alaphelyzetben 25 körüli értéket kapunk
- Mágnessel közelítve, az iránytól függően lefelé vagy felfelé eltérő értéket kapunk (az eredmény negatív szám is lehet!)

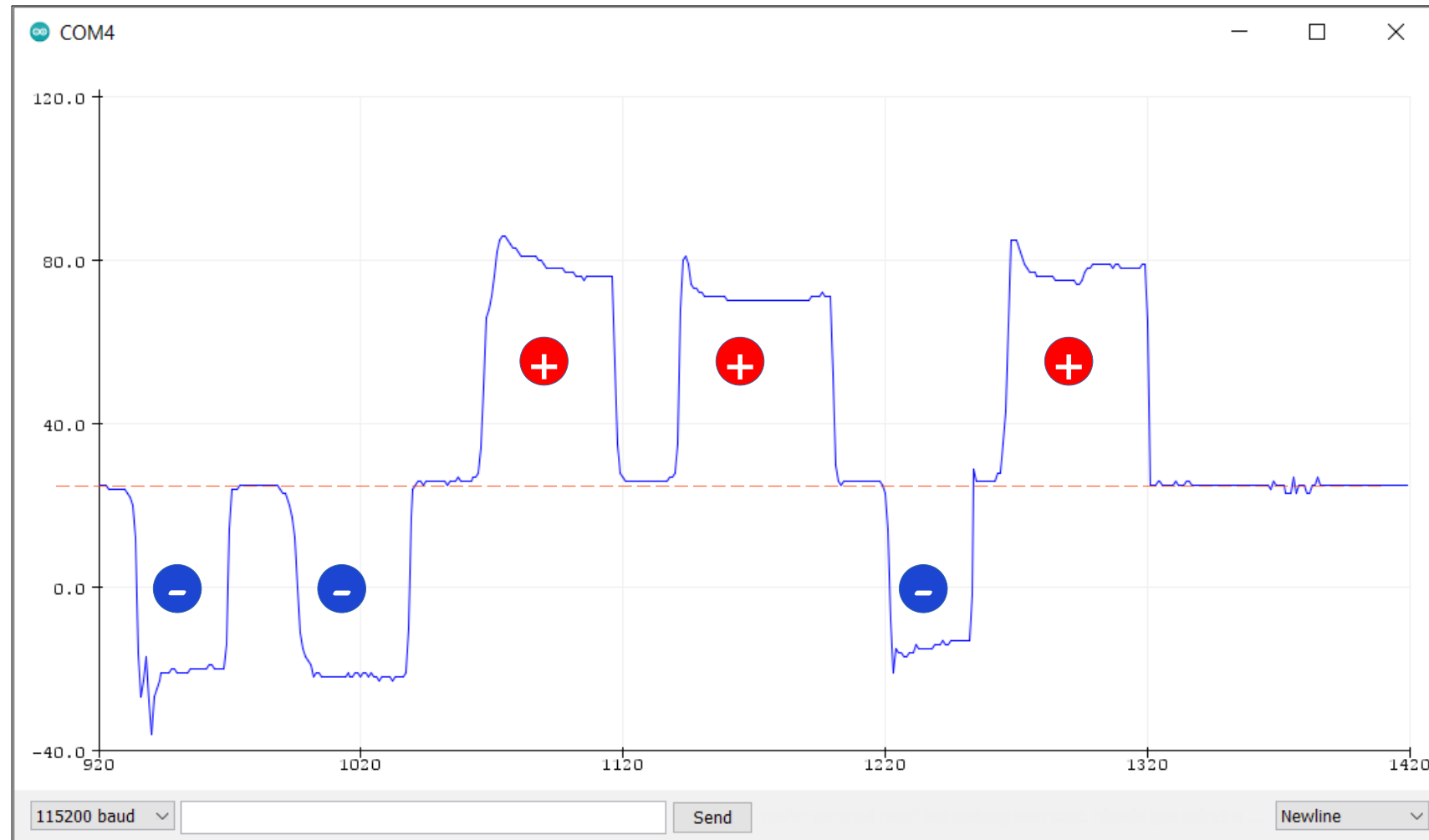


# ESP32\_HallSensor.ino – a Hall-szenzor használata

- Az alábbiakban az **ESP32\_HallSensor.ino** példaprogram listája és futási eredménye látható, futás közben a mágnezt közelítettük, távolítottuk, illetve az irányát megfordítottuk

```
int32_t val = 0;
void setup() {
  Serial.begin(115200);
}

void loop() {
  for (int i = 0; i < 1000; i++) {
    val += hallRead();
  }
  val = val / 1000;
  Serial.println(val);
}
```



# DAC – Digitál-analóg átalakító

Az ESP32 2 db 8-bites DAC kimenettel rendelkezik:

DAC1 → GPIO25

DAC2 → GPIO26

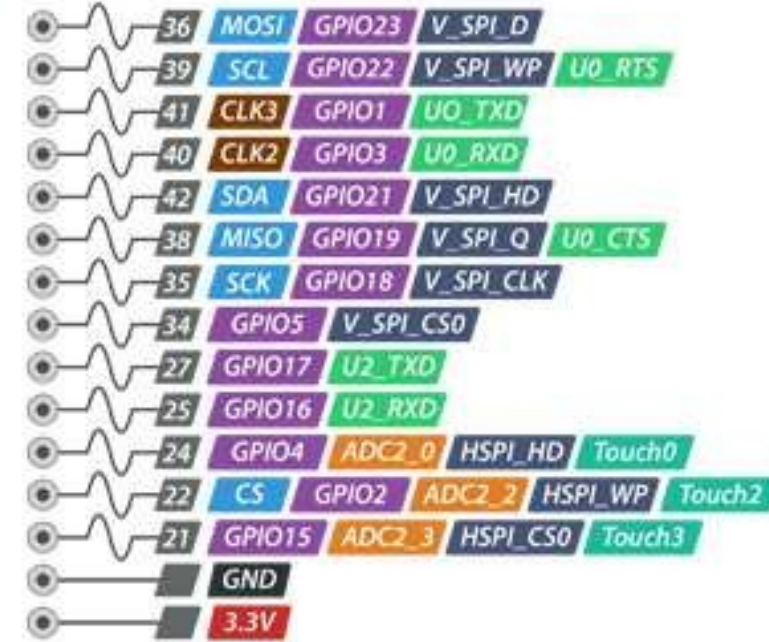
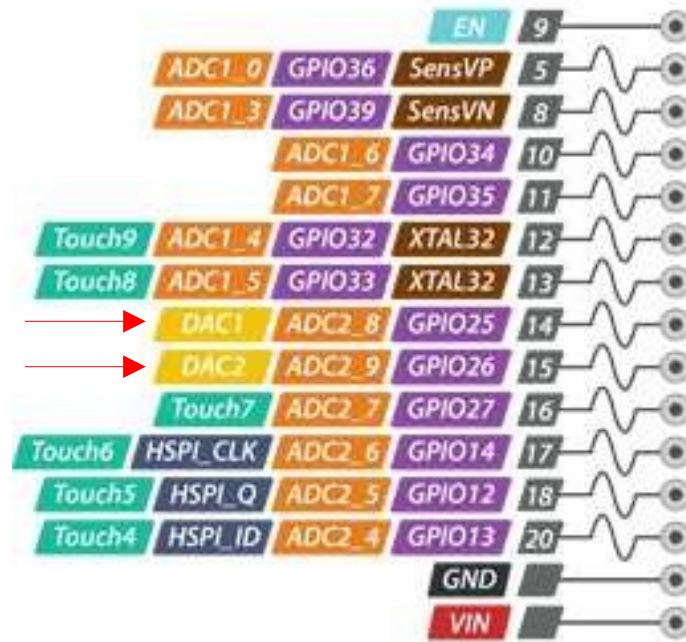
A DAC referenciaforrása a tápfeszültség (esetünkben 3,3 V), így a 0 – 255 bemeneti tartományt a 0 – 3,3 V kimeneti feszültségre képezi le, például:

0 → 0 V

64 → 0,825 V

128 → 1,65 V

255 → 3,3 V



ESP32 Dev. Board Pinout



# ESP32\_DAC.ino

- `dacWrite(pin, value)` – beállítja a megadott DAC kimenet feszültségét  
*pin* – a DAC kimenethez tartozó kivezetés (DAC1: 25, DAC2: 26)  
*value* – 0 – 255 közötti szám
- Az alábbi példában 0-tól 255-ig lépkedünk ötösével, s a GPIO25 kimeneten mérjük a feszültséget

```
#define DAC1 25

void setup() {
}

void loop() {
  for (int i = 0; i <= 255; i = i + 5) {
    dacWrite(DAC1, i);
    delay(2000);
  }
}
```

