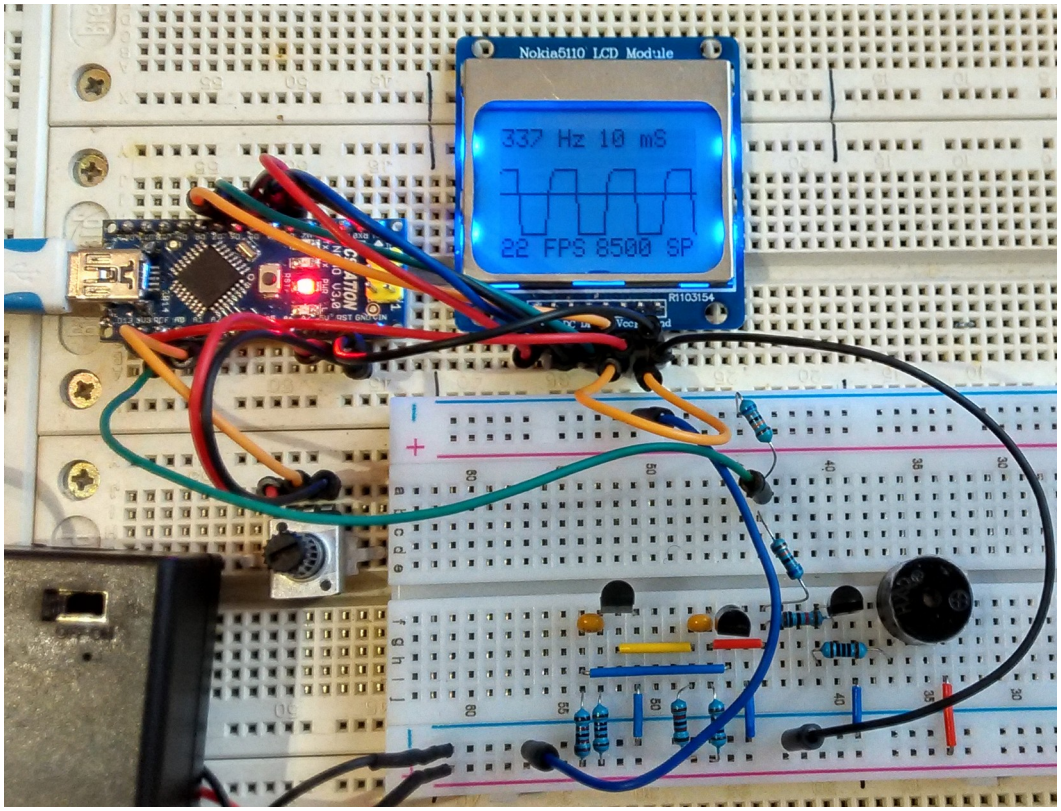
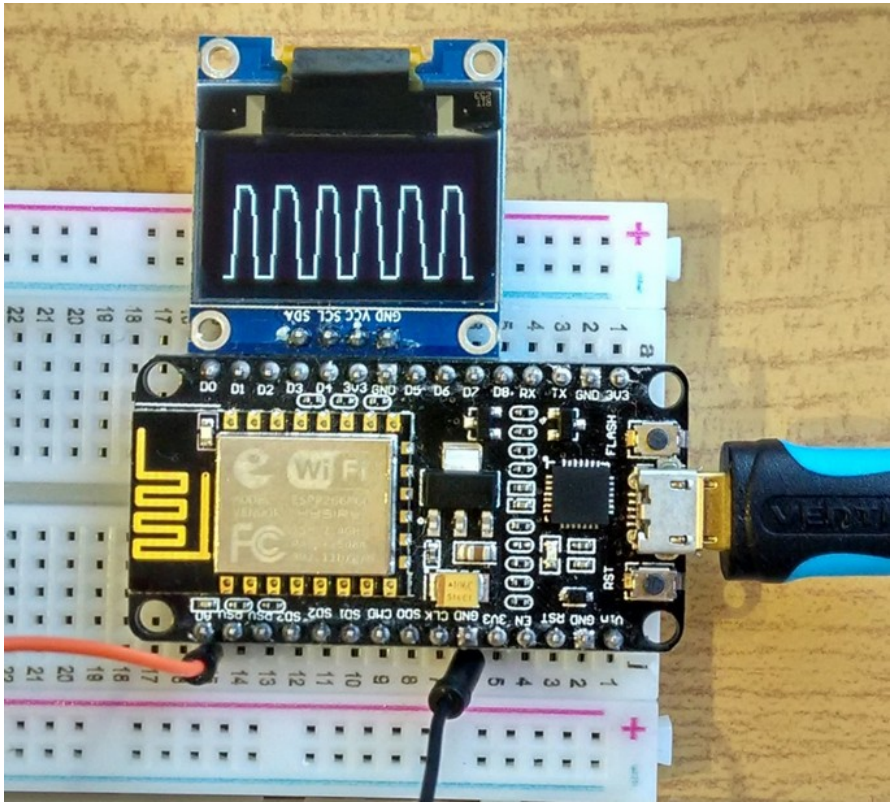
Vegyes témakörök

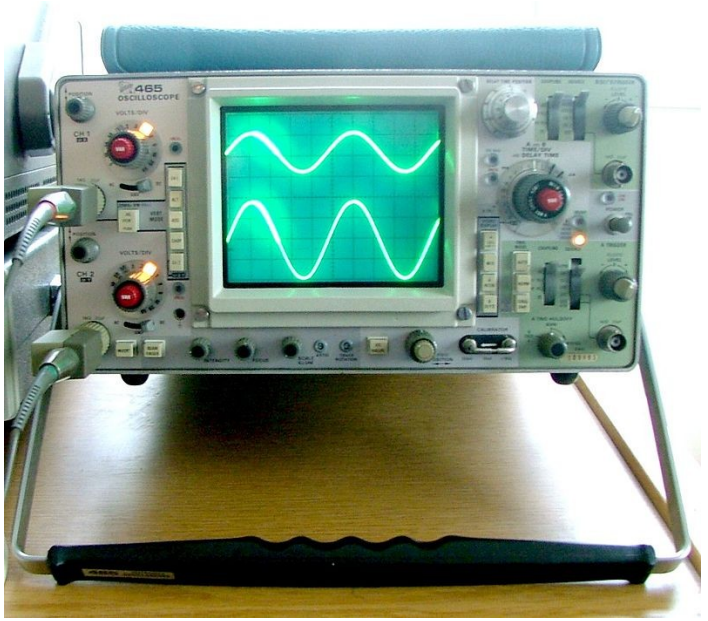


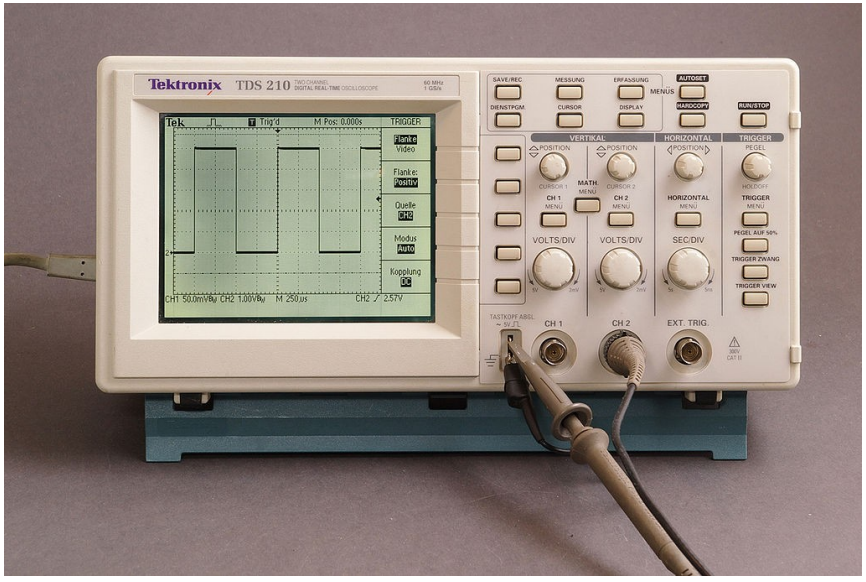


5. Gagyiszkóp házilag – hangfrekvenciás jelek vizsgálata

Analóg és digitális oszcilloszkópok

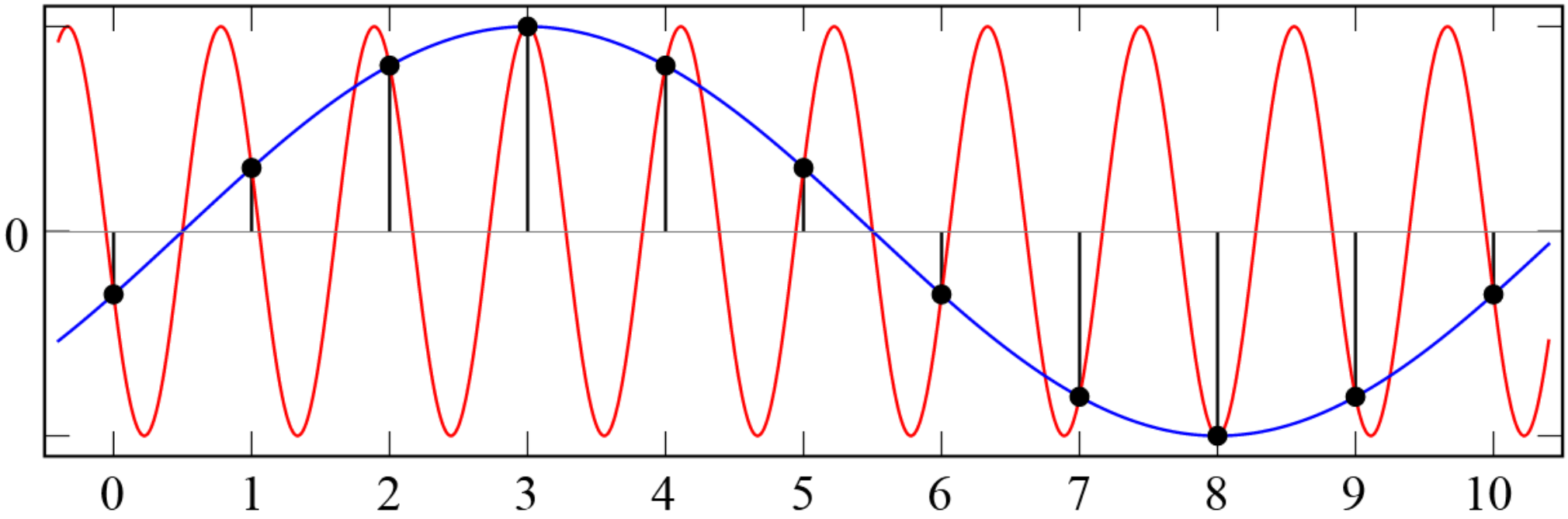
- **Analóg oszcilloszkóp:** a katódsugárcső vízszintes eltérítését egy „ramp generátor” a függőleges eltérítést pedig a felerősített (vagy leosztott) bemenő jel végzi
- Két- vagy többsugaras oszcilloszkópnál két vagy több bemenőjelet vizsgálhatunk
- Analóg eszközöknél a tárolás (a jel rögzítése) nehezen oldható meg
- **Digitális oszcilloszkóp:** a működés elve digitális: a bejövő jelet mintavételezzük, digitalizáljuk (ADC), majd az adatokat kijelzőn (LCD, TFT) megjelenítjük
- A tárolás egyszerűen megoldható





A mintavételezés buktatói

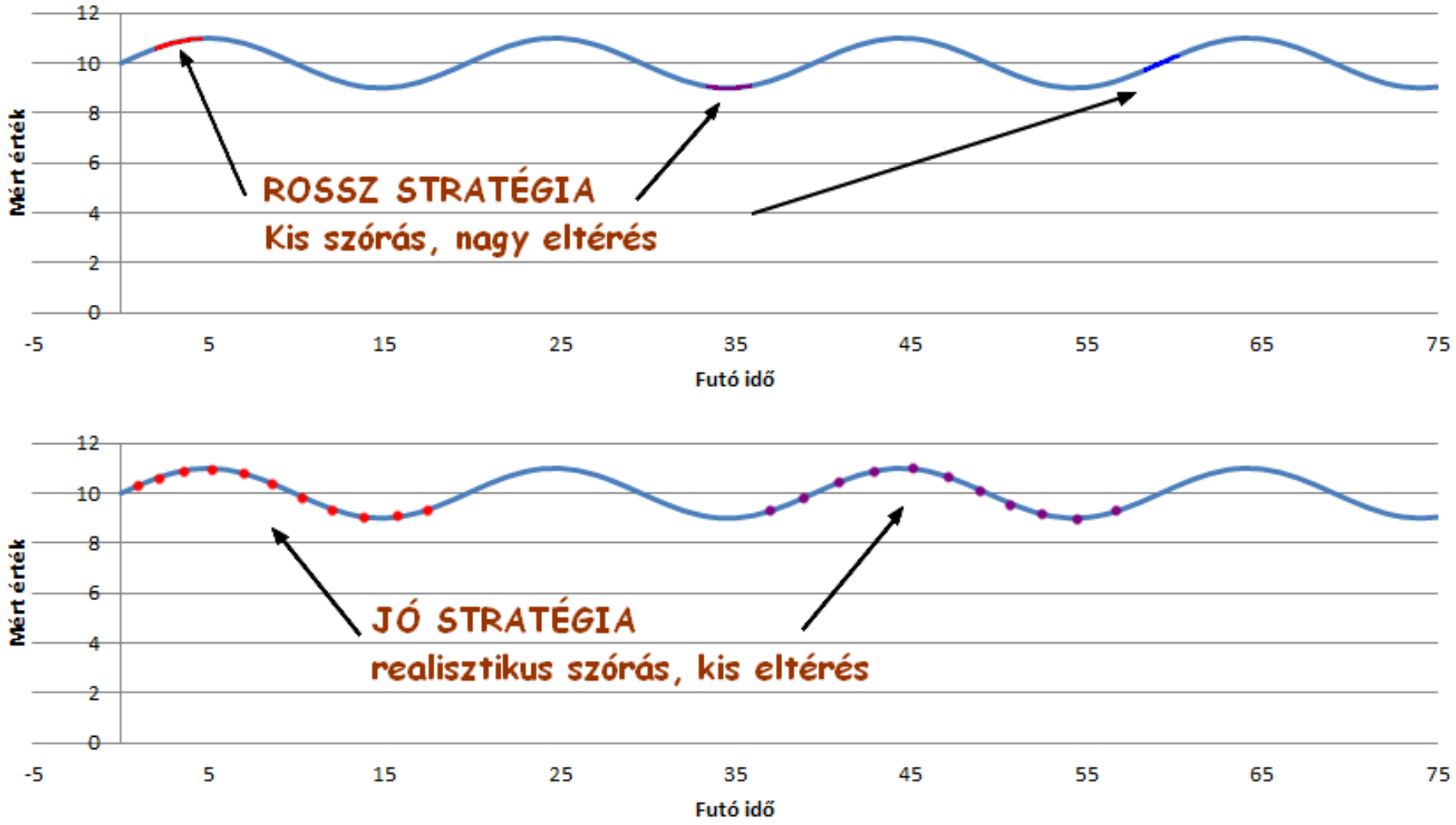
- A mintavételezési frekvencia helytelen megválasztása torzítja a bejövő jel alakját. Szélsőséges esetben olyan jelalakot látunk, ami nem is szerepel az eredeti bejövő jelben (aliasing)



- **Nyquist** [ejtsd: Nükviszt] **kritérium:** a mintavételezési frekvencia a jel frekvenciájának legalább a kétszerese legyen

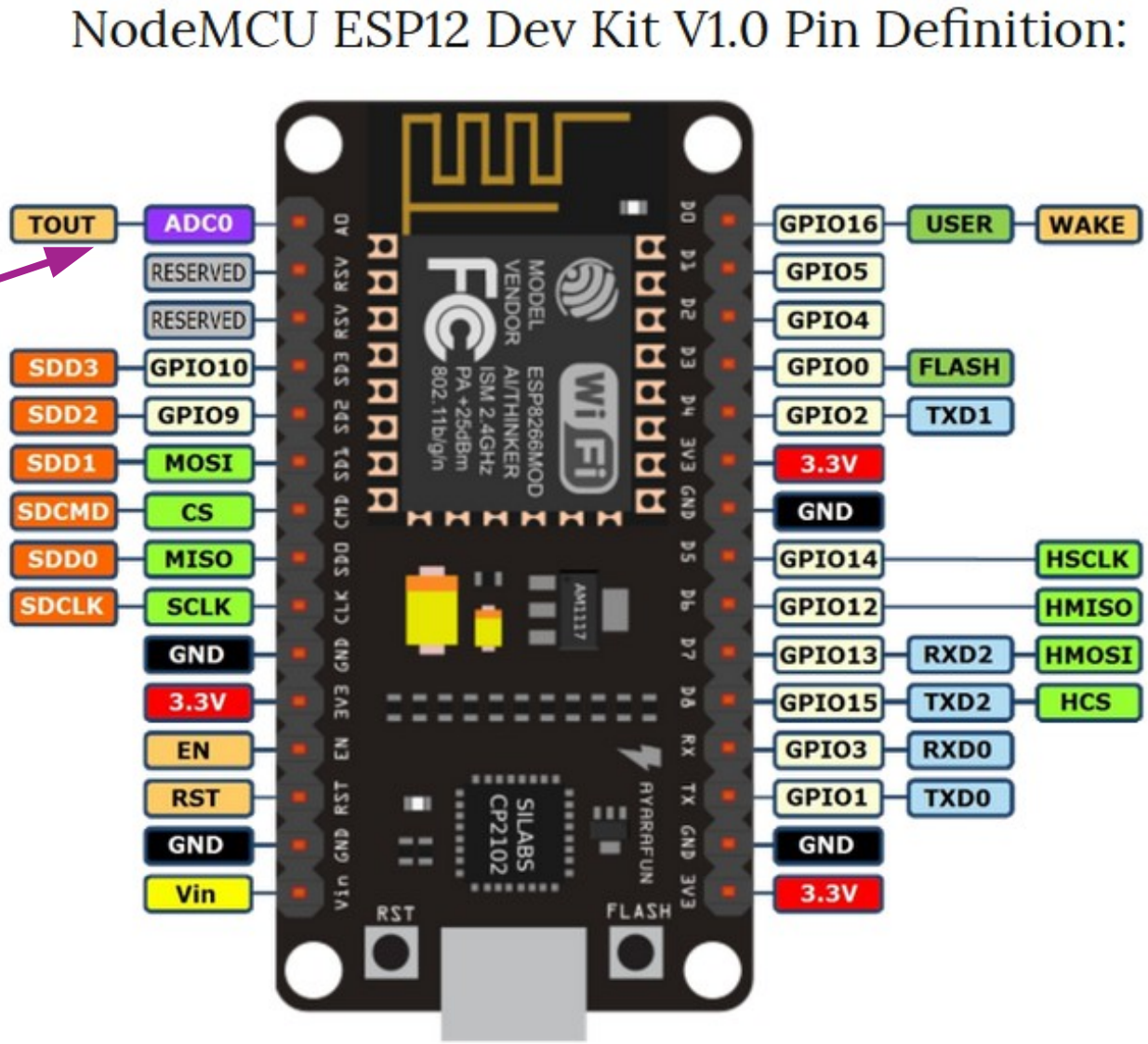
A mintavételezés buktatói

- Zajjal terhelt jel esetén a mintavételezési frekvenciát a zaj frekvenciájához célszerű igazítani (különösen, ha a zaj kiátlagolása a célunk)



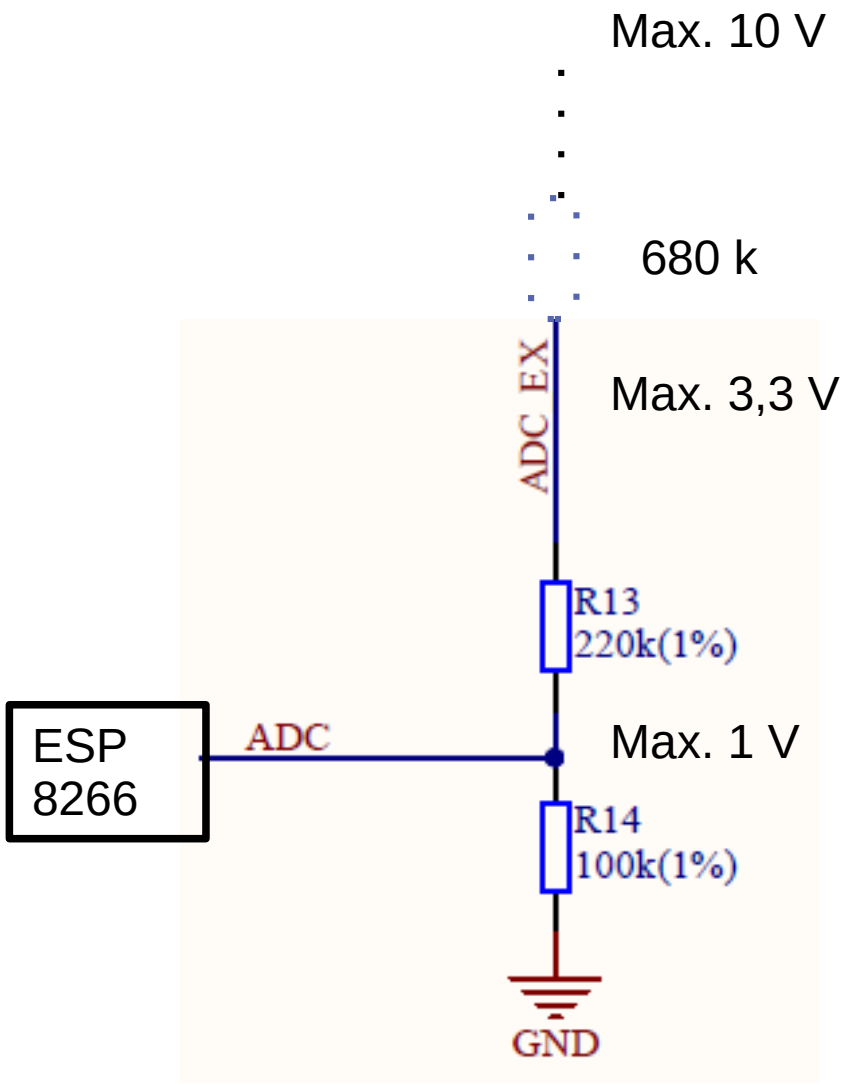
Mintavételezés NodeMCU kártyával

- ESP8266 CPU
  80 / 160 MHz
- 4 MB flash
- WiFi 2.4 GHz
- 1 analog input
- 13 digital I/O
  ebből az UART
  Rx és Tx foglalt
- 3,3 V jelszint és
  tápfeszültség
- MicroPythonban
  a GPIO számozásokat
  kell használni!



NodeMCU ADC bemenet

- Specifikáció szerint 0 – 1 V bemenőjel fogadására képes, amit 10 bites felbontással mér (0 - 1023)
- A NodeMCU már tartalmaz egy feszültségosztót, ami a bemenő feszültséget leosztja, így a méréshatárt 3,3 V-ra terjeszti ki
- Megfelelő értékű soros ellenállás segítségével a méréshatárt kiterjeszthetjük
- Például **680 kΩ** sorbakötésével a méréshatárt 10 V-ra emelhetjük (a felbontás egyidejű csökkenése árán)



adc_meas.py

- Az alábbi **microPython** program 200 mintavételezést végez el, majd az eredményt kétoszlopos formában kiírja
- A 200 mérés alatt eltelt időt egyenletes intervallumokra osztjuk, s a mérés kezdete óta eltelt idő lesz az első oszlopban, ms egységben
- A második oszlopban a V-ban mért feszültséget írjuk ki

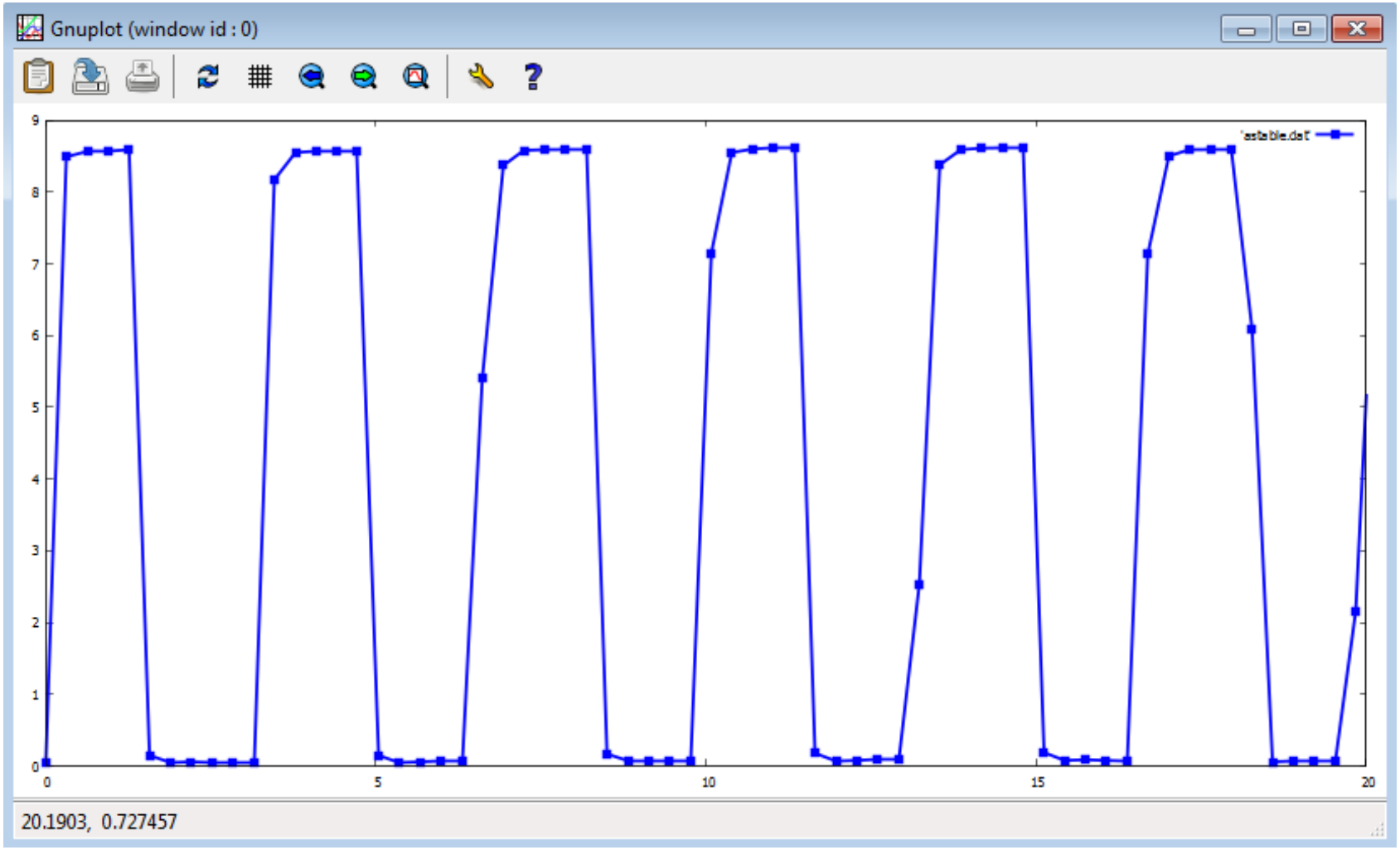
```
import machine, time
adc = machine.ADC(0)

from array import array
aa = array('I',range(200))
a = time.ticks_ms()
for i in range(200):
    aa[i] = adc.read()
b = time.ticks_ms()
c = time.ticks_diff(b,a)/200
for i in (range(200)):
    print("%0.2f %0.2f" % (i*c,aa[i]*10/1024))
```

Tapasztalat:
200 mintavételezés ideje kb. 60 ms (0,3 ms/mintavétel)

Astabil multivibrátor jelének vizsgálata

- Az alábbi ábrán a 2018. október 18-i előadásban bemutatott hangfrekvenciás astabil multivibrátor jelalakja látható. A 200 adatpont mérése kb. 60 ms, itt az adatsor kb. 1/3-a látható.

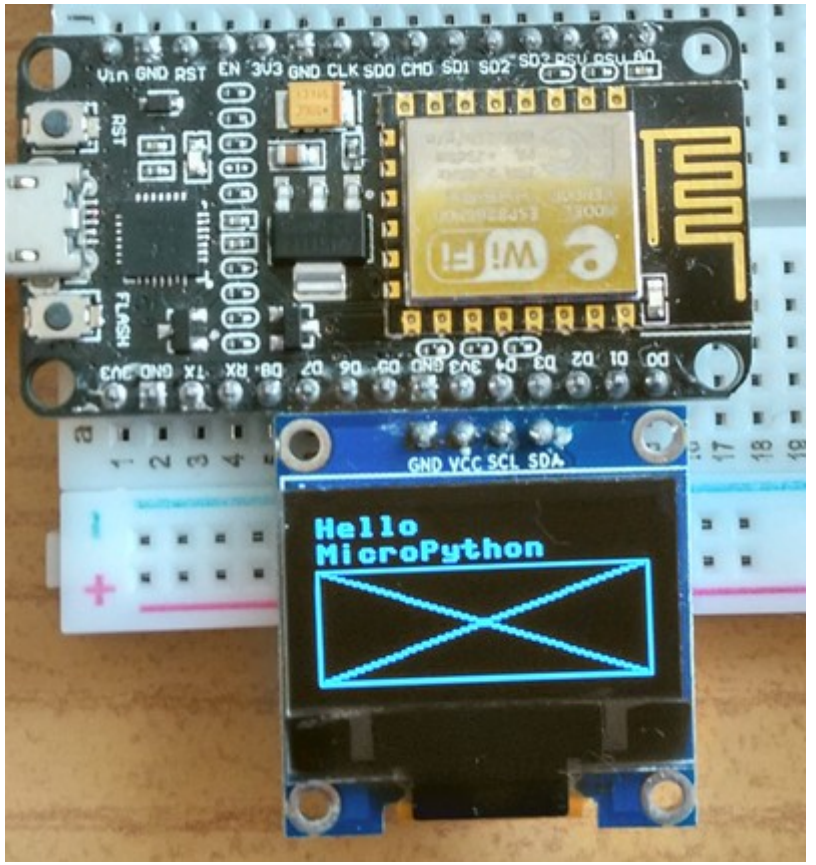


NodeMCU kártya kijelzővel

- Egyszerű és olcsó megoldás az SSD1306 OLED kijelző (I2C)
- Lusta kivitelben csak mellédugjuk a NodeMCU kártyának… (GND – GND, VCC – 3,3V, SCL – GPIO2, SDA – GPIO0)
- Az SSD1306 osztály a microPython **FrameBuffer** osztály „örököse”

oled_lazy.py

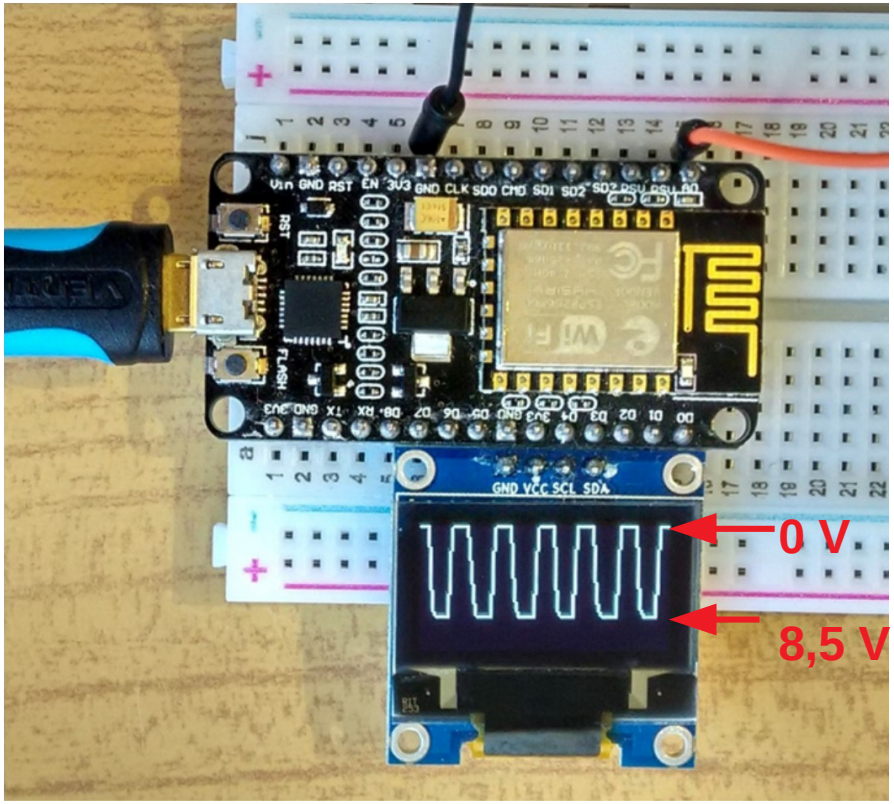
```
from machine import Pin,I2C
import ssd1306
i2c = I2C(scl=Pin(2,sda=Pin(0),freq=100000)
oled = ssd1306.SSD1306_I2C(128,64,i2c)
oled.text('Hello',0,0)
oled.text('MicroPython',0,10)
oled.line(0,20,127,20,1)
oled.line(0,20,0,63,1)
oled.line(0,63,127,63,1)
oled.line(127,20,127,63,1)
oled.line(0,63,127,20,1)
oled.line(0,20,127,63,1)
oled.show()
```



Gagyiszkóp NodeMCU kártyával

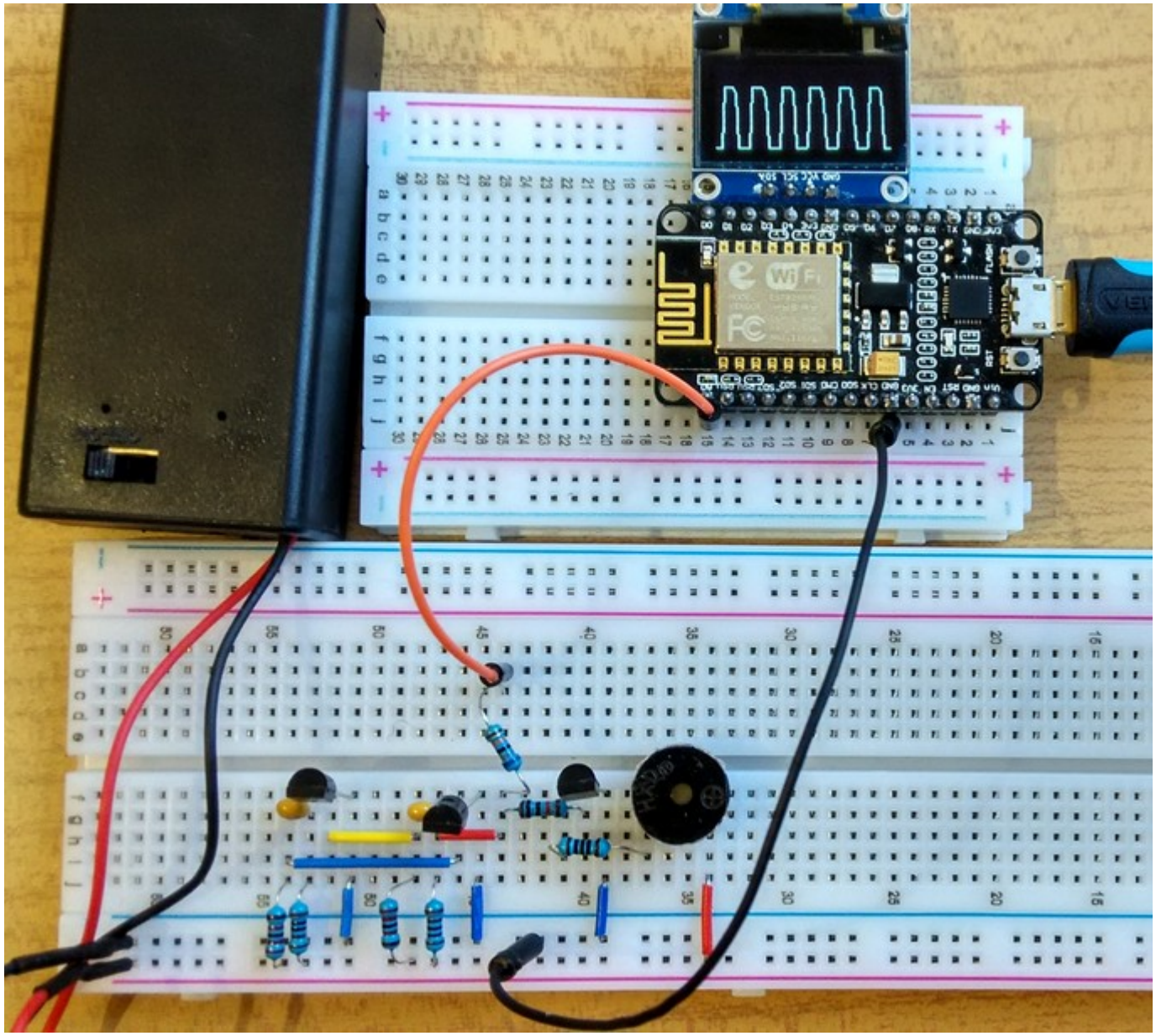
- A mintavételezés a korábban mutatott módon, kb. 3,3 kHz-en történik. Egy adatsor (128 minta) után a képernyőre skálázzuk és megjelenítjük az adatsort
- A transzformáció elhagyása miatt a kép fejjel lefelé jelenik meg!

```
import time, machine, ssd1306
from array import array
adc = machine.ADC(0)
i2c = machine.I2C(scl=machine.Pin(2),
sda=machine.Pin(0), freq=100000)
oled = ssd1306.SSD1306_I2C(128,64,i2c)
aa = array('I',range(128))
for i in range(128):
  aa[i] = adc.read()
x0 =0; y0 = 0
oled.fill(0)
for i in (range(64)):
  x=i*2; y=aa[i]//16          #Nagyítás!!!
  oled.line(x0,y0,x,y,1)
  x0=x; y0=y
oled.show()
```



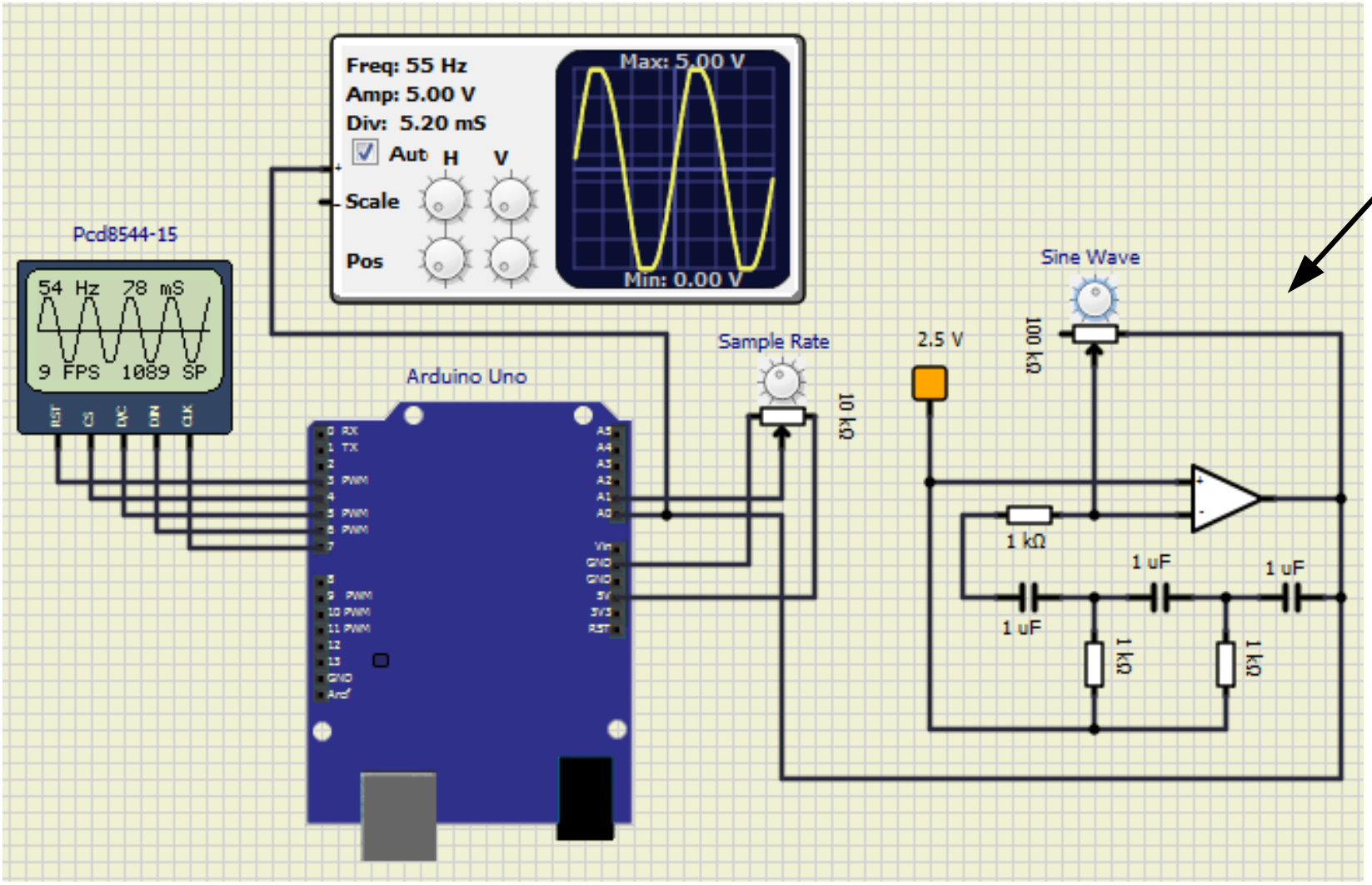
Gagyiszkóp NodeMCU kártyával

- Az ábrán a 2018. október 18-i előadásban bemutatott tranzisztoros hangfrekvenciás astabil multivibrátor jelalakjának vizsgálata látható
- Az idő tengelyen 64*0,3 = 19,2 ms-ra közel 6 periódus esik, így a frek-vencia kb. 312 Hz



Van másik! - Gagyiszkóp Arduinoval

- Oszcilloszkóp Arduinoval és Nokia5110 LCD-vel
- A projekt a SimulIDE egyik Arduino mintapéldája (oscope8544)



Nokia 5110 kijelző vezérlése

- Kiegészítő könyvtárak: Adafruit_GFX és Adafruit_PCD8544

```
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
#include <SPI.h>
#define DISPLAY_WIDTH 84
#define DISPLAY_HEIGHT 48
Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);
void setup(void) {
    display.begin();
    display.setContrast(60);          // Ehelyett más érték kellhet!
    display.clearDisplay();
}
void loop() {
  display.clearDisplay();
  display.drawLine( 0, 24, 83, 24, BLACK); // 0 line
  Lasty = data[0];
  for(i = 1; i <= DISPLAY_WIDTH; i++) {
    display.drawLine(i-1, lasty, í, data[i], BLACK);
    lasty = data[i];
  }
  display.display();
}
```

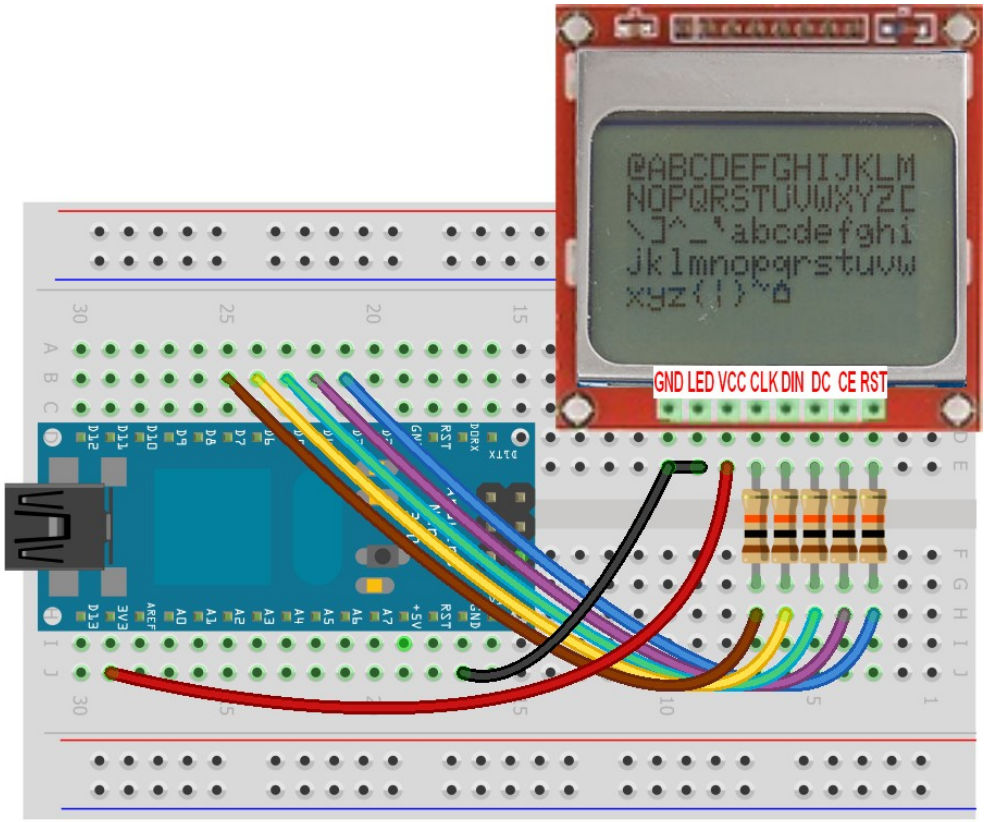
Nem teljes program, csak programvázlat!

(CLK, CIN, DC, CE, RST)

Most ne firtassuk, hogy kerül az adat a data[] tömbbe!

NOKIA 5110 LCD bekötése

- A Nokia 5110 kijelző általában 3,3 V-os tápfeszültséget igényel és az adatbemeneteket is célszerű védeni. Némely esetben azonban az 5 V-ot is jól bírja…
- A lábak kiosztása:
  GND – GND
  GND – BL (LED katód)
  3,3V – VCC
  D7 – CLK (SPI órajel)
  D6 – DIN (SPI adat)
  D5 – DC (data/command)
  D4 – CE (Chip enable)
  D3 – RST (display reset)
- Némelyik panelnél BL-re a LED háttérvilágítás anódja van kivezetve (VCC-re kell kötni)



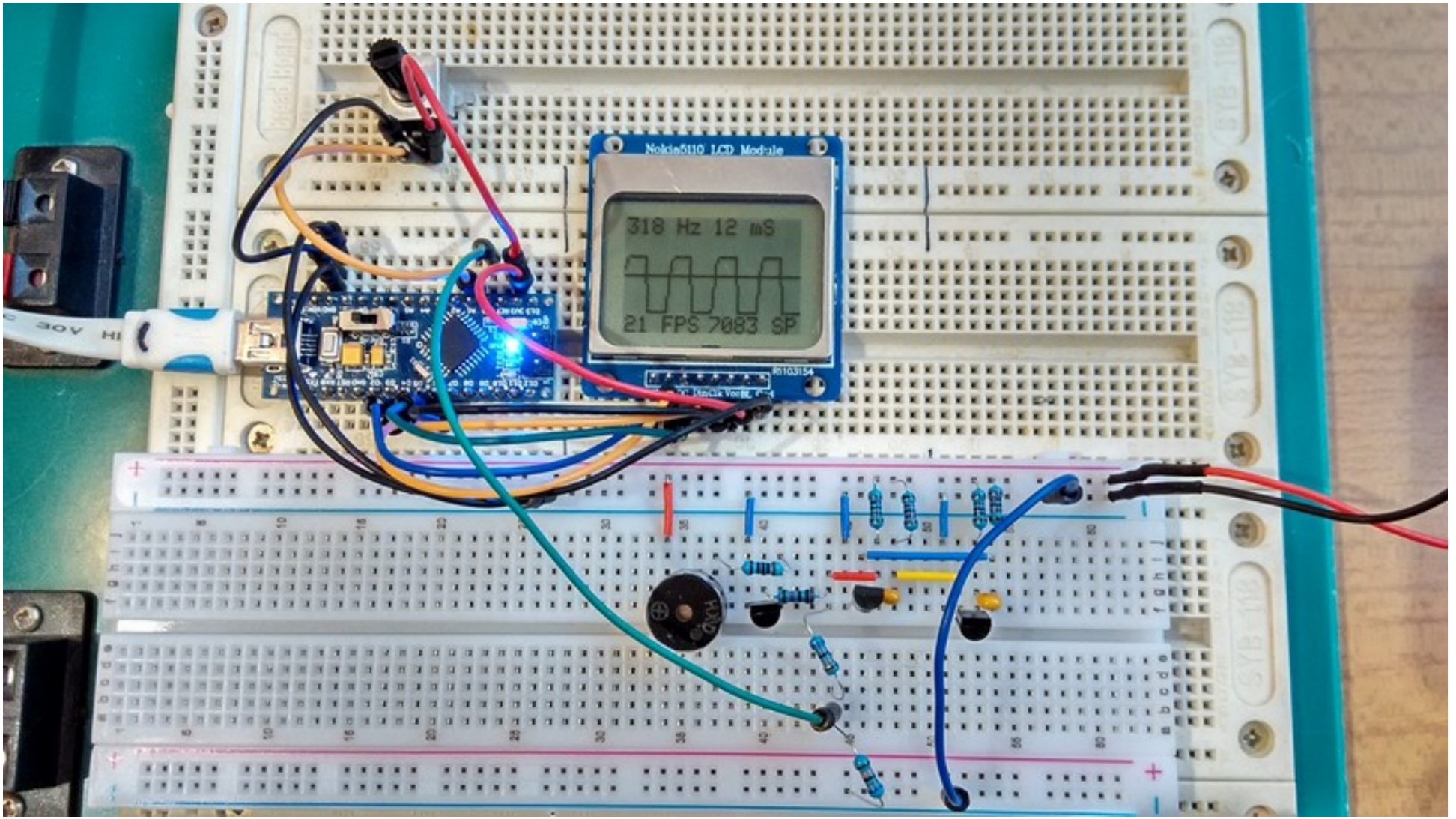
oscope8544.ino részlet: ADC kezelés

- ADC órajel F_CPU/16 legyen! (Alapértelmezetten 128)

```
void setup(void) {
  sbi(ADCSRA,ADPS2) ;   //FASTADC: előosztó legyen 16
  cbi(ADCSRA,ADPS1) ;
  cbi(ADCSRA,ADPS0) ;
}
void loop() {
  delayVariable = analogRead(A1)*2+1;
  while( analogRead(channelAI) > 512 );   // Felfutóélre várunk
  while( analogRead(channelAI) < 512 );
  unsigned long readTime=micros(),readTime2=0, totTime=0;
  bool positive = true; int  numCycles = 0;
  for(xCounter = 0; xCounter <= DISPLAY_WIDTH; xCounter++){
    yPosition = analogRead(A0);
    if( !positive && (yPosition > 512)) { // Felfutóél észlelése
      numCycles++; readTime2 = micros();
    }
    positive = yPosition > 512;
    readings[xCounter] = yPosition>>5; // 0-32 Itt kerül az adat eltárolásra
    delayMicroseconds(delayVariable);  // Potméterrel változtatható
  }                                    // késleltetés 1 - 2047 µs
```

oscope8544: futási eredmény

- A képen egy astabil multivibrátor jelét vizsgáljuk, F = 318 Hz, totTime = 12 ms, 21 fps, 7083 mintavétel/s (141 µs/minta)

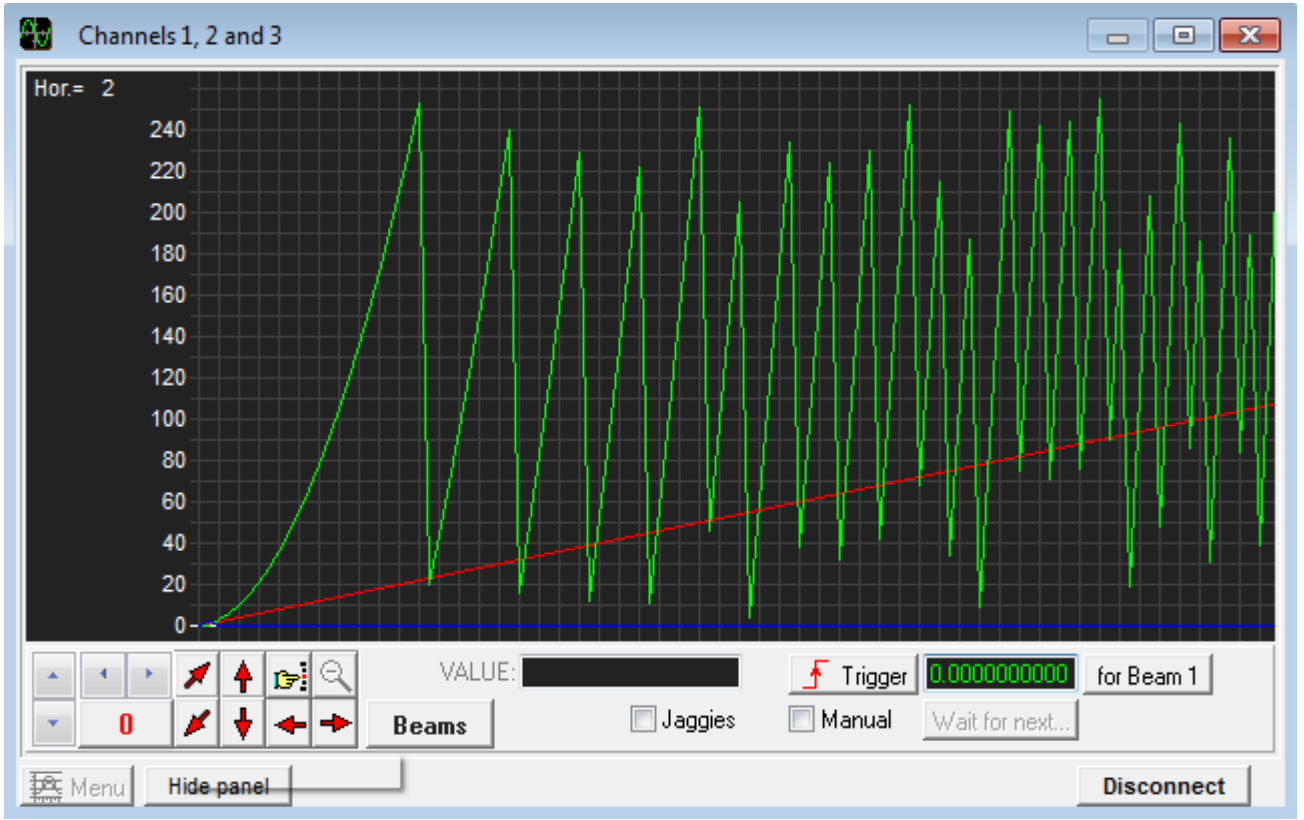


További lehetőség: Serial Oscilloscope

- A **Serial Oscilloscope** egy Windows alkalmazás, amely soros porton fogadja az adatokat (több csatorna esetén vesszővel elválasztva, sorokra tagolva) és megjeleníti azokat
- Egy primitív Arduino tesztprogram: scope_test.ino

```
byte val1 = 0;
byte val2 = 0;
void setup() {
    Serial.begin(115200);
}

void loop() {
   Serial.print(val1);
   Serial.write(",");
   Serial.println(val2);
   val1++;
   val2+=val1;
   delay(2);
}
```



Van még! - Oscilloscope Arduino-Processing

- Egy impozáns projekt az **Ocilloscope Arduino-Processing,** amely 4 csatornás oszcilloszkópot valósít meg. Sajnos, lassú!