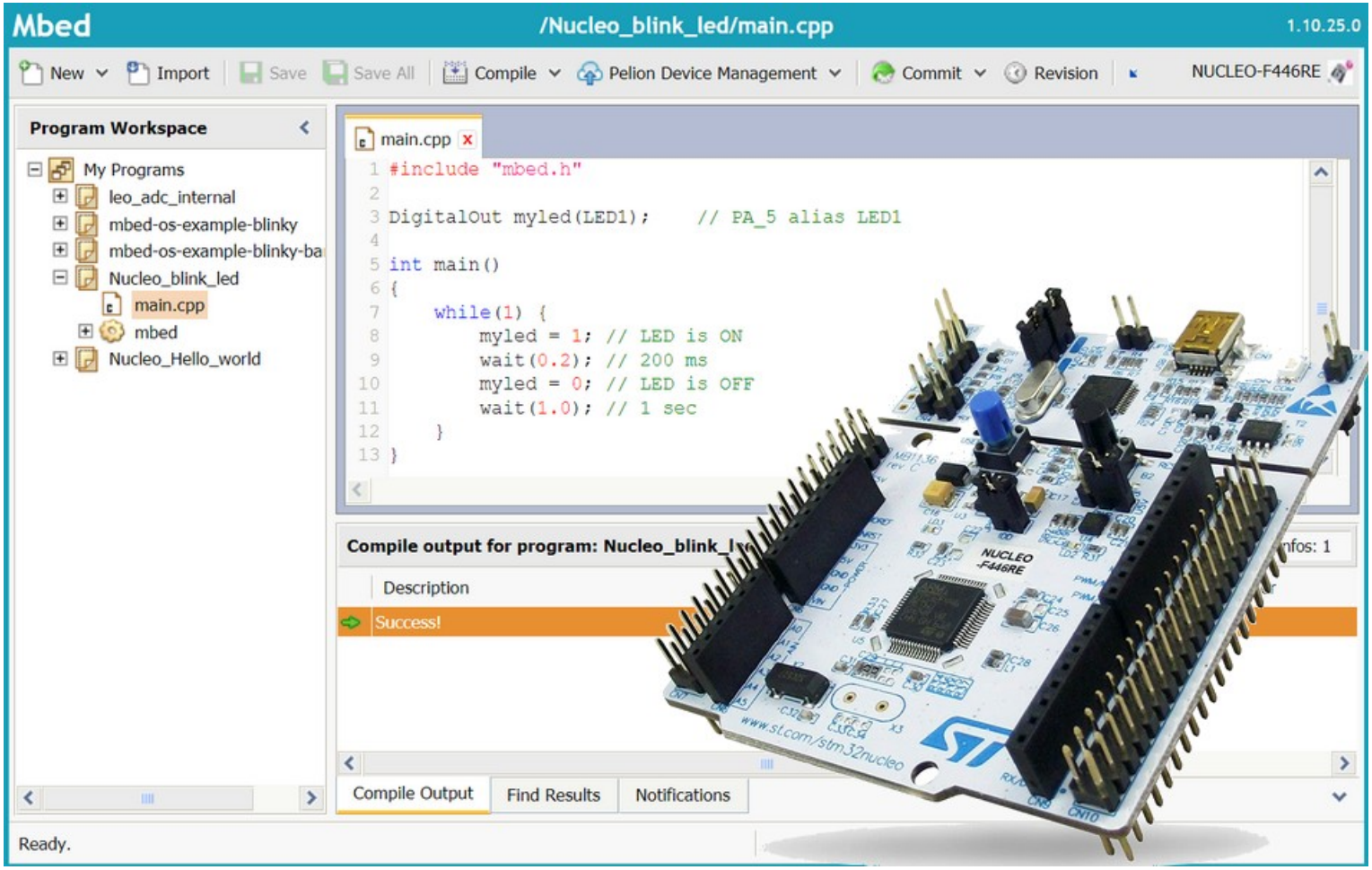
# STM32 mikrovezérlők programozása ARM mbed környezetben






## 6. OLED I2C kijelzők

# Felhasznált és ajánlott irodalom

- Cserny István: A FRDM-KL25Z kártya programozása mbed környezetben
- Rob Toulson, Tim Wilmhurst:
  Fast and Effective Embedded Systems Design: Applying the ARM mbed
- Perry Xiao: Designing Embedded Systems and the IoT eith ARM mbed
- Dogan Ibrahim: ARM-based Microcontroller Projects Using mbed
- **ARM mbed honlap: https://os.mbed.com/**
  - ARM mbed Compiler: https://ide.mbed.com/compiler/
  - ARM mbed 2 Handbook: https://os.mbed.com/handbook/Homepage
  - ARM mbed forráskód: https://github.com/ARMmbed/mbed-os

## Adatlapok:

- Solomon Sytech: **SSD1306 adatlap**
- Sino Wealth: **SH1106 adatlap**
- Bosch Sensortec: **BME280 adatlap**
- **STM32F446RE** adatlap és termékinfo
- **STM32F446** Family Reference Manual

# Példaprogramok

**OLED_SSD1306_SH1106** – az Adafruit_GFX SSD1306 programkönyvtár kiegészítése SH1106 I2C kijelző kezeléssel

**Lab06_oled_i2c** – OLED kijelzők kezelése

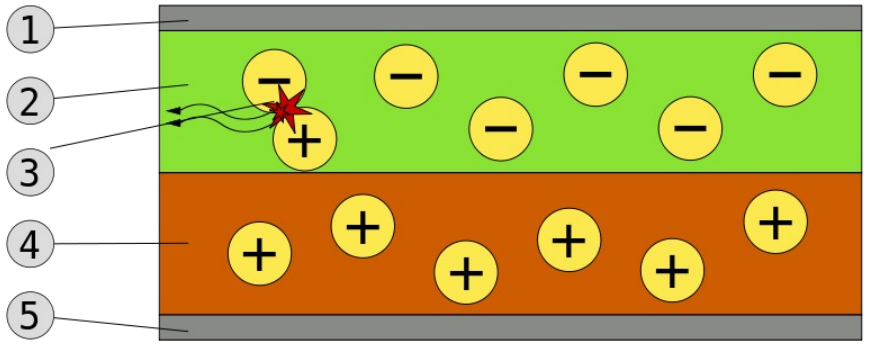
**Lab06_oled_clock** – óra, OLED kijelzővel

**Lab06_BME280_oled** – BME280 szenzor OLED kijelzővel

A mintaprogramok az os.mbed.com/users/cspista/code/ oldalon is megtalálhatók

# OLED kijelzők

- **OLED**-nek (*organic light-emitting diode*-nak) nevezzük a szerves fénykibocsátó diódát, amelyben  a fénykibocsátásért felelős elektrolumineszcens réteg szerves félvezető vegyület, mely elektromos áram hatására világít
- Az OLED leggyakrabban két elektróda között elhelyezkedő szerves vegyületrétegből áll (ez utóbbi lehet két, vagy többrétegű)
- Az aktív mátrix kijelzők (AMOLED) a kijelző mögött egy TFT mátrixot tartalmaznak, melynek segítségével minden képpont egyidejűleg vezérelhető



1. katód, 2. emissziós réteg, 3. fényemisszió,
4. vezető réteg 5. anód

- A kisebb felbontású kijelzőknél passzív mátrix (multiplex) kijelzés van – egyszerre csak egy sor vagy oszlop kap vezérlést

# OLED I2C kijelző SSD1306 vezérlővel

**Jellemzők:**


- OLED technológia
- **128x64** képpont
- 0,96” (2,4 cm) képátló
- 2.5V-5.5V tápfeszültség
- **SSD1306** vezérlő (I2C vagy SPI), I2C címe **0x3C** (vagy 0x3D)
- Monokróm (egyes változatoknál a felső harmad más színű)
- Grafikus megjelenítés
- Inverz mód
- Görgetés több irányba
- Dokumentáció: Solomon Systech SSD1306 adatlap

# OLED I2C kijelző SSD1306 vezérlővel

**Jellemzők:**


- **OLED** technológia
- **128x32** képpont
- 0,91” (2,3 cm) képátló
- I2C illesztő
- 2.5V-5.5V tápfeszültség
- **SSD1306** vezérlő, I2C címe **0x3C** (vagy 0x3D)
- Monokróm
- Grafikus megjelenítés
- Inverz mód
- Görgetés több irányba
- Dokumentáció: Solomon Systech SSD1306 adatlap

Kompatibilis az SSD1306 vezérlőjű 0.96” képátlójú, 128x64 felbontású kijelzővel, de az inicializálásban van néhány különbség

# OLED I2C kijelző SH1106 vezérlővel

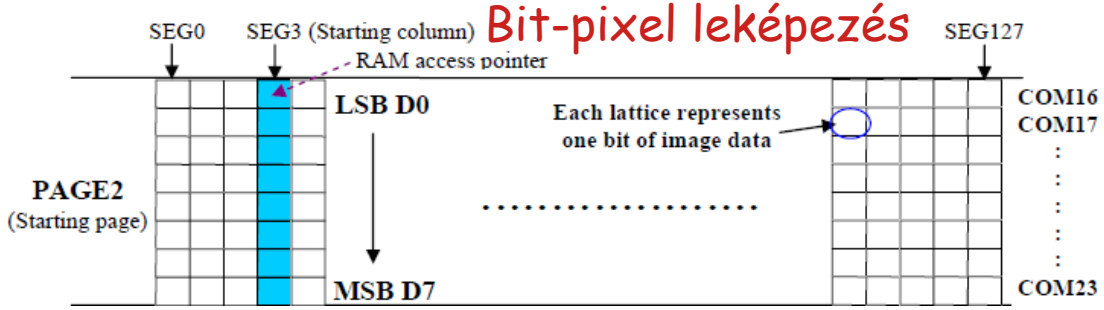
**Jellemzők:**


- OLED technológia
- 128x64 képpont (a vezérlő IC 132x64-at tudna!)
- 1.3” (3,3 cm) képátló
- I2C illesztő (létezik SPI változat is)
- 3,3V-5.0V tápfeszültség
- **SH1106** vezérlő, I2C címe **0x3C** (vagy 0x3D)
- Monokróm (egyes változatoknál a felső harmad más színű)
- Grafikus megjelenítés
- Inverz mód
- **Eltérések SSD1306-tól:** nincs görgetés, és 2 pixellel el van tolva a kép, a kivezetések sorrendje más (VCC és GND fordított sorrendű)
- Dokumentáció: Sino Wealth SH1106 datasheet

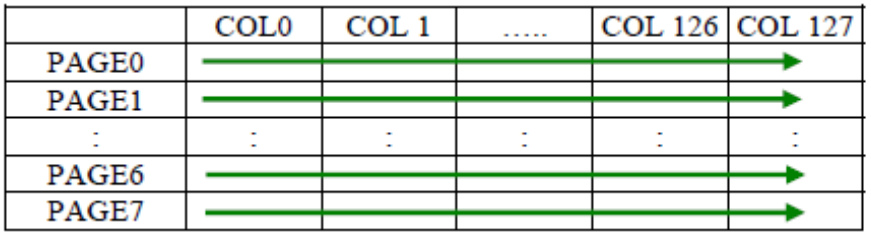
# SSD1306/SH1106 parancsok

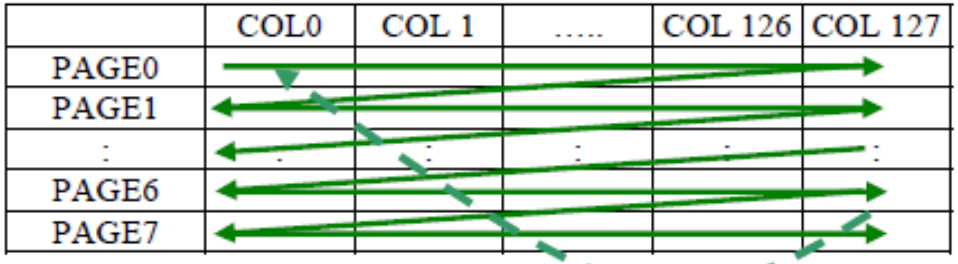
- Set Lower Column Start Address for Page Addressing Mode (00h~0Fh) az első oszlop címének alsó 4 bitje (ssd1306: 0000b, sh1106: 0010b)
- Set Higher Column Start Address for Page Addressing Mode (10h~1Fh) az első oszlop címének felső 4 bitje (általában 10h)
- Set Memory Addressing Mode (20h + adat[1:0]) – beállítja a címzésmódot 00: Horizontal, 01: Vertical, **10: Page,** 11: Invalid adressing mode (sh1106: csak a page addressing módot ismeri)

Bit-pixel leképezés

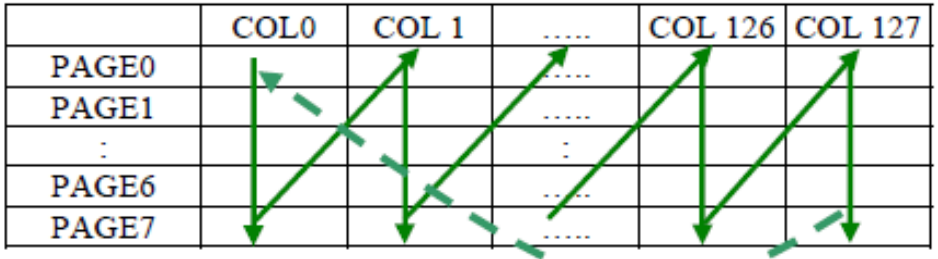



Page addressing




Horizontal addressing



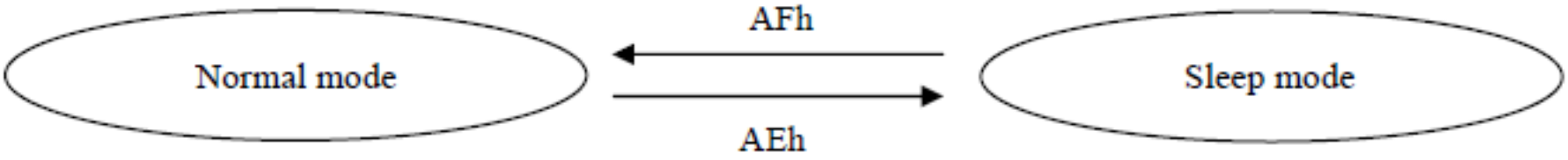

Vertical addressing

# SSD1306/SH1106 parancsok

- Set Column Address (21h + start + end) – kezdő és végző oszlopcím megadása Horizontal címzésmódhoz (csak SSD1306 estén)
- Set Page Address (22h + start + end) – kezdő és végző lapcím megadása Horizontal/Vertical címzésmódhoz (csak SSD1306 estén)
- Set Pump Voltage value (30H~33H) – a Vpp értékét állítja be (30: 6.4V, 31: 7.4V, **32: 8.0V** (POR), 33: 9.0V) (csak SH1106 esetén!)
- Set Display Start Line (40h~7Fh) – a kezdősor megadása (POR=40h)
- Set Contrast Control (81h + data) – kontraszt beállítás
- **Set Segment Re-map** (A0h/A1h) – a leképezést állítja be (POR=A0h)
  **A0:** col. addr 0 ⟶ segment 127, **A1:** col. addr 127 ⟶ segment 0
- Entire Display ON (A4h/A5h) – az egész képernyő bekapcsolása
  **A4:** normál módba, **A5:** minden képpont kigyújtása
- Set Normal/Inverse Display (A6h/A7h) – normál/inverz mód

# SSD1306/SH1106 parancsok

- Set Multiplex Ratio (A8h + 0Fh~3Fh) – multiplexelt sorok számának megadása, com0 - com63 közül ennyit kapcsolgatunk a COM jelre
- Set DC-DC ON/OFF (ADh + 8Ah/8Bh) – a DC-DC konverter ki/be kapcsolása **8A:** ki, **8B:** be (POR) (csak SH1106 esetén)
- Set DC-DC ON/OFF (8Dh + 10h/14h) – a DC-DC konverter ki/be kapcsolása **10:** ki, 14: be (POR) (csak SSD1306 esetén)
- Set Display ON/OFF (AEh/AFh) – a kijelző ki és bekapcsolása Konfigurálás elején is ki kell kapcsolni!




| DC-DC STATUS | DISPLAY ON/OFF STATUS | Description |
|---|---|---|
| 0 | 0 | Sleep mode |
| 0 | 1 | External $V_{PP}$ must be used. |
| 1 | 0 | Sleep mode |
| 1 | 1 | Built-in DC-DC is used, Normal Display |

# SSD1306/SH1106 parancsok

- Set Page Address for Page Addressing Mode (B0h~B7h) – a lap címét adja meg (B0~B7: 64 soros, B0~B3: 32 soros kijelzőhöz)
- **Set COM Output Scan Direction** (C0h/C8h) – a sorok sorrendjét lehet megfordítani **C0:** com0 ⟶ comN, **C8:** comN ⟶ com0
  Képernyő forgatás: **A0**+**C0** normál, illetve **A1**+**C8** „fejre állított”



Az Adafruit könyvtárakban ez az alapértelmezett állás

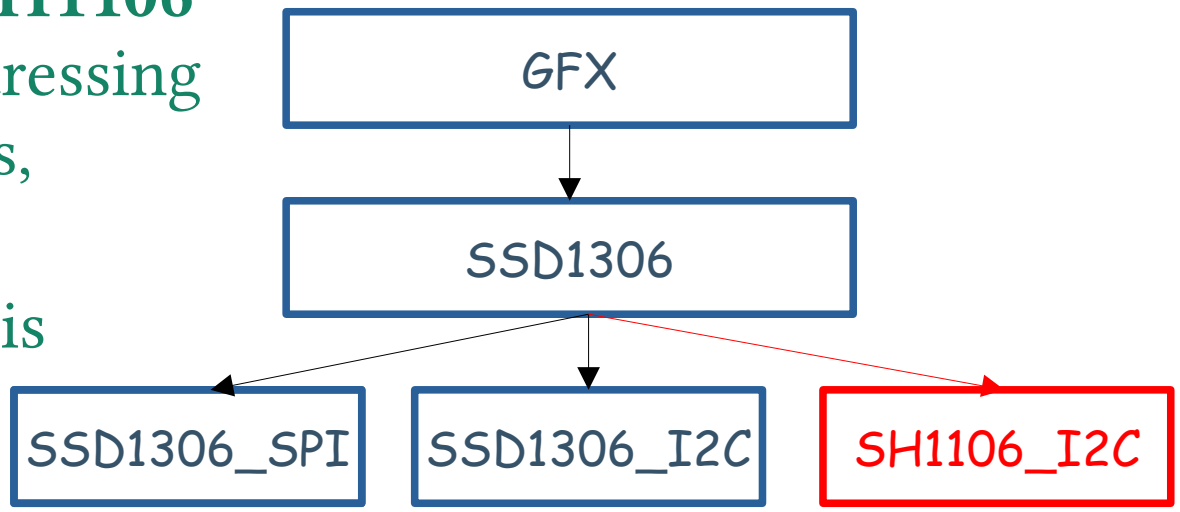
- Set Display Offset (D3h + data) – sorok függőleges eltolása (normálisan 0 eltolást használunk)
- Set Display Clock Divide Ratio/ Oscillator Frequency (D5h + data) oszcillátor frekvencia bit[7:4] és leosztás bit[3:0] megadása
  (a POR érték SSD1306: 80h, SH1106: 50h)
- Set Dis-charge/Pre-charge Period (D9h data) – a képpont kioltás/felkapcsolás ideje DCLK egységben (POR data = 22h)

# SSD1306/SH1106 parancsok

- Set COM Pads Hardware Configuration (DAh + 02h/12h) – a com sorvezérlők aktiválási sorrendje (02: szekvenciális, 12: alternáló) (64 soros kijelzőnél 12h, 32 soros kijelzőnél 02h értéket használunk) Megjegyzés: SSD1306 esetén az 5. bit is használható: COM Left/Right remap enable, így 22h és 32h paraméter is használható, ha a hardver ezt igényli
- Set $V_{COMH}$ Deselect Level (DBh + data) – a $V_{COMH}$ feszültséget állítja be (POR érték SSD1306: 20h, 0.77*Vcc, SH1106: 35h, 0.77*Vref)

# Az Adafruit_GFX programkönyvtár

- A kiindulásként használt Adafruit_GFX programkönyvtár az **ARM Mbed Components** szekcióban található, *Neal Horman* adaptálta
- Az általunk átdolgozott könyvtár új neve: **OLED_SSD1306_SH1106**
- A **GFX** absztrakt grafikus programkönyvtár mellett az SPI vagy I2C csatolófelülettel ellátott, **SSD1306** vezérlőjű 128x32 vagy 128x64 felbontású OLED kijelzőket támogatja
- A hierarchikus felépítés lehetővé tette, hogy egy új alosztályt definiáljunk, amely az I2C **SH1106** kijelzőt támogatja (page addressing memória mód, 2 pixel eltolás, speciális parancsok)
- Az **SSD1306 I2C** kijelzőnél is áttértünk a page addressing módú kezelésre

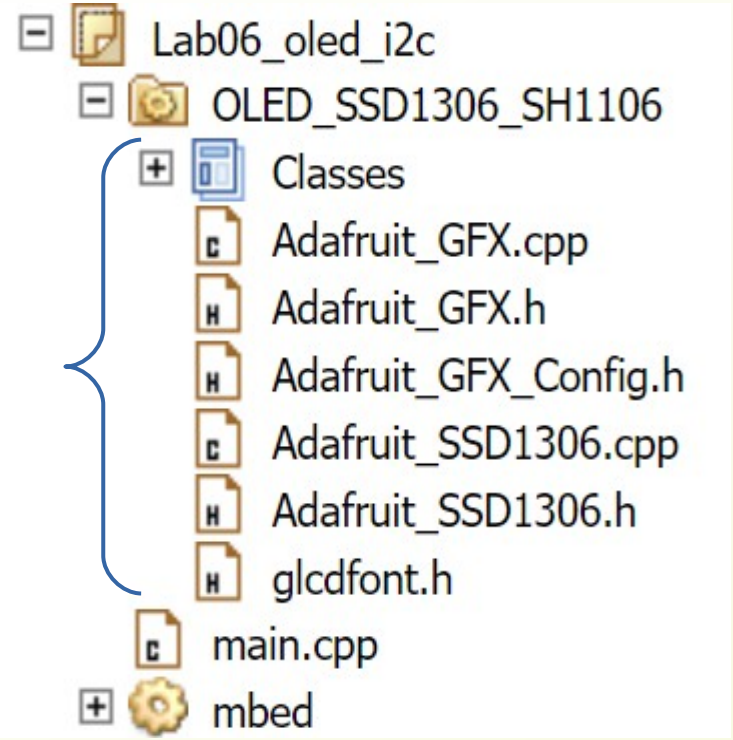


# Az inicializálás lépései

- Az **SH1106** és az **SSD1306** kijelzők inicializálásának lépéseit már korábban is tárgyaltuk (pl. 2021. február 18-án):
  **9. Az I2C kommunikációs csatorna** - 2. rész
  (előadásvázlat, mintaprogramok, video)
- A módosítások az **Adafruit_SSD1306.h** állományban találhatók
- **Adafruit_SSD1306_I2c osztály:**
  - Inicializálás után beállítjuk a Page addressing módot
  - A **sendDisplayBuffer()** virtuális függvényt átírtuk a Page addressing mód használatához
- **Adafruit_SH1106_I2c osztály:**
  - Az Adafruit_SSD1306_I2C mintájára hoztuk létre
  - Lecseréltük benne az inicializálást az SH1106 vezérlőhöz igazodva
  - A **sendDisplayBuffer()** virtuális függvényt átírtuk az SH1106-hoz igazodva (a kezdő oszlopcím itt 2, nem pedig nulla)

# A programkönyvtár forrásállományai

- Az **OLED_SSD1306_SH1106** programkönyvtárnál megőriztük az **Adafruit_GFX** programkönyvtár eredeti fájlszerkezetét s a módosítások, bővítések csak az **SSD1306.h** állományt érintik
- A fájlszerkezet (némi magyarázattal) az alábbi ábrán látható




Dokumentáció
Absztrakt grafika függvénydefiníciók
Absztrakt grafika függvénydeklarációk
Konfigurációs beállítások
OLED kijelzőt kezelő osztály implementáció
**OLED kijelző interfész osztályok (SPI, I2C)**
Standard 5x7 font

# Adafruit_SSD1306_I2C módosítása

```
Adafruit_SSD1306_I2c(I2C &i2c, PinName RST, uint8_t i2cAddress = SSD_I2C_ADDRESS,
uint8_t rawHeight = 32, uint8_t rawWidth = 128) : Adafruit_SSD1306(RST, rawHeight,
rawWidth) ,
   mi2c(i2c) ,
   mi2cAddress(i2cAddress) {
   begin();
   command(0x20);        // SSD1306_MEMORYMODE
   command(0x10);        // Page addressing mode (compatible with SH1106)
   splash();
   display();
};

protected:
    virtual void sendDisplayBuffer() {
        char buff[256];
        char cmd[4] = {0, 0xB0, 0x00, 0x10};
        for (uint8_t m = 0; m < _rawHeight/8; m++) {
            buff[0] = 0x40;     // Send data
            cmd[1] = 0xB0 + m;  // Set Page Address
            for(int i=0; i<128; i++) {
                buff[i+1]= buffer[m*128+i];
            }
            mi2c.write(mi2cAddress, cmd, 4);
            mi2c.write(mi2cAddress, buff, 129);
        }
    };
```

# Adafruit_SH1106_I2C létrehozása

```
Adafruit_SH1106_I2c(I2C &i2c, PinName RST, uint8_t i2cAddress = SSD_I2C_ADDRESS,
uint8_t rawHeight = 32, uint8_t rawWidth = 128) :
   Adafruit_SSD1306(RST, rawHeight, rawWidth) ,
   mi2c(i2c) ,
   mi2cAddress(i2cAddress) {
   begin2();       // New initialization sequence was introduced
   splash();
   display();
};

protected:
    virtual void sendDisplayBuffer() {
        char buff[256];
        char cmd[4] = {0, 0xB0, 0x02, 0x10};  // Column addresses start at 2!
        for (uint8_t m = 0; m < _rawHeight/8; m++) {
            buff[0] = 0x40;      // Send data
            cmd[1] = 0xB0 + m;  // Set Page Address
            for(int i=0; i<128; i++) {
                buff[i+1]= buffer[m*128+i];
            }
            mi2c.write(mi2cAddress, cmd, 4);
            mi2c.write(mi2cAddress, buff, 129);
        }
    };
```

# Az SH1106 inicializálása

```
void begin2(void) {
    command(0xAE); //--display off
    command(0xA1); //--set segment re-map 127 to 0
    command(0xC8); //--Set COM Out Scan Dir 63 to 0
    command(0x32); //--set pump voltage value to 8.0V (SH1106 only)
    command(0x40); //--set start line address
    command(0x81); //--set contrast control register
    command(0x80); //  POR value = 80
    command(0xA4); //--set normal mode (A5: test mode)
    command(0xA6); //--set normal display (i.e non-inverted mode)
    command(0xA8); //--set multiplex ratio(1 to 64)
    command(0x3F); //
    command(0xAD); //--set DC-DC mode
    command(0x8B); //  DC-DC converter ON
    command(0xD3); //--set display offset
    command(0x00); //  no offset
    command(0xD5); //--set display clock divide ratio/oscillator frequency
    command(0x50); //  set frequency and divide ratio
    command(0xD9); //--set dis-charge/pre-charge period
    command(0x22); //
    command(0xDA); //--set com pins hardware configuration
    command(0x12);
    command(0xDB); //--set vcomh
    command(0x35); // SH1106: 0x35, SSD1306: 0x20 0.77xVcc
    command(0xAF); //--turn on SSD1306 panel
};
```

Az Adafruit könyvtárakban alapértelmezett állás

# Adafruit_GFX_Config.h

- A konfigurációs állományban a projekt lefordítása előtt engedélyezhetjük ill. letilthatjuk a programkönyvtár egyes elemeit (splash kép, absztrakt rajzoló függvények, skálázható font)

```
#ifndef _ADAFRUIT_GFX_CONFIG_H_
#define _ADAFRUIT_GFX_CONFIG_H_

// Uncomment this to turn off the builtin splash
#define NO_SPLASH_ADAFRUIT

// Uncomment this to enable all functionality
#define GFX_WANT_ABSTRACTS

// Uncomment this to enable only runtime font scaling,
// without all the rest of the Abstracts
// #define GFX_SIZEABLE_TEXT


#endif
```

# Az interfészek használata

- A „gyári mintapélda” leegyszerűsítve így nézne ki (*mosi, miso, clk, dc, rst* és *cs* helyére a használni kívánt kivezetések nevét kell megadni)

```
#include "mbed.h"
#include "Adafruit_SSD1306.h"

SPI spi(mosi,miso,clk);
Adafruit_SSD1306_Spi oled1(spi,dc,rst,cs,64,128);

I2C i2c(I2C_SDA, I2C_SCL);
Adafruit_SH1106_I2c oled2(i2c,NC,0x3c<<1,32,128);

int main() {
    uint16_t x=0;
    oled1.printf("%ux%u OLED Display\r\n", oled1.width(), oled1.height());
    oled2.printf("%ux%u OLED Display\r\n", oled2.width(), oled2.height());
    while(1) {
        oled1.printf("%u\r",x);
        oled1.display();
        oled2.printf("%u\r",x);
        oled2.display();
        x++;
        wait(1.0);
    }
}
```

A kommunikációs objektumpéldány referenciáját adjuk át

# Lab06_oled_i2c – egyszerű bemutató

- A programban bemutatjuk az I2C csatoló felülettel rendelkező **SSD1306** (128x32 vagy 128x64), ill. **SH1106** (128x64) kijelző egyszerű használatát
- A program sikeres lefordításához az **Adafruit_GFX_Config.h** állományban engedélyezni kell az absztrakt grafikus függvényeket
- main.cpp – részlet:

```
#include "mbed.h"
#include "Adafruit_SSD1306.h"

I2C i2c(D14,D15);
Adafruit_SH1106_I2c oled(i2c, NC, 0x78, 64, 128);     // SH1106  I2C 128x64, no reset
// Adafruit_SSD1306_I2c oled(i2c, NC, 0x78, 64, 128); // SSD1306 I2C 128x64, no reset
// Adafruit_SSD1306_I2c oled(i2c, NC, 0x78, 32, 128); // SSD1306 I2C 128x32, no reset
```

0x78 = 0x3C << 1

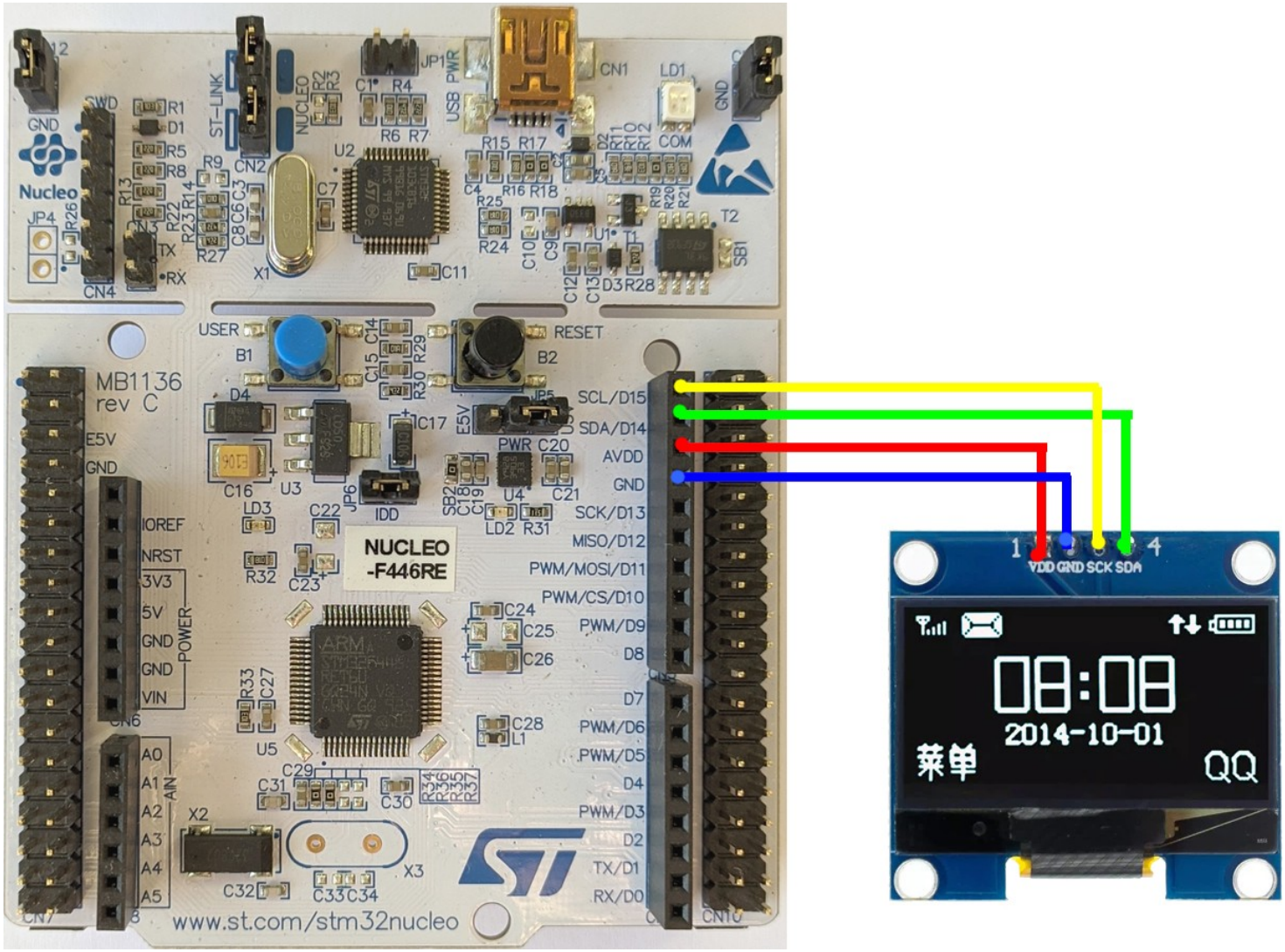
- Hardveres reset hiányában NC (no connection) is megadható az **RST** kivezetés helyén
- Az utolsó három paraméter opcionális (van alapértelmezett értékük)

# Lab06_oled_i2c /main.cpp – folytatás

```
int main() {
    uint16_t x=0;
    i2c.frequency(400000);
    oled.setRotation(0);   // 0, 1, 2, 3          WHITE
    oled.clearDisplay();                          ↓
    oled.drawRect(0,0,oled.width(),oled.height(),1);
    oled.display();
    oled.setTextColor(WHITE);
    oled.setTextSize(1);
    oled.setTextCursor(10,8);
    oled.printf("SH1106 %ux%u", oled.width(), oled.height());
    oled.display();
    wait(2.0);
    oled.setTextSize(2);
    while(1) {
        oled.clearDisplay();
        oled.drawRect(0, 0, oled.width(), oled.height(),1);
        oled.setTextCursor(10,8);
        oled.printf("x = %u", x++);
        oled.display();
        wait(1.0);
    }
}
```

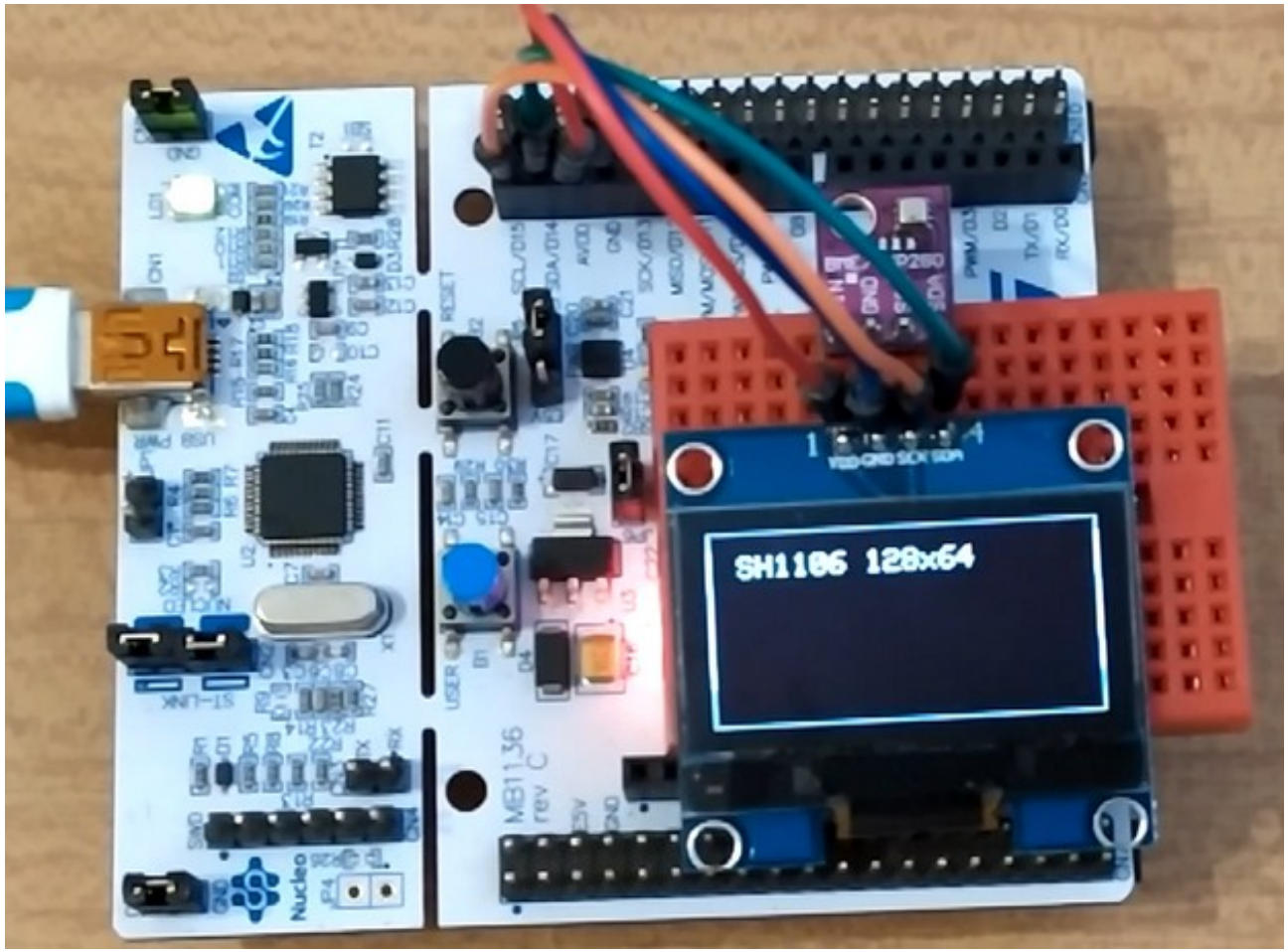
# Bekötési vázlat

- Az alapértelmezett I2C kivezetések: I2C_SDA=D14, I2C_SCL=D15
  Az I2C busz felhúzó ellenállásai be vannak építve a kijelzőn kártyán



# Lab06_oled_i2c futási eredménye

- Az ábrán az **SH1106** 128x64 I2C kijelző kipróbálása látható



# Lab06_oled_clock

- Keressük elő az STM32 Mbed tanfolyam 2021. november 11-i, **4. Programmegszakítások, időzítők** c. előadásából a Lab04_rtc_time mintapéldát és egészítsük ki OLED kijelzéssel!
- main.cpp (részlet)

```
#include "mbed.h"
#include "Adafruit_SSD1306.h"

I2C i2c(D14,D15);
Adafruit_SH1106_I2c oled(i2c, NC, 0x78, 64, 128);     // SH1106  I2C 128x64
Serial pc(USBTX,USBRX);           // UART via OpenSDA
time_t seconds;
Ticker myticker;
static char msg[20];              // character buffer
volatile uint8_t myflag = 0;      // Ticker event flag
#define DATE_20200222_222222      1582377742              // 2020/2/22 22:22:22

void processSerialCommand();

void setflag(void) {              // Ticker callback
    myflag = 1;                   // Set Ticker event flag
}
```

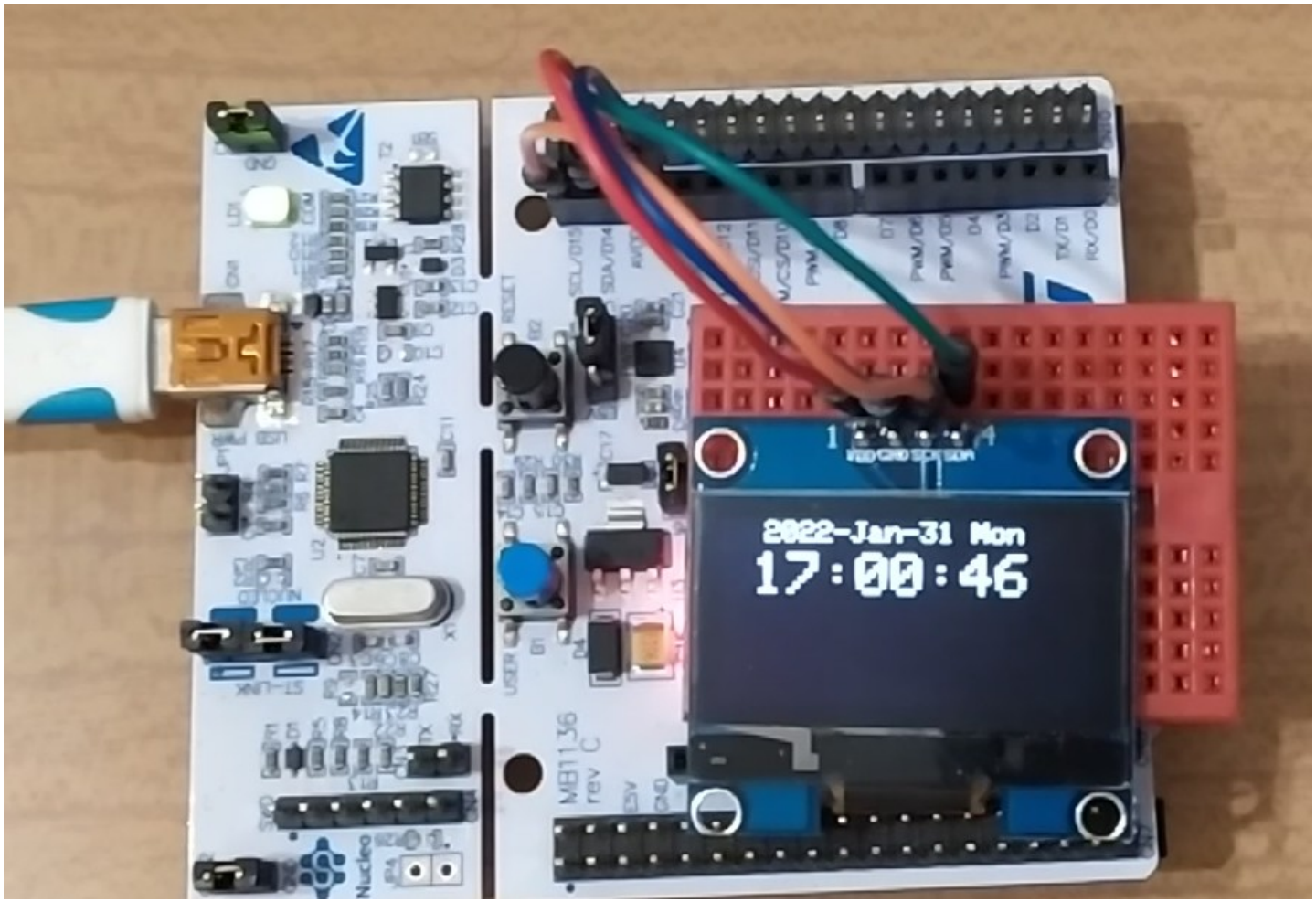
# Lab06_oled_clock/main.cpp (folytatás)

```
int main() {
    pc.baud(115200);
    seconds = time(NULL);
    if (seconds < DATE_20200222_222222) { set_time(DATE_20200222_222222); }
    i2c.frequency(400000);
    oled.setRotation(0);
    oled.clearDisplay();
    oled.display();
    myticker.attach(&setflag,1); // Create Ticker even in each 1 second
    while(1) {
        if (pc.readable()) { processSerialCommand(); }
        if (myflag) {
            seconds = time(NULL);
            oled.clearDisplay();
            oled.setTextSize(1);
            oled.setTextCursor(14,1);
            strftime (msg, 15, "%Y-%b-%d %a ", localtime(&seconds));  // Display date
            oled.printf("%s",msg);
            strftime (msg, 10, "%T ", localtime(&seconds));           // Display time
            oled.setTextSize(2); oled.setTextCursor(10,12);
            oled.printf("%s",msg);
            oled.display();
            pc.printf("RTC time: %s\r\n",ctime(&seconds));
            myflag = 0;             // Clear event flag
        }
    }
}
```

A **processSerialCommand()** függvényt itt nem részletezzük, ugyanaz maradt, mint az eredeti programban

# Lab06_oled_clock futási eredménye

- A program futási eredménye az ábrán látható
- Az óra beállítása a soros porton küldött **`Tyymmddhhmmss`** paranccsal történhet, ugyanúgy, mint az eredeti programban



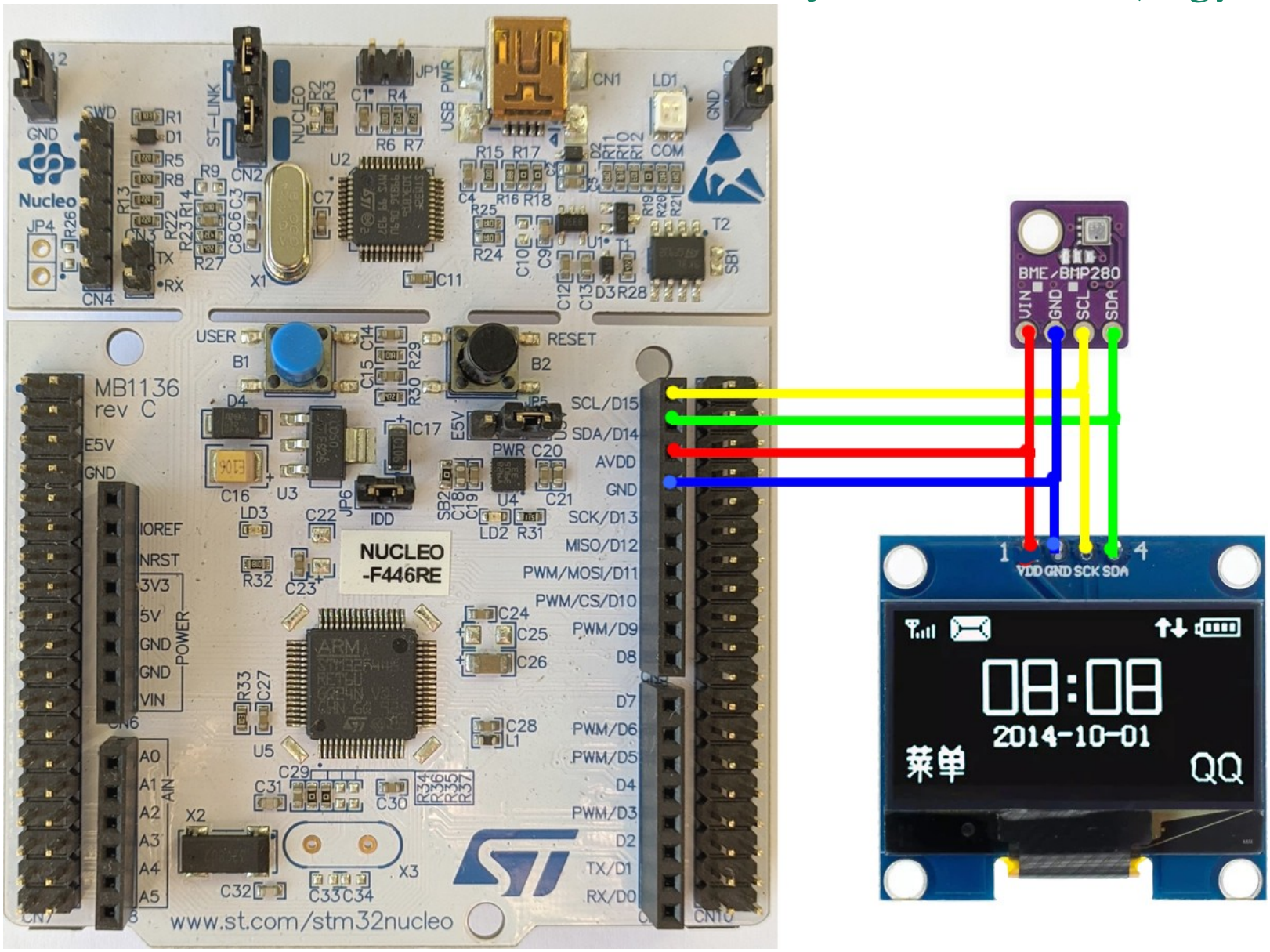

A kijelzés elrendezését az SSD1306 128x32 kijelzőhöz méreteztük

# A BME280 szenzor

- Az otthoni környezet monitorozásához jól használható a **BME280** szenzor, amellyel hőmérsékletet, légnyomást és páratartalmat mérhetünk
- Az I2C csatolófelületű **BME280** szenzor kezelésére **Toyomasa Watarai** BME280 programkönyvtárát használjuk fel
- **Konstruktorok:**
  - **BME280**(PinName sda, PinName sck, char slave_adr =0x76<<1)
  - **BME280**(I2C &i2c_obj, char slave_adr =0x76<<1)
- **Metódusok:**
  - void **initialize**(void) – inicializálás
  - float **getPressure**(void) – nyomás kiolvasása hPa egységben
  - float **getHumidity**(void) – relatív páratartalom kiolvasása %-ban
  - float **getTemperature**(void) – hőmérséklet kiolvasása °C-ban

# Bekötési vázlat

- A **BME280** szenzort is az **I2C** buszra kötjük, címe **0x76** (vagy 0x77)



# Gyári mintapélda a szenzor kipróbálására

- Az alábbi mintaprogrammal a soros porton íratjuk ki az eredményt
- A projektbe importálnunk kell az **mbed 2** és a **BME280** könyvtárt

```
#include "mbed.h"
#include "BME280.h"

Serial pc(USBTX, USBRX);
BME280 sensor(I2C_SDA, I2C_SCL);

int main() {
    pc.baud(115200);
    sensor.initialize();      // Ez a sor valójában fölösleges
    while(1) {
        pc.printf("%2.2f degC, %04.2f hPa, %2.2f %%\n", sensor.getTemperature(),
                  sensor.getPressure(), sensor.getHumidity());
        wait(1);
    }
}
```

# Lab06_BME280_oled/main.cpp

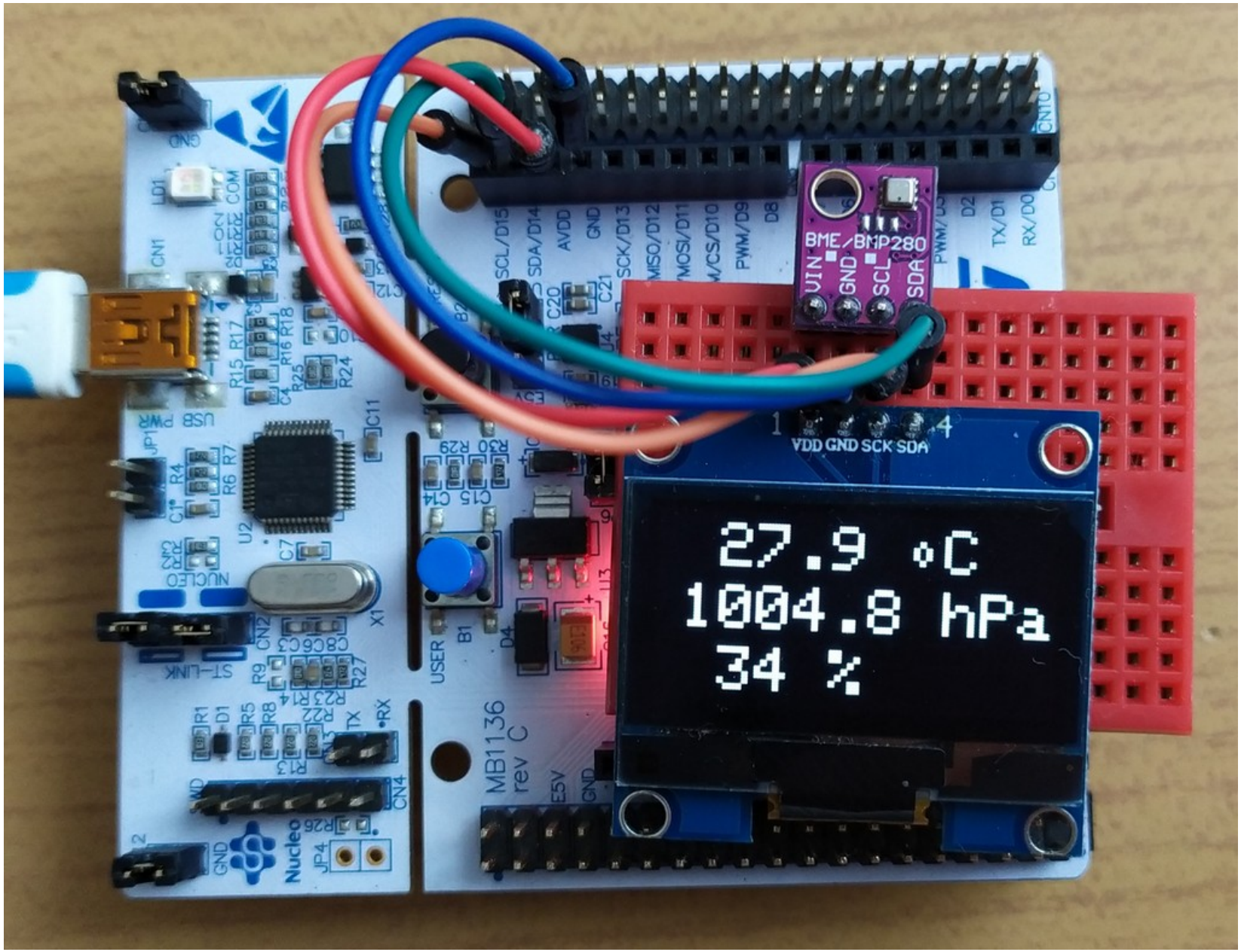
```
#include "mbed.h"
#include "BME280.h"
#include "Adafruit_SSD1306.h"
I2C i2c(D14,D15);
BME280 sensor(i2c, 0x76<<1);
Adafruit_SH1106_I2c oled(i2c, NC, 0x3C<<1, 64, 128);     // SH1106  I2C 128x64

int main() {
    char fok = 9;
    i2c.frequency(400000);
    oled.setRotation(0);  oled.clearDisplay();      oled.setTextColor(WHITE);
    oled.setTextSize(2);  oled.setTextCursor(10,8);
    oled.printf("BME280 \r\n  demo");
    oled.display();
    wait(5.0);
    while(1) {
        oled.clearDisplay();
        oled.setTextCursor(20,4);
        oled.printf("%.1f %cC",sensor.getTemperature(),fok);
        oled.setTextCursor(8,24);
        oled.printf("%.1f hPa",sensor.getPressure()/0.985);
        oled.setTextCursor(20,44);
        oled.printf("%.0f %%",sensor.getHumidity());
        oled.display();
        wait(2);
    }
}
```

Itt az i2c objektum referenciáját adjuk át a konstruktor függvényeknek

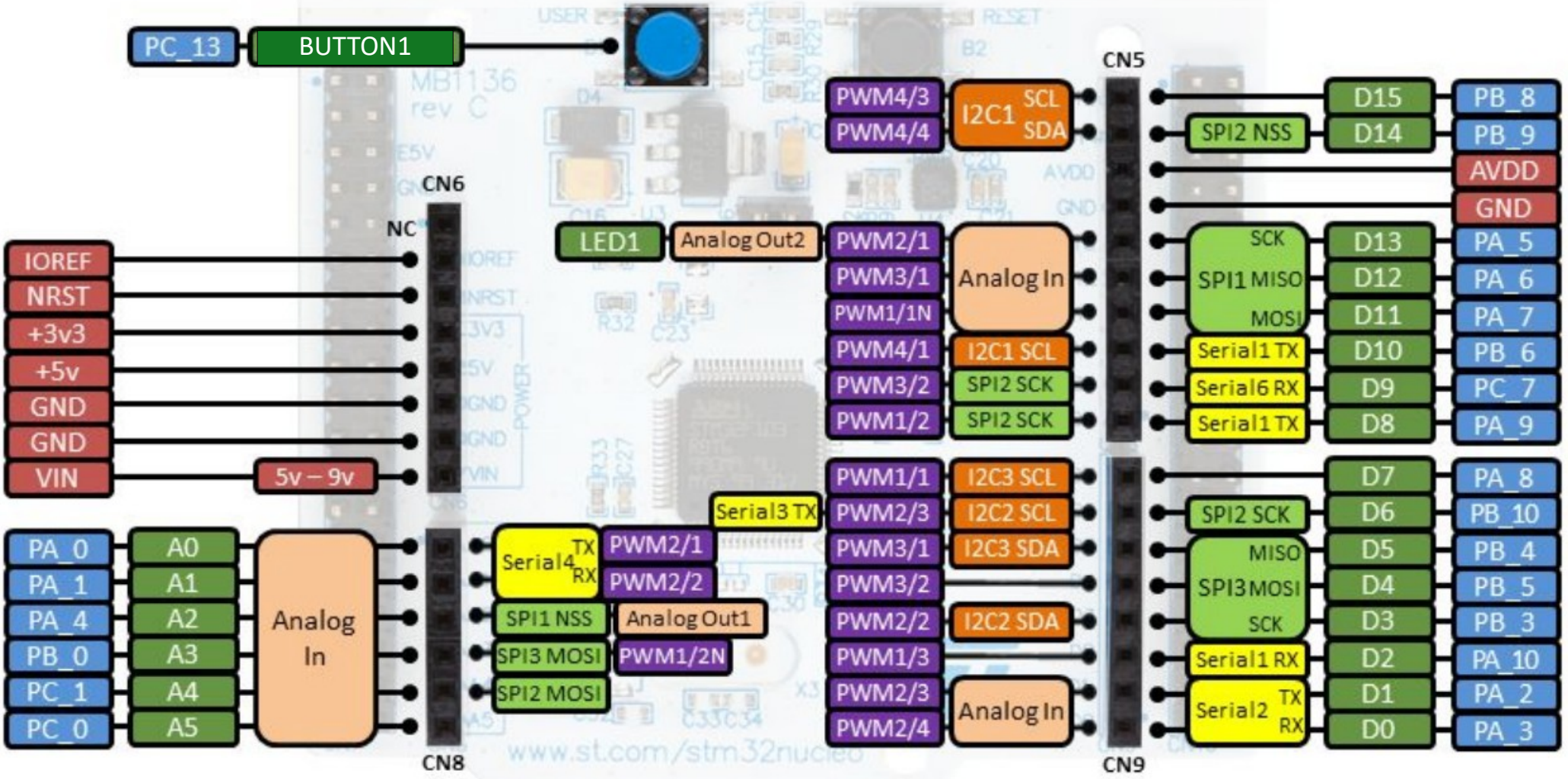
Átszámítás tengerszinti nyomásra ~ 140 m esetén

# Lab06_BME280_oled futási eredménye



# Arduino kompatibilis kivezetések

- Kivezetés azonosításra a kék, illetve a sötétzöld címkék használhatók (pl. PA_5, D13, vagy LED1)



# Morpho kivezetések

**Nucleo F446RE**

**Morpho Headers**