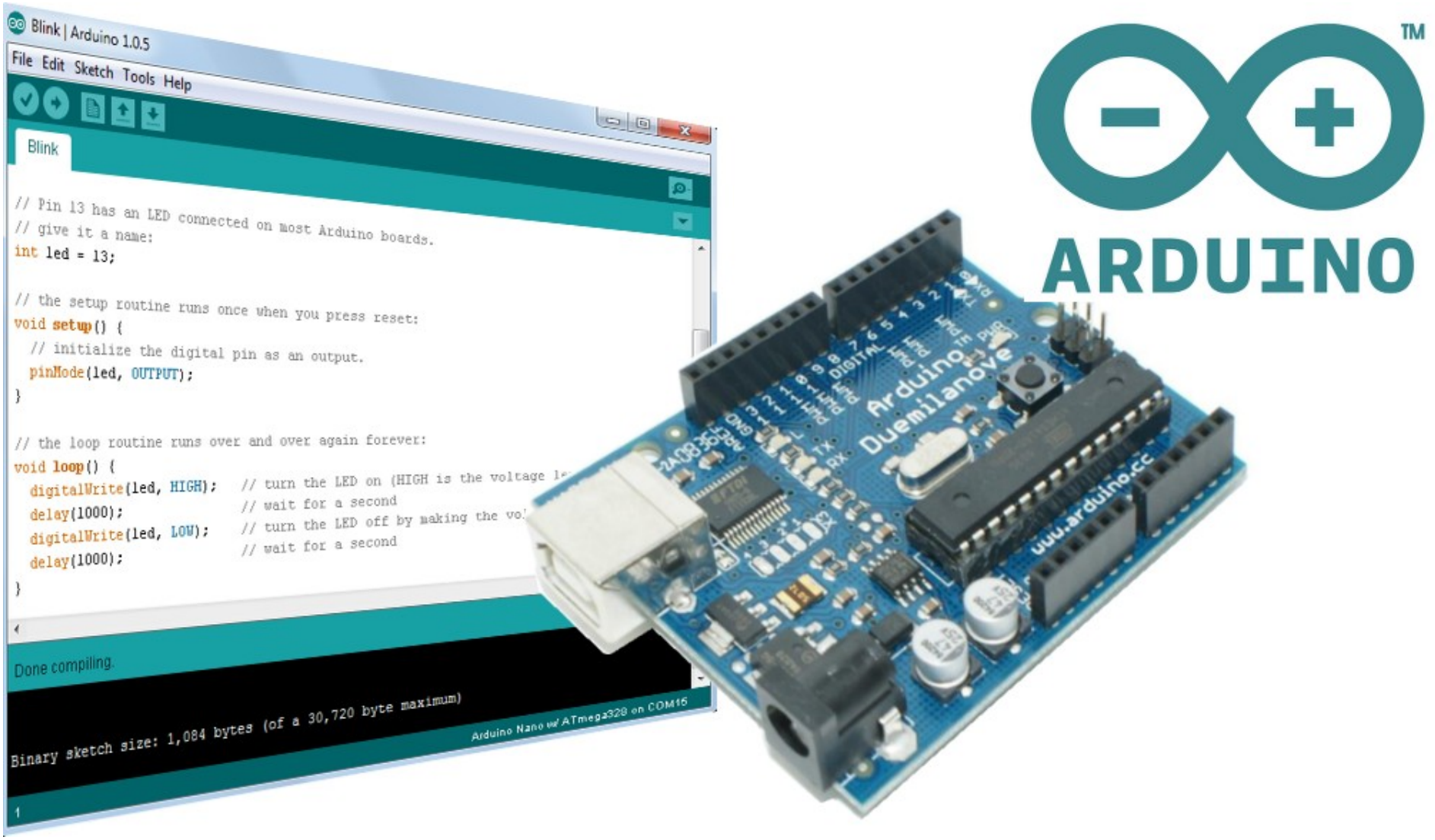
Arduino tanfolyam kezdőknek és haladóknak



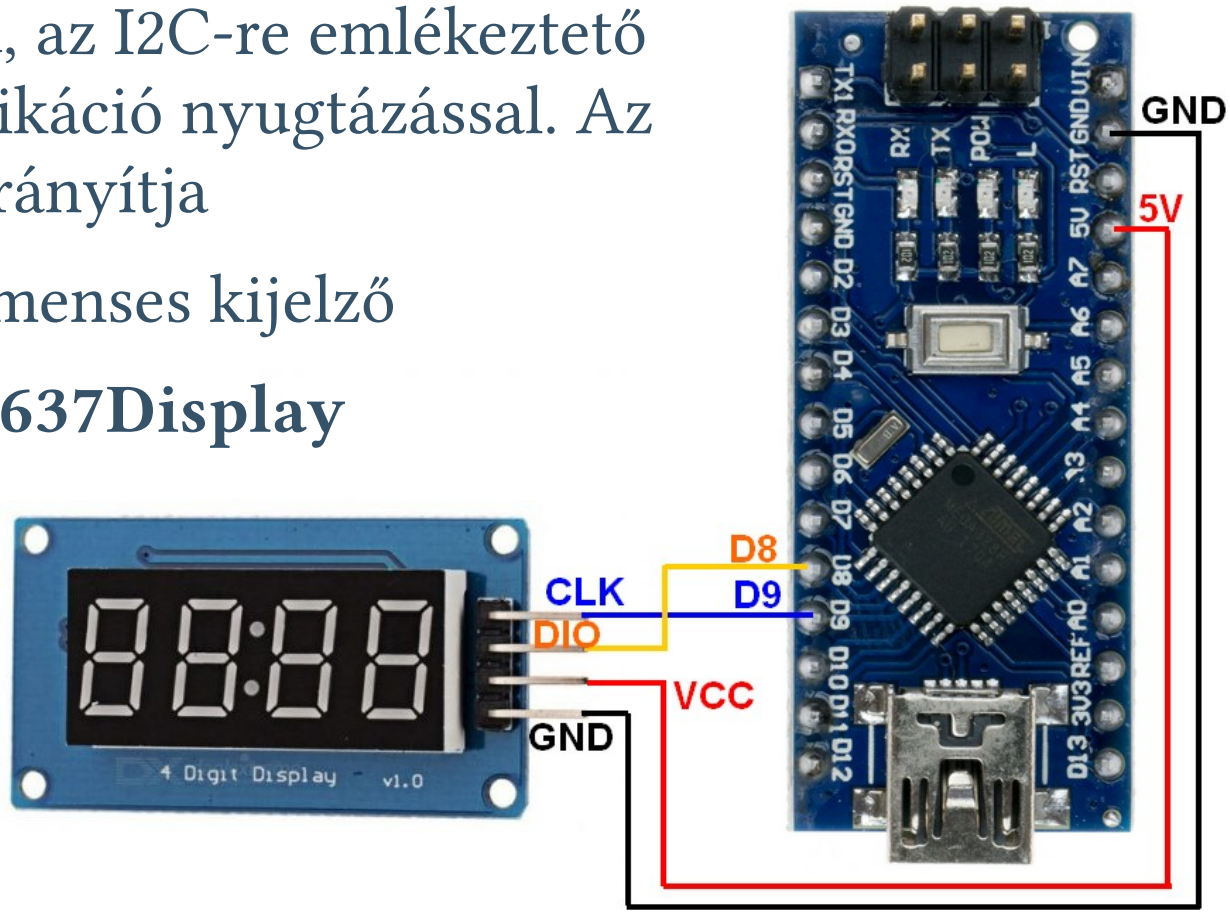
16. Egyszerű menüvezérlés, óra projektek

Ajánlott olvasmányok

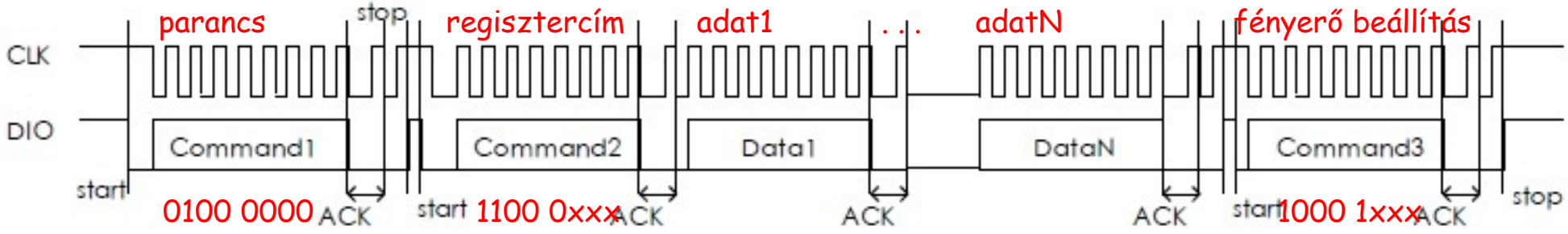
- **TM1637 4-digites kijelző:**
  - Titan Microelectronics: TM1637 adatlap
  - Hétszegmenses LED kijelzők 2. rész (2019. május 2.)
    előadásvázlat, mintaprogramok
  - Hétszegmenses kijelző alkalmazások (2019. május 16.)
    előadásvázlat mintaprogramok
- **DS3231 RTC modul:**
  - Maxim Integrated: DS3231 adatlap
  - A DS1307 és DS3231 valósidejű órák használata (2016. március 10.)
    előadásvázlat, mintaprogramok
- **SSD1306 I2C OLED kijelző:**
  - SSD1306 adatlap
  - Ultrahangos távolságmérés, OLED kijelzők (2020. február 20.)
    előadásvázlat, mintaprogramok

Emlékeztető: TM1637 4-jegyű kijelző

- Két vezetékes, kétirányú, az I2C-re emlékeztető szinkron soros kommunikáció nyugtázással. Az átvitelt a mikrovezérlő irányítja
- Négy számjegy, hétszegmenses kijelző
- Programkönyvtár: **TM1637Display** (Avishay Orpaz)
- Telepítés: Arduino IDE Tools/Manage Libraries menüpontban



Write SRAM data in address auto increment 1 mode.



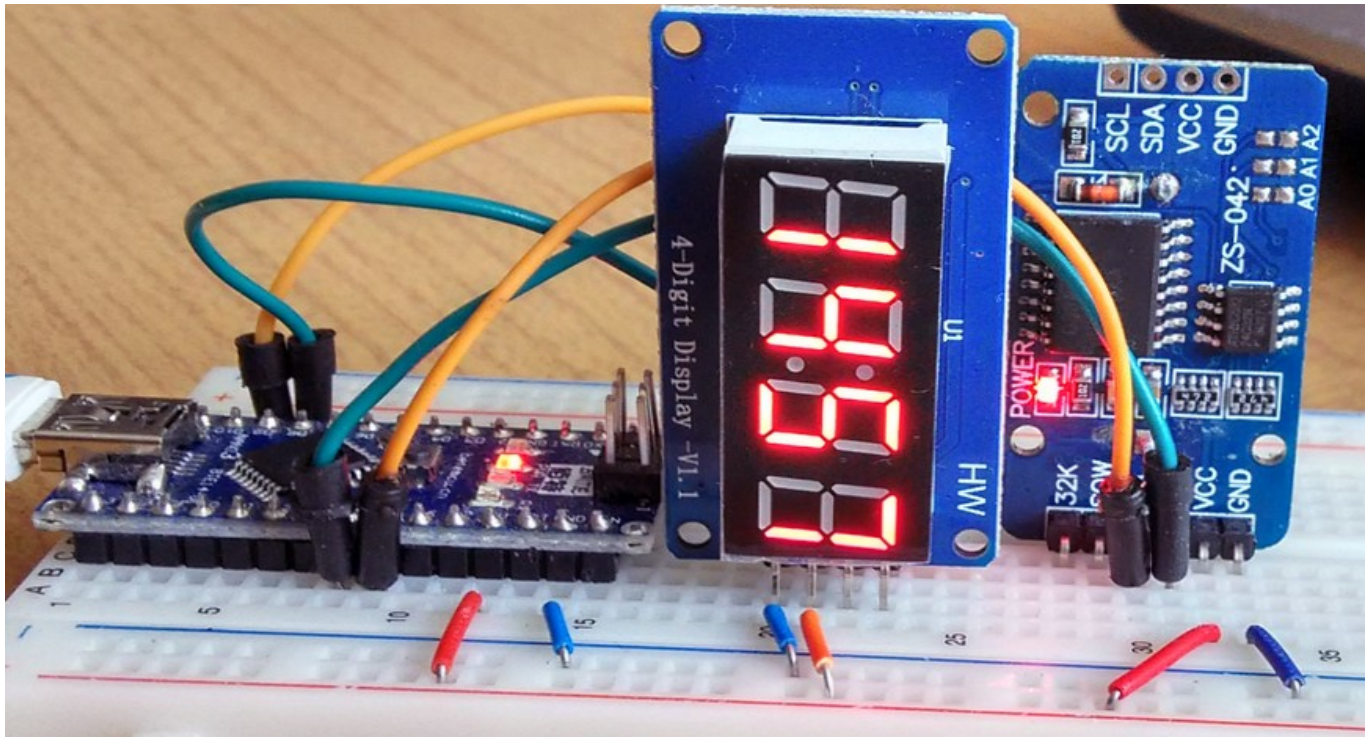
A TM1637Display programkönyvtár metódusai

- **setBrightness**(brightness) – fényerő beállítása (0 – 7) és bekapcsolás (8)
- **showNumberDec**(*num, leading_zero = false, length=4, pos=0*) – szám kiíratása decimális alakban, ahol ***num*** = 0 – 9999, negatív számok esetén pedig 0-tól –999-ig, ***length*** a számjegyek száma, ***pos*** a kezdő pozíció (0-3)
- **showNumberDecEx**(*num, dots=0, leading_zero=false, length=4, pos=0*) – szám kiíratása tizedespont vezérléssel (***dots = 64*** esetén kigyújtja a középső kettőspontot, ***dots = 0*** esetén pedig kioltja)
- **setSegments**(*segments[], length=4, pos=0*) - szegmensek beállítása a ***segments[]*** tömb szerint
- byte **encodeDigit**(digit) – visszaadja a ***digit*** számjegy szegmenseinek kódját, pl. 1 esetén `SEG_B | SEG_C` értékét

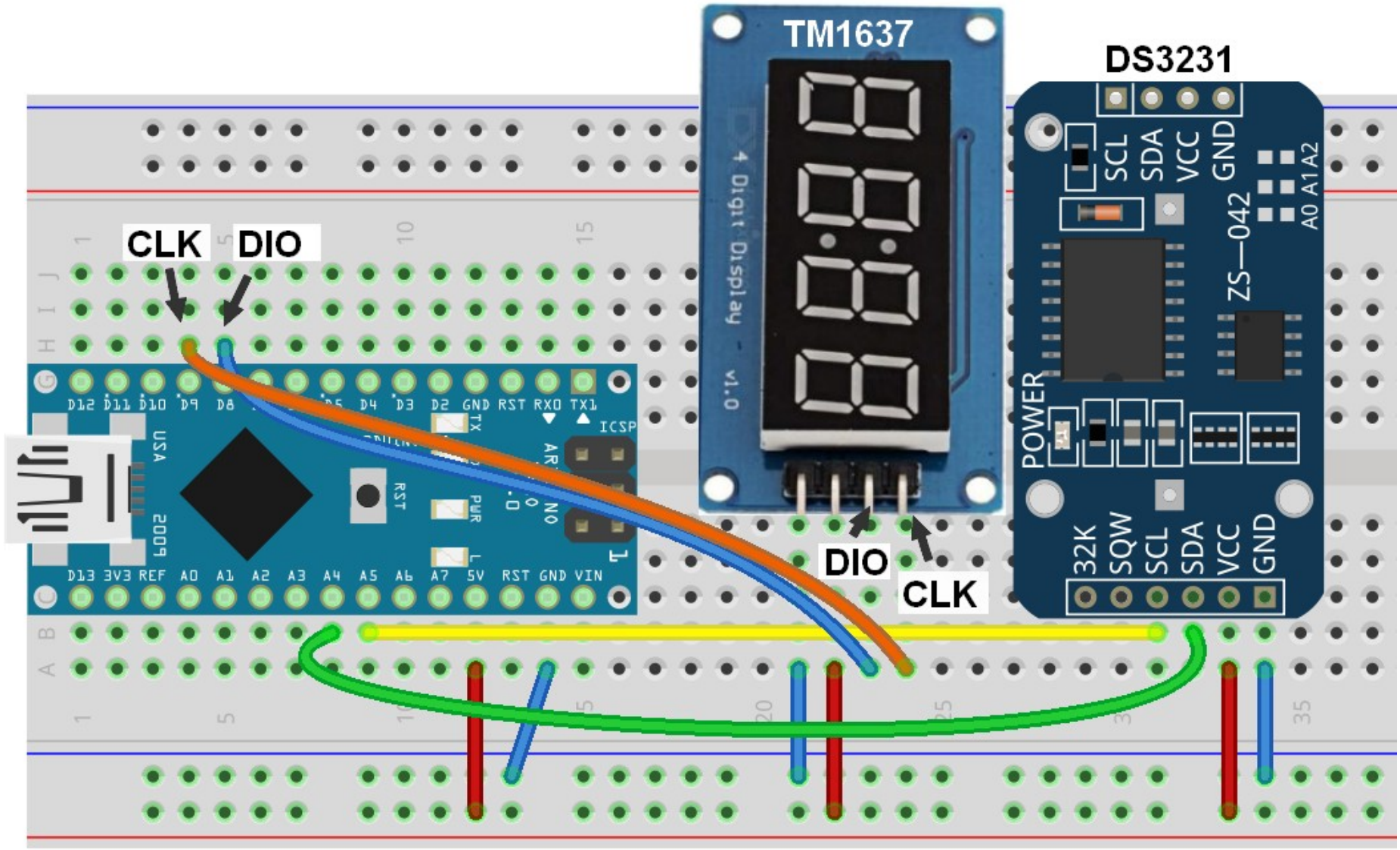
```
#define SEG_A   0b00000001
#define SEG_B   0b00000010
#define SEG_C   0b00000100
#define SEG_D   0b00001000
#define SEG_E   0b00010000
#define SEG_F   0b00100000
#define SEG_G   0b01000000
```

Digitális kijelzésű óra

- A kijelző modulunk egyik lehetséges felhasználása egy digitális kijelzésű óra (óra és perc kijelzéssel)
- A hackster.io/pentiumcadiz/4-digit-rtc-clock-85 068b korábban bemutatott projektje egy korszerűtlen, **DS1307** modult használt, jobb helyette a fejlettebb **DS3231** modult használni, ami pontosabb és kompatibilis a LIR2032 Li-ion akkumulátorral is
- **Hozzávalók:**
  - TM1637 kijelző
  - DS3231 RTC modul
  - Arduino
  - 9V-os elem vagy hálózati adapter

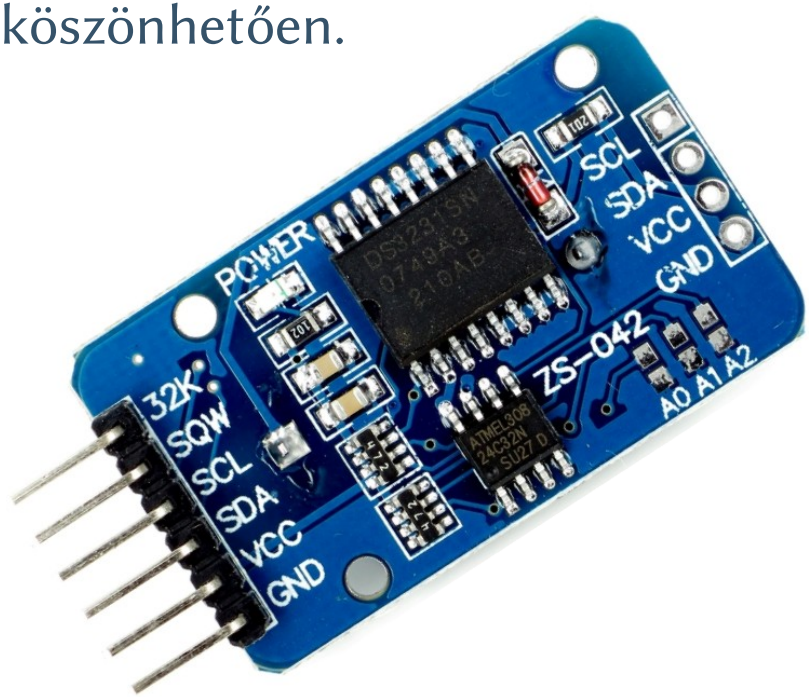


DS3231_clock kapcsolási elrendezés



DS3231 Real-time óra modul

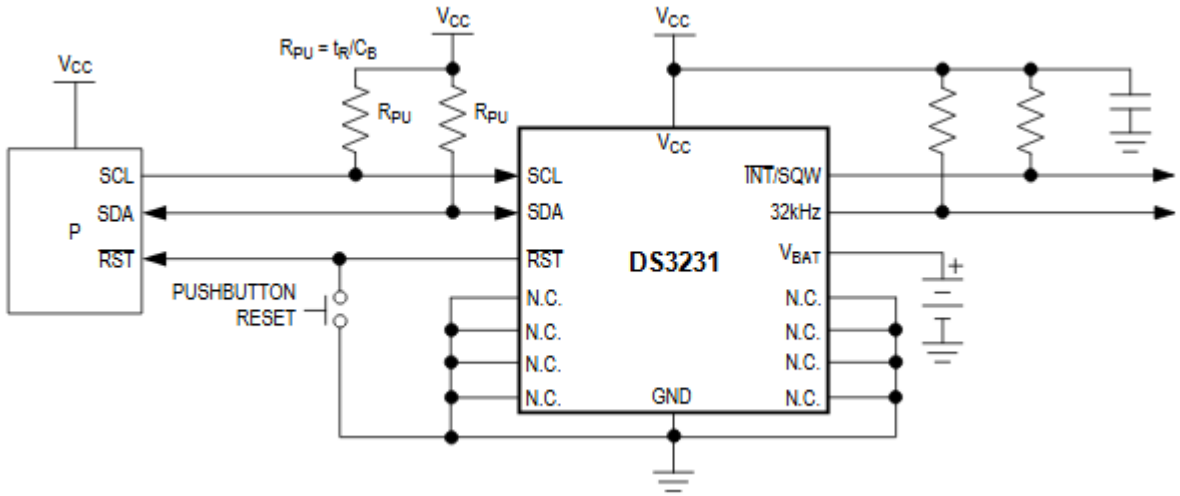
- Oszcillátor, óra és naptár egy tokban. A tápfeszültség megszűnésekor a hátoldalán elhelyezett telepről üzemel tovább.
- A modul többé-kevésbé cserekompatibilis a DS1307-tel, egy 4 kB EEPROM is tartalmaz, de van néhány eltérés:
  - pontosabb óra (±2 ppm a –40 – 80 °C tartományban) a beépített hőkompenzált oszcillátornak köszönhetően.
  - két riasztási időpont is megadható
  - van beépített hőmérője
  - nincs belső RAM
  - Vbat 4.2 V-os Li akkuval is táplálható!
- EEPROM I2C címe: **0x57** (állítható)
- DS3231 I2C címe: **0x68**



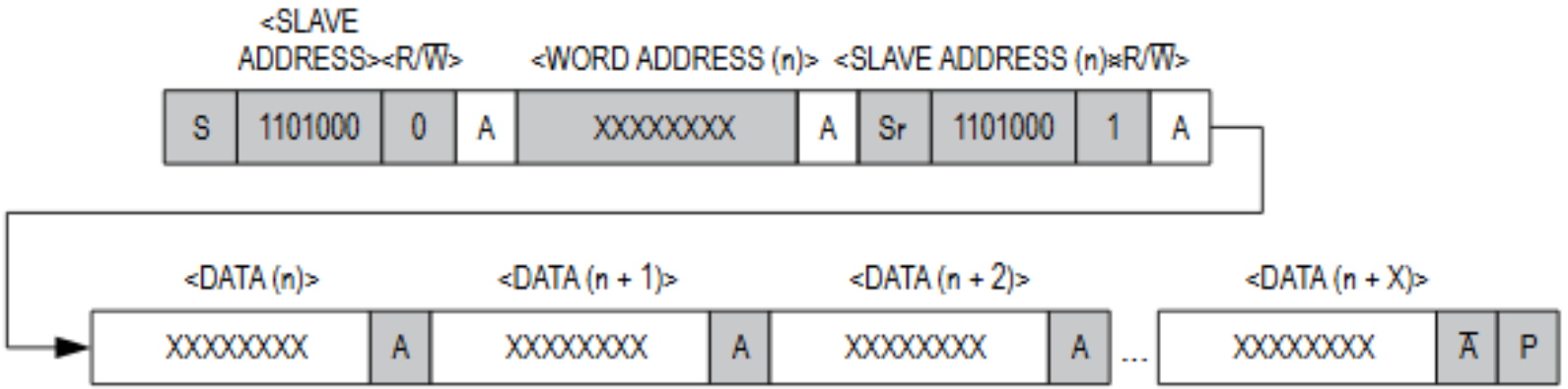
- Adatlap: datasheets.maximintegrated.com/en/ds/DS3231.pdf

A DS3231 bekötése, használata

- Egy tipikus áramköri elrendezés:



- Adatforgalom az I2C buszon:



DS3231 regiszterek

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

A DS3231 programkönyvtár

- Mintapéldánkban ***Andrew Wickert*** programkönyvtárát használjuk: https://github.com/NorthernWidget/DS3231

A legfontosabb függvények:

- byte **getHour**(bool& h12, bool& PM) – az óra regiszter kiolvasása mellett megadja a 12/24 beállítás és a PM bit értékét is
  (a **&** jel azt jelzi, hogy cím szerinti hivatkozást használunk)
- byte **getMinute**() – kiolvassa a perc regiszter értékét
- **enableOscillator**(true, 0, 0);
- **setHour(), setMinute()** – az óra, illetve a perc regiszter beállítása

Kezdeti beállítás:

- A DS3231 RTC kezdeti beállítását a programkönyvtár egyik mintapéldája, a **DS3231_set.ino** program segítségével végezhetjük el. A program a terminál ablakban YYMMDDWhhmmssx alakban beírt dátum és idő adatokat tárolja el, ahol W a hét napja (1-7) és x a kötelezően beírandó zárókarakter (ez zárja a parancsot)

tm1637_clock1.ino

```
#include <TM1637Display.h>
#include <DS3231.h>
#include <Wire.h>
#define CLKPIN  9                          // TM1737 CLK láb D9-re
#define DIOPIN  8                          // TM1737 CLK láb D9-re
DS3231 Clock;                              // RTC példányosítása
TM1637Display display(CLKPIN, DIOPIN);     // 4-számjegyű kijelző példányosítása

void setup() {
  Wire.begin();                            // Az I2C illesztő inicializálása
  display.setBrightness(0x0F);             // a kijelző bekapcsolása max. fényerőn
}

void loop() {
  bool h12, PM;   int hh, mm;
  hh = Clock.getHour(h12, PM);             // Óra kiolvasása
  display.showNumberDecEx(hh,64,false,2,0); // Óra kijelzése kettősponttal
  mm = Clock.getMinute();                  // Percek kiolvasása
  display.showNumberDec(mm,true,2,2);      // percek kijelzése
  delay(1000);                             // 1 másodperc  várakozás
  display.showNumberDecEx(hh,0,false,2,0); // kettőspont kioltás
  delay(1000);                             // 1 másodperc várakozás
}
```

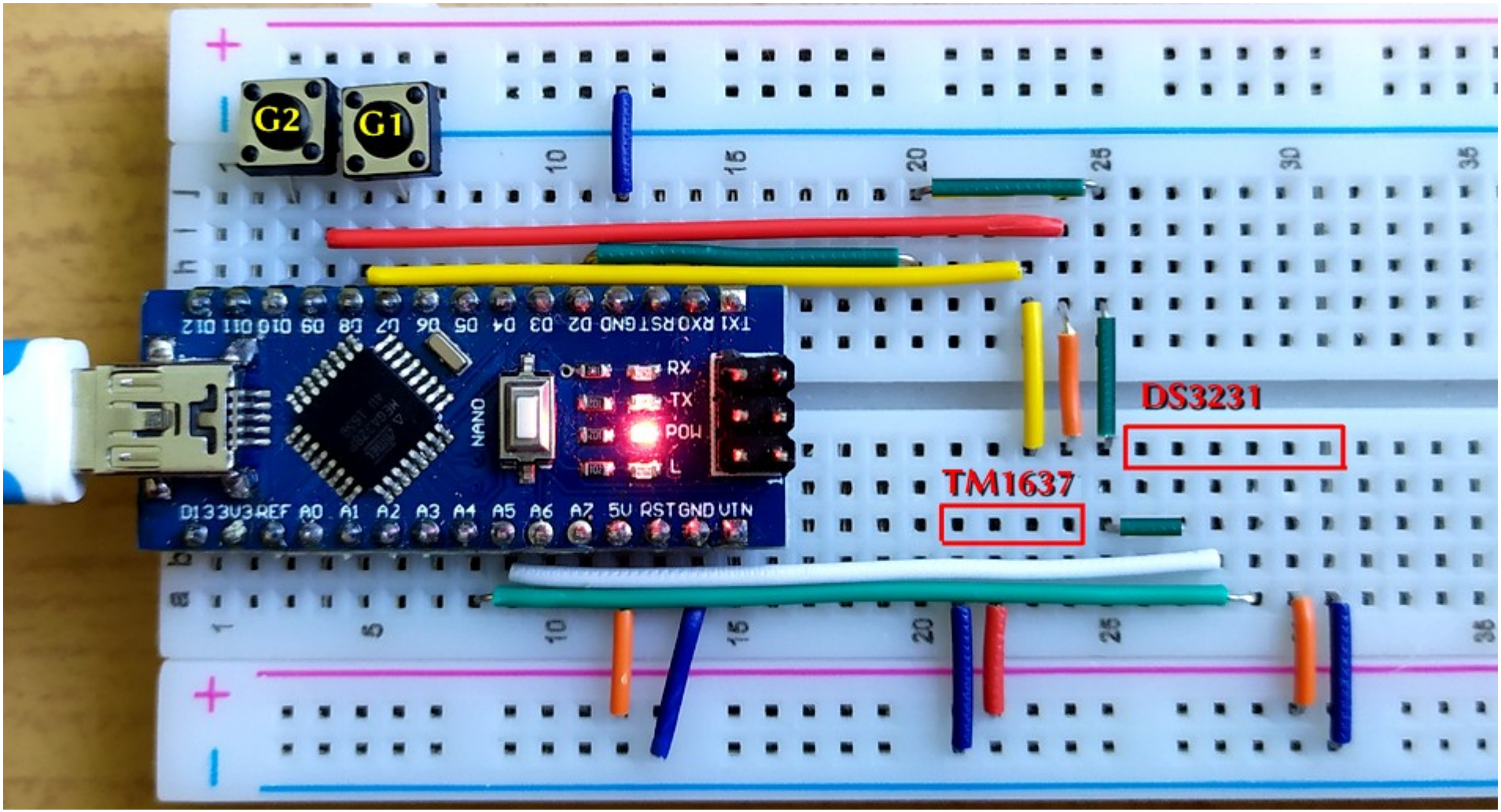
Jó, jó, de hiányzik a beállíthatóság!

Első nagy óraprojektünk

- Egészítsük ki az előző kapcsolást két nyomógombbal!
  - Az első gombot megnyomjuk, akkor eggyel megnöveljük az órák számát
  - Ha a második gombot nyomjuk meg, akkor pedig a percek számát növeljük eggyel
- A túlcsordulások kezelése
  - 0 – 24h kijelzést használunk, a növelésnél a `hh = (hh+1) % 24` formula kezeli a túlcsordulást: 24 helyett már 0 lesz a kijelzett érték
  - A percek száma 0 – 59 közötti szám lehet, itt növelésnél a `mm = (mm+1) % 60` formula kezeli jól a túlcsordulást
- A kettőspont ütemes villogtatását úgy oldhatjuk meg egyszerűen, hogy a **DS3231** modul 1 Hz-es négyszögjel kimenetét bekötjük az **Arduino** egyik bemenetére (pl. **D2**) és ennek a jelnek az állapotváltásaikor váltjuk a kettőspont állapotát is (minden változásnál váltva: 1 Hz, csak felfutásnál váltva: 0,5 Hz)

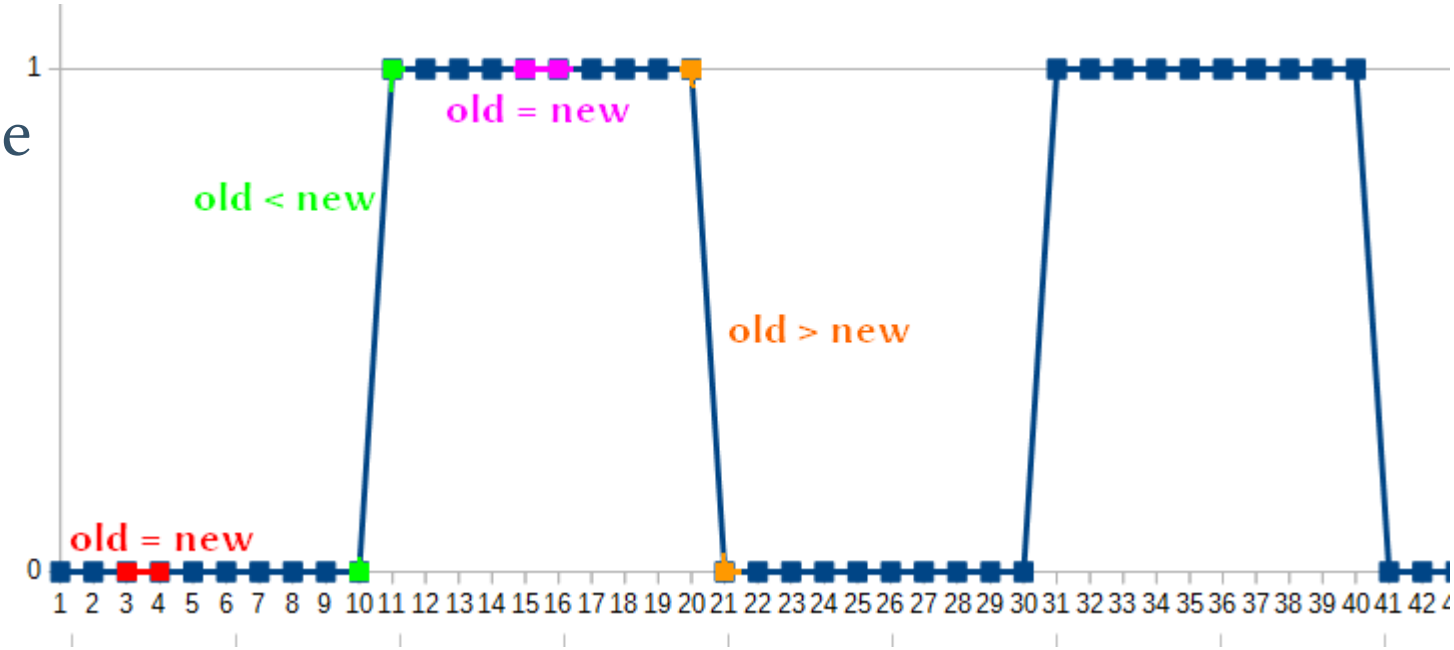
Egészítsük ki az előző kapcsolást!

- Kössünk két nyomógombot **D7**, ill. **D10** és a **GND** közé!
- Kössük össze a **DS3231** modul **SQW** kimenetét a **D2** bemenettel!



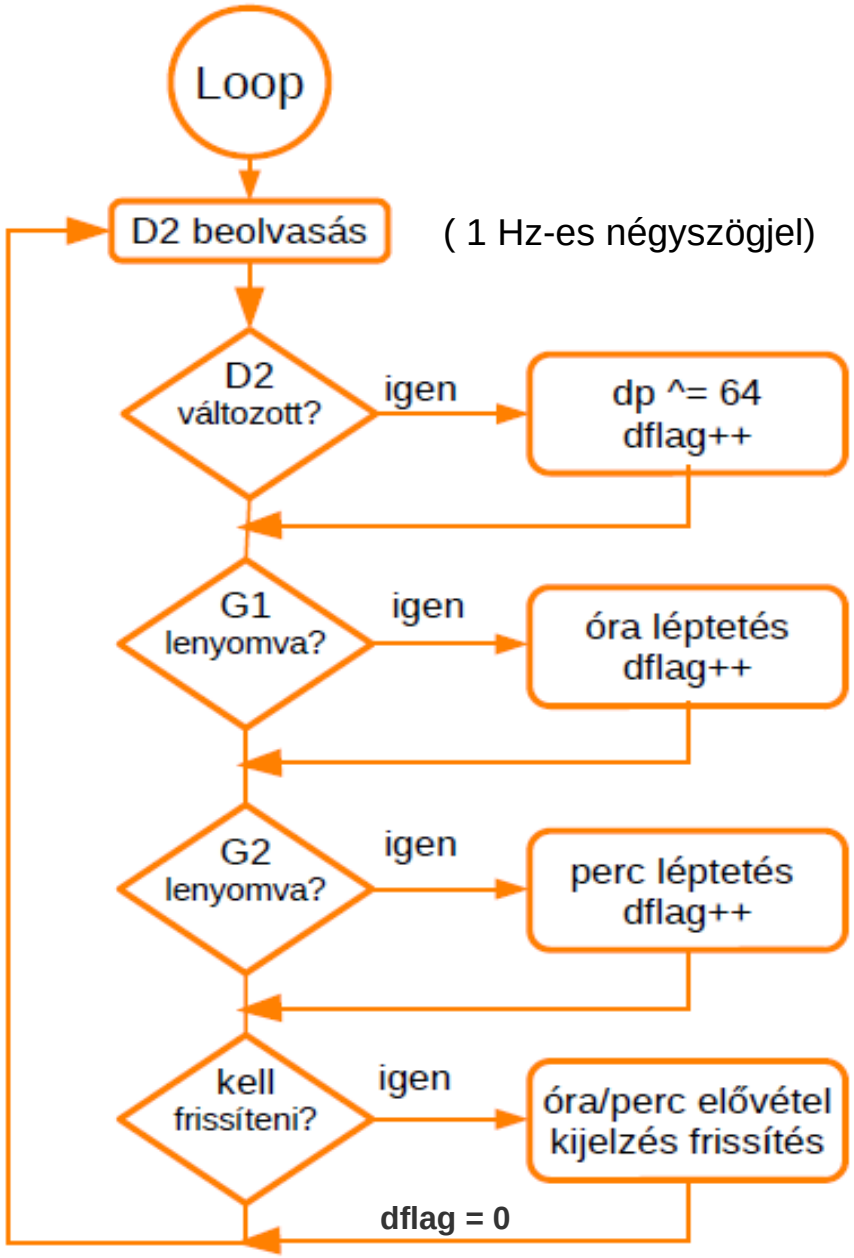
Jelváltozások detektálása

- Azonos időközönként mintavételezzük a bemenetet
- Az új értéket mindig az előzővel hasonlítjuk össze:
  - Ha a két érték egyenlő (alacsony, vagy magas), akkor nincs változás
  - Ha a régi érték alacsony, az új pedig magas szint, akkor **felfutó átmenetet** detektáltunk
  - Ha a régi érték magas, az új pedig alacsony szint, akkor pedig **lefutó átmenetet** detektáltunk
- Az `old != new` feltétel teljesülése pedig azt jelzi, hogy fel- vagy lefutó élt detektáltunk



Egyszerű eseménykezelés

- Egyszerű esetben az események (a **D2** bemeneti állapot megváltozása, a **G1** vagy **G2** gombok lenyomása) egymástól függetlenül kezelhetők
- A `loop()` függvény végtelen ciklusában feltételvizsgálatokkal eldöntjük, hogy egy esemény bekövetkezett-e, s ha igen, akkor elvégezzük az adott eseményhez tartozó feladatot
- A `dflag` jelző nem nulla állapota jelzi, hogy a kijelzést frissíteni kell, mert valami megváltozott
- A frissítés tehát „származtatott” esemény



tm1637_clock2.ino 2/1. oldal

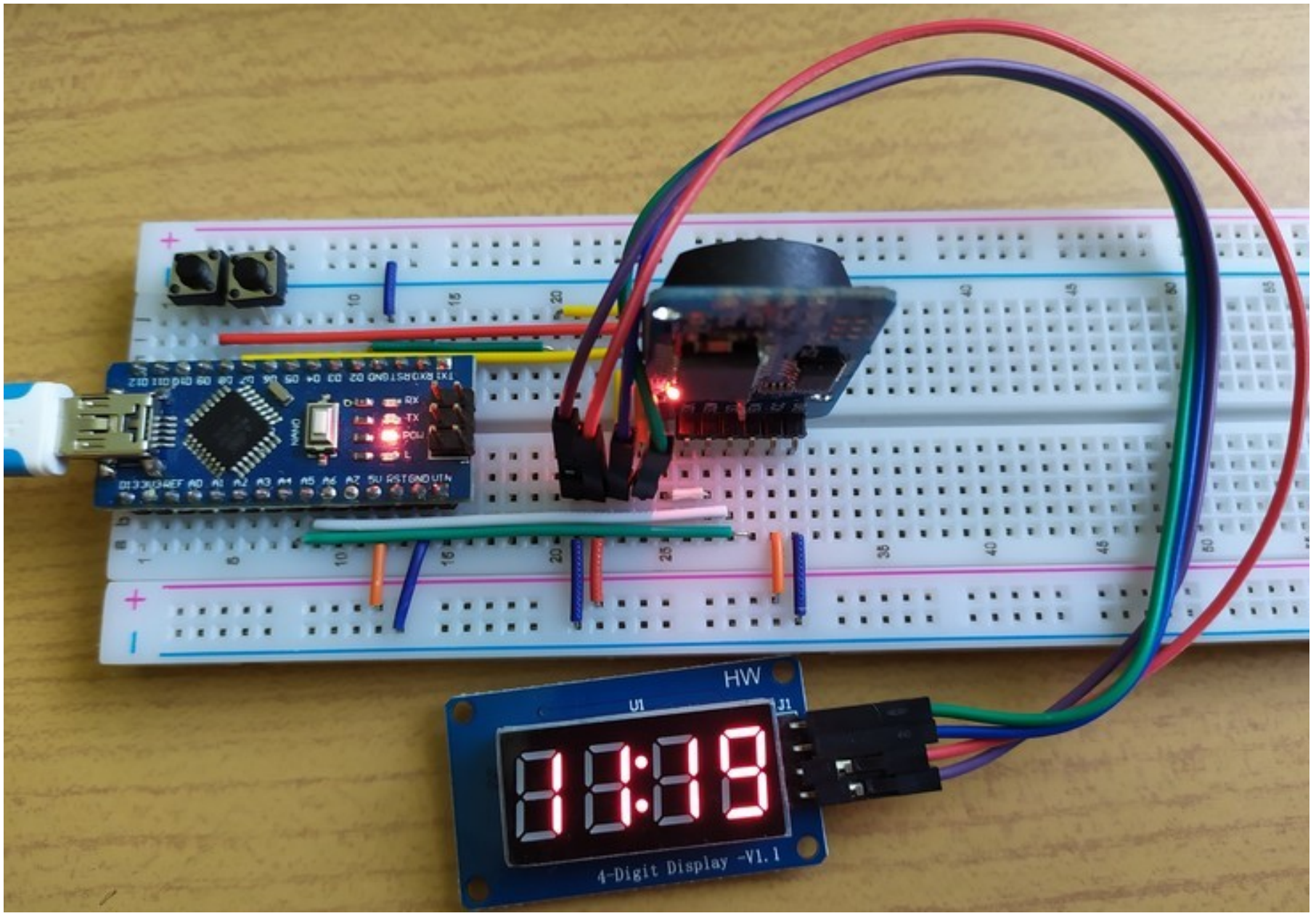
```
#include <TM1637Display.h>
#include <DS3231.h>
#include <Wire.h>
#define CLKPIN  9                       // TM1637 CLK láb D9-re
#define DIOPIN  8                       // TM1637 DIO láb D8-ra
#define G1      7                       // Button 1 (óra léptetés)
#define G2     10                       // Button 2 (perc) léptetés
#define SQW     2                       // 1 Hz-es négyszögjel

DS3231 Clock;                           // Az RTC példányosítása
TM1637Display display(CLKPIN,DIOPIN); // 4-számjegyű kijelző példányosítása
bool h12, PM;                           // 12/24 mód és AM/PM jelzőbitek
int hh = 0;                             // órák száma
int mm = 0;                             // percek száma
int dp = 0;                             // Kettőspont kijelzés (0 vagy 64)
int d2new = 0;                          // D2 bemenet új állapota
int d2old = 0;                          // D2 bemenet előző állapota
int dflag = 0;                          // nem nulla, ha frissíteni kell a kijelzést

void setup() {
  pinMode(G1, INPUT_PULLUP);
  pinMode(G2, INPUT_PULLUP);
  pinMode(SQW, INPUT_PULLUP);
  Wire.begin();                         // Az I2C illesztő inicializálása
  display.setBrightness(0x0F);          // a kijelző bekapcsolása max. fényerőn
  Clock.enableOscillator(true, 0, 0); // clock ON/off, battery, frequency (0: 1Hz)
}
```

tm1637_clock2.ino 2/2. oldal

```
void loop() {
  d2new = digitalRead(SQW);      //-- A D2-re kötött négyszögjel vizsgálata -----
  if (d2new != d2old) {                // D2 állapot beolvasása
    dp ^= 64;                          // dp = 0 vagy 64
    dflag++;                           // frissítés kijelzés, ha változás van
  }
  d2old = d2new;                  //-- A G1 nyomógomb állapotának vizsgálata -----
  if ( digitalRead(G1) == LOW) {       // Ha a G1 gomb le van nyomva...
    hh = (hh + 1) % 24;                // Az órák léptetése
    Clock.setHour(hh);                 // Eltároljuk az új értéket
    dflag++;                           // Képfrissítés kell
    delay(100);
  }                               //-- A G2 nyomógomb állapotának vizsgálata -----
  if ( digitalRead(G2) == LOW) {       // Ha a G2 gomb le van nyomva...
    mm = (mm + 1) % 60;                // A percek léptetése
    Clock.setMinute(mm);               // Eltároljuk az új értéket
    dflag++;                           // Képfrissítés kell
    delay(100);
  }                               //-- A kijelzés frissítése, ha szükséges ----
  if (dflag) {                                  // Kell-e frissíteni?
    hh = Clock.getHour(h12, PM);                // Órák kiolvasása
    mm = Clock.getMinute();                     // Percek kiolvasása
    display.showNumberDecEx(hh,dp,false,2,0); // Óra kijelzése kettősponttal
    display.showNumberDec(mm, true,2,2);        // percek kijelzése
    dflag = 0;
  }
  delay(100);
}
```

Itt megfontolandó egy
`Clock.setSecond(0);`
parancs kiadása is!

A megépített kapcsolás

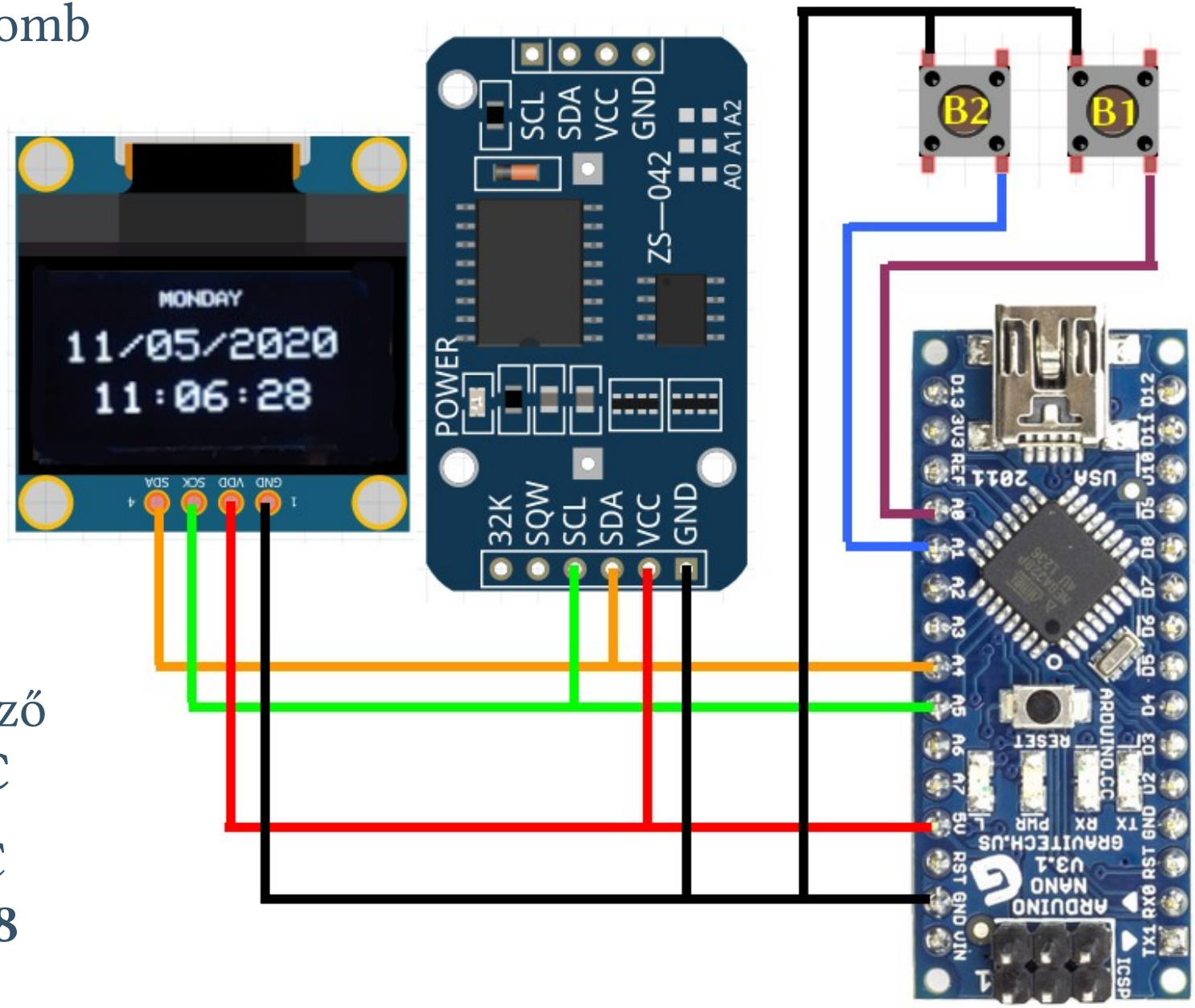


Második nagy óraprojekt

- Az **SSD1306** I2C OLED kijelzővel és az azt kezelő **Adafruit_ssd1306** valamint **Adafruit_GFX** könyvtárakkal a **2020. február 20**-i előadásban foglalkoztunk részletesen (előadásvázlat, mintaprogramok)
- Most egy olyan óraprojektbe fogunk, amelynél szintén **DS3231** óramodult használunk, a kijelző pedig egy **SSD1306** I2C OLED modul
- A grafikus kijelző lehetővé teszi, hogy több információt jelenítsünk meg: pl. a nap neve, a dátum, és az idő (óra, perc és a másodperc is)
- A beállítást most is két nyomógombbal végezzük:
  - Az első gomb végigléptet az adatokon, az éppen kiválasztott adat villog
  - A második gomb lépteti (növeli) az értéket, s kezeljük a túlcsordulást
  - A percbeállítás után az első gombot megnyomva a másodperc nullázódik, s kilépünk a menüből
- **Simple Projects:** Arduino real time clock using DS1307 and SSD1306 OLED című projektjéből indultunk ki, amelyet többször átírtunk...

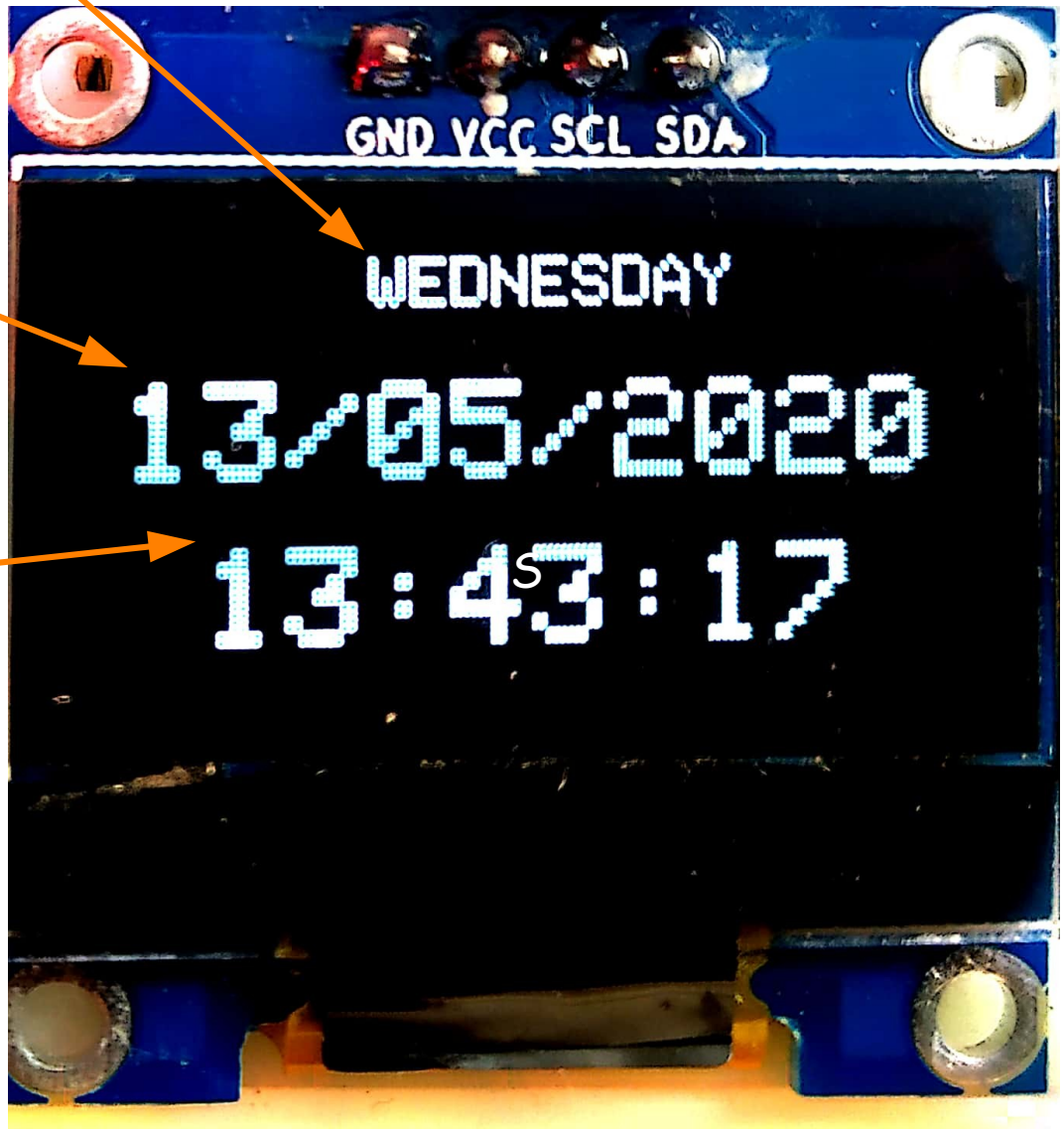
A kapcsolási vázlat

- A két nyomógomb az **A0** és **A1** bemenetekre van kötve
- Az **I2C** busz: **A4 – SDA** és **A5 – SCL** (a felhúzás be van építve a modulokra)
- Az OLED kijelző I2C címe **0x3C**
- A DS3231 RTC I2C címe: **0x68**



Kijelzés pozicionálás

- **A hét napja**
  xpos = 40
  ypos = 0
  font size = 1
- **Dátum**
  xpos = 4
  ypos= 18
  font size = 2
- **Idő**
  xpos = 16
  ypos= 42
  font size = 2



clock_params.h: az óra paraméterei

- A napok neveit egy tömbben tároljuk
- Az RTC paramétereit egy struktúratömbben tároljuk, abban a sorrendben, ahogy az RTC regiszterekben tárolódnak
- A struktúrán belül **val** az érték, **lim** a határ **base** az induló érték (ami 0 vagy 1 lehet) **xpos** és **ypos** pedig a kijelzési pozíciót adják meg

```
char dayname [8] [10] = {
  "          ",
  " SUNDAY   ",
  " MONDAY   ",
  " TUESDAY  ",
  "WEDNESDAY",
  "THURSDAY  ",
  " FRIDAY   ",
  "SATURDAY  "
};

struct clkparms {
  byte val;            // a paraméter értéke
  byte lim;            // a túlcsordulási határ
  byte base;           // a legkisebb érték (1 vagy 1)
  byte xpos;           // kijelzési pozíció x koordináta
  byte ypos;           // kijelzési pozíció y koordináta
} par[] = {
  {0,60,0,88,42},      // Seconds   [00-59]
  {0,60,0,52,42},      // Minutes   [00-59]
  {0,24,0,16,42},      // Hours     [00-23]
  {0,7,1,40,0},        // Day of week [1-7]
  {0,31,1,4,18},       // Date       [1-31]
  {0,12,1,40,18},      // Month      [1-12]
  {0,100,0,100,18}     // Year       [0-99]
};
```

oled_clock3.ino 5/1. oldal

```
#include <Wire.h>                          // Include Wire library (required for I2C)
#include <DS3231.h>
#include <Adafruit_GFX.h>                  // Include Adafruit graphics library
#include <Adafruit_SSD1306.h>              // Include Adafruit SSD1306 OLED driver
#include "clock_params.h"
#define SCREEN_WIDTH 128                   // OLED display width, in pixels
#define SCREEN_HEIGHT 64                   // OLED display height, in pixels
#define OLED_RESET    -1                   // Reset pin # (or -1)
#define button1    A0                      // Button B1 is connected to Arduino pin A0
#define button2    A1                      // Button B2 is connected to Arduino pin A1

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
DS3231 Clock;                              // Az RTC példányosítása

void setup(void) {
  pinMode(button1, INPUT_PULLUP);
  pinMode(button2, INPUT_PULLUP);
  Clock.enableOscillator(true, 0, 0);
  delay(1000);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // the I2C addr: 0x3C or 0x3D
  display.clearDisplay();
  display.display();
  display.setTextColor(WHITE, BLACK);
}
```

oled_clock3.ino 5/2. oldal

```
void loop() {
  if (!digitalRead(button1)) {           // If button B1 is pressed
    edit(3);                             // Edit day of week
    edit(4);                             // Edit date
    edit(5);                             // Edit month
    edit(6);                             // Edit year
    edit(2);                             // Edit hours
    edit(1);                             // Edit minutes
    par[0].val=0; Clock.setSecond(0);    // Set Seconds to zero
    Clock.setMinute(par[1].val);
    Clock.setHour(par[2].val);
    Clock.setDoW(par[3].val);
    Clock.setDate(par[4].val);
    Clock.setMonth(par[5].val);
    Clock.setYear(par[6].val);
    delay(200);                          // Wait 200ms
  }
  par[0].val = Clock.getSecond();
  par[1].val = Clock.getMinute();
  par[2].val = Clock.getHour(h12, pm); // if (h12&&pm) par[2].val = (par[2].val+12)%24;
  par[3].val = Clock.getDoW();
  par[4].val = Clock.getDate();
  par[5].val = Clock.getMonth(century);
  par[6].val = Clock.getYear();
  display_parameter(3, true);           // Display day of week
  DS1307_display();   delay(50);        // Diaplay time & calendar
}
```

oled_clock3.ino 5/3. oldal

```
//-- Edit a time or date parameter ----------------------
void edit(byte i) {
  while (!digitalRead(button1));                          // Wait until button B1 released
  while (true) {
    while (!digitalRead(button2)) {                       // If button B2 is pressed
      par[i].val = (par[i].val - par[i].base + 1) % par[i].lim + par[i].base;
      display_parameter(i, true);
      delay(200);                                         // Wait 200ms
    }
    display_parameter(i, false);                          // Hide parameter
    blink_parameter();                                    // Wait for max. 250 ms
    display_parameter(i, true);                           // Display parameter
    blink_parameter();                                    // Wait for max. 250 ms
    if (!digitalRead(button1)) {                          // If button B1 is pressed
      return;                                             // Return parameter value and exit
    }
  }
}

//-- Wait for button press for maximum 250 ms -----------------
void blink_parameter() {
  byte j = 0;
  while (j < 10 && digitalRead(button1) && digitalRead(button2)) {
    j++;
    delay(25);
  }
}
```

oled_clock3.ino 5/4. oldal

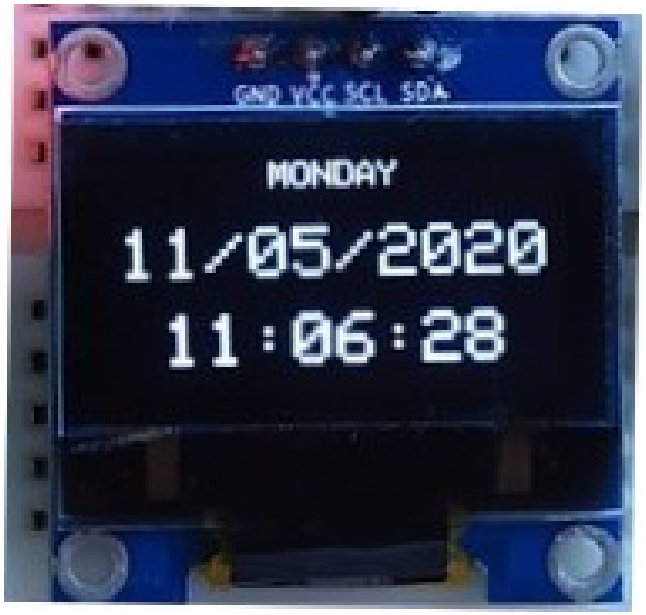
```
//-----------------------------------------------------
// Display the ith parameter or blank space
//-----------------------------------------------------
void display_parameter(byte i, boolean flag) {
  char text[3];
  if (flag) {
    if (i != 3) {
      text[1] = par[i].val%10 + 48;
      text[0] = par[i].val/10 + 48;
      text[2] = 0;
      draw_text(par[i].xpos, par[i].ypos, text, 2);
    }
    else {
      draw_text(par[3].xpos,par[3].ypos,dayname[par[3].val],1);
    }
  }
  else {
    if (i != 3) {
      draw_text(par[i].xpos, par[i].ypos, "  ", 2);
    }
    else {
      draw_text(par[3].xpos,par[3].ypos,dayname[0],1);
    }
  }
}
```

Ha flag értéke igaz, akkor kiírjuk az i-edik paramétert

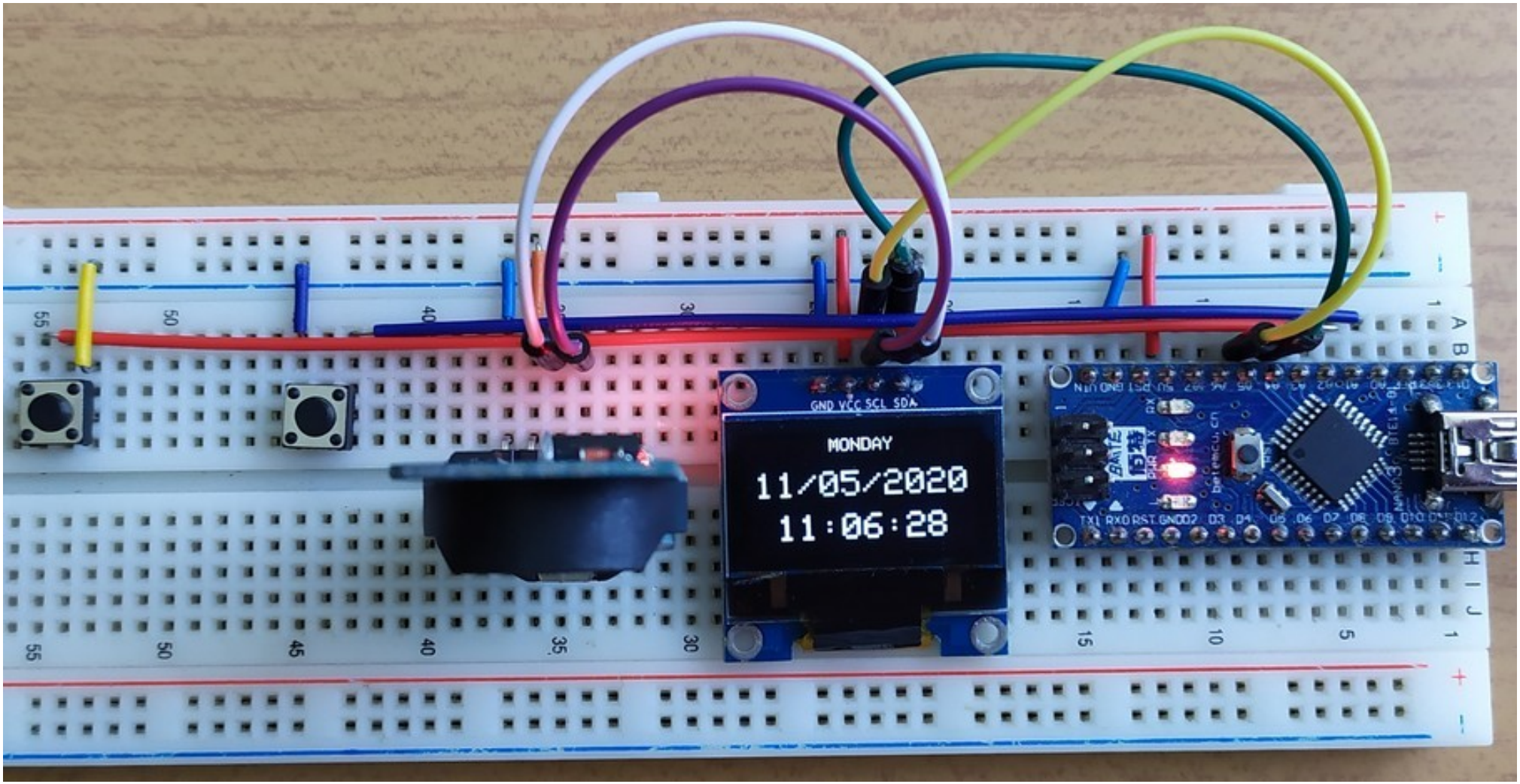
Ha flag értéke hamis, akkor csak helykitöltő szóközöket írunk ki

oled_clock3.ino 5/5. oldal

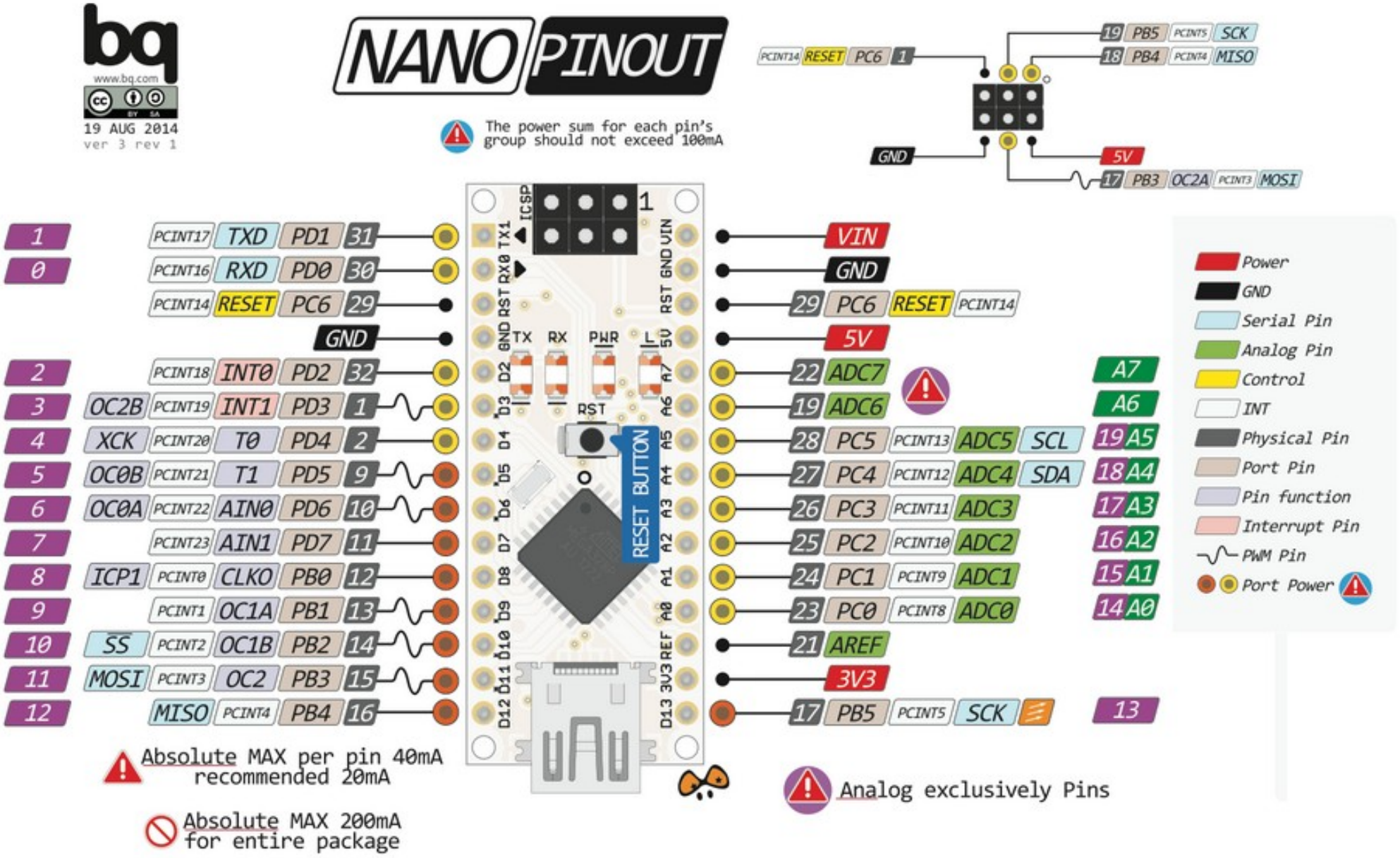
```
char Time[]     = "  :  :  ";
char Calendar[] = "  /  /20  ";
bool h12, pm, century;

void DS1307_display() {
  Time[7]     = par[0].val % 10 + 48; // másodpercek
  Time[6]     = par[0].val / 10 + 48;
  Time[4]     = par[1].val % 10 + 48; // percek
  Time[3]     = par[1].val / 10 + 48;
  Time[1]     = par[2].val % 10 + 48; // órák
  Time[0]     = par[2].val / 10 + 48;
  Calendar[9] = par[6].val % 10 + 48; // évszám [0-99]
  Calendar[8] = par[6].val / 10 + 48;
  Calendar[4] = par[5].val % 10 + 48; // hónap
  Calendar[3] = par[5].val / 10 + 48;
  Calendar[1] = par[4].val % 10 + 48; // nap
  Calendar[0] = par[4].val / 10 + 48;
  draw_text(4,  18, Calendar, 2);     // dátum kiírása
  draw_text(16, 42, Time, 2);         // idő kiírása
}

void draw_text(byte x_pos, byte y_pos, char *text, byte text_size) {
  display.setCursor(x_pos, y_pos);
  display.setTextSize(text_size);
  display.print(text);
  display.display();
}
```



A megépített kapcsolás



Az Arduino nano kártya kivezetései



Ellenállás színkódok