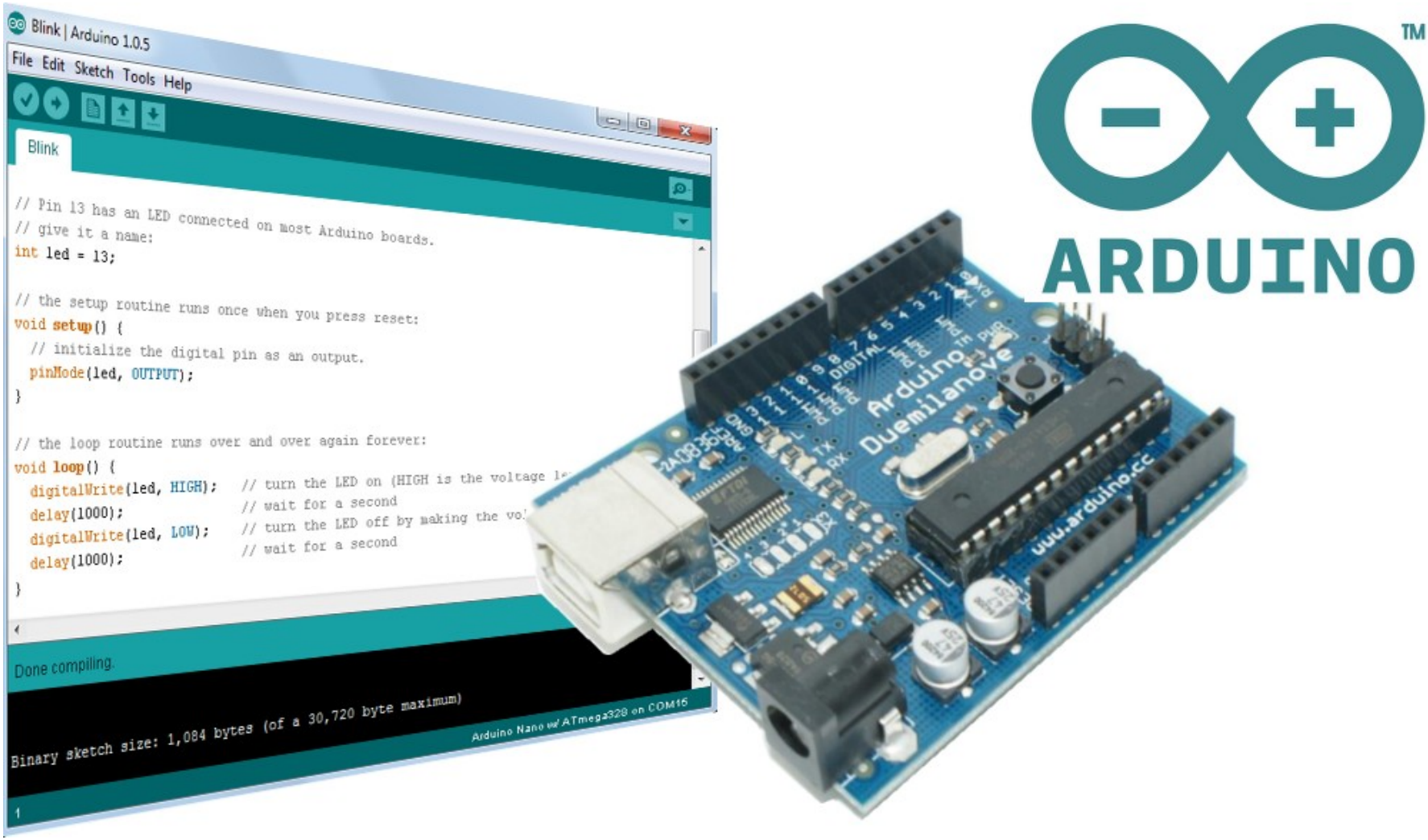
Bevezetés az elektronikába

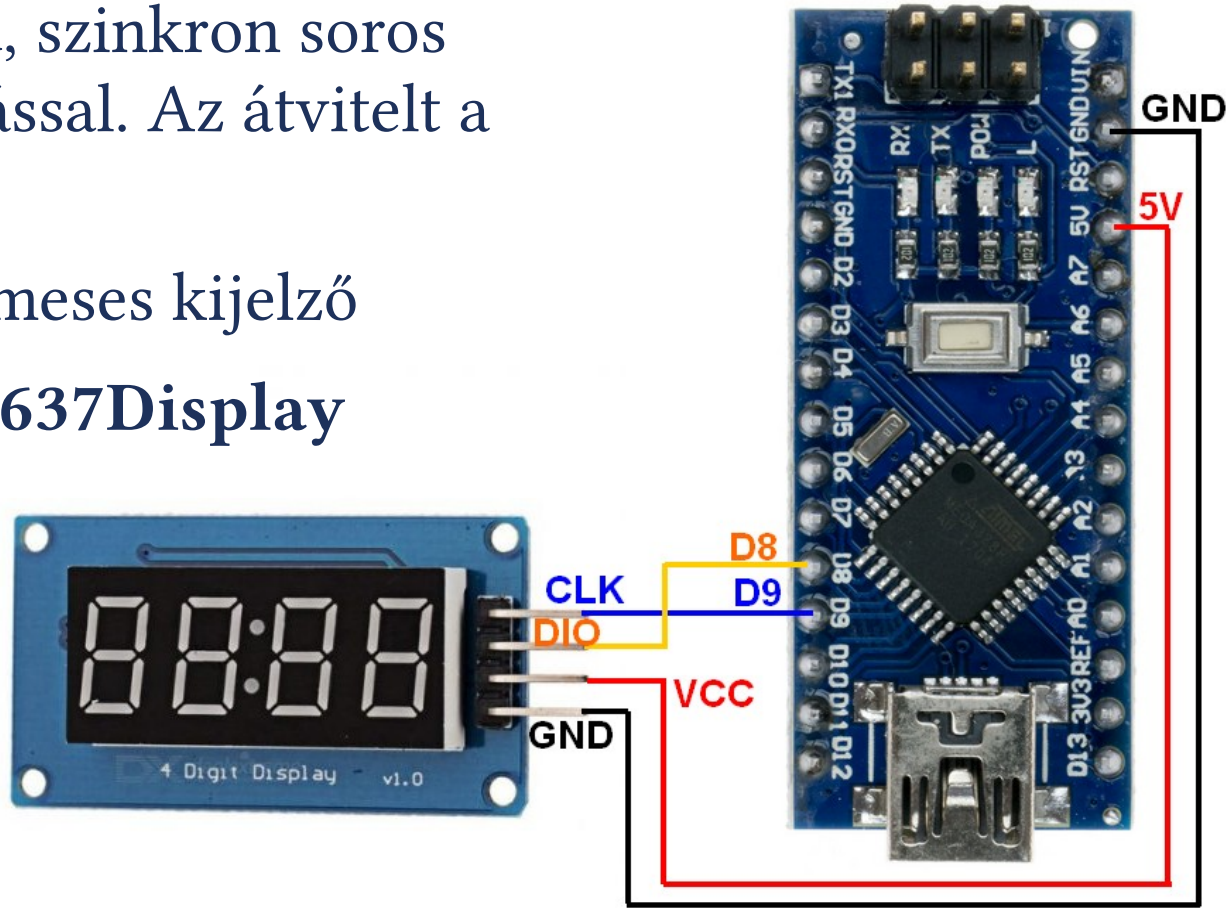


16. Arduino programozás

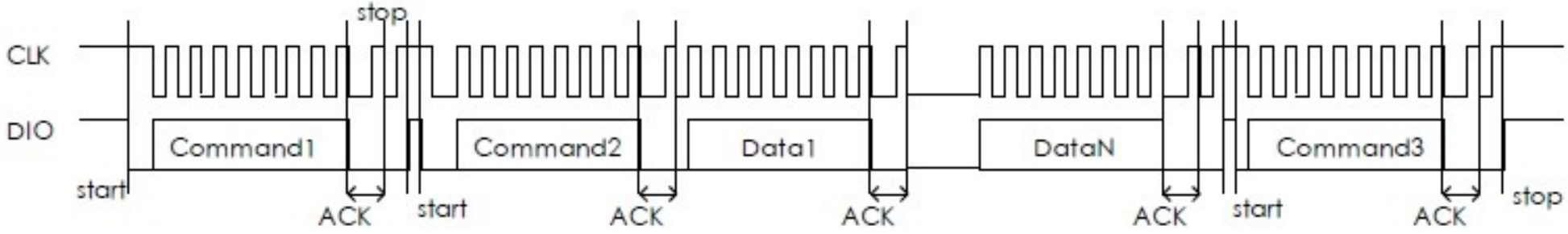
Hétszegmenses kijelző alkalmazások

Emlékeztető: TM1637 4-jegyű kijelző

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



Write SRAM data in address auto increment 1 mode.



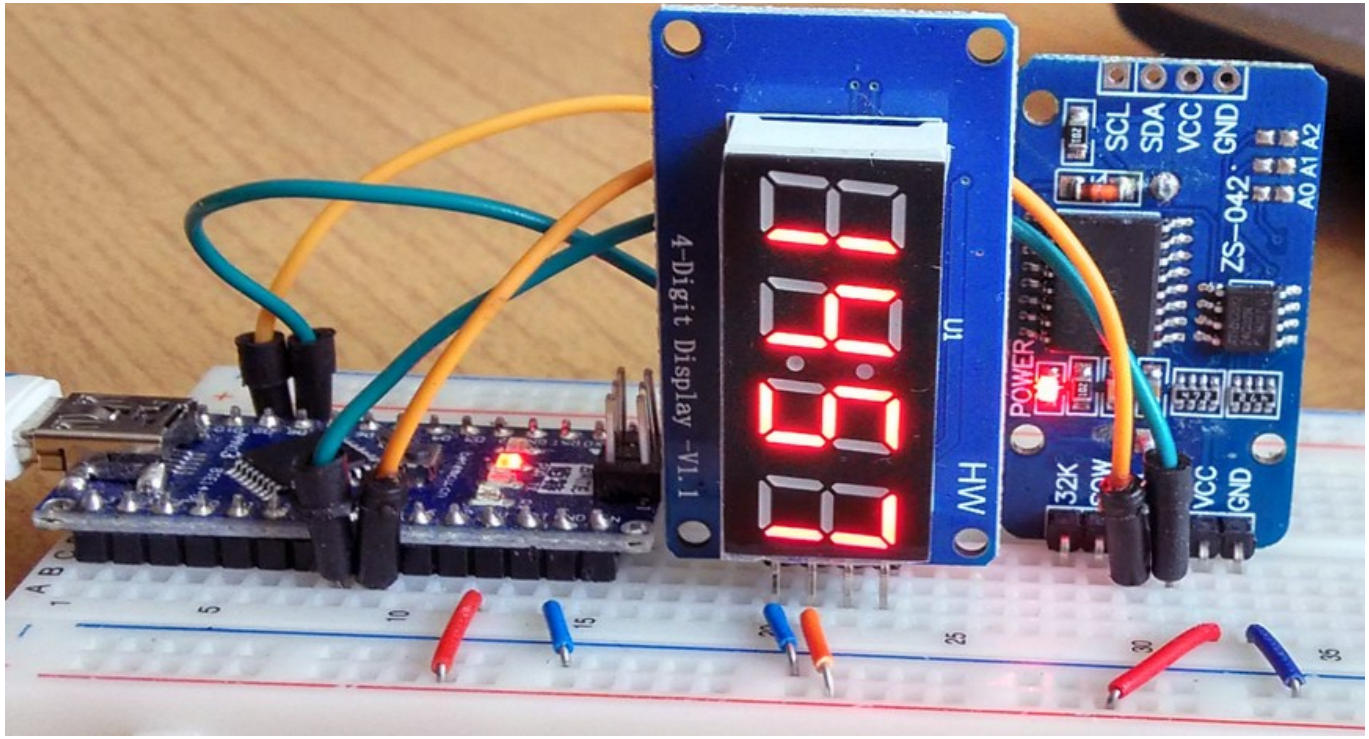
A TM1637Display programkönyvtár metódusai

- **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, 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

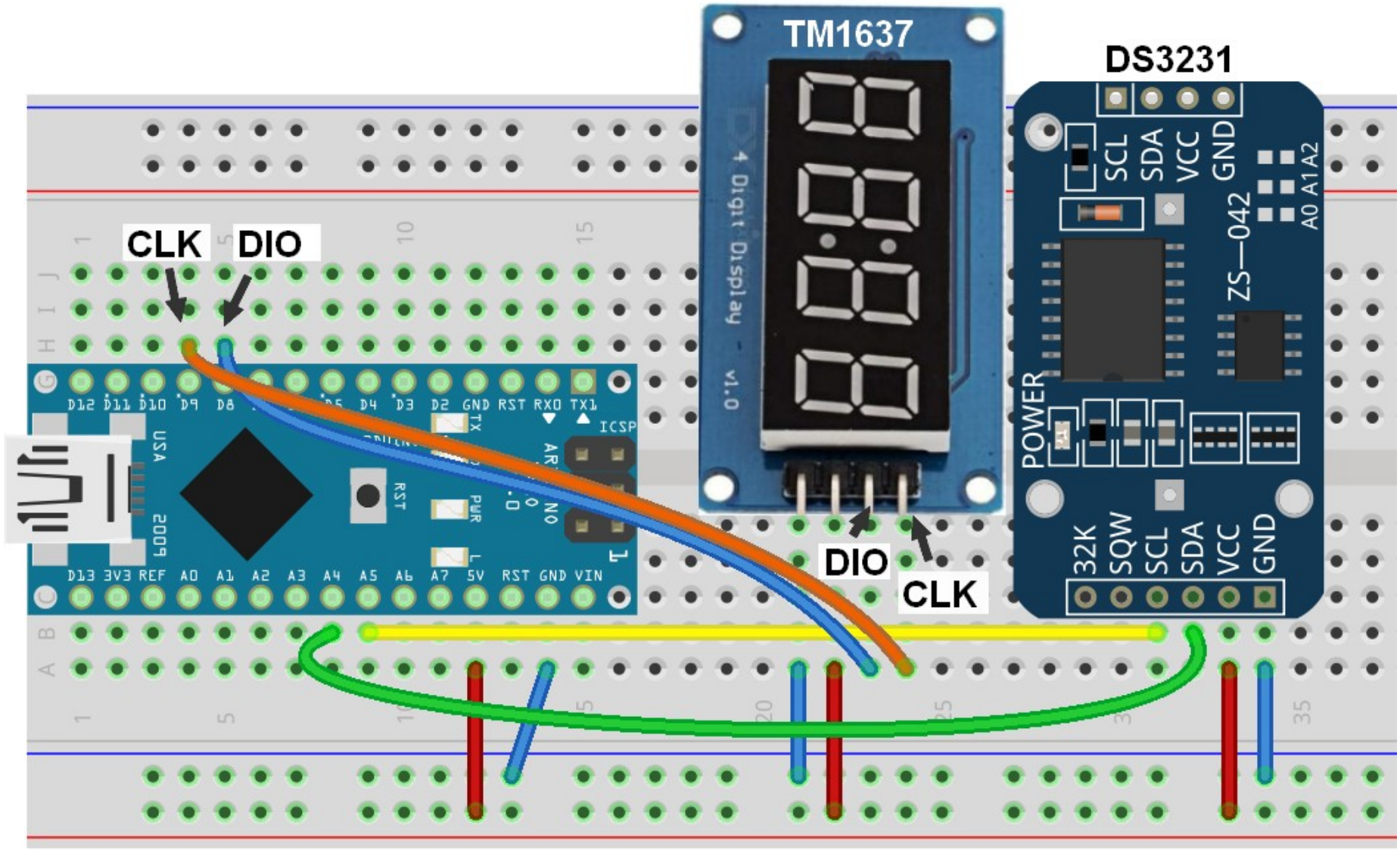
```
#define SEG_A   0b00000001
#define SEG_B   0b00000010
#define SEG_C   0b00000100
#define SEG_D   0b00001000
#define SEG_E   0b00010000
#define SEG_F   0b00100000
#define SEG_G   0b01000000
```

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

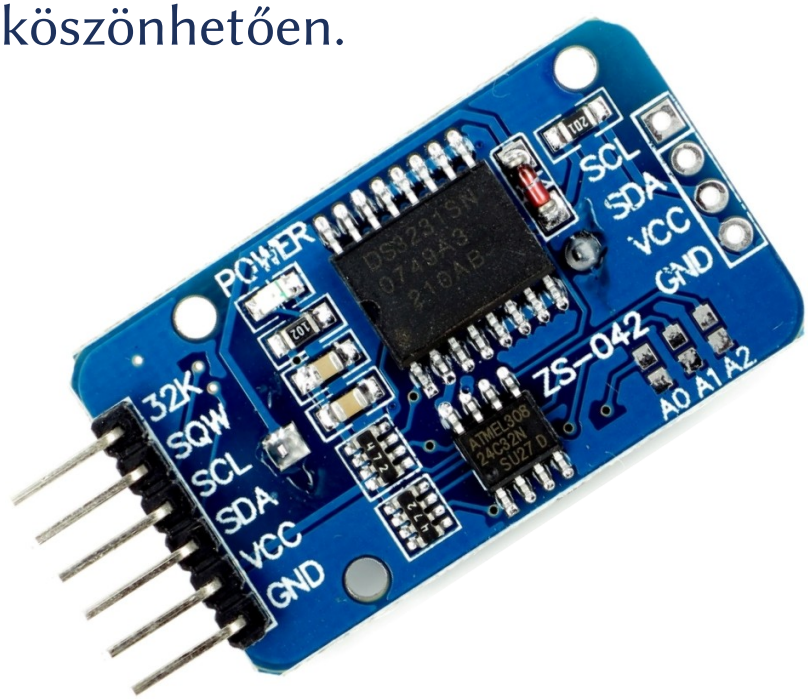


DS3231_clock kapcsolási elrendezés



DS3231 Real-time óra modul

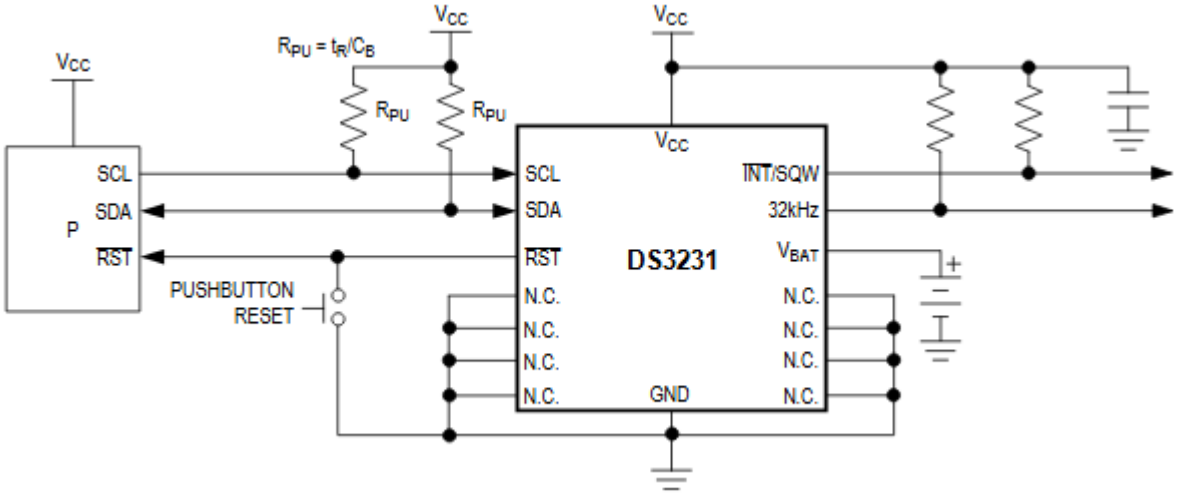
- Oszcillátor, óra és naptár egy tokban. A tápfeszültség megszűnésekor a hátoldalán elhelyezett telepről üzemel tovább.
- A modul többé-kevésbé cserekompatibilis a DS1307-tel, egy 4 kB EEPROM is tartalmaz, de van néhány eltérés:
  - pontosabb óra (±2 ppm a –40 – 80 °C tartományban) a beépített hőkompenzált 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**



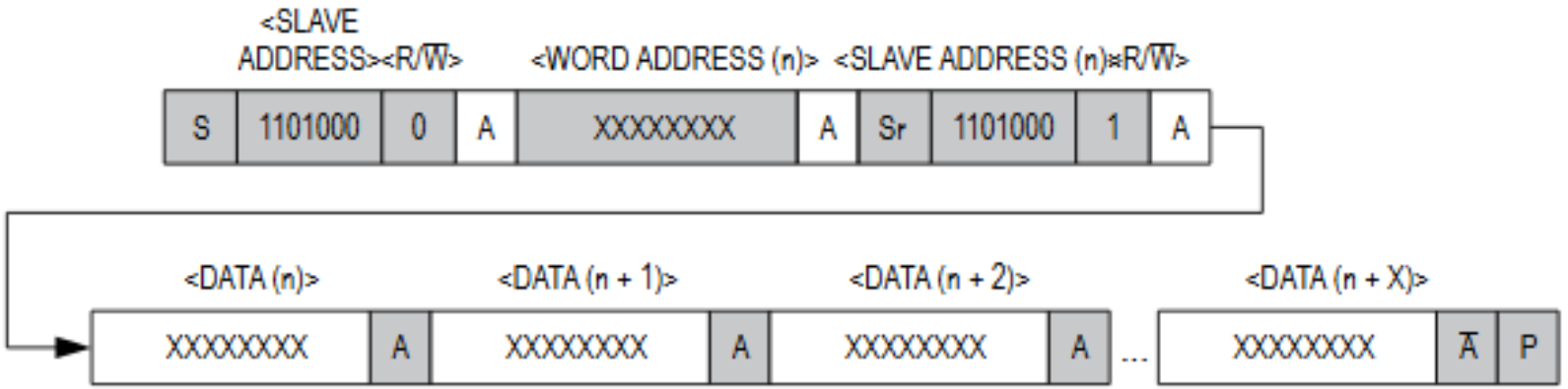
- Adatlap: datasheets.maximintegrated.com/en/ds/DS3231.pdf

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

Kezdeti beállítás:

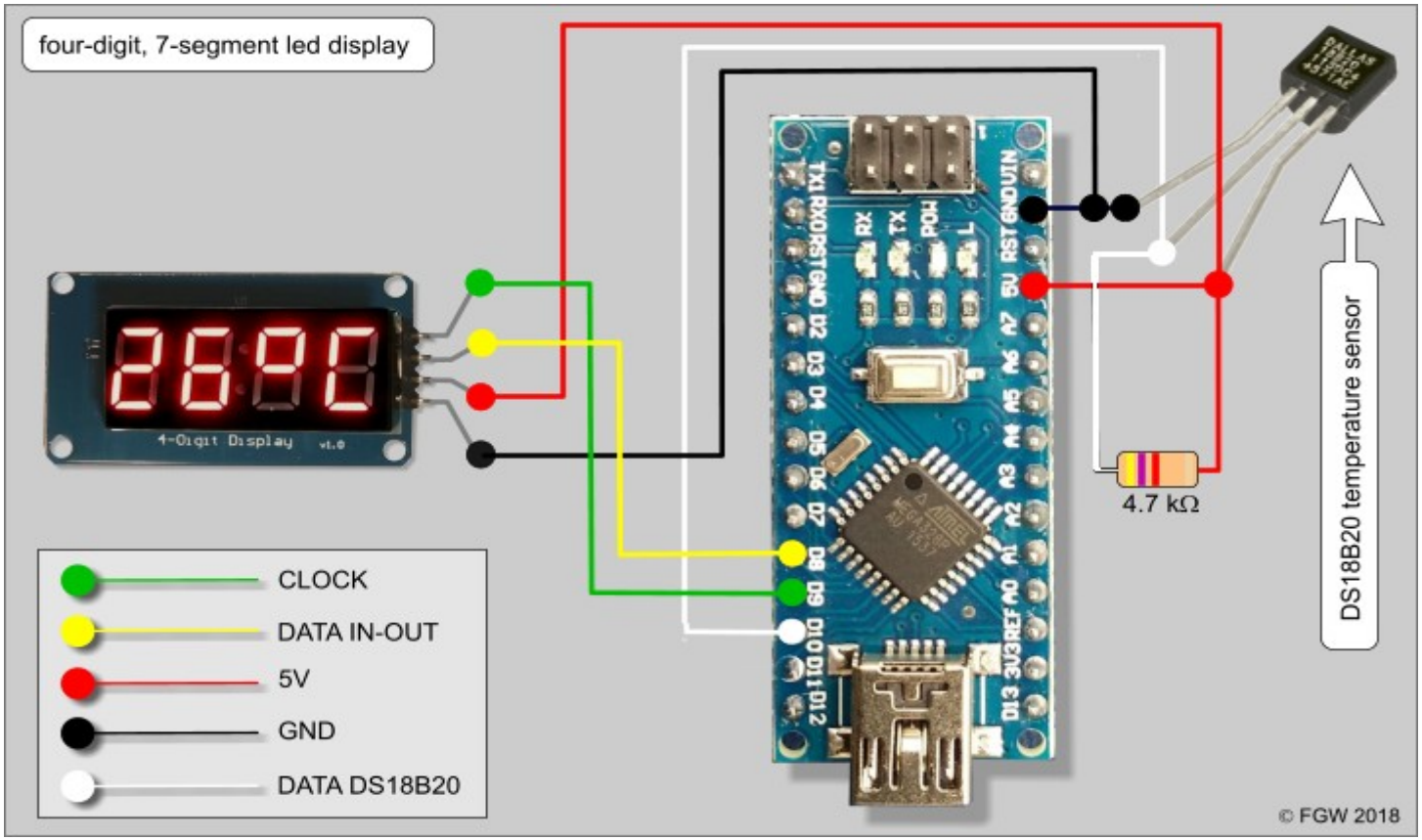
- A DS3231 RTC kezdeti beállítását a programkönyvtár egyik mintapéldája, a **DS3231_set.ino** program segítségével végezhetjük el. A program a terminál ablakban YYMMDDWhhmmssx alakban beírt dátum és idő adatokat tárolja el, ahol W a hét napja (1-7) és x a kötelezően beírandó zárókarakter (ez zárja a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

DS3231_clock.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
}
```

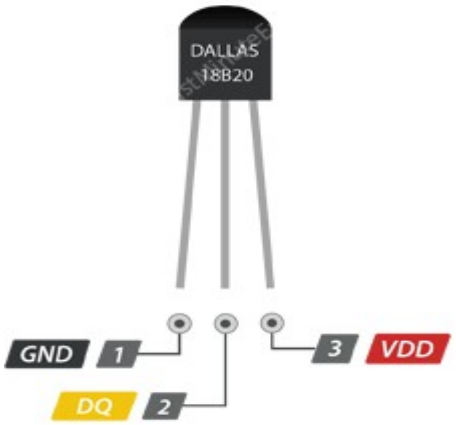
Digitális hőmérő

- Ez a projekt egy **DS18B20** hőmérőt és egy négy digites **TM1637** kijelző modult használ. A kijelzőt a **TM1637Display** könyvtár segítségével kezelik. Az eredeti kapcsolás hibás, itt kijavítottuk.
- Four digit, 7-segment led display for Arduino based on the TM1637 driver
  zonnepanelen.wouterlood.com/10-four-digit-7-segment-led-display-for-arduino-based-on-the-tm1637-driver/

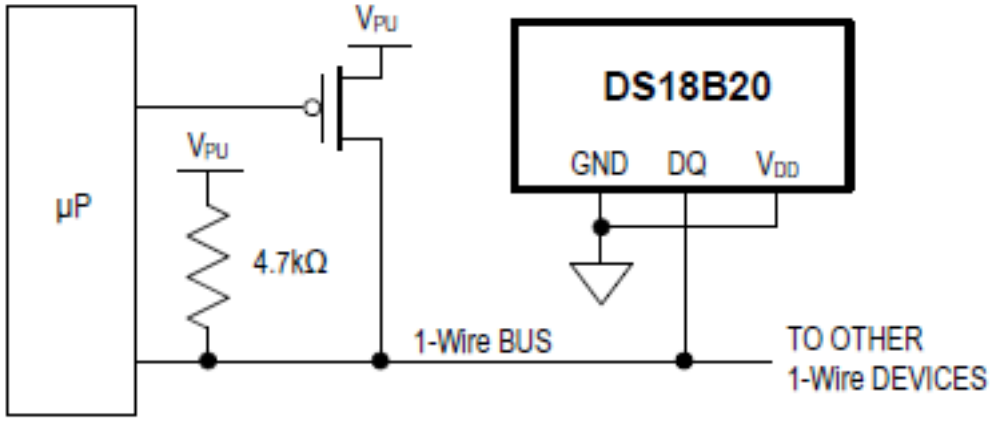


A DS18B20 hőmérő

- A **DS18B20** egy 1-Wire illesztővel rendelkező hőmérő szenzor a Dallas Semiconductor Corp.-tól
- Mérési tartomány: –55℃ -tól +125℃-ig
- Pontosság: ±0.5℃
- Felbontás: 9 – 12 bit (0.5℃ – 0.0625℃)
- Konverziós idő: < 750 ms
- Tápfeszültség: 3 V-tól 5.5 V-ig (1 mA áramfelvétel)
- Mindegyik **DS18B20** 48-bites egyedi kóddal rendelkezik, így az 1-wire buszra felfűzött hőmérők megkülönböztethetők
- Adatlap: datasheets.maximintegrated.com/en/ds/DS18B20.pdf



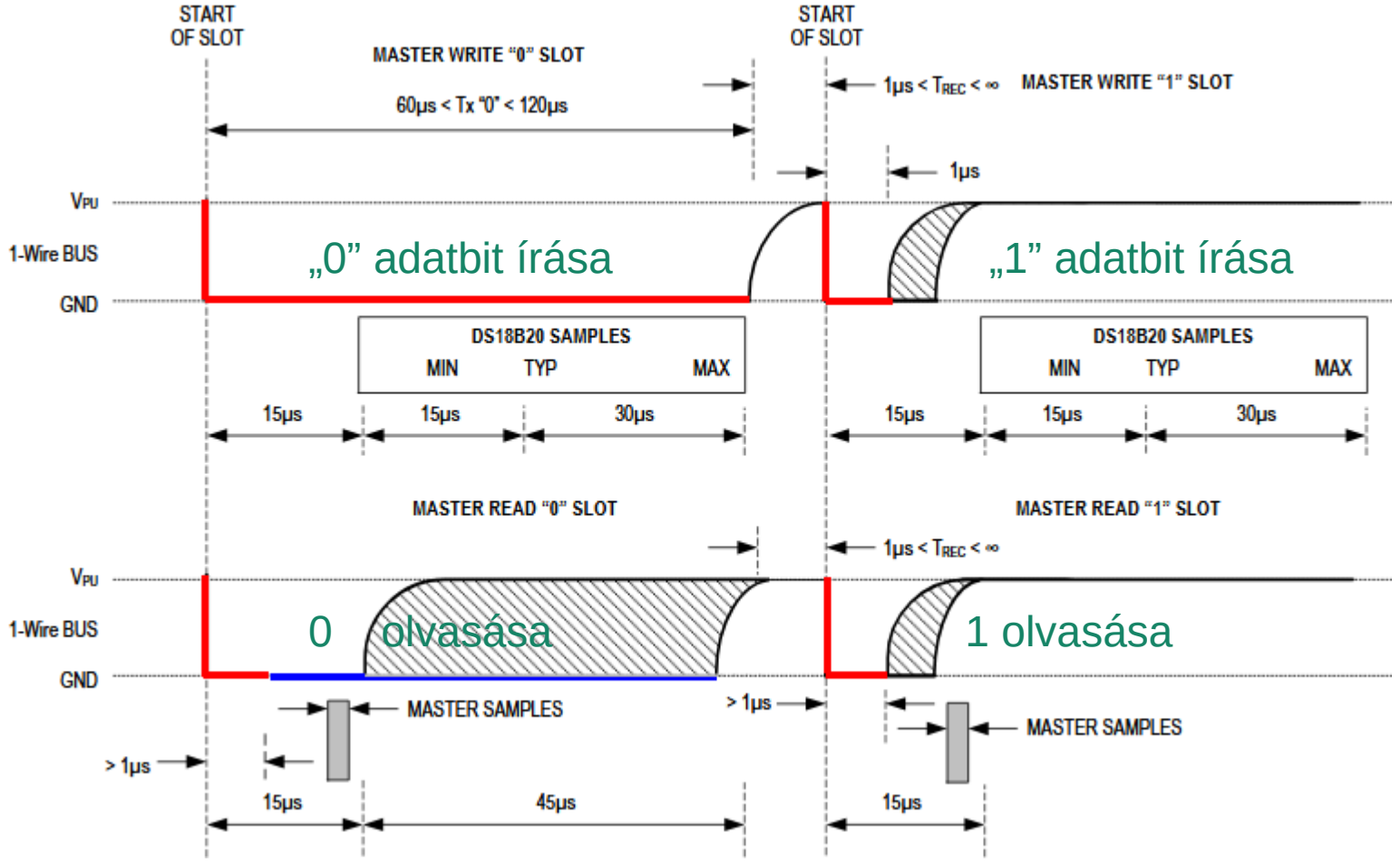
Forrás: lastminuteengineers.com



A táplálás „parazita módban” is megoldható

A DS18B20 hőmérő

- Kommunikáció az 1-Wire buszon:
- Master jele: —— Slave válasz: —— Ellenállás felhúzás: ——



A DS18B20 hőmérő

- A 64 bites egyedi (Lézerrel beégetett) kód felépítése és a regiszterek memóriatérképe (forrás: adatlap)

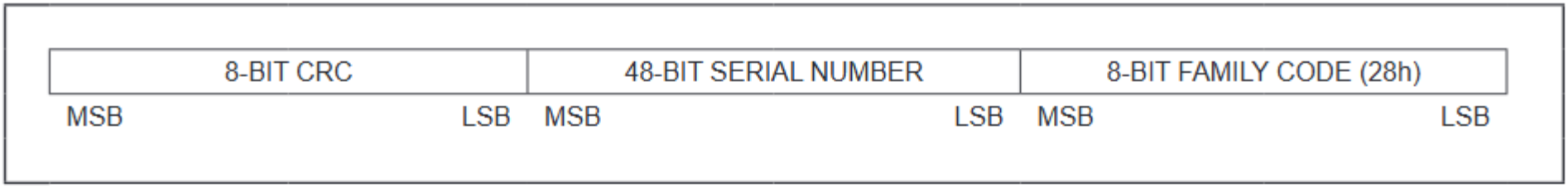


Figure 8. 64-Bit Lasered ROM Code

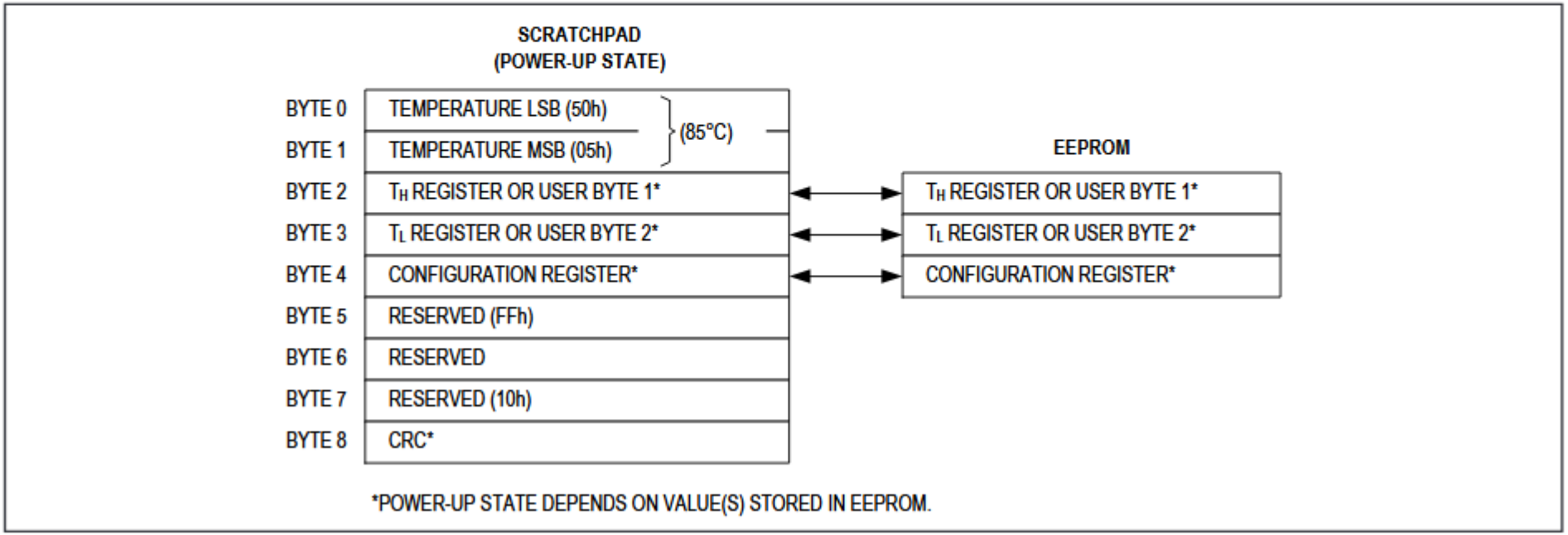
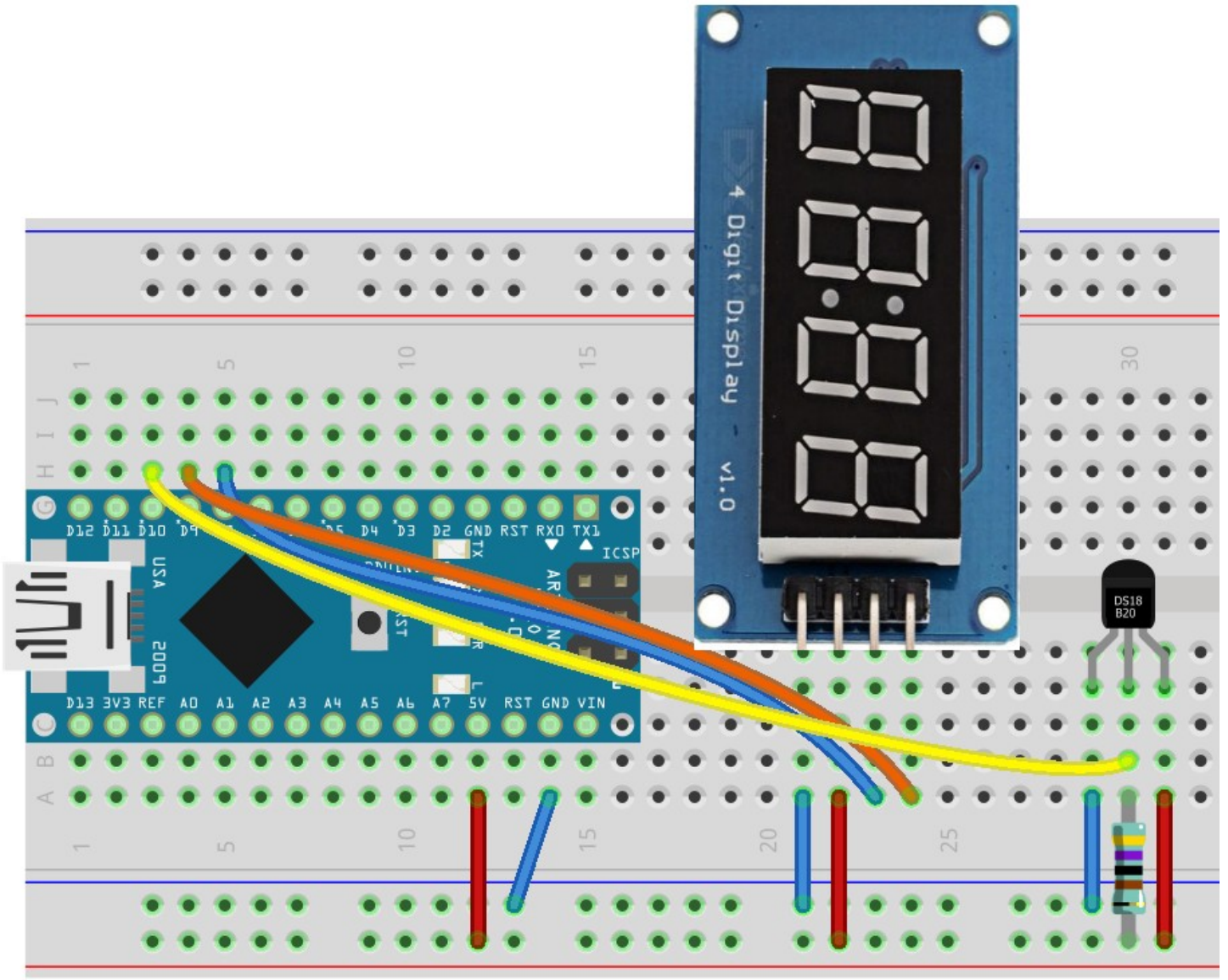


Figure 9. DS18B20 Memory Map

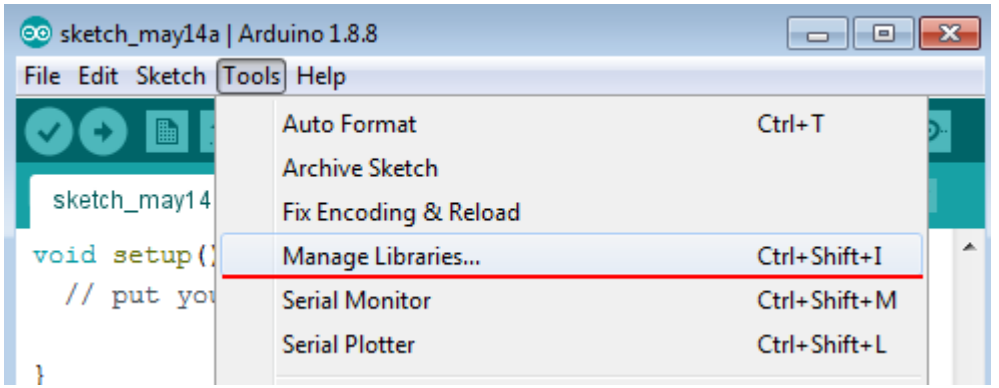
Kapcsolási elrendezés

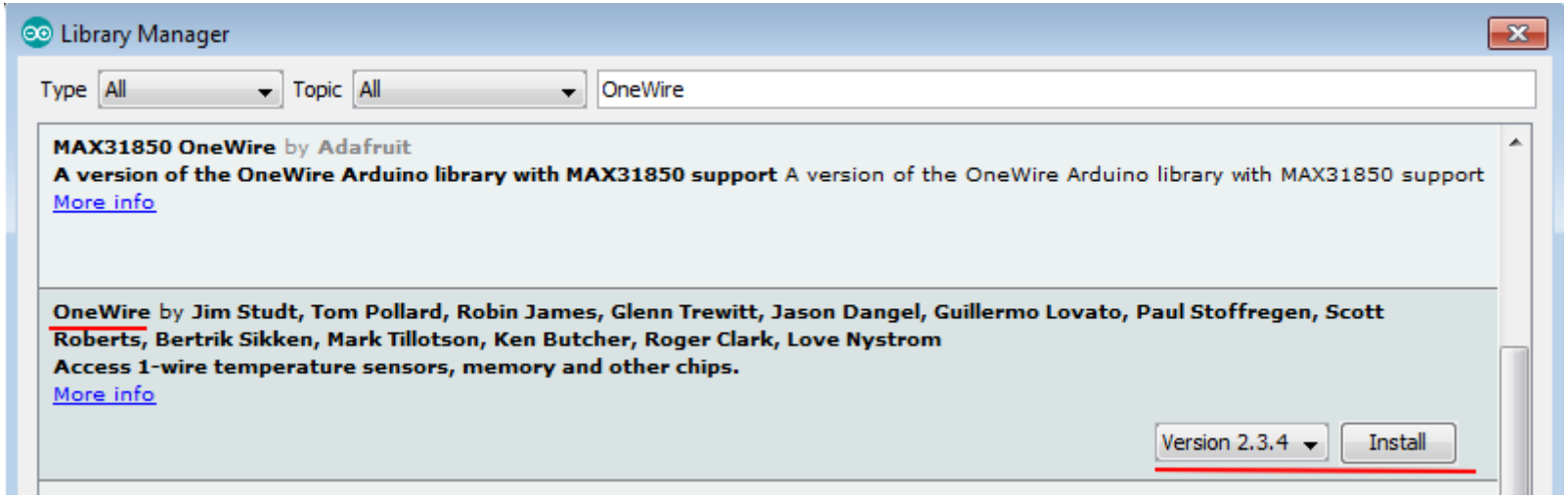
- **Hozzávalók:**
  - Arduino nano
  - TM1637 4-digit k
  - DS18B20 hőmérő
  - 4,7 k ellenállás
  - Breadboard, vezetékek
- **Bekötések:**
  - DIO ⟶ D8
  - CLK ⟶ D9
  - 1-Wire ⟶ D10

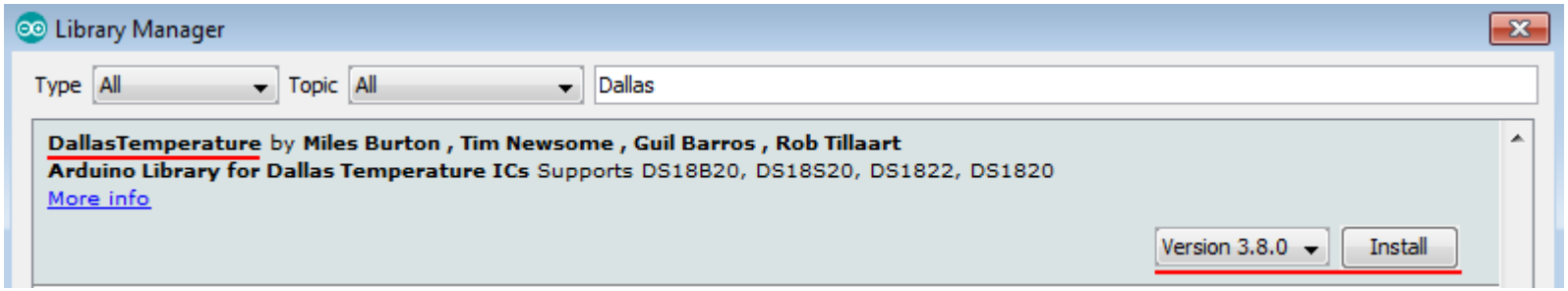


Programkönyvtárak telepítése

- Menüpont: Tools/Manage Libraries
- Keressük meg és telepítsük a **OneWire** és a **Dallas Temperature** programkönyvtárakat!







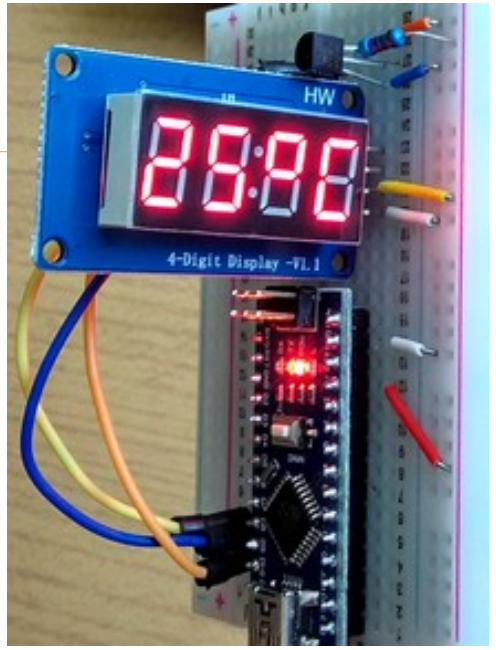
TM1637_DS18B20_2.ino

- Ebben a programban egyetlen hőmérőt és egy 4-digites TM1637 kijelzőt használunk

```
#include <TM1637Display.h>            // TM1637Display 1.2.0 (Avishay Orpaz)
#include <OneWire.h>                  // OneWire 2.3.4 (Paul Stoffregen)
#include <DallasTemperature.h>        // DallasTemperature 3.8.0 (Miles Burton)

#define ONEWIREPIN 10                 // Az 1-wire busz a D10 lábra csatlakozik
#define DIOPIN 8                      // A kijelző DIO kivezetése a D8 lábra csatlakozik
#define CLKPIN 9                      // A kijelző CLK kivezetése a D8 lábra csatlakozik
int tempC = 0;                        // hőmérséklet
const byte DEGREE = SEG_A | SEG_B | SEG_G | SEG_F;  // fok jel szegmensrajzolata
const byte CELSIUS = SEG_A | SEG_F | SEG_E | SEG_D; // C betű szegmensrajzolata

OneWire oneWire(ONEWIREPIN);           // 1-wire busz példányosítása
DallasTemperature sensors(&oneWire);   // DS18B20 hőmérő példányosítása
TM1637Display display(CLKPIN, DIOPIN);// kijelző objektum példányosítása

void setup() {
  sensors.begin ();                    // a szenzor(ok) inicializálása
  display.setBrightness(0x0F);         // Kijelző bekapcsolás és max. fényerő beállítása
}
```

Folytatás a következő oldalon ...

TM1637_DS18B20_2.ino



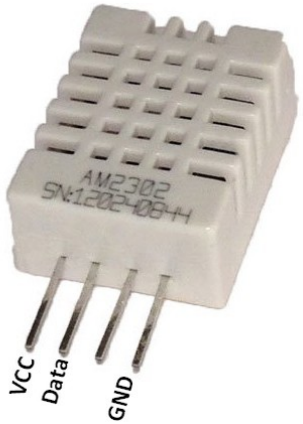
- Egyszerűsítsünk: egyedi azonosítók kezelése helyett a legelső hőmérőt olvassuk ki!
- Kiírásnál a speciális jelek (°C kiírása) miatt a szegmenseket kell vezérelnünk (**setSegments**)
- A számjegyek kódját az **encodeDigit()** függvény segítségével készen kapjuk

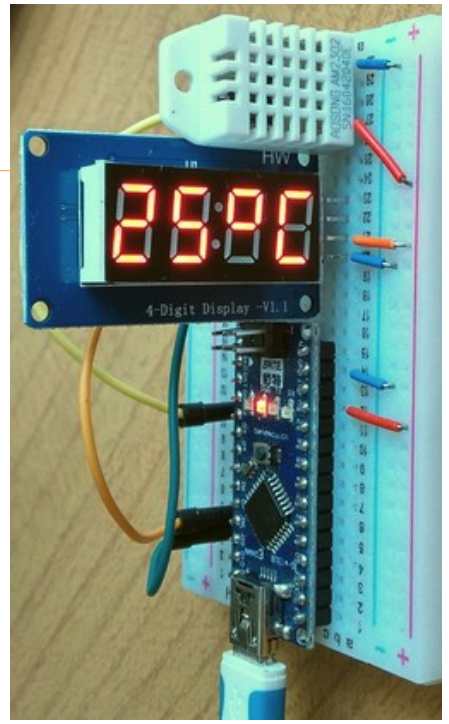
```
void loop() {
  byte data[4];
  sensors.requestTemperatures();          // Hőmérsékletmérés parancs kiküldése
  tempC = sensors.getTempCByIndex(0);     // Az első hőmérő kiolvasása
  int d1 = tempC % 10;                    // egyesek
  int d0 = (tempC / 10) % 10;             // tízesek
  data[0] = display.encodeDigit(d0);      // szegmensrajzolat elővétele
  data[1] = display.encodeDigit(d1);      // szegmensrajzolat elővétele
  data[2] = DEGREE;                       // fok jel
  data[3] = CELSIUS;                      // nagy C szegmensei
  display.setSegments(data);              // a szegmensek megjelenítése
  delay(2000);
}
```

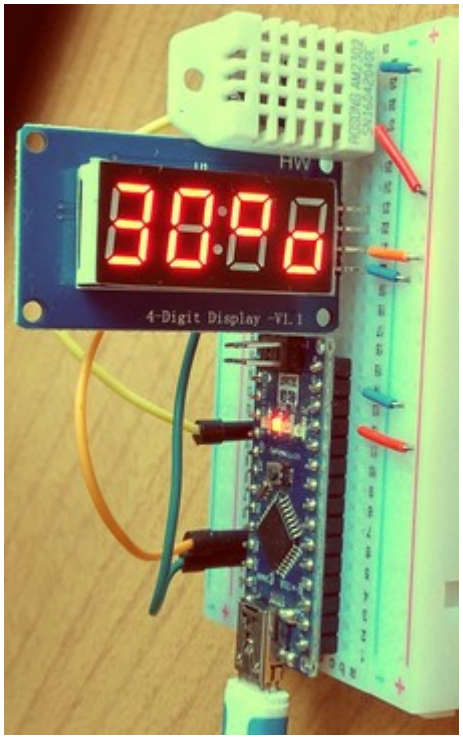
Hőmérő DHT22 szenzorral

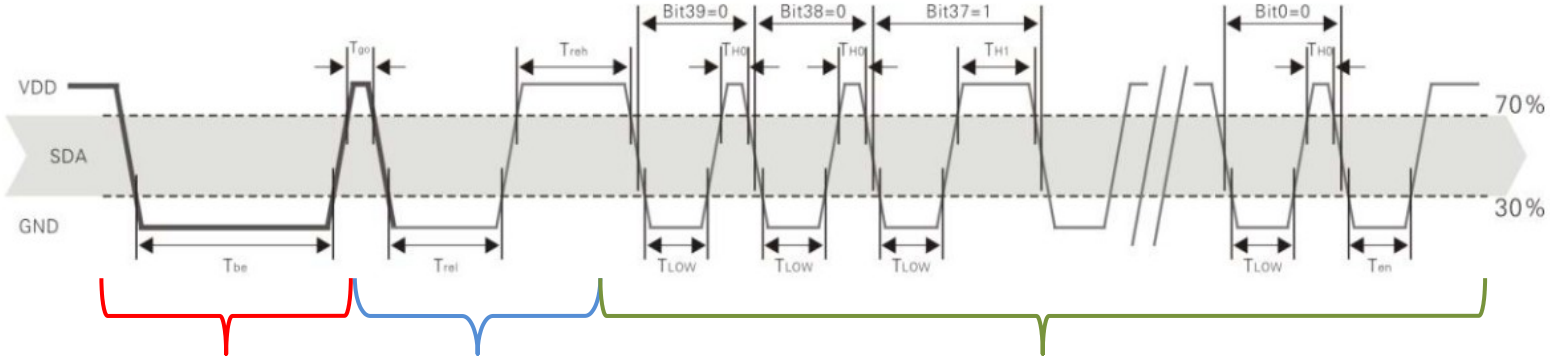
Az Am2302 (DHT22) szenzor főbb jellemzői:

- Hőmérséklet és rel. páratartalom mérésére
- Felbontás: 0.1 ^{0}C, illetve 0.1 %
- 1-wire, nem szabványos protokol
- Mintavételezési gyakoriság: 2 s
- Tápfeszültség: 3,3 – 6 V
- Adatlap: sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf







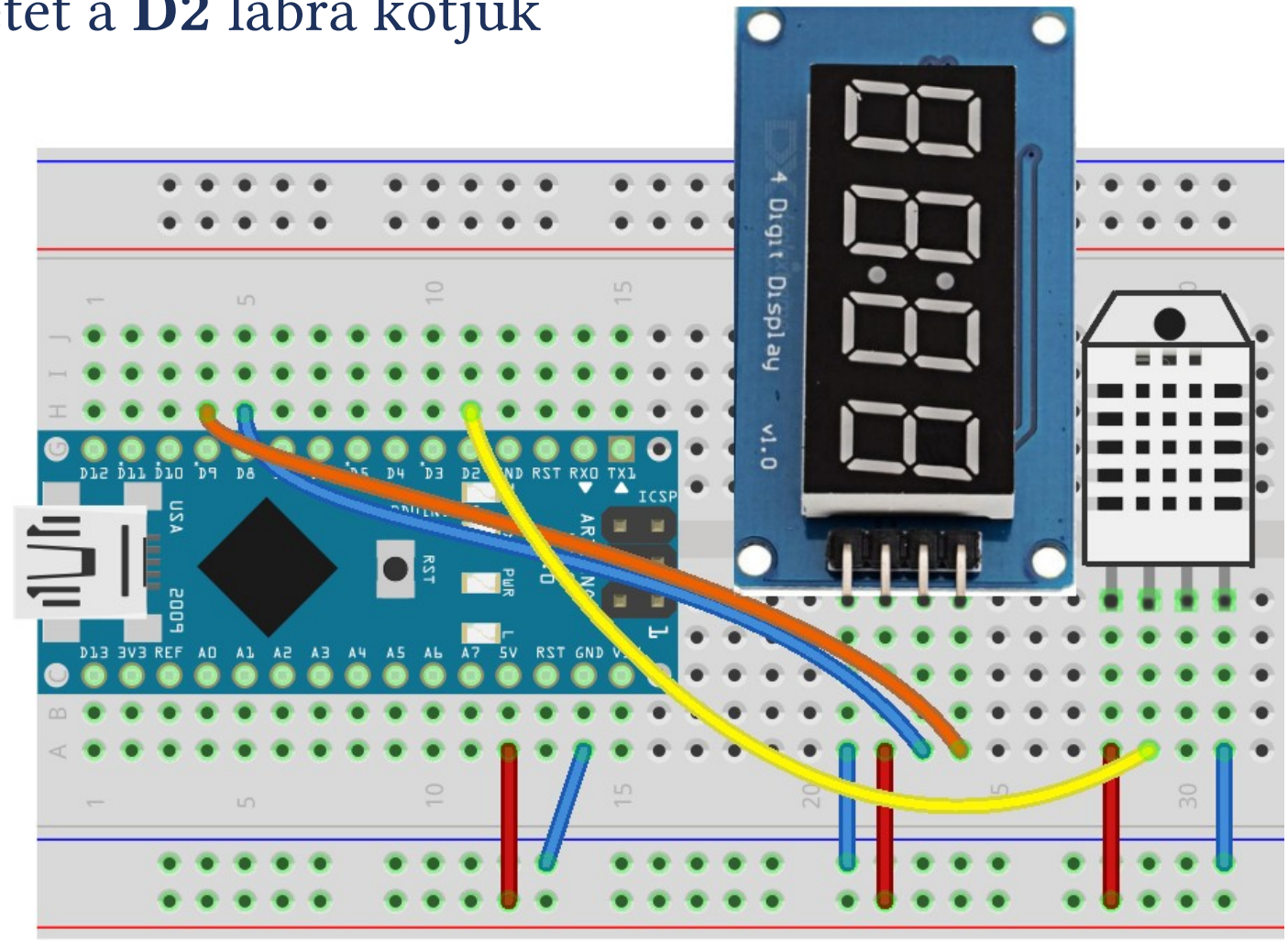


Host indítójel | Szenzor nyugtázó jel | 40 bitnyi adat: 32 bit információ + 8 bit ellenőrző összeg

Összesen tehát 85 időzítést tartalmaz egy-egy tranzakció …

Kapcsolási elrendezés

- A **TM1637** kijelzőt a korábbiakhoz hasonlóan kötjük be (CLK ⟶ D9, DIO ⟶ D8)
- A **DHT22** kimenetét a **D2** lábra kötjük
- Kék sín: GND
- Piros sín: +5 V



DHT programkönyvtár

- A **DHT** könyvtár eredetileg az Arduinohoz készült, az általam módosított változat az Energia IDE-hez is használható
- Telepítés: a Vázlatfüzet (Sketchbook) „libraries” mappájába kell bemásolni (DHT.h és DHT.cpp állományok)

A DHT.h kivonata

```
class DHT {                                          //A DHT objektum deklarálása
 private:
  uint8_t data[6];
  uint8_t _pin, _type;
  boolean read(void);
  unsigned long _lastreadtime;
  boolean firstreading;

 public:
  DHT(uint8_t pin, uint8_t type);                    //Konstruktor
  void begin(void);                                  //Inicializálás
  float readTemperature(bool S=false);               //T kiolvasás
  float convertCtoF(float);                          //C ➔ F konverzió
  float readHumidity(void);                          //RH kiolvasás
};
```

TM1637_DHT22_2.ino

```
#include <TM1637Display.h>
#include "DHT.h"
#define DHTPIN 2          // A DHT szenzor a D2 lábra csatlakozik
#define DIOPIN 8          // A kijelző DIO kivezetése a D8 lábra csatlakozik
#define CLKPIN 9          // A kijelző CLK kivezetése a D8 lábra csatlakozik
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22     // DHT 22  (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

const byte DEGREE = SEG_A | SEG_B | SEG_G | SEG_F;   // fok jel szegmensrajzolata
const byte CELSIUS = SEG_A | SEG_F | SEG_E | SEG_D;  // C betű szegmensrajzolata
const byte O_SMALL = SEG_C | SEG_D | SEG_E | SEG_G;  // kis O betű szegmensrazziolata
const byte HIBA[] = {                                // HIbA felirat szegmensrajzolata
  SEG_B | SEG_C | SEG_E | SEG_F | SEG_G,
  SEG_B | SEG_C ,
  SEG_C | SEG_D | SEG_E | SEG_F | SEG_G,
  SEG_A | SEG_B | SEG_C | SEG_E | SEG_F | SEG_G,
};

DHT dht(DHTPIN, DHTTYPE);                    // Szenzor objektum példányosítása
TM1637Display display(CLKPIN, DIOPIN);       // kijelző objektum példányosítása
byte data[] = {0, 0, 0, 0};                  // adattár a kiíráshoz
void setup() {
  dht.begin ();                              // a szenzor inicializálása
  display.setBrightness(0x0F);               // maximális fényerő beállítása
}
```

Folytatás a következő oldalon ...

TM1637_DHT22_2.ino

```
void loop() {
  float h = dht.readHumidity();           // páratartalom kiolvasása (százalék)
  float t = dht.readTemperature();        // hőmérséklet kiolvasása (Celsius)
  if (isnan(t) || isnan(h)) {             // Hibás kiolvasás: NaN (not a number)
    display.setSegments(HIBA); delay(2000);  // HIBA kiírása
  }
  else {
    int tempC = int(t + 0.5);             // egészre kerekítünk
    int d1 = tempC%10;                    // egyesek
    int d0 = (tempC/10)%10;               // tízesek
    data[0] = display.encodeDigit(d0);    // szegmensrajzolat elővétele
    data[1] = display.encodeDigit(d1);    // szegmensrajzolat elővétele
    data[2] = DEGREE;                     // fok jel
    data[3] = CELSIUS;                    // nagy C szegmensei
    display.setSegments(data);            // a szegmensek megjelenítése
    delay(2000);                          // 2 sec várakozás
    int humidity = int(h + 0.5);          // egészre kerekítünk
    d1 = humidity%10;                     // egyesek
    d0 = (humidity/10)%10;                // tízesek
    data[0] = display.encodeDigit(d0);    // szegmensrajzolat elővétele
    data[1] = display.encodeDigit(d1);    // szegmensrajzolat elővétele
    data[2] = DEGREE;                     // fok jel (százalékjelhez)
    data[3] = O_SMALL;                    // kis O   (százalékjelhez)
    display.setSegments(data);            // kiírás a kijelzőre
    delay(2000);                          // 2 sec várakozás
  }
}
```

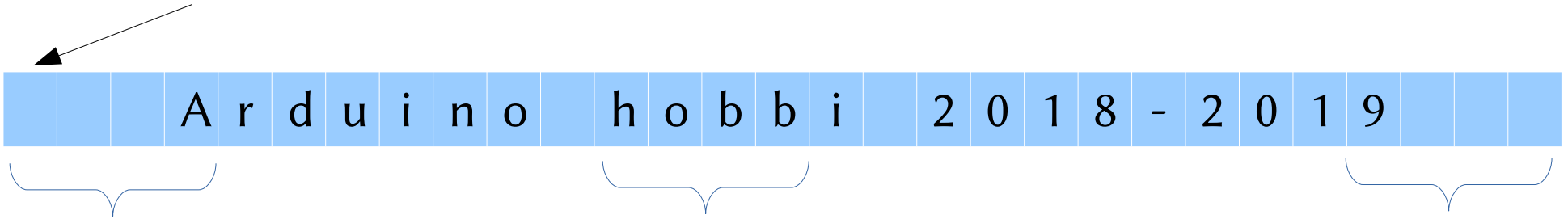
Hőfok kijelzése

Páratartalom kijelzése

Szöveg animáció

- Ha hosszabb a kiírandó szöveg, mint amennyi a kijelzőre fér, akkor görgessük a szöveget!
- A kijelzéshez egy *mutató típusú* változó kell, ami a memória egy adott címére mutat

  `byte *p` ← Ez egy *mutató* típusú változó, ami *byte* típusú adatra mutat
- `byte data[]` ← a tömb neve is mutató, ami az első elem címére mutat



`p = data + 0` `p = data + 11` `p = data + 25`

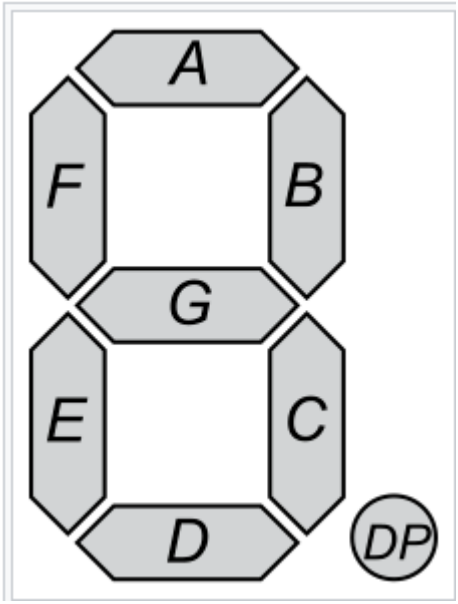
- A **TM1637** kijelzőnk esetében egy négykarakteres ablakot tologatunk végig a szövegen, majd kezdjük elölről
- A tömb végén az első 3 karaktert meg kell ismételni…

TM1637_animation.ino

A **const** módosítójú változók a programmemóriában tárolódnak!

```
const byte data[] = {
  0,
  0,
  0,
  SEG_A | SEG_B | SEG_C | SEG_E | SEG_F | SEG_G,         // A
  SEG_E | SEG_G,                                         // r
  SEG_B | SEG_C | SEG_D | SEG_E | SEG_G,                 // d
  SEG_C | SEG_D | SEG_E,                                 // u
  SEG_E,                                                 // i
  SEG_C | SEG_E | SEG_G,                                 // n
  SEG_C | SEG_D | SEG_E | SEG_G,                         // o
  0,
  SEG_C | SEG_E | SEG_F | SEG_G,                         // h
  SEG_C | SEG_D | SEG_E | SEG_G,                         // o
  SEG_C | SEG_D | SEG_E | SEG_F | SEG_G,                 // b
  SEG_C | SEG_D | SEG_E | SEG_F | SEG_G,                 // b
  SEG_E,                                                 // i
  0,
  SEG_A | SEG_B | SEG_D | SEG_E | SEG_G,                 // 2
  SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F,         // 0
  SEG_B | SEG_C,                                         // 1
  SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F | SEG_G,// 8
  SEG_G,                                                 // -
  SEG_A | SEG_B | SEG_D | SEG_E | SEG_G,                 // 2
  SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F,         // 0
  SEG_B | SEG_C,                                         // 1
  SEG_A | SEG_B | SEG_C | SEG_D | SEG_F | SEG_G,         // 9
  0,
  0,
  0,
};
```

Ez lesz a kiírandó szöveg, a 7-szegmenses kijelzőre tervezve

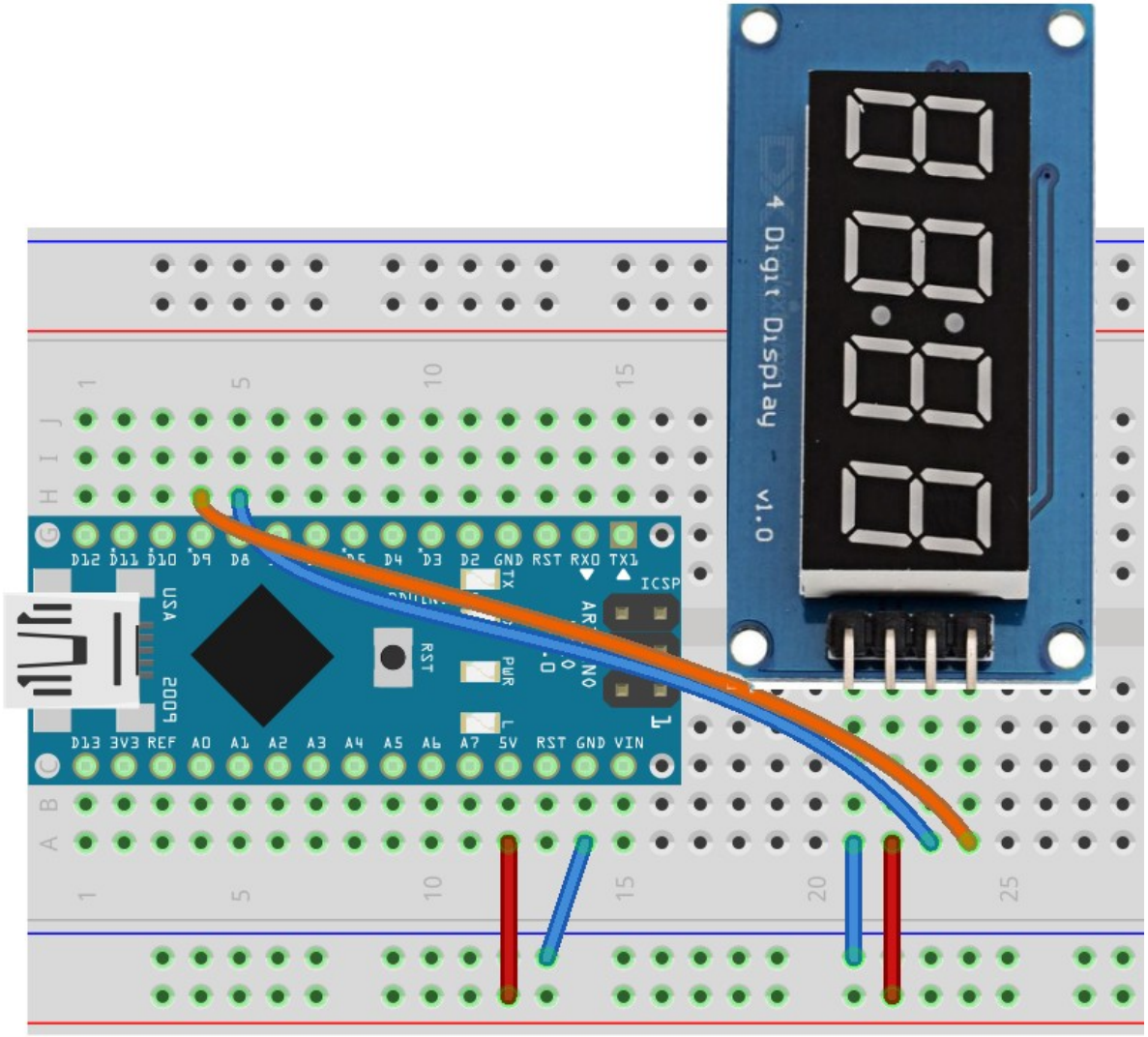


TM1637_animation.ino

```
#include <TM1637Display.h>

#define CLK      9                    // CLK a D9 lábra kötve
#define DIO      8                    // DIO a D8 lábra kötve

const byte data[] = { ... lásd az előző oldalon! ... }


TM1637Display display(CLK, DIO);      // példányosítás és konfigurálás

void setup() {
  display.setBrightness(0x0F);        // Maximális fényerő
}

void loop() {
  byte *p;                            // byte típusú adatra mutató pointer
  for (int i = 0; i < 26; i++) {      // Számlálás 0-tól 25-ig
    p = data + i;                     // Mutató beállítása
    display.setSegments(p);           // 4 karakter kiírása
    delay(500);                       // 500 ms várakozás
  }
}
```

Kapcsolási elrendezés

- **Hozzávalók:**
  - Arduino nano
  - TM1637 4-digit kijelző
  - Breadboard
  - Vezetékek
- A kijelző **CLK** kivezetése a **D9** lábra van kötve
- A kijelző **DIO** kivezetése a **D8** lábra van kötve
- Kék sín: GND
- Piros sín: +5 V



Az Arduino nano kártya kivezetései