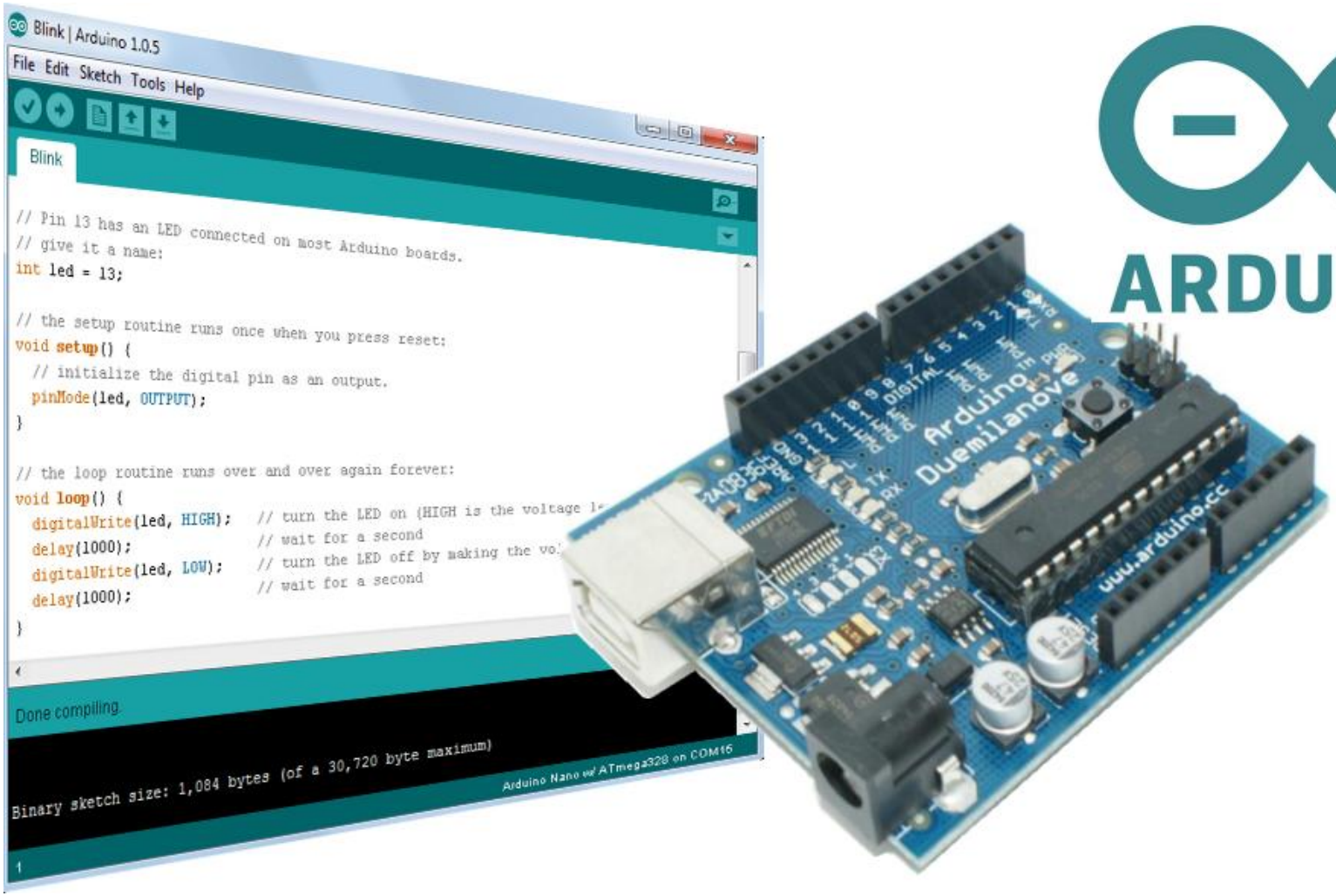


Bevezetés a mikrovezérlők programozásába:

A DS1307 és DS3231 valósidejű órák használata

Lab 22 projektek

- **rtc_baremetal.ino** – egyszerű RTC alkalmazás DS1307 vagy DS3231 modulhoz.
- **RTClib_ds3231.ino** – tesztprogram a DS3231 RTC modul kipróbálásához az RTClib programkönyvtár támogatásával
- **RTClib_ds3231_adjust.ino** – az előző program interaktív beállítási lehetőséggel kibővített változata
- **DS3232RTC_SetSerial.ino** – mintaprogram a DS3232RTC programkönyvtár használatának bemutatására (Time és Streaming programkönyvtár is kell hozzá)
- **DriftCorrectedRTC.ino** – szoftveres pontosítású óra és hőmérő alkalmazás DS1307 RTC modulhoz (az eredeti program Howard Lim Chong Han honlapján található).

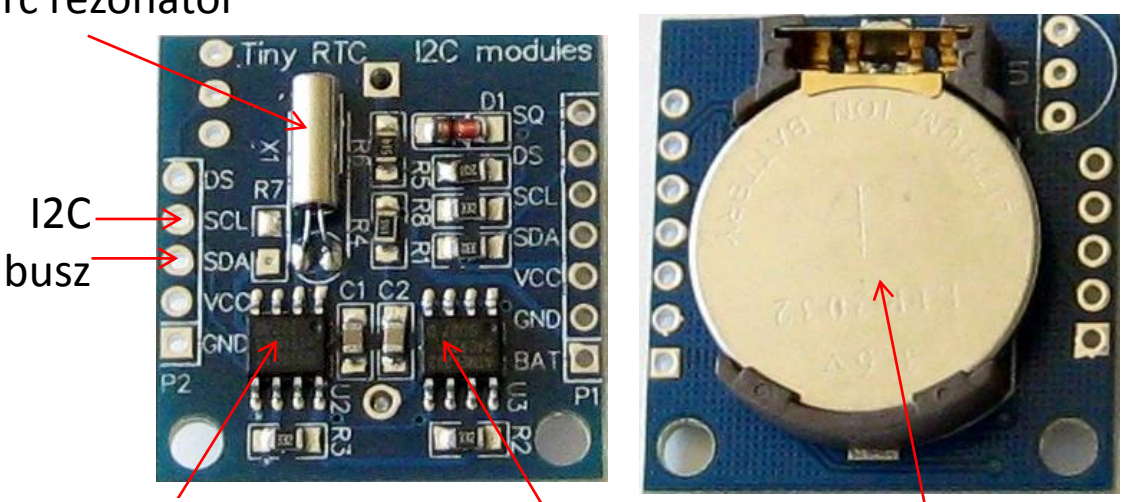
Megjegyzés: A honlapról letölthető **Lab22.zip** állományt az alábbi mappába bontsuk ki:
c:\Fehasználók\<usernév>\Dokumentumok\Arduino

Fontos, hogy a kibontott csomagban található *libraries* almappa tartalmát az alábbi mappába mozgassuk át: c:\Fehasználók\<usernév> \Dokumentumok\Arduino\libraries

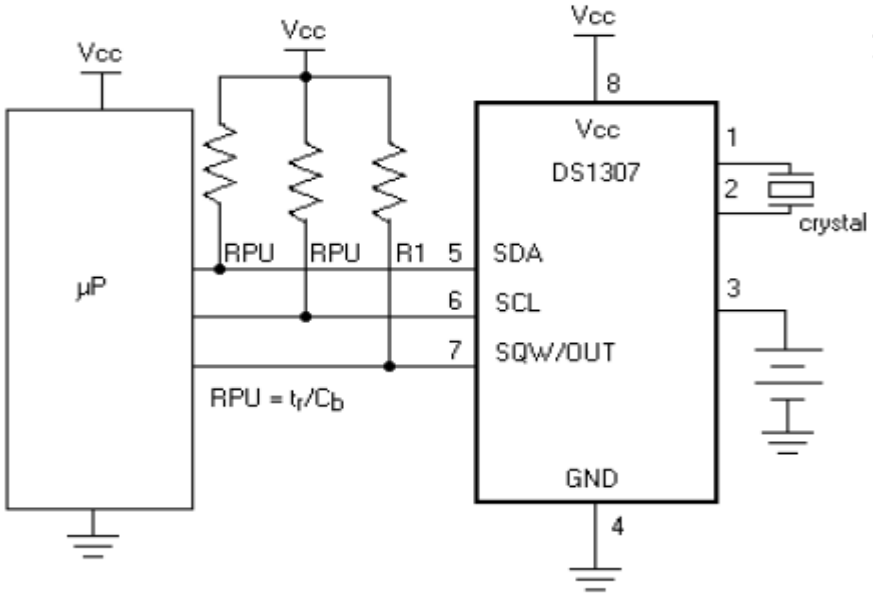
DS1307 RTC kártya

Főbb jellemzők:

- DS1307 RTC IC (MAXIM)
- 32 kHz kvarc
- 3 V telepes háttértáplálás
- I2C busz kommunikáció
- 3,3 – 5 V tápellátás
- 24C32 8 kByte EEPROM
- 56 byte RAM
- < 500nA backup módban



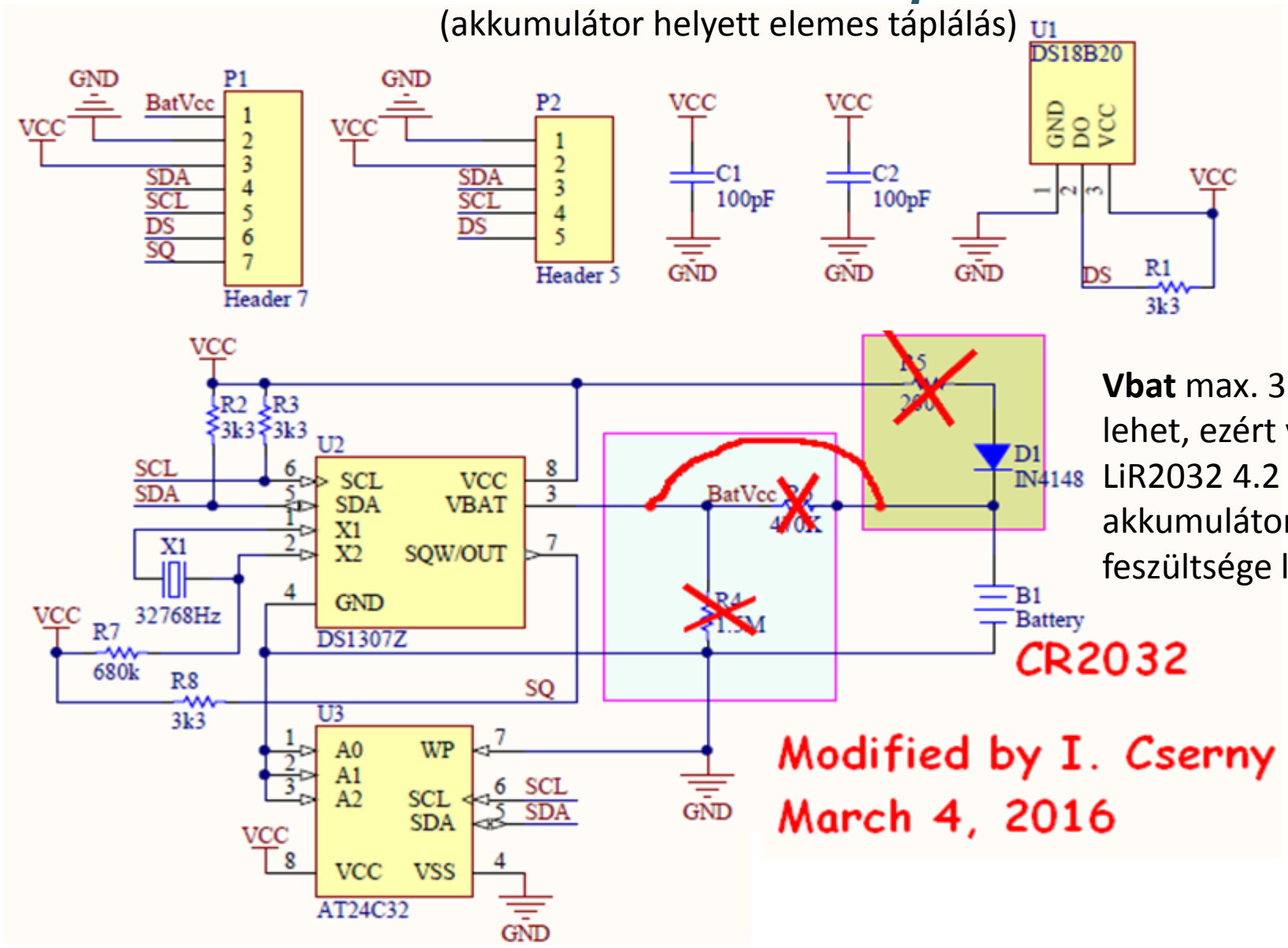
1307 Real-time óra elemes háttértáplálással



A pontosság az alkalmazott kvarckristály és a hőmérséklet függvénye. Szűk hőmérséklet-tartományban ≈ 10 – 20 ppm (naponta 1-1,5 másodperc).

A módosított kapcsolás

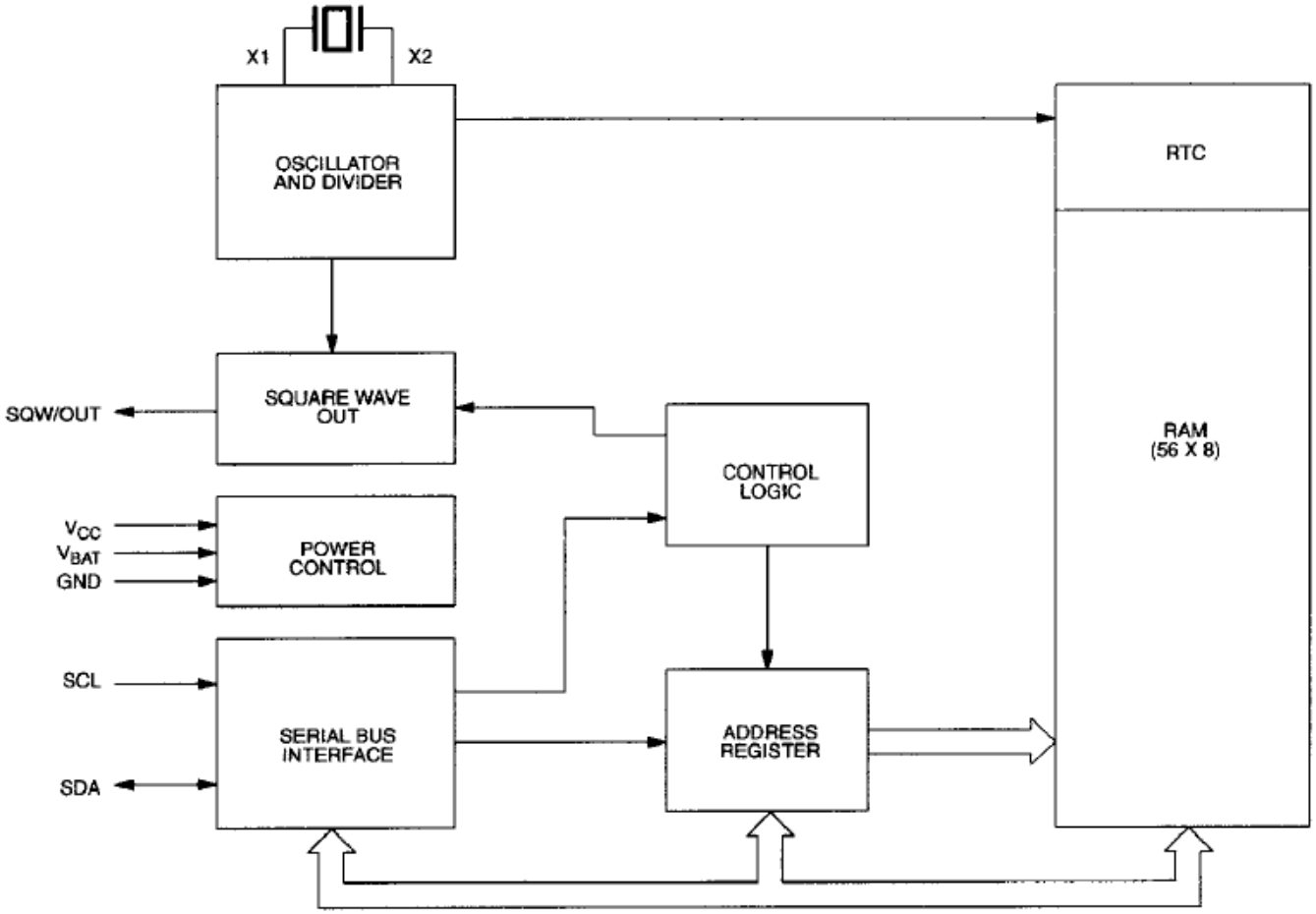
(akkumulátor helyett elemes táplálás)



Vbat max. 3.5 V lehet, ezért volt a LiR2032 4.2 V-os akkumulátor feszültsége leosztva.

DS1307 blokkvázlat

Az oszcillátor backup módban (Vcc < Vbat) is jár, a számláló pedig számlálja és nyilvántartja az időt és a dátumot (óra és kalendárium). A leosztott órajel az SQW kimenetre is kiadható, de mi nem fogjuk használni…



DS1307 regiszter/memória térkép

Az első 8 címen az adatregiszterek és a vezérlő regiszter található.
A fennmaradó 56 bájt adattárolásra (pl. kalibrációs adatok) használható.
Az adatok BCD kódolással (decimális számjegyenként bináris) tárolódnak.

00	SECONDS
01	MINUTES
02	HOURS
03	DAY
04	DATE
05	MONTH
06	YEAR
07	CONTROL
0x8 0x3F	RAM 56 x 8

	BIT7							BIT0	
00H	CH	10 SECONDS			SECONDS				00-59
	0	10 MINUTES			MINUTES				00-59
	0	12/24	10 HR A/P	10 HR	HOURS				01-12 00-23
	0	0	0	0	0	DAY			1-7
	0	0	10 DATE		DATE				
	0	0	0	10 MONTH	MONTH				01-12
	10 YEAR				YEAR				00-99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	

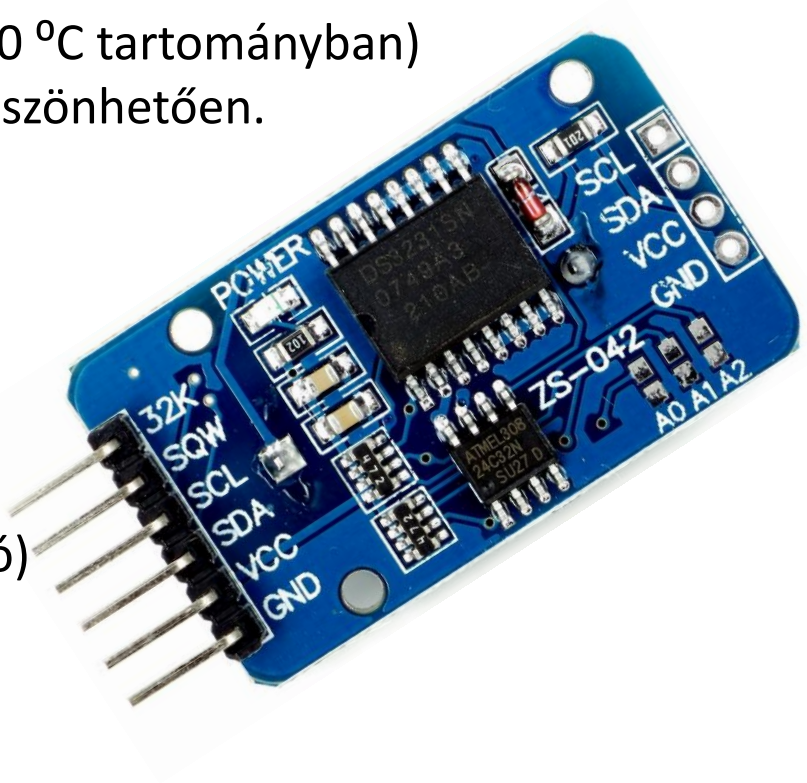
CH – óra leállítás (0: számlál, 1: leállítva)
OUT – kimenet vezérlés (ha SQWE =0)
SQWE – órajel kimenet engedélyezés (0: tilt, 1: enged)
RS1, RS0 – kimeneti frekvencia (00: 1 Hz, 01: 4096 Hz, 10: 8192 Hz, 11: 32768 Hz)

DS3231 Real-time óra modul

Oszcillátor, óra és kalendárium egy tokban. A tápfeszültség megszűnésekor az óra a modul hátoldalán elhelyezett telepről üzemel tovább.

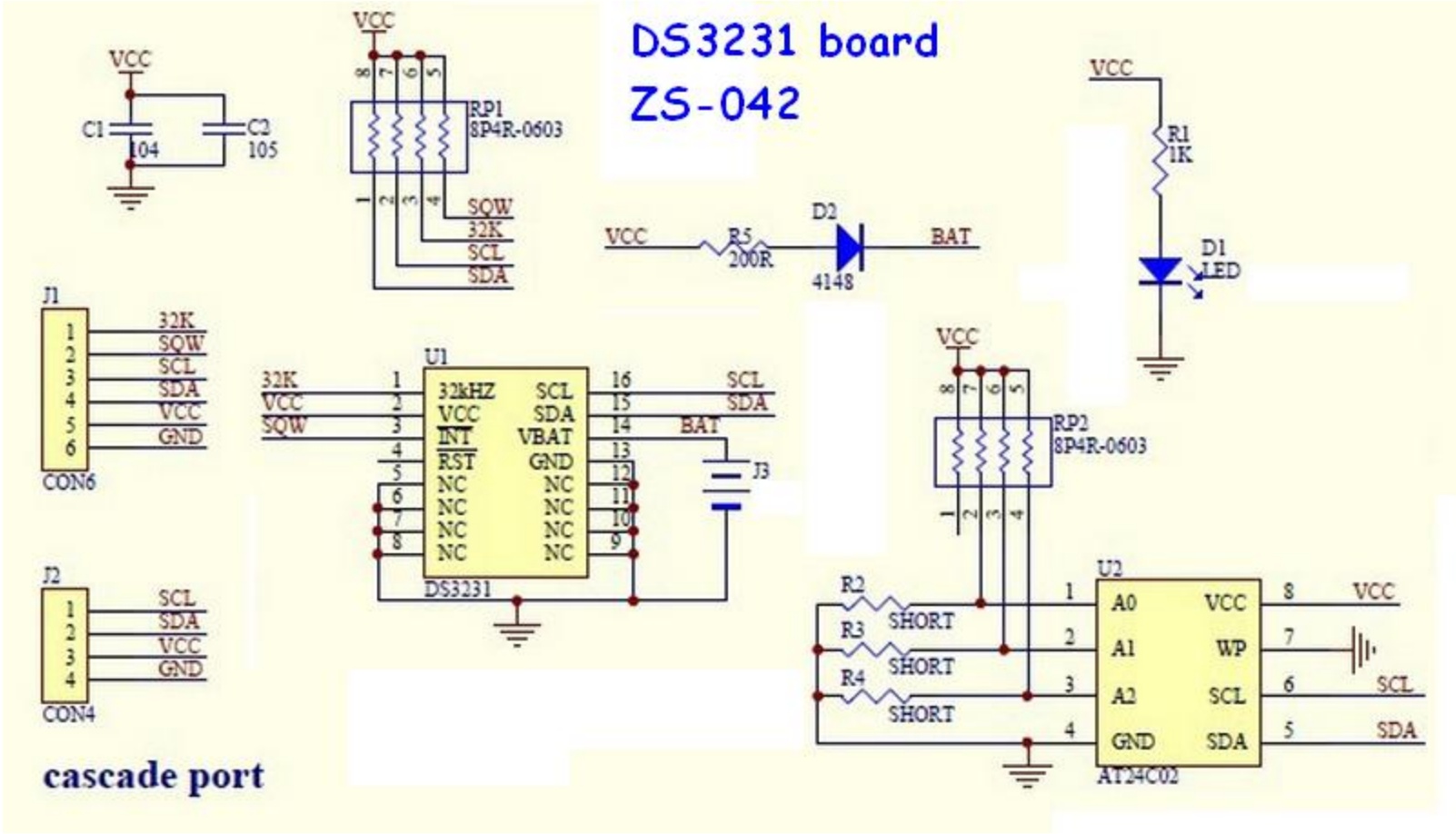
A modul többé-kevésbé cserekompatibilis a ds1307-tel,
ebben is van 4 kB EEPROM, de:

- sokkal pontosabb az óra (±2 ppm a -40 – 80 ^{o}C tartományban) a beépített hőkompenzált oszcillátornak köszönhetően.
- két riasztási időpont is megadható
- van beépített hőmérője
- nincs belső RAM
- Vbat akár 5 V is lehet (4.2 V-os Li akkumulátorról is táplálható!)



EEPROM I2C címe: 0x57 (forrasztással állítható)
DS3231 I2C címe: 0x68

DS3231 kártya kapcsolási rajz



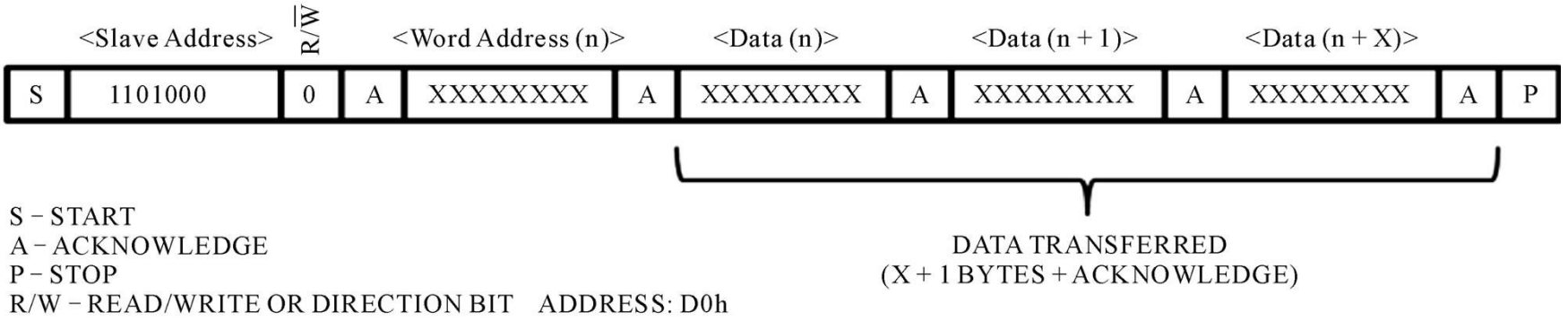
- Az **R5/D2** töltőáramkörnek csak akkor van értelme, ha akkumulátort (pl. **LiR2032**) használunk a **CR2032** elem helyett! Elemes táplálásnál célszerű megszakítani a töltőáramkört.
- A **D1** LED tulajdonképpen fölösleges.
- Az EEPROM címe (**A0, A1, A2**) forrasztással állítható.

A DS3231 IC regiszterei

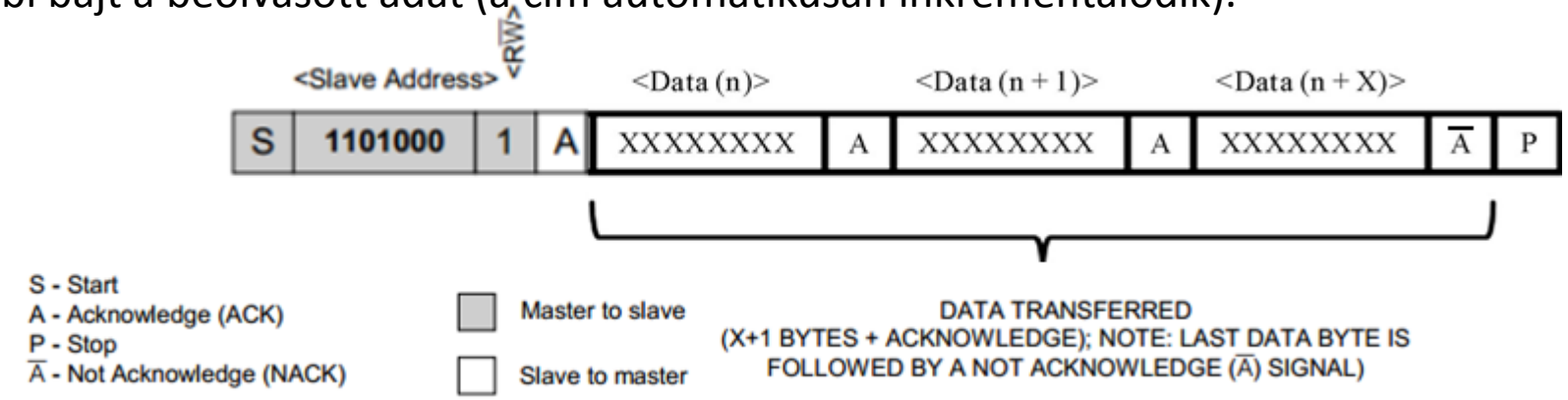
ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
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	—

I2C kommunikáció

Íráskor megcímezzük az eszközt (0x68), az R/W bit pedig = 0.
A második bájt a regiszter (vagy memória) cím. A többi bájt a kiküldendő adat (a cím automatikusan inkrementálódik).



Olvasáskor megcímezzük az eszközt (0x68), az R/W bit pedig = 1.
A többi bájt a beolvasott adat (a cím automatikusan inkrementálódik).



DS1307 és a Mini Pirate program

Az **Arduino Mini Pirate** program egy egyszerű parancsértelmező, amelyet az Arduino kártyára letölthetünk, s melynek segítségével egyszerűen vezérelhetjük, az **Arduino** kártya kivezetéseit, vagy lekérdezhetjük azok állapotát.

A parancskészlet olyan összetett műveleteket is támogat, mint például **I2C** tranzakciók végrehajtása, vagy **PWM** kimenet konfigurálása.

Itt most a **DS1307** regisztereit vesszük szemügyre.

Leírás: www.instructables.com/id/Arduino-comand-line-tool-MiniPirate/

Forráskód: github.com/chatelao/MiniPirate/

Az „i” parancs az I2C Busz lekérdezése, 2 eszközt talál. RTC (0x68) kiválasztása

```
I2C> i
SEARCHING I2C DEVICES...
========================
I2C devices found:
0: 0x57 - 0b01010111
1: 0x68 - 0b01101000

I2C[1 - 0x68] > 1
Device changed to 1

I2C[1 - 0x68] >
```

EEPROM

RTC

Az I2C eszköz kiválasztása a sorszám megadásával történik.

DS1307 és a Mini Pirate program

A regisztereket a w0r8 paranccsal olvashatjuk ki.

```
I2C[1 - 0x68] > w0r8
Requested 8 bytes  got 8 bytes from 0x68
0x00: 0b00000100 0x04 4      4 másodperc
0x01: 0b01000010 0x42 66   42 perc
0x02: 0b00010011 0x13 19   13 óra
0x03: 0b00000111 0x07 7      Szombat
0x04: 0b00000101 0x05 5      5. nap
0x05: 0b00000011 0x03 3      3. hó
0x06: 0b00010110 0x16 22   (20)16. év
0x07: 0b00010011 0x13 19   SQWE
                                  és 32kHz
```

A regiszterek beírása:

```
> w0 0 0x12 0x14 7 5 3 0x16 0
Wrote 9 bytes to 0x68
```

2016-03-05 Sat 14:12:00 beállítása

A 10-nél nagyobb számokat BCD kódolás (Binary Coded Decimal) szerint kell értelmezni!

A 12/24 bit az üzemmód választását jelzi (**0:** 24H mód, **1:** 12H mód).

A 12H módban az **A/P bit** a délelőtt (0), ill. délután (1) jelzésére szolgál.

	BIT7							BIT0
00H	CH	10 SECONDS			SECONDS			
	0	10 MINUTES			MINUTES			
	0	12/24	10 HR A/P	10 HR	HOURS			
	0	0	0	0	0	DAY		
	0	0	10 DATE		DATE			
	0	0	0	10 MONTH	MONTH			
	10 YEAR				YEAR			
07H	OUT	0	0	SQWE	0	0	RS1	RS0

Kézi turkálás – DS3231 és a Mini Pirate

Vegyük szemügyre a DS3231 regisztereit is!

Busz lekérdezése, 2 eszközt talál.
DS3231 (0x68) kiválasztása

```
I2C> i
SEARCHING I2C DEVICES...
=======================
I2C devices found:
0: 0x57 - 0b01010111
1: 0x68 - 0b01101000

I2C[1 - 0x68] > 1
Device changed to 1

I2C[1 - 0x68] >
```

I2C eszköz kiválasztása a sorszám megadásával történik.

Az RTC regiszterek kiolvasása

```
I2C[1 - 0x68] > w0r10
Requested 10 bytes  got 10 bytes from 0x68
0x00: 0b01010010 0x52 82      52 másodperc
0x01: 0b00010110 0x16 22      16 perc
0x02: 0b00010100 0x14 20      14 óra
0x03: 0b00000100 0x04 4       csütörtök
0x04: 0b00100110 0x26 38      26. nap
0x05: 0b00010001 0x11 17      11. hó
0x06: 0b00010101 0x15 21      (20)15. év
0x07: 0b01110010 0x72 114
0x08: 0b01000110 0x46 70
0x09: 0b00000010 0x02 2
```

RTC_baremetal.ino

Egyszerű Arduino program a **DS1307** (vagy **DS3231**) kiolvasására és az aktuális dátum/idő kiírására a soros porton, John Boxall (Tronixstuff) útmutatója és példaprogramja alapján.

```
#include "Wire.h"                          //I2C programkönyvtár
#define  RTC_ADDRESS 0x68                  //I2C eszköz címe
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
     "Thursday", "Friday", "Saturday"};

void setup() {
  Wire.begin();
  Serial.begin(9600);
  //--- set time as: seconds, minutes, hours, day, date, month, year
  //setTime(30,42,21,4,26,11,14);
}

void loop() {
  displayTime();                           // display time/date
  delay(1000);                             // every second
}
```

- Az elemes háttértáplálásnak köszönhetően nem kell minden újraindításkor beírni az RTC regisztereket, ezért van itt megjegyzésben a **setTime()** függvényhívás.
- A **setTime(), getTime()** és utóbbit meghívó **displayTime()** részleteit lásd a következő oldalakon!

RTC_baremetal.ino – folytatás

```
//--- Convert decimal numbers to BCD ----------------------
byte decToBcd(byte val) {
  return( (val/10*16) + (val%10) );
}

//--- Convert BCD numbers to decimal  ---------------------
byte bcdToDec(byte val) {
  return( (val/16*10) + (val%16) );
}

//--- Set RTC time/date -----------------------------------
void setTime(byte second, byte minute, byte hour,
     byte dayOfWeek, byte dayOfMonth, byte month, byte year) {
  Wire.beginTransmission(RTC_ADDRESS);
  Wire.write(0);                        // set register pointer to 00h
  Wire.write(decToBcd(second));         // set seconds
  Wire.write(decToBcd(minute));         // set minutes
  Wire.write(decToBcd(hour));           // set hours
  Wire.write(decToBcd(dayOfWeek));      // set day of week (1=Sunday, 7=Saturday)
  Wire.write(decToBcd(dayOfMonth));     // set date (1 to 31)
  Wire.write(decToBcd(month));          // set month
  Wire.write(decToBcd(year));           // set year (0 to 99)
  Wire.endTransmission();
}
```

Decimálisból BCD-re alakít (beíráshoz)

BCD-ből decimálisra alakít (kiolvasáskor)

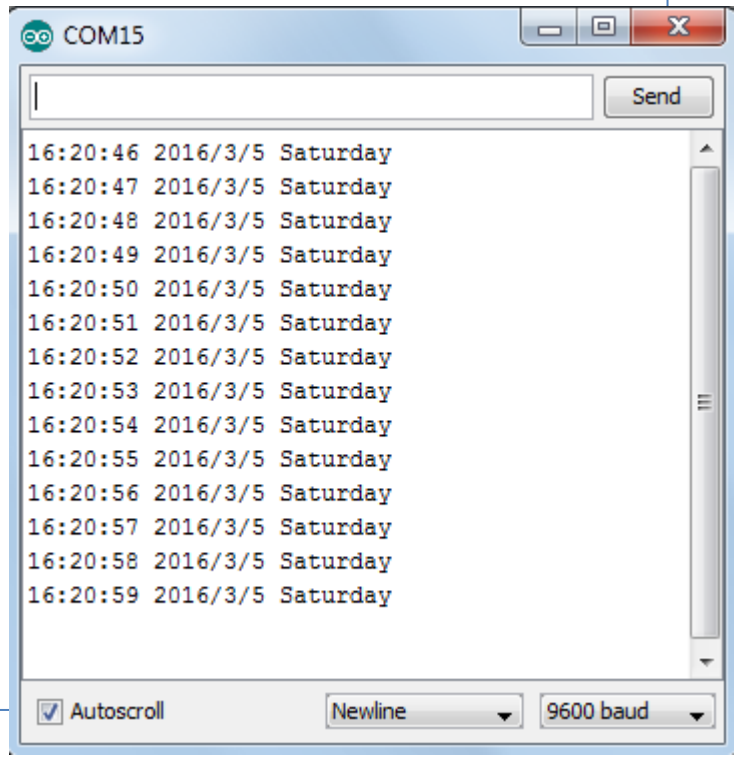
RTC_baremetal.ino – folytatás

```
//--- Read time/date from RTC -----------------------------
void getTime(byte *second, byte *minute, byte *hour,
     byte *dayOfWeek, byte *dayOfMonth, byte *month, byte *year) {
  Wire.beginTransmission(RTC_ADDRESS);
  Wire.write(0);                              //set register pointer to 00h
  Wire.endTransmission();
  Wire.requestFrom(RTC_ADDRESS, 7);           //request 7 bytes starting from 00h
  *second = bcdToDec(Wire.read() & 0x7f); //Don't care CH bit
  *minute = bcdToDec(Wire.read());
  *hour = bcdToDec(Wire.read() & 0x3f);    //Assume 24H mode
  *dayOfWeek = bcdToDec(Wire.read());
  *dayOfMonth = bcdToDec(Wire.read());
  *month = bcdToDec(Wire.read());
  *year = bcdToDec(Wire.read());
}
```

- **getTime()** paraméterei kimenő értékek, ezért mutatóként kell kezelni azokat - elegánsabb volna egy struktúrába szervezni ezeket!
- Kiolvasáskor a kiolvasott adatokat BCD-ből decimális ábrázolásúvá kell alakítani.
- Kiolvasásnál előre meg kell adnunk, hogy hány bájtot akarunk kiolvasni.

RTC_baremetal.ino – folytatás

```
void displayTime() {
  byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
  // retrieve data from DS3231
  getTime(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);
  Serial.print(hour, DEC);
  Serial.print(":");
  if (minute < 10) Serial.print("0");
  Serial.print(minute, DEC);
  Serial.print(":");
  if (second < 10) Serial.print("0");
  Serial.print(second, DEC);
  Serial.print(" ");
  Serial.print(year+2000, DEC);
  Serial.print("/");
  Serial.print(month, DEC);
  Serial.print("/");
  Serial.print(dayOfMonth, DEC);
  Serial.print(" ");
  Serial.println(daysOfTheWeek[dayOfWeek-1]);
}
```



- **dayOfWee**k 1-7 közötti, de az indexeléshez 0-6 kell!
- A **getTime()** hívásnál a & jel segítségével a változók címét adjuk meg (az értékük helyett)
- A perc és másodperc esetében az értéktelen nullát is kiíratjuk.

Támogató könyvtárak – a bőség zavara

Az itt felsoroltakon mellett számos RTC programkönyvtár található az Interneten, de ezek célja és kidolgozottsága nem feltétlenül találkozik az igényeinkkel.

Time programkönyvtár : github.com/PaulStoffregen/Time Arduino rendszeridő és dátum kezelés, szinkronizálás különféle időalap forrásokhoz.

A DS3232RTC_SetSerial projektben használjuk

A számunkra fontos struktúrák:

```
typedef uint32_t time_t;  //Unix idő (1970.január 1-től eltelt másodpercek)

typedef struct  {
  uint8_t Second;
  uint8_t Minute;
  uint8_t Hour;
  uint8_t Wday;                    // a hét napja, vasárnap az 1. nap
  uint8_t Day;
  uint8_t Month;
  uint8_t Year;                    // 1970 óta eltelt évek ;
} tmElements_t;
```

Konverziós függvények:

```
void breakTime(time_t time, tmElements_t &tm);  // break time_t into elements
time_t makeTime(tmElements_t &tm);  // convert time elements into time_t
```

Támogató könyvtárak – a bőség zavara

RTClib programkönyvtár : github.com/adafruit/RTClib Önállóan használható könyvtár, idő és dátum kezelés, DS1307, DS3231 és PCF8523 RTC egyszerűsített (értsd: nincs minden funkció kihasználva) kezelése, időpontok közötti műveletek.

Ez helyettesíti az előző oldalon mutatott `tmElements_t` és `time_t` adattípusokat

Objektumosztályok:

```
class DateTime          //UNIX idő és időelemek közötti konverzióhoz, stb
class TimeSpan          //Időtartam számolásokhoz
class RTC_DS1307        //DS1307 RTC kezelése
class RTC_DS3231        //DS3231 RTC lebutított kezelése
class RTC_PCF8523       //NXP gyártmányú PCF8523 alapú RTC kezelése
```

Az RTC_DS1307 osztály legfontosabb tagfüggvényei:

```
bool begin(void);                   //Inicializálás
void adjust(const DateTime& dt);    //Óra beállítása
bool isrunning(void);               //Működés ellenőrzése
static DateTime now();              //Aktuális idő kiolvasása
```

Az RTC_DS3231 osztály legfontosabb tagfüggvényei:

```
bool begin(void);                   //Inicializálás
void adjust(const DateTime& dt);    //Óra beállítása
bool lostPower(void);               //Tápkimaradás ellenőrzése
static DateTime now();              //Aktuális idő kiolvasása
```

Projektek:
RTClib_ds3231
RTClib_ds3231_adjust

Támogató könyvtárak – a bőség zavara

DS3232RTC programkönyvtár : github.com/JChristensen/DS3232RTC a Time könyvtárral együtt használandó, DS3231 és DS3232 RTC komplex kezelése (ébresztés és hőmérő funkció is). A DS1307 kezeléséhez Michael Margolis DS1307RTC programkönyvtára használható…

Objektumosztályok:

```
class DS3232RTC                 //DS3231 és DS3232 RTC kezelése
```

Az DS3232 osztály legfontosabb tagfüggvényei:

```
static time_t get();                    //Aktuális UNIX idő kiolvasása
static byte read(tmElements_t &tm); //Aktuális idő kiolvasása tmElements_t formában
byte set(time_t t);                     //RTC óra beállítása adott UNIX idő szerint
byte write(tmElements_t &tm);           //RTC óra beállítása
//-- Riasztás típusának és idejének beállítása ------
void setAlarm(ALARM_TYPES_t alarmType, byte sec, byte min, byte hour, byte day);
void alarmInterrupt(byte alarmNumber, bool alarmEnabled); //Interrupt engedélyezés
bool alarm(byte alarmNumber);           //Riasztás állapot lekérdezés és törlés
bool oscStopped(bool clearOSF = true);  //óra állapot lekérdezése és újraindítása
int temperature(void);                  //Hőmérséklet kiolvasása ¼ C fok egységekben
```

Projektek:
DS3232RTC_SetSerial

RTClib_ds3231.ino

```
#include <Wire.h>
#include "RTClib.h"

RTC_DS3231 rtc;

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
                             "Thursday", "Friday", "Saturday"};

void setup () {
  Serial.begin(9600);
  delay(3000);              // wait for console opening
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while(1);
  }

  if (rtc.lostPower()) {
    Serial.println("RTC lost power, lets set the time!");
    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
    // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  }
}
```

Idő és dátum kiíratás **RTClib** felhasználásával
DS3231 RTC órát használunk
DS1307 esetén RTC_DS1307 típust adjunk meg!

Folytatás a következő oldalon…

RTClib_ds3231.ino

```
void loop() {
  DateTime now = rtc.now();
  Serial.print(now.year());
  Serial.print('/');
  Serial.print(now.month());
  Serial.print('/');
  Serial.print(now.day());
  Serial.print(" (");
  Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
  Serial.print(") ");
  Serial.print(now.hour());
  Serial.print(':');
  Serial.print(now.minute());
  Serial.print(':');
  Serial.print(now.second());
  Serial.println();
  delay(3000);
}
```

Kiíratás **2016/3/10 (Thursday) 17:5:13** formátumban.

Folytatás a következő oldalon…

RTClib_ds3231.ino

Értelmetlennek látszó vagánykodás: UNIX idő és egy jövőbeli időpont kiíratása.

```
    Serial.print(" since midnight 1/1/1970 = ");
    Serial.print(now.unixtime());
    Serial.print("s = ");
    Serial.print(now.unixtime() / 86400L);
    Serial.println("d");

//Dátum/idő kiszámítása: 7 nap 12 óra 30 perc és 6 másodperc múlva
    DateTime future (now + TimeSpan(7,12,30,6));
    Serial.print(" now + 7d + 30s: ");
    Serial.print(future.year(), DEC);
    Serial.print('/');
    Serial.print(future.month(), DEC);
    Serial.print('/');
    Serial.print(future.day(), DEC);
    Serial.print(' ');
    Serial.print(future.hour(), DEC);
    Serial.print(':');
    Serial.print(future.minute(), DEC);
    Serial.print(':');
    Serial.print(future.second(), DEC);
    Serial.println();
```

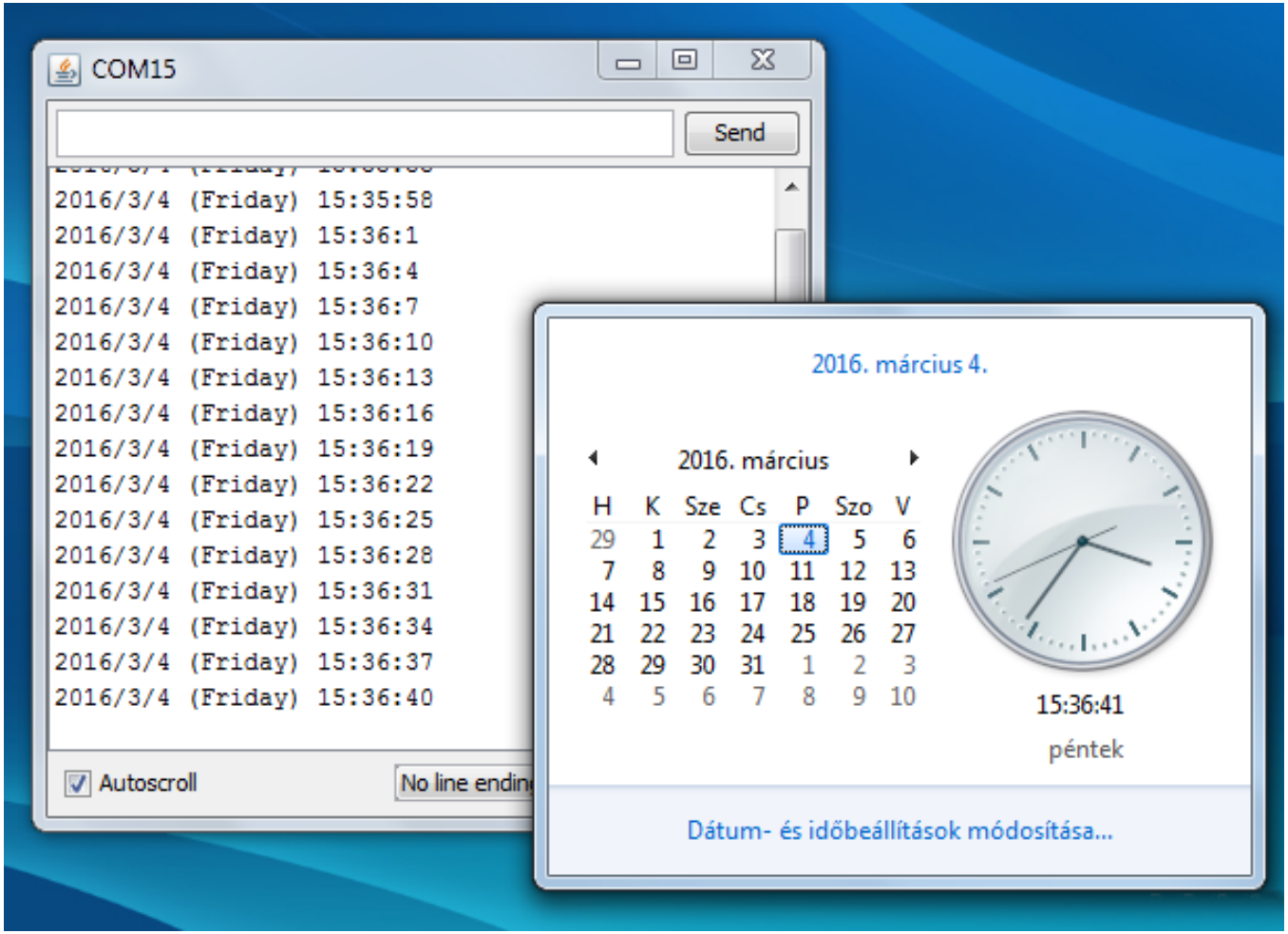
RTClib_ds3231_adjust.ino

```
void loop () { if(Serial.available()) processSerialCommand(); ... stb. ...}
void processSerialCommand() {
  uint16_t y;
  uint8_t m, d, hh, mm, ss;
  char c = Serial.read();
  switch(c){
  case 'T':                    // Command format: TYYMMDDHHMMSS
    delay(100);                // Wait for all data to arrive
    if( Serial.available() >= 12 ){  // process only if all expected data is available
                               // Parse incomming 12 ASCII charaters into y,m,d,hh,mm,ss
      c = Serial.read();        y = c - '0';
      c = Serial.read();        y = 10*y; y += c-'0'; y += 2000;
      c = Serial.read();        m = c - '0';
      c = Serial.read();        m = 10*m; m += c-'0';
      c = Serial.read();        d = c - '0';
      c = Serial.read();        d = 10*d; d += c-'0';
      c = Serial.read();        hh = c - '0';
      c = Serial.read();        hh = 10*hh; hh += c-'0';
      c = Serial.read();        mm = c - '0';
      c = Serial.read();        mm = 10*mm; mm += c-'0';
      c = Serial.read();        ss = c - '0';
      c = Serial.read();        ss = 10*ss; ss += c-'0';
      rtc.adjust(DateTime(y, m, d, hh, mm, ss));
      Serial.print("RTC Set to: ");
    }
    break;
  }
  while(Serial.available()) Serial.read();  // clear serial buffer
}
```

Az előző példaprogramot kiegészítettük az időpont beállításának lehetőségével.

RTClib_ds3231_adjust.ino

A futási eredmény összevetése a számítógép Interneten szinkronizált órájával



DS3232RTC_SetSerial.ino

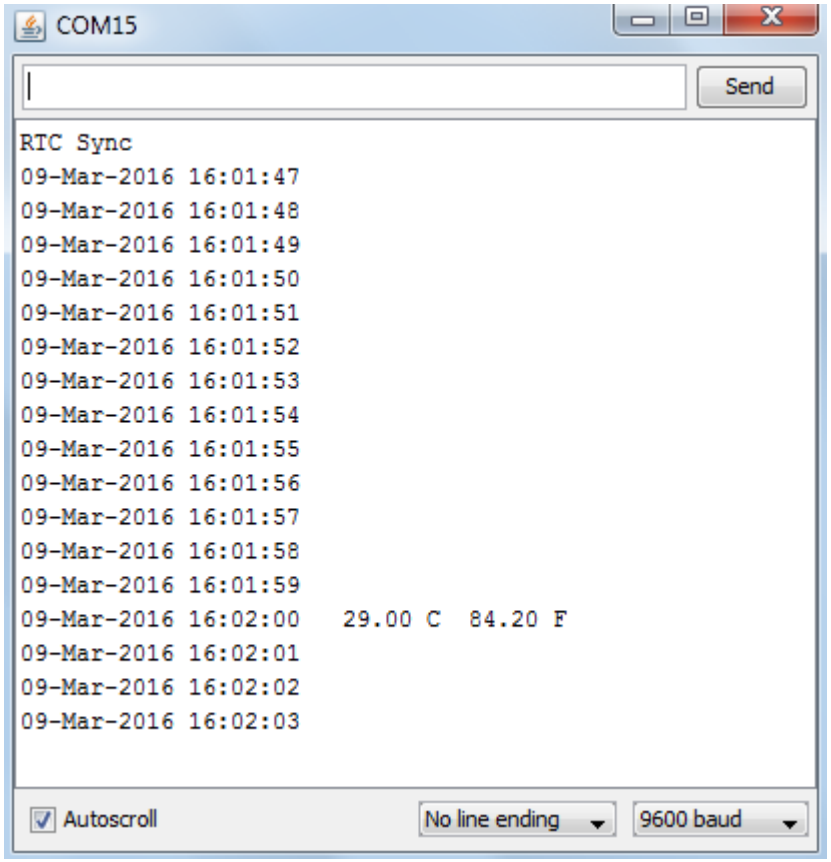
```
#include "DS3232RTC.h"          //http://github.com/JChristensen/DS3232RTC
#include <Streaming.h>          //http://arduiniana.org/libraries/streaming/
#include <Time.h>               //http://playground.arduino.cc/Code/Time
#include <Wire.h>               //http://arduino.cc/en/Reference/Wire

void setup(){
  Serial.begin(9600);
  setSyncProvider(RTC.get);
  Serial << F("RTC Sync");
  if (timeStatus() != timeSet) {
    Serial << F(" FAIL!");
  }
  Serial << endl;
}
```

Folytatás a következő oldalon…

Ez a program hasonló az előzőhöz, de

- Más könyvtárakat használunk (Time, DS3232RTC, Streaming)
- Csak időnként (5 percenként) szinkronizálunk az RTC-hez
- Percenként kiolvassuk és kiírjuk a hőmérsékletet



```
void loop(void) {
  static time_t tLast; time_t t; tmElements_t tm;
  if (Serial.available() >= 12) {
    int y = Serial.parseInt();
    if (y >= 1000) tm.Year = CalendarYrToTm(y); else tm.Year = y2kYearToTm(y);
    tm.Month = Serial.parseInt();
    tm.Day = Serial.parseInt();
    tm.Hour = Serial.parseInt();
    tm.Minute = Serial.parseInt();
    tm.Second = Serial.parseInt();
    t = makeTime(tm);
    RTC.set(t);         //use the time_t value to ensure correct weekday is set
    setTime(t);
    Serial << F("RTC set to: ");
    printDateTime(t);
    Serial << endl;
    while (Serial.available() > 0) Serial.read();
  }
  t = now();
  if (t != tLast) {
    tLast = t;
    printDateTime(t);
    if (second(t) == 0) {
      float c = RTC.temperature() / 4.;
      Serial << F("  ") << c << F(" C  ");
    }
    Serial << endl;
  }
}
```

Folytatás a következő oldalon…

```
void printDateTime(time_t t) {     //print date and time to Serial
    printDate(t);
    Serial << ' ';
    printTime(t);
}

void printTime(time_t t) {         //print time to Serial
    printI00(hour(t), ':');
    printI00(minute(t), ':');
    printI00(second(t), ' ');
}

void printDate(time_t t) {         //print date to Serial
    printI00(day(t), 0);
    Serial << "-" << monthShortStr(month(t)) << "-" << _DEC(year(t));
}

//Print an integer in "00" format (with leading zero)
void printI00(int val, char delim) {
    if (val < 10) Serial << '0';
    Serial << _DEC(val);
    if (delim > 0) Serial << delim;
    return;
}
```

Megjegyzések:

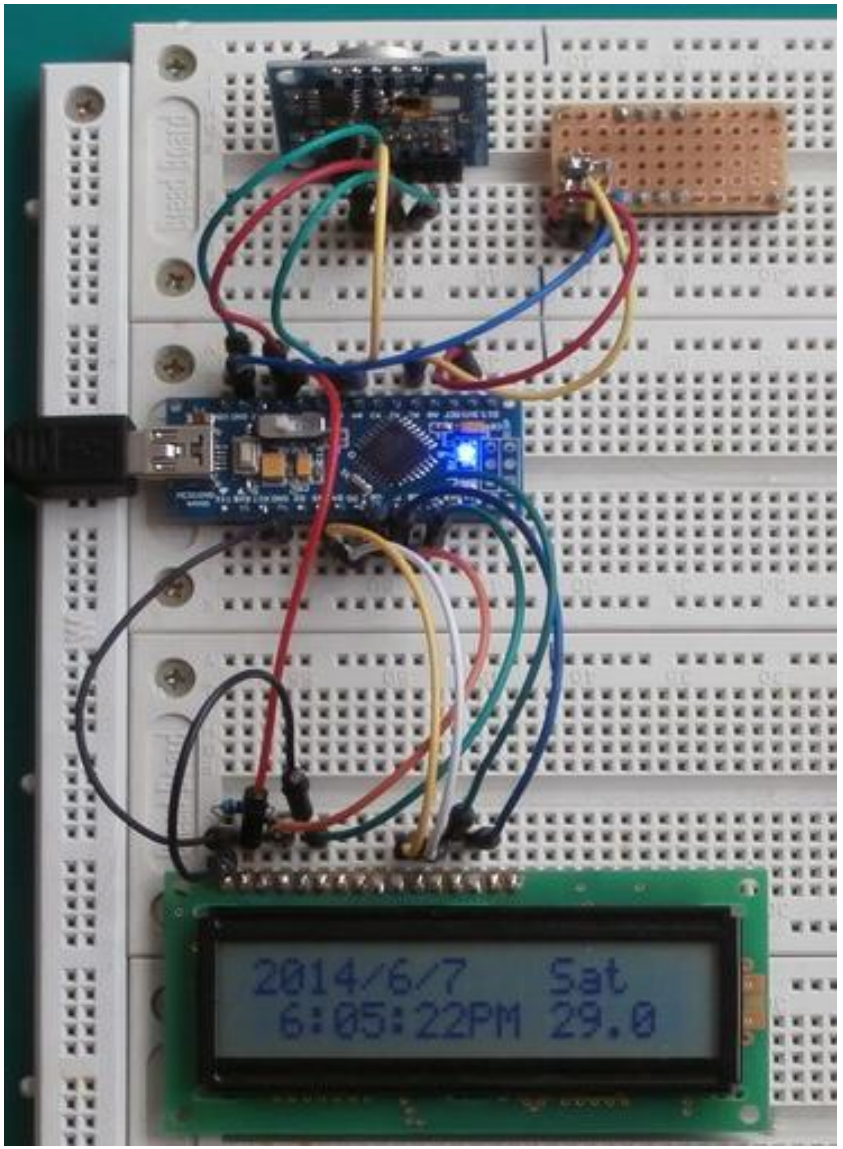
- A C++ stílusú kiíratás nem létszükséglet.
- A számok kiíratása itt mindig két jegyre történik (az értéktelen nullát is kiírjuk)

Drift korrekció a nagyobb pontosságért

A **DS3231 RTC hardveresen támogatja** az elöregedésből vagy más okból pontatlan óra korrigálását az *aging offset register* beírása révén. Az előjeles érték szobahőmérsékleten kb. 0.1 ppm lépésekben lassít (ha pozitív) gyorsít (ha negatív) az oszcillátoron.

A **DS1307 esetében** szoftveresen kell korrigálnunk, melyhez a szükséges adatokat (utolsó órabeállítás időpontja, az óra késésére vagy sietésére vonatkozó adat számlálója (másodpercek) és nevezője (napok) az 56 bájtos NVRAM-ban eltárolható.

Egy komplett mintaalkalmazás (DS1307 RTC, LCD kijelző, MCP9700 analóg hőmérő) a Howard Lim Chong Han honlapján található. Ennek egy utánépített változata a képen látható. A program: **DriftCorrectedRTC.ino**



További lehetőségek

- A **DS3232RTC** programkönyvtár (github.com/JChristensen/DS3232RTC) lehetővé teszi a **DS3231** ébresztő funkciójának felhasználását (Alarm1, Alarm2).

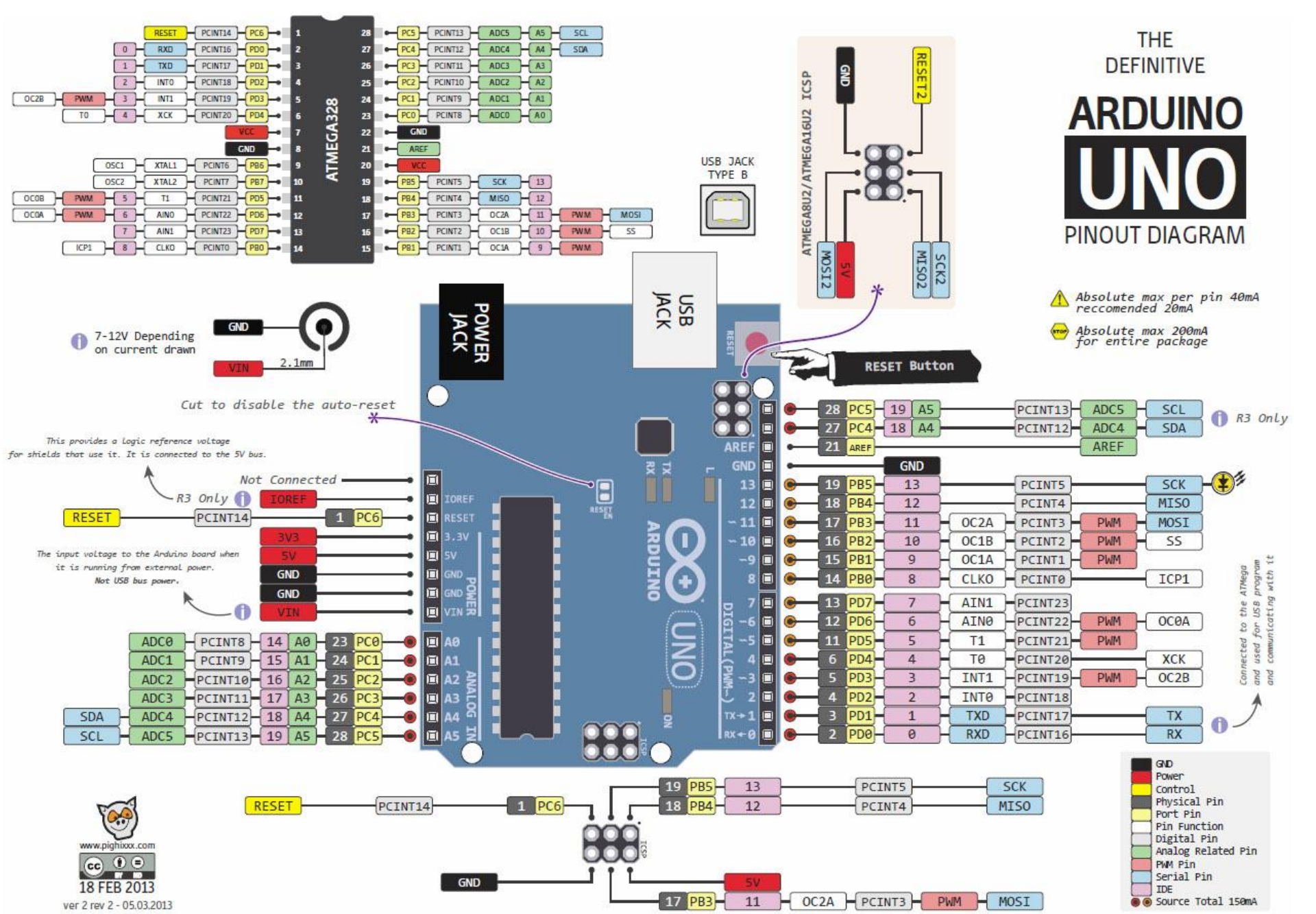
- Az Arduino rendszeróráját nem muszáj szinkronizálni az RTC-vel. Használhatjuk az RTC közvetlen kiolvasását végző `static byte read(tmElements_t &tm);` függvényt is, különösen ha ritkábban kell frissíteni a kijelzést (pl. perckijelzésű óra).

- Soros port helyett **kisfogyasztású kijelzők** (pl. Nokia 5110 LCD kijelző) használata kisfogyasztású (pl. elemes táplálású) óra készítéséhez.

- **Nagyméretű LED kijelző** használata köztéri/beltéri órákhoz.

- RTC óra kombinálása névnap kijelzéssel (ha már a dátum úgyis kéznél van…).

- Szökőév, időzóna, nyári/téli időszámítás kezelése.


THE DEFINITIVE
ARDUINO UNO
PINOUT DIAGRAM
ATMEGA328
ATMEGA8U2/ATMEGA16U2 ICSP
USB JACK TYPE B
POWER JACK
USB JACK
RESET Button
7-12V Depending on current drawn
2.1mm
Cut to disable the auto-reset
This provides a Logic reference voltage for shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
Connected to the ATMega and used for USB program and communicating with it
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

Emlékeztető: Arduino nano v3.0