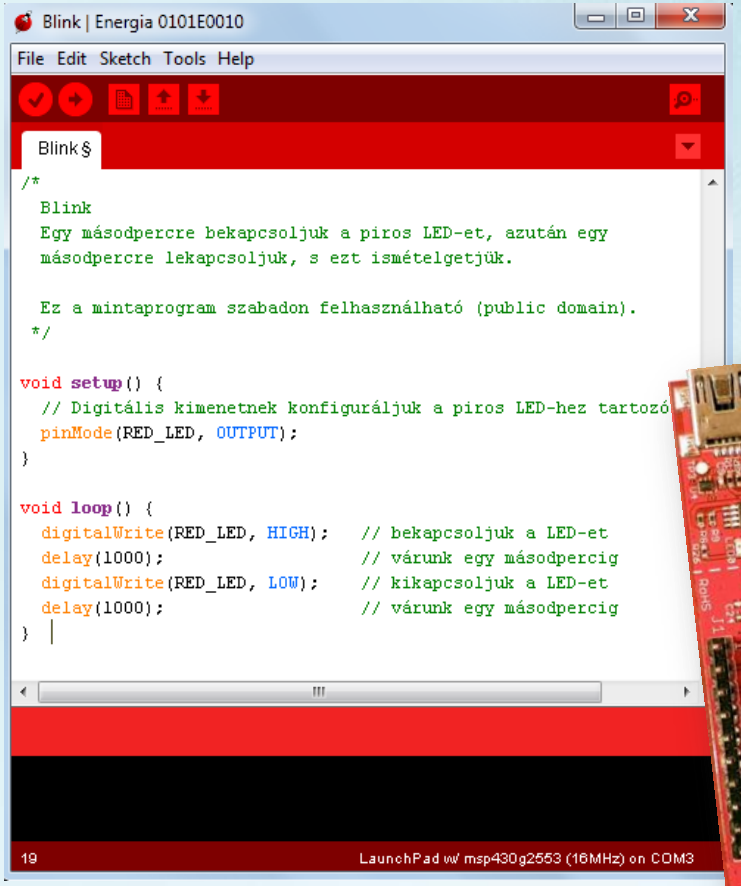
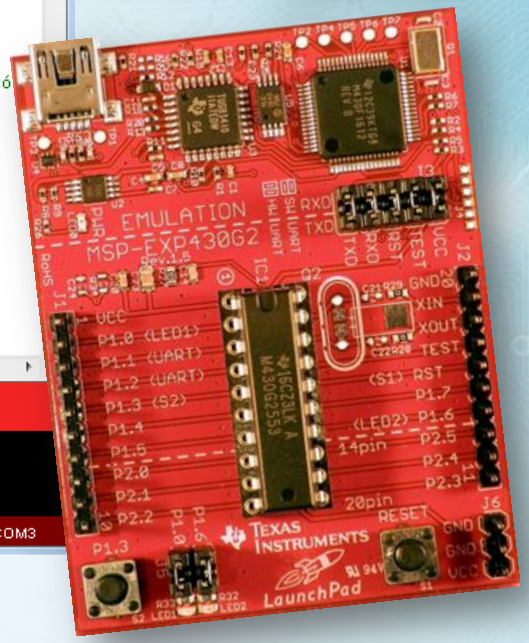




Energīa

MSP430 programozás Energia környezetben

Digitális szenzorok I2C kommunikációval

I2C kommunikáció

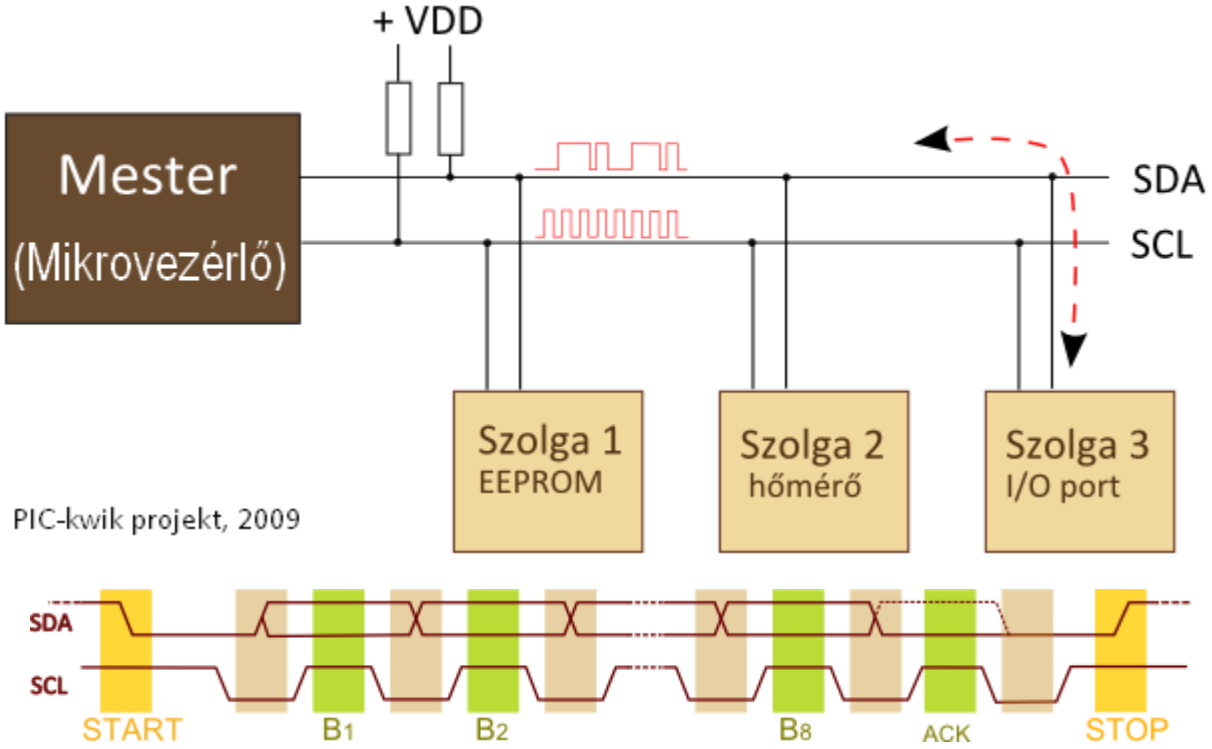
Az **I2C** (Inter-Integrated Circuit = integrált áramkörök közötti) kétvezetékes soros kommunikációs sínt a Philips fejlesztette ki. Az 1990-es évek közepétől számos versenytárs is fejlesztett ki I2C kompatibilis eszközöket.

Az I2C busz néhány jellemzője:

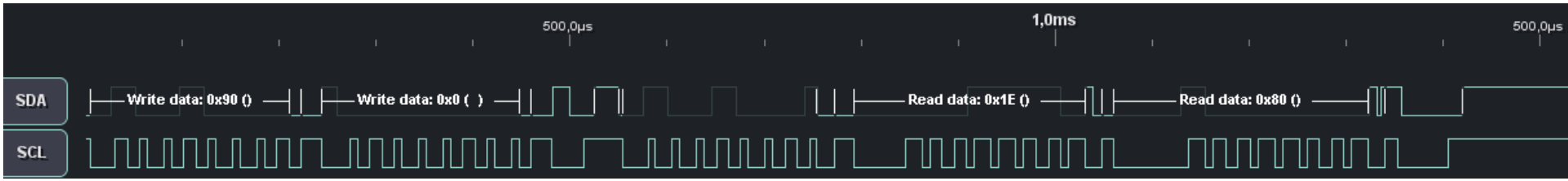
- Két vonalat használ: **SCL** a szinkronjel (órajel), **SDA** pedig kétirányú adat jel.
- A busz akkor szabad, ha egy **STOP** feltételt követően mind az **SDA**, mind az **SCL** vonal magas szinten van (nincs aktív kimenet).
- Soros, 8-bit-es, kétirányú adatforgalom, sebessége tipikusan max 100 kbit/s, vagy 400 kbit/s. Újabb eszközöknél előfordul max. 3,4 mbit/s sebességgel.
- Mindegyik csatlakoztatott eszköz címezhető egy egyedi címmel (7 v. 10 bit).
- Master/slave kapcsolat, melyben a master kezdeményez, vezérel és szolgáltatja az órajelet.
- Bonyolultabb esetekben több master is csatlakozhat a buszra (arbitáció!).

*Leírás: [www.muszeroldal.hu/measurenotes/**i2c**_hu.pdf](www.muszeroldal.hu/measurenotes/i2c_hu.pdf)*

Az I2C busz



Tipikus események az I2C buszon

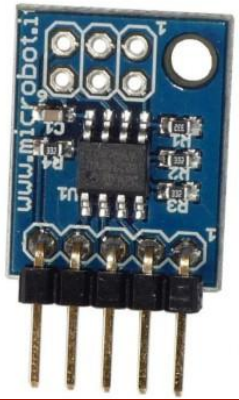


START Slave cím + W regisztercím RESTART Slave cím + R Két adatbájt beolvasása STOP

Lab10

- **TCN75_hw** – TCN75 digitális hőmérő kiolvasása hardveres I2C támogatással
- **TCN75_sw** – TCN75 digitális hőmérő kiolvasása szoftveres I2C támogatással (bit banging)
- **PressureSensor_hw** – BMP180 digitális nyomásmérő és hőmérő kiolvasása (HW I2C)
- **PressureSensor_sw** – BMP180 digitális nyomásmérő és hőmérő kiolvasása (SW I2C)
- **libraries** – SoftwareWire és BMP085_t programkönyvtárak

Hozzávalók:



TCN75 digitális hőmérő

A Microchip **TCN75** digitális hőmérő, illetve termosztát I2C slave eszközként kezelhető, 7 bites címzéssel.

Működési tartomány: -55 °C – +125 °C

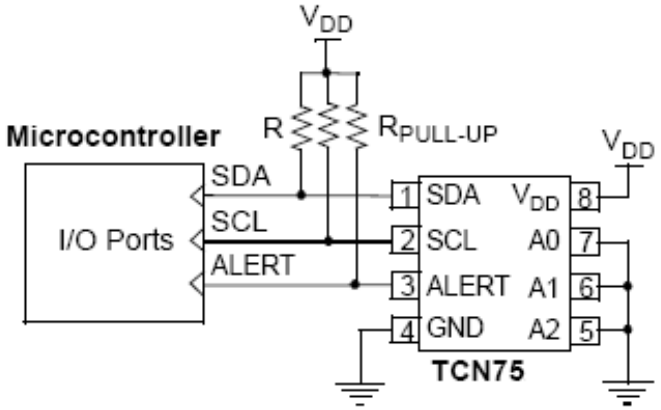
Felbontás: 0.5 °C

Tápfeszültség: 2,7 - 5.5 V

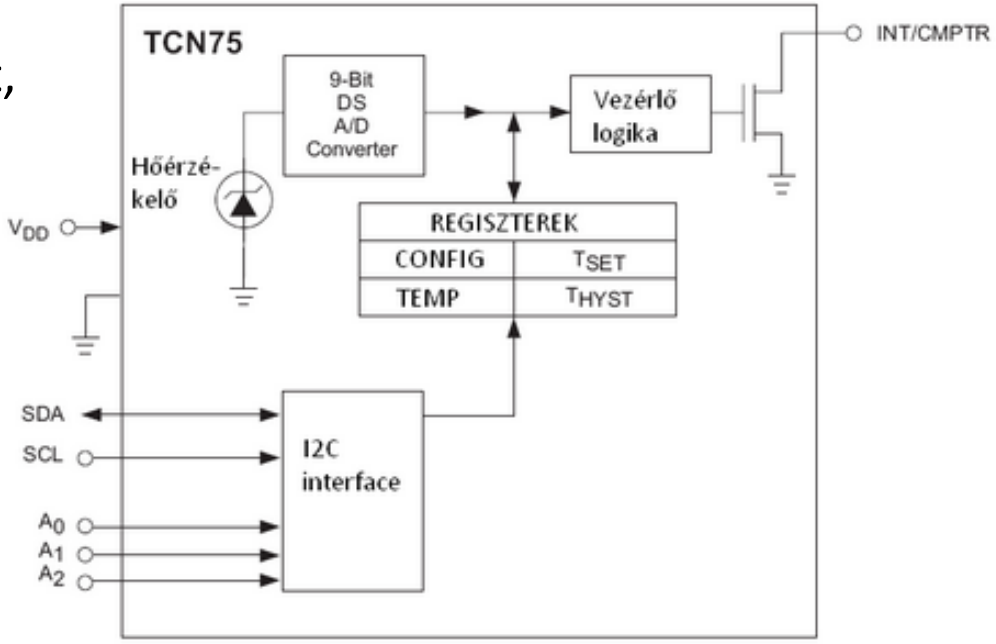
Felhasználás: beépített hőmérőként, Önálló, vagy mikrovezérlőhöz kötött Termosztátként.

TABLE 3-1: TCN75 SLAVE ADDRESS

1	0	0	1	A2	A1	A0
MSB						LSBS



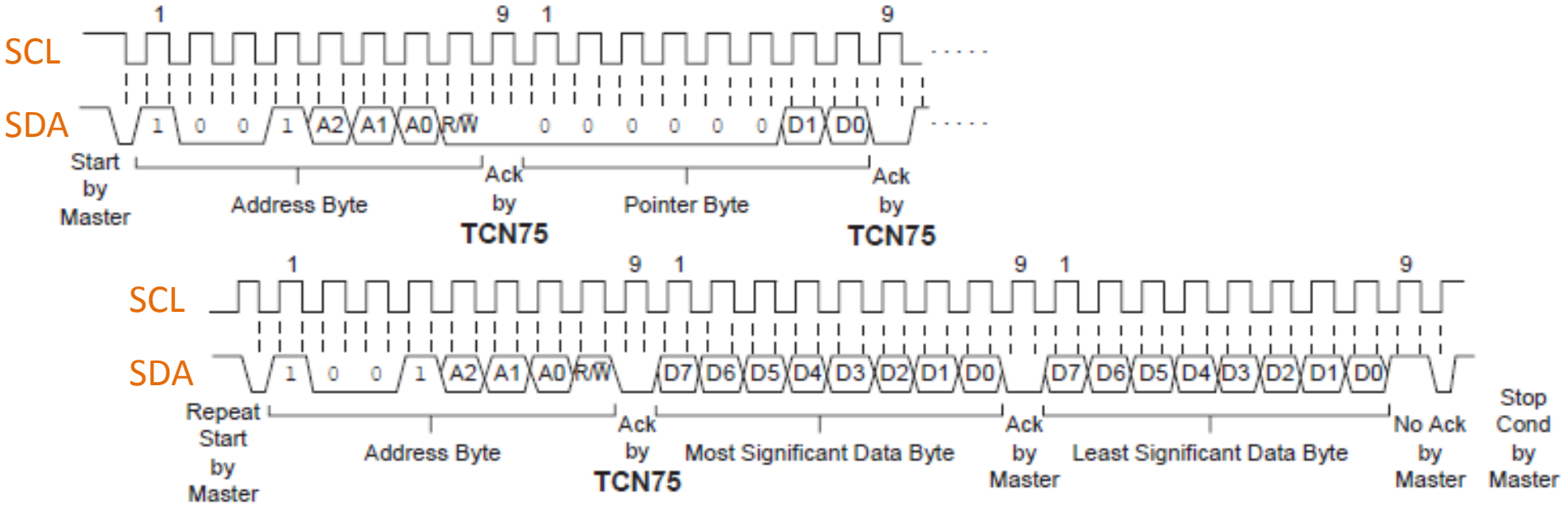
Bekötési rajz



Belső felépítés/programozói modell

TCN75 kiolvasásának protokollja

Tipikus tranzakció: Regisztermutató írása, majd 2-bájtos regiszter kiolvasás (TEMP, TSET, THYST regiszterekre használható)



Az olvasást általában a regiszter cím beállítása (írás) előzi meg. Irányváltás STOP és START vagy RESTART feltétellel lehetséges.

Emlékeztető: I/O térkép

Energia

LaunchPad with MSP430G2553

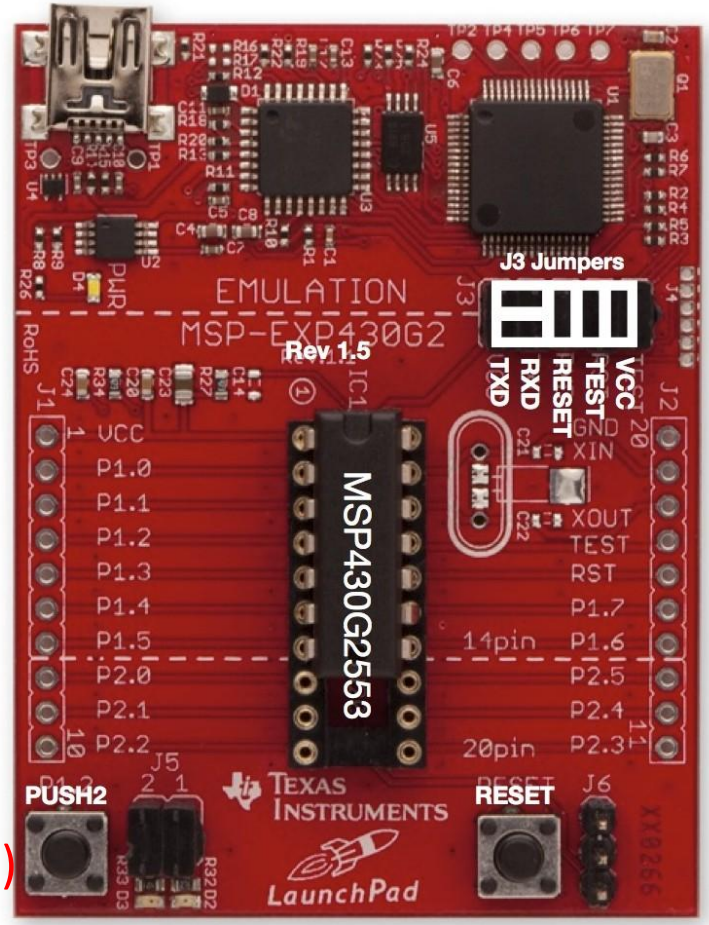
Revision 1.5

Flash 16 KB
Serial Hardware

Hardware
Pin number

I²C
Serial UART
SPI

analogRead()
digitalRead() and digitalWrite()
digitalRead(), digitalWrite() and analogWrite()



+3.3V				1
RED_LED		A0	P1_0	2
	RXD	A1	P1_1	3
	TXD	A2	P1_2	4
PUSH2		A3	P1_3	5
		A4	P1_4	6
	SCK (B0)	A5	P1_5	7
	CS (B0)	SCL	P2_0	8
		SDA	P2_1	9
			P2_2	10

sw I2C

(önkényes választás)

20					GROUND
19	P2_6				XIN
18	P2_7				XOUT
17					TEST
16					RESET
15	P1_7	A7	SDA	MOSI (B0)	
14	P1_6	A6	SCL	MISO (B0)	GREEN_LED
13	P2_5				
12	P2_4				
11	P2_3				

hw I2C

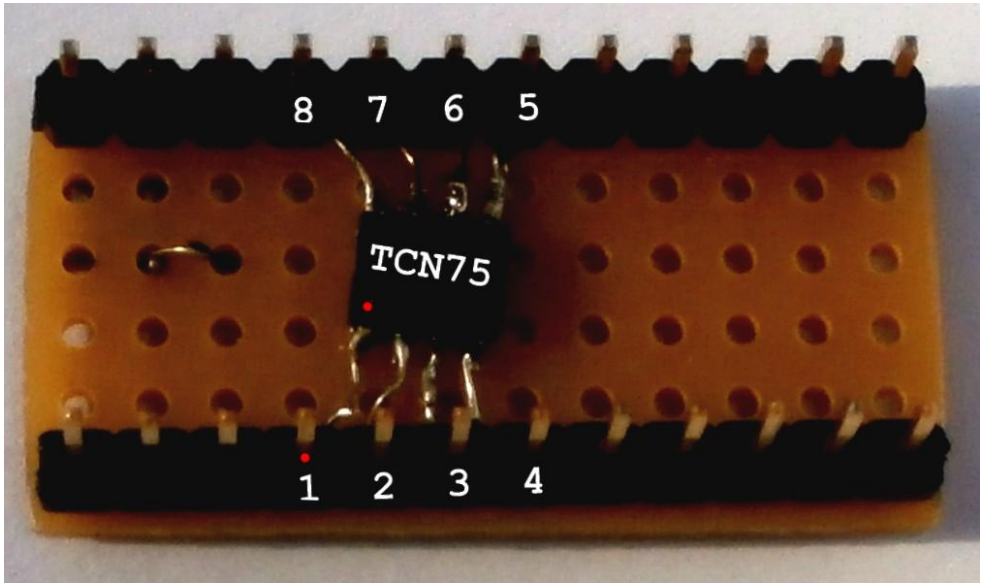
(fix kiosztás)

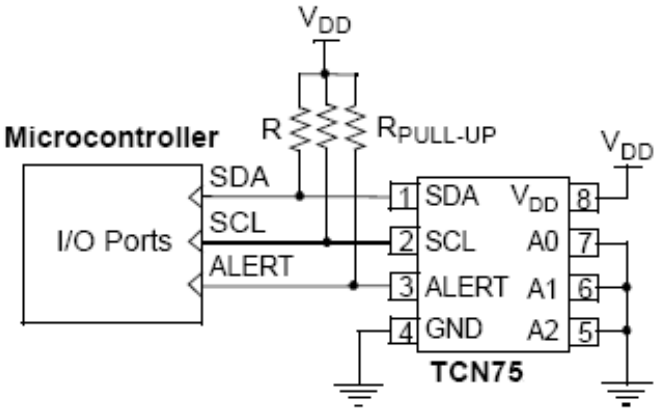
Rei Vilo, 2012
embeddedcomputing.weebly.com

version 1.3 2102-09-09

Kapcsolás, szerelés

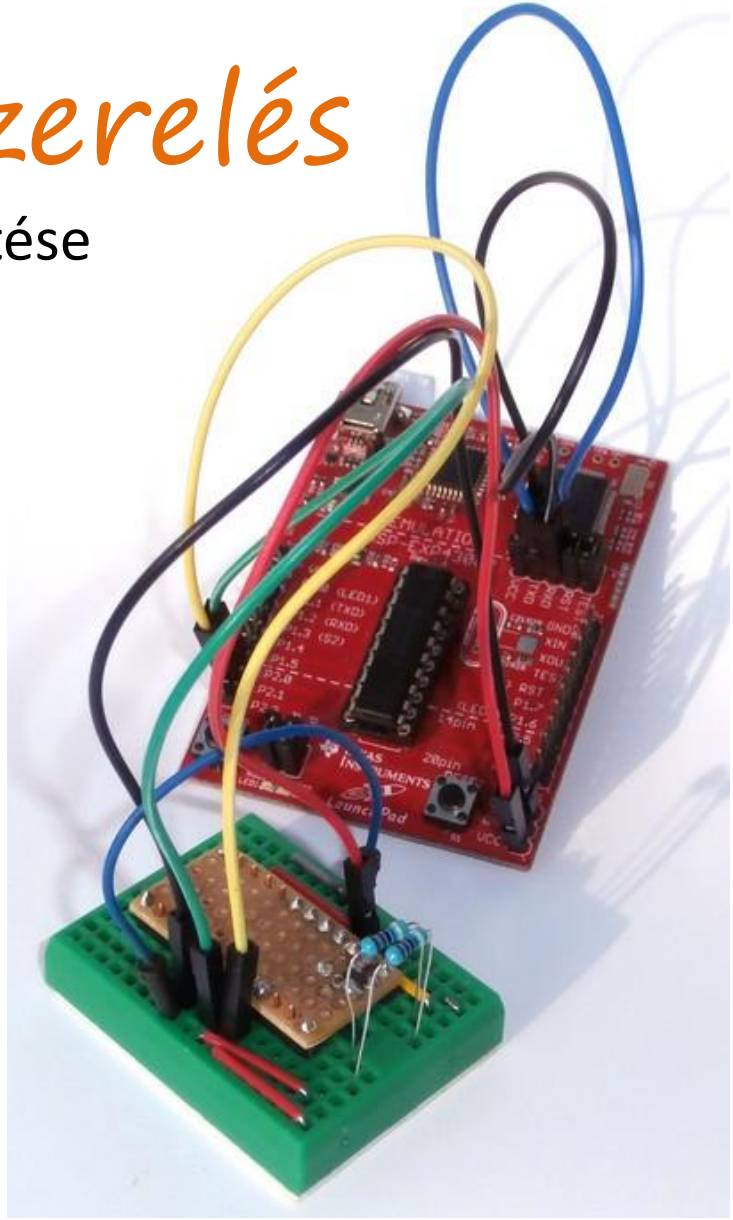
- Felületszerelt (SMD) IC kivezetése





A 3. lábat most nem használjuk, nem kell bekötni!

R = 2 x 4,7 kΩ



A képen a szoftveres I2C bekötése látható

TCN75_hw.ino

Hardveres I2C használathoz a LED2 átkötést (zöld LED) vegyük le a kártyáról!

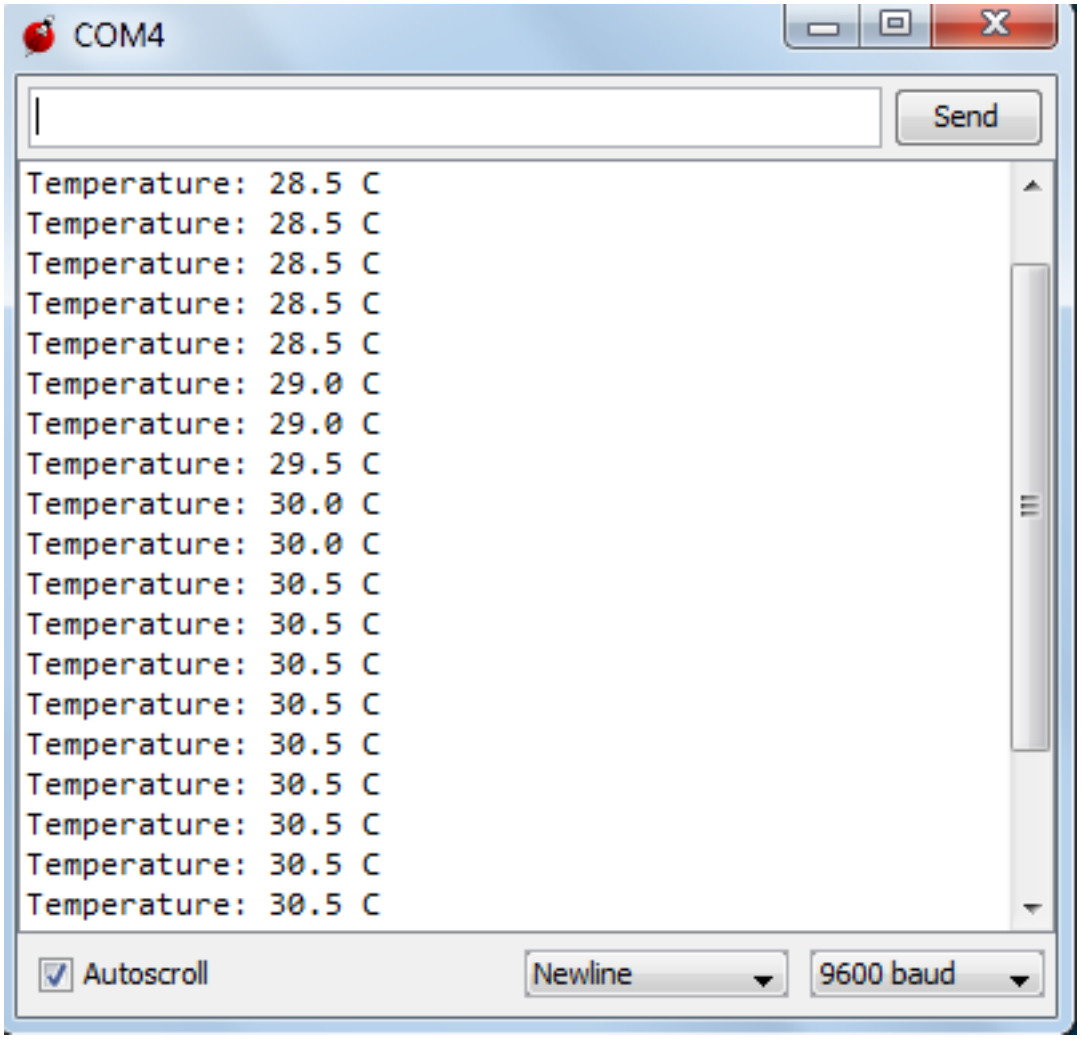
```
#include "Wire.h"                          // HW supported I2C library (requires USCI)
#define tcn75_addr 0x48                    // I2C address when A0,A1,A2=0

int16_t tcn75_read(void) {
  Wire.beginTransmission(tcn75_addr); // Start I2C transaction
  Wire.write(0);                           // Address of the TEMP register
  Wire.endTransmission();                  // Set reg and stop writing
  Wire.requestFrom(tcn75_addr, 2);     // Read 2 bytes from the preset register
  return (Wire.read() << 8 | Wire.read());
}

void setup() {
  Serial.begin(9600);                      //Soros kapcsolat 9600 bit/s
  Wire.begin();                            //I2C kapcsolat inicializálása
}

void loop() {
  delay(2000);
  float temp = tcn75_read();               // Read temperature in 1/256 C degrees
  Serial.print("Temperature: ");
  Serial.print(temp/256,1);                // Print result with 1 decimal
  Serial.println(" C");
}
```

A programfutás eredménye



A hőmérő kézzel történő melegítésekor látható, hogy a mért értékek fél fokonként növekednek
(ez a felbontás).

I2C szoftveres támogatással

Jellemzők:

- Az SCL órajel időzítése és az adatok bitenkénti kiküldése vagy beolvasása szoftveres időzítéssel történik.
- A szoftveres I2C kommunikáció nem igényel speciális hardver eszözöket (Timer, USI vagy USCI)
- Bármelyik általános célú I/O kivezetés felhasználható

Felhasznált programkönyvtár:
SoftwareWire (Master I²C Software Library for MSP430G2553)

Linkek:

http://embeddedcomputing.weebly.com/master-isup2c-software-library.html
http://forum.43oh.com/topic/3617-energia-library-software-i2c-master-for-msp430g2553/

TCN75_sw.ino

```
* #include "SoftwareWire.h"                  // SW supported I2C library
  #define tcn75_addr 0x48                   // I2C address when A0,A1,A2=0
* #define SCL_PIN P2_0                      // I2C bus clock line
* #define SDA_PIN P2_1                      // I2C bus data line
* SoftwareWire Wire(SDA_PIN, SCL_PIN);      // Instantiate SW I2C channel
  int16_t tcn75_read(void) {
    Wire.beginTransmission(tcn75_addr); // Start I2C transaction
    Wire.write(0);                      // Address of the TEMP register
    Wire.endTransmission();             // Set reg and stop writing
    Wire.requestFrom(tcn75_addr, 2);    // Read 2 bytes from the preset register
    return (Wire.read() << 8 | Wire.read());
  }
  void setup() {
    Serial.begin(9600);                 //Soros kapcsolat 9600 bit/s
    Wire.begin();                       //I2C kapcsolat inicializálása
  }
  void loop() {
    delay(2000);
    float temp = tcn75_read();          // Read temperature in 1/256 C degrees
    Serial.print("Temperature: ");
    Serial.print(temp/256,1);           // Print result with 1 decimal
    Serial.println(" C");
  }
```

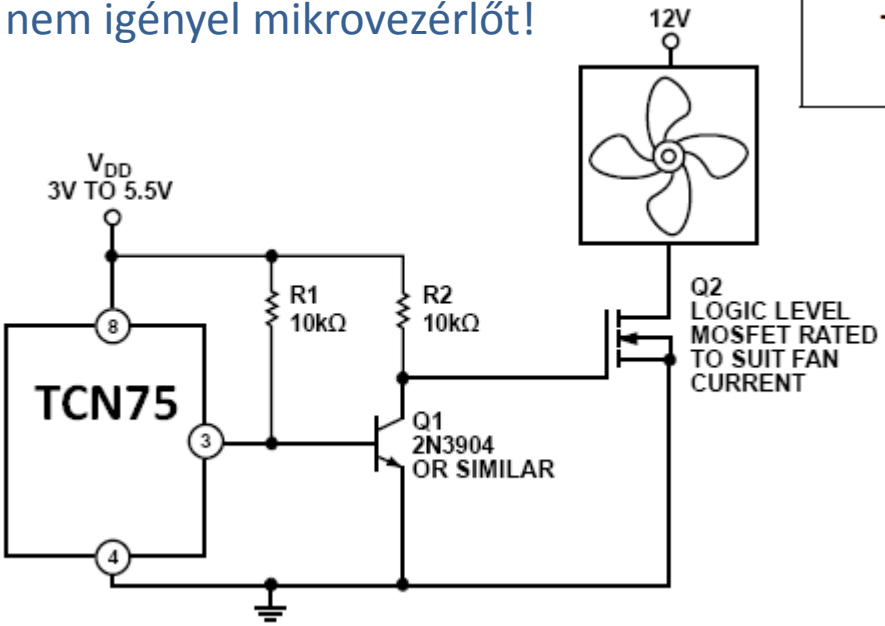
Csak a megjelölt sorokban van változás!

További lehetőségek

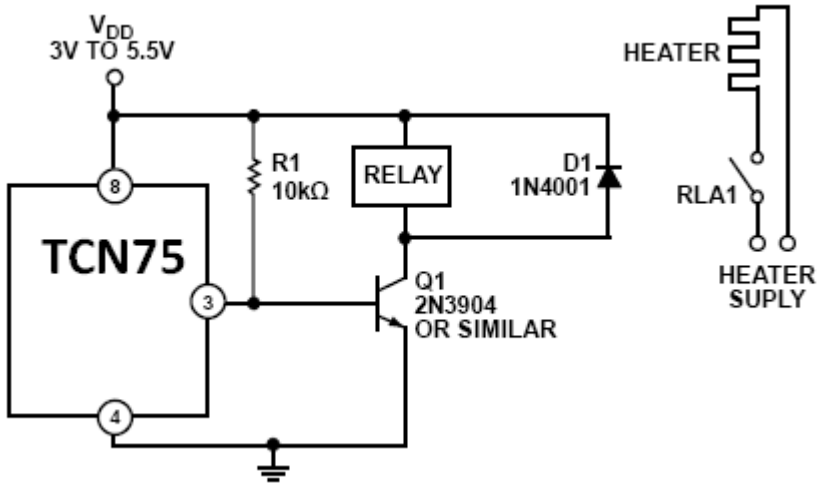
A termosztát funkció használatához be kell állítani a **TSET** és **THYST** regisztereket, és a **CONFIG** regiszter írásával be kell állítani a kívánt komparálási módot és polaritást.

Megjegyzés: A regiszterek beállítása után a termosztát nem igényel mikrovezérlőt!

D [7]	D [6]	D [5]	D [4]	D [3]	D [2]	D [1]	D [0]
Must Be Set To Zero			Fault Queue		INT/ CMPTR, Polarity	COM P/INT	Shut-down

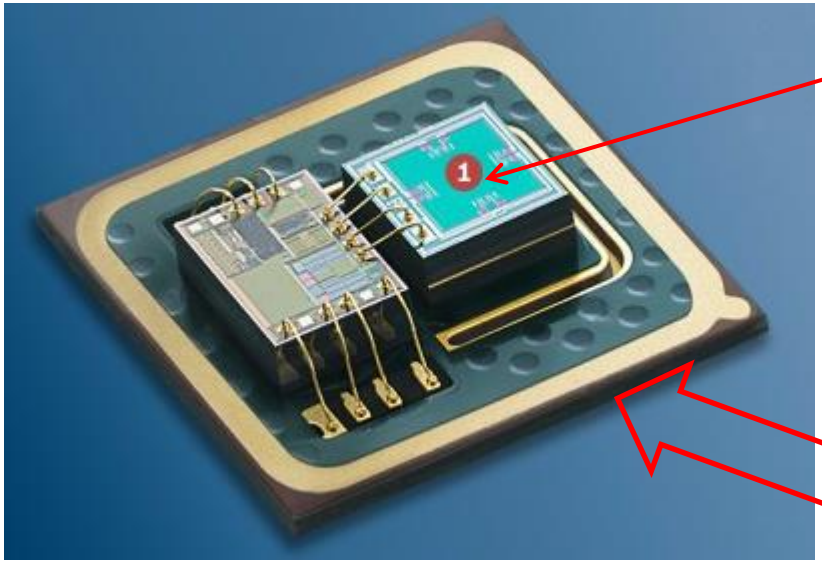


Hűtés ventillátorral



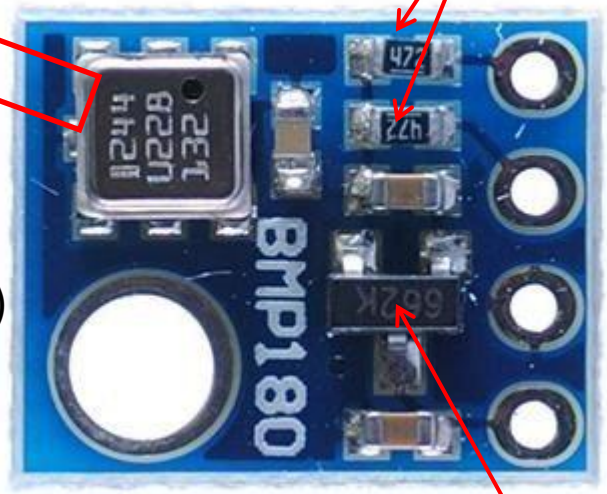
Fűtés elektromos fűtőtesttel

Légnyomás mérése BMP180 szenzorral



Piezorezisztív nyúlásmérő, amely a nyomás hatására bekövetkező deformációt érzékeli

Felhúzó ellenállások



Feszültségstabilizátor (3,3V)

Bosch SensorTec BMP180

Nyomásmérés: 300 – 1100 hPa (9000 - -300m)
Tápfeszültség: 1,8 – 3,6 V
Áramfelvétel: 5 µA (1 mintavétel/s esetén)
Kis zaj: 0.06hPa (0.5m) kisfogyasztású mód
0.02hPa (0.17m) nagyfelbontású mód
Jellemzők: Hőmérő, I2C felület, gyárilag kalibrált

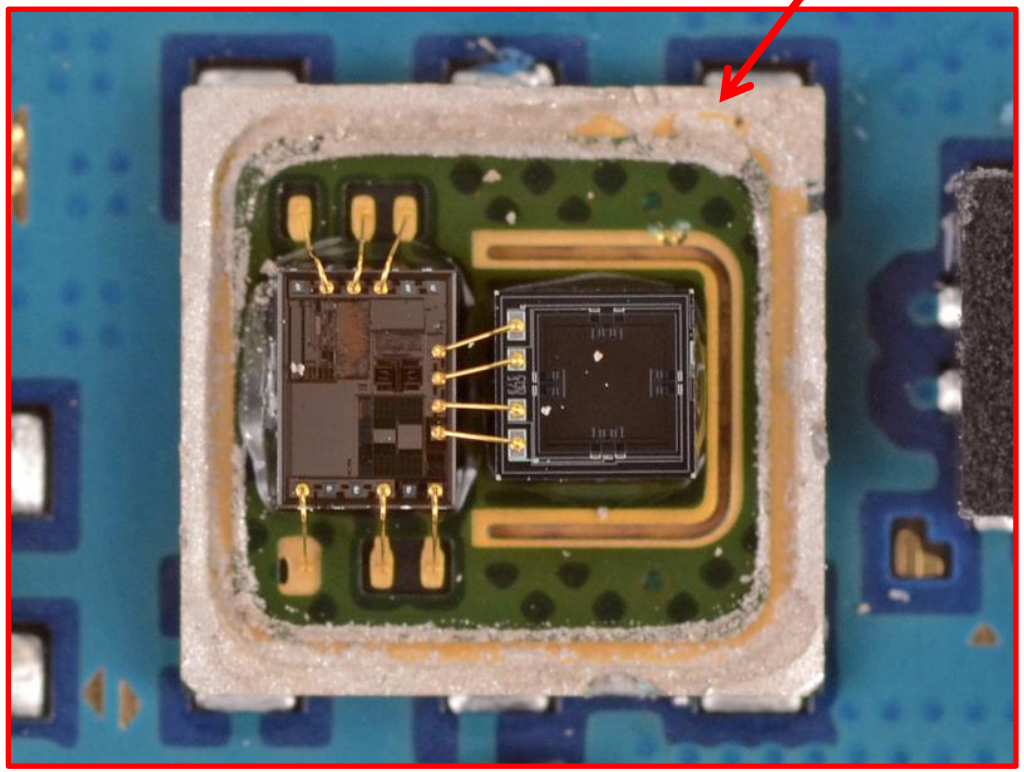
Alkalmazási területek

- GPS navigáció kiegészítése (magasságmérés)
- Kültéri és beltéri navigáció
- Sport és szabadidős tevékenység (pl. magasságmérés túrázás közben)
- Időjárás előrejelzés (a légnyomás helyi süllyedése vihar közeledtét jelzi)
- Függőleges sebesség ellenőrzése (emelkedési/süllyedési sebesség)

Példa: http://www.ifixit.com/Teardown/Samsung+Galaxy+Nexus+Teardown/7182

A szétszedett Samsung Galaxy Nexus belsejében is található egy Bosch BMP180 légnyomásmérő!

BMP180 szenzor



Kapcsolás, szerelés

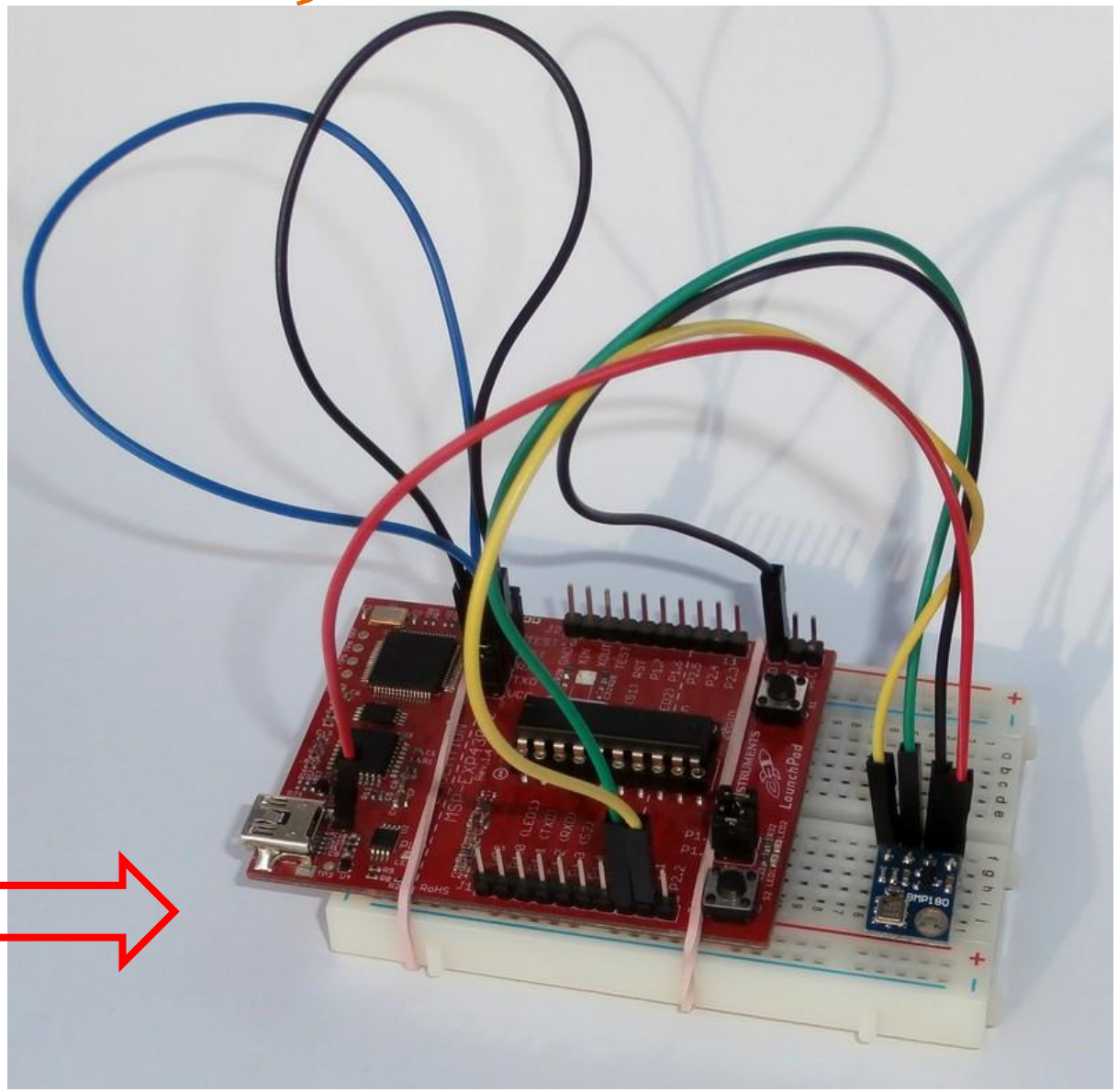
Hardveres I2C

Launchpad	**BMP085 module**
TP1 (+5V)	VIN
GND	GND
P1_6 (pin14)	SCL
P1_7 (pin15)	SDA

A LED2 átkötést vegyük le!

Szoftveres I2C

Launchpad	**BMP085 module**
TP1 (+5V)	VIN
GND	GND
P2_0 (pin8)	SCL
P2_1 (pin9)	SDA



PressureSensor_hw.ino

Felhasználjuk Adrian Struder BMP085 templát könyvtárát is
https://github.com/astuder/BMP085-template-library-Energia

```
#include "Wire.h"                       // HW supported I2C library (requires USCI)
#include "BMP085_t.h"                   // BMP180 template library
BMP085<3> PSensor;                      // instantiate and define precision (0..3)

void setup() {
  Serial.begin(9600);                    //Soros kapcsolat 9600 bit/s
  Wire.begin();                          //I2C kapcsolat inicializálása
  PSensor.begin();                       //initalize pressure sensor
}

void loop() {
  PSensor.refresh();                     // read current sensor data
  PSensor.calculate();                   // calculate temperature and pressure
  float t = PSensor.temperature;         // temperature in 0.1 Celsius units
  float p = PSensor.pressure;            // pressure in Pascal units
  Serial.print("Temperature: ");
  Serial.print(t/10,1);                  // Print result with 1 decimal
  Serial.println(" C");
  Serial.print("Pressure: ");
  Serial.print((p+1440)/100,1);          // pressure in hPa with altitude correction
  Serial.println(" hPa");
  delay(5000);
}
```

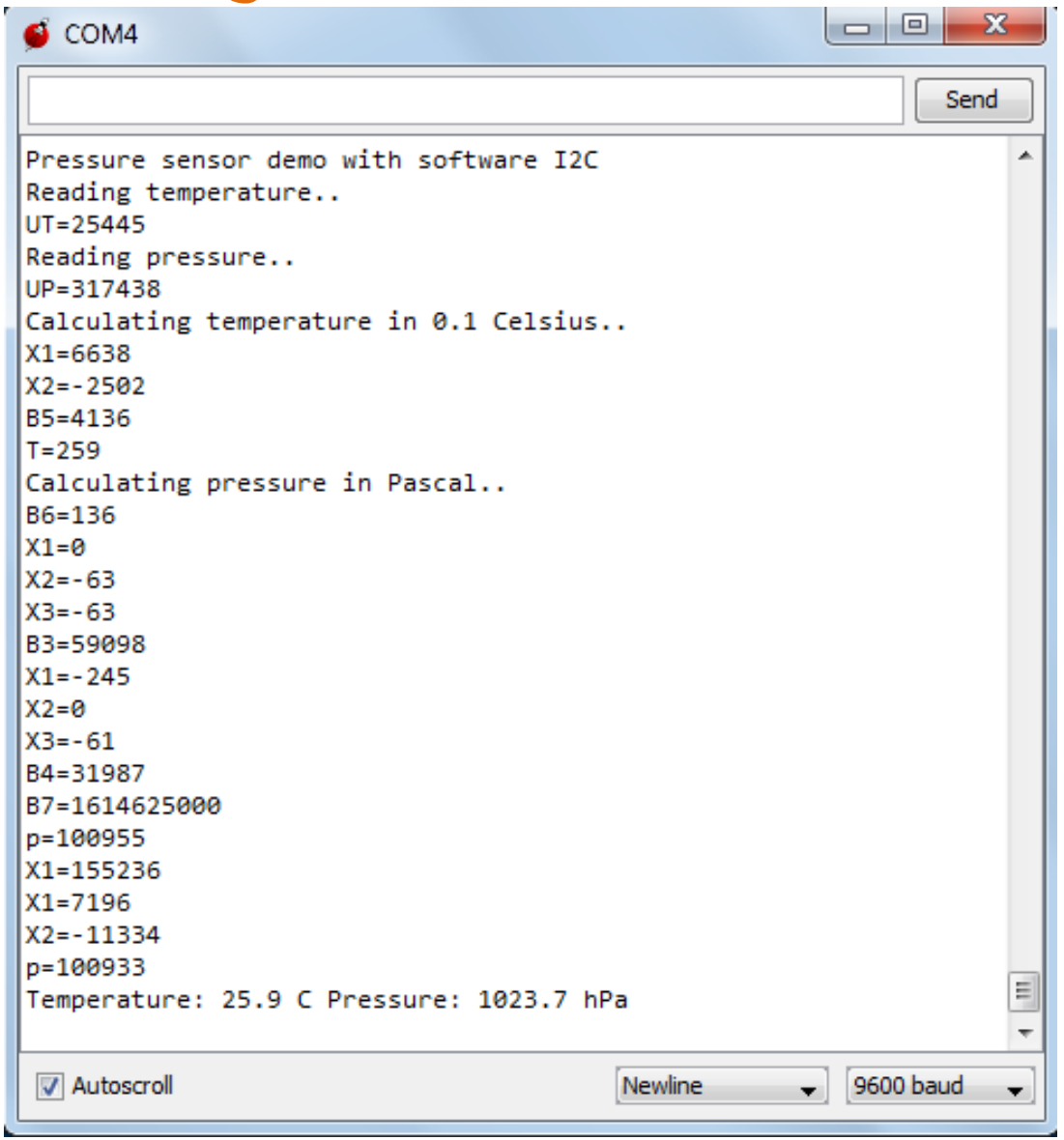
Ellenőrzés, nyomkövetés

A **BMP085_t.h** állományban definiáljuk a DEBUG_BMP085 makrót, ekkor aktiválódnak a nyomkövető kiíratások, amelyekkel ellenőrizhetjük a szenzorból kiolvasott nyers adatokat, valamint a hőmérséklet és a nyomás kiszámítását.

Tipp: Töröljük ki a komment jelet!

```
// #define DEBUG_BMP085
```

A kiszámítás képleteit a BMP180 szenzor adatlapja tartalmazza.
http://ae-bst.resource.bosch.com/media/products/dokumente/bmp180/BST-BMP180-DS000-09.pdf



A kiszámítás menete

Table 5: Calibration coefficients

	BMP180 reg adr	
Parameter	**MSB**	**LSB**
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

UT és **UP** a nyers hőmérséklet és nyomás adat
oss – oversampling (0 – 3)

calculate true temperature

$X1 = (UT - AC6) * AC5 / 2^{15}$
$X2 = MC * 2^{11} / (X1 + MD)$
$B5 = X1 + X2$
$T = (B5 + 8) / 2^4$ [0,1 °C]

calculate true pressure

$B6 = B5 - 4000$
$X1 = (B2 * (B6 * B6 / 2^{12})) / 2^{11}$
$X2 = AC2 * B6 / 2^{11}$
$X3 = X1 + X2$
$B3 = ((AC1*4+X3) << oss + 2) / 4$
$X1 = AC3 * B6 / 2^{13}$
$X2 = (B1 * (B6 * B6 / 2^{12})) / 2^{16}$
$X3 = ((X1 + X2) + 2) / 2^2$
$B4 = AC4 * (unsigend\ long)(X3 + 32768) / 2^{15}$
$B7 = ((unsigned\ long)UP - B3) * (50000 >> oss)$
if (B7 < 0x80000000) { p = (B7 * 2) / B4 }
else { p = (B7 / B4) * 2 }
$X1 = (p / 2^8) * (p / 2^8)$
$X1 = (X1 * 3038) / 2^{16}$
$X2 = (-7357 * p) / 2^{16}$
$p = p + (X1 + X2 + 3791) / 2^4$ [Pa]

PressureSensor_sw.ino

```
#include "SoftwareWire.h"  // required by BMP085 library
#define SCL_PIN P2_0       //I2C bus clock line
#define SDA_PIN P2_1       //I2C bus data line
SoftwareWire Wire(SDA_PIN, SCL_PIN); //Instantiate SW I2C channel
#include "BMP085_t.h"      // import BMP085 template library
BMP085<3> PSensor;         // instantiate sensor, 0..3 = precision of measurement
void setup() {
  Serial.begin(9600);                  //Soros kapcsolat 9600 bit/s
  Wire.begin();                        //I2C kapcsolat inicializálása
  PSensor.begin();                     //initalize pressure sensor
}
void loop() {
  PSensor.refresh();                   // read current sensor data
  PSensor.calculate();                 // calculate temperature and pressure
  float t = PSensor.temperature;       // temperature in 0.1 Celsius units
  float p = PSensor.pressure;          // pressure in Pascal units
  Serial.print("Temperature: ");
  Serial.print(t/10,1);                // Print result with 1 decimal
  Serial.println(" C");
  Serial.print("Pressure: ");
  Serial.print((p+1440)/100,1);        // pressure in hPa with altitude correction
  Serial.println(" hPa");
  delay(5000);
}
```

Helyi légnyomás átszámítása tengerszintre

Tudnunk kell a helyi tengerszint feletti magasságot (**altitude**) és a szenzorral meg kell mérnünk az abszolút helyi nyomás értékét (**p**).

A tengerszintre átszámított légnyomás ($\mathbf{p_0}$) a következő képlettel számítható ki:

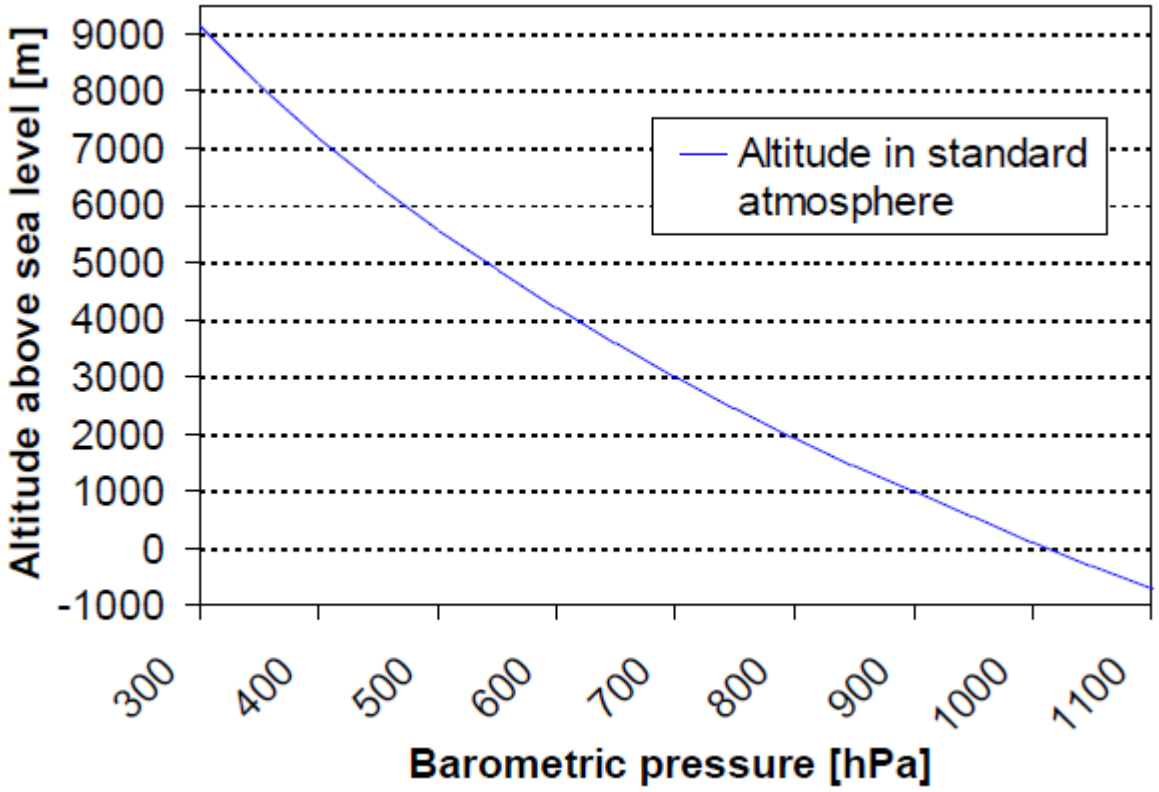
$$p_0 = \frac{p}{\left(1 - \frac{altitude}{44330}\right)^{5.255}}$$

Egyszerű közelítés:
Debrecenben (kb. 120 m) 1440 Pa-t hozzáadunk a mért légnyomáshoz

Magasság meghatározása

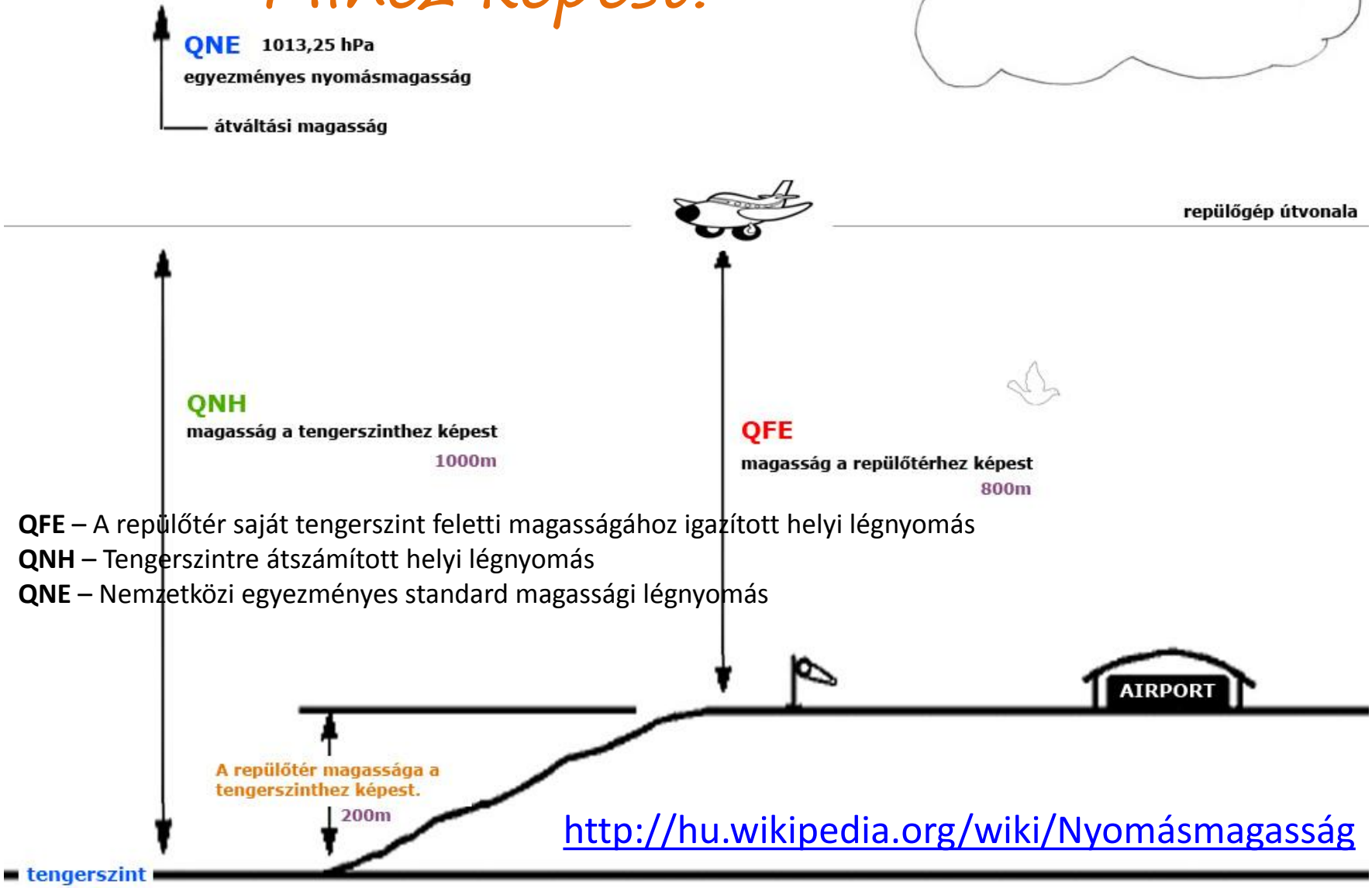
$$\text{magasság} = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right)$$

Ahol **p** az általunk mért nyomás, $\mathbf{p_0}$ pedig a tengerszinti nyomás (pl. 1013.25 hPa)



A gyakorlatban a nyomás nemcsak a magasságtól, hanem a meteorológiai viszonyoktól is függ (hőmérséklet, páratartalom, stb.)

– Milyen magasan repül a repülő?
– Mihez képest?

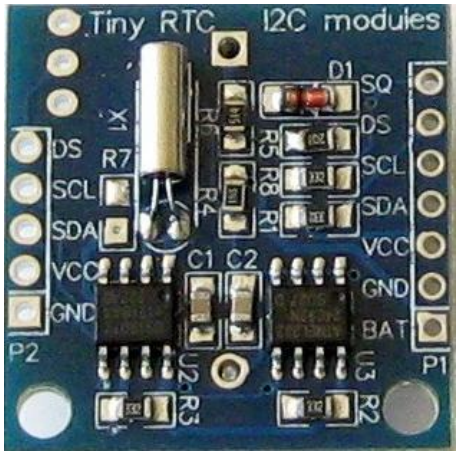


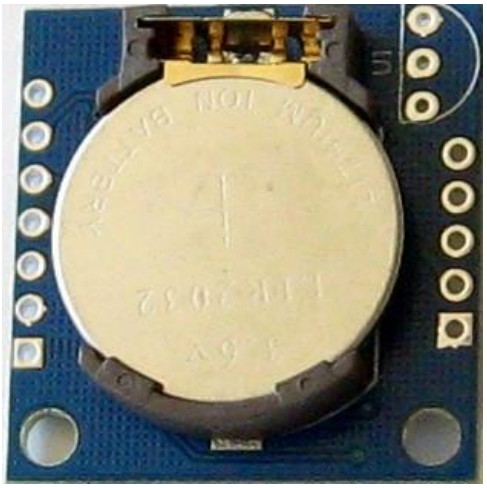
QFE – A repülőtér saját tengerszint feletti magasságához igazított helyi légnyomás
QNH – Tengerszintre átszámított helyi légnyomás
QNE – Nemzetközi egyezményes standard magassági légnyomás

http://hu.wikipedia.org/wiki/Nyomásmagasság

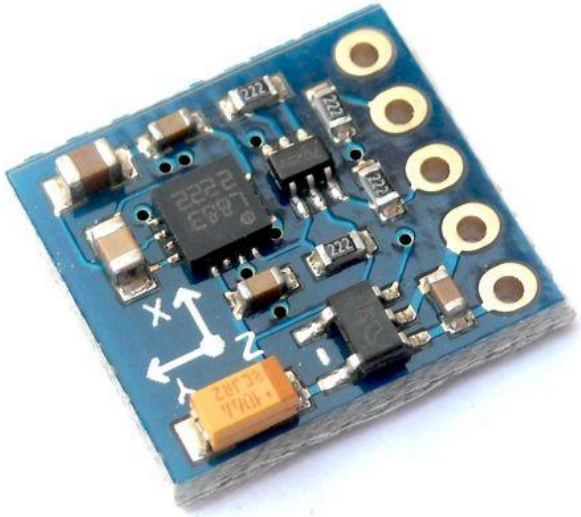
További I2C eszközök

24FC515 64K x 8 bit EEPROM

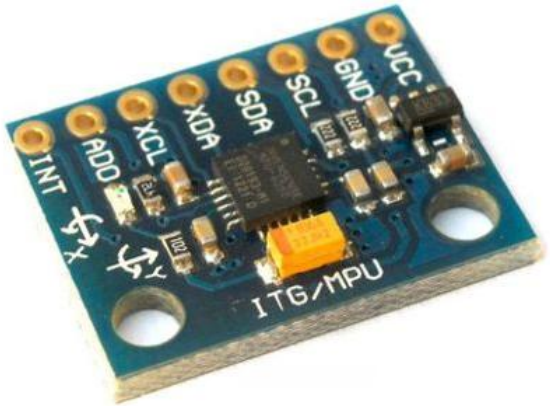




DS1307 Real-time óra elemes háttértáplálással



HMC5883 digitális iránytű/magnetométer



MPU6050 gyorsulásmérő és giroszkóp