Debreceni Egyetem
Informatikai Kar

Beágyazott mérő- és szabályzórendszerek

Témavezető:
dr. Szabó István
egyetemi docens

Készítette:
Juhász Tibor
programtervező
matematikus

Debrecen
2007

Tartalomjegyzék

Bevezetés

A mikroprocesszor mint a számítógép központi egysége a hozzákapcsolt memóriával és be-kimeneti egységekkel együtt használható a gyakorlatban. Azokat az áramköröket, amelyek egy tokba integrálva tartalmazzák az elõbb felsorolt elemeket, mikrovezérlőknek nevezzük.[6]

Meglepően sok modern berendezés rejt magában mikrovezérlőt, pl. bármely készülék, amelyhez távirányító is jár. Ha mikrohullámú sütőnkben LED-es vagy LCD-kijelző és billentyűpad található, biztosra vehetjük, hogy mikrovezérlőt is tartalmaz. A mai autók mindegyike tartalmaz legalább egy mikrovezérlőt, de akár 6-ot, 7-et is (motor, ABS, sebességszabályozó, stb). Digitális fényképezőgépek, mobiltelefonok, lézernyomtatók, mosogatógépek, …alapvetően minden termék, amely párbeszédet folytat felhasználójával.

A mikrovezérlők a PC-khez képest erősen korlátozott erőforrásokkal rendelkeznek: kB-okban mérhető memória, néhány MHz-es órajel, nincs operációsrendszer, nincs merevlemezes meghajtó, a perifériákat a programozónak kell szoftveresen illesztenie.

Ezen dolgozat célja az ATMEL AVR Atmega128-as mikrovezérlőjével történő mérés, vezérlés és szabályozás bemutatása konkrét feladatokon, illetve megvalósult rendszereken - egy felfedezőroboton, egy klímakamrán és egy mérőállomáson - keresztül.
A feladatok megoldása során a mikrovezérlő időzítőinek/számlálóinak üzemmódjait, a különböző kommunikációs protokollok (soros, I^2C, SPI) alkalmazási lehetőségeit, illetve az analóg-digitális átalakító használatát tanulmányoztam.

A fejlesztéshez kezdetben a BASCOM-AVR[1] nevű IDE-t használtam. Ez amellett, hogy egyszerűsége miatt kiválóan alkalmas az első programok megírására, súgója révén rávilágít a mikrovezérlő lehetőségeire, és felhívja a figyelmet az egyes lehetőségekkel kapcsolatos legfontosabb fogalmakra és paraméterekre. Nyelvi szinten támogatja számos alapvető periféria kezelését (soros csatlakozás, LCD-kijelző, időzítő/számlálók, ADC, 1WIRE, TWI, WATCHDOG, stb.) nem csak egyetlen, hanem számos AVR processzor használata mellett.

Komoly hátránya viszont, hogy nem engedi kihasználni a processzor összes lehetőségét, és a forráskódot nehéz karbantartani, nagyon ügyetlen a nyelv maga, így az alapfogalmak

[1] A tanszék vásárolt egy példányt

megismerése után a WinAVR nevű (ingyenes) IDE-t kezdtem el használni, ami már C-programozást igényel. Ez nem tartalmaz programkönyvtárakat az egyes kommunikációs lehetőségek, illetve perifériák kezelésére, viszont az Interneten szép számmal találunk többé-kevésbé megfelelő megoldásokat, ilyen pl. az AVRLib.

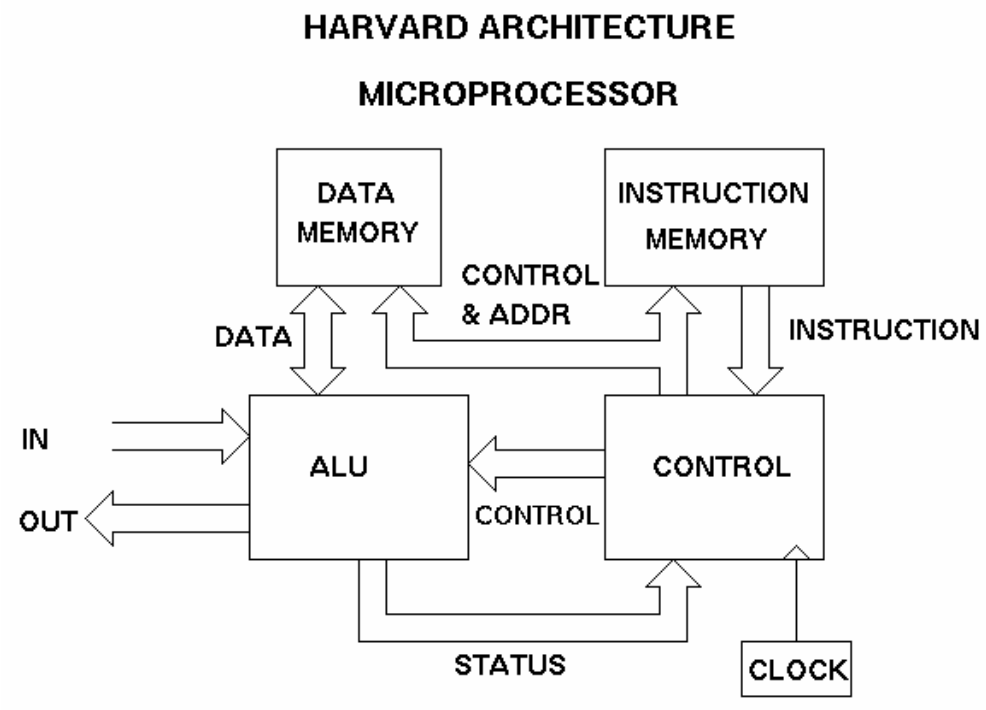
A mikrovezérlő felépítését és működését az ATMEL honlapján elérhető közel 400 oldalas angol nyelvű adatlapból igyekeztem megérteni, amely mind a programozó, mind a villamosmérnök számára kimondottan nehéz olvasmány.

A mikrovezérlős fejlesztés olyan határterület, amely egyaránt igényel programozói és elektronikai ismereteket. Az ember nem lehet biztos a szoftver helyességében, amíg a hardver nem tökéletes, és fordítva; így ha a programozó nem villamosmérnök is egyben, nem tud egyedül dolgozni. Harasztosi Lajos mérnöktanártól kaptam segítséget.

Szoftverfejlesztés mikrovezérlős környezetben

A processzor

A programokat az Atmel AVR processzorcsaládjához tartozó egyik processzorára, az ATmega128-ra írtam. Ez egy Harvard-architektúrájú (a programot és az adatokat nem ugyanabban a memóriában tároló) RISC 128 kB flash memóriával. Ez a memória elegendő volt programjaim tárgykódjának befogadására. Az adatok tárolására szolgáló bájtszervezésű memória kényelmesebb fejlesztést biztosít, mint pl. a PIC-ek szószervezésű memóriája. Ugyanakkor a program memóriája szószervezésű, 16 bites szavakból áll. Az utasítások 16 és 32 bitesek. A flash memória előnye, hogy a futó program nem tudja írni, így a feszültségingadozások, és egyéb zavaró tényezők nem módosíthatják a futó kódot. Az utasítások többsége egy vagy kettő órajelet igényel, éppen ezért az AVR-ek gyorsnak számítanak a 8-bites mikrovezérlők között. Az AVR processzorcsalád processzorait úgy tervezték, hogy a lefordított C-kód hatékonyan fusson rajtuk.[4]



1. ábra: Harvard-architektúrájú processzor

A processzor perifériái

Időzítők/számlálók

Az időzítők/számlálók perifériák, amelyek a CPU-tól függetlenül működnek, és csak olyankor „zaklatják” a CPU-t, amikorra előírtuk nekik. A „zaklatás” formája lehet egy jelzőbit beállítása, amelyet a CPU folyamatosan le tud kérdezni, vagy megszakítás kiváltása, amellyel megszakíthatja a normál működést.
Az időzítőt és a számlálót azért szoktuk együtt említeni, mert az időzítő/számláló periféria impulzusok számlálásával tartja az ütemet. Ha az impulzusok egy szinkron periodikus forrásból származnak, akkor segítségükkel pontosan mérhetjük az eltelt időt, ha pedig egy aszinkron nemperiodikus forrásból jönnek, a bejövő impulzusokat számlálhatjuk pontosan. Az első esetben számlálhatnánk egy 32,768 kHz-es kvarckristály impulzusait és így pontosan mérhetnénk az időt. A második esetben számlálhatnánk egy fénymegszakító áramkör impulzusait, és pontosan mérni tudnánk, hogy hány ember lép be az ajtón és szakítja meg a fénynyalábot.
Az ATmega128-asnak négy időzítője/számlálója van, két 8-bites, és két 16-bites. Amikor az időzítő elszámol a maximum értékig (a 8-bitesnél ez 255, a 16-bitesnél 65535), túlcsordul, majd visszaáll nullára. Adott időzítő esetén annak OCR (Output Compare Register) regiszterébe írva egy értéket elérhetjük, hogy az időzítő kisebb értékeknél csorduljon túl; az időzítő összehasonlítja az értéket a számlálója értékével, és amikor a kettő egyezik, beállít egy jelzőbitet, vagy megszakítást vált ki. Azt is beállíthatjuk, hogy 0-ra csorduljon túl.

Az időzítők felprogramozhatók külső események figyelésére (input capture event), amikor is egy láb állapotának megváltozására az időzítő elmenti számlálójának az esemény bekövetkeztekor aktuális értékét. Ez a fajta számlálás alkalmas külső jelek impulzusszélességének mérésére. Ha a külső impulzusok periodikusak, egy frekvenciaszámlálót kapunk.

Az időzítő/számláló a programvégrehajtástól függetlenül fut, és a program háromféleképpen tudja monitorozni és lereagálni az időzítő/számláló eseményeit.

1. Folyamatosan lekérdezzük a túlcsordulást jelző biteket
2. Megszakítással szakítjuk meg a program futását

3. Engedjük, hogy az időzítő automatikusan megváltoztassa a kimenetként működő lábak feszültségszintjeit

Az időzítők/számlálók órája egy multiplexerhez[2] csatlakoztatott előosztót használ. Az előosztót arra használjuk, hogy a bemenő órajelet (annak frekvenciáját) leosszuk, és a multiplexer választja ki, hogy mely leosztott jeleket használjuk bemenő órajelként. Az előosztó órajelének forrása lehet egy külső óra pl. egy 32,768 kHz-es kristály, vagy használhatja a rendszerórát is.[7]

ADC

Az **analóg-digitális átalakító** (AD átalakító) a digitális jelátvitel előnyei miatt alkalmazott áramkör. Az analóg jelből az áramkör meghatározott rövid időközönként mintát vesz, és annak nagyságát bináris számokkal fejezi ki, azaz digitalizálja. Minél rövidebb a mintavételi időtartam, annál pontosabb az átalakítás. A felbontás megmondja, hogy az AD átalakító az analóg bemenő jelet hány egyedi értékké tudja átalakítani. Mértékegysége minden esetben a bit. Az AD átalakító, amely az analóg jelet 256 elemi értékre tudja felbontani, 8 bites felbontással rendelkezik ($2^8 = 256$). [12]

A mi ADC-nk 10 bites felbontással rendelkezik, és 8 csatornán teszi lehetővé a jelek digitalizálását. Az ADSC (ADC Start Conversion) bit beállításával egyetlen érték átalakítását érjük el. Az átalakítás ideje alatt a bit beállítva marad, az átalakítás után pedig azonnal törlődik. Ha a konverzió közben váltunk ADC-csatornát, a konverzió olyankor is szabályosan végbemegy. Az ADC másik üzemmódjában (Free Running) a hardver folyamatosan mintavételez és frissíti az ADC adat regiszterét. Ezt a folyamatos konverziót biztosító üzemmódot az ADCSRA regiszter ADFR bitjének beállításával állítjuk be. Az üzemmód beállítása még nem jelenti a konverzió kezdetét. Az első átalakítást az ADCSRA regiszter ADSC bitjének beállításával indítjuk el. A maximális felbontás eléréséhez az átalakítónak egy 50 és 200 kHz közötti frekvenciájú bemenő órajelre van szüksége. Ha 10 bitnél kisebb felbontással is megelégszünk, a bemenő frekvencia 200 kHz-nél nagyobb is lehet, így sűrűbb lesz a mintavételezés is.

[2] A **multiplexer** (vagy **mux** vagy ritkán **muldex**) egy speciális kódoló ami két vagy több bejövő jelet egy kimenő jellé egyesít úgy, hogy azok később egyértelműen szétválaszthatók (Wikipédia)

Az ADC modul tartalmaz egy előosztót, amely a megfelelő bemenő órajel előállításáért felel. Ezt bármilyen 100 kHz fölötti CPU-frekvenciából elő tudja állítani. Az előosztás mértékét az ADCSRA regiszter ADPS bitjeivel állítjuk be. Az előosztó az ADC bekapcsolásakor – az ADEN bit beállításakor - kezd el számlálni. Amíg az ADEN be van állítva, folyamatosan számlál, amíg nincs beállítva, folyamatosan újraindul.

Egyszeri konverzió indításakor az átalakítás az ADC órajelének első felfutó élére következik be. Egyetlen átalakítás 13 ADC órajelet vesz igénybe, leszámítva a legelsőt, amely az analóg elektronika inicializálása miatt 25 órajelig tart.

Egy átalakítás elvégzése után az eredmény beíródik az ADC adat regiszterébe, az ADIF bit pedig beállítódik (azaz 1 lesz). Ha egyszeri átalakítást hajtottunk végre, az ADSC törlődik. Folyamatos átalakításkor azonban egy átalakítás befejezése után a következő azonnal elkezdődik, és az ADSC nem törlődik, beállítva marad.[8]

UART

Az RS-232 egy aszinkron soros átvitelt leíró szabvány. A megvalósító chip-et pedig UART-nak nevezzük: Universal Asynchron Receiver Transmitter. Az UART-ot a mikrovezérlő és a PC közötti kommunikáció megvalósítására használtam. Az ATmega128 két kommunikációs porttal is támogatja a szabványt: USART0 és USART1. Igazság szerint ezek többet tudnak, mint az UART-ok; ahogy nevükben az S betű jelzi, szinkron átvitelre is alkalmasak.

A PC-vel három vonal segítségével tudunk kommunikálni: TXD, RXD, GND. A TXD vonalon adunk (PC → mikrovezérlő), az RXD-n veszünk (mikrovezérlő → PC), és a GND a közös földelés. Figyeljük meg, hogy van némi bizonytalanság az elnevezésekben; a mikrovezérlő RXD kivezetése a PC TXD kivezetése, és fordítva.

A kommunikáció megkezdése előtt az USART-ot inicializálni kell: be kell állítani a jelváltási sebességet, a keretformátumot és engedélyezni kell a Transmitter-t (Adó) vagy a Receiver-t (Vevő) – értelemszerűen aszerint, hogy mi a szándékunk. Megszakításvezérelt működéshez a globális megszakítás jelzőbitet az inicializálás idejére törölni kell.

```
void USART_Init( unsigned int baud )
{
/* Set baud rate */
UBRRH = (unsigned char)(baud>>8);
UBRRL = (unsigned char)baud;
/* Enable receiver and transmitter */
UCSRB = (1<<RXEN)|(1<<TXEN);
/* Set frame format: 8data, 2stop bit */
UCSRC = (1<<USBS)|(3<<UCSZ0);
}
```

Adat küldéséhez engedélyeznünk kell az USART Transmitter-t: állítsuk be az UCSRB regiszter Transmit Enable (TXEN) bitjét. Amíg a Transmitter engedélyezve van, a TxD láb normál működése szünetel, ehelyett soros kimenetként működik. Hasonló a helyzet az XCK lábbal is szinkron átvitel esetén. A küldés kezdetén a küldendő adat betöltődik egy pufferbe. Ezt a CPU végzi. A puffer tartalmát a Shift Register továbbítja, keretenként.

```
void USART_Transmit( unsigned char data )
{
/* Wait for empty transmit buffer */
while ( !( UCSRA & (1<<UDRE)) );
/* Put data into buffer, sends the data */
UDR = data;
}
```

Adat fogadásakor a Receivert engedélyezzük az RXEN bit beállításával. A Receiver egy érvényes „start bit” érkezte után kezdi meg a vételt. Minden „start bit”-et követő bitet beshiftel a Receiver Shift Register-be mindaddig, míg a keret első „stop bit”-je meg nem érkezik. Ha újabb „stop bit” jön, azzal nem foglalkozik. Ezt követően a Shift Register tartalma átkerül egy pufferbe, amelyet a CPU olvasni tud.

```
unsigned char USART_Receive( void )
{
/* Wait for data to be received */
while ( !(UCSRA & (1<<RXC)) );
/* Get and return received data from buffer */
return UDR;
}
```
[8]

TWI

A TWI (Two-wire Serial Interface) protokoll 128 különböző eszköz csatlakoztatását teszi lehetővé pusztán kettő kétirányú buszvezeték segítségével; az egyik (SCL) az órajel, a másik (SDA) az adatok továbbítására szolgál. Minden egyes felcsatlakoztatott eszköznek egyedi címe van.

A TWI terminológiája:

- Master: az az eszköz, amely előkészíti és végrehajtja az adat küldését; ez állítja elő az órajelet is
- Slave: az az eszköz, amelyet a Master címez
- Transmitter: az az eszköz, amely adatot helyez a buszra
- Receiver: az az eszköz, amely adatot olvas a buszról

Minden adatbitet egy órajelimpulzus kíséretében küldünk. Az adatátvitel azzal kezdődik, hogy a Master egy ún. START feltételt küld, és egy STOP feltétellel végződik. A START és a STOP feltétel között a buszt foglaltnak tekintjük, ilyenkor más Masterek várakoznak. Ha egy START és egy STOP között megjelenik egy újabb START, azt ISMÉTELT START-nak nevezzük. Olyankor alkalmazzuk, amikor a Master új átvitelt akar elindítani anélkül, hogy átengedné a buszt másoknak. A START és STOP feltételeket úgy állítjuk elő, hogy megváltoztatjuk az SDA vonal szintjét, míg az SCL-t a magas szinten tartjuk.[9]

A Philips kifejlesztett egy egyszerű, kétirányú, kétvezetékes buszrendszert hatékony IC-k közti vezérlésre. A buszt Inter IC-nek, vagy csak röviden I²C-busznak nevezik. Napjainkban a Philips gyártmányú IC-k választéka több mint 150 CMOS és bipoláris I²C - busszal kompatíbilis típust tartalmaz. Minden I²C - busszal kompatíbilis eszköz tartalmaz egy on-chip interfészt, ami az I²C buszon lehetővé teszi a többi eszközzel a közvetlen kommunikációt. E tervezési koncepció számos illesztési problémát old meg digitális vezérlésű áramkörök tervezésekor. Az Atmega128-as TWI-je is kompatibilis az I^2C protokollal.

Az I²C-busz néhány jellemzője:

• Csak két buszvezeték szükséges a működéséhez, egy soros adatvonal (SDA) és egy soros órajel (SCL)

• Mindegyik csatlakoztatott eszköz programból címezhető egy egyedi címmel, és a köztük fennálló egyszerű master/slave kapcsolat segítségével a master képes adóként és vevőként is üzemelni

• Valódi több master-es busz ütközésdetektálással és arbitrációval az adatvesztés elhárítására, ha két vagy több master egyidejűleg kezdene küldeni

• Soros, 8-bit-es, kétirányú adatforgalom, melynek sebessége normál üzemmódban 100 kbit/s, gyors üzemmódban 400 kbit/s

• A chipbe épített szűrő az adatvonalon lévő zavarokat szűri ki, megőrizve ezzel az adatintegritást

• Az egy buszra csatlakoztatható IC-k számát csak a busz kapacitása korlátozza, ami maximum 400 pF lehet.[10]

SPI

A Serial Peripheral Interface (SPI) nagysebességű szinkron adatátvitelt tesz lehetővé a processzor és számos periféria között. Gyorsabb adatátvitelt biztosít, mint az I^2C, és a kommunikáció kétirányú. A rendszer két Shift regiszterből és egy Master órajel generátorból áll. Az SPI Master a kívánt Slave SS (Slave Select) lábának alacsony szintbe húzásával kezdi meg a kommunikációt. A Master és a Slave saját Shift regiszterébe helyezi a küldendő adatokat, és a Master órajelére (SCK vonal) megtörténik a kölcsönös adatcsere. Az adatok mindig a MOSI (Master Out – Slave In) vonalon érkeznek a Mastertől a Slave-hez, és a MISO (Master In – Slave Out) vonalon a Slave-től a Master-hez. A szinkronizálás úgy történik, hogy minden egyes adatcsomag után a Master felhúzza a Slave SS vonalát.[8]

Fejlesztői környezet

A fejlesztéshez ingyenes szoftvereket használtam. Programjaimat C nyelven írtam és GCC-vel fordítottam. Egy WinAVR nevű IDE-t használtam, ez egy nyílt forráskódú program, amelyet kimondottan Atmel AVR processzorokra történő fejlesztésre szántak. A WinAVR nem nyújt komolyabb hibakeresési (debug) lehetőségeket, de mint lentebb látni fogjuk, ez nem probléma.

A lefordított programokat AVR Studio-val töltöttem le a processzorra. Amikor PC-re fejlesztünk, ez a viszonylag időigényes, a processzor életidejét csökkentő lépés kimarad.

- Felhasznált programkönyvtárak:
    - AVRLib (soros kommunikáció, parancssor-interpreter)
    - General purpose Embedded Filesystem library: MMC kártya kezelése

    - A valós idejű órát és az I^2C kommunikációt megvalósító függvények az Internetről származnak
- A hibakeresés eszközei:
    - Soros portos kommunikáció + terminál emulátor: a PC-n egy terminál ablakban utasíthatjuk a mikrovezérlőn futó programot, illetve megkapjuk a válaszait
    - LCD-kijelző: a mikrovezérlőre LCD-kijelző illeszthető, melyre a program futás közben írhat
    - Oszcilloszkóp: az oszcilloszkópot a processzor adott kimenetére csatlakoztatva meg tudjuk mérni, hogy megjelennek-e a kívánt impulzusok
    - LED-ek és nyomógombok: a mikrovezérlőre szerelt LED-eket a program kigyújthatja és elolthatja, így akár villogtathatja is; a program bizonyos eseményeit nyomógomb megnyomásához lehet kötni

„Hello, World" mintaprogram

Mikrovezérlős környezetben a klasszikus „Hello, World" program egy LED villogtatását jelenti. Ennek az a magyarázata, hogy nem áll rendelkezésre a PC-s környezetben természetesnek számító kijelző, és ezt akár LCD-kijelzővel, akár soros kommunikáció révén akarjuk helyettesíteni, a kezdők számára komoly feladatra vállalkozunk. Viszont akár egyetlen bit beállításával is kigyújthatunk egy LED-et.

A következő mintakód egy C portra csatlakotatott LED-et villogtat:

```
#define F_CPU 4000000UL /* a proc. órajel-frekvenciája [Hz] */
#include <util/delay.h>
#include <avr/io.h>

int main(void) {
     DDRC = 0xff; /*a C port minden lába kimenetként működjön*/
     for (;;) {
          PORTC |= 1<<2; /* a 2-es lábat bekapcsoljuk */
          _delay_ms(20); /* várunk 20 ms-ot */
          PORTC &= ~(1<<2); /* a 2-es lábat kikapcsoljuk */
          _delay_ms(20); /* várunk 20 ms-ot */
     }
     return 0;
}
```

Vezérlés – felfedezőrobot

Célkitűzés

A mikrovezérlő teljesítőképességének leglátványosabb megnyilvánulása egy a természetes környezetben autonóm mozgásra képes robot vezérlése. Az első projekt célkitűzése ezért egy felfedezőrobot elkészítése volt, amely igyekszik minél több helyre eljutni, észleli az elé kerülő akadályokat, és megpróbálja kikerülni azokat, valamint egy PC számára videóképet továbbít. Nem volt célunk, hogy a robot gyors legyen, sem pedig az, hogy precízen mozogjon.

A robot fizikai felépítése

Az első lépés a feladat hatékony megoldásához szükséges minimális hardverkonfiguráció megtervezése volt.

- Mozgás és irányváltoztatás: legalább két motor szükséges. Mivel a gyorsaság nem követelmény, a pontos vezérelhetőség miatt módosított szervó motorokat választottunk. A robotnak két hajtott és egy kitámasztó kereke van.
- Navigáció: A távoli tárgyak érzékelésére minimálisan egy távolságmérőre van szükség. Erre a célra egy infravörös érzékelőt választottunk. Hogy a környezet a robot mozgása nélkül is feltérképezhető legyen, amire szűk helyen nem lenne lehetőség, az érzékelőt egy szervó motorra rögzítettük, amely így pásztázni tudja a környezetét.
- Ütközés detektálása: A környezet változása és a távolságérzékelő pontatlanságai miatt előfordulhat, hogy a robot akadályba ütközik. Az ütközés irányának azonosítása is fontos a szükséges korrekciós manőverek hatékony végrehajtására: A minimális igényeket két mikrokapcsolós kontaktus elhelyezése biztosíthatja.
- A kamera elhelyezése: Mivel a robot elsődlegesen a távolságérzékelő segítségével navigál, célszerű, ha a kamera a navigálás során feltáruló környezetet közvetíti, ezért a kamerát az érzékelő mozgatóplatformjára helyeztük el.

2. ábra: a felfedezőrobot

A robot viselkedése

A robot folyamatosan mozgatja az infravörös érzékelőt balról jobbra. Amelyik irányban az infravörös érzékelő messzebb érzékel akadályt, arra kanyarodik a robot. Ha valamelyik mechanikus érzékelő ütközik egy akadállyal, a robot tolat egy kicsit, majd helyben elforog kb. 45 fokot a megfelelő irányban. Ha az infravörös érzékelő túl közel érzékel akadályt, a robot visszatolat, „körülnéz", majd beforog abba az irányba, amerre legmesszebb „ellátott", és elindul.

Körülnézés

A körülnézés úgy történik, hogy a robot az infravörös érzékelőt szakaszosan forgatja balról jobbra. Azért szakaszosan, hogy az infravörös érzékelőnek legyen ideje pontos mérést végrehajtania.

Infravörös érzékelő – ADC

A Sharp GP2D12-es infravörös érzékelő egy háromvezetékes interfészen (VCC, GND, „kimenő feszültség") csatlakozik a mikrovezérlőhöz. A „kimenő feszültség" vezeték a 0-s AD

csatornára csatlakozik. A csatorna olvasásával az érzékelő előtti legközelebbi akadály távolságával arányos számot kapunk. A 10 cm-nél közelebbi és a 80 cm-nél távolabbi akadályokat azonban nem érzékeli. [13]

Robotvezérlő parancsok

```
void Haladj(long time);
// A robot max. sebességgel halad előre a megadott „time” elteltéig
void Haladji(int Balseb, int Jobbseb, long time);
// A bal kerék „balseb”, a jobb pedig „jobbseb” sebességgel
  forog „time” ideig, ahol a sebesség -20..20 lehet !!!

void Tolass(long time);
// A robot tolat max. sebességgel „time” ideig

void Jobbra(long time);
// A robot folyamatosan jobbra forog „time” ideig

void Balra(long time);
// A robot folyamatosan balra forog „time” ideig

void Allj(void);
// A robot megáll

uint16_t Korulnez(void);
// Megadja, hogy az infravörös érzékelő melyik pozícióban lát el legmesszebbre
// A visszatérési érték 120 és 500 között van
void Radar(uint16_t pos, uint16_t ido);
// Pos = 0..180

void Csap(void);
```

Szervók vezérlése PWM-jellel

A két kereket egy-egy módosított szervó hajtja meg. Ezeket impulzusszélesség modulált 50 Hz-es jelekkel vezéreljük. A vezérlőjeleket a processzor egy PWM-üzemmódban működő időzítő/számlálója szolgáltatja.

```
void ioinit(void) {
     TCCR1A = (1<<WGM11) | (1<<COM1A1) | (1<<COM1B1);
     TCCR1B = (1<<WGM13) | (1<<WGM12)  | (1<<CS12);
     /* set PWM value to 0 */
     OCR1A = 0;
     OCR1B = 0;
```

```
      DDRB |= (1<<PB5);
      DDRB |= (1<<PB6);
      ICR1 = 777;  /* 40 Hz * 1024/777 = 50 Hz */
      DDRD = 0;
      DDRD |= (1<<PD3);
}
```

Eredmények

A kitűzött célt sikerült elérni: megvalósult egy szemmel láthatóan ügyesen közlekedő, saját építésű robot. Szerkezetét és érzékelőit szabadon választhattuk meg, így felépítése hatékonyan igazodhatott a kitűzött célhoz. A megvalósított algoritmus a tapasztalatok szerint hatékonyan birkózik meg a természetes környezet akadályaival, akadálycsoportjaival. A robot életszerű mozgásának megfigyelése az általa közvetített videóképekkel kombinálva alkalmas a robotika és a mikrovezérlők iránti érdeklődés felkeltésére. Az akadályokat szemmel láthatóan igyekszik elkerülni, és ha mégsem sikerül neki, a mechanikus érzékelők segítségével észleli az ütközést, és a megfelelő irányba befordulva folytatja útját. A magasan lévő akadályokat megfelelő érzékelő híján nem veszi észre.

Mérőrendszer - Szélmérés

Célkitűzés

Célunk egy hosszabb távon (több hónapon át) emberi beavatkozás nélkül működő energiatakarékos és környezetbarát adatgyűjtő eszköz megvalósítása volt, amelytől a gyűjtött adatok rádiós kapcsolat segítségével lekérdezhetők, illetve a mérés paraméterei megváltoztathatók, a mérés leállítható, és újraindítható.

A konkrét feladat egy szélsebességmérő állomás megvalósítása. Az anemométer (1. ábra) a szélsebességgel arányos, változó frekvenciájú jelet szolgáltat. Ezt a frekvenciát kell mérnünk, és tárolnunk, illetve kommunikációs csatornát kell biztosítanunk az eszköz és a felhasználó között.

3. ábra: az anemométer

Az adatokat az eszköznek a mérés időpontjával együtt kell rögzítenie, így az eszköznek tartalmaznia kell egy valós idejű órát, amely az eszközön futó program újraindulásakor sem felejti el a pontos időt. Esetleges áramkimaradás, vagy helytelen használat miatt ne történhessen adatvesztés! Az eszköz legyen alacsony fogyasztású, és használjon valamilyen megújuló energiaforrást!

Szakértelem nélkül is telepíthető legyen, és a lehető legkevesebb felhasználói beavatkozást igényelje, ugyanakkor a tárolt adatokat bármikor le tudjuk tőle kérni! A mérés paramétereit – pl. a mintavételezési frekvenciát – felhasználóként is meg tudjuk változtatni!

A lekért adatok feldolgozásához ne legyen szükség speciális programra!

Az eszközzel ne csak közvetlen kábel kapcsolattal, hanem rádiós kapcsolattal is lehessen PC segítségével kommunikálni!

A kommunikáció soros porton keresztül történjen, és PC-oldalon egy terminálablakban tudjuk utasítani az eszközt, illetve fogadni a válaszait. Ez a kommunikáció alkalmazkodjon valamilyen szabványhoz, hogy PC-oldalon egy általános terminál emulátor programot használhassunk!

Hasonló feladatok megoldására természetesen vásárolható lenne egy ipari adatgyűjtő rendszer is. A tapasztalatok szerint azonban ezek a rendszerek zártak, nem teszik lehetővé a funkciók szabad módosítását, korlátozott a csatlakoztatható érzékelők száma, költségesek, és általában nem a legkorszerűbb technológiát alkalmazzák. Ezért döntöttünk egy saját berendezés kifejlesztése mellett. A nagykapacitású MMC (ill. SD) memóriakártyára gyakrabban rögzíthetünk adatokat, így lehetőségünk nyílik a későbbiekben a szélsebesség változásának statisztikai elemzésére is.

A szokásos adatgyűjtési funkció mellett, amely a berendezés időszakos meglátogatását igényli, az eszköz bővíthető közvetlen internetes kommunikációval.

Szélenergia

A hagyományos energiahordozók kimerülése és a használatukból adódó környezetszennyezés napjaink legfontosabb kérdései közé tartoznak. Az állandóan megújuló energiaforrások - köztük a szélenergia – kihasználása mindkét problémát jelentősen enyhíti.

Magyarországon az első világháború előtt közel 700 szélmalom működött, ebből több mint 500 az Alföldön. Ennek oka, hogy az Alföld volt a legnagyobb összefüggő gabona- és szemestakarmány termesztő vidék Magyarországon, és természetesen megfelelő volt a széljárás gyakorisága és intenzitása. De a robbanómotorok elterjedésével, majd a szénbányászat fejlesztése, az olcsó olaj és földgáz elterjedése révén teljesen háttérbe szorult. Azonban az ipar által okozott környezetkárosítás már 1955-ben világszerte olyan méreteket öltött, hogy Neumann János „Túlélhetjük-e a technológiát" című publikációjában párhuzamot vont egy esetleges atomháború következménye és az ipar által okozott környezeti pusztítás között.

Az első olajválság és az OECD ajánlás együttes hatásának köszönhető, hogy Európában végérvényesen elindult a megújuló energiaforrások, így a szélenergia hasznosításának fejlesztése és elterjedése.[2]

A közelmúltban kezdte meg működését két, egyenként 1,5 MW-os szélerőmű az Alföldön, Mezőtúron és Törökszentmiklóson. Nemrégiben pedig arról érkezett hír, hogy a Felsőzsolcán már egy 1,8 MW-os szélerőművet működtető cég Debrecen környékére szándékszik újabbat telepíteni.[1]

Bármely szélerőgép/szélerőmű telepítés első fázisa a hely kijelölése. A cél a legjobb széljárású helyszín meghatározása, figyelembe véve a lehetséges műszaki-gazdasági korlátokat is.

A lehetséges helyszínek kiválasztásához a széltérképek, ennek hiányában a közeleső mérőhelyek széladatbázisai (pl. meteorológiai, repülőtéri, mezőgazdasági mérőhelyek adatai) nyújthatnak támpontot. Energetikai szempontból azok a helyszínek ígéretesek, ahol a telepítés tervezett magasságában a várható évi átlagos szélsebesség legalább 6 m/s. A szélerőgépek élettartama is véges, a gyártók általában 20-25 évre tervezik a berendezéseket.

A szélmérés egyik változata, amikor hosszabb időszakon keresztül, szélmérő tornyokra telepített anemométerekkel vizsgálják a szelet. Ehhez a szélmérő eszközhöz egy adatgyűjtő egységet csatlakoztatnak, mely az adatokat egy tároló egységben tárolja, majd ezeket az adatokat szakemberek elemzik.[3]

A szélerőmű által kinyerhető energia meghatározása szempontjából rendkívül fontos a szélsebesség ismerete. A szél által hordozott kinetikusenergia-sűrűség $1/2\rho v^2$ amely a v szélsebességgel áramlik, így a teljesítményáram-sűrűség $1/2\rho v^3$, azaz a szélsebesség köbével arányos. A szélerőmű által kinyerhető energia ezért nagyon érzékeny - a harmadik hatványtól függ - a telepítés helyén mérhető szélsebességre, és annak időbeli lefutására.[5]

4. ábra: szélerőművek Hegyeshalomnál

A hardver leírása

Anemométert az Alter Energetikai Irodától (www.alterweb.eu) kaptunk. Az anemométer erősítetlen analóg jelet szolgáltat, ezért szükségünk volt egy erősítőre, amely a jelet felerősíti, és egy komparátorra, amely digitális jellé alakítja át. A jel feldolgozására az Atmel cég ATmega128L alacsony fogyasztású AVR processzorát használtam.

A processzort közös panelra integráltuk egy PFC8583 típusú valós idejű órával, egy soros csatlakozóval és egy memóriakártya-foglalattal.

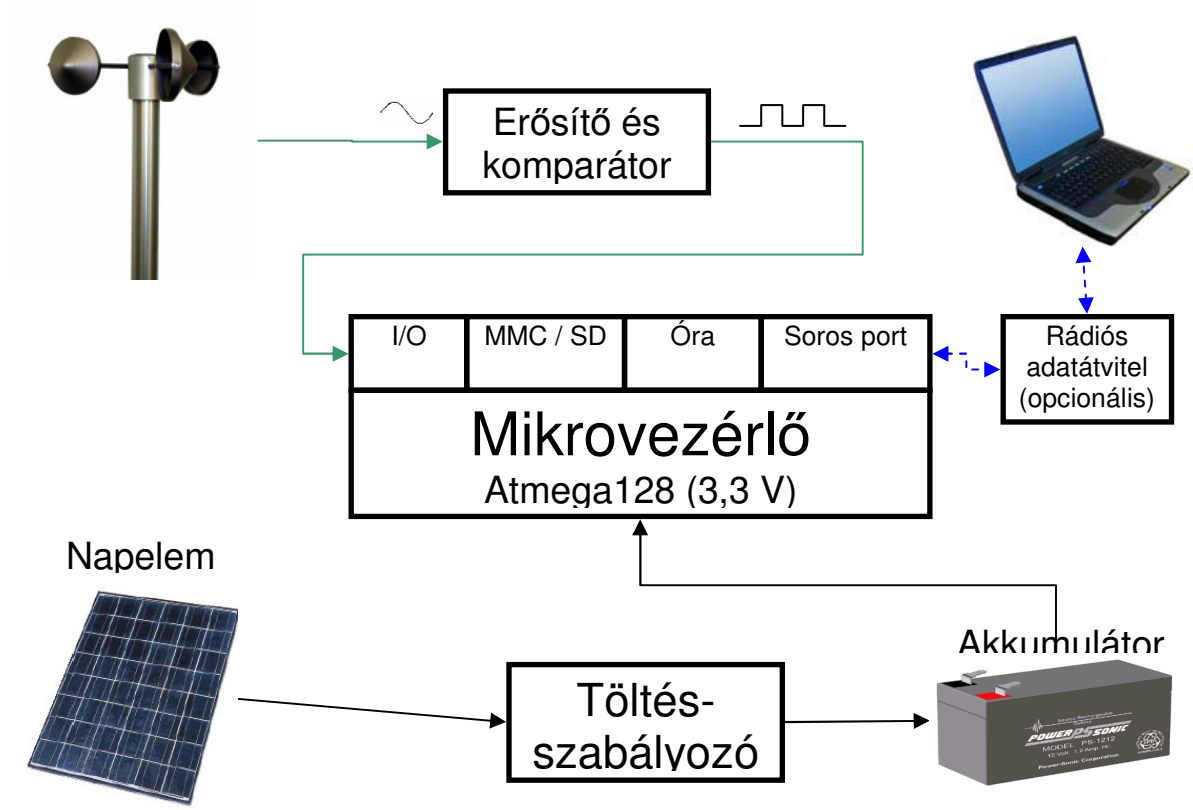
Az egyes hardverelemeket akkumulátorról tápláljuk, kivéve a valós idejű órát, mert az saját elemmel rendelkezik, így a mikrovezérlő áramtalanítása miatt nem kell újra és újra beállítani az órát.

Az MMC kártya foglalatát először egy hosszabb vezetékkel csatlakoztattuk a processzorhoz, azonban így nem működött, ezért a foglalatot ráintegráltuk a mikrovezérlő paneljére. Mivel a kártya magas (MHz-es nagyságrendű) frekvencián kommunikál, a hosszú vezetéken a jel jelentősen torzult.

A mikrovezérlőt akkumulátorról tápláljuk, az akkumulátort pedig napelemek segítségével (nappal) folyamatosan töltjük. A napelem az áramot nem közvetlenül az akkumulátornak, hanem egy töltésszabályozónak adja le, amely biztosítja az akkumulátor napfényintenzitás-független töltőfeszültségét.

A processzort, hogy keveset fogyasszon, 1 MHz-es órajelen működtetjük. Ilyen órajel-frekvencia mellett a soros kommunikáció sebessége legfeljebb 4800 bit/sec lehet. Az így kapott kommunikációs csatorna hibája 0.16%. Ha 9600 bit/sec-mal kommunikálnánk, a hiba 7.84%-os lenne, ami gyakorlatilag ellehetetlenítené a kapcsolatot.

A mikrovezérlőt a soros portján keresztül csatlakoztathatjuk PC-hez. A PC tipikusan egy hordozható számítógép, amelyet magunkkal vihetünk a szélmérés helyszínére. Azonban a szélmérőt egy torony tetejére telepítik, így a PC-t és a mérőeszközt összekötő kábelt érdemes rádiós kommunikációval kiváltani. A Szilárdtest Fizika Tanszék vásárolt a ProControl nevű cégtől két darab RFP-3 típusú adóvevőt, melyek a PC-vel illetve a mikrovezérlővel soros porton keresztül kommunikálnak, így a mikrovezérlőn futó programot nem is kellett átírnom a rádiós kommunikáció megvalósítása miatt.



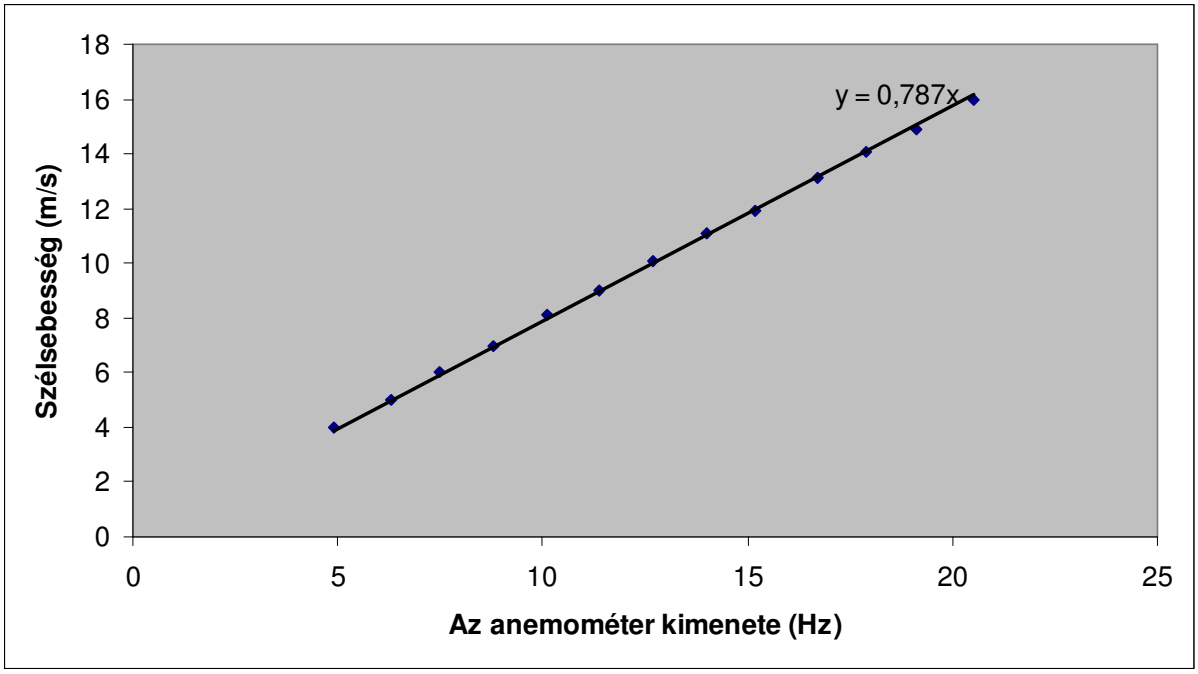
5. ábra: a rendszer blokkdiagramja és a kihelyezett rendszer

Szélsebességmérés

A szélsebesség és frekvencia kapcsolata

Nem közvetlenül a szélsebességet, hanem az anemométertől kapott elektromos jel frekvenciáját mérem. A felhasznált anemométert a gyártója hitelesítette, a szélsebesség és frekvencia kapcsolatát az alábbi táblázat illetve diagram szemlélteti:

Referencia szélsebesség (m/s)	Az anemométer kimenete (Hz)
4	4,9
6	7,5
8,1	10,1
10,1	12,7
11,9	15,2
14,1	17,9
16	20,5
14,9	19,1
13,1	16,7
11,1	14
9	11,4
7	8,8
5	6,3



6. ábra: az anemométer hitelesítési diagramja

Frekvenciamérés

Nem közvetlenül a jel frekvenciáját, hanem a periódusidejét mérem (T=1/f). Az idő mérésére a processzor egyik számlálóját használom. Ez egyenletesen számlál 0-tól 65535-ig, majd ismét 0-65535-ig, sít. A mérésnek egyetlen paramétere van, a mintavételezési periódusidő. A mintavételezés egy periódusának kezdetén a számlálót lenullázom. Valahányszor a szélkerék fordul egyet, a processzor ICP1 (Input Capture Pin 1) bemenetén megjelenik egy impulzus, amely megszakítást okoz. A megszakításkezelő kódrészlet megjegyzi a számláló aktuális értékét, majd lenullázza a számlálót (és a számláló nulláról indulva számlál tovább). Azonban ha a szélkerék sokáig áll, a számláló túlcsordul. A túlcsordulás szintén megszakítást vált ki, és a megfelelő megszakításkezelő kód pedig feljegyzi, hogy túlcsordulás történt, így a program számon tartja, hogy az adott periódus alatt hány túlcsordulás volt. A periódus végén a megjegyzett számlálóértékek és túlcsordulások segítségével egy értéksorozatot kapunk, amely legjellemzőbb elemét tekintjük az adott periódus periódusidejének. Ezeket a periódusidőket egy pufferbe gyűjtöm. A legjellemzőbb elemnek a mért értékek mediánját tekintem, így szűröm ki a mérési hibákat (melyek pl. a nulla-átmenet detektálásakor esetlegesen fellépő elektronikus zajok).

Az ICP1-re érkező impulzus megszakítását a következő eljárás kezeli le:

```
SIGNAL(SIG_INPUT_CAPTURE1) {
     pido[pidoi++] = TCNT1+hozadek*65536;
     hozadek=0; /* a túlcsordulásokat lekezeltük */
     TCNT1 = 0; /* a számláló nullázása */
}
```

A túlcsorduláskor a következő eljárás fut le:

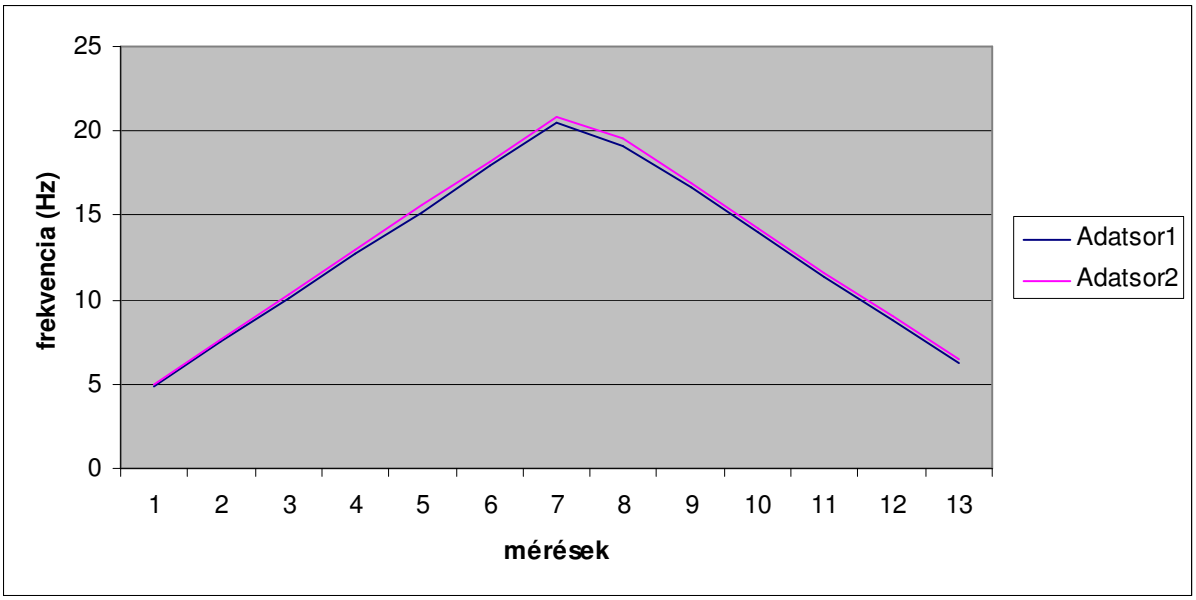
```
SIGNAL(SIG_OVERFLOW1) {
     hozadek++;
}
```

A frekvenciamérés hitelesítése

A periódusidőket 1 / (1 MHz / 1024) ≅ 1 ms-os egységekben kapjuk meg, de a pontos értéket méréssel állapítottam meg. Jelgenerátor segítségével meghatározott frekvenciájú jeleket kapcsoltam a processzor ICP1 bemenetére, és feljegyeztem, hogy a processzor milyen mérőszámot rendel a mért periódusidőhöz.

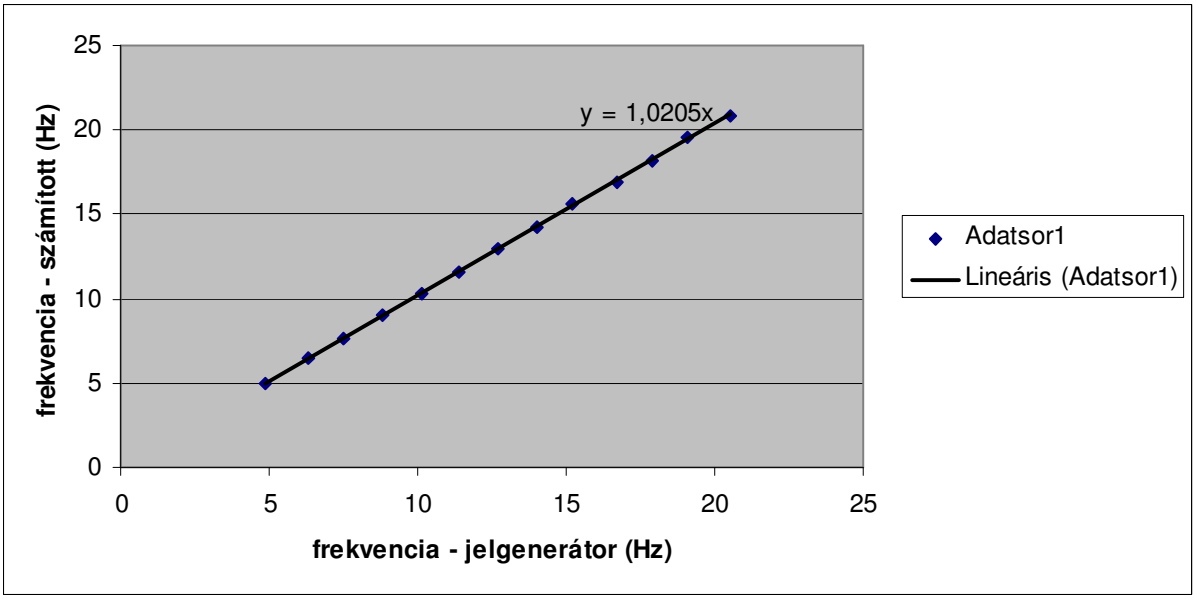
Jelgenerátor (Hz)	Mért periódusidő (ms?)	Számított frekvencia (Hz)
4,9	200	5,00
7,5	130	7,69
10,1	97	10,31
12,7	77	12,99
15,2	64	15,63
17,9	55	18,18
20,5	48	20,83
19,1	51	19,61
16,7	59	16,95
14	70	14,29
11,4	86	11,63
8,8	111	9,01
6,3	155	6,45

A következő diagramon kékkel jelöltem a processzorra kapcsolt jel frekvenciáját, és pirossal a mért periódusidőkből számolt frekvenciát (a számláló egy egységét 1ms-nak feleltettem meg).



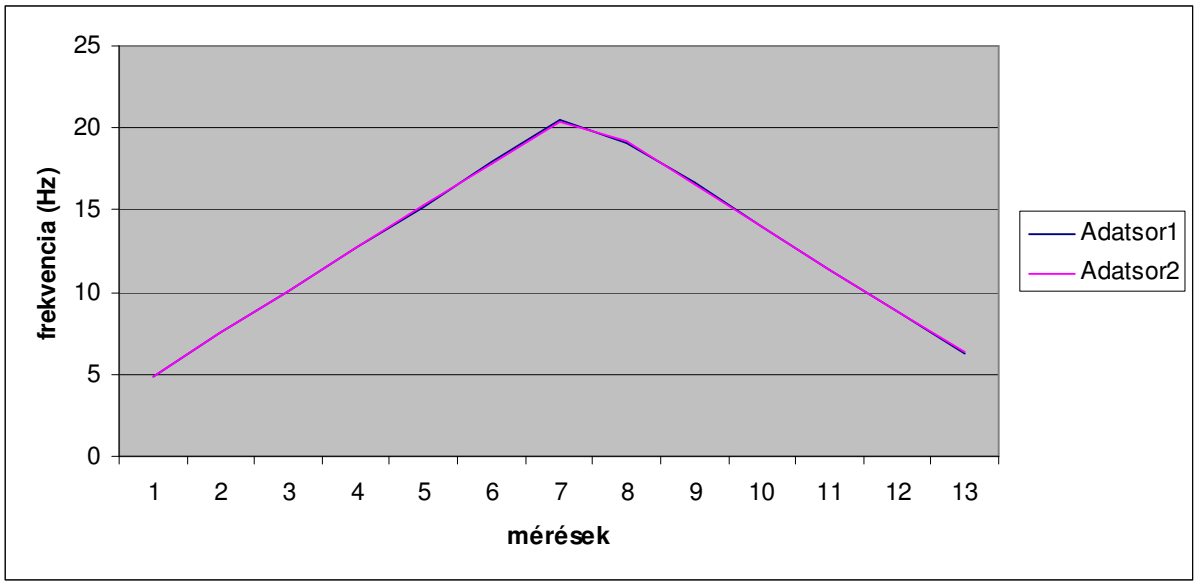
7. ábra: a frekvenciamérő hitelesítés előtti eredménye

A két értéksor szemmel láthatóan eltér. Az egyik értéksort a másik függvényében ábrázolva legkisebb négyzetes illesztéssel meghatároztam az arányossági faktort.



8. ábra: a regressziós egyenes

Az 1,0205-ös faktorral megszorozva a számított frekvenciákat jó közelítéssel megkapjuk a megfelelő hitelesítési frekvenciákat. A hitelesítési és az így számított frekvenciákat a következő diagram mutatja:



9. ábra: a frekvenciamérő hitelesítés utáni eredménye

Mindezek alapján a mért periódusidőből a szélsebességet a következő képlettel számíthatjuk:
v = ((1000/T)/1,0205)*0,787 [m/s] = 771,19 / T [m/s]

Adatgyűjtés

Az adatgyűjtés a „start” parancs kiadására - illetve az eszköz bekapcsolása utána öt másodperccel automatikusan - indul el, és a „stop” parancs hatására áll le. Amikor kiadjuk a „start” parancsot, a program lekérdezi a rendszeridőt, ennek segítségével generál egy egyedi fájlnevet, és ettől kezdve az ilyen nevű fájlba ír.

Mivel a médiumon FAT16 fájlrendszer van, a fájlnév csak 8+3 karaktert tartalmazhat. A valós idejű óra az aktuális időt 6 bájton tárolja, azonban ezek közül a legelsőre (amely a századmásodperceket kódolja), nincs szükség. Így a fennmaradó 5 bájtot 16-os számrendszerben ábrázolva 10 bájtot kapunk. A 10 bájt mindegyikéhez hozzáadjuk a ’A’ betű ASCII kódját. Az így kapott 10 bájt első 8 bájtja lesz a fájlnév pont előtti része, a maradék kettő pedig a kiterjesztése.

```
i2c_init();
GetRTC(RtcBuffer);
sprintf(filename,”%c%c%c%c%c%c%c%c.%c%c”,
   ’A’+RtcBuffer[1]/16,’A’+RtcBuffer[1]%16,
   ’A’+RtcBuffer[2]/16,’A’+RtcBuffer[2]%16,
   ’A’+RtcBuffer[3]/16,’A’+RtcBuffer[3]%16,
   ’A’+RtcBuffer[4]/16,’A’+RtcBuffer[4]%16,
   ’A’+RtcBuffer[5]/16,’A’+RtcBuffer[5]%16
        );
rprintfStr(filename); rprintf(„ created.\r\n”);
```

Amikor betelik a puffer, kiszámoljuk a puffer értékeinek összegét, átlagát, szórását, meghatározzuk minimumát és maximumát, majd ezeket a fájl végéhez fűzzük.

A kártyát csak a „stop” parancs kiadása, azaz az adatgyűjtés leállítása után szabad kivenni. Ha adatgyűjtés közben az adathordozót eltávolítjuk az eszközből, az addig fájlba írt adatok megmaradnak, nem vesznek el, azonban a kártya visszahelyezése önmagában nem elegendő az adatgyűjtés folytatásához, hanem újra kell indítani az eszközt a reset gomb

megnyomásával. Természetesen, ha a mérést szabályosan leállítottuk a kártya eltávolítása előtt, nem szükséges újraindítani az eszközt a mérés folytatásához.

Adattárolás

A mért adatokat a program fájlokba írja, a fájlokat MMC (Multi Media Card) vagy SD (Secure Digital) memóriakártyán tárolja. A fájlba húszbájtos rekordokat ír ki, melyek a következő szerkezettel rendelkeznek:

- dátum és idő: 6 bájt
- mintavételezési periódusidő: 2 bájtos egész
- a puffer mérete: 2 bájtos egész
- a puffer elemeinek
    - összege: 2 bájtos egész
    - átlaga: 2 bájtos egész
    - szórása: 2 bájtos egész
    - minimuma: 2 bájtos egész
    - maximuma: 2 bájtos egész
- egy szorzófaktor: az átlagot és a szórást ezzel a számmal szorzom fel, majd egészekre kerekítem a tárolás előtt

Így három másodperces mintavételezési periódusidő és 20-as pufferméret esetén percenként egy rekordot írunk a fájlba. Ilyen paraméterekkel egy 256 MB-os MMC kártyával kb. öt és fél hónapig lehet folyamatosan adatot gyűjteni.

A fájlokban a dátumidő-kódokat leszámítva kizárólag egész számokat tárolok, elkerülve így a számkonverziós problémákat.

A kártyát egy USB-s kártyaolvasóba helyezve és a kártyaolvasót egy PC-hez csatlakoztatva a PC-n futó operációs rendszer a kártyát felismeri, mint adathordozót, és elérhetővé teszi a kártyán tárolt fájlokat. A fájlokban levő adatokat egy egyszerű programmal táblázattá lehet alakítani, melyet pl. az Excel importálni tud, így az adatok további feldolgozása egyszerűvé válik. Ez a konverter program akár a kártyán is tárolható (az én változatom történetesen 20 kB-os), illetve a kártyáról futtatható (nem szükséges a PC-re másolni).

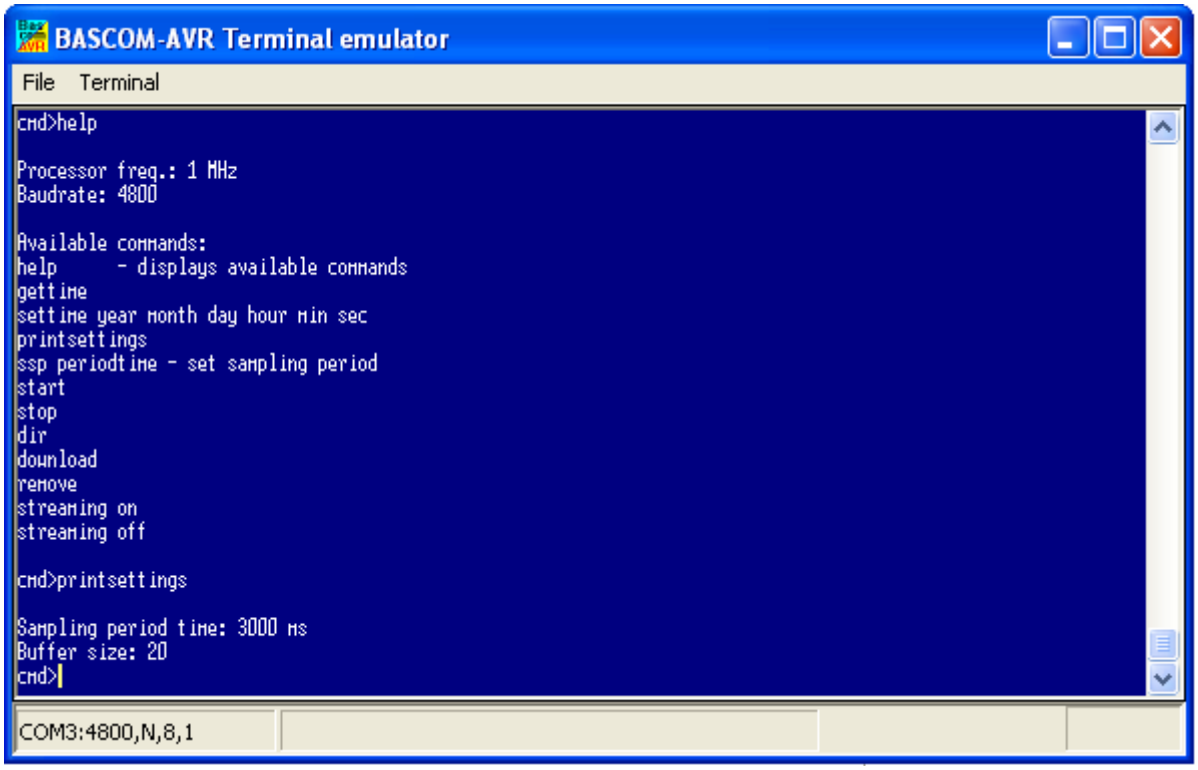
Parancsértelmező

Az eszközzel soros interface-en keresztül, parancsok kiadásával kommunikálhatunk.

Parancsok:

- help: kiírja a kiadható parancsokat
- gettime: lekérdezi a rendszeridőt
- settime <év> <hó> <nap> <óra> <perc> <másodperc>: beállítja a rendszeridőt
- printsettings: kiírja a mintavételezési periódusidőt és a puffer méretét
- ssp <periódusidő>: beállítja a mintavételezési periódusidőt (ms-okban)
- start: elindítja az adatgyűjtést
- stop: leállítja az adatgyűjtést
- dir: kilistázza a kártyán található fájlok neveit
- download <fájlnév>: kilistázza a megadott fájlt
- remove <fájlnév>: törli a megadott fájlt
- streaming on: bekapcsolja a mért adatok folyamatos kiíratását
- streaming off: kikapcsolja a mért adatok folyamatos kiíratását

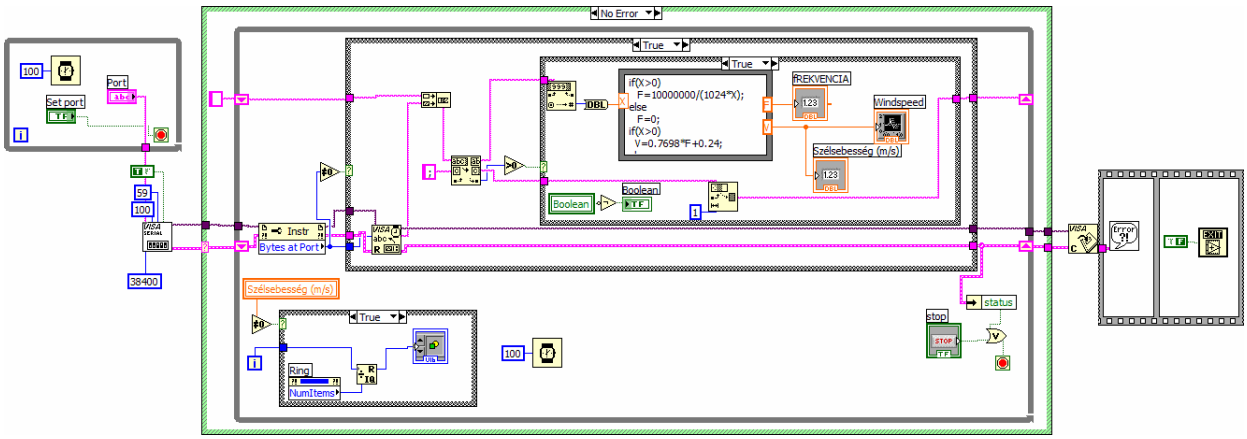
A parancsokat PC-n adhatjuk ki, valamilyen terminal emulator program segítségével.



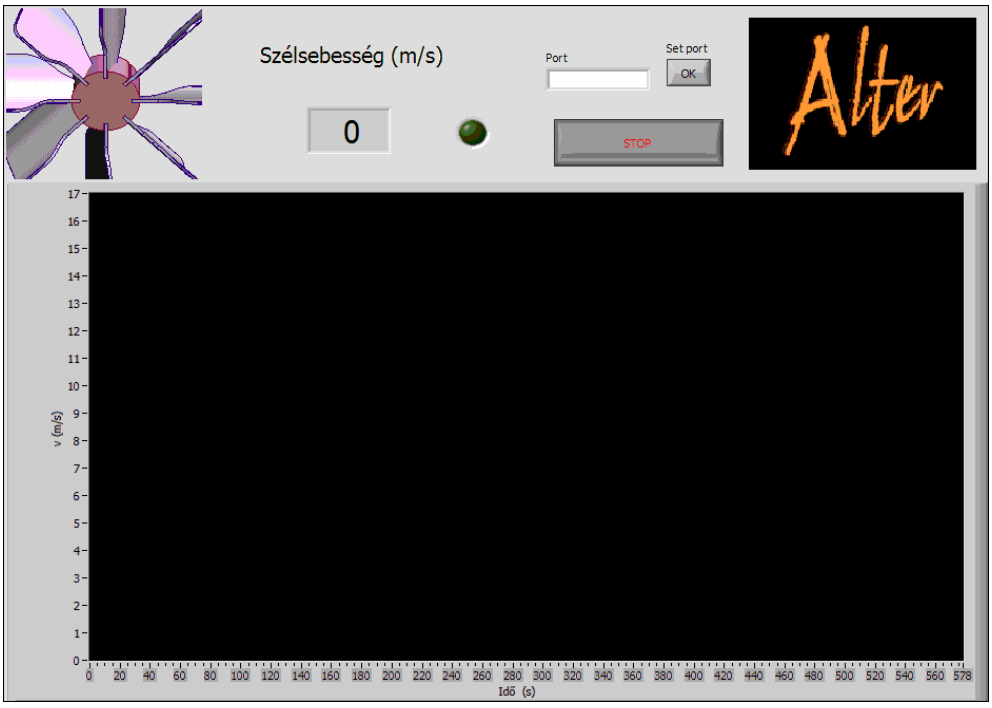
10. ábra: kommunikáció a mérőrendszerrel terminálon keresztül

A rádiós kommunikáció tesztelése

Ha kiadjuk a „streaming on” parancsot, és elindítjuk a mérést (vagy akár mérés közben adjuk ki ezt a parancsot), akkor a mikrovezérlőn futó kód folyamatosan írja ki a mért periódusidőket a soros portra. Az alábbi PC-oldali program LabView-ban készült. A soros port kiválasztása után fogadja a periódusidőket, kiszámítja belőlük az aktuálisan mért szélsebességet, majd a szélsebességértékeket grafikusan megjeleníti. A program elsősorban tesztelési és demonstrációs célokat szolgál, és akár kültéri szélsebességmérő adatait is megjeleníthetjük vele. A szélsebességmérő forgását a panelen lévő propeller forgása jelzi. Az adatok beérkezését pedig a LED villogása mutatja.



11. ábra: a program blokkdiagramja



12. ábra: a program felhasználói felülete

Óra

Az eszköz órája egy PFC8583 jelű mikrochip. Ez az év tárolására csupán 2 bitet használ, így nem tudja tárolni az évszámot, hanem csak 4 év megkülönböztetését teszi lehetővé, de mivel a mérések néhány hetesek, vagy hónaposak, ez nem jelent gondot.

Az óra I^2C buszon keresztül kommunikál a processzorral. Az AVRLib programkönyvtár tartalmaz I^2C-s kommunikációt, sőt szoftveres valós idejű órát megvalósító függvényeket is, azonban nekünk hardveres valós idejű órára van szükségünk, mert a szoftveres óra áramszünet vagy a program újraindulása esetén elfelejti, hogy mennyi az idő. Az órához sikerült az Interneten C-mintakódot találni, azonban ez nem volt kompatibilis az AVRLib I^2C-s rutinjaival, így kerestem egy megfelelő I^2C-s rutingyűjteményt is. Találtam is olyat, ami látszólag megfelelt, de az óra mégsem működött. Oszcilloszkóp segítségével találtam meg, hogy a program mely pontján, és hogyan kell megváltoztatnom bizonyos időzítéseket, hogy a mi konkrét hardverünkkel megbízhatóan működjön az óra.

Az óra az időt 6 bájton, BCD - kódolással tárolja:

0.	1.	2.	3.	4.	5.
szmp.	Mp.	perc	óra	nap	Hónap

A 4. bájt legfelső 2 bitje határozza meg az évet.

Példa:

0.	1.	2.	3.	4.	5.
120	49	69	33	160	17

0.	1.	2.	3.	4.	5.
78_{16}	31_{16}	45_{16}	21_{16}	20_{16}	11_{16}

2-es év 11-20, 21:45:31.78

GPRS kommunikáció

Az adatok begyűjtéséhez az üzemeltetőnek rendszeresen meg kell látogatni a berendezést, ami jelentős költséggel járhat. Gondot jelenthet az is, hogy a meghibásodás esetén csak utólag derül ki a hiba. Ezen problémák megoldhatók, ha távoli elérést biztosítunk az állomás számára, ami egy GPRS telefon modul segítségével valósítható meg. A kapcsolat segítségével lehetőség van az adatok időszakos feltöltésére egy webszerverre. Ha a mérőállomástól meg szeretnénk tudni, hogy működik-e még, vagy a mérőállomásnak valamilyen meghibásodást kell jeleznie, akkor egyszerűen küldhet egy SMS-t üzemeltetőjének, illetve az üzemeltetője SMS-ben utasíthatja a mérőeszközt a mérés valamely paraméterének, pl. a mintavételezés gyakoriságának megváltoztatására.

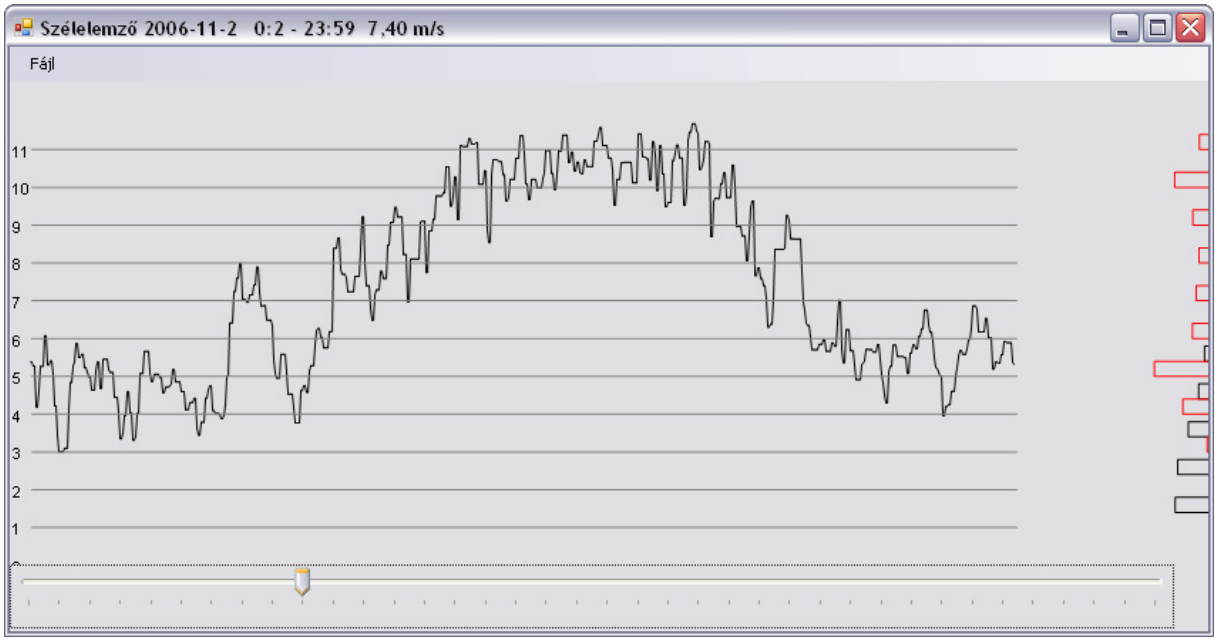
A Szilárdtest Fizika Tanszéken elkészült egy PC-n futó, soros porton keresztül mobiltelefonnal kommunikáló programkönyvtár, melynek legtöbb függvényét módosítás nélkül le lehet fordítani az általunk használt AVR processzorra. A programkönyvtár platformfüggő függvényeit – csupán néhány függvényről van szó – már megírtam AVR-platformra is, és terveink szerint a mérőeszközünk képes lesz SMS-t küldeni és fogadni illetve adatokat továbbítani.

Érkezhet SMS a készülékre a mobilszolgáltatótól is, az ilyen üzenetet a mérőeszköz automatikusan továbbítja a felhasználó felé. Ilyen üzenet lehet pl. a mobil-előfizetés állapotáról szóló értesítés. Ha a felhasználó nem tesz eleget fizetési kötelezettségeinek, akkor a mérőállomás képtelen az adatokat továbbítani.

Terveink szerint az eszköz naponta egyszer vagy kétszer nézi majd meg, hogy kapott-e SMS-t, és ilyenkor lesz lehetősége SMS küldésére, ugyanis a GPRS - kapcsolat folyamatos üzemeltetése rendkívül energiaigényes lenne.

Mérési eredmények

A mérőállomás kihelyezésre került az Ater Energetikai Iroda egyik ügyfeléhez, és már fél éve üzemel. Az adatok kiértékelésére készítettem egy elemző programot, amely a mért értékeket napi bontásban teszi áttekinthetővé.



13. ábra: szeles nap

A grafikonon a szélsebesség időbeli lefutása látható. A fejlécen olvasható, hogy éppen melyik nap adatai látszanak, illetve, hogy mennyi volt aznap az átlagos szélsebesség. Az ablak jobb szélén két hisztogram látható. A piros színnel rajzolt az aktuális napra, a másik a teljes mérésre vonatkozik. Az ablak alján található csúszkával lehet váltani a napok között (egérrel vagy a nyíl billentyűkkel). A további feldolgozáshoz (a szélerőmű várható energiahozamának kiszámításához) a binárisan tárolt adatokat a program szöveges fájllá alakítja át, mely Excellel megnyitható, és feldolgozható.

Összefoglalás

Sikerült megvalósítanunk egy szélsebességmérő eszközt, mely a mért adatokat MMC (vagy SD) kártyán tárolja. Az eszköz alacsony fogyasztású és energiaellátását nagy részben napelemekkel oldottuk meg. Soros porton keresztül PC-hez csatlakoztatható, és egy általános terminál emulátor program segítségével egy terminálablakban tudunk az eszköz processzorán futó programmal kommunikálni. A szélsebességet nem közvetlenül mérem, hanem több lépésben határozom meg. Az anemométertől (szélkerék) kapott jel periódusidejét mérem, ebből kiszámítom a jel frekvenciáját, majd a frekvenciából a szélsebességet. Az eszköz rendelkezik egy hardveres valós idejű órával, amely azért fontos, mert a mért adatokat a mérés időpontjával együtt írom fájlba. Az óra a processzor áramtalanítása vagy újraprogramozása esetén sem felejti el az időt.

Az eszköz további fejlesztésén jelenleg is dolgozom. Készül a GPRS kommunikáció átültetése PC-s környezetből a mikrogépes rendszerre. Lehetőség van további érzékelő alkalmazására is. Így az eszköz alkalmassá tehető akár hőmérséklet vagy páratartalom mérésére is.

Szabályzás - klímakamra

Célkitűzés

Célunk egy klímakamra megvalósítása volt, amelyet 85°C-os hőmérsékletre akarunk felhevíteni 85%-os relatív páratartalom mellett, és ezt napokig kívánjuk tartani. Ezt a berendezést panelek öregítési vizsgálatához szántuk.

Megvalósítási tanulmány

A kemence fűtését mikrovezérlővel vezéreljük, hőmérsékletét hőmérővel mérjük. Mivel a kemence lassan melegszik, és lassan is hűl, pusztán a hőmérő által mért adatból nem tudjuk sem azt, hogy mennyi a kemence átlagos hőmérséklete, sem azt, hogy milyen intenzív fűtést kell alkalmaznunk a kívánt hőmérséklet eléréséhez. A késlekedést a folyamat holtidejének nevezzük. [11]

A holtidő a folyamat fizikai tulajdonsága, amely a berendezés megtervezése és megvalósítása folyamán jön létre. A hőmérséklet szabályzására ún. PID-szabályzást szoktak alkalmazni. Nekünk elég volt egy proporcionális szabályzó is, hiszen egy konkrét kemencével adott hőmérsékletet akartunk tartani. A páratartalmat a hardver hiányosságai miatt csak mérjük, nem szabályozzuk.

A szoftver egy LCD-kijelzővel ellátott Atmega128-as mikrovezérlőn fut, így folyamatosan láthatjuk, hogy éppen mennyi a klímakamra hőmérséklete és páratartalma.

PID-szabályzás

A legegyszerűbb szabályozók, az állásos szabályozók, amelyek a szabályozott jellemzőt (hőmérséklet, nyomás, villamos áram, villamos feszültség, stb) összehasonlítják az alapjellel, és annak elérésekor kikapcsolják a kimenetet, majd megvárják, míg a szabályozott jellemző a túllendülés után újra lecsökken az alapjel értékére, és bekapcsolják a kimenetet. Mivel a gyakorlatban előforduló rendszereknek holtidejük van (az érzékelő csak késve tudja a szabályozott jellemzőt mérni), a szabályozás lengésekkel jön létre. Minél nagyobb a holtidő, és minél erősebb a beavatkozás, annál nagyobb a lengések mértéke. Vannak rendszerek, amelyek technológiai tulajdonságait ezek a lengések teljesen lerontják, így nem szabad állásos szabályozót használni azokhoz a technológiákhoz, amelyeknél a szabályozott jellemző pontos

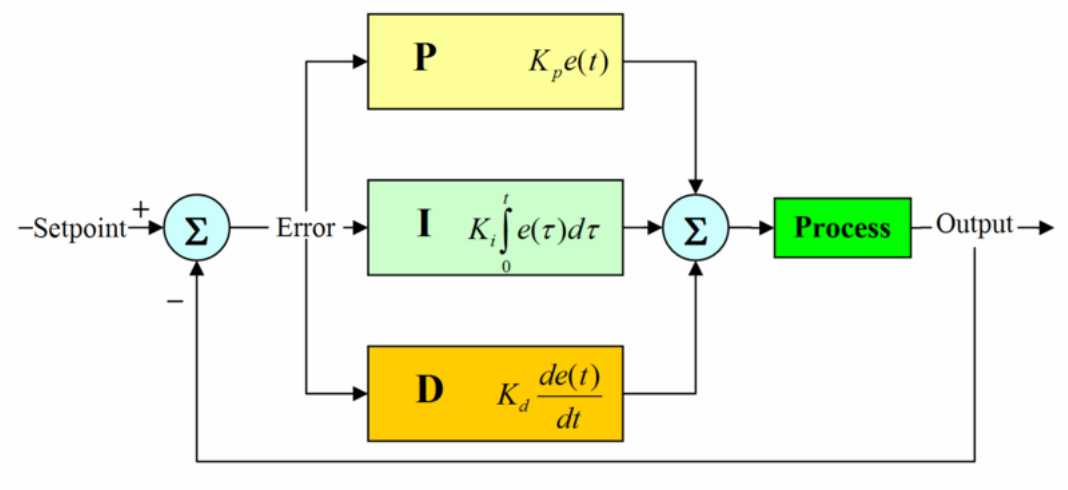
tartása fontos. Ilyen technológia a hőkezelés, de különösen a kerámiák, a fémek és műanyagok hőkezelése. Laboratóriumokban elemzés előtt a szerves anyagokat elhamvasztják. Az alacsony és a magas hőmérséklet egyaránt meghamisítja az elemzési eredményt.

A korszerű szabályozó korszerű PID algoritmussal rendelkezik, amely a szabályozó minden állapotában, minden zavarójel hatását a legrövidebb idő alatt megszünteti és PID paramétereitől függően a túllövést korlátozza.

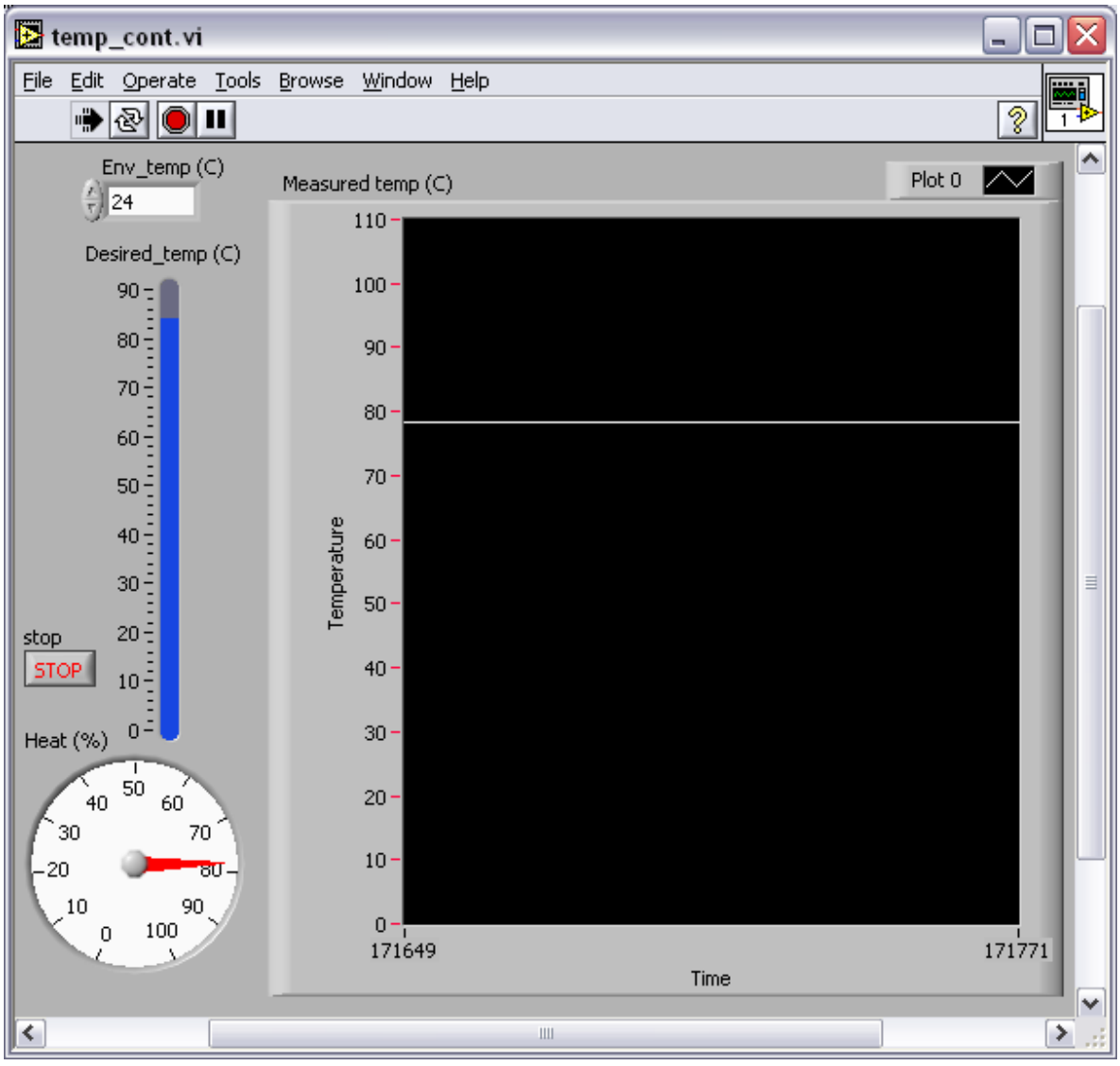
A "P" szabályozó

Vizsgáljuk meg a P szabályozókat, amelyek a szabályozott jellemzőt (tulajdonságot: nyomást, hőmérsékletet, sebességet, stb) az alapjel környezetében arányosan szabályozzák úgy, hogy a kimenetet a tartomány alsó határán 100 %-ra, felső határán 0 %-ra állítják be. Ezt a szabályozást fordított szabályozásnak nevezzük. Az egyenes szabályozásnál a határértékek felcserélődnek. A fordított szabályozást fűtésre, az egyenes szabályozást hűtésre lehet használni. Ezek a szabályozók az alapjel értékénél a kimenetet 50%-ra állítják be (léteznek olyan szabályozók is, amelyeknél az alapjel 0%-nál van). Természetesen csak véletlenül létezik olyan rendszer, amely az alapjelnél pontosan 50%-os (vagy 0%-os) kimenetet igényel. Azok a rendszerek, amelyek ennél kisebb kimeneti értéket igényelnek, túllendülnek és tartósan így maradnak. Ennek fordítottja is érvényes. Minél nagyobb a szabályozási arányossági tartomány az eltérés annál nagyobb.

Vizsgáljuk meg, hogyan szabályozza a P szabályozó a kemencét. Például egy 600 °C-ra beállított 100 kW-os kemencébe az alapjelnél a szabályozó 50 kW teljesítményt vezet be. Ha az említett kemence lényegesen kevesebb (~30 kW) teljesítménnyel tartható 600 °C-on, a szabályozó arányosan addig növeli a hőmérsékletet, ameddig szerinte a kimenet 30 % (30 kW) nem lesz. Az arányos tartománytól függően ez az érték legyen például 53 °C. A kemence tehát 653 °C-os lesz és tartósan így marad. Ez az eltérés az OFFSET. A kemencék szabályozásánál az OFFSET a hőntartás folyamán változik, mert a hővel való telítődés miatt egyre kevesebb teljesítmény szükséges. Tovább romlik a helyzet akkor, mikor az alapjelet (a beállított hőmérsékletet) váltakozóan felülről, vagy alulról közelítjük meg. Ebből következik, hogy P, PD szabályozót ilyen szabályozásokhoz nem célszerű alkalmazni. [11]



14. ábra: PID-szabályzás



15. ábra: Labview-ban megírt kemence-szimulátor

Hőmérsékletmérés

A mérést a Sensirion cég SHT11 nevű hőmérséklet- és páratartalom-mérő érzékelőjével végezzük. A chip-pel egy kétirányú, kétvezetékes interfészen lehet kommuikálni, amely azonban nem I^2C-kompatibilis. A két vezeték elnevezése: SCK, DATA.

Az SCK a kommunikáció szinkronizálására szolgál. A DATA egy háromállapotú láb, amelyen az adatok mindkét irányban közlekednek. Lefutó élre változik, és felfutó élre válik érvényessé. Egy bit átvitele alatt a DATA vezetéken nem változhat a jel mindaddig, míg az SCK magas szinten van. A mikrovezérlő csak alacsony állapotba állíthatja a DATA-t.

Átvitel indításához egy ún. "Transmission Start" jelsorozatot kell küldeni. Ez abból áll, hogy a DATA-t lehúzzuk, míg az SCK-t felhúzzuk, majd az SCK-t lehúzzuk és a DATA-t felhúzzuk, mialatt az SCK-t újra felhúzzuk.

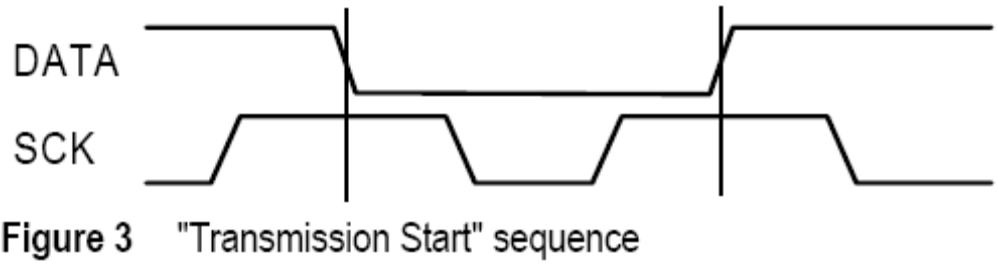


Figure 3 "Transmission Start" sequence

16. ábra: Transmission Start

Ezt követően kiadhatunk egy parancsot. A parancsok 8-bitesek: 3 címbit + 5 parancsbit. Az SHT11 egy parancs érkezését úgy nyugtázza, hogy a 8. órajel lefutó éle után lehúzza a DATA-t. A 9. órajel lefutó éle után pedig felengedi a DATA-t.

Command	Code
Reserved	0000x
Measure Temperature	**00011**
Measure Humidity	**00101**
Read Status Register	00111
Write Status Register	00110
Reserved	0101x-1110x
Soft reset, resets the interface, clears the status register to default values wait minimum 11 ms before next command	**11110**

Table 2 SHTxx list of commands

17. ábra: SHT11 parancsok

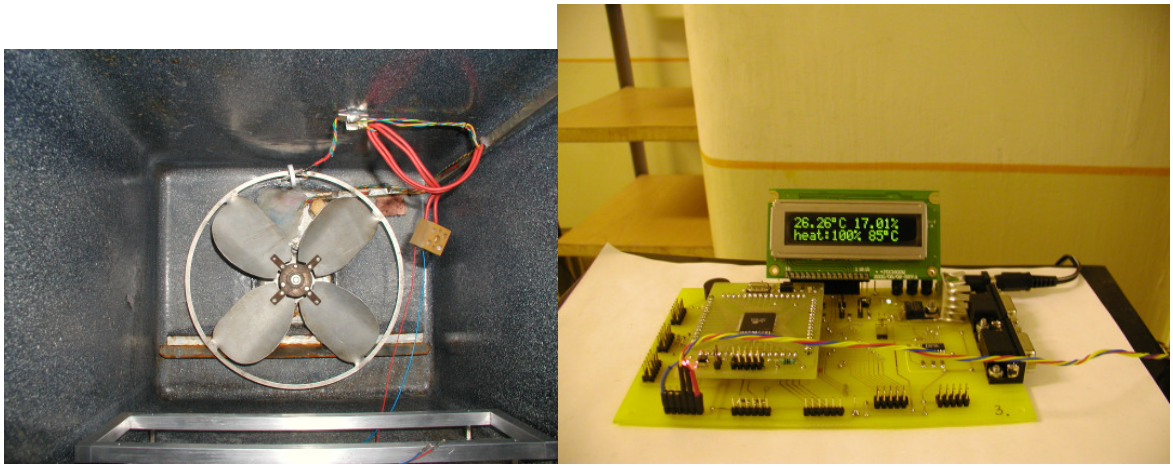
A megfelelő parancs kiadása után („00000101” a páratartalom, „00000011” a hőmérséklet méréséhez) a mikrovezérlőnek meg kell várni a mérés befejeztét. Ez kb. 11/55/210 ms ideig tart egy 8/12/14 bites mérés esetén. A mérés befejeztét a DATA vezeték alacsony szintbe kerülése jelzi, és ilyenkor a hőmérő készenléti üzemmódba kapcsol. A mikrovezérlőnek ezt meg kell várnia, mielőtt kiolvasná az adatokat. A mért adat annak kiolvasásáig tárolódik.

A hőmérő 3 bájtot küld: 2 bájton ábrázolja a mért adatot, a 3. bájt pedig ellenőrzőösszeg. A mikrovezérlőnek minden bájtot nyugtáznia kell a DATA vezeték lehúzásával. Minden értékben a legmagasabb helyiértékű bit található az első helyen, és az értékek jobbra vannak igazítva. A válasz kiolvasása után az eszköz automatikusan visszatér alvás üzemmódba, és a kommunikáció véget ér. [9]

Eredmények

A 85°C-t 0,1°C-os pontossággal sikerült tartani. Szobahőmérsékleten a kemence kb. fél óra alatt érte el ezt a hőmérsékletet. A relatív páratartalom a kemencébe helyezett vízzel teli edény és a beépített ventillátornak köszönhetően 91%-ra állt be.

18. ábra: kemence mikrovezérlővel



19. ábra: ventillátor a kemencében (bal oldali kép), a mikrovezérlő (jobb oldali kép)

Összefoglalás

A dolgozatban az ATMEL AVR mikrovezérlők alkalmazási lehetőségeit mutattam be megvalósult mérő- és szabályzórendszereken keresztül. Az ATMEL Atmega128-as mikrovezérlőjével dolgoztam és három rendszert, egy felfedezőrobotot, egy klímakamrát és egy mérőállomást készítettem el.

A felfedezőrobot saját építésű és önállóan közlekedik. Érzékeli és kikerüli az akadályokat. A villanymotorokat adott frekvenciájú impulzusszélesség modulált jelek vezérlik. A közelebbi akadályokat két mechanikus, a távolikat pedig egy infravörös érzékelő észleli. A robot igyekszik kikerülni az akadályokat. Ha mégis ütközik valamivel, kicsit visszatolat, majd azonnal folytatja útját. Az infravörös érzékelő illesztéséhez analóg-digitális átalakítás is szükséges volt.

A szélsebességmérő és adatgyűjtő berendezés pontos mérésre, illetve hosszabb idejű önálló adatgyűjtésre készült. Alapvetően soros interfészen kontrollálhatjuk, de kommunikálhatunk vele rádión keresztül és mobiltelefon segítségével is. A szélmérést az anemométertől kapott jel frekvenciájának mérésével valósítja meg. A frekvenciát a processzor egy 16-bites számlálójával méri, amely alkalmas arra, hogy külső esemény hatására megszakítást váltson ki. Az algoritmus kezeli a számláló túlcsordulásának problémáját, és kiszűri a mérési hibákat. A mikrovezérlőhöz I^2C-buszon illesztettem egy hardveres valós idejű órát, valamint SPI buszon egy memóriakártyát (MMC vagy SD). A mérőállomást bemutató dolgozat a 2006-os őszi helyi informatikai TDK-n első helyezést ért el.

A klímakamra hőmérséklet-szabályzást és páratartalom-mérést valósít meg. PID-szabályzást alkalmaz, és LCD-kijelzővel rendelkezik, amely folyamatosan mutatja a kamra hőmérsékletét és páratartalmát. A szenzor kétvezetékes interfészen (TWI) csatlakozik a mikrovezérlőhöz, amely azonban nem kompatibilis az I^2C-vel, így egy speciális protokollt kellett implementálni.

A fenti projektek megvalósítása során szerzett ismeretek és tapasztalatok döntően hozzájárultak ahhoz, hogy a villamosmérnökök számára meghirdetett Számítógép architektúrák című tárgy keretében megindult az ATMEL AVR-ek C-programozásának oktatása.

Dolgozatomat néhány tanulsággal szeretném lezárni: Az AVR mikrovezérlőkhöz többféle interfészen keresztül, egyszerűen illeszthetők mérőeszközök. A működés folyamatos ellenőrzéséhez bizonyos esetekben LCD-kijelzőt érdemes alkalmazni, máskor pedig egy soros interfészen csatlakoztatott PC jelenti a megoldást. Az időzítő/számláló perifériák egyaránt alkalmasak pontos mérésre és pontos vezérlésre. A soros kommunikáció könnyen működésbe hozható, és tökéletesen alkalmas a program tesztelésére, illetve a hibakeresésre. Ezért célszerű minden projektet a soros kommunikáció életre keltésével kezdeni.

Problémát jelent azonban, hogy mivel bizonyos lábak több funkciót is kiszolgálnak, bizonyos eszközöket nem lehet (illetve nehéz) egyszerre illeszteni a mikrovezérlőhöz, pl. az LCD-kijelzőt nem használhatjuk egyszerre az MMC-kártyával.

Köszönetnyilvánítás

Köszönetet mondok szüleimnek, akik békés családi hátteret és anyagi támogatást nyújtottak tanulmányaimhoz.

Köszönetet mondok témavezetőmnek, dr. Szabó Istvánnak, Harasztosi Lajos mérnöktanárnak, és Virág Tamásnak, akik támogatása, ösztönzése és tanácsai nélkül ez a dolgozat nem valósult volna meg.

Köszönetet mondok a Szilárdtest Fizika Tanszéknek, amiért a projektek költségeit állta, valamint nyugodt munkát tett lehetővé oldott hangulatú légkörben.

Köszönetet mondok az Alter Energetikai Irodának, amiért rendelkezésünkre bocsátott egy anemométert.

Végül, de nem utolsó sorban köszönetet mondok Istennek, aki mindig úgy alakítja a körülményeket, hogy álmaimat megvalósíthassam.

Felhasznált irodalom

1. Szélenergia az Alföldön: http://www.haon.hu/engine.aspx/page/haon-article-detail-page/cn/haon-news-je01-20061112-014028/dc/im:haon:news-hajdubihar/ag/im-naplo
2. A szélenergia, mint a megújuló energiaforrások hasznosításának legdinamikusabb formája: http://www.kvvm.hu/cimg/documents/Groo_Rudolf_2.pdf
3. Szélkerékcentrum: http://www.szelkerekcentrum.hu/
4. Atmel AVR: http://en.wikipedia.org/wiki/Atmel_AVR
5. Danish wind industry association: „Wind enery manual”: http://www.windpower.org/en/stat/units.htm
6. Beágyazott rendszerek kommunikációja: http://www.szabilinux.hu/konya/konyv/2fejezet/2fbereko.htm
7. Joe Pardue: C Programming for Microcontrollers
8. Atmega128 adatlap: http://www.atmel.com/dyn/resources/prod_documents/2467S.pdf
9. SHT11 adatlap: http://www.sensirion.com/en/pdf/product_information/Data_Sheet_humidity_sensor_SHT1x_SHT7x_E.pdf
10. Philips Semiconductors: Az I^2C busz és használata: http://www.muszeroldal.hu/measurenotes/i2c_hu.pdf
11. C+D Automatika: PID hangolás: http://meter.hu/cikkek/pid_hangolas
12. Wikipédia: Analóg-digitális átalakító: http://hu.wikipedia.org/wiki/Anal%C3%B3g-digit%C3%A1lis_%C3%A1talak%C3%ADt%C3%B3
13. Sharp GP2D12 infravörös szenzor: http://www.acroname.com/robotics/parts/SharpGP2D12-15.pdf