Af: OZ7EC, Erik Christensen

Introduktion til en nem mikroprocessor

Den tekniske udvikling har givet os radioamatører fantastiske muligheder for at berige vores traditionelle radiomæssige områder med spændende mikroprocessor projekter. Her beskriver jeg i løse vendinger en målemetode, som udnytter mikroprocessorens stærke sider. Målingen er en simpel voltmåling, som jeg har valgt for ikke at komplicere blandingen af mikroprocessor og programmering.

Beskrivelsen er i skitseform og indeholder ikke montering i en nydelig kasse, det kan de fleste sikkert klare meget bedre på egen hånd. Løsningen skal tjene som inspiration og vil i senere numre af OZ blive fulgt op af mere komplicerede projekter og programtekniske beretninger.

På markedet findes mange forskellige mikroprocessorsystemer, nogle meget komplicerede at arbejde med, men nogle få er så intuitivt ligetil, at programmeringsarbejdet går som en leg. Jeg har derfor valgt Arduino systemet ud fra mange års erfaringer med programmering tæt på selve processoren.

Arduino er et miljø til eksperimenter, men kan ligeså vel opfattes som udviklingsplatform for integration af mikroprocessor i vore konstruktioner.

Systemet består af et softwaremiljø som downloades gratis, et forud defineret printkort, som indkøbes til en ringe pris, en bootloader, og i de fleste tilfælde også et display.

Programmeringen foregår på din PC og når du har skrevet dit program beder du dit IDE (Integrated Development Environment) om "Upload".

Programmet bliver så kompileret, forhåbentlig uden fejl, og så bliver det ved hjælp af bootloaderen leveret ned i printkortets processors flash programlager. Og så kører det!

Arduino er "Open Source", så al software, alle printkorttegninger og layouter er tilgængelig. Det betyder at vi stjæler med arme og ben fra hinanden, at vi står på hinandens skuldre og derved når nye højder. Skal du programmere noget nyt, så søger du først om der allerede er en anden der har programmeret dette emne.

Netop det, at du programmerer på din PC, kompilerer med et museklik og uploader med det samme gør, at du kan koncentrere dig om programmets funktioner og ikke skal forstyrres af komplicerede oversættelser og prombrændinger.

Hardware

Som en begyndelse foreslår jeg, at du bruger Arduino Uno R3 (Revision 3). Det er et nyudviklet print med en mikrocontroller af AVR familien fra Atmel, controlleren hedder ATmega 328p. Den er forprogrammeret med en bootloader og køreklar, når du køber den.

Foto 1. Gulvbrædt er min Arduino Uno R3 til programudvikling. Den er derfor forsynet med adskillige tilføjelser:

1. Display øverst
2. DIN sokkel med A/D input og forsyning fra +5 V stabilisatoren.
3. 5 volt stabilisator som fødes fra Vin på Arduino til backlight
4. Potentiometer med formodstand til + 5 V. Undersøgelse af A/D opløsning.
5. Potentiometer til kontrastregulering.
6. Voltmeter tilslutning

Kortet har mange faciliteter, først og fremmest en USB tilslutning til din PC. Du får også brug for et display, som optimalt vil være 2 linjer med 16 karakterer. Husk at det skal være 5 V, ikke 3 V display.

Processoren kan køre på power fra din USB, men da mine gamle øjne ynder klare tekster, tilråder jeg at du laver backlight på displayet.

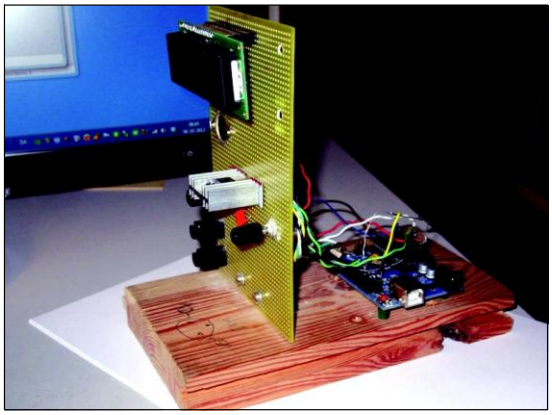

Foto 2. Gulvbrædt set fra siden. Det er spaghetti trådning så A/D får 2,5 V reference i stedet for den indbyggede 5 V. Bemærk de to tilslutninger til USB og Ekstern power.

Til det får du brug for en af de mange sorte forsyninger du har liggende nede i skuffen. Den sorte forsyning skal være med + i hullet og minus yderst, skal optimalt levere 6-7 volt, men må være helt oppe på 12 volt. Backlight kræver 5 volt 150 mA, og det kan du ikke belaste Uno'ens stabilisator med, så en ekstern 5 volt regulator må du også ned i skuffen og finde. Når du tilslutter den sorte spændingsforsyning på Uno R3, kobler den selv over på denne forsyning i stedet for USB forsyningen. Det er for resten også denne måde den strømforsynes, når den er færdigprogrammeret og skal opnå skilsmisse fra din PC.

Klare tekster

Arduino Uno har en masse faciliteter udover selv processoren; 5 volt og 3,3 volt stabilisatorer i form af SMD kredse, som ikke kan levere meget, lysdioder samt ikke mindst I/O porte fra processoren.

Foto 3. Her har vi Arduino Uno R3 forsynet med et "Shield", som er trådet således at et display umiddelbart kan stikkes ind i det, samt displayet. Alt er "Piggy Back" og meget kompakt.

I/O er fordelt således: 14 digitale, hvoraf 6 kan bruges som PWM (Pulsbreddemodulation). Hver pin kan håndtere 40 mA og i programmet bestemmer du, om den skal være input eller output. Desuden findes 6 analoge indgange som kan håndtere 0 til 5 volt og digitaliseres med en 10 bits A/D konverter med 1024 trin. Udover disse almindelige ind- og udgange findes der en række specielle faciliteter, som jeg vil overlade til dig selv at udforske. Hent beskrivelserne hos www.arduino.cc

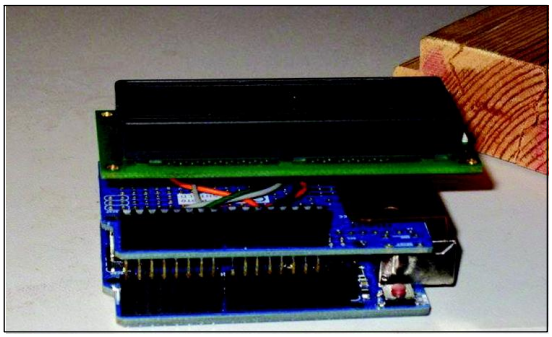

Foto 4. En stablet Arduino. Nedefra ses Arduino Uno R3, dernæst "Shield" som indeholder interface og regulator til backlight. Øverst ses displayet.

I de næste numre af OZ kommer der beskrivelser af processorens specialfaciliteter og deres benyttelse i programmer. Hvis du vil have dit display direkte ovenpå dit Arduino Uno, kan du med fordel anvende et "Arduino Shield" mellem UNO og dit display til at trække de nødvendige ledningsforbindelser. Samtidig kan du jo placere 5 volts stabilisatoren på dette shield. Dens køleplade kan netop anes på stakke billedet.

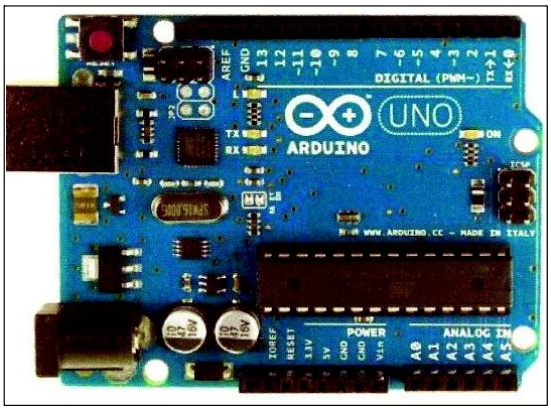

Foto 5. Sådan ser selve hjertet ud.

Bootloaderen

Det som absolut gør Arduino projektet kvikt at

arbejde med er bootloaderen. Den er flashet i processorens programlager i det øverste 512 bytes af dens 32 kbyte flashhukommelse.
Med hjælp fra en Atmel lillebror, en ATmega16U2, som håndterer USB forbindelsen, fylder du utroligt hurtigt dit kompilerede program op i 328'erens flashhukommelse og eksekverer straks programmet.
Du får feedback på dit program umiddelbart og der er kontant afregning. Nu skal du blot have udryddet dine logiske fejl.

Programsproget
Arduino har helt sit eget sprog, selvom det har aner fra sproget C og C++.
Et fornuftigt program til en Arduino består af 4 dele:
1. Definitioner,
2. Initialisering.
3. Loop (Her gøres arbejdet, det hedder MAIN i C++) og endelig
4: Functions.

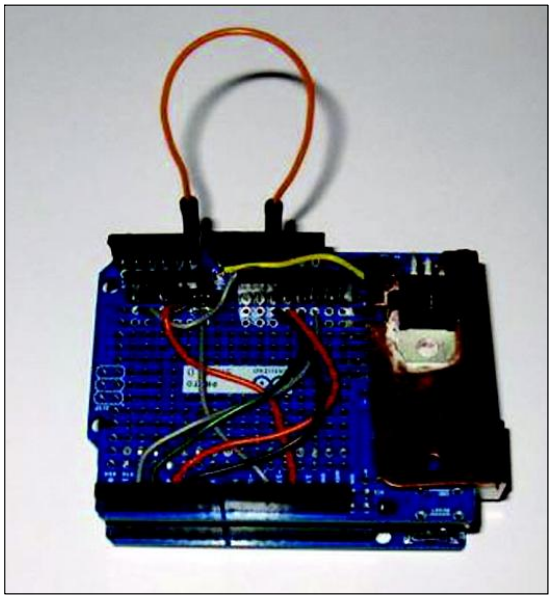

Foto 6. "Shield" med ledninger, trådet per håndkraft og vi ser backlight stabilisatoren.

Definitioner indeholder også de biblioteker du i den aktuelle situation skal bruge udover Arduino standard sprog. I miljøet findes et hav af biblioteker, så skal du bruge noget udover standard sproget: Søg og du skal finde.
Der er garanteret en der har skrevet et bibliotek du kan bruge. I nødstilfælde: Skriv dit eget tilføjelses bibliotek. Et godt eksempel: Jeg skulle bruge titalslogaritmer til dBm beregninger. Løsningen var simpel: inkluder biblioteket "math.h".

Så programmerer vi
I det følgende forudsætter jeg, at du har downloadet filen "maal3volt.ino" fra hjemmesiden www.oz1edr.dk, søg under Arduino.
En .INO fil er en Arduino sketch, som indeholder kildeteksten. Filen kan kun læses med en Arduino IDE.
Herfra forudsætter jeg, at du har filens kildetekst liggende foran dig.

I begyndelsen af mine kildetekster står der altid en hel masse tekniske forklaringer. Der er praktisk at det står her i kildeteksten i stedet for at du og jeg skal ud for at lede efter en tekstfil et eller andet sted.

Programblok 1.
Her kommer "Definitioner" således;
#include <LiquidCrystal.h>
Her fortæller jeg kompileren, at den i sin kompilering skal medtage dette bibliotek.
Er du nysgerrig: Find og hent biblioteksbeskrivelsen!

#define karaktererperlinie 16
#define antallinier 2
Her fortæller jeg kompileren om to navne, karaktererperlinie og antallinier, som skal have en værdi tilskrevet. Det betyder at jeg fremover blot kan skrive navnet, så erstatter kompileren navnet med talværdien. Umiddelbart ser det ud som at jeg skyder kanoner med gråspurve, MEN det bidrager ofte til et gøre et program letlæseligt og menneskevenligt!

typedef unsigned long int langinteger
Her definerer jeg en type data med navn langinteger, et heltal med ekstra længde udover de normale 8 bits længde.
Typedefinitioner er meget anvendelige, specielt hvis man har mange variable af en speciel type. Her har jeg medtaget den for at vise lidt af programmeringssprogets muligheder.

Langinteger raaaddata = 0;
Her definerer jeg en variabel ved navn raaaddata og giver den samtidig startværdien 0. Rå A/D data er det bitmønster jeg får, når jeg læser de binære data som min 10 bits ADC giver mig. Det er en successiv approksimations konverter. Værdien kan være fra 0 til 1024 decimal, men udtrykt på binær form.

float maaltvolt;
Float er en floating point (et tal med komma) datatype, som kan antage værdier fra 3,4028235E+38 til
-3,4027235E+38. Den optager 4 byte i RAM

Int adinputpin = A0;
Her definerer jeg et integer (heltal mellem 0 og 255) og giver det værdien A0. Det er den analoge Arduino inputpin vi skal have vores A/D konverter til at læse fra. Dermed er det altså en instruktion til mikroprocessorens multiplexer om at vælge A0 som input til A/D konverteren.

LiquidCrystal lcd(12,11,5,4,3,2);
Den er "Case Sensitive", det betyder at store bogstaver SKAL være store bogstaver og små bogstaver SKAL være små bogstaver. Det er en instruktion til biblioteket om hvilke funktioner vi har bundet hvilke forbindelser til på displayet.

Det var slut på definitioner, nu starter vi på SET-UP

Programblok 2:

```
Void  setup()
{
      lcd.begin      (karaktererperlinie,antallinier);
```

Fortæller kompileren hvilken type display vi bruger

```
      lcd.setCursor (0,0);
```

Sæt cursoren på displayet på øverste line, første position
Fordi: Når jeg skriver på displayet begynder det at vise tegnene fra den position hvor cursoren står.

```
      lcd.print ("Maaler min 3,3 volt");
```

Flytter teksten fra programmet op på displayet.

```
      lcd.setCursor (0,1);
```

Cursor på nederste linie, første position

```
      lcd.print ("               ");
```

Skriv en tom linie forneden.

```
Delay(1500)

Kaffepause?
}
```

Bemærk at program linierne er indeklemt imellem en "Opening Brace" og en "Closing Brace". (ALT + 123 og ALT + 125)

"void" betyder bare tom og navnet setup er fulgt af en tom parentes.
Det betyder, at der ikke overføres lokale data til setup og der ikke returneres lokale data fra setup.
Det var så slut på Setup.

Programblok 3:
Her residerer orkesterets dirigent, som bestemmer hvilke orkestermedlemmer der skal arbejde og i hvilken rækkefølge.
Her bliver vi kørende i loopen, indtil der ikke er mere strøm på.

```
Void loop ()
{
      tomnederstelinie();
```

Vi starter med at rense den nederste linie for gamle data

```
      hentaddata();
```

Hent bitmønster

```
      omregntilvolt();
```

Gør som skrevet står!

```
      Skrivnederstelinie();
```

Som skrevet står igen.

```
      delay(2000);
```

Pause uden pauseklovn.

```
}
```

Programblok 4:
Her bor slaverne eller arbejder kulierne. De benævnes i daglig tale "Functions".
De er egnet til genbrug fra flere forskellige steder i et program.

```
void  tomnederstelinie()
{
      lcd.setCursor (0,1);
```

Nederste linie, første position

```
      lcd.print ("               "):
```

En hel tom linie!

```
}

void  hentaddata()
{
raaaddata = analogread (adinputpin);
```

Hent det binære mønster fra A/D.

```
}

void  omregntilvolt();
{
      maaltvolt = ((raaadadat * 5.0 ) / 1024.0000):
```

Man tager bitmønsteret fra A/D og ganger med 5, A/D'ens referencespænding, for derefter at dividere med opløsningen = 1024. Resultatet er i V. Man kunne også sige:
5 volt divideret med opløsningen 1024 = 4,88 mV
Gang de 4,88 mV med det binære tal fra A/D og du får V ud af det.

```
}
```

Huskede du at forbinde A0 pinden med 3,3V fra stabilisatoren?
I de næste numre af OZ kommer der mere stof om mikroprocessoren.

Og hvordan kommer du så selv i gang med at programmere?
Det vil jeg vise dig med et eksempel. Du skal på dit display vise tal fra en HF probe med følgende

data:
Frekvensområde: 2 kHz - 200 MHz
Nøjagtighed:
2 kHz - 10 MHz +/- 1 % og +/- 50 mV
10 MHz - 100 MHz +/- 1 dB
100 MHz - 200 MHz +/- 6 dB.
Maksimal udgangsspænding 25 Vrms

Overspændingsbeskyttelse:
141 Vp på indgangen med en 100 volts zenerdiode.

Overvejelser:
1. Dine analog indgange er CMOS, d.v.s. De tåler ikke mere end 5 V.
2. Du kan ikke se frekvensen udfra den DC spænding du får fra proben.
3. Du skal levere en form for kalibrering, hvad viser du på displayet?
4. Vil du vise Vrms eller Vp eller Vpp?
5. Vil du vise dBm? (Forudsat du måler over 50 ohm = din kunstantenne)
6. Vil du korrigere for diodekrumningen ved lave spændinger?
De korte svar er:
Ad .6: Nej
Ad. 5: Ja, det koster jo kun lidt processorkraft = næsten gratis
Ad. 4: HF prober anvendes sjældent på kunstantennen, Vpp er nummer et for kredsløbsdesignere.
Ad. 3: Vi skal som option kunne kalibrere, vi har EEPROM i processoren til det formål.
Ad. 2: Det er ikke en tæller vi skal lave.
Ad. 1: Spændingsdeler og valg af flere analoge indgange lyder som sagen.
Zenerbeskytttelse af indgangene?

Så starter vi med det som kaldes en pseudokode for at fastlægge trinnene:

Hardware:
Lav en spændingsdeler, hvor hvert trin er koblet til en af de 6 analoge indgange.

Dit program i pseudokode:
Undersøg hvilken indgang der har en fornuftig værdi. (Multiplexer og ADC)
Beregn udfra indgangens nummer og A/D konvertering hvor mange volt, der er på proben.
Beregn Vpp
Beregn dBm
Vis dem på display.
Forfra igen.

Det er programmets indhold - uden kalibreringen.
Mens du er ved programudviklingen kan du jo passende kalibrere således:
Påtryk proben f.eks. 50 mVpp.
Korriger din udlæste værdi, så displayet viser 50 mVpp.
Så der det overstået.
Selvfølgelig kan du jo også kalibrere ved en anden spænding du måtte have til din rådighed.
Senere kalibrering:
Overvej hvordan det kunne lade sig gøre UDEN at du bruger programmodifikation.

Så kommer de 4 programblokke:
1. Definitioner:
Biblioteker
Variable:
Hvilken indgang
En fornuftigspænding hvad er det?
Den rigtige spænding

2. Setup
Init display
Init variable, dem kender du først når du er i gang med at skrive programmet.

3. Loop:
Undersøg hvilken indgang der har en fornuftig værdi. (Multiplexer og A/D)
Beregn udfra indgangens nummer og A/D konvertering hvor mange V der er på proben.
Omregn til Voltpp
Beregn dBm
Vis dem på display.
Forfra igen.
(Ovennævnte er tyvstjålet fra din pseudokode!)

4. Kulierne (Functions)
De kommer - næsten - af sig selv når du skriver programmet.

Gode links:
www.arduino.cc
www.earthshineelectronics.com
Vælg: "The Complete Beginner's Guide to the Arduino"
www.adafruit.com
www.esr.se I arkivet, vælg Resonans nr. 2, 2012, læs side 29 om proben.