Styrketräningshjälp

Projektrapport

Kurs:	Elektronikdesign
Titel:	Styrketräningshjälp
Grupp:	G2
Gruppdeltagare:	Rita Kasbo
	Robert Wahlström
	Mahshid Mehrparvar
	Percy Villegas Tello
	Roua Kassir
	Samin Mahfuz
Datum:	2008-10-21
Handledare:	Stefan H Karnebäck
Examinator:	Stefan H Karnebäck

Sammanfattning

Denna rapport behandlar byggandet av en EMG maskin som skall vara konstruerad på ett sådant sätt att den kan användas i tränings syfte istället för diagnostiskt. Maskinens delar består av följande: En förstärkar-/filterkoppling som skall ta in och förstärka en signal, en PIC krets som skall AD-omvandla, sampla och sända signalen vidare, en bluetoothsändarmodul vars enda uppgift är att slussa datan från PICen vidare ut i luften.
På datorsidan mottas sedan informationen och presenteras på skärmen.
Samtliga delar av maskinen som byggts under detta projekt fungerar förutom förstärkar-/filterkopplingen som vi inte lyckats få att fungera fullt ut.

Summary

This report deals with the construction of an EMG machine to be designed in such a way that it can be used for training purposes rather than diagnostic. Machine parts consist of the following: An amplifier/filter coupling to detect and amplify the signal, a PIC circuit to AD convert, joint and send the signal further, a Bluetooth module whose sole task is to relay data from the PIC into the air.
Afterwards the PC side receives information and present it on the screen.
All parts of the machine built under the course of this project is fully working except the amplifier-/filter coupling that we failed to get fully working.

Förord

Detta projekt har drivits inom kursen Elektronikdesign tredje året på Medicinteknik KTH och har varit en av de mer lärorika under vår utbildning då vi här mer har fått utnyttja ett större spann av tidigare kunskaper. Projektet har tagit en hel del tid men i slutändan så är den tiden troligtvis värd den tid vi lagt ner på det här projektet.
Vi vill därför tacka Stefan för en bra handledning och en bra kurs.
Vi vill även tacka Yvonne Borin för ytelektroderna från Huddinge sjukhus samt även Leif Pettersson – EMG läkare, vid Huddinge sjukhus som ställde upp på frågor samt Dennis Sturm som gett oss en del av delarna till projektet.

Projektgruppen G2
KTH Teknik och Hälsa, Campus Telge
2008-09-15

Innehållsförteckning

Inledning ... 1
- Bakgrund ... 1
- Problemdefinition ... 1
- Kravspecifikation ... 1
  - Projektets kravspecifikation ... 1
  - Produktens kravspecifikation ... 1
- Mål ... 2
- Lösningsmetod ... 2
- Avgränsningar ... 2

Genomförande ... 3
- EMG ... 3
- Förstärkare & Filter ... 4
  - Den nuvarande kopplingen ... 4
  - Instrumentförstärkaren ... 5
  - Bandpassfilter ... 5
  - Halvvågslikriktare ... 6
  - Andra försök med förstärkarkopplingar ... 6
  - Information gällande övriga försök ... 7
- PICens programmering ... 9
  - AD omvandlingen ... 9
  - In & ut portar på PICen ... 10
  - Seriell överföring mellan PICen och Bluetooth ... 10
  - Resultatet av PIC programmeringen ... 11
- EMG ritarna ... 12
  - Det färdiga C-programmet: ... 13
  - Excelprogram lösningen: ... 14
  - Vad lösningarna saknar: ... 14
  - För att lösa uppgiften med grafikritarna: ... 14
- Bluetooth mottagning med IAR systems ... 15
- C-program för att läsa in EMG värdena från Comport i Dev C++ ... 15

Slutsats ... 17

Appendix ... 18
- Del 1 – Flödesschema ... 18
- Del 3 – Grafikritar programmet i C ... 23
- Del 4 – Visual Basic koden ... 26
- Del 5 – Kretsschema ... 28
- Del 6 – Manualen (skriven som om vi hade alla delar fungerande) ... 29
  - Manual till MuscleArray 1000 ... 29
- Del 7 – MusclePower Omvandlaren (lådan) ... 34

Referenser ... 35

Inledning

Bakgrund

Elektromyografi (EMG) är en idag en väl etablerad teknik inom sjukvården och även inom många forskningsområden. EMG är en av kärnorna inom Medicinsk teknik. De flesta yrken idag rör sig mot en allt mer stillasittande arbetsform. Trenden går mot att allt fler börjar intressera sig för fysisk aktivitet. Ett hjälpmedel som effektiviserar träningen är efterlängtad. Gruppens uppdrag är att ta fram en styrketräningshjälpredskap som bygger på EMG teknik. Projektet kommer att testa ett flertal moment av vår utbildning och kommer därför att bli en bra träning inför det riktiga yrkeslivet.

Problemdefinition

Dagens stressiga samhälle har gjort det svårt att lägga tid på träning. Den tid man kan lägga på träningen måste därför bli effektivare.

Kravspecifikation

Projektets kravspecifikation

- Max budget: Ett tusen kronor.
- Hålla tidsplanen.

Produktens kravspecifikation

- Ta in en läsbar signal från en muskel
- Förstärkarlösning som klarar
    - 0 – 5 V förstärkt signal till Pic
    - Renad & läsbar signal
    - Genomsläpsspärr på 10 – 200 Hz
    - Filtrera bort 50 Hz
- Kravet på PIC 18F2550 som vi har använt oss av är att
    - Klara AD omvandling
    - Kunna Sampla inkommande signal i åtminstone 1000 Hz
      Detta för att få noggrannhet på 5 – 10 samplingars per period i högsta frekvenskomponenten.
    - Klara en seriell överföring av data i mottagande komponents hastighet men stabil sändning som går att motta i datorändan
- Portabel strömkälla med +5 V,-5 V samt jord till PIC samt förstärkar-/filterkoppling
- Använda oss av Bluetooth om det är möjligt
- Vi ska försöka komma så nära realtid som möjligt under datapresentation.
- Kretskortslösning
- Proengineeringlösning av produkthölje

Mål

Målet med projektet var att tillverka en EMG apparat som registrerar muskelaktiviteten. Detta gör vi genom att tillämpa och fördjupa våra kunskaper som vi har fått lära oss under utbildnings gång.

Lösningsmetod

Projektets omfattning krävde att vi tillämpade en stor del av det vi redan har lärt oss samt även samlade in och lärde oss en stor mängd kunskap inom ett kort tidsintervall.
Vi kom därför fram till att olika delar av projektet behövdes delas upp inom gruppens medlemmar. Detta gjordes fortlöpande under projektets gång då nya behov uppstod.
Vi delade upp projektet inom följande ansvarsområden för att kunna genomföra projektet i tid.

- Filter och förstärkare
- PIC programmering (MPLAB IDE v8.00)
- Bluetooth sändning (Bluetoothmodul; privattillverkad)
- Bluetooth mottagning (BlueSoleil, Hyperterminalen)
- Presentation av mottagen data på skärm (C-programmering, Excel-lösning tillsammans med Visual Basic)
- Kretskort lösning (Labbplatta, lödning)

Avgränsningar

- Det behöver inte bli vackert.
- Realtid kan uppfyllas i den mån det går.
- Vi kommer bara lära oss de register/komponenter/program som behövs för projektets slutgiltiga lösning.

Genomförande

EMG

Elektromyografi (EMG) är en teknik som används inom medicinen för att mäta de elektriska signaler som alstras när en muskel kontraheras. EMG-signalen består av en serie elektriska impulser. En elektrisk impuls, en aktionspotential, alstras varje gång en motorisk enhet aktiveras och har sin grund i den momentana spänningsskillnaden mellan muskelfiberns insida och utsida. När en muskel vilar är den elektriskt ”tyst” och när den kontraheras alstras elektriska signaler som ökar i samband med att muskelkontraktionens styrka ökar.
Aktionspotentialerna fungerar enligt ”allt eller inget” principen. Antingen blir det en impuls med full styrka eller så sker det inget alls. Muskelkontraktionens styrka regleras istället med antalet aktionspotentialer per tidsenhet och antalet aktiverade motoriska enheter, därav skillnaden i EMG-signalens styrka vid olika kontraktionskraft.
Det finns två olika sorters elektroder, nålelektroder (intramuskulära) och ytelektroder (som vi valt att använda). Intramuskulära elektroder används för djupt liggande muskler och fångar ungefär 100 motoriska enheter medan ytelektroderna används för ytliga muskler och kan fånga signaler upp till flera tusen motoriska enheter. Inom EMG är målet egentligen att fånga så få motoriska enheter som möjligt för att få större noggrannhet.
Med nålelektroder inne i muskeln kan uppmätta spänningar uppgå till flera tiotal mV medan signalerna från huden kan bli upp till 1 mV (medelvärde) vid maximal kontraktion.

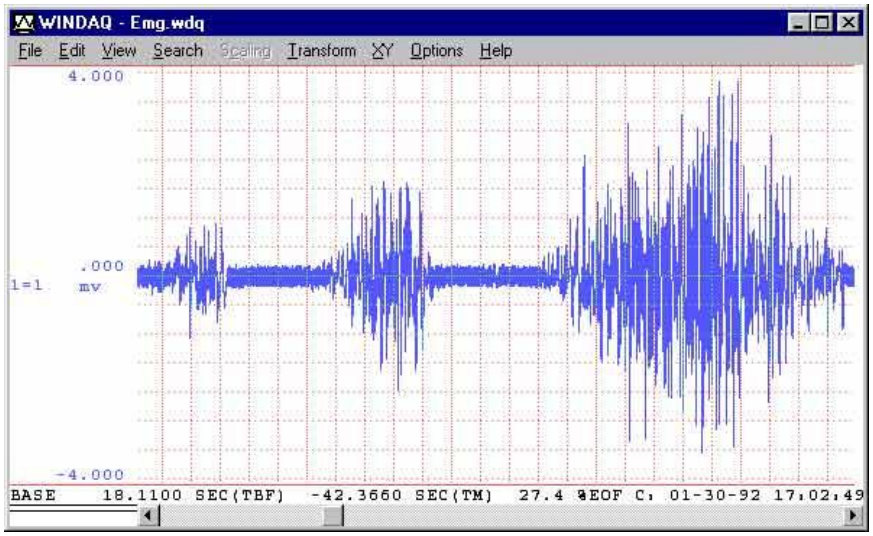


Bild1: Aktionspotential vid mätning av EMG med ytelektroder
Ref: [1]

Förstärkare & Filter

När man vill mäta mängd muskel aktivitet så är det viktigt att man förstärker och rensar ”störningar och oönskade frekvenser” innan den ska skickas vidare till ”PIC 18F2550” kretsen som vi använder oss av. Som tidigare sagts har vi använt oss av ytelektroder. För att få signalen rensad från commonmode störningar så har vi använt oss av två elektroder, en elektrod som är kopplad till plus på förstärkaren placeras på agonisten alltså den muskel som jobbar under träningen. Den andra elektroden som är kopplad till minus på förstärkaren placeras på muskelfästet till agonisten. Ytterligare en jord eller referens kan kopplas till en annan muskel på kroppen, detta för att öka commonmode dämpningen. Detta är dock inget vi i nuläget har använt oss av.

Den nuvarande kopplingen

För att kunna få en tillräckligt ren signal (för att vi ska kunna tolka den) och en tillräckligt hög förstärkning (d.v.s. så PICen kan tolka signalen), så använde vi oss av en instrumentförstärkare, ett bandpassfilter och en halvvågslikriktare i den ordningen.
Vi testade även en del andra lösningar på förstärkar-/filterkopplingar men slutsatsen var att denna koppling vara den bästa.
Kopplingen utnyttjade instrumentförstärkaren av typen INA128PA som är en differentialförstärkare med hög CMRR, låg matingsspänning, låg offsetspänning och låg brusnivå. Igenom denna tog vi först in signalen och filtrerade bort all commonmode spänning, detta för att vi visste att det var de signaler som gick in lika på plus och minus ingången på förstärkaren vid EMG som stod för en stor del av störningarna (d.v.s. vi försökte ändå hitta en motorisk enhet bland tusentals).
Vi skickar sedan signalen därifrån vidare till ett bandpassfilter med ett intervall på omkring 10 – 200 Hz. Detta för att endast släppa igenom de signaler vi är intresserade av.
D.v.s. de som troligtvis ligger runt 60 – 150 Hz. Vi gjorde den något bredare då vi inte kunde vara helt säkra på att vi inte skulle få värden högre eller lägre än detta. Efter bandpassfiltret så halvvågslikriktar vi signalen. Syftet med detta var att slå ut all minus spänning innan den fördes in i Picen (PICens AD-omvandlare går sönder om den får spänning som ligger för långt över 5V eller för långt under 0V).
Beskrivning av ingående delar står listad nedanför.
Ref: [Appendix-Del 4, Robert, Percy & Rita]

Instrumentförstärkaren

Instrumentförstärkare används för att undertrycka lika inspänningar (common mode) och för att förstärka differentiella spänningar. Hur mycket man förstärker differential spänning kan man få med hjälp av formel nedan.

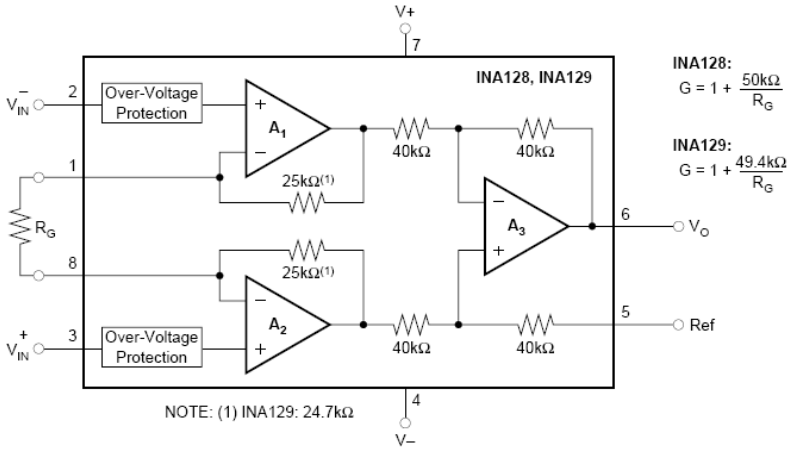


Bild 2: INA128PA operationsförstärkar
Ref: [2]

Genom att ändra på R_g så kan man reglera förstärkningen. Den simulering som gjordes i Orcad överensstämde inte helt med verkligheten. Simulering använder sig av 3 uA741 förstärkare för att simulera insidan av INA128PA samt att den inte tar hänsyn till samtliga faktorer kan vara orsaken till differensen mellan simulering och verklighet.

Ref: [2, Appendix – Del 4]

Bandpassfilter

Filter i allmänhet används för att ta bort oönskade frekvenser. Frekvenserna som vi är intresserade av ligger inom ett visst frekvensintervall och därför är samtliga av de övriga frekvenserna ointressanta. De frekvenser som är aktuella för mätningen när det gäller EMG ligger mellan 10Hz - 200Hz, för att åstadkomma detta intervall använde vi oss av ett bandpassfilter. Bandpassfiltret är ett seriekopplat högpassfilter och ett lågpassfilter (Se Appendix – Del 4). Detta filter togs fram genom Orcad simulering, varefter det byggdes. Bandpassfiltret dämpade signalen något men för övrigt så fick vi det frekvens intervall vi vill mäta på.
För att bestämma den övre och undre brytfrekvenser används formeln nedan.

$$f_c = \frac{1}{2 * \pi * R_1 * C_1}$$

Där fc är den övre och undre brytfrekvensen.
Ref: [14, Appendix-Del 4]

Halvvågslikriktare

Halvvågslikriktaren är en enkel koppling bestående av en diod samt en resistor.
Syftet med enkelheten i kopplingen var att undvika distorsion på ursprungssignalen då antalet komponenter i en koppling över lag brukar vara direkt proportionell mot störning på signalen. Halvvågslikriktarens funktion är även att den endast släpper igenom ena halvperioden av sinusspänningen.
Ref: [14, Appendix-Del4]

Andra försök med förstärkarkopplingar

- **Försök 1 – Den otestade förstärkaren:** Innan vi beslutade att bygga en egen förstärkare så var det tänkt att vi skulle få en färdigbyggd förstärkare som hade konstruerats på ett examen arbete. Denna förstärkare var dock inte testad. Medan vi testade den så visade det sig att den inte funkade helt Förstärkaren utnyttjar även den en INA 128, denna i sin tur är kopplad till ett notchfilter. Ett notchfilter är ett väldigt smalt bandspärrfilter vars syfte är att ta bort nätspänningsfrekvensen (i Sverige 50 Hz).
  För att testa förstärkaren kopplades Bodyref punkten samt en av ingångarna till jord. På andra ingången testades olika spänningar med en spänningsregulator. Enligt examens arbete så skall kretsen förstärka mellan 115ggr till 168ggr. Förstärkaren testades därefter tillsammans med Dennis Sturm. Detta visade att det man fick var en kapacitiv urladdning istället för förstärkning. Men p.g.a. tidsbrist så lämnade vi förstärkaren på lagning och tillverkade en egen EMG förstärkare.

- **Försök 2:** Vi testade en koppling med instrumentförstärkaren INA128PA kopplat till vårt nuvarande bandpassfilter, därefter förde vi signalen vidare till ett notchfilter. Detta var ett misslyckande då det närmaste vi kunde komma 50 Hz med notchfiltret var 51,125 Hz. Notchfiltret togs därför bort ur kopplingen helt och hållet.

- **Försök 3:** Vi testade att koppla på en likspänning efter bandpassfiltret, detta för att höja upp nollnivån till 2,5 V och sedan få signalen att där pendla med ett utslag på 1 – 2 V. Detta försök slog ut signalen totalt och avskrevs därför.

- **Försök4:** Vi försökte här efter bandpassfiltret koppla in signalen till en helvågslikriktare. Helvågslikriktaren gjorde det den skulle men signalen lades negativt istället. För att komma ifrån detta så gjordes ett försök med en inverterande förstärkarkoppling, detta för att få signalen positiv igen och även för att förstärka upp den något. Grova störningar upptäcktes på signalen (kraftigt brus över hela signalen samt att signalen liknade en spik mer än en halv sinusvåg) och denna lösning avskrevs därför.

- **Försök5:** Detta försök är den nuvarande lösningen, signalen var någorlunda ren och fin. Den hade dessutom ett maximalt minusutslag som ligger under 60 mV vilket var mer än välgodkänt för att ta in i PICen. Om den kunde ta in signaler från kroppen tillräckligt bra var däremot fortfarande tveksamt och den behövde troligtvis ytterligare någonting i sin koppling. En koppling för en Bodyrefpunkt (referenspunkt på kropp) kunde vara en bra fortsättning på detta men på grund av tidsbrist så kunde detta inte att

göras.
En annan lösning var att köpa en redan godkänd och testad förstärkarkoppling, detta skulle funka då övriga delar av projektet var färdiga.

Information gällande övriga försök

Notchfilter

Detta filter är egentligen ett bandspärrfilter men samtliga notchfilter har egentligen 2 saker gemensamt. Det ena är att de är väldigt smala bandspärrfilter, det andra är att det ska ta bort den frekvens som läcker från vägguttagen (d.v.s. 50 Hz här i Sverige).
Följande formler används för att räkna på notchfiltret.

$$f = \frac{1}{2 * \pi * R * C}$$

Vi använde oss av fyra resistorer, fyra kondensatorer som är lika stora, två OP-förstärkare och en reglerbar resistor ”potentiometer” som hjälpte oss att reglera bredden på notchfiltret. Detta fungerade dock inte exakt som vi önskade då den har en förskjutning på 1,125Hz (notchspärr = 51,125 Hz) och verkade inte dämpa denna frekvens tillräcklig bra. Detta togs fram genom testning av olika frekvenser.
Ref: [15]

Helvågslikriktare

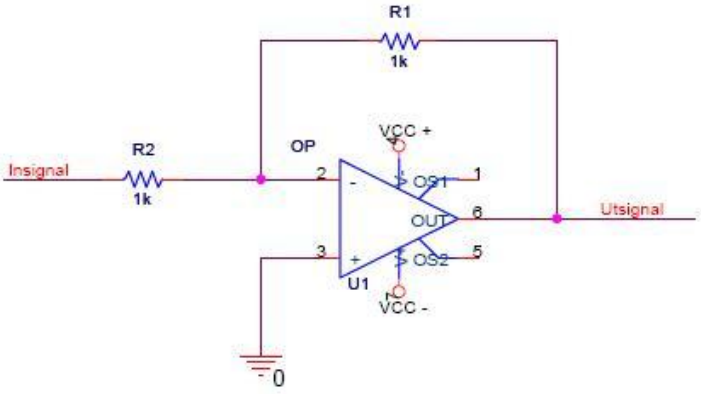
Vänder alla perioder i en sinusvåg till positiva, den vi byggde matade dock ut signalen på negativ axel. Pga. tidsbrist så felsöktes denna lösning inte något ytligare utan avskrevs som inte användbar. Det fanns dessutom en hel del störningar i kopplingen.

Ref: [14]

Inverterande förstärkare

Förstärker och inverterar en signal efter följande princip. D.v.s. den positiva halvperioden blir negativ och tvärtom. För att räkna ut förstärkning användes följande formel:

$$A_v = \frac{U_{ut}}{U_{in}} = -\frac{R_2}{R_1}$$



Ref: [14]

PICens programmering

PICen är en krets med 28 pinar, den har 3 portar som anropas med register hjälp av register, den klarar seriell överföring, AD omvandling. Den är även programmerbar.

AD omvandlingen

För att fixa AD omvandlingen så utnyttjades PICens egna register.
För att AD omvandla på PICen så behövs fem register vilka är följande:

- ADCON0 = Används till att ställa in vilken kanal (bit 5 – 2) som man skall ta in den analoga signalen ifrån. Den används även till att sätta på AD omvandlaren (bit 0) i PICen samt starta AD omvandlingen (bit 1). Bit 7-6 används inte.
- ADCON1 = Används till att välja var man skall ta in referens spänningen utifrån eller ifrån PICen (bit 5 – 4). D.v.s. 1 för extern referens spänning, 0 för intern. Just nu tas spänningen externt, detta för att själva kunna reglera referensspänningen till ett önskat värdet.
  Detta register säger även vilka portar som skall ha analog samt digital ingång (bit 3 – 0). Ben 2 & 3 ställdes in som analoga och resterande ben som digitala (1101), värt att tänka på med detta är att du helst bör sätta de analoga portarna tidigt då de blir analoga i en serie. D.v.s. om du gör ben 3 analog på PICen så kommer ben 2 att bli det också, gör du ben 4 analog så kommer ben 3 & 2 att bli det också osv.
- ADCON2 = Används till om du vill sampla den analoga insignalen med 16 eller 8 bitars noggrannhet (bit 7). Denna bit nollställde för att sampla med 8 bitars noggrannhet. D.v.s. varje analogt spänningsvärde motsvaras av ett 8 bitar långt binärt nummer.
  Detta register styr även hur lång tid varje sampling ska ta så kallat Acquisition Time denna tid säger egentligen bara hur länge den ska vänta mellan varje sampling relativt AD omvandlarens klocka (bit 5 – 3). Även för denna klocka kan hastigheten ställas in på hur snabbt den ska sampla, samplingen styrs nämligen med en latch som kommer att mata in siffrorna till det binära numret efter en viss klocka, denna klocka är PICens systemklocka vars hastighet har delats ner ett antal gånger (bit 2 – 0).
- ADRESL: L:et står för Low och d.v.s. LSB bitarna, ADRES registerna är där samplingen hamnar i. Detta register används dock inte i vår lösning då vi bara utnyttjar de mest signifikanta bitarna vid omvandling enligt inställningen i ADCON2, bit 7.
- ADRESH: H:et står för high och innebär att detta register lagrar MSB bitarna efter varje AD omvandling. Detta register måste läsas av efter varje AD omvandling då man vill ta emot samplingen enligt inställningen i ADCON2, bit 7.

Ref: [3]

In & ut portar på PICen

Egentligen så är det två saker som är värt att hålla reda på här och det är TRIS och PORT, TRIS används för att deklarera vilka portar som skall vara in samt ut portar. Om vi t.ex. vill sätta port A som utgång skriver vi TRISA = 0b00000000; eller TRISA = 0x00; om du gillar det hexadecimala skrivsättet. Vill du istället sätta port A som ingång skriver vi istället TRISA = 0b11111111; eller TRISA = 0xFF; hexadecimalt skrivsättet. Som vi redan vet så har PICen 3 utgångar (A, B & C). Vill du t.ex. påverka port B istället så skriver du TRISB samt TRISC om du vill påverka C.
T.ex.

```
TRISB = 0x00;          //Port B utgångar
PORTB= 0b01010101;     //varannan etta, varannan nolla skrivs ut på PORTB
                       //benen på PICen
```

Det är även fult möjligt att göra en port till en blandning av in och utgångar.
Ref: [4,6]

Seriell överföring mellan PICen och Bluetooth

Seriell överföring fungerar i princip på samma sätt för PICen oavsett vad du kopplar den till, d.v.s. det spelar ingen roll om vi skickar till en PIC, en dator eller en Bluetooth modul. Principen bygger på följande. Bägge komponenterna som kopplas samman måste vara inställda på samma sak vad gäller följande: Baudrate, startbit, stopbit och sändningslängd (d.v.s. 8 bitar eller 16 bitar).
De register som styr detta behandlas nedanför:

- OSCCON= I detta register väljer man om man vill använda intern eller vilken typ av extern oscillator man vill ha, i nuläget så körs den på intern oscillator d.v.s. vi använder PICens egen processorklocka för att styra hastigheten (Bit 0 & 1).
  Vi tillåter även i detta register att den är ostabil (Bit 2). Vi låter den dröja lite innan den startar efter inställning med en lite start up timer (Bit 3). Vi sätter dessutom upp klockan till max (8 MHz) med hjälp av bit 4 – 6 (viktigt bit). Bit 7 används ej.
- INTCON= Enablar enkelt förklarat lite olika interrupt som behövs vid seriell överföring.
- SPBRGH & SPBRG är de två register som i huvudsak styr Baudrate, med baudrate menas som vi redan vet symbolhastigheten, på PICen så motsvaras varje symbol av en bit vilket gör att Baudrate i detta fall är det samma som bit/s.
  För att veta vilket värde som skall placeras i dessa register så kan detta göras på två sätt.

  Antingen görs detta med följande formel $\left(\frac{\left(\frac{FOSC}{Baudrate} \right)}{64} \right) - 1 = X$

  där FOSC är processorklockan, d.v.s. 8 MHz, baudrate är den baudrate vi vill uppnå (för seriell överföring till Bluetooth krävs en baudrate av 57600 för den Bluetoothmodul som används vilket även är ungefär vad PICen klarar av) och X är det nummer som vi placerar i SPBRG eller SPBRGH beroende på hur vi ställt in övriga register. Det finns dock en bättre lösning som ger dig mer kontroll över hastigheten och det är att kolla på tabellerna som står angivna i manualen på sid. 245 & 246 (detta för att tabellen i sig förmedlar information om felmarginal etc.),

detta kräver möjligtvis något mer förståelse men ger dig större säkerhet på att resultatet blir bra.
Detta är även vad vi i slutänden använde oss av efter en tabell på sid. 246 som gav oss ett maximalt sändfel på 0.79 % vilket vi även mätt upp. Denna fina marginal uppnådde vi efter uppmätning.

- TXSTA= Står för Transmit status & Control register, d.v.s. den styr hur PICen ska sända, det vi använder oss av i detta register är att vi möjliggör sändning (bit 5), vi sätter ett synkroniserat avbrott mellan varje sändning (detta tolkar mottagande enhet troligtvis som en stopbit (bit 3), Bit 2 & 1 är ettsatt, detta gör ingenting men om vi vill växla till asyncon överföring så kommer denna stopp biten ställa in PICen för hög hastighet. Övriga bitar är nollsatta och tas därför inte upp.
- RCSTA= Vi ställer här in i stort sett samma som föregående fast för mottagning istället. Dock så finns det en bit som är viktig och det är bit 7, den ställer in ben 17 (TX = Transmitt) & 18 (RX = Recieve) som seriella portar.
- BAUDCON = Säger hur Baudrate ska bete sig, t.ex. om vi mottar/skickar inverterad data eller om man sänder/mottar ett visst värde i rätt följd (Bit 4 & 5). Den säger även om man bara ska utnyttja SPBRG eller både SPBRG & SPBRGH. Säger även till PICen hur den ska motta/sampla inkommande data.
- RCIE = Är i själva verket bit 5 i ett register som heter PIE1, denna bit enablar överföringen. D.v.s. i och med att den här biten är ettsatt så slår man på den seriella överföringsmekanismen i PICen.
- TXREG = Detta är registerna vi lägger ut vår 8 bitars variabel på då vi vill sända den.
- RCREG = Detta är de register som vi mottar vår 8 bitars variabel på om vi mottar något vilket vi i nuvarande kod inte gör men har möjlighet att göra.

Ref: [5,7]

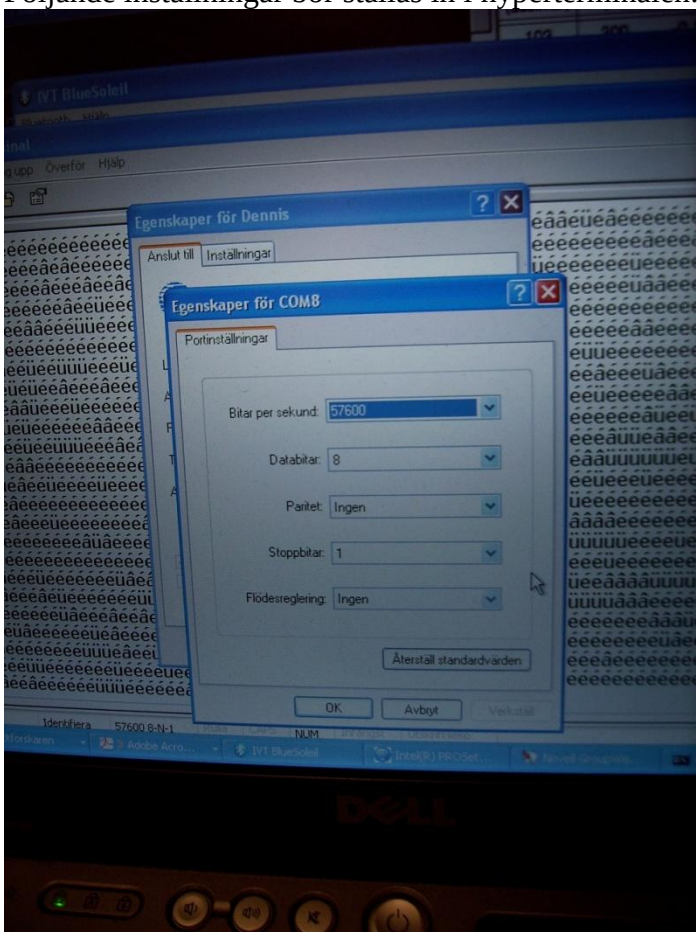
Resultatet av PIC programmeringen

Vi upptäckte att PICen har sina begränsningar, bland annat är det svårt att få en hög samplingshastighet samtidigt som vi sänder. Vår första tanke var att sampla i 4 kHz hastighet och samtidigt sända de samplade värdena med denna hastighet.
Problemet med detta var att vi fick ett fel vid sändning då vi gjorde på detta sätt. PICen har också en tendens att skriva över sina egna sända värden då du gör flera sändningar på raken för snabbt detta gör att du inte får alla värden som sänds vilket resulterar i att du måste ha en viss fördröjning mellan varje sändning.
Detta kan summeras i att sändningen i sig kommer att köra samplingen vid AD omvandling med väldigt låg frekvens (några 100 Hz) vilket absolut inte uppfyller vårt mål på åtminstone 5 - 10 samplingar per 200 Hz kurva. Vår lösning på detta blev att låta PICen sampla in en stor mängd värden (50 stycken) för att sedan räkna ut medelvärdet på dem. Vi la även in en liten spärr som inte kommer att räkna med de värden som är lägre än 50 mV då dessa värden troligtvis kommer att vara små störningar från den negativa delen (som vi slagit ut) av sinusvågen. Detta gjorde att vi bara samplade värden mellan 0,050 – 5 V.
Detta värde räknas sedan om till fem 8 bitars ASCII variabler. Varav den sista var ett komma tecken (,) vilket vi sedan utnyttjade för att skilja varje datavärde åt i C-/Excel programmet. D.v.s. vi fick in värden av strukturen 1.22,3.44,5.66,osv…
Detta medelvärde skickade vi sedan ut till Bluetoothmodulen med en baudrate på 57600 bps (57600 bit/s). Effekten av att vi först samplar och sedan sänder gjorde att vi faktiskt fick upp en samlingshastighet på PICen på 2033 Hz vilket uppfyllde vårt mål på 10 samplingar per 200

Hz kurva. Projektet med programmeringen av PICen kan sammanfattas i att vi tangerat vad PICen klarar av och utnyttjat dess fulla styrka, pressar vi ur den mer så brinner den troligtvis upp.
Allmän Ref på PICen: [Robert, Percy samt Appendix, Del1]

EMG ritarna

Två ritprogram skapades, bägge dessa utnyttjar en textfil från hyperterminalen för närvarande för att kunna köras.
Följande inställningar bör ställas in i hyperterminalen:



Man väljer sedan transfer -> capture text för att fånga in textströmmen.
Denna data ström bör för närvarande sparas ner i dokumentet Bluedata.txt som ligger i MuscleArray 1000 mappen (eller annan mapp man länkat programmet till).

C-prototypen:

I projektets början så fanns det ett par saker som var viktiga att få bekräftade.
En av dessa saker var om vi över huvud taget skulle ha möjlighet att kunna presentera informationen på skärmen. Vi gjorde därför en prototyp som tog in data från en textfil via ett C-program som sedan presenterade datan som höjden på en stapel på skärmen. Detta gjordes med hjälp av lite modifieringar av ett gammalt C-program som fanns från en tidigare kurs i C-programmering samt en GFX ritarmodul. GFX modulen är ett litet program som tillåter dig att rita ungefär som i Paint fast istället för att du använder datormusen så används nu istället koordinater från ett C-program. Tidsåtgången för att genomföra detta var en dag och resultatet

av detta var att vi hade bevisat att det troligtvis var möjligt att använda den här lösningen.
Ref: [8, Robert, Percy & Mahshid]

Det färdiga C-programmet:

C-prototypen utvecklades vidare, dels så togs tidigare buggar bort som t.ex. att den ritade lite fel från början. Det andra som gjordes var att modifiera hur den läste in datan. I den ursprungliga C-prototypen så letade den bara efter en position och tog in den. Nu fick den istället ta in varje värde och låta det värdet reglera höjden på stapeln fortlöpande. Resultatet av det försöket var ett misslyckande och visade helt enkelt att GFX ritaren drunknade i information och att den i slutändan absolut inte klarade att göra någonting alls av det snabba informationsflödet.
Vi började i detta skede titta på en alternativ lösning som utnyttjade Visual Basic och Excel. Vi fortsatte dock att modifiera C-programmet, detta dels för att när man väl kan grunderna i Visual Basic så spelade det ingen roll om man gör ett program i C eller ytterligare ett liknande program i Visual Basic, principen för det hela blir den samma. C-program byggdes sedan på med ytterligare en funktion som jagar ikapp det sista inskrivna värdet i dokumentet och skriver ut det på skärmen. Detta är för att alltid få det senaste värdet och därigenom komma så nära realtid som möjligt.
Den andra modifikationen som gjordes var att sätta in en liten delayfunktion som klarar att styra delaytiden någorlunda exakt. T.ex. om du i funktionen skriver in att du ville fördröja programmet 0.2 Sekunder så fick man i slutändan en fördröjning på 0,212 sekunder. Vilket man som användare aldrig någonsin kommer att märka. Programmet läser även nu in värdena från en textfil som konstant uppdateras från en hyperterminal och inte från ett statiskt dokument som vid C-prototypen.
Ref: [9, Robert & Percy]

Excelprogram lösningen:

Den andra lösning som togs fram var en där ett Visual Basicprogram körs i bakgrunden av Excel. Även här så tar programmet in värdena från en textfil som konstant uppdateras från en hyperterminal. Tanken bakom programmet samt dess grundstruktur liknar den hos C-programmet. Enda skillnaden var att lösningen i Microsoft Visual Basic tog in värdena en och en medan lösningen i C läste in en hel sträng på en gång som den sedan sönderdelade.
Vid varje inläst värde så kommer cellerna först att förskjutas ett steg nedåt.
Det nya värdet (näst sist inlästa) skrivs sedan in i den första cellen igen samtidigt som Diagrammet uppdateras. Anledningen till att vi tog det näst sista inlästa värdet var att vi annars kan få ett halvt värde vid öppnandet av textdokumentet t.ex. 3.55 kanske bara blir 3 i just det ögonblicket, någon tiondels sekund senare kanske 3.55 hunnit skrivas in men Excel kommer bara att ha en ögonblicksbild av inskrivningen till filen.
Detta fel fick vi inte i C då man där har större kontroll över inläsningen.
För att uppdatering till diagrammet ska ske kontrollerad så utnyttjades en liknande timmer som den i C. Vi körde dessa steg i en oändlig lopp, programmet avslutas sedan med kommandot ”ESCAPE”. Macrot kopplades sedan till en bild som fick fungera som start knapp.
Ref: [10, Robert, Percy]

Vad lösningarna saknar:

Lösningarna saknade ett viktigt element och detta var full realtid. För närvarande kommer bägge lösningarna väldigt nära realtid. Problemet är att de för närvarande utnyttjar en textfil som källa vilket gör att det saktas ner något ju större dokumentet blir. En lösning som vi tittat på men inte helt lyckats uppnå är att ta in dessa värden direkt från COM porten.
C – programmet är något snabbare men Excelprogrammet körs för närvarande på ett snyggare sätt samt att den även visar historik genom ett fortlöpande diagram som visar nuvarande samt de senaste 9 värdena. Med andra ord så kommer det nyaste värdet på linjediagrammet att vara längst till vänster och det äldsta att vara längst till höger. Vid skiftning i dataserien så trillar det äldsta bara ut samtidigt som det nyaste trillar in från vänster igen.
Ref: [Robert, Percy]

För att lösa uppgiften med grafikritarna:

- Kunde sen tidigare:
  - C-programmering
  - Grundkunskaper i Excel
- Behövde lära oss:
  - Microsoft Visual Basic (nytt programmeringsspråk)
  - Djupare kunskap i C-programmering (t.ex. time.h biblioteket för att skapa timer)
  - Djupare kunskaper i Excel 2007 samt hur man kopplar in Visual Basic Macron i den.

Ref: [Robert, Percy]

Bluetooth mottagning med IAR systems

Målet var att skapa ett c-program som läser in data från Bluetooth sändaren i realtid.
En möjlig lösning på detta var att koppla COM porten rakt in i grafikritarprogrammet i C som kördes i Dev C++. En USB bluetooth adapter från Belkin fanns tillgänglig för att ta emot dataflödet och började därför testköras. En Bluetooth enhet kan fungera som både mottagare (slave) eller sändare (master). Informationen som sänds från PICen via Bluetoothmodul är inställd som master därför måste datorsidan ställas in som slave. Målet med det här var att kunna visa informationen på en datorskärm i realtid. Bluetoothenheter ingår i olika profiler som förklarar vad enheten har för funktion. General Access Profile (GAP), Serial Discovery Application Profile (SDAP), Serial Port Profile (SPP) är de profiler som använts i IAR systems programvara. Bluetoothmodulen som ingår i ”Bluetooth starter kit” från IAR systems, kopplas till PC via en USB- eller UARTkabel. Ifall man använder en UARTkabel vilket vi gjorde kan man koppla informationsströmmen till en tillgänglig COMport på datorns baksida. När programmet är installerat tillkommer en Bluetooth protokoll stack exempel i C-programmet som är kompilerbart i Microsoft Visual C++. I programmet kan man sedan välja att vara antingen master eller slave. För att ansluta till bluetootmodulen som är kopplad till PICen så måste programmet ställas in på slave. Den kommer därefter att leta i luften efter andra Bluetoothenheter som den kan ansluta till. I programmet kan man sedan välja att skicka en förfrågan (inquiry) till en master som erbjuder service. Mottagaren måste acceptera den service som mastern skickar. Denna lösning las dock på is då programmet endast var körbart i Visual C++.
Ref: [11, 12, 13]

C-program för att läsa in EMG värdena från Comport i Dev C++

Då andra delar började fungera igen så togs comports projektet upp på nytt igen. Målet denna gång var att göra programmet körbart i Dev C++ eller hitta en annan lösning för att koppla dataflödet från en comport in i Dev C++ utan att behöva ta omvägen via en textfil.
Det visade sig möjligt att konfigurera bluetooth adapter till virtuell comport för att få in dataflödet.
Två kablar kopplades mellan comporten mellan två datorer för att testa hur den fungerade. Vi lyckades att skicka och ta emot text dokument och data filer. Den riktiga comporten ersattes sedan med en virtuell från hyperterminalen. Data kunde mottas men det fungerade väldigt långsamt samt att man fick varje värde två gånger.
Hastigheten gick inte att få upp i full realtid och denna lösning lades därför ner.
Ref: Samin

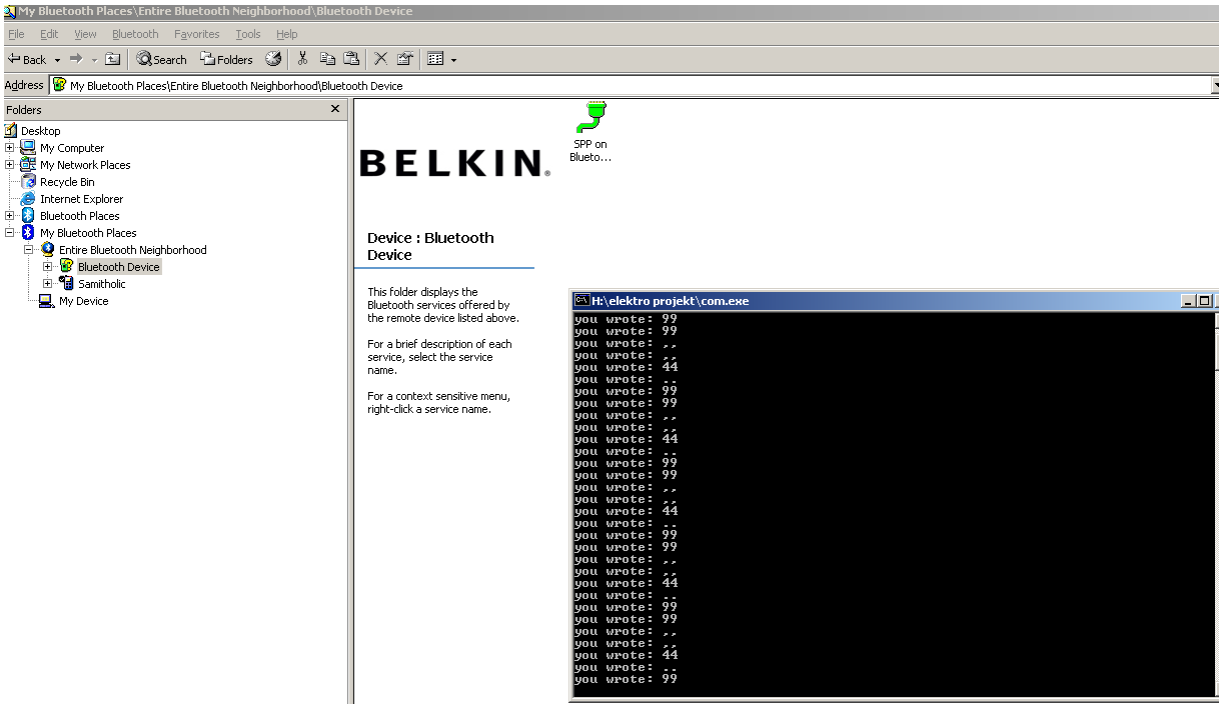


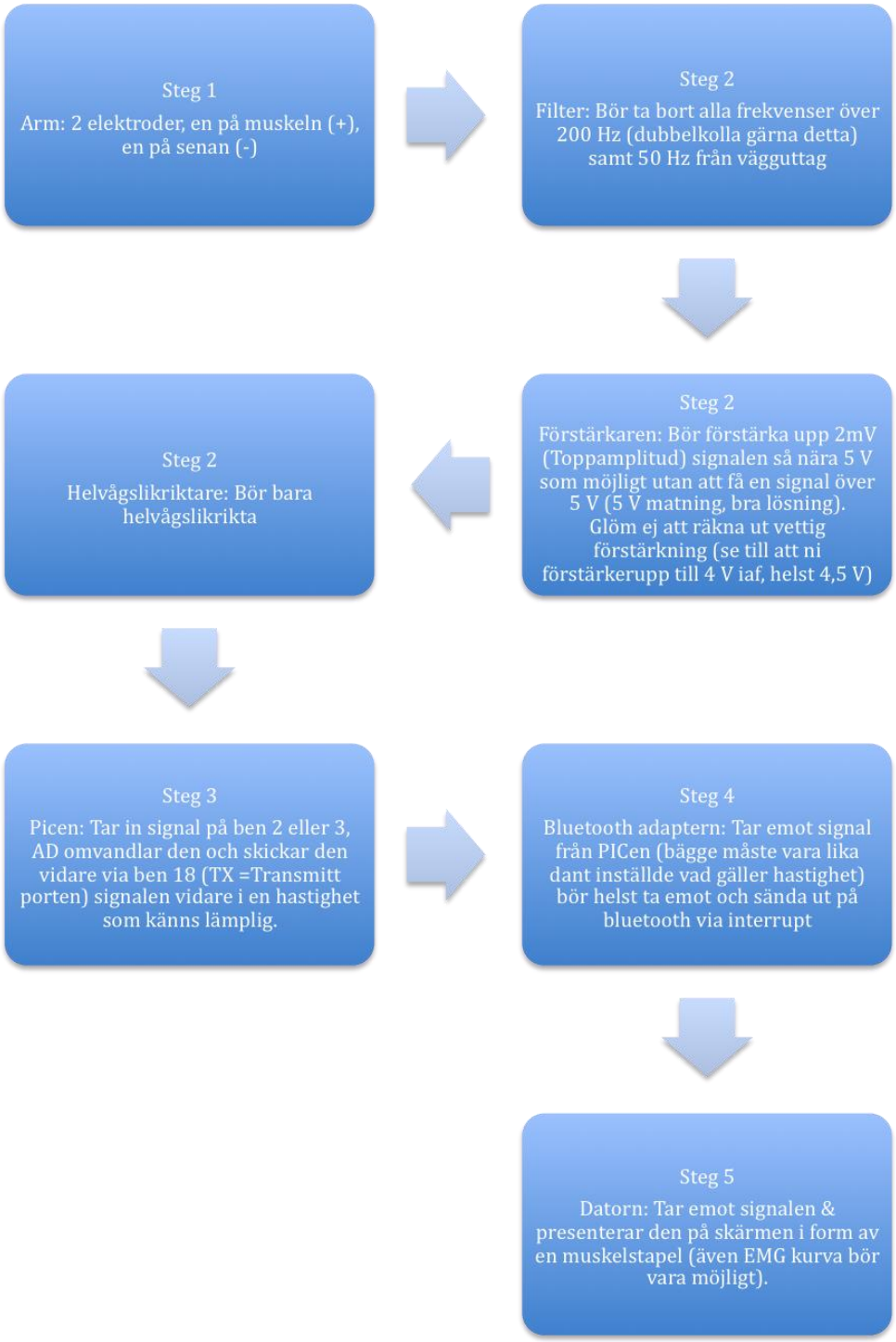
Bild 3: Den felaktiga samt långsamma mottagna datan.
Ref: Samin

Slutsats

Projektet har överlag varit ganska lyckat, vi har uppnått samtliga av våra mål för det här projektet förutom målet med en fungerande förstärkarkoppling. Mer tid behövs dock för att få detta att bli en färdig produkt, men tanken att bygga en EMG förstärkarkoppling i träningssyfte för en kostad av omkring 500:- är ändå bra. De områden som man bör fortsätta jobba med om man tar upp arbetet med denna rapport är förstärkardelen, skapa färdigt produktionsexemplar i Orcad Layout samt skapa en prototyp med hjälp av detta.
En bättre kod som uppfyller Bluetooth realtid bör även skapas, denna kod föreslår vi ska vara möjlig att koppla till en inbyggd Bluetooth enhet i en dator eller en Bluetoothsticka.
Brist på tid gjorde att vi inte hann göra klart projektet som vi hade tänkt oss från början, vi anser att projektet är värt att bygga vidare på om man skulle ha mer tid.

Appendix

Del 1 – Flödesschema



Del 2 – PICprogrammet

```
/*****************************************************************************
 *
 *
 *
 *****************************************************************************
 * FileName:        BluetoothMedFungerandeSampling.c
 * Dependencies:
 * Processor:       PIC18F2550
 * Compiler:        HighTech PICC18 (STD 9.51 or later)
 * Company:         STH KTH
 *
 * Purpose:         A short demo program for serial communication using EUSART
 * This program only loop-backs the received data
 * Interrupt is used for synchronisation
 * No Auto-Baud Rate detection implemented
 *
 * Refer to PIC18F2550 manual chapter 20
 *
 * Output:          Serial data on TX pin (RC6) pin 17
 * Input: Serial data on RX pin (RC7) pin 18
 *
 *
 *
 * Author                   Date                    Comment
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Robert & Percy           20080930
 *****************************************************************************/
#include <pic18.h>

void init(void);
void rec(void);
void init_eusart(void);
void interrupt eventAD(void);
void init_timer(void);
unsigned int i, RaknaSample=0, SparaAllt[100], MedelResultat=0, j=0, n=0, p=0;
int HelTal,RestTot,Rest1,Rest2, MedelSumma=0, SparaAD, SendDelay;
//
int main ()
{

        init();
        init_timer();
        i = 0b00110011;
        ADCON0 = 0b00000101;   //Godown , ADON
        ADCON1 = 0b00000001;
        ADCON2 = 0b00001001;   //tre sista bitar bestämmer när godone ska bli 0

        TRISA  = 0b11111111;    // 6 bits INPUT on Port A
        TRISB  = 0b00000000;    // 6 bits OUTPUT on Port B
        PORTA = 0;              // 2 bits High on Port A at start
        TMR2ON = 1;             //Enablar interupt
        TMR2IE = 1;             //Enablar interupt

        n=0;
        while(1);
}
void interrupt eventAD(void)
```

```
{
	//Grej för att slå bort en bygg som uppkommer ibland.
	if(n==0)
	{
		n=1;
		SparaAD=0;
		MedelSumma=0;
		j=0;
		SendDelay=13;
	}
	//RCIE = Enablar Timer interrupt
	//Om denna är 1 så har PICen tillåtit en timerinterupt
	//RCIF = Ser om interrupten har genomförts
	//Om 1 så har den det
	if(TMR2IE && TMR2IF)		//Har vi fått den interrupt vi ville ha
	{
		GODONE = 1;		//kontrollerar GODONE flaggan

		while( GODONE == 1 )
		{
			;
		}
		SparaAD = 1.957 * ADRESH ;
		n++;
		//Ignorerar bruset som blir efter helvågslikrikning av den negativa delen av sinusvågen
		if(SparaAD>5)					{
			MedelSumma = MedelSumma + SparaAD;
			j++;
		}

		//Efter värden så räknas medelvärdet ut, räknas om till ASCII varefter allt skickas
		//som t.ex. 2.33,
		//D.v.s. 2.33 V följt av ett komma tecken för att skilja varje värde åt.
		if(j==50)
		{
			j=0;

			MedelResultat = MedelSumma/50;
			MedelSumma=0;
			RaknaSample = 0;
			HelTal = MedelResultat/100;
			HelTal += 48;

			TXREG = HelTal; // Skickar Heltal
			RestTot = MedelResultat%100;
			for(i=0; i<SendDelay; i++)
			{
				;
			}
			TXREG = 46;	//Skickar . (Punkt)
			Rest1	= RestTot/10;
			Rest2	= RestTot % 10;
			Rest1	= Rest1 + 48;
			Rest2	= Rest2 + 48;
			for(i=0; i<SendDelay; i++)
			{
				;
			}
			TXIF = 0;
			TXREG = Rest1; // Skickar Rest 1
```

```
                        for(i=0; i<SendDelay; i++)
                        {
                                ;
                        }
                        TXIF = 0;
                        TXREG = Rest2; // Skickar Rest 2
                        for(i=0; i<SendDelay; i++)
                        {
                                ;
                        }
                        TXIF = 0;
                        TXREG = 44;     //Skickar , (Komma)
                        for(i=0; i<SendDelay; i++)
                        {
                                ;
                        }
                        TXIF = 0;
                }
                TMR2IF=0;                       //TMR2IF måste nollställas efter att den använts
        }
}
void init_timer(void)   //interapt efter en vis tid enkel sagt.
{
        IPEN = 1;                       //Enablar prioritetsnivåer för interrupts
        PR2 = 0b01111001;               // Säger hur långt timern ska räkna
        T2CON = 0b00000001;             //Sätter hastighet på räknaren

        INTCON=0xC0;                    //0b11000000 GIE and PEIE
                                        //GIE=1 Enablar alla unmasked interupt (startar global interupt)
                                        //Med andra ord interupt on
                                        //PEIE=1 Enablar alla unmasked peripheral interupt.
        TMR2 = 0b00000000;

        TMR2IP = 1;
}
//Initiering av RX & CX kommunikation
void init(void)
{
        OSCCON=0x73; //MSD 0b01110011 LSB
                        //Bit 0 & 1 = intern oscillator väljs
                        //Bit 2 = INTOSC, ostabil
                        //Bit 3 = Prim'r oscillator ej redo, start up timeout running
                        //Bit 4 - 6 = 8 MHz clocka väljs
                        //NOTIS: OSCON klockan kommer påvärka både AD omvandlaren &
                        //"Överföringshastigheten"
                        //Bit 7 nada
        TRISC=0xC0;             //0b11000000 RC7 and RC6 for USART function
                        //(Behövs för initiering)
        init_eusart();
        INTCON=0xC0;  //0b11000000 GIE and PEIE (for USART on Port C)
                        //GIE=1 Enablar alla unmasked interupt (startar global interupt)
                        //Med andra ord interupt on
                        //PEIE=1 Enablar alla unmasked peripheral interupt.
}

void init_eusart(void)
{
        SPBRGH=         0;      //Detta register genererar BAUDRATE (Hightbyte)
                                //Clearar detta register här.
                                //SPBRGH är ett sorts timer register för baudrate.
```

```
                                //Om autoBAUDRATEså kommer lämplig BRG perioden att lämnas i detta
                                //register efter den femte "rising edge" som kommer in på RX pinen
                                //Dvs. lämplig överföringshastighet lagras här.
        SPBRG=34;               //Setting up for 56.69 Baud Rate;
                                //Värdet 34 fås ur en formel på sid 244,kolla PÅ tabel 246
                                //((FOSC/Önskad baudrate)/64)-1)=Önskat SPBRG värde
        TXSTA=0x2C;             //0010 1100, BRGH = 0
                                //Transmit status & controll register
                                //Bit 7 = 0 ger slave mode (ger clocka från extern källa
                                //Bit 6 = 0 ger 8 bitars sändings storlek (9 bitar om 1) ***
                                //Bit 5 = 1 ger Enablar överföring (transmitt enable)
                                //Bit 4 = 0 ger Asyncon överföring (asynconus mode)
                                //Bit 3 = 1 ger sync break på nästa sämdbit på TX
                                //Bit 2 = 1 ger Low Speed asyncon (baudrate eller hastighet)
                                //Bit 1 = 0 ger Sätter ner transmit shift registret till 0 (dvs. full)
                                //Sätts till 1 då allt skickats (tror vi)
                                //Bit 0 = 0 ger inget alls, används vid 9 bitars överföring.
        RCSTA=0x90;             //1001 0000
                                //Recieve status & controll register
                                //Bit 7 = 1 sätter RX och TX som seriella portar
                                //Bit 6 = 0 sätter 8 bitars mottagning (9 bitar om 1)
                                //Bit 5 = 0 Don't care om asyncron vilket vi har
                                //Bit 4 = 1 Enablar mottagning här om asyncon (vilket vi har)
                                //Bit 3 = 0 Disablar adress detektion, 9 biten kan användas som
                                //Paritetsbit om denna är (möjliggör lite interrups om 1 satt)
                                //Bit 2 = 0 Tittar inte efter framing error
                                //Bit 1 = 0 Tittar inte efter overrun error
                                //Bit 0 = 0 Är tom, används till parity bit eller nått sånt
        BAUDCON=0x4A;           // 0100 1010, BRG16 = 0
                                //Baud rate controll
                                //Bit 7 = 0 Säger att ingen BRG rollover har skett ????
                                //Bit 6 = 1 kommer aktivera mottagarens operator Mottagning (RX) on lr nått
                                //Bit 5 = 0 Säger om den mottagna datan är inverterad lr ej
                                //1:a säger att den är inverterad. (Bak och fram) (detta fixar det)
                                //Kort och gott skickar du 00101110 så kommer det fram 01110100
                                //Denna bit är rolig, vi måste testa den. ****
                        //Bit 4 = 0 Säger att skickad data är inverterad (samma som ovanstående)
                        //Bit 3 = 1 Sätter 16 bitars baudrate (dvs både SPBRGH & SPBRG)
                        //Bit 2 = 0 Används ej
                        //Bit 1 = 1 Fortsätt sampla RX pin, interupt på falling edge, interupt
                        //clearas på rising edge
                        //Bit 0 = 0 ger inget alls, men om ettställt så känns baudrate av automatiskt.
        RCIE=1;         //Enable interrupt on Port C, USART
                        //TXIE=1;
        // Dummy send
        TXREG=0x00;     //Eusart transmitt (dvs. sända)
                        //Notis: Baudrate = symboler/s => På picen = bit/s
}
```

Del 3 – Grafikritar programmet i C

```
/*************************************************************
* Prototyp: Robert, Percy & Mahshid                          *
* Vidareutveckling: Robert & Percy                           *
*                                                            *
*************************************************************/
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
//GFX ritarens bibliotek
#include <GFX_Client.h>
//Klockbibliotek
#include <time.h>
#include <sys/timeb.h>
//Det filnamn man vill hämta infon ifrån
#define FILE_NAME "Bluedata.txt"

int main()
{
  double DoubleValue;
  int Rows=0, LastRow=0, LastValue=0, IntValue=0, PrevValue=0;
  static char s[100000], devider[6];
  int NumPos=0, JumpOut;
  clock_t start, end;               //Deklarering av klockvariabler från time.h

  if( !GFX_Open() )
  {
    printf("Connection failed, GFX_Server running?\n");
    getch(); // hamnar du här har du glömt starta grafikservern
    return 0;
  }
  //Initierar GFX ritaren
  //GFX ritaren måste vara igång för att det ska fungera
  GFX_SetDebugMode( GFX_DEBUG_FULL );
  GFX_SetSize     ( 400, 500 );
  GFX_SetColor    ( GFX_RGB(   0,   0,   0) );
  GFX_SetBGColor  ( GFX_RGB( 200, 200, 255) );

  GFX_ClearScreen ();
  printf("Press any button to start");
  getch();
  FILE* fp;
  fp = fopen(FILE_NAME, "rt");
  if(fp == NULL)
  {
      printf("Your toasted!");
      exit(EXIT_FAILURE);
```

```
 }
 else
    printf("\nBluedata recieved\n");

 //Skapar pekare på filen vi vill läsa
 JumpOut=0;
 while(1)
 {
    FILE* fp;
    fp = fopen(FILE_NAME, "rt");
    while(!feof(fp))
    {
        fgets(&s, "%s", fp);    //Läser in en rad
        Rows++;
        while(LastRow<=Rows && JumpOut==0)
        {
            //Detta tar in ett nummer
            devider[0]=s[NumPos*6+0];
                        devider[1]=s[NumPos*6+1];
                    devider[2]=s[NumPos*6+2];
                    devider[3]=s[NumPos*6+3];
                    NumPos++;                //Denna räknare är till för att flytta markören till
vilket nummer man vill läsa in.
            LastValue = IntValue;
            sscanf(devider, "%lf,,", &DoubleValue);
            DoubleValue*=100;
            IntValue = DoubleValue;

            if(IntValue==25499 || IntValue>555 || IntValue<0) //Jämförelse för att hitta det
senast läsbara numret.
            {
                JumpOut=1;
            }
        }
        JumpOut=0;          //Nollställer JumpOut
        NumPos=0;           //Nollställer Positionsräknaren för värdena.
    }
    LastRow=Rows;
    if(LastValue!=PrevValue)   //Liten if sats för att bara rita förändringar
    {
       PrevValue=LastValue;
       printf("New EndValue= [%d]\n", LastValue);    //Skriver ut det sista värdet i
dokummentet
       GFX_ClearScreen ();
       GFX_Rectangle ( 100, (500-LastValue), 300, 500 ); //Ritar en rektangel med sista
värdet
       start = clock();
       end = clock();
       if(LastRow<10)
       {
```

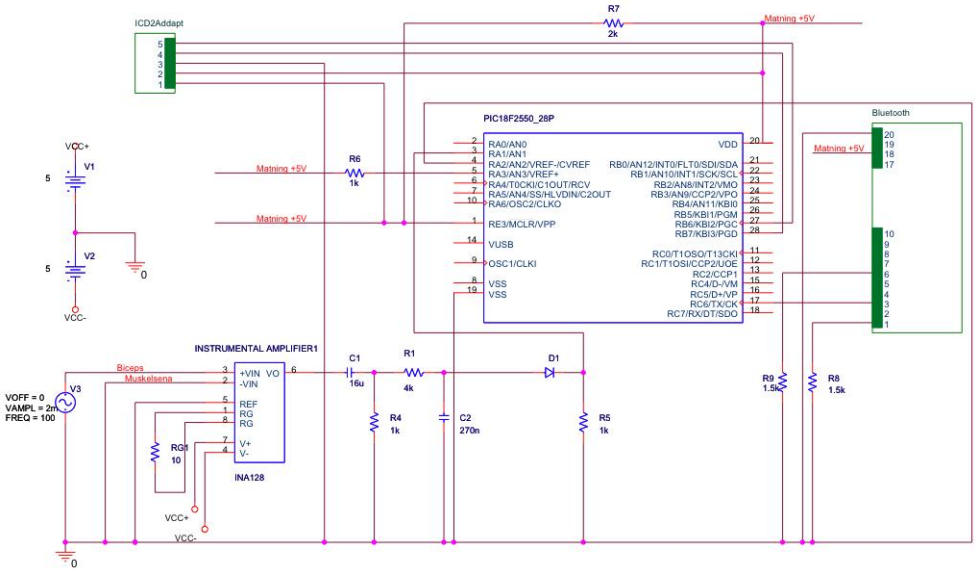
```
            while((((double)( end - start ) / (double)CLOCKS_PER_SEC ))<0.1)
            {
                end = clock();
            }
          }
      }
      //Tidsfördröjning, höger om jämförelsetecknet (<) är den tid i sekunder man vill fördröja
      //Start och end är två samplingar från sekundklockan i windows, differensen mellan dem
      //blir ett visst givet tidsintervall
      //Denna fördröjning behövs för att programmet körs mycket snabbare än ritmodulen
      //Kan även vara bra att sakta ner programmet lite för att spara lite prestanda
      //Dvs. att man gör många onödiga anrop till en fil på hårdisken.


      fclose(fp);

    }


    return 0;

}
```

Del 4 – Visual Basic koden

```
Sub EMG_ultimate_edition()
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   MuscleArray program          '
'   Skapad:     2008-10-13       '
'   Av:         Robert & Percy   '
'                                '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

    'sökväg till filen
    Dim strTextfil As String
    Dim Value, LastValue, PrevValue, BeforeValue As Double
    Dim i As Long
    Dim CountDown, f, TotalDelay As Integer
    Dim PauseTime, Start, Finish, TotalTime
    TotalDelay = 0
    Value = 0
    i = 0
    strTextfil = "C:\MuscleArray 1000\Bluedata.txt"
    While (1)
        f = FreeFile
        Open strTextfil For Input As #f
        While Not EOF(f)
            Input #f, Value              'Tar in ett värde
            If ((Value > 0) And (Value < 5.6)) Then
                LastValue = BeforeValue
                BeforeValue = Value
            End If
        Wend

        Close f
        'Shiftar allting neråt ett steg
        CountDown = 11
        While (CountDown > 2)

            Cells(CountDown, 4) = Cells((CountDown - 1), 4)
            CountDown = CountDown - 1

        Wend
        'Slut på Shifting
        Cells(2, 4) = LastValue
        PrevValue = LastValue
        Range("A1").Select
        'Timerfunktion
        'Kontrollerar farten i timer
        If Not (TotalDelay > 5) Then
            PauseTime = 0.2    ' Skriv hur lång fördröjning vi vill ha här.
            TotalDelay = PauseTime
```

```
            Start = Timer ‘ Startar timern.
            'Här kör vi tills vi når vårt timervärde
            Do While Timer < Start + PauseTime
            Loop
        End If
        'End timer
    Wend
End Sub
```

Del 5 – Kretsschema



Del 6 – Manualen (skriven som om vi hade alla delar fungerande)

Manual till MuscleArray 1000

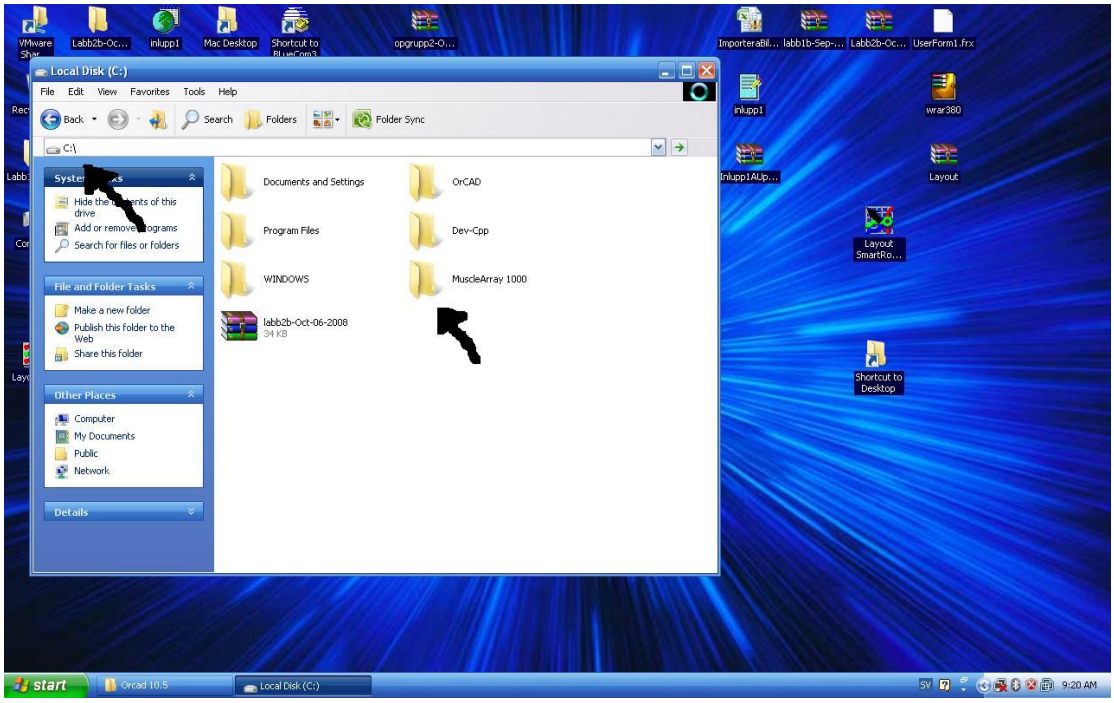
MuscleArray 1000 består av 3 komponenter: 2 elektroder, själva MusclePower omvandlaren samt en Bluetoothsticka. MuscleArray 1000 fungerar även med andra Bluetoothsticka så att just köpa detta tillbehör är inte ett krav.

Elektroderna: De två elektroderna kopplar du på följande sätt:
elektrod märkt med minus fäster du på senan till den muskel du vill mäta på, elektroden märkt med plus fäster du mitt på den muskel du vill mäta på.

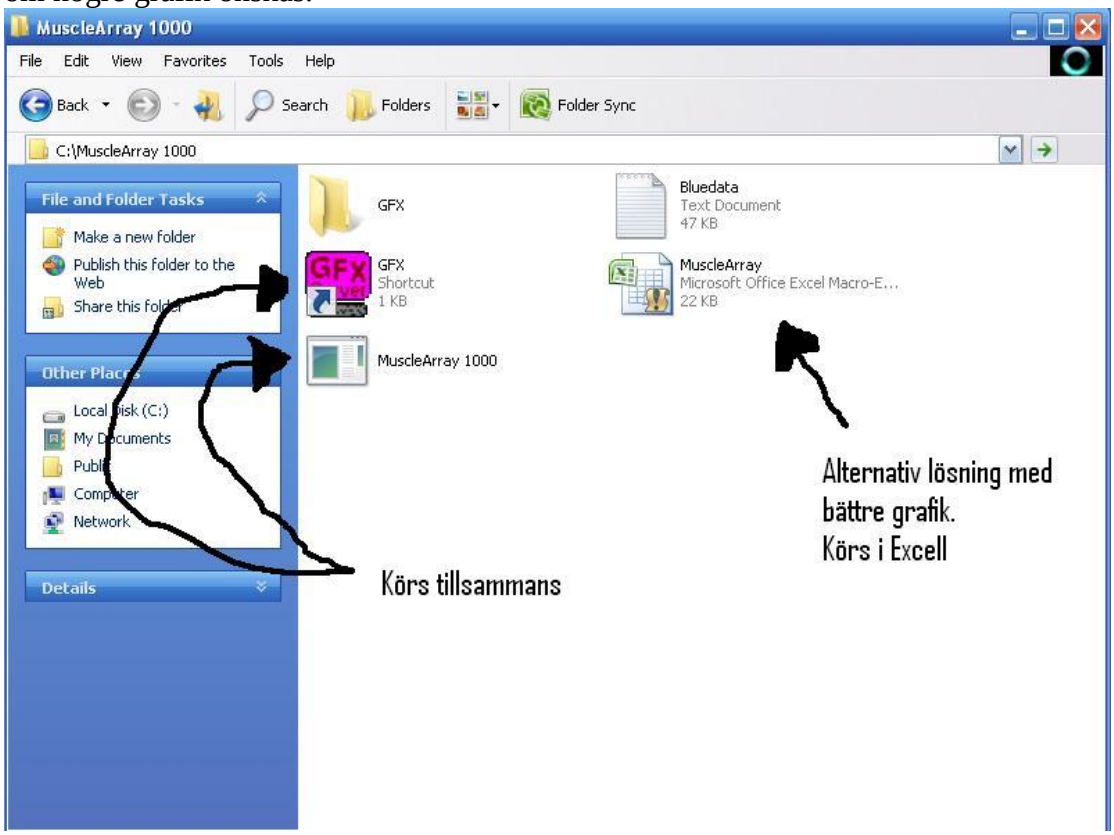
MusclePower omvandlaren består av en burk du kan fästa i skärpet, elektroderna tejpas fast på kroppen och ledningarna dras förslagsvis innanför kläderna och fästes sedan på MusclePoweromvandlaren.
MusclePower omvandlaren drivs på batteri och utnyttjar bluetooth så inga ytterligare ledningar behövs, detta för att möjliggöra största möjliga rörlighet under träningen.

Signalen från den mottas sedan på datorn där två möjliga presentationsprogram finns tillgängliga, den ena består av en exekverbar fil och är tillgänglig efter installation, den andra består av en Excel application, denna kräver ingen nämnvärd installation utan enbart att du ändrar en inställning i Excel.
Det exekverbara programmet körs något snabbare men Excel filen utnyttjar en vackrare grafik och rekommenderas därför för användare med datorer av högre prestanda.

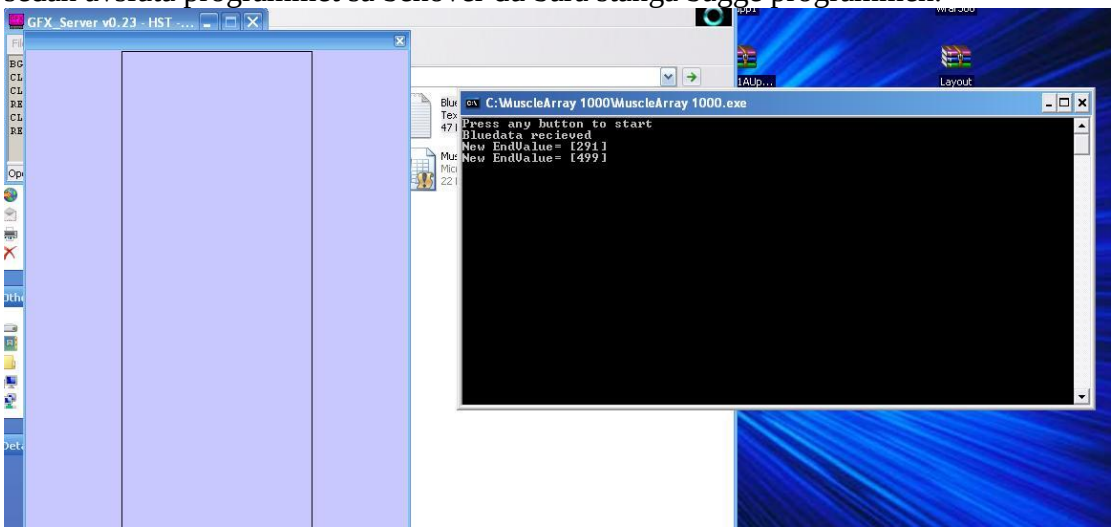
Installation: Gör följande, dra och släpp in mappen MuscleArray i C: Windows från MusclePower CD:n.



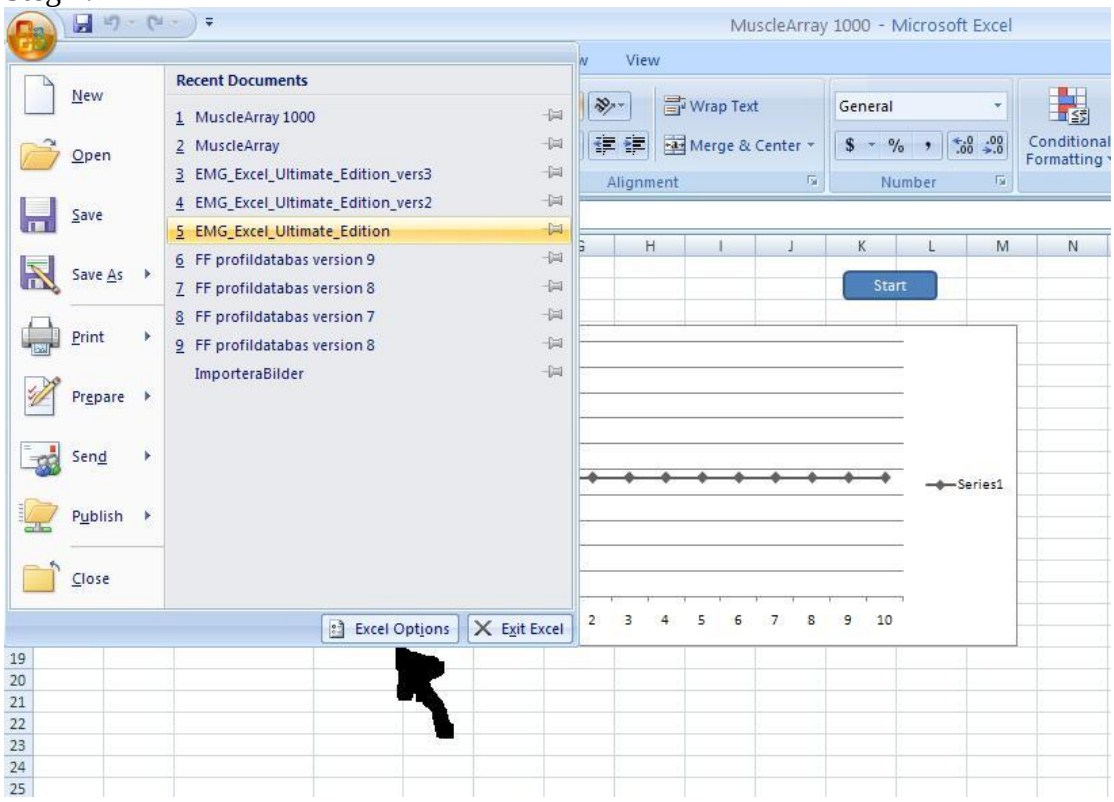
För att starta programvaran:
Öppna mappen MuscleArray 1000 som nu ligger i ditt C:.
Starta sedan den exekverbara filen MuscleArray 1000 samt GFX.exe eller programmet i Excel om högre grafik önskas.



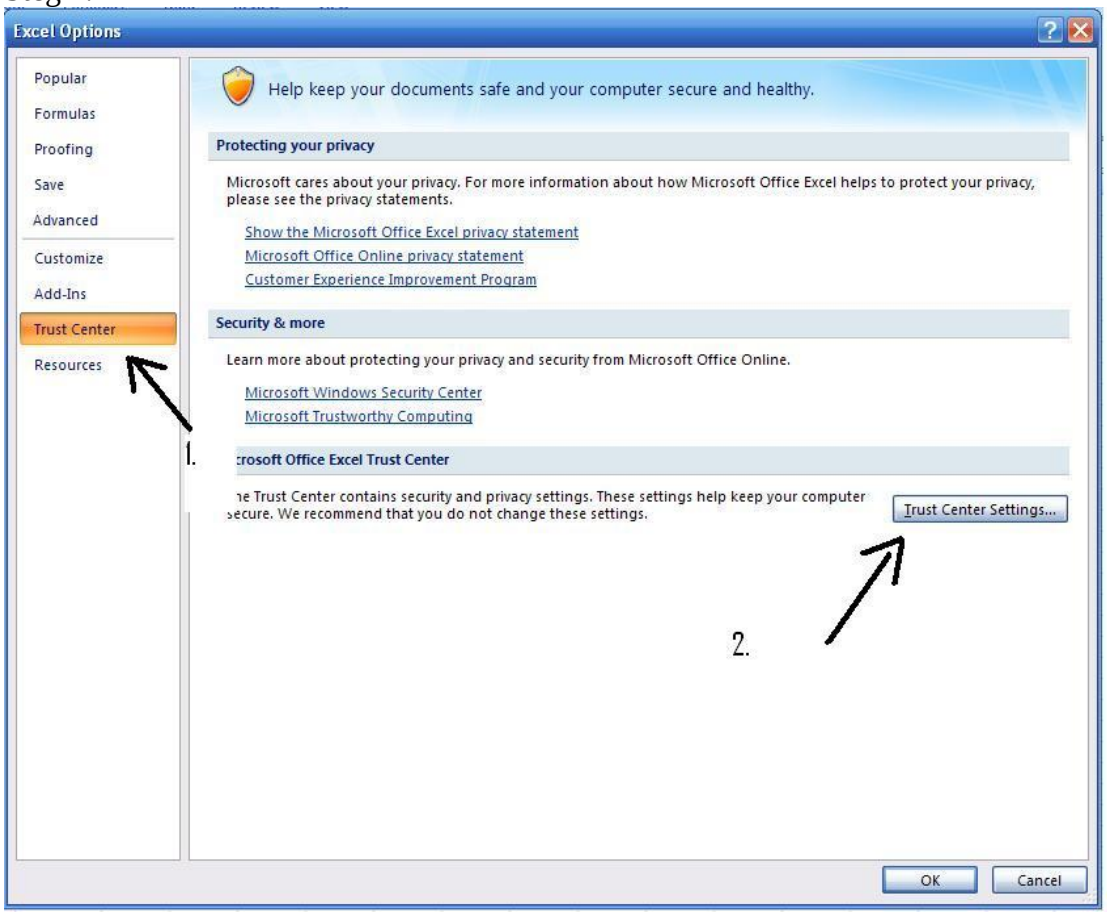
Exekveringsprogrammet (de två som startas tillsammans):
Tryck bara på mellanslag för att starta. MuskelPower stapeln kommer synas på skärmen denna stapel är den mängd muskelkraft som din muskel alstrar under tränning. För att uppnå maximal träning så bör stapeln alltid hållas så hög som möjligt under din träning. Vill du sedan avsluta programmet så behöver du bara stänga bägge programmen.



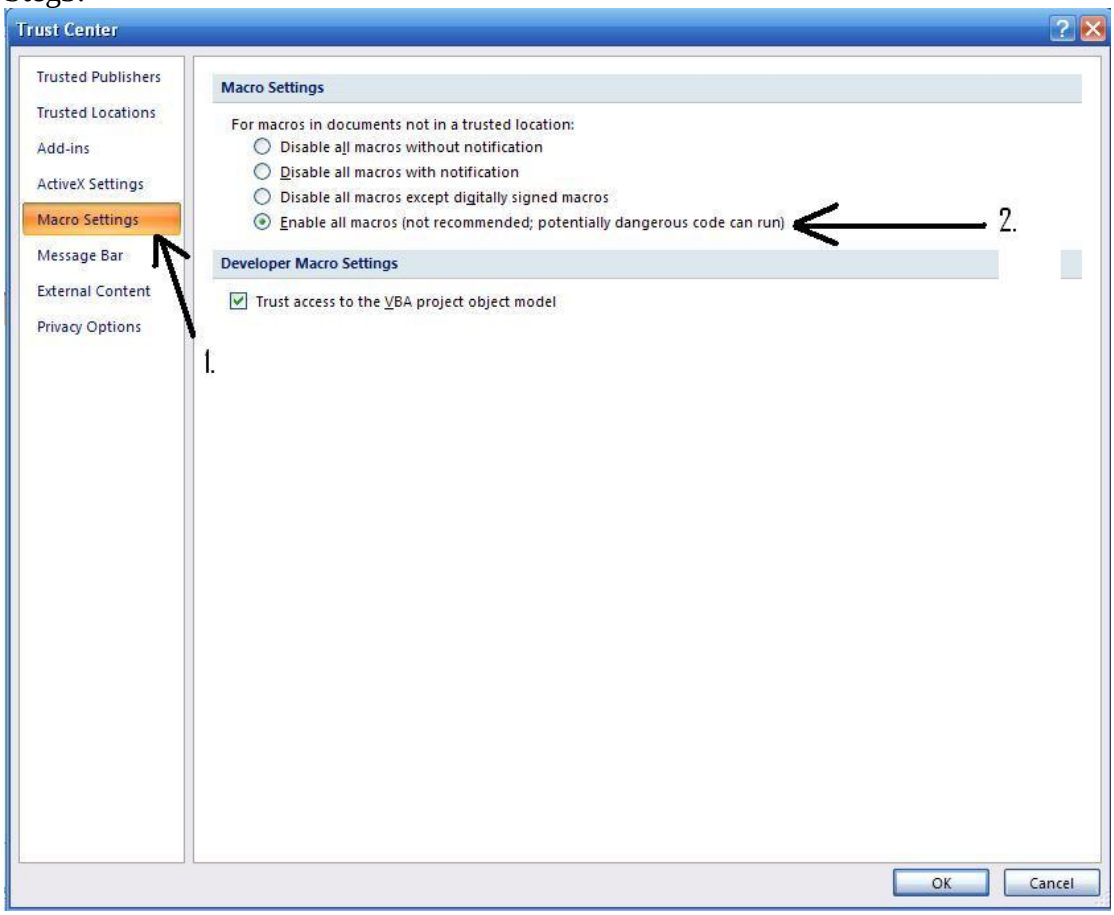
Excelprogrammet (rekommenderas för snabba datorer, kräver även en Excel 2003 eller högre):
För att kunna köra Excelprogrammet så behöver Macron funktionen låsas upp i ditt Excell program. För att göra det så behöver du endast klicka dig igenom instruktionerna från bilderna nedan i den ordning som numren visar.
Steg 1:



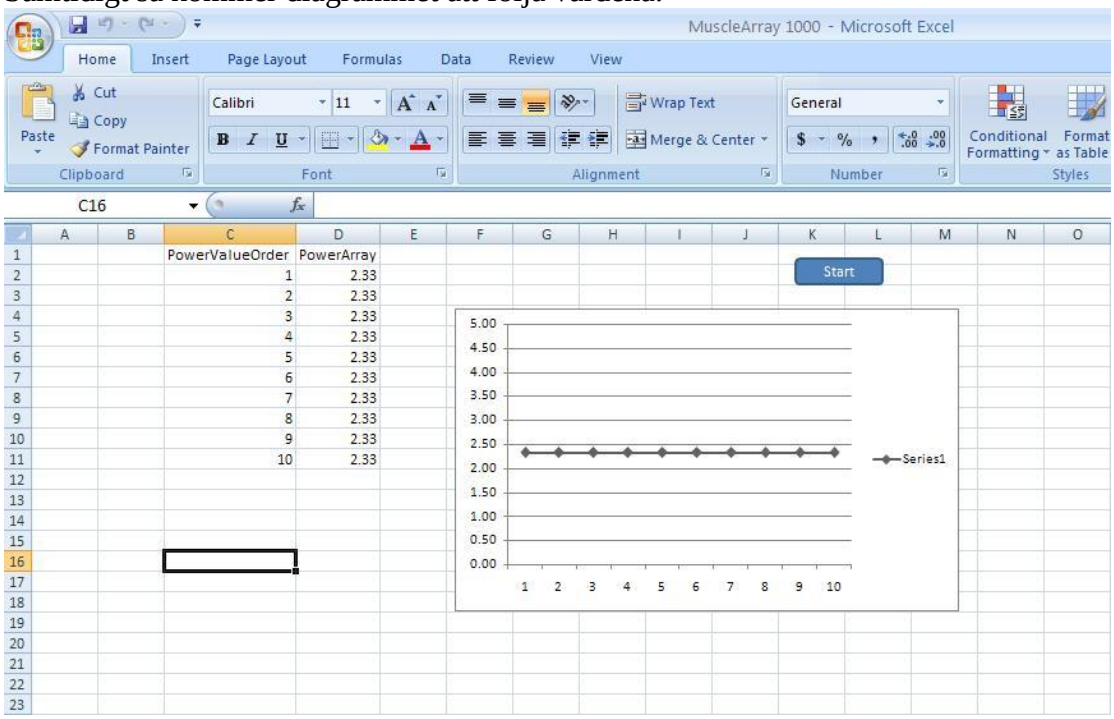
Steg 2:


Excel Options
Popular
Formulas
Proofing
Save
Advanced
Customize
Add-Ins
Trust Center
Resources
Help keep your documents safe and your computer secure and healthy.
Protecting your privacy
Microsoft cares about your privacy. For more information about how Microsoft Office Excel helps to protect your privacy, please see the privacy statements.
Show the Microsoft Office Excel privacy statement
Microsoft Office Online privacy statement
Customer Experience Improvement Program
Security & more
Learn more about protecting your privacy and security from Microsoft Office Online.
Microsoft Windows Security Center
Microsoft Trustworthy Computing
rosoft Office Excel Trust Center
he Trust Center contains security and privacy settings. These settings help keep your computer secure. We recommend that you do not change these settings.
Trust Center Settings...
1.
2.
OK
Cancel

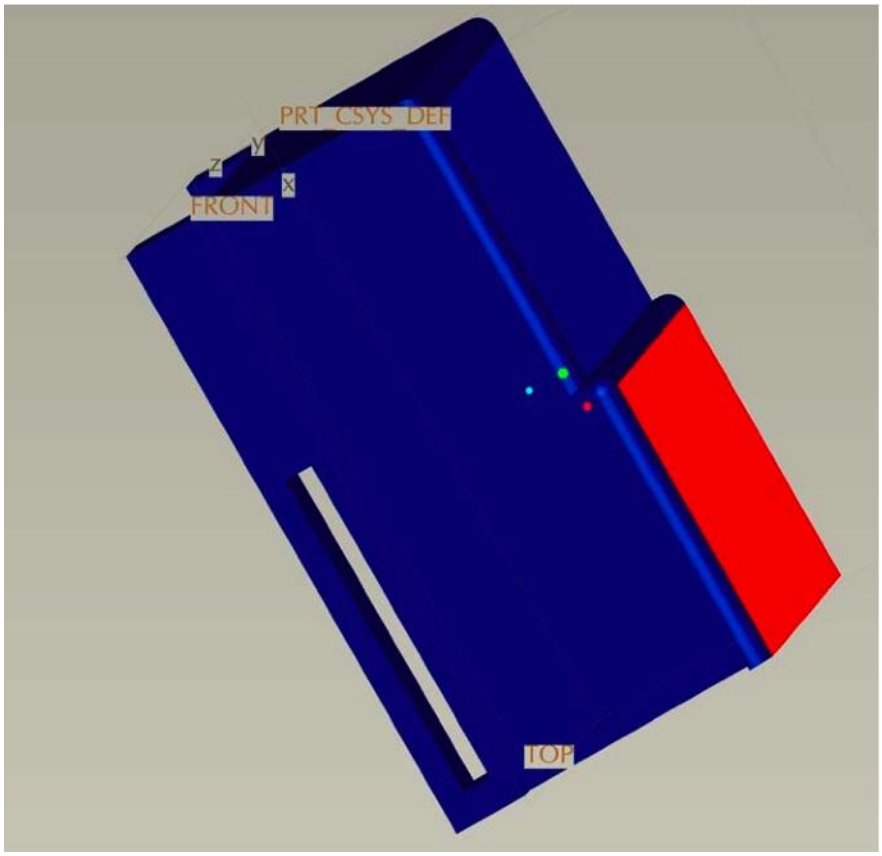
Steg3:



Själva programkörningen bygger sedan på att du trycker på Start knappen för att börja ta in värden samt Escape & End för att sluta ta in värden.
Det nyaste värdet kommer in uppifrån och sedan kommer värdena knuffas neråt.
Samtidigt så kommer diagrammet att följa värdena.



Del 7 – MusclePower Omvandlaren (lådan)



Detta är prototypen som gjorts i Cad Proenginer programmet, tanken är att den ska ha en öppning för att gå att fästa i skärpet, en del som skyddar kretsen och en annan del som går att öppna där du kan byta batterier etc.

Referenser

1. Frekvens kurva från ytelektrod
   www.dataq.com/applicat/articles/an15.htm
2. Datablad för INA128PA
   https://www1.elfa.se/data1/webroot/Z_DATA/07338171.pdf
3. Manual Kap 21 sid. 261 – 270.
4. Manual Kap 19 sid. 199
5. Manual Kap 20 sid. 239 – 260
6. Stefans exempelkod SKK_PIC18_demo.c (1 & 2).
7. Stefans exempelkod SKK_PIC18_EUSART_1.c
8. Henrik Smiths gamla hemsida
   www.kth.sth.se/hst
9. C-hjälpfunktion
10. Inbyggd hjälpfunktion i Excel & Microsoft Visual Basic
11. E_M_Datasheet_OEMSPA_311_331.pdf,
12. Bluetooth.pdf
13. BT_IO_Specification.pdf
14. Analog teknik – Bengt Molin
15. Notchfiltret som testades och var tänkt att användas
    http://www.hobby-electronics.info/projects/TwinTFilter.html