Pulsmätare
Digitala Projekt EITF11

Grupp 5

Sebastian Mattsson, 900611 ain09sma@student.lu.se

Johannes Persson, 880823 ain09jpe@student.lu.se

Handledare: Bertil Lindvall

Innehållsförteckning

Inledning

Under detta projekt har en pulsmätare konstruerats. Pulsmätaren skall fungera likt en pulsmätare på sjukhus, med en grafisk representation av pulsen samt en numerisk framställning som visar antalet pulser per minut. Två knappar används i konstruktionen för att växla mellan det grafiska och numeriska läget. Pulsmätaren skall kunna räkna antalet pulser under 10 sekunder för att sedan omvandla resultatet till pulser per minut. Samtidigt skall pulserna kunna visas i realtid på displayen i det grafiska läget.

Problemformulering

Detta projekt omfattar byggandet av en pulsmätare. Med hjälp av processortyp AT MEGA16, en grafisk display, två knappar och en signalgenerator ska pulsmätaren både kunna visa antalet pulser i minuten samt ge en grafisk representation av pulserna. Detta innebär att det inte får vara för lång fördröjning mellan den verkliga pulsen och pulsen som genereras i den grafiska displayen. Knapparna ska skifta mellan två lägen, grafiskt och numeriskt.

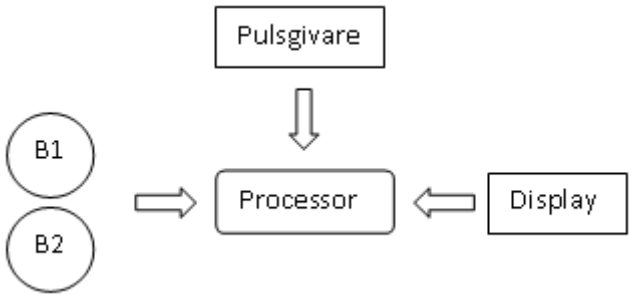
Krav

- Knapp 1 ger grafiskt läge på displayen.
- Knapp 2 ger numeriskt läge på skärmen.
- Det numeriska läget uppdateras 1 gång/ 10 sekunder.
- I det grafiska läget uppdateras grafen i realtid.

Utförande

Design och komponenter

Designen på konstruktionen byggs kring en processor av typ ATMEGA16. Processorn är en 8-bitars AVR Microcontroller med ett programmerbart flashminne på 16 KB och har 32 in- och utpinnar tillgängliga. Till processorn kopplas en grafisk display av typ GDM-12864C, detta för att möjliggöra det grafiska läget. Displayen består av 128 x 64 pixlar och delas upp i två displayhalvor som styrs med hjälp av två controllers av typ S6B0108. Val av vilken sida i displayen som skall skrivas/läsas i avgörs med signalerna Chip Select 1 och Chip Select 2. Pulsgivaren skickar ut pulser i konstant hastighet som registreras av processorn i ett inbyggt timerregister. Pulsernas frekvens kan ändras med hjälp av ett reglage på pulsgivaren. De båda knapparna är av enkel typ och sluter kretsen vid nedtryckning.

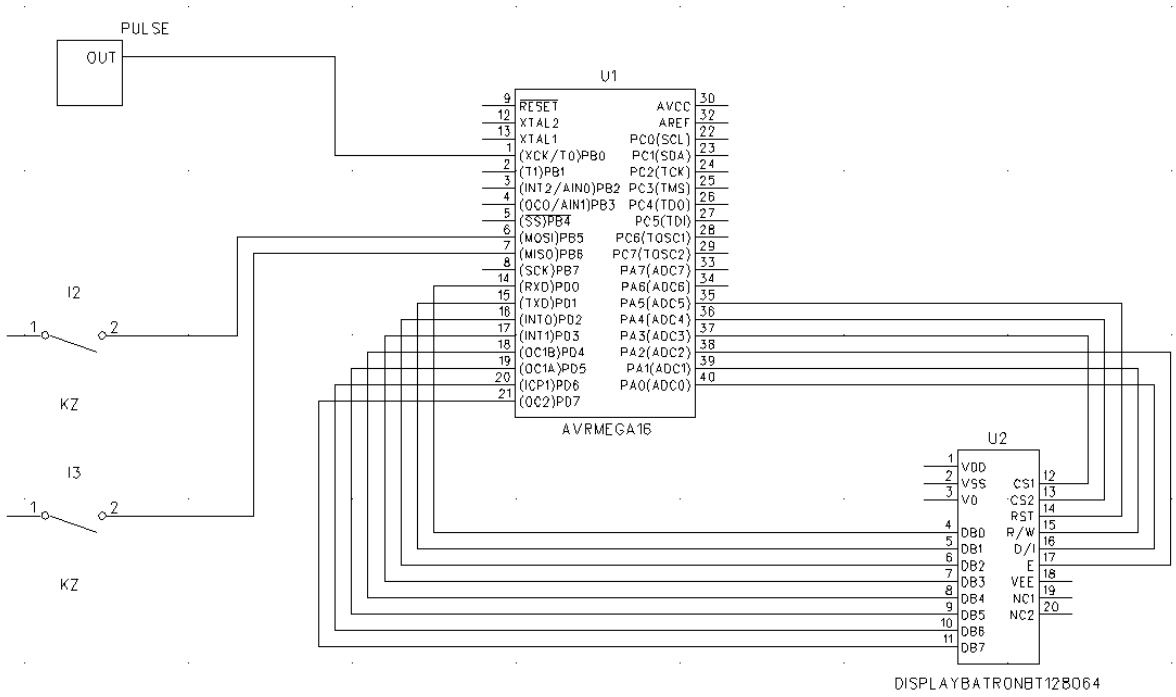


Figur 1

Konstruktion av hårdvara

I figur 2 hittas elschemat för konstruktionen av hårdvaran. Här syns att PORT A används för att koppla ihop processorn med displayen. På denna port skickas kommandon från processorn samtidigt som datan skickas på PORT D. Knapparna och signalen från pulsgeneratorn kopplas till PORT B. Värt att notera är att pulsgeneratorn kopplas till PB0 då denna kan användas som timer men även till ground för att få ett referensvärde på signalen. Knapparna kopplas även de till ground så att signalen för processorn blir låg då knappen är nedtryckt.

Sammankopplingen av de olika komponenterna skedde med hjälp av lödning och virning, där de grova kablarna löddes fast. Valet av kablarnas tjocklek grundades på strömstyrkan som passerar i dem.



Figur 2 - Elschema

Konstruktion av programvara

Efter färdigställandet av hårdvaran påbörjades kodningen av programvara. Detta genomfördes i AVR Studio 4 och programspråket som användes var C-kod. Överföring av programkod från dator till processor och testning av befintlig kod genomfördes med en JTAGICE mkII.

Programmet börjar med att sätta de olika portarna som ingående respektive utgående. Detta bestämmer om processorn ska lyssna efter signaler eller generera egna. För att styra displayen måste de berörda portarna vara utgående medan porten för knappar och pulsgenerator måste vara ingående. Därefter startas en intern och en extern klocka för att möjliggöra räkning av pulser. Den interna klockan används för att kunna beräkna tidsintervall. Eftersom frekvensen på den interna klockan är känd är det lätt att översätta detta till 10 sekunder. Den externa klockan reagerar på lägesförändringar i den inkommande signalen och fungerar därför som en räknare av pulser.

Efter detta startas displayen med kommandot startDisp() som skickar de signaler från processorn till displayen som krävs för uppstart av display. För att rensa displayen på gammal information används clearDisp() som sätter alla pixlar i displayen till värdet 0 (displayen töms). clearDisp() anropar å sin

sida kommandot writeData(…) som används för att skriva i displayen. Därefter väljs position till 1 vilket motsvarar att programmet startar i det numeriska läget.

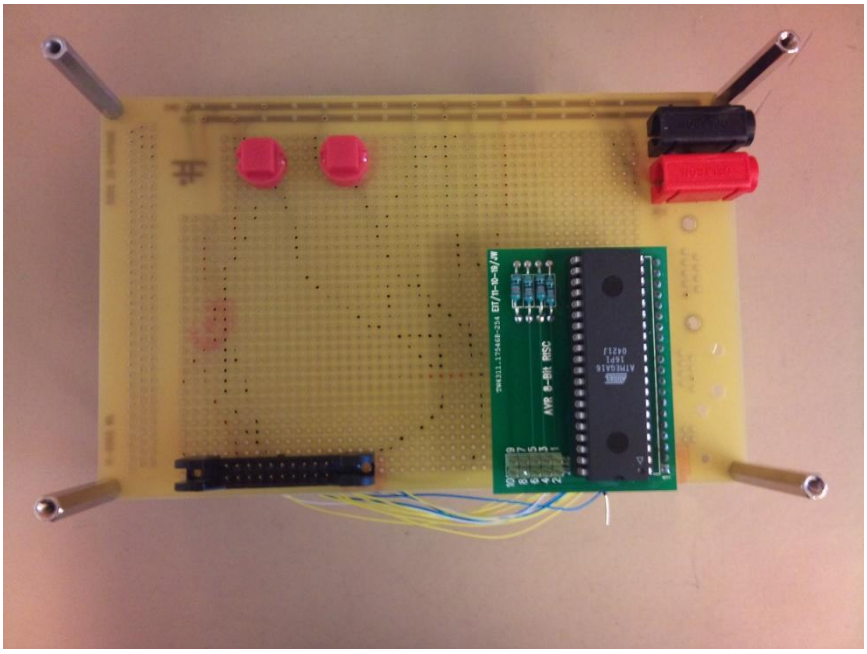
För att processorn ständigt ska uppdateras används sedan en oändlig while-loop. Loopen börjar med att sätta värdena på B-porten för att signalerna från knappar/pulsgivare ska kunna registreras. Därefter anropas kommandot buttonHandler(). Detta kommando kontrollerar om någon knapp är nedtryckt och ändrar position mellan 0 och 1 vilket i sin tur ändrar läget i displayen mellan grafisk och numerisk framställning. I buttonHandler() används bitwise för att sortera ut de pinnar från B-porten som är av intresse. Dessutom används en delay efter varje knapptryckning för att förhindra att knappen studsar mellan lägen.

Nästa steg i while-loopen är att kommandot pulseCounter() anropas. pulseCounter() använder sig av den inre och yttre klockan för att beräkna hur många pulser som tas emot under 10 sekunder och därefter omvandla antalet till pulser per minut.

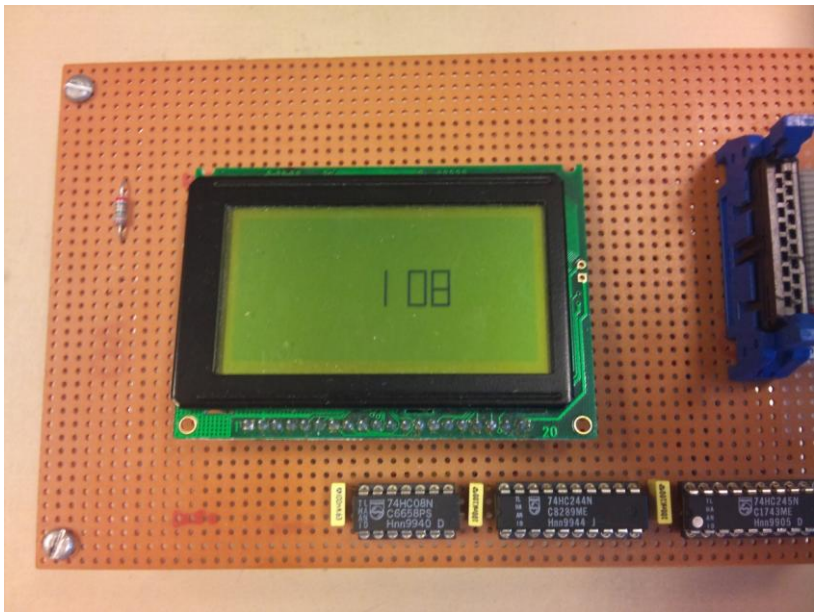
Beroende på om position är 1 eller 0 kommer sedan kommandona displayPulse() respektive displayGraph() att anropas. displayPulse() omvandlar ett int-värde för pulsen till siffror på displayen med hjälpkommandot numberHandler(…). numberHandler(…) anropar i sin tur kommandon för grafisk representation av siffrorna på displayen. displayGraph() använder sig istället av två vektorer, en för varje displayhalva. En etta i vektorn representerar en puls i kolumnen medan en nolla representerar en tom kolumn. Vektorerna gås igenom med en omvänd for-sats där varje puls flyttas ett steg åt höger i displayen med hjälp av metoderna writePulse(…) och removePulse(…). Dessa metoder ritar respektive tar bort en puls ur displayen.

Resultat

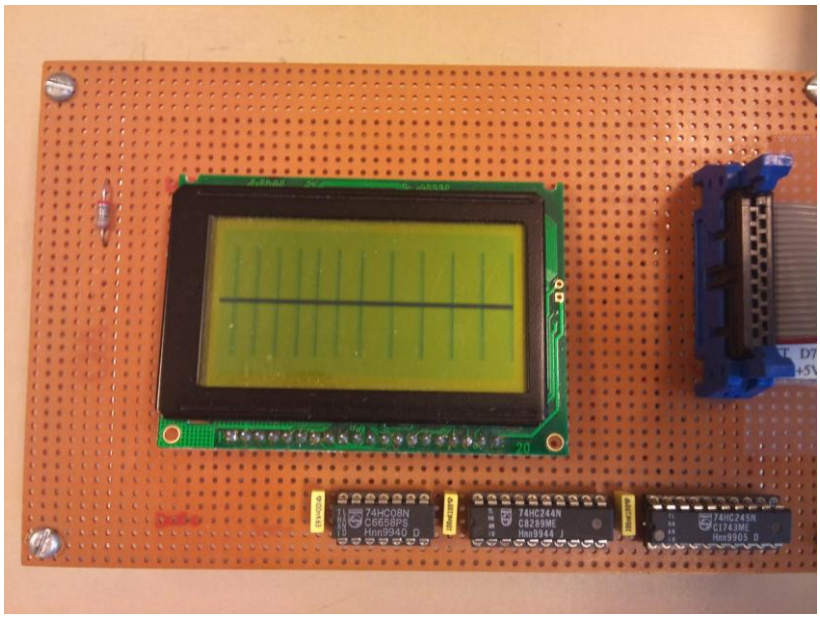
Den färdiga prototypen visas i figur 3 samtidigt som det numeriska läget visas i figur 4 och det grafiska läget i figur 5. De olika lägena nås med de två knapparna. Det numeriska läget visar pulsfrekvensen i slag per minut och det grafiska läget visar avståndet mellan pulserna då pulserna vandrar åt höger.



Figur 3 - Färdig prototyp



Figur 4 - Numeriskt läge



Figur 5 - Grafiskt läge

Slutsats

Projektet resulterade i en fungerande pulsmätare som uppnådde de förutbestämda kraven. Projektet upplevdes till en början som svårbegripligt men då förståelsen ökade i takt med projektets gång blev det lättare allt eftersom. Problem uppstod tidigt då elschemat var svårläsligt och ledde till att kopplingarna mellan processor och display blev felaktiga. När detta problem var åtgärdat vållade C-kodningen nya svårigheter. Då vi inte skrivit i C-kod samt aldrig läst datablad för hårdvara förut blev kommunikationen mellan processor och display nästa hinder. När detta löstes ökade förståelsen för hur hårdvaruprogrammering genomförs och resten av projektet flöt på utan större problem.

Referenslista

AVR Libc 1.6.7 : http://www.eit.lth.se/fileadmin/eit/courses/edi021/avr-libc-user-manual/index.html (120426)

Datablad för GDM-12864C

Digitala projekt – Atmel – ATmega16

Appendix

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/sfr_defs.h>

unsigned short int pulse; // räknar inkommande pulser
unsigned short int position; //vilket läge vi befinner oss i
unsigned short int checkPulse; //håller koll på om det kommer en puls
unsigned short int vector1[64]; //håller koll på pulser i cs1
unsigned short int vector2[64]; //håller koll på pulser i cs2

//hanterar knapparna
void buttonHandler(){
          unsigned short int buttons; // kontrollerar knapparna
          buttons = PINB&0x60;
          switch(buttons){

          case 0x40 :
                    clearDisp();
                    position = 1;
                    _delay_ms (2000);
                    break;

          case 0x20 :
                    clearDisp();
                    position = 0;
                    _delay_ms (2000);
                    writeLine();
                    break;
          }
}
void pulseCounter(){
          if(TCNT1 >= 9766){
                    pulse = TCNT0*6;
                    TCNT0 = 0;
                    TCNT1 = 0;
                    checkPulse = 0;
                    if(position == 1){
                              clearDisp();
                    }
          }
}
void startDisp(){
          PORTA = 0x2C;
          PORTD = 0x3F;
          PORTA = 0x28;

          PORTA = 0x34;
          PORTD = 0x3F;
          PORTA = 0x30;
}
```

```
void writeData(short int chipselect, short int type, short int x, short
int y){
          _delay_us (10);
          if( chipselect == 1){
                    PORTA = 0x2C;
                    PORTD = 64+y; //sätter DB7 och DB6 till 0 1
                    PORTA =   0x28; //skickar y-koordinaten på port D
                    _delay_us (10);
                    PORTA = 0x2C;
                    PORTD = 184+x; //sätter DB7 till DB3 till 1 0 1 1 1
                    PORTA =   0x28; //skickar x-koordinaten på port D
                    _delay_us (10);
                    PORTA = 0x2D;
                    PORTD = type; //väljer värdet på datan
                    PORTA = 0x29; //skickar datan på port D
                    _delay_us (10);
          }
          if(chipselect == 2){
                    PORTA = 0x34;
                    PORTD = 64+y; //sätter DB7 och DB6 till 0 1
                    PORTA =   0x30; //skickar y-koordinaten på port D
                    _delay_us (10);
                    PORTA = 0x34;
                    PORTD = 184+x; //sätter DB7 till DB3 till 1 0 1 1 1
                    PORTA =   0x30; //skickar x-koordinaten på port D
                    _delay_us (10);
                    PORTA = 0x35;
                    PORTD = type; //väljer värdet på datan
                    PORTA = 0x31; //skickar datan på port D
                    _delay_us (10);
          }
}
void clearDisp(){
          for(int i =0; i<64; i++){
                    for(int j =0; j<8; j++){
                              writeData(1,0,j,i);
                              writeData(2,0,j,i);

                    }
          }

}
void writeLine(){
          //ritar linje
          for(int i =0; i<64; i++){
                    writeData(1,0x80,3,i);
                    writeData(1,1,4,i);
                    writeData(2,0x80,3,i);
                    writeData(2,1,4,i);
          }
}
void writePulse(short int chipselect, short int y){
          for(int j =1; j<7; j++){
                    writeData(chipselect,255,j,y);
          }
}
```

```
void removePulse(short int chipselect, short int y){
          for(int j =1; j<3; j++){
                    writeData(chipselect,0,j,y);
          }
          for(int j =5; j<7; j++){
                    writeData(chipselect,0,j,y);
          }
          writeData(chipselect,0x80,3,y);
          writeData(chipselect,0x01,4,y);
}
void one(short int chipselect, short int y){
          for(int j =3; j<5; j++){
                    writeData(chipselect,255,j,y);
          }
}
void two(short int chipselect, short int y){
          for(int i =1; i<7; i++){
                    writeData(chipselect,1,3,y+i);
                    writeData(chipselect,129,4,y+i);
          }
          writeData(chipselect,255,3,y+7);
          writeData(chipselect,255,4,y);
          writeData(chipselect,1,3,y);
          writeData(chipselect,129,4,y+7);
}
void three(short int chipselect, short int y){
          for(int i =1; i<7; i++){
                    writeData(chipselect,1,3,y+i);
                    writeData(chipselect,129,4,y+i);
          }
          writeData(chipselect,255,3,y+7);
          writeData(chipselect,255,4,y+7);
          writeData(chipselect,1,3,y);
          writeData(chipselect,129,4,y);
}
void four(short int chipselect, short int y){
          for(int i =1; i<7; i++){
                    writeData(chipselect,1,4,y+i);
          }
          writeData(chipselect,255,3,y);
          writeData(chipselect,255,4,y+7);
          writeData(chipselect,255,3,y+7);
          writeData(chipselect,1,4,y);
}
void five(short int chipselect, short int y){
          for(int i =1; i<7; i++){
                    writeData(chipselect,1,3,y+i);
                    writeData(chipselect,129,4,y+i);
          }
          writeData(chipselect,255,3,y);
          writeData(chipselect,255,4,y+7);
          writeData(chipselect,1,3,y+7);
          writeData(chipselect,129,4,y);
}
```

```
void six(short int chipselect, short int y){
          for(int i =1; i<7; i++){
                    writeData(chipselect,1,3,y+i);
                    writeData(chipselect,129,4,y+i);
          }
          writeData(chipselect,255,3,y);
          writeData(chipselect,255,4,y+7);
          writeData(chipselect,1,3,y+7);
          writeData(chipselect,255,4,y);
}
void seven(short int chipselect, short int y){
          for(int i =1; i<7; i++){
                    writeData(chipselect,1,3,y+i);
          }
          writeData(chipselect,255,3,y+7);
          writeData(chipselect,255,4,y+7);
          writeData(chipselect,1,3,y);
}
void eight(short int chipselect, short int y){
          for(int i =1; i<7; i++){
                    writeData(chipselect,1,3,y+i);
                    writeData(chipselect,129,4,y+i);
          }
          writeData(chipselect,255,3,y);
          writeData(chipselect,255,4,y+7);
          writeData(chipselect,255,3,y+7);
          writeData(chipselect,255,4,y);
}
void nine(short int chipselect, short int y){
          for(int i =1; i<7; i++){
                    writeData(chipselect,1,3,y+i);
                    writeData(chipselect,1,4,y+i);
          }
          writeData(chipselect,255,3,y);
          writeData(chipselect,255,4,y+7);
          writeData(chipselect,255,3,y+7);
          writeData(chipselect,1,4,y);
}
void zero(short int chipselect, short int y){
          for(int i =1; i<7; i++){
                    writeData(chipselect,1,3,y+i);
                    writeData(chipselect,128,4,y+i);
          }
          writeData(chipselect,255,3,y);
          writeData(chipselect,255,4,y+7);
          writeData(chipselect,255,3,y+7);
          writeData(chipselect,255,4,y);
}
```

```
void numberHandler(short int number, short int pos){
          if(number == 0){
                    zero(1, pos);
          }
          if(number == 1){
                    one(1, pos);
          }
          if(number == 2){
                    two(1, pos);
          }
          if(number == 3){
                    three(1, pos);
          }
          if(number == 4){
                    four(1, pos);
          }
          if(number == 5){
                    five(1, pos);
          }
          if(number == 6){
                    six(1, pos);
          }
          if(number == 7){
                    seven(1, pos);
          }
          if(number == 8){
                    eight(1, pos);
          }
          if(number == 9){
                    nine(1, pos);
          }
}

void displayPulse(){
          unsigned short int temp;
          temp = pulse;
          if(temp >= 100){
                    if(temp/100 ==1){
                              one(1, 0);
                    }
                    if(temp/100 ==2){
                              two(1,0);
                    }
                    temp=temp%100;
                    numberHandler(temp/10, 10);
                    temp=temp%10;
                    numberHandler(temp, 20);
          }
          else if(temp < 100 && temp >= 10){
                    numberHandler(temp/10, 10);
                    temp=temp%10;
                    numberHandler(temp, 20);
          }
          else{
                    numberHandler(temp, 20);
          }
}
void displayGraph(){
```

```
            for(int i =63; i>=0 ; i--){
                        if(TCNT0 > checkPulse){
                                    checkPulse =TCNT0;
                                    vector1[0] =1;
                        }
                        if(vector1[i] == 1 && i!=63){
                                    removePulse(2,i);
                                    writePulse(2, i+1);
                                    vector1[i]=0;
                                    vector1[i+1]=1;
                        }
                        if(vector2[i] == 1 && i!=63){
                                    removePulse(1,i);
                                    writePulse(1, i+1);
                                    vector2[i]=0;
                                    vector2[i+1]=1;
                        }
                        if(vector1[i] == 1 && i==63){
                                    removePulse(2,i);
                                    writePulse(1, 0);
                                    vector1[i]=0;
                                    vector2[0]=1;
                        }
                        if(vector2[i] == 1 && i==63){
                                    removePulse(1,i);
                                    vector2[i]=0;
                        }
                        _delay_ms (1);
            }
}

void main(){
            DDRB = 0x00; //sätter alla portar som ingående
            DDRD = 0xFF; //sätter alla portar som utgående
            DDRA = 0xFF; //sätter alla som utgående
            TCCR1B = 0x05; //startar intern klocka med frekvens 1mhz/1024
            TCCR0 = 0x07; //startar extern klocka

            startDisp();
            clearDisp();
            position = 1;

            while(1){
                        PINB = 0x60; //aktiverar pull-up/down
                        buttonHandler();
                        pulseCounter();
                        if(position == 1){
                                    displayPulse();
                        }
                        if(position == 0){
                                    displayGraph();
                        }
            }
}
```