```
/*
 * Grupp13projekt.c
 *
 * Created: 2015-05-04 15:48:38
 *  Author: digpi13
 */


#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>


#define uchar unsigned char
#define uint unsigned int
#define DDR_SPI DDRB
#define DD_MOSI 5
#define DD_SCK 7
#define DD_SS 4


 int count;                //för interna klockan
 int windspeed;            //senast uppdaterad vindhastighet
 int temperature;    //senast uppdaterad temperatur
 int mode;                 //nuvarande mode vi är i. =0 betyder nonSpeechMode, =1 betyder speechMode
=2 betyder recordMode (fabrikssetting only)

 int namn;                 //namn=0, hälsa ej på användare i speech mode och =1 hälsa.


//****************************************MAINMETOD*********************************************

void main(void)
{
       //initiera allt
        temperature=-1;
        windspeed=-1;
        namn=0;
       DDRA=0b11001100;           //PA7 och PA6 är outputs (MODE-lampa), PA3 och PA2 är outputs
(REC/PLAY-lampa), PA1 är input (ISD4002-interrupt), PA0 är input (termometer)

       DDRB=0b00000000;           //PA0 är input (vindsnurra). SPI initsieras i SPI_MasterInit
       DDRD=0b00000000;           //PD2 (INT0) och PD3 (INT1) är inputs

       GICR=0b11000000;           //INT1 aktiverad
       MCUCR=0b00001101;          //MCUCR=00000111;                 //Av någon jäkla anledning så
funkar detta. ÄNDRA INTE!!//INT0 genererar interupt på any logical change, INT1 genererar interupt på
rising edge. Eftersom knappen (INT1) är känslig är rising edge bättre val än any logical change

       ADMUX=0b01000000;          //VCC är referens
       ADCSRA=0b10000111;         //ACD enable, prescaler divfactorv128

       sei();                     //enable global interrupts

       SPI_MasterInit();
       _delay_ms(100);            //vet inte om vi missar INT0/INT1 under dessa 100ms


       TCCR1A=0b00000000;   //normal settings på allt
       TCCR1B=0b00000010;   //internal clock, prescaler 1024

       TIMSK=0b00000100;    //sätt på overflow interrrupt för TIMER1 (inte för TIMER0)
       count=0;
       TCNT1H=0b00000000;   //nollställ och initsiera
```

```
	TCNT1L=0b00000000;   //nollställ och initsiera


	TCCR0=0b00000111;    //external clock source PIN T0


	if (PIND&(1<<PIND2))       //börja i rätt mode (switchen kan vara i speechmode från början)
	{
		updateData();
		speechMode();
	}
	nonSpeechMode();
 }

//************************************************SLUT PÅ MAINMETOD*********************************


//*****************************UPDATEDATA**************************
 void updateData()
 {
	updateTemperature();
 }


//*****************************NONSPEECHLOOP**************************
 void nonSpeechMode()
 {
	mode=0;
	PORTA=0b01000000;          //RÖD lampa på
	sei();
	while(1)
	{
		updateData();
	}

 }

 //*****************************SPEECHMODE**************************
 void speechMode()
 {
	mode=1;
	PORTA=0b10000000;          //GRÖN lampa på
	sei();
	while(1)
	{
		speak();
		updateData();
	}
 }

 //*****************************SPEAKING**************************
 void speak()
 {

	 if(namn==1)
	 {
		 soundClip(110);           //"Hej"
		 soundClip(230);           //namn
	 }

	//******"Nu blåser det"
	soundClip(120);

	//******värde
```

```
	soundClipNumber(windspeed);

	//******"meter per sekund"
	soundClip(140);

	//******"och temperaturen är"
	soundClip(160);

	//******värde
	soundClipNumber(temperature);

	//******"grader celsius"
	soundClip(180);

}

//*****************************SOUNDCLIP*****************************
void soundClip(uchar addrs)
{
	 SETPLAY(addrs);
	 PORTA=PORTA|0b00001000;    //grön lampa på
	 while(PINA&(1<<PINA1));    //vänta tills EOM-interrupt från ISD4002
	 PORTA=PORTA&0b11110111;    //ingen lampa på
}

 //*****************************SOUNDCLIPNUMBER*****************************

void soundClipNumber(int number)
{
	 if(number==-1)
	 {
		soundClip(100);
	 }
	 else
	 {
		int tiotal = number/10;
		int ental=number-tiotal*10;

		soundClip(tiotal*10);
		soundClip(ental*10);
	 }
}


//*****************************TEMPERATUREvoltage*****************************
 updateTemperature()
{
	uint8_t ch=0b01000000;                    //innan stod det ch=ch&0b01000000; men jag tror det är
onödigt
	ADCSRA|=ch;                                    //Start conversion
	while(!(ADCSRA & (1<<ADIF)));     //vänta på conversion att bli klar
	ADCSRA|=(1<<ADIF);                        //nollställ så vi kan göra en check senare igen
	convertTemperature(ADC);
}

//*****************************CONVERT Voltage till temperaturvärde*****************************
convertTemperature(int unconverted)
{
	temperature=unconverted-584;
}


//*************************CONVERT rotationer till en vindhastighet*****************************
int convertWindspeed(int rotationer)
```

```
 {
       return((int)rotationer/3);
 }

  //************************INTERRUPT Byter MODE*****************************

  ISR(INT0_vect)           //Då any logical change händer för switchen
  {
        if(mode!=2) //Denna knapp är bara aktiv då vi är i nonrecord-mode
        {
              if(mode==0)
              {
                    speechMode();
              }
              nonSpeechMode();
        }
  }


  //************************INTERRUPT knapp för inspelning*************************
ISR(INT1_vect)
{
       if(mode==0)   //Kolla att vi är i nonspeechmode
       {
             mode=2;
             PORTA=0b00000000;    //släck alla lampor

             SETPLAY(200);
             while(PINA&(1<<PINA1));     //vänta tills EOM-interrupt från ISD4002
             _delay_ms(700);

             PORTA=PORTA|0b00000100;           //Röd lampa på

             SETREC(230);
             _delay_ms(1200);
             STOP();

             namn=1;

             PORTA=0b00000000;    //släck alla lampor

             if (PIND&(1<<PIND2))       //skickas till speechMode eller nonSpeechMode;
             {
                   updateData();
                   speechMode();
             }
             nonSpeechMode();

       }

}

//************************Vindsnurra*****************************
ISR(TIMER1_OVF_vect)
{
       if(count==0) {
             TCNT0=0;
       }

       count++;

       if(count==10)
       {
             int rotationer=TCNT0;
```

```
                windspeed=convertWindspeed(rotationer);
                count=0;
        }
}



//***********Initsiera SPI*************************************
void SPI_MasterInit()
{
        DDRB = 0b10110000;
        SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0)|(1<<SPR1)|(1<<DORD);      //SPI Enable, Master select,
f/128, Data Order LSB first

        PORTB = PORTB|(1<<DD_SS);   //kör SS high
}

//***************************SPI-transmit*********************************
void SpiTransmit(uchar data)
{
        SPDR = (uchar)data;
        while(!(SPSR&(1<<SPIF)));   //vänta på transmission ska slutföras
}

//***************cmdSend**********************************
void cmdSend(uint Addr,uchar cmd)
{
        PORTB = PORTB&(~(1<<DD_SS));       //SS low
        SpiTransmit(Addr);                 //send address (use only 8 bit address)
        SpiTransmit(cmd);                  //send command (2 bits of the 10bit address is ignored here)
        PORTB = PORTB|(1<<DD_SS);          //SS high
}

//**********SETPLAY*************************************
void SETPLAY(uchar addrs)
{
        cmdSend(0x00,0x20);  //Power up
        _delay_ms(100);
        cmdSend(addrs,0xe0); //Play from addrs
}

//********SETREC*********************************************
void SETREC(uchar addrs)
{
        cmdSend(0x00,0x20);//Powerup
        _delay_ms(100);
        cmdSend(0x00,0x20);//Powerup
        _delay_ms(100);
        cmdSend(addrs,0xa0);//SETREC
}

//**STOP*****************************************
void STOP(void)
{
        cmdSend(0x00,0x30);
}

//**POWEROFF*************************************************
void PowerOff()
{
        cmdSend(0x00,0x10);
}
```