BoomBoomBot

Digitala projekt | EITF12 | VT19 | Lunds Tekniska Högskola

Författare: Sofia Frändberg, Calle Sandberg & William Bergmark

Handledare: Bertil Lindvall

Datum: 21/05 2019

Abstract

The purpose of this project was to learn how the process of developing a prototype may look like; choosing the appropriate hardware, building the object with it and ultimately developing the software needed to run it. The chosen prototype, the BoomBoomBot, is a primitive sort of sequencer. Its main features consist of being able to record sounds and connect them to specific button on its key-pad, so that the user when in play-mode may push the different buttons (triggering the sounds connected to them) creating his/her own beat.
The process when building the prototype began with drawing the prototypes wiring diagram in accordance to the hardware deemed necessary for it to work properly. The hardware components were thereafter put together, and when this was done the software was developed in order to satisfy the prototypes requirements-specification. The BoomBoomBot was then, after countless hours and enormous effort, completed.

Innehåll

Inledning 3

Kravspecifikation 3

Hårdvara 3

Mjukvara 5

Metod 5

Resultat 6

Diskussion 6

Bilagor 7

- Bilaga 1 - Kopplingsschema 7
- Bilaga 2 - Källkod 8

Inledning

I samband med kursen EITF12 Digitala Projekt har vi valt att utveckla en BoomBoomBot (en avskalad sequencer) som skall kunna spela in och upp ljud utifrån kommandon gjorda från en knappsats på 16 knappar. Syftet med kursen är att illustrera ett industriellt utvecklingsarbete.

Målet med projektuppgiften är att, i en grupp på 2-3 studenter, ta fram en prototyp för vidareutveckling med nödvändig dokumentation. I arbetet ingår framförallt att konstruera, bygga och testa respektive konstruktion.

Kravspecifikation

1. Användaren ska kunna byta till RECORD-mode genom att trycka på avsedd knapp. I detta läge ska en röd lampa lysa.

2. Då BoomBoomBot är satt i RECORD-mode ska användaren kunna spela in valfritt ljud på knappsatsens 12 programmerbara knappar genom att trycka på en knapp och sedan låta. Under tiden som ett ljud spelas in ska även den gula lampan lysa.

3. Användaren ska kunna byta till PLAY-mode genom att trycka på avsedd knapp. I detta läge ska en grön lampa lysa.

4. Då BoomBoomBot är satt i PLAY-mode ska det ljud som är inspelat på/kopplat till den specifika knappen som trycks, av knappsatsens 12 programmerbara knappar, spelas upp. Under tiden som det inspelade ljudet spelas upp ska även den gula lampan lysa.

5. Användaren ska kunna trycka på RESET knappen, och på så sätt radera de ljud som är lagrade på knappsatsens 12 programmerbara knappar. Då knappen trycks ner ska en gul lampa lysa, och sedan släckas.

Hårdvara

Processor
ATMEL Atmega16 är en 8-bit microcontroller-processor. Processorn har 40st pinnar varav 32st kan användas som portar. Modulen kan implementera ett Master/Slave Serial Peripheral Interface (SPI) med hjälp av 4 avsedda portar (MISO, Mosi, SS och SCK) som används för kommunikationen med ISD1740.

ISD1740
ISD1740 är en single chip, multi message record & playback modul. Längden och kvalitén på meddelanden kompromissas beroende på vilken samplingfrekvens och motstånd man använder. ISD1740 är den modul som hanterar inspelningen, lagring samt uppspelning av ljud i samarbete med kommunikation med processorn.

JTAG
En JTAG-modul användes för att exekvera koden som skrivits i Atmel Studio 7. Denna kopplades från datorn in till processorn.

Högtalare
En högtalare användes för att spela upp ljud. Högtalaren var ansluten till en strömkälla och kopplades genom en AUX-sladd till ISD1740.

Mikrofon
En mikrofon användes för att spela in ljud. Denna kopplades direkt till ISD1740.

Knappsats
En knappsats styr de funktioner som BoomBoomBot kan utföra. De kommandon som kan göras via knappsatsen är:

- Record, spelar in ljud på olika platser i minnet (efter val av knapp 1 till 12)
- Play, spelar upp ljud på olika platser i minnet (efter val av knapp 1 till 12)
- Reset, raderar alla inspelade ljud (på alla knappar 1 till 12)

Key-encoder
Kopplar samman knappsatsen med processern på ett sätt som gör att det går att urskilja vilken knapp som aktiverats. Förhindrar också problemet med att knapparna kan ge svängiga signaler som kan vara svåra att tolka.

Lysdioder
Tre stycken lysdioder i färgerna gul, grön och röd lyser i olika färger för att förtydliga vilka funktioner som är aktiva. När BoomBoomBot är i inspelningsläge lyser den röda dioden, och när man tryckt på en knapp som man vill spela in ljud på lyser den gula dioden också medan själva inspelningen sker. Uppspelningsläget fungerar på motsvarande sätt men då lyser den gröna dioden istället för den röda. När alla inspelade ljud återställs lyser endast den gula dioden.

Motstånd
Motstånd reglerade strömmen i kretsen på BoomBoomBot. Motstånd användes i koppling till knappsatsen samt ISD1740.

Kondensator
En kondensator lagrar elektrisk laddning och kopplades till ISD1740.

Mjukvara

Programmeringsspråket som har använts är C och källkoden är skriven i Atmel Studio 7.0. Själva koden är uppbyggd kring en evig while-loop, vilken leder till olika funktioner beroende på vilka av knappsatsens knappar som trycks ner. Genom en JTAG kan koden överföras till processorn och därefter köras med hjälp av hårdvaran.

Metod

1. **Val av produkt:** Första delen av projektet utgjordes av att projektgruppen fattade ett beslut om vilken sorts konstruktion som skulle tas fram, där beslutet föll på att bygga en avskalad Sequencer - BoomBoomBoten. Detta innebär en produkt som spelar in och lagrar olika ljud på olika knappar, där användaren sedan kan byta funktion så att det genom att tryckas på de olika knapparna spelar upp det ljud som lagrats på den.

2. **Planering och kopplingsschema:** i denna fas beslöt projektgruppen vilka krav (se avsnittet "kravspecifikation") som produkten skulle uppfylla, och designade därefter produkten med dessa i beaktning. Detta mynnade slutligen ut (i samråd med handledare då gruppens erfarenhet och kunskap i området var begränsat) i ett kopplingsschema där all ingående hårdvara och hur denna skulle kopplas var tydligt beskrivet (se "Kopplingsschema" i Bilaga 1).

3. **Sammansättning av hårdvaran, fas 1:** då kopplingsschemat var färdigställt och godkänt av kursledaren gick projektgruppen vidare till att sätta samman hårdvaran som skulle utgöra produkten. Gruppen fick då tillgång till en verktygslåda med lödpenna, kopplingstråd, virpistol, avknipsare samt hårdvaran som är närmare beskrivet i avsnittet "Hårdvara". I denna fas sattes allt på kopplingsschemat upp och kopplades såsom beskrivet, utom micken och högtalaren som är kopplade till ISD1740, då det ansågs mest effektivt att vänta med dessa tills det var färdigställt att det fanns kommunikation mellan processorn och ISD1740.

4. **Kodning fas 1:** Då hårdvaran var färdigställd övergick projektgruppen till att programmera den mjukvara som skulle ingå i produkten på Atmel Studio 7 i programmeringsspråket C. Först gjordes ett enklare skelett över vad som skulle krävas av mjukvaran. Sedan testades kopplingen från processorn via key-encodern fram till knappsatsen, genom att anvisa produktens led-lampor till att lysa på olika vis då olika knappar trycktes. Under denna fas felsökte gruppen mycket och tog stor hjälp av programmets debugger.

5. **Sammansättning av hårdvaran, fas 2:** efter att projektgruppen försäkrat sig om att första delen av hårdvaran (key-encoder, knappsats, led-lampor och processor) kommunicerade

som avsett, sammankopplas mikrofon och högtalare enligt kopplingsschemat. Produktens hårdvara var efter denna fas färdigställd.

6. **Kodning fas 2:** då hårdvaran var sammanställd kodades kommunikationen mellan processorn och ISD1740, för att ge den för projektet nödvändiga egenskaper. Detta gjordes genom att kolla på respektive modulers datablad, testning och felsökning tills produkten betedde sig som önskat. Då detta fungerade som önskat var produkten färdigställd.

Resultat

BoomBoomBot resulterade i en fungerande prototyp som uppfyllde de krav som fastställts i början av projektet. På grund av att den valda mikrofonen inte upptog särskilt finkänsliga ljud (utan enbart knackningar, “blåsljud” och rena skrik, alltså bara de starkaste av tryckändringar) blev den lite mer begränsad som sequencer, då idén var att alla slags ljud skulle kunna användas för att komponera egna unika kombinationer av beats. Sättet som källkoden utformats på innebar även att ljud spelades in i förutbestämda tidsspann, där endast en knapp kunde aktiveras åt gången, vilket i sin tur gjorde att sequencern inte kunde byta mellan de olika ljuden särskilt snabbt. Detta ledde till att BoomBoomBot som sequencer blev ganska undermålig.

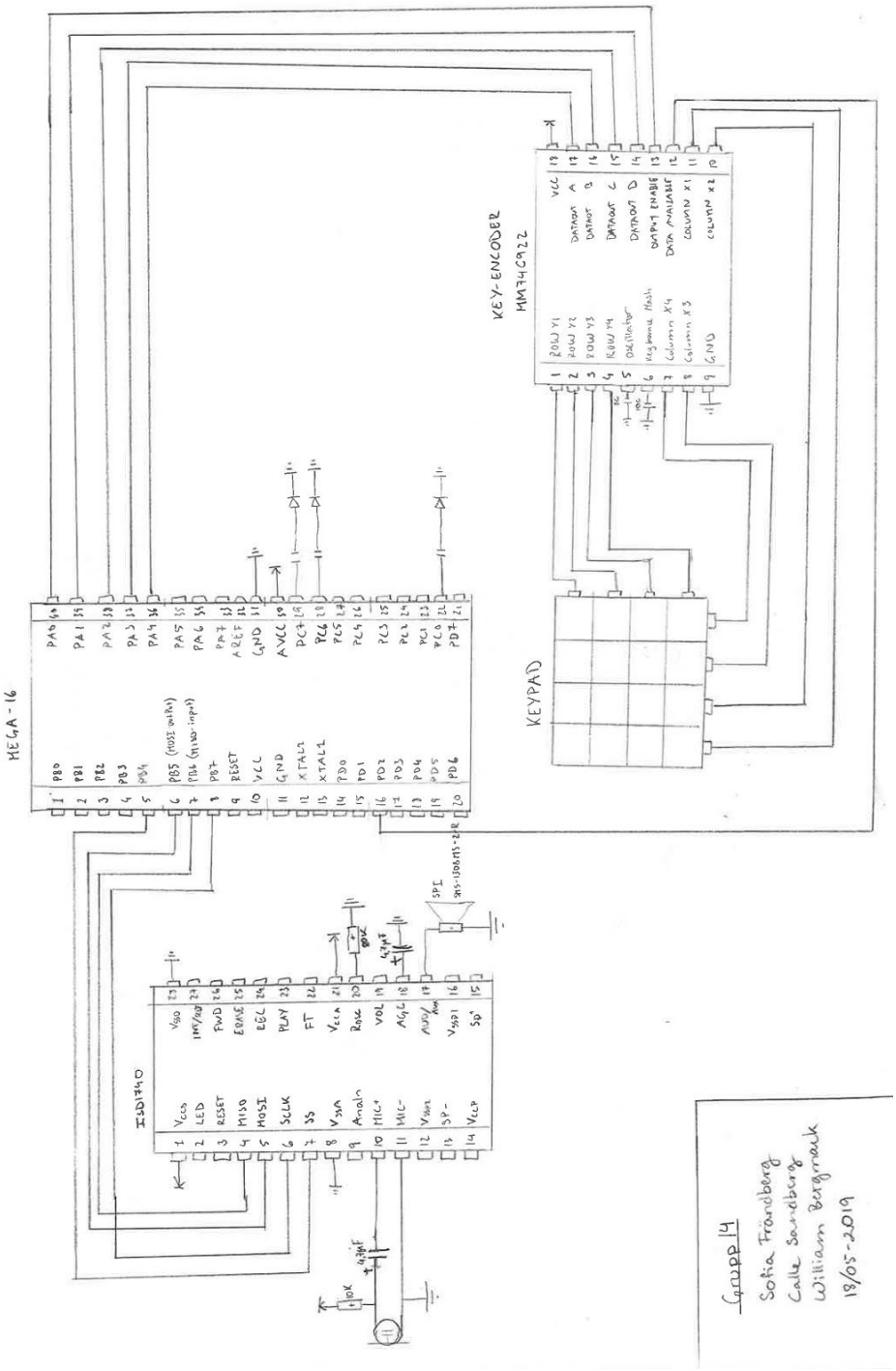
Diskussion

Hårdvaran funkade väl då knappsatsen, key-encodern och led-belysningen på ett enkelt sätt gick att koppla samman tidigt i processen. Dessa utgjorde därför en grund för våra metoder i mjukvaran och gjorde felsökningen smidig. Att förstå hur kommunikationen med ISD1740 fungerade var dock mer komplext, och gjorde att programmeringsprocessen var den del som tog längst tid under prototypens utveckling.

I en vidareutveckling av BoomBoomBot skulle det vara intressant att utveckla mjukvaran så att det bara spelades in ljud så länge som användaren ansåg det nödvändigt. Något som även skulle göra BoomBoomBoten mer åtråvärd skulle vara ifall man utvecklade den så att ljudet från flera olika knapptryckningar skulle kunna spelas samtidigt, så att prototypen därmed kunde efterlikna en sequencer bättre. Något nödvändigt för nästa version skulle även vara att byta mikrofonen för att möjliggöra inspelning av mer högkvalitativt ljud.

Bilagor

Bilaga 1 - Kopplingsschema



Bilaga 2 - Källkod

```
/*
 * GccApplication1.c
 *
 * Created: 2019-04-11 17:28:01
 * Author: Calle Sandberg, Sofia Frändberg, William Bergmark
 */

#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

//VARIABLER
#define SS PB4
#define MOSI PB5
#define MISO PB6
#define SCK PB7
#define LEDRED PC0
#define LEDGRE PC6
#define LEDYEL PC7

#define PU 0x01
#define STOP 0x02
#define CLR_INT 0x04
#define RD_STATUS 0x05
#define RD_APC 0x44
#define WR_APC2 0x65
#define FWD 0x48
#define SET_PLAY 0x80
#define SET_REC 0x81
#define SET_ERASE 0x82

volatile uint16_t pushed;
volatile uint8_t first;
volatile uint8_t second;
volatile uint8_t third;
volatile uint8_t fourth;
volatile uint8_t fifth;
volatile uint8_t sixth;
```

```
volatile uint8_t scrap;

char val;
int red;
int green;
int yellow;

//METODER
void init_spi();
void init_encoder();
void led_init();
void yellowOn();
void yellowOff();
void greenOn();
void greenOff();
void redOn();
void redOff();
void selectKey();
void reset();
void play();
void record();
void isd_clear_int();
void isd_set_APC();
void isd_power_up();
void isd_read_status();
void isd_set_rec(uint8_t startbyte1, uint8_t startbyte2, uint8_t slutbyte1 ,uint8_t slutbyte2,
uint8_t slutbyte3);
void isd_set_play(uint8_t startbyte1, uint8_t startbyte2, uint8_t slutbyte1 ,uint8_t slutbyte2,
uint8_t slutbyte3);
void send_two_bytes(uint8_t byte0, uint8_t byte1);
void send_three_bytes(uint8_t byte0, uint8_t byte1, uint8_t byte2);
void send_four_bytes(uint8_t byte0, uint8_t byte1, uint8_t byte2, uint8_t byte3);
void send_seven_bytes(uint8_t byte0, uint8_t byte1, uint8_t byte2, uint8_t byte3, uint8_t
byte4,uint8_t byte5, uint8_t byte6);

//MAIN
int main(void){
   led_init();
   init_spi();
   init_encoder();
   isd_power_up();
         isd_set_APC();
   sei();
```

```
    while (1){
        if(pushed == 1){
            selectKey();
            pushed = 0;
        }
    }
}

//INTERRUPTION
ISR(INT0_vect){
    val = PINA;
    pushed = 1;
}

//INIT-METODER
void init_spi() {
    DDRB = (1<<SCK)|(1<<MOSI)|(1<<SS);
    PORTB= (1<<SCK)|(1<<SS);
    SPCR = (1<<SPE)|(1<<MSTR)|(1<<DORD)|(1<<SPR0)|(1<<CPOL)|(1<<CPHA);
}

void init_encoder(){
    GICR = 1<<INT0;
    MCUCR = 1<<ISC01 |1<<ISC00;
    DDRA = 0b00000001;
    PORTA = 0b00000000;
}

void led_init(){
    DDRC |= (1<<LEDYEL);
    DDRC |= (1<<LEDGRE);
    DDRC |= (1<<LEDRED);
}

//LED-METODER
void yellowOn(){
    PORTC |= (1<<LEDYEL);
    yellow = 1;
}

void yellowOff(){
    PORTC &=~(1<<LEDYEL);
```

```
   yellow = 0;
}

void greenOn(){
   PORTC |= (1<<LEDGRE);
   green = 1;
}

void greenOff(){
   PORTC &=~(1<<LEDGRE);
   green = 0;
}

void redOn(){
   PORTC |= (1<<LEDRED);
   red = 1;
}

void redOff(){
   PORTC &=~(1<<LEDRED);
   red = 0;
}

//KNAPP-METODER
void selectKey(){
   if(val==0xf8){ //Button1 (1, 1), bytes: 0x10 -> 0x29
      if (green == 1) {
         //spela upp ljudet som är sparat i knappen
         isd_set_play(0x10,0x00,0x29,0x00,0x00);
      }
      if (red == 1) {
         //spela in ett ljud på vald knapp
         isd_set_rec(0x10, 0x00, 0x29, 0x00, 0x00);
      }
   }

   if(val==0xe8){ //Button2 (1, 2), bytes: 0x2A -> 0x43
      if (green == 1) {
               //spela upp ljudet som är sparat i knappen
               isd_set_play(0x2A,0x00,0x43,0x00,0x00);
      }
      if (red == 1) {
               //spela in ett ljud på vald knapp
```

```
            isd_set_rec(0x2A,0x00,0x43,0x00,0x00);
    }
}

if(val==0xf0){ //Button3 (1, 3), bytes: 0x44 -> 0x5D
    if (green == 1) {
            //spela upp ljudet som är sparat i knappen
            isd_set_play(0x44,0x00,0x5D,0x00,0x00);
    }
    if (red == 1) {
            //spela in ett ljud på vald knapp
            isd_set_rec(0x44,0x00,0x5D,0x00,0x00);
    }
}

if(val==0xe0){ //Button4 (1, 4), bytes: 0x5E -> 0x77
    if (green == 1) {
            //spela upp ljudet som är sparat i knappen
            isd_set_play(0x5E,0x00,0x77,0x00,0x00);
    }
    if (red == 1) {
            //spela in ett ljud på vald knapp
            isd_set_rec(0x5E,0x00,0x77,0x00,0x00);
    }
}

if(val==0xfc){ //Button5 (2, 1), bytes: 0x78 -> 0x91
    if (green == 1) {
            //spela upp ljudet som är sparat i knappen
            isd_set_play(0x78,0x00,0x91,0x00,0x00);
    }
    if (red == 1) {
            //spela in ett ljud på vald knapp
            isd_set_rec(0x78,0x00,0x91,0x00,0x00);
    }
}

if(val==0xec){ //Button6 (2, 2), bytes: 0x92 -> 0xAB
    if (green == 1) {
            //spela upp ljudet som är sparat i knappen
            isd_set_play(0x92,0x00,0xAB,0x00,0x00);
    }
    if (red == 1) {
```

```
            //spela in ett ljud på vald knapp
            isd_set_rec(0x92,0x00,0xAB,0x00,0x00);
    }
}

if(val==0xf4){ //Button7 (2, 3), bytes: 0xAC -> 0xC5
    if (green == 1) {
            //spela upp ljudet som är sparat i knappen
            isd_set_play(0xAC,0x00,0xC5,0x00,0x00);
    }
    if (red == 1) {
            //spela in ett ljud på vald knapp
            isd_set_rec(0xAC,0x00,0xC5,0x00,0x00);
    }
}

if(val==0xe4){ //Button8 (2, 4), bytes: 0xC6 -> 0xDF
    if (green == 1) {
            //spela upp ljudet som är sparat i knappen
            isd_set_play(0xC6,0x00,0xDF,0x00,0x00);
    }
    if (red == 1) {
            //spela in ett ljud på vald knapp
            isd_set_rec(0xC6,0x00,0xDF,0x00,0x00);
    }
}

if(val==0xfa){ //Button9 (3, 1), bytes; 0xE0 -> 0xF9
    if (green == 1) {
            //spela upp ljudet som är sparat i knappen
            isd_set_play(0xE0,0x00,0xF9,0x00,0x00);
    }
    if (red == 1) {
            //spela in ett ljud på vald knapp
            isd_set_rec(0xE0,0x00,0xF9,0x00,0x00);
    }
}

if(val==0xea){ //Button10 (3, 2), bytes: 0xFA -> 0x113
    if (green == 1) {
            //spela upp ljudet som är sparat i knappen
            isd_set_play(0xFA,0x00,0x13,0x01,0x00);
    }
```

```
   if (red == 1) {
            //spela in ett ljud på vald knapp
            isd_set_rec(0xFA,0x00,0x13,0x01,0x00);
   }
}

if(val==0xf2){ //Button11 (3, 3), bytes: 0x114 -> 0x12D
   if (green == 1) {
            //spela upp ljudet som är sparat i knappen
            isd_set_play(0x14,0x01,0x2D,0x01,0x00);
   }
   if (red == 1) {
            //spela in ett ljud på vald knapp
            isd_set_rec(0x14,0x01,0x2D,0x01,0x00);
   }
}

if(val==0xe2){ //Button12 (3, 4), bytes: 0x12E -> 0x147
   if (green == 1) {
            //spela upp ljudet som är sparat i knappen
            isd_set_play(0x2E,0x01,0x47,0x01,0x00);
   }
   if (red == 1) {
            //spela in ett ljud på vald knapp
            isd_set_rec(0x2E,0x01,0x47,0x01,0x00);
   }
}

if(val==0xfe){ //WILDCARD (4, 1)

}

if(val==0xee){ //RESET (4, 2)
   reset();
}

if(val==0xf6){ //PLAY (4, 3)
   play();
}

if(val==0xe6){ //RECORD (4, 4)
   record();
}
```

```
}

void reset(){
   yellowOn();
   greenOff();
   redOff();
        send_seven_bytes(SET_ERASE, 0x00, 0x10, 0x00, 0x47, 0x01, 0x00);
        isd_clear_int();
   _delay_ms(1500);
   yellowOff();
}

void play(){
   greenOn();
   yellowOff();
   redOff();
}

void record(){
   redOn();
   yellowOff();
   greenOff();
}

//ISD-METODER
void isd_power_up(){
   send_two_bytes(PU, 0x00);
}

void isd_RD_APC(){
   send_four_bytes(RD_APC, 0x00, 0x00, 0x00);
}

void isd_set_APC(){
        send_three_bytes(WR_APC2, 0b11101001, 0b00000100);
}

void isd_clear_int(){
   send_two_bytes(CLR_INT, 0x00);
}

void isd_read_status() {
   send_three_bytes(RD_STATUS, 0x00, 0x00);
```

```
}

void isd_stop(){
   send_two_bytes(STOP, 0x00);
}

void isd_set_play(uint8_t startbyte1, uint8_t startbyte2, uint8_t slutbyte1 ,uint8_t slutbyte2,
uint8_t slutbyte3){
        isd_clear_int();
        isd_stop();
        isd_read_status();
        _delay_ms(100);
        yellowOn();
   send_seven_bytes(SET_PLAY, 0x00, startbyte1, startbyte2, slutbyte1, slutbyte2, 0x00);
        isd_read_status();
        _delay_ms(5000);
        yellowOff();
}

void isd_set_rec(uint8_t startbyte1, uint8_t startbyte2, uint8_t slutbyte1 ,uint8_t slutbyte2,
uint8_t slutbyte3){
        isd_clear_int();
        isd_read_status();
        send_seven_bytes(SET_ERASE, 0x00, startbyte1, startbyte2, slutbyte1, slutbyte2,
0x00);
        _delay_ms(100);
        isd_clear_int();
        yellowOn();
   send_seven_bytes(SET_REC, 0x00, startbyte1, startbyte2, slutbyte1, slutbyte2, 0x00);
        isd_read_status();
        _delay_ms(5000);
        isd_stop();
        yellowOff();
}

//SEND-METODER
void send_two_bytes(uint8_t byte0, uint8_t byte1) {
   PORTB &= ~(1<<SS);
   //Load data into the buffer
   SPDR = byte0;
   //Wait until transmission complete
   while(!(SPSR & (1<<SPIF) ));
   first=SPDR;
```

```
    SPDR = byte1;
    //Wait until transmission complete
    while(!(SPSR & (1<<SPIF) ));
    second=SPDR;
    PORTB |= (1<<SS);
}

void send_three_bytes(uint8_t byte0, uint8_t byte1, uint8_t byte2) {
    PORTB &= ~(1<<SS);
    //Load data into the buffer
    SPDR = byte0;
    //Wait until transmission complete
    while(!(SPSR & (1<<SPIF) ));
    first=SPDR;
    SPDR = byte1;
    //Wait until transmission complete
    while(!(SPSR & (1<<SPIF) ));
    second=SPDR;
    SPDR = byte2;
    //Wait until transmission complete
    while(!(SPSR & (1<<SPIF) ));
    third=SPDR;
    PORTB |= (1<<SS);
}

void send_four_bytes(uint8_t byte0, uint8_t byte1, uint8_t byte2, uint8_t byte3) {
    PORTB &= ~(1<<SS);
    //Load data into the buffer
    SPDR = byte0;
    //Wait until transmission complete
    while(!(SPSR & (1<<SPIF) ));
    first=SPDR;
    SPDR = byte1;
    //Wait until transmission complete
    while(!(SPSR & (1<<SPIF) ));
    second=SPDR;
    SPDR = byte2;
    //Wait until transmission complete
    while(!(SPSR & (1<<SPIF) ));
    third=SPDR;
    SPDR = byte3;
    //Wait until transmission complete
    while(!(SPSR & (1<<SPIF) ));
```

```
   fourth=SPDR;
   PORTB |= (1<<SS);
}

void send_seven_bytes(uint8_t byte0, uint8_t byte1, uint8_t byte2, uint8_t byte3, uint8_t
byte4, uint8_t byte5, uint8_t byte6) {
        PORTB &= ~(1<<SS);
        //Load data into the buffer
        SPDR = byte0;
        //Wait until transmission complete
        while(!(SPSR & (1<<SPIF) ));
        first=SPDR;
        SPDR = byte1;
        //Wait until transmission complete
        while(!(SPSR & (1<<SPIF) ));
        second=SPDR;
        SPDR = byte2;
        //Wait until transmission complete
        while(!(SPSR & (1<<SPIF) ));
        third=SPDR;
        SPDR = byte3;
        //Wait until transmission complete
        while(!(SPSR & (1<<SPIF) ));
        fourth=SPDR;
        SPDR = byte4;
        while(!(SPSR & (1<<SPIF) ));
        fifth=SPDR;
        SPDR = byte5;
        while(!(SPSR & (1<<SPIF) ));
        sixth=SPDR;
        SPDR = byte6;
        //Wait until transmission complete
        while(!(SPSR & (1<<SPIF) ));
        scrap=SPDR;
        PORTB |= (1<<SS);
}
```