Innehållsförteckning

Abstract

In modern day society the need for controlling the amount of technology is needed. This project goes under the topic: "The intelligent home". With help from a Radio Frequency Identification (RFID) receiver/reader a person can be let in, and personal settings can be applied. Settings such as lighting and room temperature.

The application is based on a Motorola 68008 microprocessor, external memories and the RFID receiver/reader. An EEPROM is used as a database to store the personal settings for each user. To change the settings a 16 digit number keypad is applied to the system, and to visualise them a dot-matrix lcd is used.

1. Inledning

Projektet går under rubriken "Det intelligenta hemmet", och med mängden teknik som finns i dagens hem behövs en knytpunkt som styr allt. Projektet går därför ut på att om en familjemedlem kommer hem så ska en läsare känna av vem det är, och ställa in hemmet efter personens tyckte och smak. Till exempel kan ett mediacenter startas med personens favorit spellista, temperaturen i personens rum anpassas och rätt belysning ska vara tänd.

Som ett steg i den riktningen har vi valt att titta närmare på läsaren i det här fallet en Radio Frequency Identification (RFID) läsare, men även lagringen av den individuella informationen/inställningarna för var användare.

Kravspecifikation för applikationen kan ses nedan:

- Display där tydlig information ska framgå, såsom vilken användare som anländer, logiskt menysystem för att kunna ändra inställningar.
- Utvärdering av kortläsaren SonMicro SM3005.
- Databas för användare, databasen skall vara i storlek med ett hushåll, ungefär 10 användare. Databasen får inte gå förlorad även om konstruktionen blir spänningslös.
- Tangentbord så att inställningar kan ändras för var användare.

I mån av tid:

- Lägga till nya användare med hjälp av kortläsaren och tangentbordet.
- Djupare utvärdering av kortläsaren, protokoll och eventuell möjlighet att skriva till korten.

2. Komponenter

Komponenterna valdes utifrån prestandakrav som sattes i början av projektet. Såsom storlek på display och minnes storlek.

Bygget delades upp i olika delar, logik, kontrolldel och kortdel. En ny del påbörjades inte förrän föregående del var funktionsduglig.

2.1. Kontrolldel

Kontrolldelen kan ses som en dator, delen innefattar bland annat processor, minne, display och ett mindre tangentbord.

2.1.1. Processor

Till processor valdes Motorola 68008 med 48 pinnar. Denna processor har inget inbyggt minne utan externt program- och ramminne krävs. Enheterna vi använder måste separeras av buffertar eller kunna gå i tri-state för att undvika kollision på databussen. 3-state innebär att enheterna blir högimpediva (osynliga), och det får samma effekt som att skilja dem åt med buffertar.

2.1.2. Minne

Till applikationen krävs tre typer av minne. Ramminne processorns arbetsminne, Eprom för programmet och ett EEprom för användarhantering och databas. Eprom är enbart ett läsbart minne och ett EEprom är ett läs- och skrivbart minne. Anledningen till att två separata minnen används (Eprom och EEprom) är att man vill inte kunna på något sätt skriva över programkoden som styr det inbyggda systemet.

- 27C64 8K x 8 Bit Parallell EPROM, valdes till programminnet.
- 6264 8K x 8 Parallell Bit CMOS SRAM, valdes som processorns arbetsminne.
- AT28C64B 8K x 8 Bit Parallellt EEPROM, till databasen.

2.1.3. LCD-display

Behovet av en display är uppenbar, men kraven är dock inte höga. En alfanumerisk display med 2 x 16 tecken är fullt tillräcklig.

- SHARP Dot-Matrix LCD Units Alfanumerisk teckendisplay.

2.1.4. Encoder

Eftersom användaren skall kunna ändra inställningar behövs ett tangentbord. Ett 16 knappars tangentbord valdes då det anses fullt tillräckligt. Tangentbordet kopplas till en encoder (i det här fallet en 54C922). Encoderns uppgift är att indikera när en knapp tryckts ner och att lägga denna information på sin databus. Eftersom encodern registrerar var gång en knapp tryckts ner, oftast ett flertal gånger per knapptryck, behövs en latch som sparar värdet och nollställs då processorn är klar.

- 54C922 16-Key Encoder.
- 74HC74 Dual D Flip-Flop Set/Reset, latch.

2.1.5. Reset-knapp

En reset-knapp behövs för att kunna nollställa systemet, om ett fel skulle inträffa. Till exempel läsningen mellan UART och kortläsaren slutar fungera. En brytar kopplas till både processorns RESET och UARTens RESET.

2.2. Kortdel

2.2.1. Kortläsare

Kortläsaren är en så kallad Radio Frequency Identification (RFID) läsare. RFID är en teknik för att lagra och läsa information från kort från ett litet avstånd från en kombinerad mottagar/läsar-del. Allmänt finns det två typer av RFID-kort aktiva och passiva. Det aktiva kortet har en spänningskälla ansluten, medan det passiva kortet får sin spänning inducerad från mottagar/läsar-delens spoles magnetfält.

Aktiva kort används oftast i bilnycklar som stöldskydd eller i containrar för att identifiera dem på avstånd. Aktiva kort är dock dyra och de passiva korten används oftare eftersom de är förhållandevis billiga och enkla att implementera. Passiva kort återfinns till exempel bak på böcker i bibliotek och på de nya EU-passen. I det här projektet används den enklaste modellen med passiva kort.

Läsaren är i sin tur kopplad via en standard seriell databus RS232 till en UART. Läsaren är av märket SonMicro med modellbetäckningen SM3005.

- SonMicro SM3005 RFID Development Kit

2.2.2. UART

Universal Asynchronus Reciever/Transmitter (UART) används för att omvandla parallell data från processorn till seriell data till kortläsaren och vice versa. Den vanligaste UARTen är modell 16550 som ofta kan hittas i äldre datorer med serriell databus (RS232), 16550 har bland annat 16-byte FIFO-buffert och stöd för avbrott. Avbrotten styrs i sin tur av mjukvara.

- 16650 UART

2.2.3. MAX 3223

MAX kretsen är en spänningsomvandlare för UART som ska kopplas till en PC (Eller en komponent som i vanliga fall kopplas till en PC så som kortläsaren.) MAX kretsen omvandlar spänningen i det inbyggda systemets UART från 5 volt till ±12 volt som det är i PCn, dessutom inverterar den de utgående signalerna.

- Maxim MAX3223

2.2.4. EXO3 Oscillator

För att ställa klockfrekvensen på UARTen så att man får rätt Baudrate (19200 baud i det här fallet) behövs en extern oscillator.

Oscillatorn är på 20 MHz men man har möjligheten att dividera original frekvensen med en multipel enligt $1/2^{x}$ i det här fallet sätts den dividerade frekvensen till 5 MHz.

2.3. Logik

Periferikretsarna ska kunna gå i tri-state och måste aktiveras av chip select-signalen. För att detta skall kunna genomföras krävs logikkretsar som kan tolka vilken krets det är processorn vill kontakta. Avbrott genereras av Encodern och UARTen och logik som styr det krävs också.

Logiken valdes därför att delas upp i två delar, avbrottshantering och komponentadressering.

- PALCE22V10 Programmable Array Logic

3. Metod

3.1. Hårdvarukonstruktion

3.1.1. Kretsschema

Programmet Powerlogic användes för att designa ett kretsschema. För att kunna placera ut och dra ledningar mellan de valda komponenterna, krävdes en studie av de tillhörande databladen. Dock måste tilläggas att kretsschemat ritades om ett flertal gånger innan implementeringen påbörjades. Under arbetets gång har kretsschemat sen uppdaterats och har varit en väsentlig del under hela byggets gång.

3.1.2. Byggnation

Ett förborrat och företsat laborationskort, där jord- och spänningsledningar är företsade blev byggplattan. Komponenterna placerades med hänsyn till adressering- och datasignalerna, så processorn placerades i centrum med kortdelen och kontrolldelen på varsin sida.

För att få bort eventuella potentialskillnader i jordplanet, avkopplades var komponent med en kondensator. Jord och spänningspunkterna lödades fast innan virningen av signalledningarna inledes.

Tyvärr virades inte alla kopplingar rätt första gången och ett par omfattande felsökningar var tvunget! Så med facit i hand kan man lugnt säga att virning är en teknik som uppskattas väldigt efter den här kursen!

3.1.3. Testning och felsökning

När alla komponenter virats fast kopplades konstruktionen till ett utvecklingssystem (utvecklat vid it-institutionen, LTH) där var komponent startades efterhand.

Tester genomfördes beroende på typ av komponent. Då fel upptäcktes var det dags att felsöka igen och rätta till felen. Oftast berodde felen på missförstånd av datablad, eller att kretsschemat och verkligheten inte alltid stämmer överrens (benens placering).

3.2. Mjukvarukonstruktion

3.2.1. Gränssnitt

Gränssnittet är uppbyggt i en trädstruktur. Enligt bilden nedan.

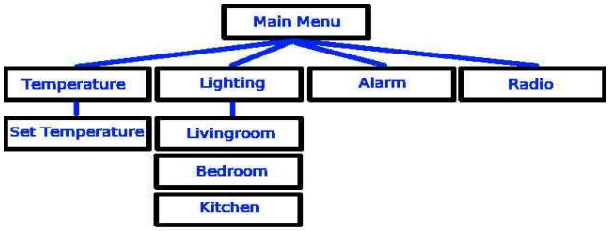


Bild 1, trädstrukturen av menyhanteringen.

Meny systemet är ganska enkelt konstruerat med hänsyn till användaren. Menyn ska enbart användas då en inställning ska ändras. Så då en användare går in på ett av menyvalen kan denna ändra det direkt, spara med enter och återgå till menyn med exit.

3.2.2. Användardatabas

Användardatabasen sparas på EEPROMet, en ganska enkel implementering av databas är gjord med hjälp av pekare. Pekarna är placerade på 32 bytes avstånd. med avstånd på 0x60 och är placerade ut enligt formeln 0x2100 + 200*id ... 0x2240 + 200*id. Minnet utnyttjas dock ej optimalt, men då applikationen är anpassad efter ett hushåll behövs inte minnesrymden optimeras mer.

För var användare sparas följande variabler:

- Id, unikt för var användare.
- Temperatur, den enskilda personen temperatur inställning i sitt rum.
- Belysning vardagsrum
- Belysning sovrum
- Belysning kök
- Radio, av eller på.
- Alarm, av eller på.
- Användarens namn.

3.3. Design av logik

3.3.1. Adressering

För att processorn skall kunna komma åt periferikomponenterna behövs en adressering av dessa. Adresseringen behövs för att rätt komponents chip-select skall aktiveras. För att styra detta behövs en logikkrets, i detta fallet en Programmable Array Logic (PAL). Minnena är på vardera 8kbyte och för enkelhetens skull placerades alla komponenterna med detta avståndet. LCDn och Encodern behöver inte den adressrymden men denna applikationen behöver inte hela processorns kapacitet. Tabellen nedan visar hur adresseringen ska vara för att komma åt de enskilda komponenterna. Alla adresser under A13 är så kallade "don't care".

Enhet	Adress	A13	A14	A15
EPROM	0x0000	0	0	0
EEPROM	0x2000	1	0	0
RAM	0x4000	0	1	0
UART	0x6000	1	1	0
LCD	0x8000	0	0	1
ENCODER	0xA000	1	0	1

3.3.2. Avbrott

Processorn generar har tre avbrottsnivåer 2, 5 och 7. Där 7 är den högsta och kan ej avbrytas. Eftersom systemet har två komponenter som kan generera avbrott (Encoder och UART) används avbrottsnivå 2 och 5. UARTen får avbrtottsnivå 5 och encodern får avbrottsnivå 2, det vill säga att encodern har lägre prioritet är UARTen. Detta med motiveringen att det är viktigare att släppa in en person än att en annan står och ändrar sina personliga inställningar.

IPL0/2 och IPL1 avgör vilken typ av avbrott som sker. Eftersom det är autovektor avbrott som inträffar måste signalen VPA sändas till processorn. Avbrottet hanteras sen i mjukvaran. För att veta vilket tillstånd processorn befinner sig i är FC0, FC1 och FC2 kopplade till avbrottslogiken. Till exempel befinner sig processorn i "interrup acknowledge" då alla tre är höga.

4. Resultat

Som en stor del av uppgiften skulle RFID-läsaren utvärderas, en del frågor ställdes redan innan projektet startades och en del frågor dök upp under arbetets gång.

Till en början undersöktes läsaren med hjälp av mjukvaran som följde med (SMRFID Programmer/Reader v2.2 for Build 5), under Microsoft Windows Version 5.1.

Läsaren ställs in med hjälp av mjukvaran, funktioner som ställs in är till exempel modulation, Spännings Mode (Full eller Eco), antal läsningar och antal block som ska läsas.

Två modulations typer kan sättas Manchester RF/32 eller Manchester RF/64, viktigt är att korten som används och kortläsaren är inställda på samma typ. Manchester RF/64 var redan förvalt och ändrades ej under arbetets gång. Två alternativ av spännings mode kan ställas in Eco eller Full. Full Mode innebär att antennen läser kontinuerligt, i full mode kan ett kort identifieras omedelbart. Eco mode drar mycket mindre ström än Full mode då antennen inte är aktiv hela tiden, är aktiv i intervall. För att identifiera och extrahera extra information från korten, som till exempel lösenord, kan fler än ett block om 4 byte aktiveras. Block 0 används för att synka och identifiera korttyp. Block 7 används för lösenord, och de resterande 30 blocken kan användas för identifikation. Läsaren kan ställas in antal gånger den ska läsa (dock ej unika kort), ställer man in läsaren på att läsa en gång läser den in alla block. Sen måste man återställa läsaren för att kunna läsa igen.

Byte Track är en funktion som letar efter kända bytesekvenser, till exempel läses inte kortet om inte Block 1 är 52 58 8B 45. EM4100/02 är en typ av Byte Track som söker efter sekvensen 1111 1111 1, vilket är EM4100/02s header.

Möjligheten finns att skriva till kortens block, för att läsa och skriva till korten på ett lämpligt avstånd från antennen, kan bland annat två parametrar ställas in för antennen: frekvens och förstärkning.

Tyvärr fick vi inte möjligheten att fullt ut utvärdera läsaren mot vår UART då den brände. Vi kom dock så långt att ett avbrott kunde genereras då ett kort lästes men kommunikations protokollet gick ej att tyda från den lilla testning som hanns med.

En bild av mjukvaran till RFID-läsaren finns i appendix C (bild 5).

5. Avslutning

Projektet i sig är väldigt givande och en av de få kurser på Elektro där man får arbeta självständigt med en uppgift som man helt själv fått välja. Från början till slut beror allting på gruppens förmåga att arbeta tillsammans, inta information och överföra denna till praktiskt användande. Vi anser att fler kurser borde ha inslag av detta i sin utbildning. Detta sätt att arbeta speglar enligt oss den verklighet vi i framtiden kommer att möta. Utanför skolans värld finns det alltid inte facit och genvägar. Ofta måste man sätta sig in i problemen, lära sig felsöka och bearbeta dem och förhoppningsvis komma fram till ett bra resultat. Tyvärr blev inte vårt resultat precis som vi tänkt oss, men ändå anser vi att vi lärt oss enormt mycket. Framförallt har vi lärt oss arbeta i projektform, söka relevant information och använda oss av den.

Vi vill avsluta med att varmt rekommendera denna kurs till alla studerande på LTH.

6. Källkod

```
/*****************************************************************************
                    test.c
                        ------
  Detta ar huvudprogrammet som ar skrivet i ANSI C. Exekveringen av hela
  programpaketet borjar i pmain.68k (lage __main).

  exp2() anropas fran assemblyprogrammet exp2.68k vid avbrott.

  _avben() anropar avben.68k vilket tillater avbrott fran Pl/T.
*****************************************************************************/
#include "functions.h"

unsigned short int *uart_rbr_thr_dll; /* Recieve Holding Register (read, DLAB=0), Transmitter Holding
Register (write, DLAB=0), LSB of Divisor Latch  (DLAB=1)*/
unsigned short int *uart_ier_dlm; /* Interrupt Enable Register (DLAB=0), MSB of Divisor Latch  (DLAB=1)*/
unsigned short int *uart_iir_fcr; /* Interrupt Status Register (read), FIFO Control Register (write) */
unsigned short int *uart_lcr; /* Line Control Register. */
unsigned short int *uart_mcr; /* Modem Control Register. */
unsigned short int *uart_lsr; /* Line Status Register. */
unsigned short int *uart_msr; /* Modem Status Register. */
unsigned short int *uart_scr; /* Scratchpad Register. */


main(){
init_display();
init_uart();
init_encoder();
init_user(0);
load_user();
ready();
_avben();
for(;;);
return;
}

/* Initierar UART */
void init_uart(void) {
uart_rbr_thr_dll = (unsigned short int *) 0x006000;
uart_ier_dlm = (unsigned short int *) 0x006001;
uart_iir_fcr = (unsigned short int *) 0x006002;
uart_lcr = (unsigned short int *) 0x006003;
uart_mcr = (unsigned short int *) 0x006004;
uart_lsr = (unsigned short int *) 0x006005;
uart_msr = (unsigned short int *) 0x006006;
uart_scr = (unsigned short int *) 0x006007;

*uart_lcr = 0x80; /* Sätter DLAB=1 */
*uart_rbr_thr_dll = 0x104; /* LSB of Divisor Latch (=> 5000000 Hz/0x104 = 19200 baud) */
*uart_ier_dlm = 0x00; /* LSB of Divisor Latch = 0 */
*uart_lcr = 0x03; /* Sätter de olika värdena DLAB=0, 8-bit word, 1 stop bit, no parity, no break */
*uart_mcr = 0x00; /* Reset Modem Control Register */
*uart_iir_fcr = 0x07; /* Enable FIFO, clear FIFO, reciever buffer trigger level = 1 byte */
*uart_ier_dlm = 0x07; /* Enable RHD interrupt. */
*uart_iir_fcr;
```

```
*uart_rbr_thr_dll;

}

void concat( char *a, char *b, char *c)
{
 while( *a )  {          /* while( *c++ = *a++ );  */
  *c = *a; ++a; ++c;
 }
 while( *b )  {
  *c = *b; ++b; ++c;
 }
 *c = '\0';
}


exp2(void)              /* avbrottsprogram */
{
 enc_intrpt();
}

exp5(void)              /* avbrottsprogram */
{
main_menu();
}
```

```
#include "functions.h"

unsigned short int i, knappTryck;
unsigned short int menuIndex;
char *rad1,*rad2;
char *id,*temperature,*livingroom,*bedroom,*kitchen,*radio,*alarm, *name;
char tempChars[20], tempChars2[2],*tempChar,*tempChar2;

/* Klar att läsa in kort */
void ready(void){
rad1 = (unsigned short int *) 0x004400; /* Rad1 pekar på en minnesadress i SRAM */
rad2 = (unsigned short int *) 0x004500; /* Rad2 pekar på en minnesadress i SRAM */
/* Tömmer skärmen och tar bort cursor */
clear_display();
hide_cursor();
/* Skriver ut text */
rad1 = " READY TO READ! ";
upper_row(rad1, 16, 0);

/* Temp lösning */
main_menu();
}

/* Huvudmenyn */
void main_menu(void) {

/* Tömmer skärmen och tar bort cursor */
clear_display();
hide_cursor();
rad1 = "    WELCOME!    ";
```

```
upper_row(rad1, 16, 0);

/* Skriver ut "loadingpluttar" */
for (i = 0; i < 45; i++) {
 w_info(0xff);
 /*wait2();*/
 wait();
}
menuIndex = 0;
menu_handler(99);
}

/* Menyhanterare */
void menu_handler(unsigned short int btn) {

if ((menuIndex == 0 && btn == 99) || (menuIndex == 0 && btn == 10)) { /* Huvudmenyn, första två
raderna*/
 rad1 = "1. TEMPERATURE  ";
 rad2 = "2. LIGHTING     ";
 clear_display();
 upper_row(rad1, 16, 0);
 lower_row(rad2, 15, 0);
 w_info(0x7e);
}

if (menuIndex == 0 && btn == 11) { /* Huvudmenyn, sista två raderna*/
 clear_display();
 rad1 = "3. ALARM        ";
 rad2 = "4. RADIO        ";
 upper_row(rad1, 15, 0);
 w_info(0x7f);
 lower_row(rad2, 16, 0);
}



/* Spara undan i livingroom */
if (menuIndex == 21 && btn < 13) {
 if (btn == 12) {
  knappTryck = 0;
  livingroom = tempChar;
 }
 if (btn == 1 && knappTryck < 2) {
  knappTryck = 1;
  w_info('1');
  tempChar = "1";
 }
 if (btn == 0 && knappTryck < 2) {
  knappTryck = 1;
  w_info('0');
  tempChar = "0";
 }
}

/* Spara undan i bedroom */
if (menuIndex == 22 && btn < 13) {
```

```
 if (btn == 12) {
  knappTryck = 0;
  bedroom = tempChar;
 }
 if (btn == 1 && knappTryck < 2) {
  knappTryck = 1;
  w_info('1');
  tempChar = "1";
 }
 if (btn == 0 && knappTryck < 2) {
  knappTryck = 1;
  w_info('0');
  tempChar = "0";
 }
}


/* Spara undan i kitchen */
if (menuIndex == 23 && btn < 13) {
 if (btn == 12) {
  knappTryck = 0;
  kitchen = tempChar;
 }
 if (btn == 1 && knappTryck < 2) {
  knappTryck = 1;
  w_info('1');
  tempChar = "1";
 }
 if (btn == 0 && knappTryck < 2) {
  knappTryck = 1;
  w_info('0');
  tempChar = "0";
 }
}

/* Spara undan i radio */
if (menuIndex == 4 && btn < 13) {
 if (btn == 12) {
  knappTryck = 0;
  radio = tempChar;
 }
 if (btn == 1 && knappTryck < 2) {
  knappTryck = 1;
  w_info('1');
  tempChar = "1";
 }
 if (btn == 0 && knappTryck < 2) {
  knappTryck = 1;
  w_info('0');
  tempChar = "0";
 }
}

/* Spara undan i alarm */
if (menuIndex == 3 && btn < 13) {
 if (btn == 12) {
```

```
  knappTryck = 0;
  alarm = tempChar;
 }
 if (btn == 1 && knappTryck < 2) {
  knappTryck = 1;
  w_info('1');
  tempChar = "1";
 }
 if (btn == 0 && knappTryck < 2) {
  knappTryck = 1;
  w_info('0');
  tempChar = "0";
 }
}

if (menuIndex == 2 && btn == 11){ /* Belysningsmenyn, sista två raderna */
  clear_display();
  menuIndex = 2;
  rad1 = "2. BEDROOM     ";
  rad2 = "3. KITCHEN      ";
  upper_row(rad1, 15, 0);
  w_info(0x7f);
  lower_row(rad2, 16, 0);
}

if (menuIndex == 2 && btn == 1){ /* Belysningsmenyn, inuti livingroom */
  clear_display();
  menuIndex = 21;
  rad1 = "LIVINGROOM:    ";
  concat(rad1,livingroom,tempChars);
  rad2 = "NEW STATE:     ";
  upper_row(tempChars, 16, 0);
  lower_row(rad2, 15, 0);
}
if (menuIndex == 2 && btn == 2){ /* Belysningsmenyn, inuti bedroom */
  clear_display();
  menuIndex = 22;
  rad1 = "BEDROOM:       ";
  concat(rad1,bedroom,tempChars);
  rad2 = "NEW STATE:     ";
  upper_row(tempChars, 16, 0);
  lower_row(rad2, 15, 0);
}

if (menuIndex == 2 && btn == 3){ /* Belysningsmenyn, inuti kitchen */
  clear_display();
  menuIndex = 23;
  rad1 = "KITCHEN:       ";
  concat(rad1,kitchen,tempChars);
  rad2 = "NEW STATE:     ";
  upper_row(tempChars, 16, 0);
  lower_row(rad2, 15, 0);
}

if ((menuIndex == 0 && btn == 2) || (menuIndex == 2 && btn == 99) || (menuIndex == 2 && btn == 10))
{ /* Belysningsmenyn, första två raderna */
```

```
 clear_display();
 rad1 = "CHOOSE ROOM:    ";
 rad2 = "1. LIVINGROOM   ";
 menuIndex = 2;
 upper_row(rad1, 16, 0);
 lower_row(rad2, 15, 0);
 w_info(0x7e);
}

if (menuIndex == 0 && btn == 3) { /* Alarm-menyn */
 clear_display();
 menuIndex = 3;
 rad1 = "ALARM:         ";
 concat(rad1,alarm,tempChars);
 rad2 = "NEW STATE:     ";
 upper_row(tempChars, 16, 0);
 lower_row(rad2, 15, 0);

}

if (menuIndex == 0 && btn == 4) { /* Radiomenyn */
 clear_display();
 menuIndex = 4;
 rad1 = "RADIO:         ";
 concat(rad1,radio,tempChars);
 rad2 = "NEW STATE:     ";
 upper_row(tempChars, 16, 0);
 lower_row(rad2, 15, 0);
}
if (menuIndex == 1 && btn < 13) { /* För att kunna skriva in temperaturen */
 if (knappTryck == 2 && btn == 12) {
  concat(tempChar,tempChar2,tempChars2);
  temperature = tempChars2;
  knappTryck = 0;
 }
 if (knappTryck == 1 && btn < 10) {
  knappTryck = 2;
  switch(btn) {
   case 0: w_info('0');
    tempChar2 = "0"; break;
   case 1: w_info('1');
    tempChar2 = "1"; break;
   case 2: w_info('2');
    tempChar2 = "2"; break;
   case 3: w_info('3');
    tempChar2 = "3"; break;
   case 4: w_info('4');
    tempChar2 = "4"; break;
   case 5: w_info('5');
    tempChar2 = "5"; break;
   case 6: w_info('6');
    tempChar2 = "6"; break;
   case 7: w_info('7');
    tempChar2 = "7"; break;
   case 8: w_info('8');
    tempChar2 = "8"; break;
```

```
   case 9: w_info('9');
    tempChar2 = "9"; break;
  }
 }
 if (knappTryck == 0 && btn < 4) {
  knappTryck = 1;
  switch(btn) {
   case 0: w_info('0');
    tempChar = "0"; break;
   case 1: w_info('1');
    tempChar = "1"; break;
   case 2: w_info('2');
    tempChar = "2"; break;
   case 3: w_info('3');
    tempChar = "3"; break;
  }
 }
}

if (menuIndex == 0 && btn == 1) { /* Temperaturmenyn */
 clear_display();
 rad1 = "CURRENT TEMP: ";
 concat(rad1,temperature,tempChars);
 rad2 = "ENTER TEMP: ";
 menuIndex = 1;
 knappTryck = 0;
 upper_row(tempChars, 16, 0);
 lower_row(rad2, 12, 0);
}
}


/* För att gå tillbaka ett steg i menyn */
void menu_exit(void) {
if (menuIndex == 0){
  main();
}
menuIndex = menuIndex/10;
menu_handler(99);
}

/* Användar hanteringen börjar här */

void init_user(unsigned short int identification) {
id = (unsigned short int *) (0x002100 + 200*identification); /* Pekar på en minnesadress i EEPROM */
temperature = (unsigned short int *) (0x002120 + 200*identification);
livingroom = (unsigned short int *) (0x002140 + 200*identification);
bedroom = (unsigned short int *) (0x002160 + 200*identification);
kitchen = (unsigned short int *) (0x002180 + 200*identification);
radio = (unsigned short int *) (0x002200 + 200*identification);
alarm = (unsigned short int *) (0x002220 + 200*identification);
name = (unsigned short int *) (0x002240 + 200*identification);
tempChar = (unsigned short int *) 0x004200;
tempChar2 = (unsigned short int *) 0x004300;
}
void load_user(void) {
```

```
id = "1";
temperature = "22";
livingroom = "1";
bedroom = "0";
kitchen = "1";
radio = "1";
alarm = "0";
name = "Gustav";
}
```

```
#include "functions.h"

unsigned short int *instr_display, *data_display;

/* Initierar diplayen */
void init_display(void) {
instr_display = (unsigned short int *) 0x008000; /* Pekar på displayen */
data_display = (unsigned short int *) 0x008001; /* Pekar på displayens databuss */
w_instruction(0x1); /* Clear display */
w_instruction(0x38); /* Function set */
w_instruction(0xF); /* Display on/off */
w_instruction(0x6); /* Entry mode set */
w_instruction(0xC); /* Döljer cursor */
}

/* Tömmer displayen på tecken */
void clear_display(void) {
w_instruction(0x1);
}

/* Cursor döljs */
void hide_cursor(void) {
w_instruction(0xC);
}

/* Placerar texten på övre raden */
void upper_row(char text[], unsigned short int text_len,unsigned short int index){

unsigned short int i;

w_instruction(0x80); /* Skriver minnets innehåll längst till vänster på översta raden */

for (i = 0; i < index; i++) {
 w_instruction(0x14); /* Cursor flyttas åt höger */
}

for (i = 0; i < text_len; i++) {
 w_info(text[i]); /* Skriver ut data */
}
}

/* Placerar texten på nedre raden */
void lower_row(char text[], unsigned short int text_len,unsigned short int index){

unsigned short int i;
```

```
w_instruction(0xc0); /* Skriver minnets innehåll längst till vänster på nedre raden */

for (i = 0; i < index; i++) {
 w_instruction(0x14); /* Cursor flyttas åt höger */
}

for (i = 0; i < text_len; i++) {
 w_info(text[i]); /* Skriver ut data */
}
}
/* Skriver instruktion */
void w_instruction(unsigned short int instruction) {
wait(); /* Kräver en viss väntetid. Gör på detta sätt pga att vi använder DS */
*instr_display = instruction;
}

/* Skriver information */
void w_info(char data) {
wait(); /* Kräver en viss väntetid. Gör på detta sätt pga att vi använder DS */
*data_display = data;
}

/* Tidsfördröjning */
void wait(void) {
int i = 0;
while (i < 1000) { /* Väntar tills i=1000, behövs till skärmen */
 i++;
}
}


/* Temporär tidsfördröjning */
void wait2(void) {
int i = 0;
while (i < 10000) { /* Väntar tills i=10000 */
 i++;
}
}
```

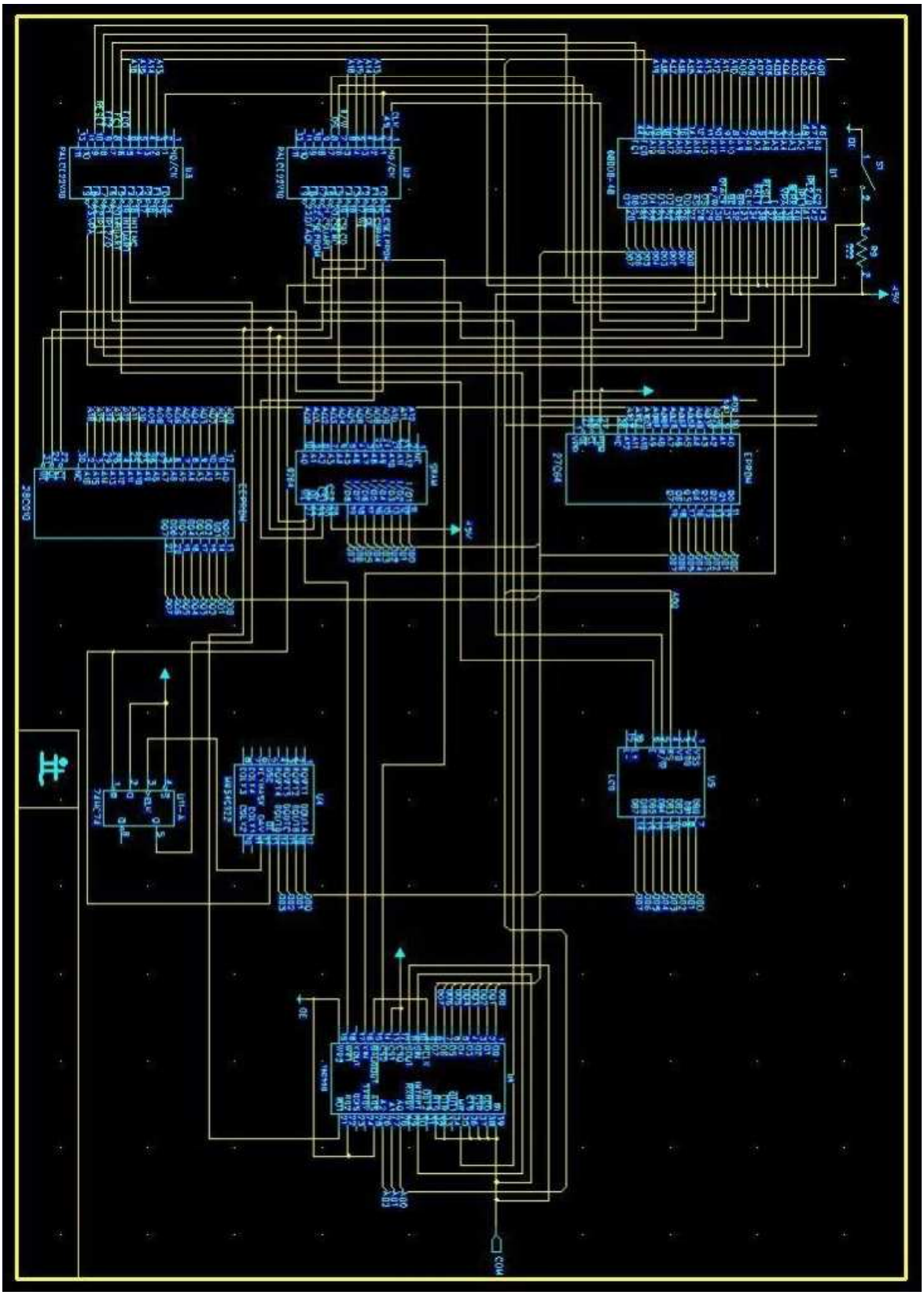
```
#include "functions.h"

#define a0_BTN 0xf3
#define a1_BTN 0xf2
#define a2_BTN 0xf1
#define a3_BTN 0xf0
#define a4_BTN 0xf7
#define a5_BTN 0xf6
#define a6_BTN 0xf5
#define a7_BTN 0xf4
#define a8_BTN 0xfB
#define a9_BTN 0xfA
#define ENTER_BTN 0xff
#define EXIT_BTN 0xfe
#define UP_BTN 0xfd
#define DOWN_BTN 0xfc
```

```
unsigned short int *enc_data;

void init_encoder(void) {
enc_data = (unsigned short int *) 0x00a000; /* Pekar på encodern */
}

/* Knappvalen i menyerna */
void enc_intrpt(void) {
unsigned short int btn = *enc_data;
switch(btn) {
 case a0_BTN: menu_handler(0); break;
 case a1_BTN: menu_handler(1); break;
 case a2_BTN: menu_handler(2); break;
 case a3_BTN: menu_handler(3); break;
 case a4_BTN: menu_handler(4); break;
 case a5_BTN: menu_handler(5); break;
 case a6_BTN: menu_handler(6); break;
 case a7_BTN: menu_handler(7); break;
 case a8_BTN: menu_handler(8); break;
 case a9_BTN: menu_handler(9); break;
 case ENTER_BTN: menu_handler(12); break;
 case EXIT_BTN: menu_exit(); break;
 case UP_BTN: menu_handler(10); break;
 case DOWN_BTN: menu_handler(11); break;
}
}
```

```
#ifndef functions_h
#define functions_h

/* Kopplar samman två schars */
void concat( char *a, char *b, char *c);

/* Initierar Användare */
void init_user(unsigned short int identification);

/******************************** TEMP ***********************************/
void load_user(void);
/*************************************************************************/

/* Initierar UART */
void init_uart(void);

/* Initierar encodern */
void init_encoder(void);

/* Hanterar avbrott från encodern */
void enc_intrpt(void);

/* Meny hanterare */
void menu_handler(unsigned short int btn);

/* Exit i menyn */
void menu_exit(void);
```

```
/* knapp ETT nedtryckt */
void menu_btn1(void);

/* Initierar displayen */
void init_display(void);

/* Tömmer displayen på tecken */
void clear_display(void);

/* Cursor döljs */
void hide_cursor(void);

/* Tidsfördröjning */
void wait(void);

/* Temporär tidsfördröjning */
void wait2(void);

/* Placerar texten på översta raden och lägger till blanksteg */
void upper_row(char text[], unsigned short int text_len,unsigned short int index);

/* Placerar texten på översta raden och lägger till blanksteg */
void lower_row(char text[], unsigned short int text_len,unsigned short int index);

/* Huvudmenyn */
void main_menu(void);

/* Klar att läsa in kort */
void ready(void);

/* Skriver information */
void w_info(char data);

/* Skriver instruktion */
void w_instruction(unsigned short int instruction);

#endif
```

7. Kretsschema



8. Bilder

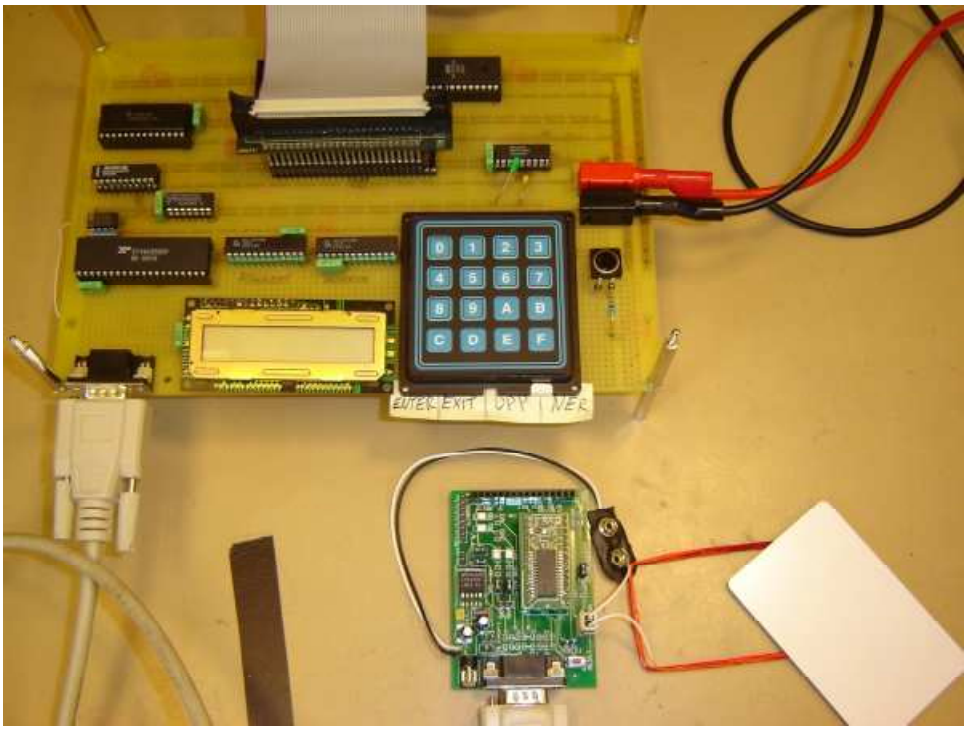


Bild 2, en överblick över konstruktionen med RFID läsaren nere till höger i bild.

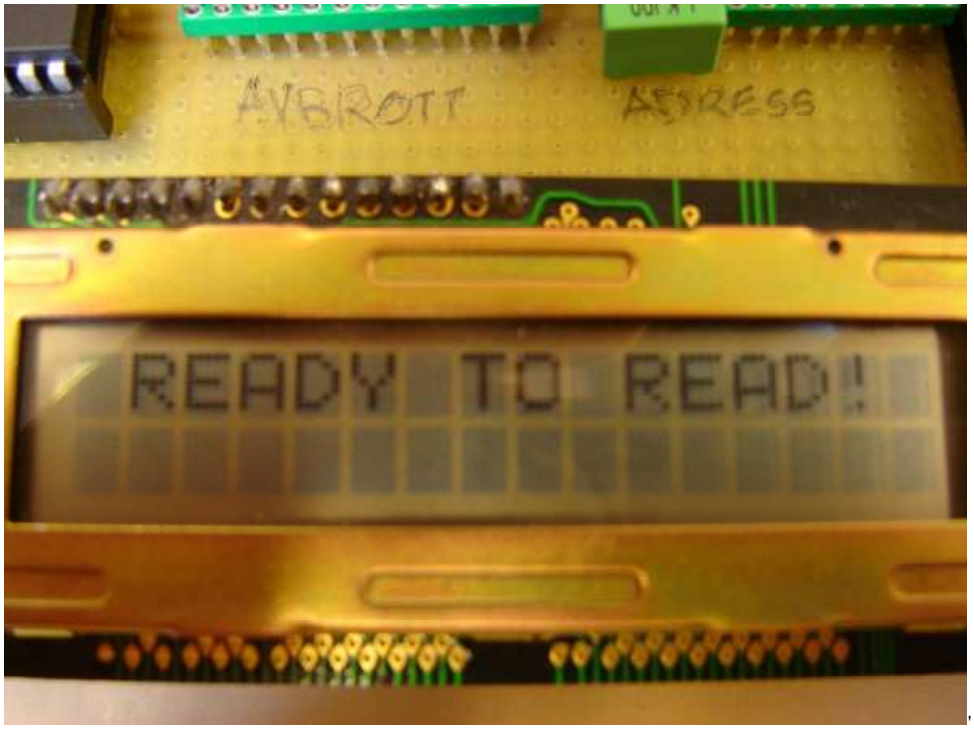


Bild 3, bild på skärmen när kortläsaren är i viloläget och väntar på användare.

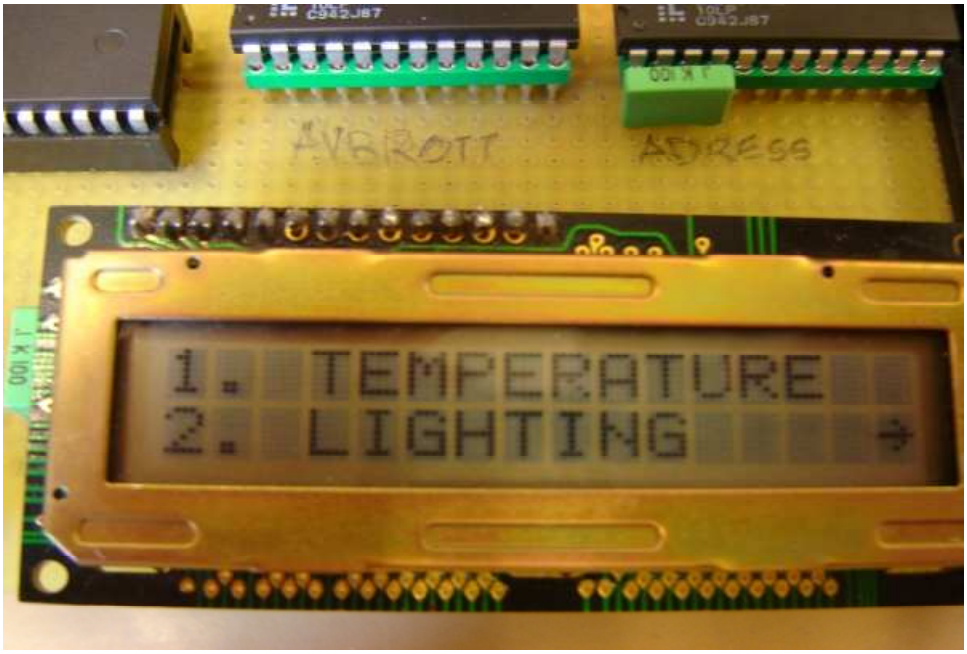


Bild 4, visar exempel på vilka menyer som finns.

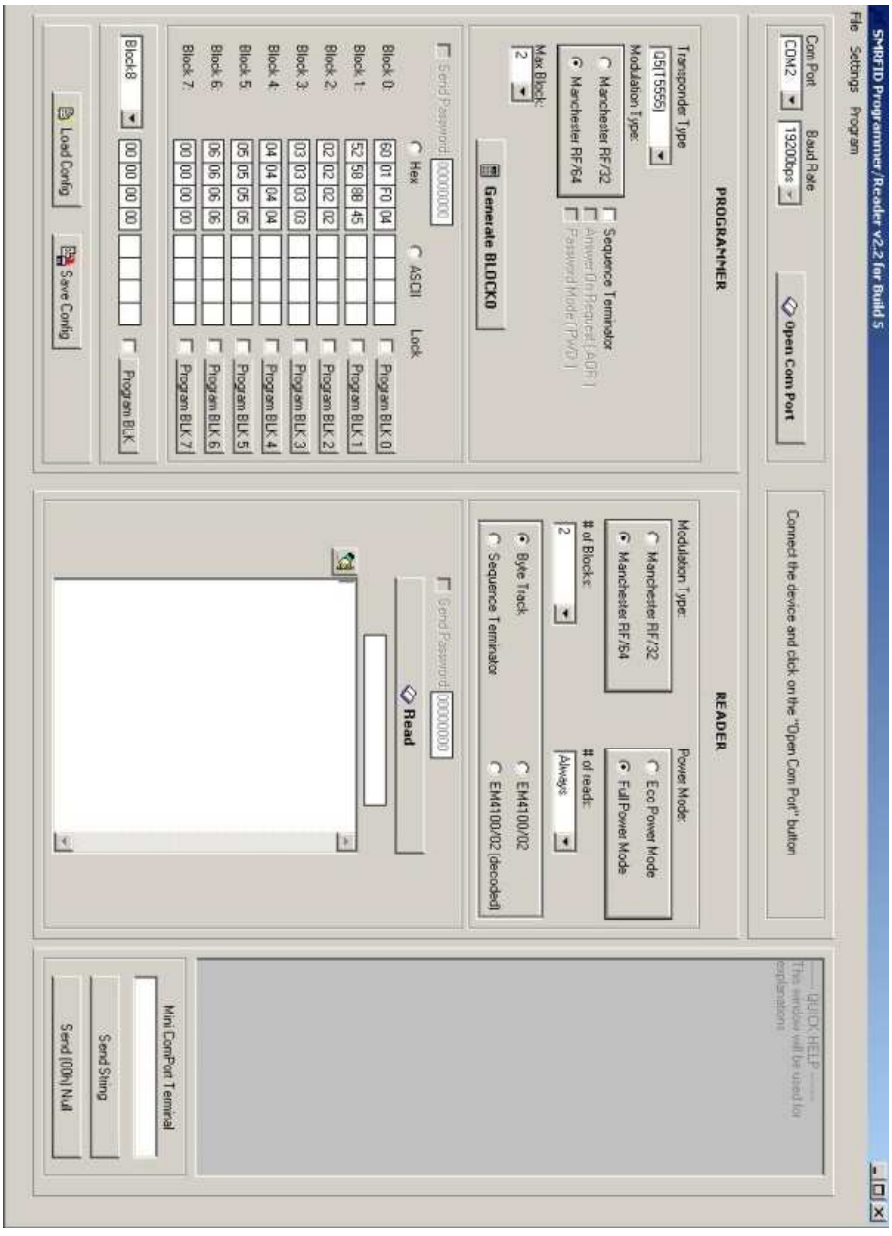


Bild 5, skärmdump av mjukvaran till SonMicros RFID läsare.