LUNDS UNIVERSITET

Larmanläggning

Digitala Projekt EITF11

2018-05-17

Marcus Wuttke I15
Alexander Gabi Goobar I15
Jesper Ehlers I15

Handledare: Bertil Lindvall

1. Sammanfattning

Denna rapport är en sammanställning av ett projekt i kursen Digitala Projekt EITF11 på LTH. Projektet gick ut på att konstruera och programmera en larmcentral för hushåll som ska larma dels vid inbrott och dels vid brand. Larmet använder två infraröda rörelsesensorer och två temperatursensorer som styrs av processorn AVR ATmega16. För att kunna aktivera och avaktivera larmet krävs en fyrsiffrig pinkod som användaren kan ställa in. Prorgrammet skrevs i språket C i Atmel Studio 7.0.

2. Innehållsförteckning

3. Inledning

Vår grupp har valt att konstruera ett övervaknings- och larmsystem anpassat för hushåll. Larmsystemet består dels av två stycken infraröda rörelsedetektorer och dels av två stycken temperatursensorer. När larmet är aktiverat kan det utlösas antingen av att någon av rörelsedetektorerna upptäcker rörelse eller av att någon av temperatursensorerna uppmäter en temperatur över gränsen som användaren får ställa in. Rörelsedetektorerna är tänkta att motsvara ett hemlarm som skyddar mot bland annat inbrottstjuvar och temperatursensorerna skall motsvara ett vanligt brandlarm.

Likt ett modernt hemlarm kan vårt larmsystem avgöra och kommunicera vilken sensor som utlöst, vilket skulle motsvara olika rum eller sektioner i ett hus. Larmet kan även komma ihåg vid vilken tidpunkt larmet utlösts, vilket gör att användaren kommer kunna få följande information:

1. Vilken typ av larm som utlösts.
2. Vilken sektion av huset som larmet utlöstes i.
3. Vid vilken tidpunkt samt hur många dagar sedan larmet utlöstes.

För att aktivera larmet, och för att stänga av det när det utlösts, krävs en fyrsiffrig pinkod. Precis som ett vanligt larm så har användaren endast en kort tidsperiod på sig att deaktivera larmet innan larmet går vidare till en larmcentral. I vårt fall har denna tidsperiod begränsats till 45 sekunder. I vårt larm kommer det inte skickas iväg någon signal till någon larmcentral på riktigt, men användaren kommer informeras om att tiden gått ut via displayen.

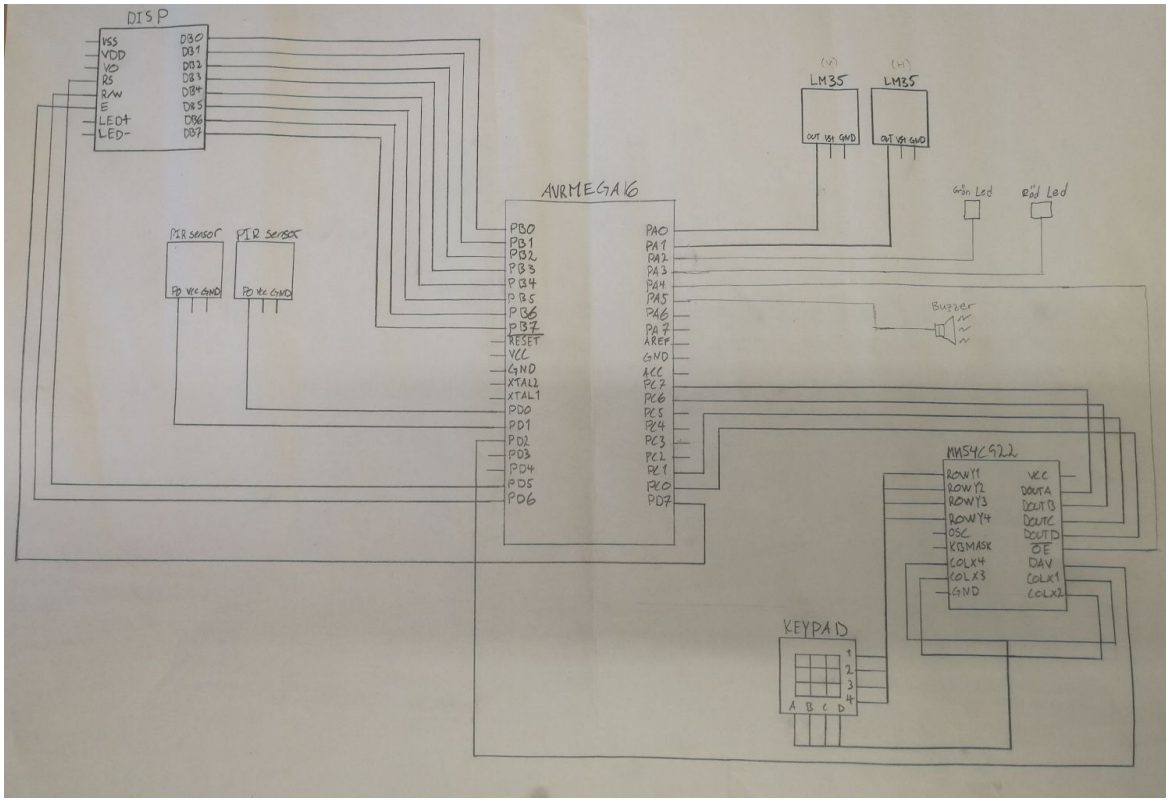
4. Kravspecifikation

Kraven vi ställer på vårt larmsystem är följande:

1. När larmet är avaktiverat ska en grön LED-lampa lysa och displayen skall visa “Larm avaktiverat”
2. När larmet är aktiverat ska en röd LED-lampa lysa och displayen skall ge instruktioner om hur man avaktiverar.
3. När larmet är utlöst ska en buzzer pipa, den röda LED-lampan blinka och displayen skall visa att larmet blivit utlöst. Displayen skall även visa vilken typ av larm som utlösts, vilken sektion larmet kommer ifrån och vad klockan var när larmet utlöstes samt hur många dagar sedan det skedde.
4. När användaren trycker på knappen “A” skall displayen uppmana användaren att mata in en pinkod. Användaren skall då kunna välja pinkoden, och sedan ska larmet aktiveras.
5. När larmet är aktivt (oavsett om det är utlöst eller ej) ska användaren kunna trycka “D” och med den rätta pinkoden avaktivera larmet.
6. Larmet skall utlösas om det är aktiverat och någon av rörelsesensorerna upptäcker rörelse.

7. Larmet skall utlösas om det är aktiverat och någon av temperatursensorerna uppmäter en temperatur över gränsen som användaren har angivit.

5. Kopplingsschema



6. Hårdvara

6.1. Processor

Processorn vi har använt är en AVR ATmega16. Denna processor har 4 portar sorterade i PA till PD och 40 olika pinnar. Processorn har programmerats i programmeringsspråket C och koden har bränts in i processorn med hjälp av en JTAG.

6.2. Display

Till Display har vi använt en Optrex DM16433 med fyra rader och 16 tecken på varje. Detta för att få plats med att visa larmets status samt information om vad som hände när larmet utlöses.

6.3. Knappsats

Vår knappsats är en 4x4 hexadecimal knappsats. Siffrorna används för att mata in pinkoden och bokstäverna för olika kommandon som byta pinkod, aktivera larmet eller avaktivera larmet.

6.4. Key encoder

Till vår knappsats har vi använt en key encoder av modell MM74C922 för att omvandla signalerna från knappsatsen till processorn.

6.5 Rörelsesensorer

För att kunna upptäcka rörelse har vi använt oss av två stycket PIR-sensorer (Passive infrared sensor). Sensorerna använder infrarött ljus för att känna av om någonting rör sig i dess omgivning och utlöser då larmet.

6.6. Temperatursensorer

För att mäta temperaturen har vi använt två stycken LM35 Centigrade temperature sensor.

6.7. LED-lampor

Vi har använt två LED-lampor för att kunna se när larmet är aktiverat respektive avaktiverat. Grön LED för avaktiverat och röd LED för aktiverat. Blinkade röd LED för utlöst larm.

6.8. Buzzer

För att bli uppmärksammad på när larmet går och för att skrämma iväg alla tjuvar använder vi en buzzer som börjar pipa när larmet utlöses.

7. Mjukvara

Koden är uppbyggd runt en envig while-loop som körs så länge larmet får ström. While-loopen kallar sedan på olika metoder beroende på vad som händer med larmet. Exempel på vad som kan hända är att larmet aktiveras eller avaktiveras, någon trycker på knappsatsen, eller att larmet utlöses av antingen IR-sensorerna eller temperatursensorerna.

Mjukvaran är programmerad i språket C i Atmel Studio 7.0.

8. Metod

För att få en bra överblick över hur vårt larmsystem skulle se ut började vi med att leta upp komponenter som skulle passa vårt syfte och sedan rita upp ett kopplingsschema. Så fort vi fått tillgång till våra komponenter började vi koppla ihop komponenterna efter vårt kopplingsschema. Medan vi kopplade gjorde vi kontinuerliga tester för att kolla så att våra kopplingar fungerade som vi hade tänkt. Testen utfördes dels med en multimeter och dels med att provköra små kodavsnitt anpassade för att testa om kopplingarna fungerade som vi hade tänkt oss.

När vi hade gjort alla kopplingar och det var dags att programmera själva processorn delade vi upp koden i olika avsnitt baserat på de funktioner som de skulle tillföra till larmet. Efter att vi gjort vardera avsnitt testade vi att köra den separat och göra eventuella ändringar för att få den tilltänkta funktionaliteten. Efter att vi var nöjda med varje enskilt avsnitt separat lade vi ihop allt och brände in det i processorn med hjälp av en JTAG.

9. Diskussion

Det har varit spännande och lärorikt att jobba med att bygga sitt eget larmsystem. Att komma igång var en utmaning för vår grupp då ingen programmerat i C innan och våra kunskaper inom att löda och koppla komponenter varit begränsad. När vi väl börjat förstå hur databladen fungerar och dynamiken mellan de olika komponenterna gick det snabbt fram med arbetet. Vi var dock tvungna att göra om en del kopplingar då vi varit lite för ivriga att komma vidare innan vi haft full förståelse för hur komponenterna fungerar.

Vi var noga inom gruppen att alla skulle förstå hur dynamiken fungerade och varför vi gjorde de saker vi gjorde. Detta för att undvika missförstånd då det kan bli rörigt när tre personer jobbar på samma sak. Detta är något vi lyckades bra med och vi slapp därmed större missförstånd och omarbetningar.

Något som vi skulle ha kunna förbättra med vårt larm är att larmet ger ifrån sig olika signaler när olika saker inträffar. T.ex. skulle man kunna ha en typ av signal när larmet aktiveras, ett när det avaktiveras och sen olika signaler beroende på om det är rörelsesensorn eller temperatursensorn som utlöser larmet. För att snabbare kunna upptäcka brand hade det varit mer effektivt med en röksensor som utlöser larmet när den kommer i kontakt med brandrök. Dock så är röksensorer dels mer komplicerade att jobba med och dels smått radioaktiva, så för att smidigt ta fram en enkel prototyp har vi istället valt att arbeta med temperatursensorer.

10. Källkod

```
/*
 * AlarmProject.c
 *
 * Created: 2018-04-20 13:47:12
 * Author : ine15ago
 */




//***************************************************
// Include and define
//***************************************************
#define F_CPU 80000000
#include <avr/io.h>
#include <avr/portpins.h>
#include <avr/interrupt.h>
#include <util/delay.h>



//***************************************************
// Variables
//***************************************************

#define green PA2
#define red PA3

volatile unsigned char keypad_char = 'M';
volatile unsigned int keypad_update = 0;
unsigned int alarm1_level_motion = 0; // 0=Off, 1=On, 2=Triggered
unsigned int alarm2_level_motion = 0; // 0=Off, 1=On, 2=Triggered
unsigned int alarm1_level_temp = 0; // 0=Off, 1=On, 2=Triggered
unsigned int alarm2_level_temp = 0; // 0=Off, 1=On, 2=Triggered
unsigned int intruder_alert = 0;
unsigned int pinCode[4] = {0,0,0,0};
unsigned int triggered = 0;
unsigned int timestamp = 0;
unsigned int timerTick = 0;
unsigned int clockHour = 0;
unsigned int clockMinute = 0;
```

```
unsigned int clockSeconds= 0;
unsigned int secondsPassed = 0;
unsigned int daysPassed = 0;
unsigned int tempLimit = 0;
volatile int16_t adc_res = 0;
volatile int16_t adc2_res = 0;
volatile uint16_t temp1_trig = 0;
volatile uint16_t temp2_trig = 0;
volatile uint16_t temp1buffer[8] = {0,0,0,0,0,0,0,0};
volatile uint16_t temp2buffer[8] = {0,0,0,0,0,0,0,0};
volatile uint16_t temp1Counter = 0;
volatile uint16_t temp2Counter = 0;
volatile uint16_t avgTemp1 = 0;
volatile uint16_t avgTemp2 = 0;


void alarm_motion(void);
void test(void);
void initSystem(void);
void clearScreen(void);
void stringPrint(char *text);
void initPins(void);
void initOther(void);
void initScreen(void);
void enableReset(void);
void textEntry(char c);
void keypad_read(void);
void initADC(void);
void setClock(void);
void printTime(void);
void addMinute(void);
void red_flash(void);
void sound(void);
void printBigInt();
void initTimers(void);
void printIntTime();
void alarm_temp(void);
void alarm1_motion(void);
void alarm2_motion(void);
void activate_alarm(void);
void inactivate_alarm(void);
void setTempLimit(void);
void printTemp(void);
```

```
//*************************************************
// Main
//*************************************************

int main(){

            initSystem();
            clearScreen();
            initADC();
            sei();
            setClock();
            setTempLimit();

            ADCSRA |= 1<<ADSC;

            while(1){

            if(alarm_status() == 0){
                  stringPrint("Deactivated. Press A to activate.");
                  while (keypad_char != 'A'){
                  }
                  clearScreen();
                  stringPrint("Choose a 4-digitPIN:");
                  while (keypad_char == 'A'){
                  }
                  int pinCount = 0;
                  while (pinCount < 4){
                        if (keypad_update == 1){
                              if (keypad_char == '0' ||  keypad_char == '1'
|| keypad_char == '2' || keypad_char == '3' || keypad_char == '4' ||
keypad_char == '5'
                              || keypad_char == '6'   || keypad_char == '7'
|| keypad_char == '8' || keypad_char == '9'){
                                    textEntry(keypad_char);
                                    pinCode[pinCount] = keypad_char;
                                    keypad_update = 0;
                                    pinCount++;
                              }
                        }
                  }
                  _delay_ms(20);
                  if (pinCount == 4) {
                        clearScreen();
                        stringPrint("Activated!");
                        _delay_ms(50);
```

```
                        clearScreen();
                        stringPrint("Press D to      deactivate");

                  }
                  activate_alarm();
                  _delay_ms(300);

            }

            if(alarm1_level_motion == 1 ){
                  alarm1_motion();
            }
            if(alarm2_level_motion == 1){
                  alarm2_motion();
            }
            if(alarm1_level_temp == 1){
                  if (temp1_trig == 1){
                        alarm1_level_temp = 2;
                  }
            }
            if (alarm2_level_temp == 1){
                  if (temp2_trig == 1){
                        alarm2_level_temp = 2;
                  }
            }

            if(alarm_status() == 1 && keypad_update == 1 && keypad_char ==
'D'){
                  clearScreen();
                  stringPrint("Type PIN:");
                  keypad_update = 0;
                  int pinCount = 0;
                  int pinAttempt[4] = {9,9,9,9};
                  while (pinCount < 4){
                        if (keypad_update == 1){
                              if (keypad_char == '0' ||  keypad_char == '1'
|| keypad_char == '2' || keypad_char == '3' || keypad_char == '4' ||
keypad_char == '5'
                              || keypad_char == '6'   || keypad_char == '7'
|| keypad_char == '8' || keypad_char == '9'){
                                    textEntry(keypad_char);
                                    pinAttempt[pinCount] = keypad_char;
                                    keypad_update = 0;
                                    pinCount++;
                              }
```

```
                        }
                  }
                  _delay_ms(20);
                  clearScreen();
                  if (pinCode[0] == pinAttempt [0] && pinCode[1] ==
pinAttempt [1] && pinCode[2] == pinAttempt [2] && pinCode[3] == pinAttempt
[3]){
                        clearScreen();
                        stringPrint("PIN correct!");
                        _delay_ms(200);
                        inactivate_alarm();
                        triggered = 0;
                        clearScreen();
                  }
                  else {
                        clearScreen();
                        stringPrint("PIN incorrect! Press D to try
again.");
                        _delay_ms(50);
                  }
            }


            if(alarm_status() == 2){
                  clearScreen();
                  triggered = 1;
                  int trigHour = clockHour;
                  int trigMinute = clockMinute;
                  stringPrint("Triggered! ");
                  printIntTime(trigHour);
                  textEntry(':');
                  printIntTime(trigMinute);
                  printBigInt(daysPassed);
                  stringPrint("D ");
                  if(alarm1_level_motion == 2){
                        stringPrint("R1M ");
                  }
                  if(alarm2_level_motion == 2){
                        stringPrint("R2M ");
                  }
                  if(alarm1_level_temp == 2){
                        stringPrint("R1T ");
                        printBigInt(adc_res/2);
                        stringPrint("C ");
                  }
```

```
                  if(alarm2_level_temp == 2){
                        stringPrint("R2T ");
                        printBigInt(adc2_res/2);
                        stringPrint("C ");
                  }
                  while (triggered == 1){
                        while (keypad_char != 'D'){
                              sound();
                        }
                        if (keypad_char == 'D'){
                              clearScreen();
                              stringPrint("Type PIN:");
                              keypad_update = 0;
                              sound();
                        }
                        int pinCount = 0;
                        int pinAttempt[4] = {9,9,9,9};
                        while (pinCount < 4){
                              sound();
                              if (keypad_update == 1){
                                    if (keypad_char == '0' ||  keypad_char
== '1' || keypad_char == '2' || keypad_char == '3' || keypad_char == '4'
|| keypad_char == '5'
                                    || keypad_char == '6'   || keypad_char
== '7' || keypad_char == '8' || keypad_char == '9'){
                                          textEntry(keypad_char);
                                          pinAttempt[pinCount] =
keypad_char;
                                          keypad_update = 0;
                                          pinCount++;
                                    }
                              }
                        }
                        _delay_ms(20);
                        clearScreen();
                        if (pinCode[0] == pinAttempt [0] && pinCode[1] ==
pinAttempt [1] && pinCode[2] == pinAttempt [2] && pinCode[3] == pinAttempt
[3]){
                              clearScreen();
                              stringPrint("PIN correct!");
                              _delay_ms(200);
                              inactivate_alarm();
                              triggered = 0;
                              clearScreen();
                        }
```

```
                        else {
                              clearScreen();
                              stringPrint("PIN incorrect! Press D to try
again.");
                              sound();
                              _delay_ms(50);
                        }
                  }
            }
      }
}

void activate_alarm( ){
            alarm1_level_motion = 1;
            alarm1_level_temp = 1;
            alarm2_level_motion = 1;
            alarm2_level_temp = 1;
            PORTA &= ~(1<<green);  // drive PA2 low
            PORTA |= (1<<red);   // drive PA2 high
}

void inactivate_alarm(){
            alarm1_level_motion = 0;
            alarm1_level_temp = 0;
            alarm2_level_motion = 0;
            alarm2_level_temp = 0;
            temp1_trig = 0;
            temp2_trig = 0;
            PORTA &= ~(1<<red);  // drive PA2 low
            PORTA |= (1<<green);   // drive PA2 high
}

//*** Alarm av = 0, Alarm aktiverat = 1, Någon sensor triggad = 2; ***/
int alarm_status(){
      if(alarm1_level_temp == 0 && alarm2_level_temp == 0 &&
alarm1_level_motion == 0 && alarm2_level_motion == 0){ return 0;}
      if(alarm1_level_temp == 1 && alarm2_level_temp == 1 &&
alarm1_level_motion == 1 && alarm2_level_motion == 1){ return 1;}
      if(alarm1_level_temp == 2 || alarm2_level_temp == 2 ||
alarm1_level_motion == 2 || alarm2_level_motion == 2){ return 2;}
}


//*****************************************************
// Motion sensor readings
```

```
//*****************************************************

void alarm1_motion(){
      char read = PIND & 0b00000001; //PD0 2^0
      if(read == 1){
            alarm1_level_motion = 2;
      }
      if(read == 0){
            alarm1_level_motion = 1;
      }
}

void alarm2_motion(){
      char read = PIND & 0b00000010; //PD1     2^1
      if(read == 2){
            alarm2_level_motion = 2;
      }
      if(read == 0){
            alarm2_level_motion = 1;
      }
}

//*************************************************
// Initiation
//*************************************************

void initSystem()
{
      initPins();
      initScreen();
      initOther();
      initTimers();
      clearScreen();

}

void initPins(){
      DDRA = 0b00111100; //endast PA0 och PA1 används. För LM35. Samtliga
pins ska vara input.
      DDRB = 0b11111111; // till Display. Alla pins output
      DDRC = 0b00000000; //PC0, PC1, PC6, PC7 används för key encoder
      DDRD = 0b11100000;  //PD5,PD6, PD7 för LCD,  PD0 och PD1 för
PIR-sensor och PD2 för INT (keypad)
}
```

```
void initOther(){
	PORTA = 0b00000100; //PA0 och PA1 är LM35, PA2 är grön led och PA3
är röd led. 1100 slutet är på.
	MCUCR = (1<<ISC01)|(1<<ISC00);
	GICR  = (1<<INT0);
}

// **********************Timers*******************
void initTimers(){
	TCCR2 = TCCR2 | 0b00000110;
	TIMSK = TIMSK | 0b01000001; // Activate overflow interrupt for timer
}


void initADC()
{
	ADCSRA |= 1<<ADPS2;
	ADMUX |= 1<<REFS0 | 1<<REFS1;
	ADCSRA |= 1<<ADIE;
	ADCSRA |= 1<<ADEN;
}


//*****************************************************
// LCD + LCD Init
//*****************************************************
void initScreen(){

		PORTD = (1 << PD7) | (0 << PD6) | (0 << PD5);
		  //RS = 0, E = 1, R/W = 0

		  PORTB =     (0 << PB7) | (0 << PB6) | (1 << PB5) | (1 <<
PB4) |		//Initiation, function set
				(1 << PB3) | (0 << PB2) | (0 << PB1) | (0 <<
PB0);

		  PORTB =     (0 << PB7) | (0 << PB6) | (0 << PB5) | (0 <<
PB4) |		//Display off
				(1 << PB3) | (0 << PB2) | (0 << PB1) | (0 <<
PB0);
		  enableReset();

		  PORTB =     (0 << PB7) | (0 << PB6) | (0 << PB5) | (0 <<
PB4) |		//Display clear
				(0 << PB3) | (0 << PB2) | (0 << PB1) | (1 <<
```

```
PB0);
                enableReset();

                PORTB =     (0 << PB7) | (0 << PB6) | (0 << PB5) | (0 <<
PB4) |                //Entry mode
                            (0 << PB3) | (1 << PB2) | (1 << PB1) | (0 <<
PB0);
                enableReset();

                PORTB =     (0 << PB7) | (0 << PB6) | (0 << PB5) | (0 <<
PB4) |                //Display on, cursor off, blink on (DB7-DB0)
                            (1 << PB3) | (1 << PB2) | (0 << PB1) | (1 <<
PB0);
                enableReset();
}

void enableReset(){
                      PORTD = (0 << PD7) | (0 << PD6) | (0 << PD5);
                      //Enable
                      PORTD = (1 << PD7) | (0 << PD6) | (0 << PD5);
                //Enable, then reset
                      _delay_ms(80);
}

void clearScreen()
{
      PORTD = (1 << PD7) | (0 << PD6) | (0 << PD5);
      //RS = 0, E = 1, R/W = 0
      PORTB =     (0 << PB7) | (0 << PB6) | (0 << PB5) | (0 << PB4) |
      //Display clear
      (0 << PB3) | (0 << PB2) | (0 << PB1) | (1 << PB0);
      PORTD = (0 << PD7) | (0 << PD6) | (0 << PD5);
      //Enable
      PORTD = (1 << PD7) | (1 << PD6) | (0 << PD5);
      //RS = 0, E = 1, R/W = 0
      _delay_ms(4);
}

//*****************************************************
// String and char LCD Software
//*****************************************************
void textEntry(char c)
{
      int i;
      unsigned short output[8];
```

```
      for (i = 0; i < 8; i++) {
        output[i] = (c >> i) & 1;
      }

      PORTD = (1 << PD7) | (0 << PD6) | (1 << PD5);
                        //sending data
      PORTB =     (output[7] << PB7) | (output[6] << PB6) | (output[5] <<
PB5) | (output[4] << PB4) |
                  (output[3] << PB3) | (output[2] << PB2) | (output[1] <<
PB1) | (output[0] << PB0);
      PORTD = (0 << PD7) | (0 << PD6) | (1 << PD5);
                        //Enable
      PORTD = (1 << PD7) | (0 << PD6) | (1 << PD5);
                        //Reseting enable
      _delay_us(80);
                                          //Wait
}

void type_char(){
      if (keypad_update == 1) {
            textEntry(keypad_char);
            keypad_update = 0;
      }
}

void stringPrint(char *text){
      while(*text){
            textEntry(*text++);

      }
}

void printIntTime(int number){
      char buffer[3];
      sprintf(buffer,"%02d", number);
      stringPrint(buffer);
}

void printBigInt(int number){
      char buffer[12];
      sprintf(buffer,"%d", number);
      stringPrint(buffer);
}
```

```
//*****************************************************
// Keypad
//*****************************************************

ISR(INT0_vect){
      keypad_update = 1;
      keypad_read();
      _delay_us(500);
}

void keypad_read(){
      if(PINC ==0x00){keypad_char = '0';}
      if(PINC ==0x80){keypad_char = '1';}
      if(PINC ==0x40){keypad_char = '2';}
      if(PINC ==0xC0){keypad_char = '3';}
      if(PINC ==0x02){keypad_char = '4';}
      if(PINC ==0x82){keypad_char = '5';}
      if(PINC ==0x42){keypad_char = '6';}
      if(PINC ==0xC2){keypad_char = '7';}
      if(PINC ==0x01){keypad_char = '8';}
      if(PINC ==0x81){keypad_char = '9';}
      if(PINC ==0x41){keypad_char = 'A';}
      if(PINC ==0xC1){keypad_char = 'B';}
      if(PINC ==0x03){keypad_char = 'C';}
      if(PINC ==0x83){keypad_char = 'D';}
      if(PINC ==0x43){clearScreen(); printTemp();;}
      if(PINC ==0xC3){clearScreen(); printTime();}

}

void waitForInput() {
      while (keypad_update == '0');
}


//*****************************************************
// Alarm noise/flashing
//*****************************************************

void sound(){
      for (int i = 0; i < 80; i++) {
                  PORTA |= (1<<PA5);   // drive PA2 high
                  _delay_us(5);         // delay 100 ms
                  PORTA &= ~(1<<PA5);  // drive PA2 low
                  _delay_us(5);         // delay 900 ms
```

```
	}
	red_flash();
}

void red_flash(){
	PORTA |= (1<<red);   // drive PA2 high
	_delay_ms(5);
	PORTA &= ~(1<<red);  // drive PA2 low
	_delay_ms(5);

}

//*****************************************************
// Clock/timer
//*****************************************************

//Overflow Interrupt
ISR(TIMER2_OVF_vect) {
	timerTick++;
	clockSeconds = timerTick/132;
	secondsPassed = timerTick/132;
	if (clockSeconds >= 60) {
		timerTick = 0;
		clockSeconds = 0;
		addMinute();
	}
}

void addMinute(){
	clockMinute++;
	if (clockMinute > 59) {
		clockHour++;
		clockMinute = 0;
		if (clockHour > 23){
			daysPassed++;
			clockHour = 0;
		}
	}
}

void setClock(){
	stringPrint("Current time:   ");
	waitForInput();
	int hourCount = 0;
	int hourTens = 0;
```

```
      int hourOnes = 0;
      while (hourCount < 2){
            if (hourCount == 0 && keypad_update == 1){
                  if (keypad_char == '0' ||  keypad_char == '1' ||
keypad_char == '2'){
                        hourTens = keypad_char - '0';
                        textEntry(keypad_char);
                        keypad_update = 0;
                        hourCount = 1;
                  }
            }
            waitForInput();
            if (hourCount == 1 && keypad_update == 1){
                  if (hourTens == 2){
                        if (keypad_char == '0' ||  keypad_char == '1' ||
keypad_char == '2' || keypad_char == '3'){
                              hourOnes = keypad_char - '0';
                              textEntry(keypad_char);
                              keypad_update = 0;
                              hourCount = 2;
                        }
                        } else {
                        if (keypad_char == '0' ||  keypad_char == '1' ||
keypad_char == '2' || keypad_char == '3' || keypad_char == '4' ||
keypad_char == '5'
                        || keypad_char == '6'   || keypad_char == '7' ||
keypad_char == '8' || keypad_char == '9'){
                              hourOnes = keypad_char - '0';
                              textEntry(keypad_char);
                              keypad_update = 0;
                              hourCount = 2;
                        }
                  }

            }

      }
      clockHour = hourTens*10 + hourOnes;
      textEntry(':');
      int minuteCount = 0;
      int minuteTens = 0;
      int minuteOnes = 0;
      while (minuteCount < 2){
            if (minuteCount == 0 && keypad_update == 1){
                  if (keypad_char == '0' ||  keypad_char == '1' ||
```

```
keypad_char == '2' || keypad_char == '3' || keypad_char == '4' ||
keypad_char == '5'){
                        minuteTens = keypad_char - '0';
                        textEntry(keypad_char);
                        keypad_update = 0;
                        minuteCount = 1;
                  }
            }
            waitForInput();
            if (minuteCount == 1 && keypad_update == 1){
                  if (keypad_char == '0' ||  keypad_char == '1' ||
keypad_char == '2' || keypad_char == '3' || keypad_char == '4' ||
keypad_char == '5'
                  || keypad_char == '6'   || keypad_char == '7' ||
keypad_char == '8' || keypad_char == '9'){
                        minuteOnes = keypad_char -'0';
                        textEntry(keypad_char);
                        keypad_update = 0;
                        minuteCount = 2;
                  }

            }
      }
      clockMinute = minuteTens*10 + minuteOnes;
      stringPrint(" OK!");
      timerTick = 0;
      _delay_ms(200);
      clearScreen();
}

void printTime(){
      printIntTime(clockHour);
      textEntry(':');
      printIntTime(clockMinute);
      textEntry(':');
      printIntTime(clockSeconds);
}

//*****************************************************
// Analog interrupt. Temperature functions
//*****************************************************

ISR(ADC_vect) {
      uint16_t adc_read = ADC;
      switch (ADMUX)
```

```
	{
		case 0xC0:
			ADMUX = 0xC1;
			adc_res = adc_read;
			temp1buffer[temp1Counter] = adc_read;
			temp1Counter++;
			break;
		case 0xC1:
			ADMUX = 0xC0;
			adc2_res = adc_read;
			temp2buffer[temp2Counter] = adc_read;
			temp2Counter++;
			break;
		default:
			stringPrint("FEL");
			//Default code, not used
		break;
	}
	if (temp1Counter == 8) {
		avgTemp1 = (temp1buffer[0] + temp1buffer[1] + temp1buffer[2] +
temp1buffer[3] + temp1buffer[4] + temp1buffer[5] + temp1buffer[6] +
temp1buffer[7]) / 8;
		if (avgTemp1 > tempLimit){
			temp1_trig = 1;
		}
		temp1Counter = 0;
	}
	if (temp2Counter == 8) {
		avgTemp2 = (temp2buffer[0] + temp2buffer[1] + temp2buffer[2] +
temp2buffer[3] + temp2buffer[4] + temp2buffer[5] + temp2buffer[6] +
temp2buffer[7]) / 8;
		if (avgTemp2 > tempLimit){
			temp2_trig = 1;
		}
		temp2Counter = 0;
	}
	ADCSRA |= 1<<ADSC;
}

void setTempLimit(){
	stringPrint("Temp limit, end with C:");
	waitForInput();
	int input[3] = {0,0,0};
	int tempCount = 0;
	while (keypad_char != 'C' && tempCount < 3){
```

```
            if (keypad_update == 1){
                  if (keypad_char == '0' ||  keypad_char == '1' ||
keypad_char == '2' || keypad_char == '3' || keypad_char == '4' ||
keypad_char == '5'
                  || keypad_char == '6'   || keypad_char == '7' ||
keypad_char == '8' || keypad_char == '9'){
                        textEntry(keypad_char);
                        input[tempCount] = keypad_char - '0';
                        keypad_update = 0;
                        tempCount++;
                  }
            }
      }
      if (tempCount == 1){
            tempLimit = 2*input[0];
            } else if (tempCount == 2){
            tempLimit = 2*(10*input[0] + 10*input[1]);
            } else {
            tempLimit = 2*(100*input[0] + 10*input[1] + input[2]);
      }
      stringPrint(" OK!");
      timerTick = 0;
      _delay_ms(200);
      clearScreen();
}

void printTemp(){
      char buffer1[12];
      char buffer2[12];
      sprintf(buffer1,"%d", avgTemp1/2);
      sprintf(buffer2,"%d", avgTemp2/2);
      stringPrint("Room 1 (L): ");
      stringPrint(buffer1);
      textEntry('C');
      stringPrint(" ");
      stringPrint("Room 2 (R): ");
      stringPrint(buffer2);
      textEntry('C');
}
```