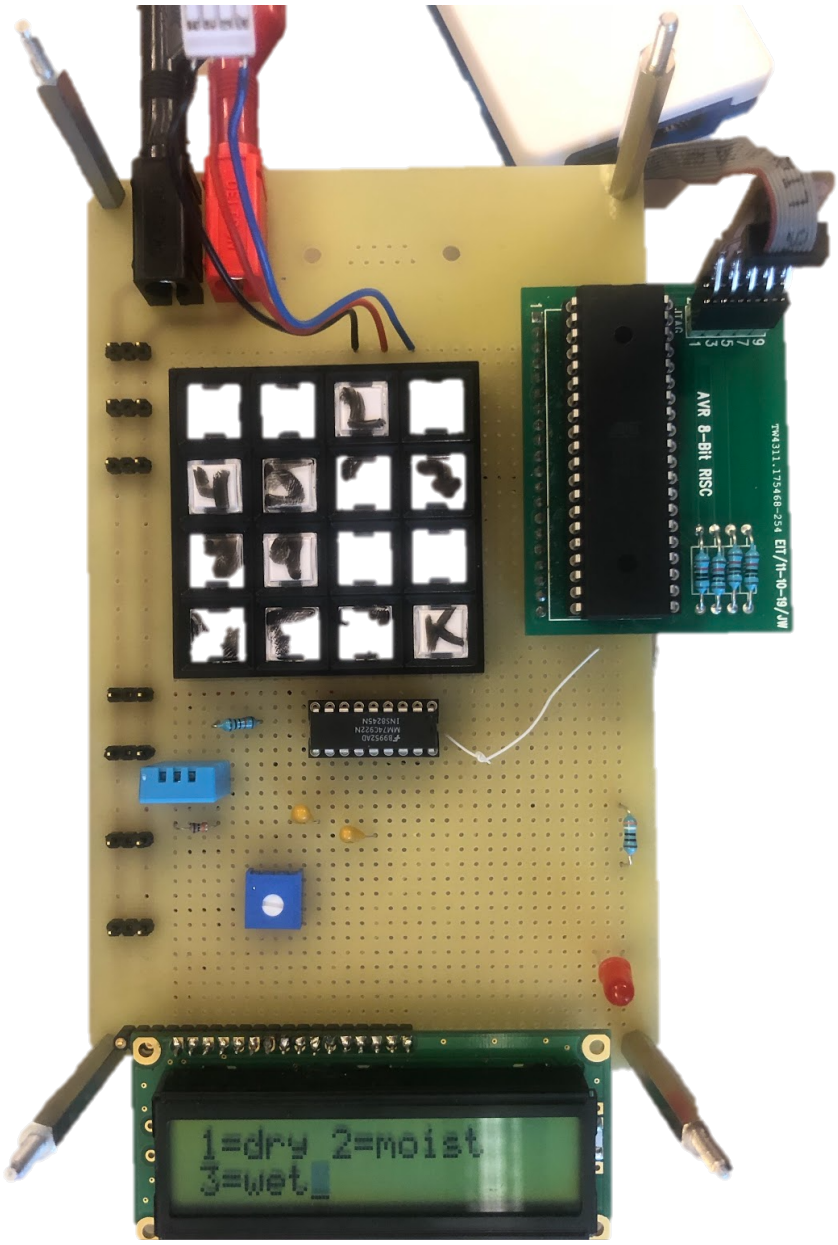
Rafiki Wa Kupanda

EITF11, Digitala projekt
VT18

Linnea Håkansson, Anton Gunneberg, Ruben Schultz



Lunds Tekniska Högskola
Institutionen för elektro- och informationsteknik
Handledare: Bertil Lindvall och Christoffer Cederberg
2018-05-21

Abstract

Rafiki Wa Kupanda - Or “Friend of plant” as it is called on the English market, is what every home has been missing. Our innovation combines the raw power of the top of the line processor, ATmega16, with the alphanumerical display SHARP DotMatrix to make sure you get a crystal clear view of what your plant needs in real time. The settings are adjustable to meet your plant’s needs, so you can be sure that no plant has to suffer at your watch.

Innehållsförteckning

1. Inledning 1
2. Kravspecifikation 1
3. Hårdvara 1
    3.1 Processor - ATmega16 1
    3.2 Display - SHARP DotMatrix 1
    3.3 Keypad 1
    3.4 Key Encoder 2
    3.5 Fuktsensor - Moisture Sensor v1.3 2
    3.6 Fukt- och temperatursensor - DHT11 2
    3.7 JTAG - Atmel JTAG ICE 2
    3.8 Lysdiod 2
    3.9 Övrigt 2
4. Mjukvara 2
5. Utförande 3
    5.1 Planering 3
    5.2 Montering 3
    5.3 Kodning och testning 3
6. Resultat 3
7. Diskussion och slutsats 3
8. Referenser 4
9. Bilagor 5
    9.1 Bilaga 1: Kopplingsschema 5
    9.2 Bilaga 2: Källkod 6

1. Inledning

I kursen Digitala Projekt EITF11 fick vi som uppgift att konstruera en prototyp av något slag för att illustrera industriellt utvecklingsarbete. Som start på arbetet fick vi själva bestämma vad prototypen skulle ha för funktioner. Vi valde att bygga en modell som vi kallar Rafiki Wa Kupanda, en växtvårdare. Denna består bland annat av en LCD-skärm som visar temperatur och fuktighet, en fuktighetsmätare för jord, en temperatur- och fuktighetssensor för luft och ett tangentbord som används för att göra diverse inställningar. Arbetsprocessen som ledde fram till den färdiga prototypen var både frustrerande och utmanande men också väldigt givande.

Denna rapport innehåller en kravspecifikation som tydliggör projektets mål och riktlinjer. Därefter följer korta beskrivningar av både hård- och mjukvara. Utöver detta beskrivs arbetsprocessen och till sist förs en diskussion om arbetet och dess resultat.

2. Kravspecifikation

Nedan följer de krav som prototypen ska uppfylla:

1. Modellen ska kunna mäta fuktighet i jord.
2. Modellen ska kunna mäta luftfuktighet och temperatur.
3. Det ska finnas en varningslampa som talar om när växten behöver vattnas.
4. Modellen ska ha en LCD-skärm som talar om hur växten mår. Den ska:
    a. Tala om vilken temperatur det är.
    b. Tala om hur fuktig luften är.
    c. Tala om vad som är fel när lampan lyser, t.ex. om jorden är för torr och behöver vattnas.

3. Hårdvara

3.1 Processor - ATmega16

ATmega16 är en MPU (MicroProcessing Unit) med 40 pinnar och kan ta emot digitala signaler samt omvandla analoga signaler till digitala med hjälp av processorn A/D-konverterare.

Processorn är kopplad till alla andra komponenter i prototypen då den styr inputs och outputs av signaler. Processorn programmeras i programmeringsspråket C med hjälp av en Atmel JTAG ICE, som gör att den vet hur den ska hantera in och utsignaler vid olika scenarion.

3.2 Display - SHARP DotMatrix

SHARP DotMatrix är en alfanumerisk LCD-display där kontrasten kan justeras genom en justerbar resistor. Displayen kopplas till processorns digitala Input/Output-pinnar samt till spänningsmatning och jordning.

3.3 Keypad

En keypad med 4x4 knappar används för att användaren ska kunna justera modellens inställningar efter sina behov.

3.4 Key Encoder

För att tolka signalerna från knappsatsen används en Key encoder som gör det möjligt att kunna koda vektorerna av SPST-switchar.

3.5 Fuktsensor - Moisture Sensor v1.3

Fukten mäts genom att två metallplattor med en mellanliggande spänning sticks ner i jord varpå resistansen mellan de två mäts. En fuktig jord leder ström bättre och ger därför en lägre resistans. Eftersom signalerna från fuktsensorn är analoga skickas de in i processorn till ben 40, som är en A/D-konverterare.

3.6 Fukt- och temperatursensor - DHT11

DHT11 använder en kalibrerad digital signal som output för temperaturen och luftfuktigheten. Sensorn består av två delar, en termistor och en kapacitiv fuktsensor. Modellen innehåller även en A/D-konverterare som omvandlar de analoga signalerna till digitala innan de skickas in till processorn, därför kan insignalen till processorn kopplas till en digital I/O-port. Processorn tar emot ettor och nollor som representeras av två olika insignaler med varierande tid med hög respektive låg signal. Dessa läses sedan av och kontrolleras med hjälp av en paritetsbit med vars hjälp man räknar ut en checksum för att kontrollera att informationen mottagits korrekt. Den binära insignalen omvandlas sedan till ett decimalt luftfuktighetsvärde i procent och en temperatur med enhet celsius.

3.7 JTAG - Atmel JTAG ICE

För att kunna exekvera mjukvaran behövs en JTAG som överför koden som programmeras på en dator i programmeringsspråket C till processorn.

3.8 Lysdiod

En lysdiod med rött ljus används för att varna användaren när något inte står rätt till. Dioden är kopplad till en resistor på 10kΩ.

3.9 Övrigt

I övrigt så användes resistorer, kondensatorer och diverse sladdar.

4. Mjukvara

Mjukvaran programmerades i C i programmet Atmel Studio 7.

5. Utförande

5.1 Planering

Efter en del efterforskning och idéspånande så beslutades att en växtvårdare skulle byggas. Innan det fysiska arbetet satte igång togs en kravspecifikation fram för att tydliggöra vad målet med projektet var. Utifrån specifikationen valdes sedan de nödvändiga komponenterna till prototypen ut och ett kopplingsschema ritades. För att se till att inget var fel i detta tidiga skede av projektet studerades även komponenternas datablad, för att i största möjliga mån undvika att få problem efter monteringsfasen. Efter att kopplingsschemat godkänts av projektets handledare Bertil Lindvall inleddes arbetets nästa fas, monteringen.

5.2 Montering

I detta steg så hanterades den hårdvara som skulle användas. Komponenter kopplades ihop enligt det kopplingsschema som tagits fram. Sladdar virades och vissa delar, såsom LCD-skärmen, löddes fast.

5.3 Kodning och testning

När hårdvaran var färdigmonterad påbörjades testningen. Först testades att det inte fanns några uppenbara monteringsfel som kunde leda till kortslutning eller att komponenter saknade strömtillförsel. När en spänning på 5 volt kopplats in och det verifierats att samtliga delar tycktes fungera i grova drag fortsatte arbetet med att de olika komponenternas funktioner undersökas så att de fungerade. Det visade sig att encodern inte fungerade som förväntat. Problemet var att ett av encoderns pinnar varierade mellan högt och lågt utan uppenbar anledning och för att lösa detta kopplades ett extra motstånd in. Även fukt- och temperatursensorn DHT11 behövde kopplas om. Denna drogs då till PD3 (INT1) vilket gjorde att avbrott kunde användas. I övrigt fungerade kopplingarna bra och med hjälp av databladen för respektive komponenter kunde koden för de önskade funktionerna skrivas.

6. Resultat

Modellen blev en helt funktionsduglig produkt som kan läsa av fukt i jord och luft samt lufttemperatur. Den är interaktiv och tillåter användaren att anpassa önskad temperatur och fuktighet. En varningslampa är installerad för att snabbt dra till sig uppmärksamhet från användaren om något inte står rätt till. Lyser lampan finns det information på skärmen om vad som ska göras, om den behöver mer eller mindre vatten eller en miljö med högre eller lägre temperatur.

7. Diskussion och slutsats

Att programmera var något vi kände oss bekväma med innan kursen, och den stora utmaningen trodde vi skulle bli att koppla hårdvaran rätt och att koden skulle skrivas i ett för oss nytt språk: C. Vår uppfattning om detta var, som så mycket annat under arbetets gång, helt fel. Det svåra var istället att koppla ihop mjukvara med hårdvara och förstå hur koden mer handgripligt påverkade hur våra enheter agerade. Det är svårt att uppskatta komplexiteten av ett problem då ens kunskap inom området är bristfälligt, vilket vi ofta märkte under arbetet.

Ett av de svåraste problemen som vi stötte på var när vi skulle hantera signalen från DHT11, vår temperatur- och luftfuktighetsmätare. Den första enheten var defekt och fick bytas ut. När den nya enheten var på plats var kunde vi börja klura på hur signalen skulle läsas av på bästa sätt. Efter en del diskussion tycker vi att vi relativt självständigt kom fram till en bra lösning och tog oss över hindret.

Slutligen så lyckades vi ta fram en fullt funktionell prototyp och med den har vi också fått grundläggande kunskaper inom ämnet och stor förståelse för utmaningarna med produktutveckling.

8. Referenser

Datablad för processor, *ATmega16*
High-performance AVR 8-bit Microcontroller
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf

Datablad för LCD-skärm, *SHARP Dot-Matrix*
LCD Units Alfanumerisk teckendisplay
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf

Datablad för Key Encoder,
16-Key Encoder
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Periphery/Other/MM54C922.pdf

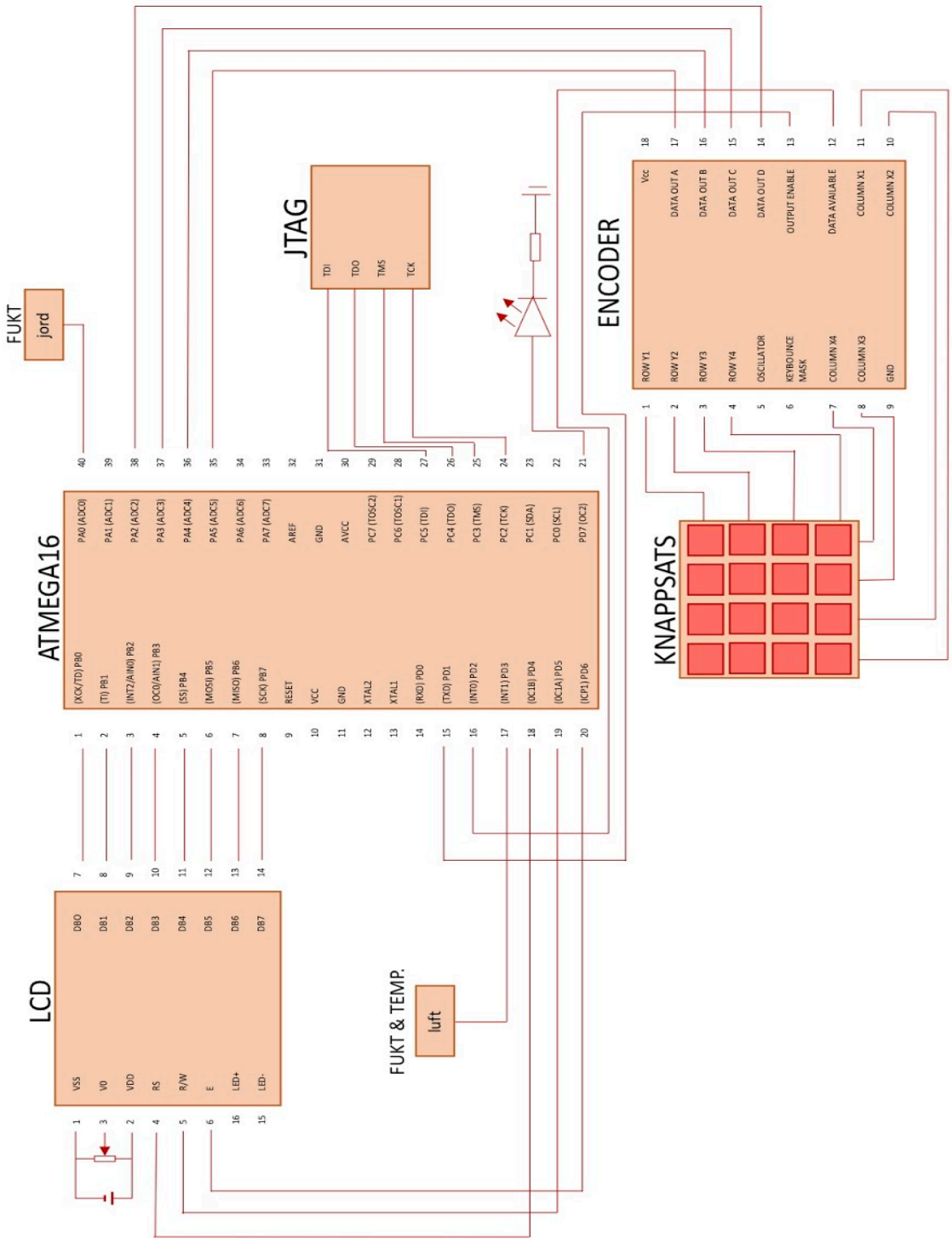
Datablad för temperatur- och fuktsensor, *DHT11 Humidity and Temperature Sensor*
http://robocraft.ru/files/datasheet/DHT11.pdf

https://akizukidenshi.com/download/ds/aosong/DHT11.pdf

Video om fuktsensor, *Moisture Sensor v1.3*
https://www.youtube.com/watch?v=rOOv-GF28n4

9. Bilagor

9.1 Bilaga 1: Kopplingsschema



9.2 Bilaga 2: Källkod

```
/*
 * GccApplication2.c
 *
 * Created: 2018-05-15 10:33:46
 * Author : ine15rsc
 */

#define F_CPU 8000000UL
#define set_bit(PORT,px)            (PORT |= _BV(px))
#define clear_bit(PORT,px)          (PORT &= ~_BV(px))
#define _BV(bit)                                (1<<(bit))

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

volatile uint16_t count;
unsigned int setTemp_enable=0;
unsigned int setMoist_enable=0;
unsigned int readMoist_enable=0;
unsigned int temperature;
unsigned int humidity;
unsigned int dhc[150];
unsigned char realTemp[4];
unsigned char realHum[4];
unsigned int write_enable=0;
unsigned char keytest;
unsigned int minTemp;
unsigned int minMoist;
unsigned char moistVal;
unsigned char LCD_clear = 0b00000001;
unsigned int keyVal;
unsigned int tempVal;
unsigned int new_input;
unsigned int keyCode;
unsigned int dhc_counter= 0;
unsigned int parity[6];
unsigned int moistSet;

void command_LCD(char cmd){
        PORTD |= _BV(PD6); // E hög
        PORTD &= ~_BV(PD4); // RS låg
        PORTD &= ~_BV(PD5); // R/W låg
        PORTB = cmd;
        _delay_ms(5);
        PORTD &= ~_BV(PD6); // E låg
        //_delay_ms(5);
        PORTD |= _BV(PD6); // E Hög

}

void writeData(char d){
        PORTD |= _BV(PD6); // E hög
        PORTD |= _BV(PD4); // RS hög
```

```
	PORTD &= ~_BV(PD5); // R/W låg
	PORTB = d;
	_delay_ms(10);
	PORTD &= ~_BV(PD6); // E låg
	PORTD |= _BV(PD6); // E hög

}

void write_String(char list[]){
	int i = 0;
	while(list[i] !=NULL){
		writeData(list[i]);
		i++;
	}

}

void initDHC(){

	MCUCR |= _BV(ISC11);
	MCUCR &= ~_BV(ISC10);



	// set up timer with prescaler = 8
	TCCR1B |=_BV(CS11);
	count= 0;
}

void DHC_run(){

	ADCSRA &= ~_BV(ADIE);

	DDRD |= _BV(PD3);

	PORTD |= _BV(PD3);

	PORTD &= ~_BV(PD3);
	_delay_ms(25);
	PORTD |= _BV(PD3);
	DDRD &= ~_BV(PD3);
	_delay_us(80);
	TCNT1 = 0;
	GICR |= _BV(INT1);

}

void DHC_translate(){
	ADCSRA |= 1<<ADIE;
	int c = 0;
	dhc[0]=0;
	for(int i=0; i<5; i++){
		for(int e = 0; e<8;e++){
			if(dhc[c]==1){
				parity[i]|= _BV(7-e);
			}
			else if(dhc[c]==0){
				parity[i] &= ~_BV(7-e);
```

```
                        }
                        c++;
                }
        }

        if((parity[0]+parity[1]+parity[2]+parity[3]) == parity[4]){
                if((parity[0]==0)||(parity[2]==0)){
                                DHC_run();
                                }
                else{
                        humidity=parity[0];
                        temperature=parity[2];
                        dhc_counter=0;
                        ADCSRA |= 1<<ADIE;
                }
        }
        else{

                DHC_run();

        }
}

void startADC(){

        //starta prescaler - klocka dividera 1 000 000 med 50-200 khz
för att klockan ska vara rätt inställd, prescaler mellan 5 och 20
        //vi använder 16 prescaler - se datablad
        ADCSRA |= 1 << ADPS2;

        // väljer om 8bit eller 10bit
        //ADMUX |= 1<<ADLAR;

        //Voltage reference mha AVCC vid AREF pinne
        ADMUX  |= 1<<REFS0;

        //Enable interrupt
        ADCSRA |= 1<<ADIE;

        //startar ADC
        ADCSRA |= 1<<ADEN;
        //conversion
        ADCSRA |= 1<<ADSC;
}

void init(){
        DDRD = 0b01110010;
        DDRB = 0b11111111;

        command_LCD(0b00111100); //function set
        command_LCD(0b00001111);
        command_LCD(LCD_clear); //clear display



        char list[9] = "Welcome";
        write_String(list);
        _delay_ms(1000);
```

```
        command_LCD(LCD_clear);

        DDRA = 0b00000000;
        PORTD =0b00000000;
        GICR |= _BV(INT0);
        MCUCR |=_BV(ISC01);
        MCUCR |=_BV(ISC00);
        startADC();
        initDHC();
        sei();
}

void set_main_screen(){

        write_String("Temperature:  ");

        write_String("C");
        command_LCD(0b11000000);
        write_String("Humidity: ");
        write_String(" %");

}

void setTemp(){
        command_LCD(LCD_clear);
        command_LCD(0b00001111);

        char list2[19] = "Set Temperature:";


        write_String(list2);
        command_LCD(0b11000000);
        int v[2];
        v[1]= 100;

        write_enable = 1;
        new_input=0;
        for(int f = 0; f<2;f++){

                while(new_input== 0){
                }
                new_input = 0;
                if(keyVal!=10){
                        v[f] = keyVal;
                }
        }
        write_enable = 0;
        while(keyVal!=10){
        }
        if(v[1]!=100){
                tempVal = v[0]*10+ v[1];
        }
        else{
                tempVal = v[0];
        }
        _delay_ms(500);
        command_LCD(LCD_clear);
```

```
}

void setMoist(){
        command_LCD(LCD_clear);
        char list[16] = "Set Moisture";
        write_String(list);
        _delay_ms(1000);
        command_LCD(LCD_clear);
        char list2[19] = "1=dry 2=moist";
        write_String(list2);
        command_LCD(0b11000000);
        char list3[8] = "3=wet";
        write_String(list3);
        write_enable = 2;
        while (!((keyVal==1)||(keyVal==2)||keyVal==3)){
                while(new_input== 0){
                }
                new_input = 0;
        }
        moistSet = keyVal;
        write_enable = 0;
        while(keyVal!=10){
        }
        if(moistSet==1){


                minMoist=20;
                char list4[6] = "DRY";
                command_LCD(LCD_clear);
                write_String(list4);
        }
        if(moistSet==2){


                minMoist=100;
                char list4[9] = "MOIST";
                command_LCD(LCD_clear);
                write_String(list4);
        }
        if(moistSet==3){


                minMoist=500;
                char list4[6] = "WET";
                command_LCD(LCD_clear);
                write_String(list4);
        }
        _delay_ms(500);
        command_LCD(LCD_clear);
}

char button(int val){
        switch(val){
                case 0b00000000:
                keyVal= 1;
                if((write_enable ==
1)||(write_enable==2)||(write_enable==3)){
```

```
                writeData('1');
            }
            return '1';
            case 0b00100000:

            keyVal= 2;
            if( (write_enable ==
1)||(write_enable==2)||(write_enable==3)){
                writeData('2');
            }
            return '2';
            case 0b00010000:
            keyVal= 3;
            if((write_enable ==
1)||(write_enable==2)||(write_enable==3)){
                writeData('3');
            }
            return '3';
            case 0b00110000:
            keyVal= 4;
            if(write_enable == 1||(write_enable==3)){
                writeData('4');
            }
            return '4';
            case 0b00001000:
            keyVal= 5;
            if(write_enable == 1||(write_enable==3)){
                writeData('5');
            }
            return '5';
            case 0b00101000:
            keyVal= 6;
            if(write_enable == 1||(write_enable==3)){
                writeData('6');
            }
            return '6';
            case 0b00011000:
            keyVal= 7;
            if(write_enable == 1||(write_enable==3)){
                writeData('7');
            }
            return '7';
            case 0b00111000:
            keyVal= 8;
            if(write_enable == 1||(write_enable==3)){
                writeData('8');
            }
            return '8';
            case 0b00000100:
            keyVal= 9;
            if(write_enable == 1||(write_enable==3)){
                writeData('9');
            }
            return '9';
            case 0b00100100:
            keyVal= 0;
            if(write_enable == 1||(write_enable==3)){
                writeData('0');
```

```
                }
                return '0';
                case 0b00010100:
                keyVal= 10;

                return 'k';

                case 0b00110100:
                if(write_enable==3){
                readMoist_enable=1;
                }
                return;

                case 0b00001100:
                return;
                case 0b00101100:
                return;
                case 0b00011100:
                if(write_enable==3){
                setTemp_enable=1;
                }
                return;

                case 0b00111100:
                if(write_enable==3){
                        _delay_ms(50);
                setMoist_enable=1;
                }
                return;

                default :
                return;
        }
}

void readMoist(){
        _delay_ms(30);
        command_LCD(0b1000000);
        command_LCD(LCD_clear);

        if(moistSet == 1){
                if(moistVal<20){
                        unsigned char list[10] = "Vattna";
                        write_String(list);
                        _delay_ms(1000);
                }
                else{

                        unsigned char list[6] ="Bra";
                        write_String(list);
                        _delay_ms(1000);
                }
        }
        if(moistSet == 2){
                if(moistVal<100){
                        unsigned char list[10] = "Vattna";
                        write_String(list);
                        _delay_ms(1000);
```

```
                }
                else{

                        unsigned char list[6] ="Bra";
                        write_String(list);
                        _delay_ms(1000);
                }
        }
        if(moistSet == 3){
                if(moistVal<500){
                        unsigned char list[10] = "Vattna";
                        write_String(list);
                        _delay_ms(1000);
                }
                else{

                        unsigned char list[6] ="Bra";
                        write_String(list);
                        _delay_ms(1000);
                }

                }
        command_LCD(LCD_clear);
}

int main(void){
        init();
        setTemp();
        command_LCD(LCD_clear);
        setMoist();
        command_LCD(LCD_clear);
        DHC_run();
        _delay_ms(20);

        if(dhc_counter>38){
                dhc_counter=0;
                DHC_translate();
        }

        write_String("Temperature:  ");


        write_String("C");
        command_LCD(0b11000000);
        write_String("Humidity: ");

        write_String(" %");

        while(1){
                startADC();
                write_enable=3;
                if(setMoist_enable==1){
                        setMoist();
                        setMoist_enable=0;
                        set_main_screen();
                }
                if(setTemp_enable==1){
                        setTemp();
```

```
                        setTemp_enable=0;
                        set_main_screen();
                }
                if(readMoist_enable==1){
                        readMoist();
                        readMoist_enable=0;
                        set_main_screen();
                }
                itoa(temperature, realTemp,10);

                command_LCD(0b00001100);

                command_LCD(0b10001100);

                write_String(realTemp);
                _delay_ms(200);
                command_LCD(0b11001001);
                itoa(humidity, realHum, 10);
                write_String(realHum);
                DHC_run();
                if(dhc_counter>38){
                        dhc_counter=0;
                        DHC_translate();

                }

                command_LCD(0b00001100);
                if((moistVal<minMoist)||(tempVal>temperature)){
                        DDRD |=_BV(PD7);
                        PORTD |=_BV(PD7);
                }
                else{
                        DDRD &=~_BV(PD7);
                }
        }
}
//ISR - Interrupt service routine
ISR(ADC_vect){
        int res = ADC;
        moistVal=res;
        ADCSRA |= 1<<ADSC;
}

ISR(INT0_vect){
        GIFR |= _BV(INTF0);
        _delay_ms(30);
        new_input = 1;
        button(PINA & 0b00111100);
        _delay_ms(30);
}

ISR(INT1_vect){

        if(TCNT1>70&&TCNT1<90){
                dhc[dhc_counter]=0;
                TCNT1=0;
                dhc_counter++;
```

```
        }
        else if ((TCNT1>90) && (TCNT1 <140)){
                TCNT1=0;
                dhc[dhc_counter]=1;
                dhc_counter++;
        }
        GIFR |= _BV(INTF1);

}
```