Water Buddy - din växts bästa vän

Caroline Ekelund
Amanda Nystedt
Sofia Ottosson
Moritz Rübner

under handledning av
Christoffer Cederberg och Georgij Michaliutin
Institutionen för Elektro- och informationsteknik
Lunds Tekniska Högskola

EITF12 Digitala Projekt
27 maj, 2022

Innehållsförteckning

1. Inledning

Målet med projektet var att ta fram en valfri prototyp för vidareutveckling som skulle konstrueras, byggas och testas. Syftet var att illustrera industriellt utvecklingsarbete. Vi valde att utveckla en prototyp för automatisk bevattning eftersom samtliga gruppmedlemmar ibland hade svårt att hinna med att ta hand om sina växter. Växtvattnaren Water Buddy fungerar för flera olika typer av växter då den vattnar efter jordens fuktighetsnivå och växtens behov.

Rapporten beskriver produkten och metoden för framtagning, samt en redogörelse för dess hårdvara och mjukvara.

1.1 Kravspecifikation

Water Buddy ska användas för att automatiskt förse en växt med vatten när detta behov finns. I projektets början togs en kravspecifikation fram som innehöll följande krav:

- WaterBuddy ska känna av jordens fuktighetsnivå i en krukväxt.
- Användaren kan ställa in en önskad fuktighetsnivå för växten och Water Buddy ger sedan vatten upp till önskad fuktnivå.
- Water Buddy ska kunna meddela användaren när tanken börjar ta slut och behöver fyllas på.
- Water Buddys display ska uppdatera användaren om växtens fuktighetsnivå.
- Användaren ska kunna vattna växten manuellt genom ett knapptryck.
- Water Buddy ska kunna användas för att vattna flera blommor samtidigt.

2. Teori

2.1 Hårdvara

2.1.1 Processor

ATMega16 High-performance AVR 8-bit Microcontroller användes som processor för att styra och programmera de olika komponenterna. ATMega16 innehåller JTAG som används bland annat för testning och felsökning.

2.1.2 Jordfuktighetsmätare

Jordfuktighetsmätaren användes för att mäta fuktighetsgraden i krukans jord.

2.1.3 Vattentank med ventil

För att förse växten med vatten användes en vattentank i form av en uppochnervänd PET-flaska och en ventil. När växten ska vattnas öppnas ventilen och vattnet kan flöda ut. Ventilen som användes är en 12V

magnetventil i koppar.

2.1.4 Flödesmätare

Flödesmätaren användes för att mäta hur mycket vatten som försvinner från tanken vid varje vattning. Genom detta kan tankens återstående vätskenivå beräknas för att därefter kunna varna användaren när vattnet börjar ta slut.

2.1.5 Display

Displayen användes för att uppdatera användaren. Den informerar om att vattning pågår samt om den aktuella fuktighetsgraden i jorden. Displayen som användes var en 128x128 Oled display.

2.1.6 Knappsats

Genom knappsatsen kan användaren ställa in önskad fuktighetsnivå samt vid vilken fuktnivå som växten ska vattnas upp till den önskade fuktnivån. Det ska också gå att starta bevattningen manuellt.

2.1.7 Lysdiod

Lysdioden används för att varna användaren om att tanken är tom. Lysdioden lyser rött då det är dags att fylla på tanken.

2.1.8 Högtalare

Högtalaren spelar en jingel när växten har vattnas och uppnått rätt fuktighetsnivå samt då vattentanken är tom och behöver fyllas på.

2.2 Mjukvara (programmering)

Vid programmeringen av Water Buddy användes programspråket C. Koden hanterades i programvaran Atmel Studio 7 och JTAG användes för felsökning av hårdvaran och mjukvaran. Programmet består av en main-metod som innefattar de metoder som behövs för att driva växtvattnaren. Det finns metoder för att vattna upp till önskad nivå, skriva meddelanden till displayen, rita ut stapeln som visar fuktnivån, spela ljud från högtalaren etc. Programkoden återfinns i bilaga 7.2.

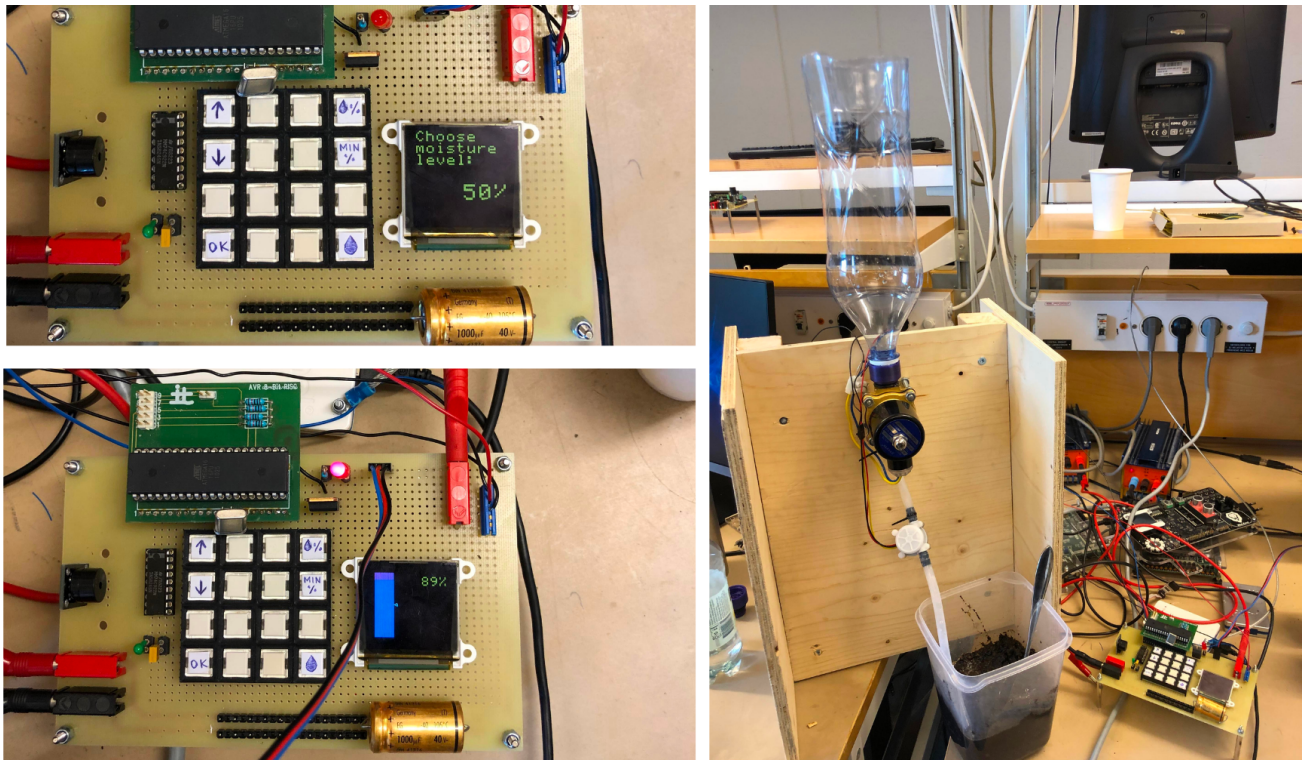
3. Utförande

Projektet började med att projektgruppen fattade ett beslut om vilken prototyp som skulle konstrueras och därefter togs en kravspecifikation fram. Kravspecifikationen specificerar de krav som Water Buddy ska uppfylla. Därefter undersöktes vilka komponenter som skulle krävas för att uppfylla kraven, vilket mynnade ut i en grov skiss av ett kopplingsschema. Med hjälp av respektive komponents datablad togs

därefter ett mer detaljerat kopplingsschema fram som ritades upp med hjälp av ritprogrammet Excalidraw (se bilaga 7.1).

Därefter påbörjades sammansättningen av hårdvaran. Komponenterna kopplades ihop enligt kopplingsschemat med hjälp av en verktygslåda som tillhandahölls av handledaren. Det första som kopplades på var lysdioden och knappsatsen. Parallellt med att hårdvaran konstruerades programmerades mjukvaran i Atmel Studio 7, för att se att respektive komponent fungerar enligt kravspecifikationen. Först programmerades knappsatsen så att lysdioden sattes på och stängdes av vid knapptryck. Displayen kopplades in och programmerades så att ett en sträng kunde skickas och visas på skärmen. Ventilen och flödesmätaren kopplades på och en konstruktion i trä byggdes för att montera på ventilen och en vattentank i form av en PET-flaska med tillhörande slang, se bilder nedan. Displayen programmerades så att användaren kan ställa in önskad fuktnivå och en stapel som visar nuvarande och önskad fuktnivå. Det sista som monterades på var högtalaren. Den programmerades så att en varningssignal spelas när vattnet i tanken är slut och en kort jingel då växten vattnas.

Då konstruktionen av hårdvaran och mjukvaran var klar testades prototypen genom att använda jord med olika fuktighet och observera att växten bevattnas upp till den valda fuktnivån.



Bilderna visar konstruktionen samt vad displayen visar i utgångsläget med en stapel som visar den nuvarande fuktighetsnivån samt när man ska välja önskad fuktighetsnivå.

4. Resultat

De krav som var specificerade enligt kravspecifikationen i början av projektet var följande:

1. Water Buddy ska känna av jordens fuktighetsnivå i en krukväxt.

2. Användaren kan ställa in en önskad fuktighetsnivå för växten och Water Buddy ger sedan vatten upp till önskad fuktnivå.
3. Water Buddy ska dessutom kunna meddela användaren när tanken börjar ta slut och behöver fyllas på.
4. Water Buddy display ska uppdatera användaren om växtens fuktighetsnivå
5. Användaren ska kunna vattna växten manuellt genom ett knapptryck.
6. Water Buddy ska kunna användas för att vattna flera blommor samtidigt.

Under arbetets gång tillkom tre krav för att öka projektets komplexitet:

7. Användaren ska kunna ställa in vid vilken fuktighetsnivå bevattning ska börja.
8. Water Buddys display ska visa jordens fuktighetsnivå i form av en stapel som anger fuktigheten i procent.
9. Water Buddy ska spela en jingel när bevattningen är klar och växten har uppnått önskad fuktnivå.

Samtliga krav, förutom krav 6, uppfylldes av den färdiga produkten. Water Buddy kan framgångsrikt användas för att bevattna en växt. Detta genom att Water Buddy registrerar jordens fuktighet och därmed anpassar vattningen efter den manuellt inställda fuktighetsnivån. Water Buddy kan även användas för att manuellt vattna växter genom att använda knappsatsen. Water Buddy signalerar med en röd lysdiod samt en ljudsignal när vattentanken är tom. Water Buddys display meddelar dessutom växtägaren kontinuerligt hur fuktig jorden är i procent samt om vattning sker eller inte. När växten har uppnått önskad fuktnivå spelar Water Buddy en jingel för att meddela ägaren om att växten är färdigvattnad.

Krav 6, som innebar att Water Buddy skulle kunna användas för att vattna flera blommor samtidigt, bortsågs från under arbetets gång då det ansågs att detta gjorde arbetet för komplext och tidskrävande.

5. Diskussion

Projektet med utvecklingen av Water Buddy kunde genomföras enligt plan och de krav som ansågs rimliga uppfylldes. För att öka komplexiteten blev vi ombedda av handledare att lägga till en stapel på displayen som minskade i takt med att vätskenivån minskade och visade den nuvarande vattennivån. Vi skulle även lägga till en högtalare som meddelar användaren att tanken är tom, och ger en liten signal när växten är färdigvattnad. Svårigheter som gruppen har stött på är bland annat att få displayen att fungera enligt kraven och att göra beräkningar för att omvandla fuktnivå från sensorn till procent. De största svårigheterna vi har behövt hantera är att vi inte hade någon tidigare erfarenhet varken programmering i C eller de flesta delarna av konstruktionsarbetet. Då mycket har varit nytt har vi behövt lägga mycket tid på att förstå grunderna i det vi har jobbat med.

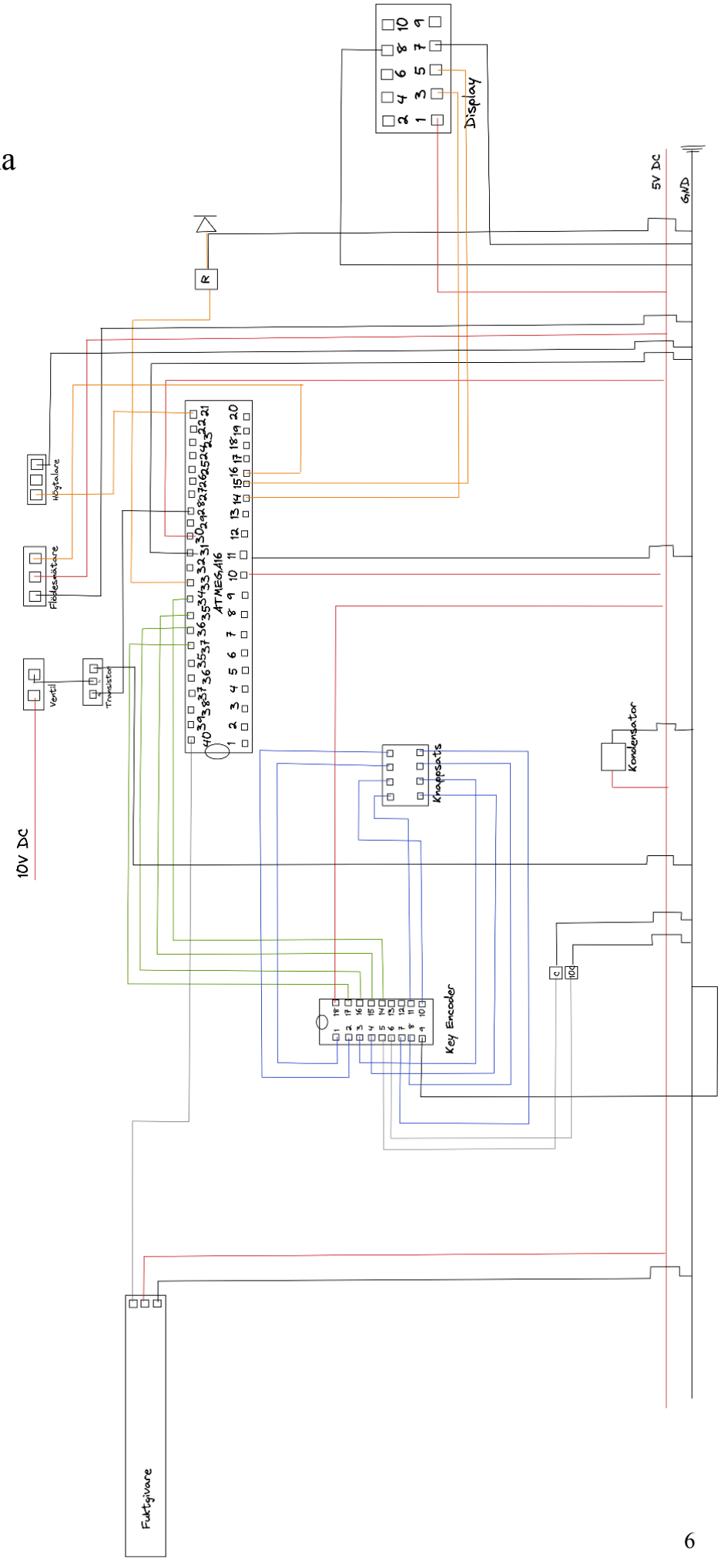
Vidare skulle produkten kunna utvecklas till att ha en nättare konstruktion samt kunna vattna flera växter samtidigt. Med en fuktighetsmätare med större noggrannhet hade vi kunnat vattna upp till önskad fuktighetsnivå med mer precision.

6. Slutsats

Projektet har gett oss ökad kunskap och förståelse kring hur är ett industriellt utvecklingsarbete går till. Det har stundtals varit svårt men också väldigt lärorikt och vi tar flera nya lärdomar med oss. Med det lyckade slutgiltiga resultatet i hand är vi nöjda och stolta över vår Water Buddy. Vi vill rikta ett stort tack till handledarna Christoffer Cederberg och Georgij Michaliutin.

7. Bilagor

7.1 Kopplingsschema



7.2 Programkod

main.c

```
/*
 * waterBuddy.c
 *
 * Created: 2022-04-08 15:13:37
 */
#define F_CPU 12000000UL
#include <avr/interrupt.h>
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include "oled.h"

#define LED                 (1 << PA7)
#define VALVE               (1 << PC6) // 0b0100 0000
#define SPEAKER             (1 << PD7)
//when prompting user for desired water level
#define LEVEL               0
//when prompting user for watering threshold
#define THRESHOLD           1

void ext_int1_init(void);
void ext_int0_init(void);
void adc_init(void);

uint16_t adc_start(void);
ISR(INT1_vect);
ISR(INT0_vect);

void oled_test(void);
void valve_test(void);
void led_test(void);
void speaker_failure(void);
void speaker_success(void);

void oled_plant(void);

void oled_prompt(int setting);
void oled_update(void);

void water_to_desired_level(void);

uint16_t moisture_air = 500;
uint16_t moisture_water= 260;

uint16_t flow_counter = 0;

uint16_t moisture_level, desired_moisture_level, moisture_level_threshold;

uint8_t moisture_percentage, desired_moisture_percentage, moisture_percentage_threshold;

volatile uint8_t btn_val, btn_flag;

int main(void)
{
        DDRA |= LED;
        DDRC |= VALVE;
        DDRD |= SPEAKER;

        ADMUX |= (1 << REFS0);
        ext_int1_init();
        ext_int0_init();
        adc_init();
        sei();

        init_uart_oled();
        _delay_ms(3000);
```

```
        oled_filled_rectangle(0, 0, 128, 128, 0x00);
        oled_disable_screensaver();

        //Prompt user for desired moisture level
        oled_prompt(LEVEL);
        _delay_ms(1000);
        //Prompt user for desired moisture threshold
        oled_prompt(THRESHOLD);

    while (1)
    {
                //Check water level
                moisture_level = adc_start();

                //Match water level with desired water level
                if (moisture_level > moisture_air) {
                        moisture_level = moisture_air;
                } else if (moisture_level < moisture_water){
                        moisture_level = moisture_water;
                }
                moisture_percentage = 100 - (100*(moisture_level-moisture_water))/(moisture_air-moisture_water);
                oled_update();

                //Water plant if lower than threshold
                if (moisture_percentage < moisture_percentage_threshold) {
                        water_to_desired_level();
                }

                //When button pressed
                //Buttons 0&4 higher/lower
                //Button 3 set desired water level
                //Button 7 set desired threshold
                //Button 15 water to desired water level
                //Button 12 for OK

                if (btn_flag == 1) {
                        if(btn_val == 3) {
                                oled_prompt(0);
                        } else if (btn_val == 7) {
                                oled_prompt(1);
                        } else if (btn_val == 15) {
                                water_to_desired_level();
                        }
                        btn_flag = 0;
                }
    }
}

void ext_int1_init () {
        MCUCR |= (1 << ISC11) | (1 << ISC10);
        GICR |= (1 << INT1);
}

void ext_int0_init () {
        MCUCR |= (1 << ISC01) | (1 << ISC00);
        GICR |= (1 << INT0);
}


void adc_init ()
{
        ADCSRA |= (1 << ADEN);
        ADCSRA |= (1 << ADPS1);
        ADCSRA |= (1 << ADPS2);
}

uint16_t adc_start()
{
        ADCSRA |= (1<<ADSC);
        while(ADCSRA & (1<<ADSC));
```

```
        return (ADC);
}

ISR(INT1_vect)
{
        btn_val = ((PINA & 0b01111000) >> 3);
        btn_flag = 1;
}

ISR(INT0_vect)
{
        flow_counter += 1;
}

void valve_test()
{
        while(1){
        PORTC |= VALVE;
        _delay_ms(1000);
        PORTC &= ~VALVE;
        _delay_ms(1000);
        }
}

void speaker_failure()
{
        for (int num = 0; num < 3; num++) {

                _delay_ms(100);
                for (int j = 0; j < 100; j++) {
                        PORTD |= SPEAKER;
                        _delay_ms(2);
                        PORTD &= ~SPEAKER;
                        _delay_ms(2);
                }
                _delay_ms(10);
                for (int k = 0; k < 150; k++) {
                        PORTD |= SPEAKER;
                        _delay_ms(3);
                        PORTD &= ~SPEAKER;
                        _delay_ms(3);
                }
                _delay_ms(100);
        }
}

void speaker_success()
{
        for (int i = 0; i < 300; i++) {
                PORTD |= SPEAKER;
                _delay_us(900);
                PORTD &= ~SPEAKER;
                _delay_us(900);
        }
        _delay_ms(100);

        for (int m = 0; m < 600; m++) {
                PORTD |= SPEAKER;
                _delay_us(800);
                PORTD &= ~SPEAKER;
                _delay_us(800);
        }


}

void oled_test(void)
{
        unsigned char text_width[] = {0xff, 0x7c, 0, 2};
        unsigned char text_height[] = {0xff, 0x7b, 0, 2};
        unsigned char text_color[] = {0xff, 0x7f, 0xff, 0x00};
        unsigned char clear_screen[] = {0xff, 0xd7};
```

```
        unsigned char move_cursor[] = {0xff, 0xe4, 0, 0, 0, 0};
        unsigned char text[] = {0x00, 0x06, 'H', '4', 'c', 'k', 'e', 'd', 0x00};

        oled_send_string(text_width, sizeof(text_width));
        oled_send_string(text_height, sizeof(text_height));
        oled_send_string(text_color, sizeof(text_color));
        oled_send_string(move_cursor, sizeof(move_cursor));
        oled_send_string(text, sizeof(text));
}


void oled_update(void)
{
        _delay_ms(1000);
        //hela balken
        oled_filled_rectangle(10, 10, 40, 110-moisture_percentage, 0xf034);

        //water level
        oled_filled_rectangle(10, 110-moisture_percentage, 40, 110, 0x001F);

        //desired moisture level
        oled_filled_triangle(41, 110-desired_moisture_percentage, 46, 110-desired_moisture_percentage+2, 46,
110-desired_moisture_percentage-2, 0xFFFF);

        //desired moisture threshold
        oled_filled_triangle(41, 110-moisture_percentage_threshold, 46, 110-moisture_percentage_threshold+2, 46,
110-moisture_percentage_threshold-2, 0xFFFF);


        unsigned char text_percentage[] = {0x00, 0x06, ' ', ' ', ' ', '%', 0x00};

        oled_set_text_width_height(2, 2);
        oled_move_cursor(1,5);
        _delay_ms(100);

        text_percentage[2] =(char) ((moisture_percentage/100 ) + 48);
        text_percentage[3] =(char) ((moisture_percentage % 100)/10 + 48);
        text_percentage[4] =(char) ((moisture_percentage % 10) + 48);

        if (text_percentage[2] == 48){
                text_percentage[2] = 32;
        }

        if (text_percentage[3] == 48){
                text_percentage[3] = 32;
        }


        oled_send_string(text_percentage, sizeof(text_percentage));
        oled_move_cursor(1,5);

}

//0 for level, 1 for threshold
void oled_prompt(int setting)
{
        uint8_t percentage =  50;
        unsigned char text_color[] = {0xff, 0x7f, 0xff, 0x00};
        unsigned char text_choose[] = {0x00, 0x06, 'C', 'h', 'o', 'o', 's', 'e', 0x00};
        unsigned char text_moisture[] = {0x00, 0x06, 'm', 'o', 'i', 's', 't', 'u', 'r', 'e', 0x00};
        unsigned char text_level[] = {0x00, 0x06, 'l', 'e', 'v', 'e', 'l', ':', 0x00};
        unsigned char text_threshold[] = {0x00, 0x06, 't', 'h', 'r', 'e', 's', 'h', 'o', 'l', 'd', ':', 0x00};
        unsigned char text_percentage[] = {0x00, 0x06, ' ', ' ', ' ', '%', 0x00};


        oled_set_text_width_height(2,2);
        oled_send_string(text_color, sizeof(text_color));
        oled_move_cursor(0,0);
        oled_send_string(text_choose, sizeof(text_choose));
        oled_move_cursor(1,0);
        oled_send_string(text_moisture, sizeof(text_moisture));
        oled_move_cursor(2,0);
```

```
text_percentage[2] =(char) ((percentage/100 ) + 48);
text_percentage[3] =(char) ((percentage % 100)/10 + 48);
text_percentage[4] =(char) ((percentage % 10) + 48);


switch (setting)
{
        case LEVEL:
                oled_send_string(text_level, sizeof(text_level));
        break;

        case THRESHOLD:
                oled_send_string(text_threshold, sizeof(text_threshold));
        break;
}


oled_set_text_width_height(3,3);
oled_move_cursor(3,2);
oled_send_string(text_percentage, sizeof(text_percentage));


//Let user choose level with up and down buttons
uint8_t ok_pressed = 0;

while (!ok_pressed){
        text_percentage[2] =(char) ((percentage/100 ) + 48);
        text_percentage[3] =(char) ((percentage % 100)/10 + 48);
        text_percentage[4] =(char) ((percentage % 10) + 48);

        if (text_percentage[2] == 48){
                text_percentage[2] = 32;
        }

        if (text_percentage[3] == 48 && text_percentage[2] != 48){
                text_percentage[3] = 32;
        }

        if (btn_flag == 1) {
                if(btn_val == 0 && percentage < 100) {
                        percentage = percentage + 5;
                        _delay_ms(100);
                } else if (btn_val == 4 && percentage > 0) {
                        percentage = percentage - 5;
                        _delay_ms(100);
                } else if (btn_val == 12) {
                        ok_pressed = 1;
                }
                btn_flag = 0;
                btn_val = 0;
                _delay_ms(300);
        }

        oled_move_cursor(3, 2);
        oled_send_string(text_percentage, sizeof(text_percentage));

        _delay_ms(300);

}

oled_filled_rectangle(0, 0, 128, 128, 0x00);

if (setting == 0){
        desired_moisture_percentage = percentage;
        desired_moisture_level = ((((percentage-100)*(moisture_air-moisture_water))/100)+moisture_water);
} else if (setting == 1) {
        moisture_percentage_threshold = percentage;
        moisture_level_threshold = ((((percentage-100)*(moisture_air-moisture_water))/100)+moisture_water);
}

percentage = 0;
```

```
        ok_pressed = 0;

        _delay_ms(200);

}

void water_to_desired_level()
{
        //valve_test();
        uint8_t water_empty = 0;

        if(water_empty){
                return;
        }

        uint8_t water_need = desired_moisture_percentage - moisture_percentage_threshold;

        PORTC |= VALVE;
        for (int i = 0; i < water_need; i++){
                _delay_ms(100);
        }
        PORTC &= ~VALVE;

        if(flow_counter < 5) {
                water_empty = 1;
        } else {
                water_empty = 0;
        }

        flow_counter = 0;

        //if flow meter gives 0, display error (needs more water)
        if (water_empty) {
                PORTA |= LED;
                speaker_failure();
        } else {
                PORTA &= ~ LED;
                speaker_success();
        }

}

void led_test()
{
        PORTA |= LED;
        _delay_ms(1000);
        PORTA &= ~LED;
        _delay_ms(1000);
}
```

oled.h

```
#define F_CPU 12000000UL
#include <avr/io.h>
#include <util/delay.h>

void init_uart_oled(void);
void oled_send_string(unsigned char* msg, int len);
```

```
void oled_send_char(unsigned char c);
void oled_filled_rectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color);
void oled_filled_triangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t x3, uint16_t y3, uint16_t color);
void oled_disable_screensaver(void);
void oled_move_cursor(uint16_t row, uint16_t column);
void oled_set_text_width_height(uint16_t width, uint16_t height);
```

oled.c

```
#include "oled.h"

void init_uart_oled()
{
	UBRRH = 0;
	UBRRL = 77;
	UCSRB |= (1 << RXEN) | (1 << TXEN);
	// the rest is default
}

void oled_send_char(unsigned char c)
{
	while (!(UCSRA & (1 << UDRE)));
	UDR = c;
}

void oled_send_string(unsigned char* msg, int len)
{
	for (int i = 0; i < len; i++){
		oled_send_char(msg[i]);
	}
	_delay_ms(10);
}

void oled_filled_rectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
{
	unsigned char rectangle[] = {0xff, 0xce,
		(unsigned char) (x1 >> 8), (unsigned char) x1,
		(unsigned char) (y1 >> 8), (unsigned char) y1,
		(unsigned char) (x2 >> 8), (unsigned char) x2,
		(unsigned char) (y2 >> 8), (unsigned char) y2,
		(unsigned char) (color >> 8), (unsigned char) color,};
	oled_send_string(rectangle, sizeof(rectangle));
	_delay_ms(100);
}

void oled_filled_triangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t x3, uint16_t y3, uint16_t color)
{
	unsigned char triangle[] = {0xff, 0xc9,
		(unsigned char) (x1 >> 8), (unsigned char) x1,
		(unsigned char) (y1 >> 8), (unsigned char) y1,
		(unsigned char) (x2 >> 8), (unsigned char) x2,
		(unsigned char) (y2 >> 8), (unsigned char) y2,
		(unsigned char) (x3 >> 8), (unsigned char) x3,
		(unsigned char) (y3 >> 8), (unsigned char) y3,
		(unsigned char) (color >> 8), (unsigned char) color,};
	oled_send_string(triangle, sizeof(triangle));
	_delay_ms(100);

}


void oled_disable_screensaver()
{
	unsigned char disable_screensaver[] = {0x00, 0x0c, 0, 0};
	oled_send_string(disable_screensaver, sizeof(disable_screensaver));
}

void oled_move_cursor(uint16_t row, uint16_t column)
{
	unsigned char move_cursor[] = {0xff, 0xe4,
		(unsigned char) (row >> 8), (unsigned char) row,
```

```
                    (unsigned char) (column >> 8), (unsigned char) column,
                    };
          oled_send_string(move_cursor, sizeof(move_cursor));
}

void oled_set_text_width_height(uint16_t width, uint16_t height)
{
          unsigned char text_width[] = {0xff, 0x7c,
                    (unsigned char) (width >> 8), (unsigned char) width};
          unsigned char text_height[] = {0xff, 0x7b,
                    (unsigned char) (height >> 8), (unsigned char) height};

          oled_send_string(text_width, sizeof(text_width));
          oled_send_string(text_height, sizeof(text_height));
}
```

8. Referenser

Datablad för processor, ATMega16. *High-performance AVR 8-bit Microcontroller (Complete)*. https://www.eit.lth.se/fileadmin/eit/courses/datablad/Processors/ATmega16.pdf

Datablad för display, 128x128 Oled display. *1.5" microOLED GOLDELOX Display uOLED-128G2*. https://www.eit.lth.se/fileadmin/eit/courses/datablad/Display/uOLED_128_G2_datascheet.pdf

Serial command reference. https://4dsystems.com.au/mwdownloads/download/link/id/31/.