Feeding Daisy

En automatisk blomvattnare

EITF11 – Lunds Tekniska Högskola

Grupp 12:
Pontus Akervall
Louise Landare
Anton Schölin

Handledare:
Bertil Lindvall

Introduktion

Grupp 12 har som vision att förenkla livet
för alla oss som tar växter för givet.
De flesta har någon gång råkat ta död på sin planta
då man med vattningen sig klantat.
Vi kan alla lida med denna sorg,
när man fått köpa sig en ny växt på närmsta torg.
Detta ska inte behöva hända igen,
en automatisk blomvattnare är lösningen, äntligen!

Kravspecifikation

Den automatiska blomvattnaren ska:

1. Önskad jordfuktighet ska kunna bestämmas av användaren
2. Kontinuerligt mäta nuvarande jordfuktighen
3. De uppmätta värdena ska kunna skickas till en dator genom seriell kommunikation
4. Baserat på mätningarna och den efterfrågade fuktigheten ska blomman vattnas med en konstant mängd vatten efter behov.

Hårdvara

Nedan beskrivs de komponenter som kommer krävas för projektet.

Processor

Vi kommer använda en 8-bitars ATMega16-processor på 8MHz. Den har 32 programmerbara Input/Output portar. Processorn kan programmeras i C på en dator med JTAG som interface och ska ha ett programmerbart minne på 16kB.

Temperatursensor

Temperatursensorn LM335 kommer användas som ger en spänning som är proportionell mot temperaturen i Kelvin. Vid 0 K ges utsignalen 0 volt och varje centigrad motsvarar 10mV som output. Signalen sänds till processorn som konverterar volt till temperatur.

LCD display

En SHARP Dot-Matrix LCD Units LM162XXX används som har plats för 32tecken som byggs upp med en punktmatris på 8 bitar vardera. Displayen visar tecknen på två rader.

Knappar

Två stycken brytare används som två knappar till att justera den önskade fuktigheten.

Servo

En servo används för att öppna ventilen till vattnet på en given signal.

Fuktsensor

H25K5 resistance humidity sensor mäter fuktigheten i jorden.

Övrigt

Utöver ovan beskriven hårdvara användes även:

- En NPN- transistor för att processorn ska kunna hantera en 12V spänning till servot.
- Fem stycken resistanser, en användes innan transistorn, två stycken användes till varsin knapp innan signalen leddes till jord, en användes till servot och slutligen användes en till temperaturmätaren.
- En potentiometer som justerar kontrasten på LCD-displayen.
- Två spänningsaggregat på 5V respektive 12V.

Mjukvara

Till en början sätts in- och utportar rätt och LCD-skärmen startas. Därefter startas en while-loop som kontinuerligt mäter fukten och väntar på signal från upp- och nerknapp. Om någon av knapparna aktiveras kommer den önskade fuktigheten ökas alternativt minskas med 100(?) och displayen kommer uppdateras till att visa det aktuella faktiska värdet i krukan samt det nya önskade värdet.

I while-loopen är även en räknare deklarerad som fungerar som en klocka, och då räknaren har nått till 15000 exekveras en if-sats som undersöker huruvida den önskade fuktnivån är högre än den uppmätta fuktnivån. Om så är fallet öppnas servot för ett genomflöde av vatten och stängs sedan igen efter 5sekunder. Därefter startas while-loopen igen med en nollställd räknare och det önskade samt aktuella värdet uppdateras på displayen.

Metod

Planering

När beslutet hade fattats att vi skulle göra en blomvattnare diskuterade vi fram vilka funktioner en sådan måste ha och skrev utefter det en kravspecifikation. Därefter togs en grov skiss fram där vi fastställde vilka komponenter som krävdes. Sedan följde en tid där vi läste in oss på komponenternas funktioner och specifikationer, främst genom information hämtad från Internet och datablad. Utifrån den informationen ritades ett kopplingsschema i datorprogrammet PowerLogic som blev grunden för såväl det praktiska arbetet som för det tekniska.

Hårdvarukonstruktion

Komponenterna sattes samman enligt kopplingsschemat. Hårdvarumässigt var vi tvungna att vänta ett tag på att få våra speciella komponenter, så som jordfuktighetsmätaren men även magnetventilen. Det visade sig att en del pinnar valts fel, exempelvis att PORT A är de enda pinnarna som kan läsa analoga signaler, varpå en del hårdvara fick kopplas om. Vid något tillfälle gick en pinne av och vid ett annat tillfälle som gick lödningar på magnetventilen sönder.

Initialt var tanken att den även skulle innehålla en temperaturgivare, vilket dock aktivt valdes bort. Dels på grund av platsbrist på skärmen men också för att det inte fyller någon primär funktion, då vattningsfunktionen inte beror av temperaturen.

Programmering

Med JTAG som interface mellan processorn och datorn kodades programmet i C i Atmel Studio. På så vis testades nu hur komponenterna fungerade ihop. LCD-skärmen visade sig vara mer ut av en utmaning än övriga komponenter, och krävde en del kunskap för att få att fungera som önskat. Det kan nämnas för den intresserade att default-läget på skärmen valdes enligt: Enabler som hög; Read/Write som hög samt Read Signal som low. Vidare programmerades flera andra saker för skärmen, så som automatisk radbrytning samt metoder för radförflyttning.

Kalibrering

När alla komponenter och processorn fungerade som önskat började vi med att fastställa hur många gånger while-loopen skulle genomföras mellan varje vattning. För att fastställa det utfördes empiriska tester av hur lång tid det tog för att utföra loopen olika antal gånger. Därefter undersökte vi hur vattningen går till. Det visade sig att ventilen släpper igenom vatten först vid ett tryck på 3 psi.

Resultat

Nedan presenteras de krav som i början av projektet bestämdes:

Den automatiska blomvattnaren ska:

1. *Önskad jordfuktighet ska kunna bestämmas av användaren*
2. *Kontinuerligt mäta nuvarande jordfuktighet.*
3. *De uppmätta värdena ska kunna skickas till en dator genom seriell kommunikation*
4. *Baserat på mätningarna och den efterfrågade fuktigheten ska blomman vattnas med en konstant mängd vatten efter behov.*

Utfall av kraven:

1. Användaren kan ställa in önskad jordfuktighet med hjälp av två knappar, en för att öka och en annan för att minska.
2. Systemet kan kontinuerligt mäta jordfuktighet, dock har ett intervall på ca 30 sekunder

bestämts mellan mätningarna, för att den inte behöver mäta oftare.

3. De uppmätta värdena kan skickas till en dator enligt kravet.
4. Baserat på mätningarna och given önskad fuktighet så skickas en signal om öppning till magnetventilen att vara öppen under en konstant tid. Som tidigare nämnts så fungerar magnetventilen inte om vattentrycket är för lågt, men den öppnas och stängs som den skall.

Diskussion och slutsatser

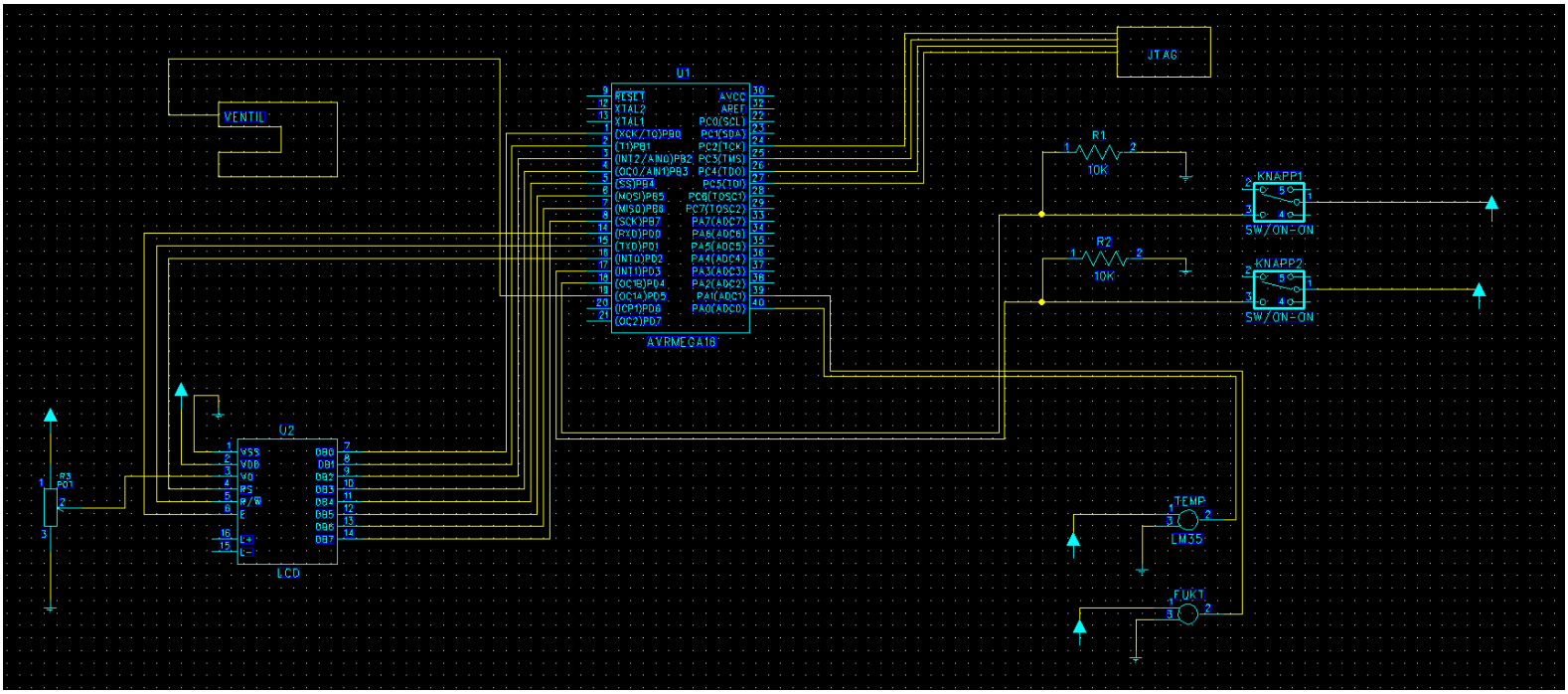
Efter att projektet har färdigställts så är vi mycket nöjda med utfallet, förutom att det visade sig att magnetventilen inte fungerar så att blomman inte kan vattnas. Något anant som tydligt märktes var att tidsåtgången var mycket svåruppskattad, för att det var första gången vi gjorde någonting sånthär. Vi märkte tidigt att det var väldigt svårt att vara flera personer som skriver kod samtidigt, även om kommentarer används. Detta berodde till stor del av att komponentbaserad kunskap dominerade programmeringsprocessen.

Ur ett projektperspektiv var detta ett ganska enkelt projekt men väldigt lärorik process från idé till hårdvara för att sedan skriva mjukvara. Vi lärde oss också en hel del grundläggande C, vilket också var väldigt lärorikt.

Vi är nöjda med utfallet av projektet, i termer om både användarvänlighet men även för våra blommors skull!

Kopplingsschema

För högupplöst bild, se hemsidan.



Källkod

```
#define F_CPU 1000000UL // 1 MHz
#include <util/delay.h>
#include <avr/io.h>

/*Function declaration*/
void InitADC();
uint16_t ReadADC(uint8_t);
void lcd_init();
void RSHigh();
void RSLow();
void RWHigh();
void RWLow();
void EHigh();
void ELow();
void lcd_reset();
void write_command(char val);
void write_data(char val);
void LCD_print (char *str);
void VentHigh();
void VentLow();
void knappNer();
void knappUpp();
void print_status();
void vattning();

/*Variable declaration*/
char btn1;
char btn2;
uint16_t fukt;
int lcd_delay = 10;
int global_delay = 0 ;
int fuktapp = 0;

int main(void)
{
	InitADC();

	DDRA = 0b00000000;
	DDRB = 0b11111111;
	DDRD = 0b00100111;


	VentLow();

	lcd_reset();
	lcd_init();
	LCD_print("My name is");
	new_line();
	LCD_print("Feeding Daisy");
	// _delay_ms(2000);


	while(1)
	{
		btn1 = (0b00001000 & PIND);
```

```
		btn2 = (0b00010000 & PIND);

		fukt=ReadADC(1);

		global_delay++;
		if(btn1 == 0b00001000){ // pin 3
			knappUpp();
		}
		if (btn2 != 0){
			knappNer();

		}

		if(global_delay > 15000){
			if(fukt<fuktapp) {
			vattning();
			}
			global_delay = 0;
			print_status();
		}
		_delay_ms(1);
	}
}

void InitADC(){
	ADMUX=(1<<REFS0);
	ADCSRA=(1<<ADEN) | (1<<ADPS2) | (1<<ADPS1);
}

uint16_t ReadADC(uint8_t ch){
	ch &= 0b00000111;
	ADMUX |= ch;

	//Start single conversion
	ADCSRA |= (1<<ADSC);

	//Wait for conversion to complete
	while(!(ADCSRA & (1<<ADIF)));

	ADCSRA |= (1<<ADIF);
	return(ADC);
}

void knappUpp(){
	if (fuktapp == 600) {
		return;
	}
	fuktapp = fuktapp + 100;
	print_status();
}

void print_status(){
		lcd_clear();
		char str[15];
		sprintf(str, "%d", (int) fuktapp);
		char stv[15];
		sprintf(stv, "%d", (int) fukt);
		LCD_print("Nuv fukt: ");
```

```
                LCD_print(stv);
                new_line();
                LCD_print("Önskad fukt: ");
                LCD_print(str);
                _delay_ms(200);
}

void vattning(){
       VentHigh();
       _delay_ms(10000);
       VentLow();
}

void knappNer(){
       if (fuktapp == 0){
              return;
       }
       fuktapp = fuktapp - 100;
       print_status();
}

void lcd_init(){
       write_command(0b00111000); //Function set
       write_command(0b00001110); //Display on
       write_command(0b00000110); //Entry mode set
       lcd_clear();
}

void write_command(char val)
{
       PORTB = val; // DATA in
       _delay_ms(lcd_delay);
       RSLow();
       _delay_ms(lcd_delay);
       RWLow();
       _delay_ms(lcd_delay);
       ELow();
       lcd_reset();
}

void write_data(char val){
       if (val == 'Ö') {
              PORTB = 0xEF;
       } else {
       PORTB = val;
       }
       _delay_ms(lcd_delay);
       RSHigh();
       _delay_ms(lcd_delay);
       RWLow();
       _delay_ms(lcd_delay);
       ELow();
       lcd_reset();
}

void lcd_reset(){
       _delay_ms(lcd_delay);
       EHigh();
```

```
	_delay_ms(lcd_delay);
	RWHigh();
	_delay_ms(lcd_delay);
	RSLow();
	_delay_ms(lcd_delay);
}

void lcd_clear(){
	write_command(0b00000001);
}

void VentHigh(){
	PORTD |= (1<<5);
}

void VentLow(){
	PORTD &= ~(1<<5);
}

void ELow()
{
	// Clear E to 0
	PORTD &= ~(1<<0);
}
void EHigh()
{
	// Set E to 1
	PORTD |= (1<<0);
}
void RSLow()
{
	// RS (D-Port 2) to 0
	PORTD &= ~(1<<2);
}
void RSHigh()
{
	// RS to 1
	PORTD |= (1<<2);
}
void RWLow()
{
	// RW to 0
	PORTD &= ~(1<<1);
}
void RWHigh()
{
	// RW to 1
	PORTD |= (1<<1);
}

void LCD_print (char *str)
{
	int i;
	for(i=0; str[i]!='\0'; i++)
	{
		if (i == 16)
		{
			new_line();
```

```
                }
                if (i > 31)
                {
                        return;
                }
                write_data(str[i]);
        }
        return;
}


void new_line(){
        PORTB = 0xC0;
        _delay_ms(lcd_delay);
        RWHigh();
        _delay_ms(lcd_delay);
        EHigh();
        _delay_ms(lcd_delay);
        RWLow();
        _delay_ms(lcd_delay);
        ELow();
        _delay_ms(lcd_delay);
        EHigh();
        lcd_reset();
}
```