EITF11 – Digitala Projekt VT14

CYKELLARM

Kajsa Alenmyr
Hanna Karlberg
Antonia Nilsson

Handledare: Bertil Lindvall

Sammanfattning

Detta projekt går igenom arbetsprocessen för framtagandet av en prototyp av ett cykellarm. Rapporten behandlar projektet från planeringsstadium, montering av hårdvara samt programmering av mjukvara. Den behandlar även de utmaningar som stöttes på under vägens gång samt de ändringar och tillägg som bör göras till den slutgiltiga produkten.

Innehållsförteckning

1. Inledning

Detta projekt är en del av kursen Digitala Projekt EITF11 som ges till studenter vid Lunds Tekniska Högskola. Kursen har som syfte att ge studenter en inblick i hur konstruktionsarbete går till genom att skapa en prototyp av en teknisk produkt med tillhörande dokumentation.

Projektet har som syfte att skapa en prototyp av ett cykellarm som känner av om cykeln förflyttas. Insignaler till larmet är en acceleroemeter som känner av om cykeln förflyttar sig märkbart och sätter då igång larmet. Utsingalen utgörs av en lampa som blinkar samt en summer som ger ifrån sig ett klickande ljud. Dessa sätts igång om cykeln rör sig för att alarmera omgivningen om att cykeln förflyttas.

2. Kravspecifikation

Ett cykellarm som reagerar på om cykeln rör sig ska skapas. Larmet ska kunna vara i två tillstånd, på- och avlarmat. När larmet är pålarmat ska det vara känsligt för förflyttning och alarmera användaren om detta sker. När larmet är avlarmat ska inget hända om cykeln förflyttas.

- Larmet ska reagera på rörelse genom en accelerometer som ska sätta igång larmet om det är pålarmat.
- En blå LED-lampa ska vara monterad på larmet och börja blinka om larmet går igång.
- En summer ska vara monterad på larmet och ge ifrån sig ett klickande ljud om larmet går igång.
- För att larma av och på används en knappsats med fyra knappar (1, 2, 3, på/av) på följande vis:
    - Larma på: På/av-knappen trycks in.
    - Larma av: En tresiffrig kod trycks in och sedan på/av-knappen.
    - Om någon av knapparna trycks in på annat sätt ska ingenting hända.
- Larmet ska fortsätta blinka/låta tills strömmen bryts.
- Larmet ska bestå av:
    - Summer
    - Knappsats
    - Accelerometer
    - Kristall
    - Lysdiod
    - Processor

3. Hårdvara

3.1 Kopplingsschema

Ett schema som visar hur komponenterna är kopplade till varandra upprättades innan montering av larmet. För kopplingsschema, se bilaga 1.

3.2 Processor

Processorn som används är en 8-bitars ATMega32. Processorn har ett programmeringsbart minne på 32kB och fyra portar (A-D) som är fördelade över 32 stycken I/O-pinnar. Den har även en inbyggd A/D-omvandlare för att hantera analoga signaler samt ett JTag interface för kommunikation mellan processorn och dator.

3.3 Knappar

För att slå av/på larmet samt för att mata in PIN-koden användes fyra stycken röda knappar. Dessa knappar fungerar likt fyra separata strömbrytare. De är i sin tur kopplade till en AND-port för att hantera knapptryckningar.

3.4 Summer

För att larmet ska kunna ge ifrån sig ett ljud när det triggas igång användes en summer. Summern ger ifrån sig ett klickande ljud när larmet triggas.

3.5 LED-lampa

En blå LED-lampa användes och ger ifrån sig ett blinkande ljus när larmet triggas.

3.6 Kristall

En extern kristall med frekvensen 10 Hz användes och kopplades till processorn.

3.7 Acceleroemter

För att larmet ska veta när cykeln rör på sig användes en accelerometer. Denna känner av acceleration i tre led, x, y, och z. När dessa värden ändras avsevärt triggas larmet igång.

3.8 AND-port

För att styra knapparna på ett smidigt sätt användes en AND-port. Programmeringskod skrevs sedan för hur systemet skulle reagera då en knapp blev nedtryckt.

4. Mjukvara

I programmets startfas är alarmOn och alarmTriggered satta till 0. De talar om för systemet om larmet är på- eller avlarmat, det vill säga om det kommer triggas igång om cykeln flyttar på sig, samt om alarmet är triggat, det vill säga om det blinkar och låter för att cykeln förflyttar sig. AlarmTriggered kan alltså bara bli 1 om alarmOn är 1. Dessa värden kontrolleras av användaren via på/av-knappen och PIN-koden.

För att larma på krävs bara att användaren trycker in på/av-knappen men för att larma av krävs att användaren även knappar in sin PIN-kod, som i prototypen är 123.

När larmet är på-larmat kommer det att svara mot förändringar hos accelerometern. Om accelerometerns värden avviker avsevärt från det "stationära" värdet kommer larmet att triggas igång. När larmet triggas igång börjar lampan blinka och summern låta. Larmet kan endast stängas helt av genom att bryta strömmen.

För att få summern och lampan att blinka/klicka användes ett interrupt. När programmet då gick in i interruptet alarmerade summern och lampan enligt kravspecifikationen.

Hela källkoden finns i bilaga 2.

5. Arbetsprocessen

5.1 Planering

Vid starten av projektet utformades en kravspecifikation som godkändes av handledaren innan vidare arbete fortsattes. Samtliga hårdvarukomponenter valdes därefter ut och ett kopplingsschema, med samtliga delar och dess kopplingar, skapades.

5.2 Montering av hårdvara

Monteringen skedde på ett mönsterkort där komponenterna kopplades samman med tunn metalltråd. Metalltråden virades antingen fast med virpistol eller virades och löddes fast. Efter montering testades samtliga komponenter och kopplingar genom att skriva kort testkod som kommunicerades till systemet med JTagen. Detta för att undvika fel i hårdvaran då mjukvaran skapades.

5.3 Programmering av mjukvara

Programmet för att styra microkontrollern är skrivet i programmeringsspråket C. Det skrevs i utvecklingsmiljön Atmel Studio 6. Med hjälp av en JTag kopplad till processorn kunde sedan programmet kopplas till microkontrollern.

5.4 Utmaningar

Då ingen tidigare erfarenhet av varken programspråket C eller montering på mönsterkort fanns i gruppen var hela arbetsprocessen en utmaning. Detta löstes genom nära kontakt med handledare och noggrann läsning av datablad och annan viktigt information. Då tidigare erfarenhet fanns inom andra programmeringsspråk, framför allt Java löste sig även programmeringen. Andra mer specifika utmaningar som träffades på under arbetets gång är listade nedan:

- Knapparna lästes av flera gånger när de trycktes ned en gång.
  Lösning: En delay skapades som såg till att knapparna inte lästes av flera gånger.

- Lampan och summern blinkade/klickade med för låg ljusstyrka/volym.
  Lösning: Problemet var att programmet inte hann gå in i interrupten särskilt länge. Därför hann inte lampan och summern gå till sitt maxläge innan den slocknade igen. När problemet hade identifierats löstes det snabbt genom liten modifikation av koden.
- Accelerometern var för känslig och sedan för okänslig.
  Lösning: Accelerometerns "stationära" värden identifierades och sedan gjordes flertalet tester för att se hur olika värden i de olika leden påverkade känsligheten. Efter testerna skapades medelvärden som sedan gav en god känslighet hos accelerometern.

6. Resultat

Resultatet av projektet var en fungerande prototyp av ett cykellarm i enighet med kravspecifikationen som upprättades i början av arbetsprocessen.

7. Slutsats och diskussion

Projektet har givit oss erfarenheter inom planering, montering av hårdvara, programmering av mjukvara i C samt hur dessa kommunicerar med varandra.

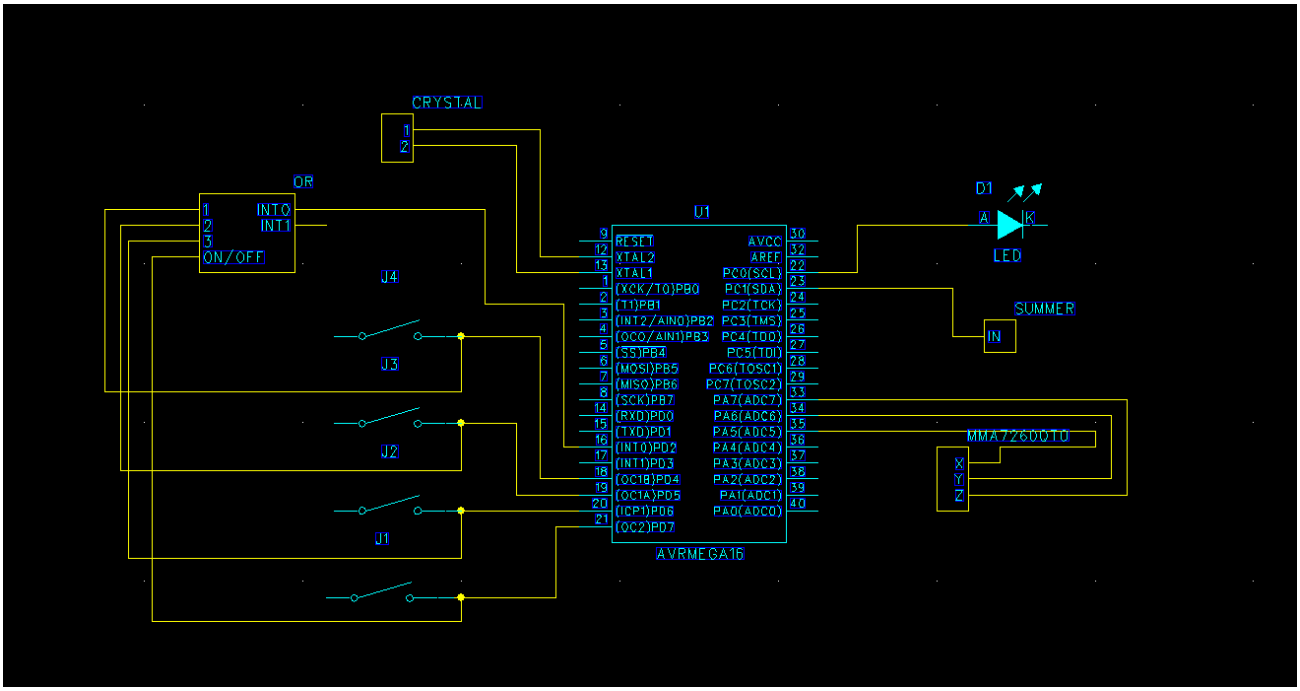
En hel del utmaningar stöttes på under vägens gång vilket har gett gruppen erfarenheter inom problemlösning.

Då detta endast är en prototyp skulle en del förändringar krävas till det riktiga larmet. Några av dessa listas nedan:

- Larmet ska vara mindre känsligt och endast reagera på förflyttning av minst en meter.
- Larmet ska gå på batteri.
- Larmet ska ha en längre kod som väljs användaren och som sedan kan ändras.
- Larmet bör låta mer då det utlöses.

Bilagor

Bilaga 1: Kopplingsschema



Bilaga 2: Källkod

```
/*
 * cykellarm.c
 *
 * Created: 2014-03-31 09:39:57
 *  Author: digpi09
 */

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

unsigned int alarmOn = 0;
unsigned int alarmTriggered = 0;
unsigned int countTime = 0;
unsigned int countCode = 0;
unsigned int codeOK = 0;
unsigned char code[4] = {1, 2, 3};
unsigned char enteredCode[4] = {0, 0, 0};
unsigned int countAD = 0;
unsigned char x = 120;
unsigned char y = 120;
unsigned char z = 120;
unsigned int xSum = 0;
unsigned int ySum = 0;
unsigned int zSum = 0;
unsigned int countFourX = 0;
unsigned int countFourY = 0;
unsigned int countFourZ = 0;
unsigned int bol1 = 1;
unsigned int bol2 = 1;
unsigned int bol3 = 1;
unsigned int bolOn = 1;

int main(void) {
	DDRA = 0b00000000;
	ADCSRA = 0b11001111;
	ADMUX = 0b00100101;

	DDRC |= 0b00000011;
	TCCR0 = 0b00000101;
	TCNT0 = 0b00000000;
	TIMSK = 0b00000001;		//Overflow interrupt enabled
	TIFR = 0b00000001;

	PORTD = 0b11111111;
	DDRD = 0b00000000;
	MCUCR = 0b00000010;
	GICR = 0b01000000;
	GIFR = 0b01000000;

	sei();
    while(1) {
		movement();
```

```
	}
	return;
}

void movement(){
	if(alarmOn == 1){
		if(x <= 60 || x >= 170){
			alarmTriggered = 1;
		}
		if(y <= 65 || y >= 170){
			alarmTriggered = 1;
		}
		if(z <= 65 || z >= 170){
			alarmTriggered = 1;
		}
	}
}

void enterButton(int a){
	if (alarmOn == 1){
		countCode ++;
		if( countCode < 3){
			enteredCode[countCode-1] = a;
		}
		else if( countCode == 3){
			enteredCode[countCode-1] = a;
			enterCode();
			countCode = 0;
		}
	}
	bol1 = 1;
	bol2 = 1;
	bol3 = 1;
}

void enterCode() {
	if (code[0] == enteredCode[0] && code[1] == enteredCode[1] && code[2] ==
enteredCode[2]) {
		codeOK = 1;
	}
}

void turnOn() {
		if (alarmOn == 0) {
			alarmOn = 1;
		} else if (alarmOn == 1 && codeOK == 1) {
			alarmOn = 0;
			codeOK = 0;
			alarmTriggered = 0;
		}
		PORTC |= 0b00000011; //ger användaren gensvar att alarmet larmats på
		_delay_ms(100);
		PORTC &= 0b11111100;
		bolOn = 1;
		countCode = 0;
		empty();
}
```

```
void empty(){
	for(int i = 0; i < 3; i++){
		enteredCode[i] = 0;
	}
}

ISR(TIMER0_OVF_vect) {
	TIFR = 0b00000001;
	countTime++;
	if (countTime == 5) {
		if (alarmTriggered == 1) {
			char var = PINC;			//Mask
			if ((var & 0b00000011) == 0b00000011) {
				PORTC &= 0b11111100;
			} else {
				PORTC |= 0b00000011;
			}
		}
		countTime = 0;
	}
	TCNT0 = 0b00000000;
	cli();
}

ISR(INT0_vect) { // Hamnar här vid interrupt. Vill se vilken knapp som trycks in samt
utföra önskat kommando.
	_delay_ms(100);
	char var = PIND;
	char var2 = var & 0b11110000;
	switch(var2){
		case 0b11100000:
			if(bol1 == 1){
				bol1 = 0;
				PORTC |= 0b00000011; //ger användaren gensvar att knapp
tryckts in
				_delay_ms(100);
				PORTC &= 0b11111100;
				enterButton(1);
				_delay_ms(400);
			}
			break;
		case 0b11010000:
			if( bol2 == 1){
				bol2 = 0;
				PORTC |= 0b00000011; //ger användaren gensvar att knapp
tryckts in
				_delay_ms(100);
				PORTC &= 0b11111100;
				enterButton(2);
				_delay_ms(400);
			}
			break;
		case 0b10110000:
			if( bol3 == 1){
				bol3 = 0;
				PORTC |= 0b00000011; //ger användaren gensvar att knapp
tryckts in
				_delay_ms(100);
```

```
                        PORTC &= 0b11111100;
                        enterButton(3);
                        _delay_ms(400);
                }
                break;
        case 0b01110000:
                if( bolOn == 1){
                        bolOn = 0;
                        PORTC |= 0b00000011; //ger användaren gensvar att knapp
tryckts in
                        _delay_ms(100);
                        PORTC &= 0b11111100;
                        turnOn();
                        _delay_ms(400);
                }
                break;
        }
        cli();
}

ISR(ADC_vect){
        if (alarmOn == 1){
                countAD ++;
                if(countAD == 1){
                        ADMUX = 0b00100101;
                        unsigned char xTemp = 0;
                        xTemp = ADCH;
                        countFourX ++;
                        xSum = xSum + xTemp;
                        if(countFourX == 4){
                                x = xSum/4;
                                countFourX = 0;
                                xSum = 0;
                        }
                } else if(countAD == 2){
                        ADMUX = 0b00100110;
                        unsigned char yTemp = 0;
                        yTemp = ADCH;
                        countFourY ++;
                        ySum = ySum + yTemp;
                        if(countFourY == 4){
                                y = ySum/4;
                                countFourY = 0;
                                ySum = 0;
                        }
                } else if(countAD == 3){
                        ADMUX = 0b00100111;
                        unsigned char zTemp = 0;
                        zTemp = ADCH;
                        countFourZ ++;
                        zSum = zSum + zTemp;
                        if(countFourZ == 4){
                                z = zSum/4;
                                countFourZ = 0;
                                zSum = 0;
                        }
                        countAD = 0;
                }
```

```
        }
        ADCSRA = 0b11001111;
        cli();
}
```