LUNDS UNIVERSITET

DIGITALA PROJEKT - EITF11

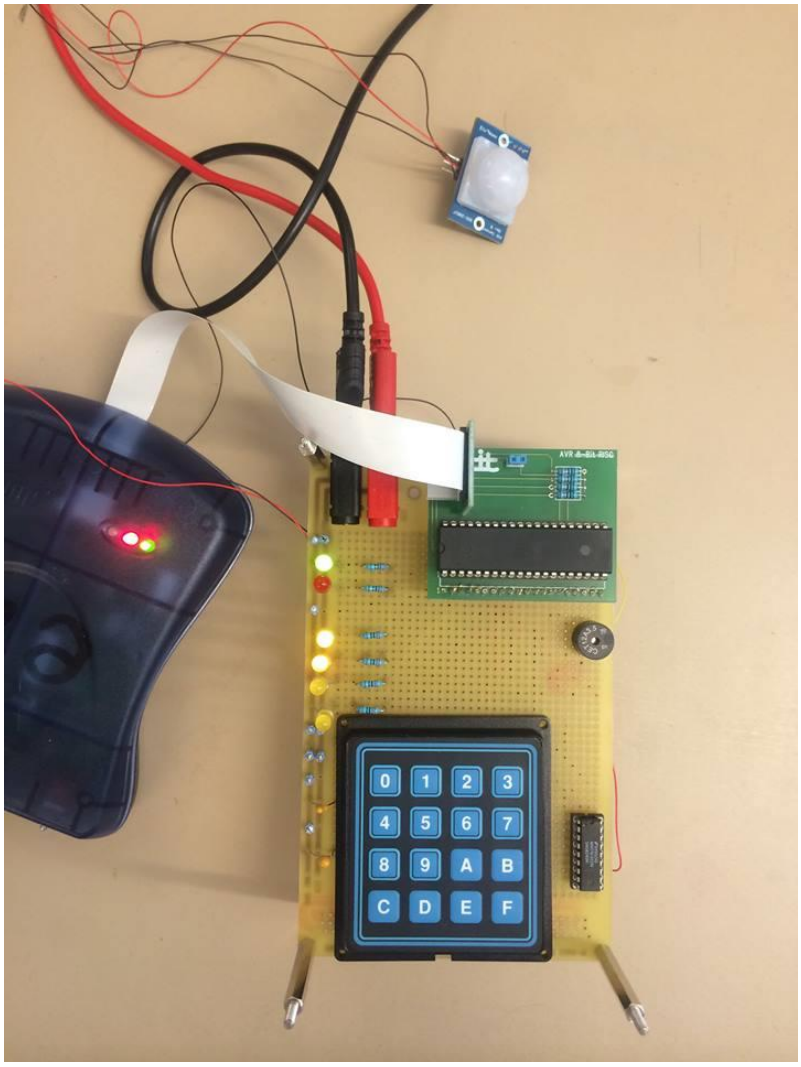
KYLSKÅPSLARM

Maja Sölve
Maxine Rior
Handledare: Bertil Lindvall
2015-04-15

Maja Sölve
Maxine Rior

Sammanfattning

Syftet med kursen Digitala Projekt (EITF11) är att konstruera en fungerande prototyp innehållandes både hård- och mjukvara. Den här rapporten sammanfattar arbetet med prototypen och ämnar belysa processen i alla skeden: från projektering, via planering, till montering av hårdvara och implementering av mjukvara. I detta fall rör det sig om en larmanläggning, syftat att användas i kylskåp, ett så kallat kylskåpslarm. Rapporten kommer att behandla en detaljerad beskrivning av arbetsprocessen för framtagandet av detta kylskåpslarm och utöver det även blockschema för komponenterna i hårdvaran samt den aktuella källkoden för mjukvara. Rapporten knyts ihop av en reflektion över arbetsprocessen där författarna även diskuterar en eventuell vidareutveckling av det konstruerade kylskåpslarmet.



Figur 1: Larm-prototypen med två siffror inslagna

Innehållsförteckning

I Introduktion

Digitala Projekt (EITF11) är en konstruktionskurs där studenterna lär sig att på egen hand bygga en prototyp från grunden. Prototypen skall ha noga utvalda komponenter samt en effektiv källkod som enkelt går att följa. Syftet är att prototypen skall kunna utföra en rad enkla kommandon, och även eventuellt kunna utvecklas vidare. Vad för sorts prototyp som skall konstrueras är upp till studenterna att välja ur en uppgiftskatalog, som består av diverse elektroniska produkter.

I det här fallet valdes en larmanläggning, närmare specifikt ett kylskåpslarm. Med detta menas att larmet med hjälp av en rörelsesensor ska reagera om en obehörig är inne på larmägarens kylskåpshylla. Om detta sker kommer larmanläggning (på utsidan av kylskåpet) att signaler detta via både ljud och ljus (blinkande LED). Ägaren till hyllan kan med hjälp av en PIN-kod larma av och larma på, samt byta PIN-kod, via en knappsats. Bakgrunden till detta val av konstruktion ligger i den boendetyp som studenter ofta nyttjar: korridor/kollektiv med delat kök. Stöld av mat förekommer frekvent året runt och eskalerar i festtid såsom Valborg. Inofficiella undersökningar visar på att det oftast är färdiglagade matlådor som stjäls ur kylskåp.

II Kravspecifikation

II.1 Produktbeskrivning

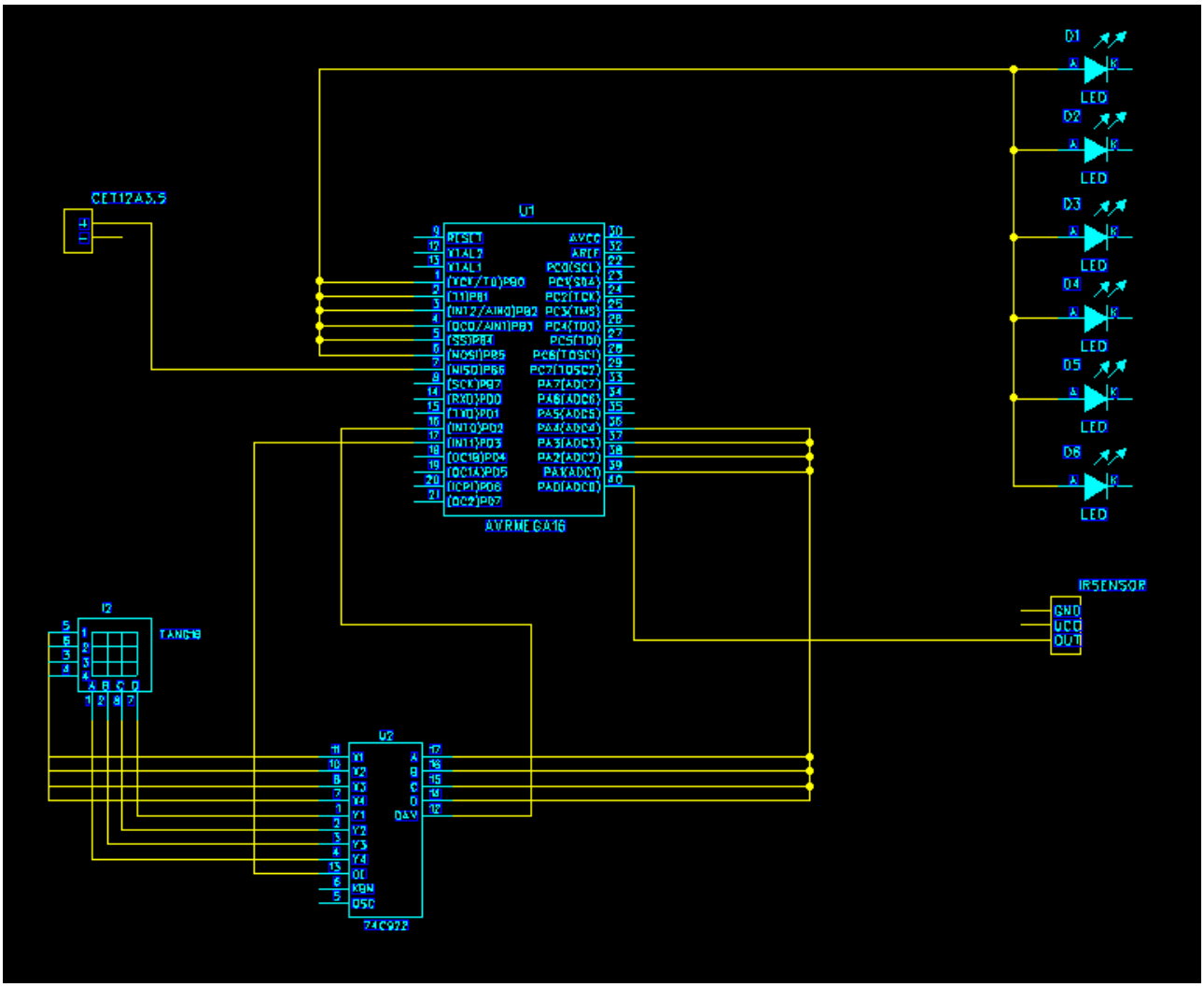
- En digital IR-sensor för att genom temperatur registrera rörelse på kylskåpshyllan.
- En knappsats med siffrorna 0-9 samt ytterligare knappar A-F för att bestämma funktioner och kommandon.
- Sex stycken dioder: en grön, en röd och fyra gula som visar larmets status samt antal intryckta knappar.
- En siren som låter när larmet har gått, dvs. att IR-sensorn har reagerat.

II.2 Användningsbeskrivning

- Larmet aktiveras (från avaktiverat läge) efter det att A samt rätt PIN-kod slagits in.
- Om fel kod slås i avaktiverat läge händer inget, oändligt antal gånger.
- Om man inte stängt av larmet går det igång direkt när rörelsedetektorerna känt av att någon rört något på hyllan.
- Om IR-sensorn registrerar rörelse när larmet är aktiverat går larmet igång direkt, det vill säga en röd lysdiod börjar blinka och sirenen ger ifrån sig ljud.
- När knapp D trycks in avaktiveras larmet efter att rätt PIN-kod slagits in. Knapp C ändrar pinkoden och knapp E nollställer intryckt PIN-kod så att man kan ångra sig om man råkar slå in en felaktig siffra i PIN-koden.

III Hårdvara

III.1 Kopplingsschema



Figur 2: Kopplingsschema

III.2 Processor

Den processor som används till larmanläggningen är en ATmega16 AVR 8-bit Microcontroller. Processorn har 16 kB inbyggt, programmerbart flashminne samt 32 programmerbara pins som kan sättas till in eller ut (I/O). Dessa I/O-pinnar är fördelade jämnt på fyra portar: A, B, C och D. Kommunikationen mellan processor och dator för att kunna implementera mjukvara sker via en JTag, automatiskt förberedd i port C. Slutligen är nämnvärt även att två av I/O-pinnarna är så kallade interrupt pins, vilket betyder att man kan programmera dem att avbryta en process när en specificerad händelse sker (exempelvis att någon trycker ner en knapp på en knappsats).

III.3 IR-sensor

IR-Sensorn mäter temperaturer i rummet och registrerar markant avvikelse (eller förändring). Detta märks genom att IR-sensorn både skickar en digital signal in till processorn samt att en röd LED-diod inbyggd i sensorn tänds. Dioden förblir tänd tills sensor inte längre reagerar på temperaturförändring. Det innebär att en person kan sitta still intill sensorn, och det går bra. När personen sedan börjar röra sig ger sensorn utslag. IR-sensorn heter PIR Sensor (#555-28027) och fungerar i temperaturer från 0 till 50 grader Celsius.

III.4 Knappsats

Larmet styrs av en knappsats bestående av en 4x4-matris. Siffrorna 0-9 samt bokstäverna A-F finns med på knapparna. Knappsatsen är ansluten till en key encoder, se III.5, som läser av knappsatsen och skickar en digital signal till processorn om vilken knapp det är som just blev nedtryckt. Knappsatsen som används heter TANG16.

III.5 Key Encoder

För att kunna läsa av vilken knapp som tryckts ner på knappsatsen behövs en key encoder. Key encodern fungerar som mellanhand mellan knappsatsen och processorn, och använder sig av fyra stycken I/O-pinnar (ger totalt 16 möjligheter, vilken är antalet knappar) för att berätta för processorn vilken knapp som tryckts ner. Encodern som använts till larmet heter MM74C922 16-Key Encoder.

III.6 Lysdioder

Lysdioder (i detta projekt LED) används i flera syften i larmanläggningen. En grön diod lyser när larmet är avstängt och det är fritt fram att hämta mat samt en röd diod lyser när larmet är påslaget och således måste avaktiveras för att mat ska kunna hämtas. Utöver det blinkar även den röda dioden ilsket då larmet har gått, d.v.s. när IR-sensorn har gett utslag. Slutligen tänds en gul diod för varje siffra i den fyrsiffriga PIN-koden när användaren slår in dessa.

III.7 Siren

En enkel siren kopplas in till processorn genom en vanlig I/O-pinne. Sirenen har två lägen: antingen är den på och emitterar ljud, eller så är den av och då låter den inte. Alltså kommer sirenen att aktiveras av att larmet går igång (IR-sensorn ger utslag när larmet är aktiverat). Sirenen som används i detta projekt heter CET12A3.5.

IV Mjukvara

Mjukvaran som styr larmet är relativ enkel. Alla variabler, till exempel status för larmet, deklareras i början så att de får ursprungsvärden vid varje körning av mjukvaran. I main-metoden återfinns sedan ett par enkla startmetoder, som definierar PIN-kod, öppnar pinnar och liknande. Efter detta går programmet in i en ”evig” while-loop som innebär att larmet hela tiden söker av om till exempel IR-sensorn har gett utslag. Eftersom att vår knappsats är kopplad till interrupt-pinnar på processor så kommer mjukvaran omedelbart att reagera när någon trycker in en knapp, sluta med vad den höll på med innan, och ta hänsyn till knapptryckningen och det som den är inställd på att förvänta sig då. Interruptmetoden läser av vilken knapp som har tryckts ned och beroende på knapp så skickar den vidare vilken undermetod som ska exekveras. Exempelvis, om status för larmet är 0, dvs larmet är ej aktiverat, och knappen A trycks ned betyder detta att larmet ska aktiveras. Mjukvaran väntar då på en PIN-kod efter att A tryckts ner, och aktiveras om PIN-koden är korrekt. Annars händer ingenting. Likadant är det för andra bokstavsknappar, men de har andra funktioner. Knappen D exempelvis betyder deactivate. Status på larmet, samt nedtryckta knappar kommer att få lysdioderna att tändas på olika sätt. Om larmet är avaktiverat kommer exempelvis en grön LED att lysa, och när det är aktiverat kommer en röd att ljus. När larmet går kommer sirenen att låta samt en röd LED att blinka.

V Arbetsprocessen

V.1 Planeringen

Kravspecifikationen utfördes i ett väldigt tidigt stadie och diskuterades med handledaren. Därefter kunde val av komponenter göras och därefter ett kopplingsschema börja planeras. Komponenter och elektroniska kretsar upplevdes som tämligen okända områden jämför med tidigare kurser inom programmet vilket är anledningen till att beslut kring detta togs extra noggrant. Gott om tid avsattes till planeringen av hårdvara, då författarna resonerade som så att det är viktigt att hårdvaran stämmer från början.

V.2 Montering av hårdvara

Då val av komponenter hade gjorts samt ett korrekt kopplingsschema var ritat konstruerades prototypen. Med hjälp av en lödningspenna fästes bland annat sladdar, lysdioder och resistorer på kretskortet. När allt var på plats kunde JTagen kopplas in till processorn och vidare in i datorn. Från datorn och programmet Studio6 kunde sedan samtliga komponenter testas, så att det kunde fastställas att de var korrekt inkopplade och inte skulle kortsluta.

V.3 Implementering av mjukvara

När samtliga komponenter hade visat sig fungera med hjälp av lite enkel testkod, exempelvis så tändes dioderna en efter en, så implementerades den stora main-metoden. Här skrevs undermetoderna först, och efter hand som varje metod skrevs så deklarerades de variabler som behövdes till den metoden. På så sätt byggdes hela programmets undermetoder upp sakta med säkert. Till slut skrevs main-metoden, där alla undermetoder används och själva programmet körs. När hela programmet var färdigskrivet skedde lite finputsning så att koden skulle bli så effektiv som möjligt.

V.4 Hinder under processens gång

Att löda de elektroniska komponenterna så att de tog in spänning på rätt sätt visade sig lite klurigare än väntat då det enkelt glappade. Annars skedde montering utan större problem då det enda kravet var tålamod att pilla med väldigt små saker. När det gäller koden kunde det mesta skrivas tack vare vana i Java. Däremot så tog det lite tid att sätta sig in i språket C och vad som gäller i exekvering där. Då tiden var generöst budgeterad var detta egentligen dock aldrig ett hinder, utan mer en fartbula. Att få interrupt-systemet att fungera visade sig vara det svåraste av all kod, då det innebar att flertalet främmande kommandon och kodrader i C behövdes enbart för att kunna säkerställa kommunikation med knappsatsen.

VI Diskussion och slutsats

Den larmanläggning detta projekt har handlat om grundades i en verkligen behovssituation. Tanken har från början varit att detta borde kunna utvecklas till en riktig produkt, som sedan ska kunna säljas till studenter eller andra kylskåpsdelare på exempelvis TGR eller på hemsidor såsom coolstuff.se. Då detta har varit baktanken hela tiden, och därför även slutpriset på produkten så har vissa förenklingar gjorts, som lysdioder istället för en LCD-skärm.

Projektet med att bygga en prototyp, i detta fall en larmanläggning, har varit mycket lärorikt. Det flesta arbetsmomenten som har stötts på har varit en förstagångsupplevelse, vilket har inneburit ganska mycket tid för att läsa sig in på hur saker faktiskt fungerar. Programmeringskunskaper är värdefulla och det finns ett roligt moment i att på egen hand på en lysdiod att tändas. Självständigheten i projektet har uppskattats ordentligt och gett mersmak. Även erfarenheten att ha planerat ett projekt och sedan hållit sig till interna deadlines har varit nyttigt.

VII Bilagor

VII.1 Källkod

```
/*
 * larm.c
 *
 * Created: 2015-03-27 09:38:35
 *  Author: digpi06
 */


#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define F_CPU 8000000UL //set the frequency of the CPU unsigned long, 8 MHz

#define LEDGREEN PB0
#define LEDRED PB1
#define LEDYELLOW1 PB2
#define LEDYELLOW2 PB3
#define LEDYELLOW3 PB4
#define LEDYELLOW4 PB5
#define SENSOR PA0
#define BUZZER PB6
#define OUTPUT PD3

unsigned int status = 0;
unsigned int expectPin = 0;
unsigned int changingCode = 0;
unsigned int alarmTriggered = 0;
unsigned char pinCode[4] = {'1', '2', '3', '4'};
unsigned char pinCodeAttempt[5];
unsigned int codeOk;
unsigned char symbol;
unsigned int keyNbr = 0;

char readKeyPad(char code) {
	if (code == 0b00000000){
		return '3';
		}else if(code==0b00010000){
		return '7';
		}else if(code==0b00001000){
		return 'B';
		}else if(code==0b00011000){
		return 'F';
		}else if(code==0b00000100){
		return '2';
		}else if(code==0b00010100){
		return '6';
		}else if(code==0b00001100){
		return 'A';
		}else if(code==0b00011100){
		return 'E';
		}else if(code==0b00000010){
		return '1';
		}else if(code==0b00010010){
		return '5';
		}else if(code==0b00001010){
		return '9';
		}else if(code==0b00011010){
```

```
			return 'D';
			}else if(code==0b00000110){
			return '0';
			}else if(code==0b00010110){
			return '4';
			}else if(code==0b00001110){
			return '8';
			}else if (code==0b00011110){
			return 'C';
		}
}


int main(void) {

setUp();
enableKeypadInterrupt();

while(1){

if (status==1 && checkSensor()) {
		setLed(LEDRED,1);
		buzz();
} else if (status == 0) {
		setLed(LEDRED,0);

}


}

}
void buzz(){
		while(1){
		_delay_ms(1);
		setPin('B',BUZZER,1);
		_delay_ms(1);
		setPin('B',BUZZER,0);
		if (status==0) {
			break;
		}
		}
		}
void enableKeypadInterrupt() {
		GICR = (1 << INT0); //General Interrupt Control Register, says that we are enabling
INT0
		MCUCR = (1 << ISC01) | (1 << ISC00); // MCU Control and Status Register, register the
condition on which the interrupt will be fired = rising edge here
		//SREG = 0x80; //set till 10000010, i.e. that bit 7 är 1 och bit 1 är 1, bit 7 =
enable global interrupt
		GIFR = 0;
		sei(); //enable global interrupts, sätter bit 7 till 1 i SREG
}


 ISR(INT0_vect){
		cli();

		char code = PINA & 0b00011110;
		char symbol = readKeyPad(code);

		if(symbol=='A') { //activate alarm
			if(status==0){
				expectPin = 1;
```

```
		}

	} else if(symbol=='D') { //deactivate alarm
				if(status==1){
					expectPin = 1;
				}
		}
		else if(symbol=='C'){ //change
		if(status==0){
			expectPin = 1;
			changingCode = 1;
		}

		 }
		 else if(symbol=='E'){
			resetCode();
		 }

		if (expectPin==1 && changingCode == 0){
			insertCode(symbol);
				if (keyNbr==5){
					isPinCorrect();
					if(codeOk==1 && status == 0){
						activate();
						} else if(codeOk == 1 && status==1){
							deactivate();
						}
				 resetCode();
					}
					}
		else if (expectPin==1 && changingCode==1) {
				insertCode(symbol);
					if(keyNbr==5) {
						isPinCorrect();
							if(codeOk==1){
								changingCode=2;
							}
						resetCode();
					}
		}
		if (changingCode==2){

						insertCode(symbol);
						if (keyNbr==5){
						changePin();
						setLed(LEDRED,1);
						_delay_ms(100);
						setLed(LEDRED,0);
						setLed(LEDGREEN,1); //larmet är ju avaktiverat
fortfarande
						resetCode();
						changingCode=0;
						}
					}


	 sei();
 }


void resetCode() {
	keyNbr = 0;
	expectPin = 0;
```

```
	codeOk=0;
	setLed(LEDYELLOW1,0);
	setLed(LEDYELLOW2,0);
	setLed(LEDYELLOW3,0);
	setLed(LEDYELLOW4,0);
	pinCodeAttempt[0]=0;
	pinCodeAttempt[1]=0;
	pinCodeAttempt[2]=0;
	pinCodeAttempt[3]=0;
	pinCodeAttempt[4]=0;
}

void insertCode(char letter){
	pinCodeAttempt[keyNbr] = letter;
	ledCode(keyNbr);
	keyNbr++;
}


void ledCode(int i){
	if(i==1) {
		setLed(LEDYELLOW1,1);
	} else if(i==2){
		setLed(LEDYELLOW2,1);
	} else if(i==3){
		setLed(LEDYELLOW3,1);
	} else if(i==4){
		setLed(LEDYELLOW4,1);
		_delay_ms(1000); //så att alla 4 lyser tillsammans en sekund
	}
}

void changePin() {

	for(unsigned int i = 0; i<=3; i++) {
		pinCode[i] = pinCodeAttempt[i+1];
	}
}


void isPinCorrect() {
	if(pinCode[0]==pinCodeAttempt[1] && pinCode[1]==pinCodeAttempt[2] &&
pinCode[2]==pinCodeAttempt[3] && pinCode[3]==pinCodeAttempt[4]) {
		codeOk = 1;
	}
}

void activate() {
	if(status==0) {
		status = 1;
		setLed(LEDRED,1);
		setLed(LEDGREEN,0);
	}

}

void deactivate() {
	if(status==1) {
		status = 0;
		setLed(LEDGREEN,1);
		setLed(LEDRED,0);
	}
}
```

```
int checkSensor(){
	char set= PINA & 0b00000001;
	if(set == 0b00000001){
		return 1;
	} else {
		return 0;
	}
}

void blinkLed(char led) { //används inte i dagsläget men känns bra att ha om man vill lägga
till en blinkande lampa i senare skede
	setLed(led,1);
	_delay_ms(500);
	setLed(led,0);
	_delay_ms(500);
	setLed(led,1);
	_delay_ms(500);
	setLed(led,0);
	_delay_ms(500);
	setLed(led,1);
	_delay_ms(500);
	setLed(led,0);
	_delay_ms(500);
	setLed(led,1);
	_delay_ms(500);
	setLed(led,0);
}

void setLed(char led,char state){
	setPin('B',led,state);
}


void setPin(char port, char pin, char state){
	char set=1<<pin;
	if(port=='B'){
		set&=PORTB;
	if(set&&!state){
		PORTB^=set;
	}
	if(set==0&&state){
		set=1<<pin;
		PORTB^=set;
	}
} else if(port=='A'){
set&=PORTA;
if(set&&!state){
	PORTA^=set;
}
if(set==0&&state){
	set=1<<pin;
	PORTA^=set;
}
} else if(port=='C'){
set&=PORTC;
if(set&&!state){
	PORTC^=set;
}
if(set==0&&state){
	set=1<<pin;
	PORTC^=set;
}
}else if(port=='D'){
```

```
set&=PORTD;
if(set&&!state){
	PORTD^=set;
}
if(set==0&&state){
	set=1<<pin;
	PORTD^=set;
}
}
}

void setUp() {
	DDRA = 0b00000000;
	DDRB = 0b01111111;
	DDRC = 0b00000000;
	DDRD = 0b00001000;
	setPin('D', OUTPUT,0);
	setLed(LEDGREEN,1);
	codeOk=0;
}
```