SNOOZE SNOOZE REVOLUTION

2015-05-18

GRUPP 11
Gustav Halling, Jacob Mejvik och Axel Ryding

Sammanfattning

Digitala projekt (EITF11) är en kurs vid Lunds tekniska i vilken eleverna får välja ett projekt som innefattar design av både hårdvara och mjukvara. Syftet med kursen är att illustrera industriellt utvecklingsarbete. I detta projekt har en väckarklocka konstruerats. Rapportens innehåll består av presentation av arbetsgången, de ingående komponenterna och den kod som har använts för att konstruera den färdiga produkten.

Innehållsförteckning

Inledning och projektpresentation

Följande rapport redovisar genomfört projekt inom ramen för kursen Digitala Projekt (EITF11) vid Lunds Tekniska Högskola. Kursen går över två läsperioder (VT1 och VT2) och ingår i teknikprofilen System- och programvaruutveckling på I-programmet. Kursen är en konstruktionskurs där en fungerande prototyp tas fram, tillsammans med tillhörande dokumentation i form av denna rapport och bilagor. Grupp 11 har valt att ta fram en prototyp av en digital väckarklocka som använder en 8-bitars ATmega16-processor som beräkningsenhet. Arbetet genomfördes på Institutionen för Elektro- och informationsteknik i E-huset på LTH.

Metod och arbetsgång

När idén om en väckarklocka slagits fast började utvecklingsarbetet. Utvecklingsarbetet genomfördes i fyra steg:

1. Kravspecifikation och design
2. Konstruktion
3. Programvaruutveckling
4. Testning

Dokumentation av projektet har skett löpande.

Kravspecifikation och design

Först gjordes en kravspecifikation för väckarklockans funktioner. Aspekter som funktionalitet, omfattning/komplexitet, tillgängliga komponenter och tidsåtgång togs i beaktande när kraven och funktionerna designades.

Den färdiga produkten ska:

1. Ha en godkänd tidsangivelse.
2. Ge användaren möjlighet att ställa in tiden.
3. Ge användaren möjlighet att läsa av tiden,
4. Ge användaren möjlighet att ställa in ett alarm.
5. Ge användaren möjlighet att läsa av vilken tid alarmet är satt till samt om alarmet är på eller av.
6. Generera en alarmsignal när alarmet aktiveras.
7. Ge användaren möjlighet att använda snooze-funktionen tre gånger.
8. Generera ett problem på displayen vilket användaren måste lösa för att stänga av alarmet.
9. Visa hur många gånger snooze-funktionen har använts med hjälp av lampor.

När kraven fastställts producerades ett abstrakt kopplingsschema och passande komponenter valdes. Därefter skapades ett exakt kopplingsschema med hjälp av programmet PowerLogic som användes som mall för själva monterandet (se bilaga 2).

Konstruktion

För att konstruera väckarklockan användes följande komponenter. Komponenterna löddes fast i en monteringsskiva och sladdar mellan komponenterna surrades fast.

Processor

ATmega16 8bit Microcontroller med 16kb programmerbart flashminne, 1kb SRAM och ett JTAG interface för sammankoppling med dator. Processorn utgör grunden för väckarklockan, utför alla beräkningar och exekveringar samt kopplar samman de övriga komponenterna med varandra.

Display

En LCD skärm av modell SHARP DotMatrix, LCD Alfanumerisk teckendisplay har använts för att visa tiden och för att kommunicera med användaren.

Hexadecimal knappsats

För att sköta inställning av väckarklockan utrustas den med en 4x4 hexadecimal knappsats. Varje rad och kolumn är kopplade till en pin på processorn och aktiveras vid knapptryckning.

Summer

En summer används för att generera alarm-ljudet i väckarklockan. En summer är en elektrisk komponent som kan framställa enklare ljud när den förses med ström med rätt frekvens.

Kristalloscillator

Eftersom väckarklockan kräver en mer exakt tidsåtergivelse än vad som kan erbjudas av processorn används en extern oscillerande kristall för ändamålet.

LED

Tre LED-lampor har använts för att visa hur många gången snooze-funktionen har använts.

Strömbrytare (Snooze-knapp)

För att öka användarvänligheten finns en separat snooze-knapp. Knappen är en enkel strömbrytare som sluter en krets när den trycks ned.

Resistorer

Fyra stycken resistorer har använts i konstruktionen. Tre stycken har kopplats till LED-lamporna och en har kopplats till snooze-knappen.

Kondensator

En kondensator har kopplats till snooze-knappen för att förbättra avläsningen vid nedtryckningar.

Programvaruutveckling

Programkoden har skrivits uteslutande i C. Nedan följer korta kommentarer kring utvalda delar av programmets källkod. Fullständig källkod finns i bilaga 3.

Uppstart:

När programmet startas genomförs en setup vilken sätter portarna till ut- eller insignal samt aktiverar summer och display. Vidare startas metoden setTime() och tidsräkningen aktiveras. Därefter går programmet in i main-loopen och ger användaren möjlighet att använda väckarklockan.

Tid:

För att öka precisionen i tidsangivelsen används en extern kristall på 16 MHz. För att hantera detta i källkoden utnyttjas processorns inbyggda timerfunktioner där ett register räknas upp vid varje klockcykel. Uppräkningen av registret sker från 3036 till 65536 vilket innebär att fyra uppräkningar motsvarar en sekund. Vid var fjärde uppräkning ökas därför tidsvariablerna med en sekund.

Summer:

För att få summern att låta på önskat vis används en inbyggd funktion i processorn vilken skickar utsignaler med en vald frekvens. Den valda frekvensen bestämdes för att matcha summerns egenfrekvens.

Display:

Vid uppstart förbereds displayen genom att grundläggande inställningar görs. När tecken ska skrivas till displayen skickas en binärkod motsvarande det tecken som ska visas. För att kontrollera att displayen är redo att ta emot ny information används metoden isBusy() som läser av displayens busy flag.

Knappsats

För att avgöra vilken knapp som är nedtryckt på knappsatsen loopas en signal över de fyra raderna och de fyra kolonnerna läses av. För att garantera att det alltid finns en signal att läsa av används processorns inbyggda pull-up-motstånd.

Testning

För testning och felsökning i koden har JTag använts. Testningen har delvis skett löpande genom hela projektet men intensifierades när produkten närmade sig sin slutgiltiga form. I mån av behov har Atmel Studios debugger använts för att identifiera var oönskade händelser inträffar. Samtliga krav i kravspecifikationen uppfylldes vid avslutande testning.

Bilagor:

Instruktionsmanual

Uppstart:

Koppla in väckarklockan till ett 5V nätaggregat. På skärmen visas funktionen för att ställa in tiden.

Ställ in tiden:

För att komma till funktionen för att ställa in tiden tryck på A. Vid tidsinställning anges önskat klockslag genom att trycka på en siffra. Bekräfta valet av varje siffra genom att trycka på den röda (snooze) knappen.

Ställ in alarm:

För att komma till funktionen för att ställa in alarmet tryck på B. Vid alarminställning anges önskat klockslag genom att trycka på en siffra. Bekräfta valet av varje siffra genom att trycka på den röda (snooze) knappen. Efter att alarmtiden har ställts in ges möjlighet att ställa in önskad snooze-tid. Bekräfta som tidigare med den röda (snooze) knappen.

Sätt alarm på/av:

För att inaktivera/aktivera alarmfunktionen tryck på C. Ett A i högra hörnet på displayen indikerar att alarmet är aktiverat.

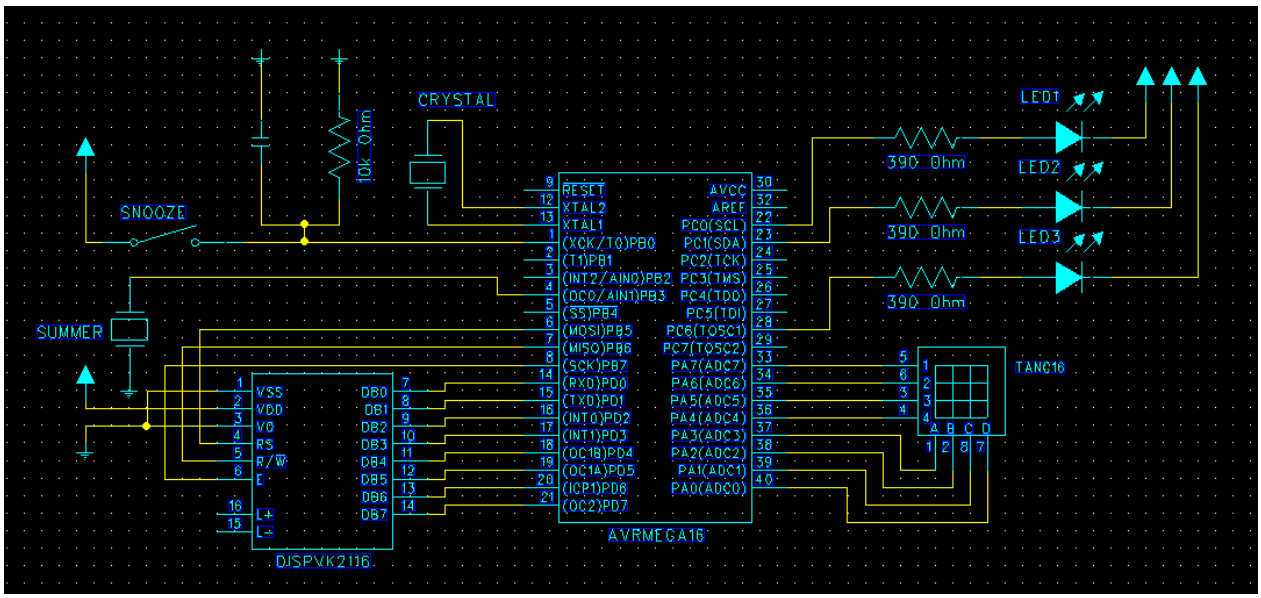
Snooze:

När alarmet gått tryck på den röda (snooze) knappen för att skjuta upp alarmet med den valda snooze-tiden. Snooze-funktionen kan användas maximalt tre gånger och de röda lamporna indikerar hur många gånger den använts.

Stäng av alarm:

För att permanent stänga alarmet tryck på F efter att alarmet gått. Ett problem presenteras då på displayen. Lös problemet och avsluta varje inmatning genom att trycka på den röda (snooze) knappen.

Kopplingsschema



Källkod

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <stdlib.h>
#include <math.h>
volatile uint8_t totOverflow;

char isAlarmOn = 0;
char snoozeCount = 0;
int snoozeTime = 10;
int password = 1000;
int interupptCounter=0;
int soundCounter=0;
int isAlarmRinging=0;
int res=-1;

int H1 = 0;
int H2 = 0;
int M1 = 0;
int M2 = 0;
int S1 = 0;
int S2 = 0;

int SNM1 = 0;
int SNM2 = 0;
```

```
int AH1 = 0;
int AH2 = 0;
int AM1 = 0;
int AM2 = 0;

void timerStart()
{
	// set up timer with prescaler = 64;
	TCCR1B = 0b00000011;
	// initialize counter
	TCNT1 = 3036;

	// enable overflow interrupt
	TIMSK |= (1 << TOIE1);

	// enable global interrupts
	sei();

	// initialize overflow counter variable
	totOverflow = 0;
}

void incrementTime () {
	if (S2<9) {
		S2++;
	}
	else {
		S2=0;
		if (S1<5) {
			S1++;
		}
		else {
			S1=0;
			if (M2<9) {
				M2++;
			}
			else {
				M2=0;
				if (M1<5) {
					M1++;

				}
				else {
					M1=0;
					if (H1<2) {
						if(H2<9) {
							H2++;
						}
						else {
							H2=0;
							H1++;
						}
					}
					else {
						if (H2<3) {
							H2++;
							}else {
							H1=0;
```

```
                                                H2=0;
                                                M1=0;
                                                M2=0;
                                                S1=0;
                                                S2=0;
                                        }

                                }
                        }
                }
        }
}



ISR(TIMER1_OVF_vect)
{
        // keep a track of number of overflows
        totOverflow++;
        //counter used to introduce delays
        interupptCounter++;
        soundCounter++;


        // 4 overflows = 1 seconds delay (approx.)
        if (totOverflow >= 4) {
                incrementTime();
                // reset overflow counter
                totOverflow = 0;
        }
        TCNT1 = 3036;
}

char maskB(){
        char currentB=PORTB;
        char bLastFive=currentB & 0b00011111;
        return bLastFive;
}

void isBusy(){
        DDRD=0b00000000;
        char bLastFive=maskB();
        PORTB = 0b01000000 | bLastFive;
        PORTB = 0b11000000 | bLastFive;
        char value = PIND;
        char result = value & 0b10000000;

        while (result==0b10000000) {
                PORTB = 0b01000000 | bLastFive;
                PORTB = 0b11000000 | bLastFive;
                value = PIND;
                result = value & 0b10000000;
        }
        DDRD=0b11111111;

}
```

```
void setDisplayFunctionSet () {
	//sätter displayen i function set
	isBusy();
	char bLastFive=maskB();
	PORTB=0b10000000|bLastFive; //Sätter E high
	PORTD=0b00111000; //Sätter function set
	PORTB=0b00000000|bLastFive; //Sätter E low
}

void turnOnDisplay () {
	//sätter på displayen, fixar med cursor
	char bLastFive=maskB();
	isBusy();
	PORTB=0b10000000|bLastFive;
	PORTD=0b00001111;
	PORTB=0b00000000|bLastFive;
}

void clearDisplay() {
	//clearar display
	isBusy();
	char bLastFive=maskB();
	PORTB=0b10000000|bLastFive; //Sätter E high
	PORTD=0b00000001;
	PORTB=0b00000000|bLastFive; //Sätter E low
}

void returnHomeDisplay () {
	char bLastFive=maskB();
	isBusy();
	PORTB=0b10000000|bLastFive;
	PORTD=0b00000010;
	PORTB=0b00000000|bLastFive;
}

void setCursorPosition(int value)
{
	if(value<0b01111111){
		isBusy();
		char bLastFive = maskB();
		PORTB=0b10000000|bLastFive; //Sätter E high
		PORTD=0b10000000 |value;
		PORTB=0b00000000|bLastFive; //Sätter E low
	}
}

void cursorOn () {
	char bLastFive=maskB();
	isBusy();
	PORTB=0b10000000|bLastFive;
	PORTD=0b00001111;
	PORTB=0b00000000|bLastFive;
}

void cursorOff () {
	char bLastFive=maskB();
	isBusy();
	PORTB=0b10000000|bLastFive;
```

```
	PORTD=0b00001100;
	PORTB=0b00000000|bLastFive;
}

void startUp(){

	//Ställer in om bits är ut eller in

	DDRA=0b11110000;
	DDRB=0b11101000;
	DDRC=0b01000011;
	DDRD=0b11111111;
	OCR0=0b01000000; // Förbereder summer;
	setDisplayFunctionSet();
	clearDisplay();
	turnOnDisplay();
	returnHomeDisplay();
}

int printChar (int value) {
	isBusy();
	char bLastFive=maskB();
	PORTB=0b10100000|bLastFive; //Sätter E high och i skrivläge
	PORTD=value;
	PORTB=0b00000000|bLastFive; //Sätter E low
	return value;
}

void printNumber (int nbr) {
	switch (nbr) {
		case 0:
		printChar(0b00110000);
		break;

		case 1:
		printChar(0b00110001);
		break;

		case 2:
		printChar(0b00110010);
		break;

		case 3:
		printChar(0b00110011);
		break;

		case 4:
		printChar(0b00110100);
		break;

		case 5:
		printChar(0b00110101);;
		break;

		case 6:
		printChar(0b00110110);
		break;
```

```
			case 7:
			printChar(0b00110111);
			break;

			case 8:
			printChar(0b00111000);
			break;

			case 9:
			printChar(0b00111001);
			break;
		}
}

int checkKeypad() {

		// returnerar int (-1)-15 där -1 står för ingen knapptryckning.

		//kollar kolumn 1
		PORTA=0b11101111;
		char value = PINA;
		char result = value & 0b00001111;

		switch(result) {
			case 0b00001110:
			while (1) {
				_delay_ms(50);
				value = PINA;
				result = value & 0b00001111;
				if (result!=0b00001110) {
					return 0;
				}
			}

			case 0b00001101:
			while (1) {
				_delay_ms(50);
				value = PINA;
				result = value & 0b00001111;
				if (result!=0b00001101) {
					return 4;
				}
			}

			case 0b00001011:
			while (1) {
				_delay_ms(50);
				value = PINA;
				result = value & 0b00001111;
				if (result!=0b00001011) {
					return 8;
				}
			}

			case 0b00000111:
			while (1) {
				_delay_ms(50);
				value = PINA;
```

```
			result = value & 0b00001111;
			if (result!=0b00000111) {
				return 12;
			}
		}
}

//kollar kolumn 2
PORTA=0b11011111;
value=PINA;
result= value & 0b00001111;

switch(result) {
		case 0b00001110:
		while (1) {
			_delay_ms(50);
			value = PINA;
			result = value & 0b00001111;
			if (result!=0b00001110) {
				return 1;
			}
		}

		case 0b00001101:
		while (1) {
			_delay_ms(50);
			value = PINA;
			result = value & 0b00001111;
			if (result!=0b00001101) {
				return 5;
			}
		}

		case 0b00001011:
		while (1) {
			_delay_ms(50);
			value = PINA;
			result = value & 0b00001111;
			if (result!=0b00001011) {
				return 9;
			}
		}

		case 0b00000111:
		while (1) {
			_delay_ms(50);
			value = PINA;
			result = value & 0b00001111;
			if (result!=0b00000111) {
				return 13;
			}
		}
}

//kollar kolumn 3
PORTA=0b10111111;
value=PINA;
result= value & 0b00001111;
```

```
switch(result) {
	case 0b00001110:
	while (1) {
		_delay_ms(50);
		value = PINA;
		result = value & 0b00001111;
		if (result!=0b00001110) {
			return 2;
		}
	}
	case 0b00001101:
	while (1) {
		_delay_ms(50);
		value = PINA;
		result = value & 0b00001111;
		if (result!=0b00001101) {
			return 6;
		}
	}

	case 0b00001011:
	while (1) {
		_delay_ms(50);
		value = PINA;
		result = value & 0b00001111;
		if (result!=0b00001011) {
			return 10;
		}
	}

	case 0b00000111:
	while (1) {
		_delay_ms(50);
		value = PINA;
		result = value & 0b00001111;
		if (result!=0b00000111) {
			return 14;
		}
	}
}

//kollar kolumn 4
PORTA=0b01111111;
value=PINA;
result= value & 0b00001111;

switch(result) {
	case 0b00001110:
	while (1) {
		_delay_ms(50);
		value = PINA;
		result = value & 0b00001111;
		if (result!=0b00001110) {
			return 3;
		}
	}
```

```
		case 0b00001101:
		while (1) {
			_delay_ms(50);
			value = PINA;
			result = value & 0b00001111;
			if (result!=0b00001101) {
				return 7;
			}
		}

		case 0b00001011:
		while (1) {
			_delay_ms(50);
			value = PINA;
			result = value & 0b00001111;
			if (result!=0b00001011) {
				return 11;
			}
		}

		case 0b00000111:
		while (1) {
			_delay_ms(50);
			value = PINA;
			result = value & 0b00001111;
			if (result!=0b00000111) {
				return 15;
			}
		}
	}
	return -1;
}

int buttonClicked () {
	char value = PINB;
	char result = value & 0b00000001;
	if(result==0b00000001) {
		while (1) {
			_delay_ms(50);
			value = PINB;
			result = value & 0b00000001;
			if (result==0b00000000) {
				return 1;
			}
		}
	}
	return 0;
}


int setH1 () {
	setCursorPosition(64);
	int result=0;
	while (1) {
		switch (checkKeypad()) {
			case 0:
			printChar(0b00110000);
			result = 0;
```

```
                        setCursorPosition(64);
                        break;

                        case 1:
                        printChar(0b00110001);
                        result = 1;
                        setCursorPosition(64);
                        break;

                        case 2:
                        printChar(0b00110010);
                        result = 2;
                        setCursorPosition(64);
                        break;
                }
                if(buttonClicked())
                {
                        return result;
                }
        }
}

int setH2 () {

        setCursorPosition(65);
        int result=0;

        while (1) {

                switch (checkKeypad()) {
                        case 0:
                        printChar(0b00110000);
                        result = 0;
                        setCursorPosition(65);
                        break;

                        case 1:
                        printChar(0b00110001);
                        result = 1;
                        setCursorPosition(65);
                        break;

                        case 2:
                        printChar(0b00110010);
                        result = 2;
                        setCursorPosition(65);
                        break;

                        case 3:
                        printChar(0b00110011);
                        result = 3;
                        setCursorPosition(65);
                        break;

                        case 4:
                        if (H1!=2) {
                                printChar(0b00110100);
                                result = 4;
```

```
                        setCursorPosition(65);
                }
                break;

                case 5:
                if (H1!=2) {
                        printChar(0b00110101);
                        result = 5;
                        setCursorPosition(65);
                }
                break;
                case 6:
                if (H1!=2) {
                        printChar(0b00110110);
                        result = 6;
                        setCursorPosition(65);
                }
                break;

                case 7:
                if (H1!=2) {
                        printChar(0b00110111);
                        result = 7;
                        setCursorPosition(65);
                }
                break;

                case 8:
                if (H1!=2) {
                        printChar(0b00111000);
                        result = 8;
                        setCursorPosition(65);
                }
                break;
                case 9:
                if (H1!=2) {
                        printChar(0b00111001);
                        result = 9;
                        setCursorPosition(65);
                }
                break;
        }
        if(buttonClicked())
        {
                return result;
        }
    }
}

int setM1 (int pos) {

    setCursorPosition(pos);
    int result=0;

    while (1) {

            switch (checkKeypad()) {
```

```
                        case 0:
                        printChar(0b00110000);
                        result = 0;
                        setCursorPosition(pos);
                        break;

                        case 1:
                        printChar(0b00110001);
                        result = 1;
                        setCursorPosition(pos);
                        break;

                        case 2:
                        printChar(0b00110010);
                        result = 2;
                        setCursorPosition(pos);
                        break;

                        case 3:
                        printChar(0b00110011);
                        result = 3;
                        setCursorPosition(pos);
                        break;

                        case 4:
                        printChar(0b00110100);
                        result = 4;
                        setCursorPosition(pos);
                        break;

                        case 5:
                        printChar(0b00110101);
                        result = 5;
                        setCursorPosition(pos);
                        break;
                }
                if(buttonClicked())
                {
                        return result;
                }
        }
}

int setM2 (int pos) {

        setCursorPosition(pos);
        int result=0;

        while (1) {

                switch (checkKeypad()) {
                        case 0:
                        printChar(0b00110000);
                        result = 0;
                        setCursorPosition(pos);
                        break;

                        case 1:
```

```
                printChar(0b00110001);
                result = 1;
                setCursorPosition(pos);
                break;

                case 2:
                printChar(0b00110010);
                result = 2;
                setCursorPosition(pos);
                break;

                case 3:
                printChar(0b00110011);
                result = 3;
                setCursorPosition(pos);
                break;

                case 4:
                printChar(0b00110100);
                result = 4;
                setCursorPosition(pos);
                break;

                case 5:
                printChar(0b00110101);
                result = 5;
                setCursorPosition(pos);
                break;

                case 6:
                printChar(0b00110110);
                result = 6;
                setCursorPosition(pos);
                break;

                case 7:
                printChar(0b00110111);
                result = 7;
                setCursorPosition(pos);
                break;

                case 8:
                printChar(0b00111000);
                result = 8;
                setCursorPosition(pos);
                break;
                case 9:
                printChar(0b00111001);
                result = 9;
                setCursorPosition(pos);
                break;
        }
        if(buttonClicked())
        {
                return result;
        }

}
```

```
}
void showAlarmActive(int on){
	setCursorPosition(15);
	if(on==1) {
		printChar(0b01000001);
	}
	if (on==0) {
		printChar(0b00100000);
	}
}

void toggleAlarm(){
	if(isAlarmOn==1){
		isAlarmOn=0;
		showAlarmActive(0);
	}
	else{
		isAlarmOn=1;
		showAlarmActive(1);
	}
}

 void setTime(){
	setCursorPosition(0);
	printChar(0b01010011); //S
	printChar(0b01100101); //e
	printChar(0b01110100); //t
	printChar(0b00100000);
	printChar(0b01110100); //t
	printChar(0b01101001); //i
	printChar(0b01101101); //m
	printChar(0b01100101); //e
	printChar(0b00111010); //:

	cursorOn();
	setCursorPosition(64);
	printNumber(0);
	printNumber(0);
	printChar(0b00111010);
	printNumber(0);
	printNumber(0);
	setCursorPosition(64);
	H1=setH1();
	H2=setH2();
	M1=setM1(67);
	M2=setM2(68);
	S1=0;
	S2=0;
	clearDisplay();
	//toggleAlarm();
}

void showTime (int displayPos) {
	setCursorPosition(displayPos);

	printNumber(H1);
	printNumber(H2);
	printChar(0b00111010);
```

```
        printNumber(M1);
        printNumber(M2);
        printChar(0b00111010);
        printNumber(S1);
        printNumber(S2);
        cursorOff();
}


void summerOn(){
        TCCR0=0b00011011;
}

void summerOff(){

        TCCR0=0b00001011;
}

void lightOneOn(){
        PORTC=PORTC | 0b00000001;
}

void lightOneOff(){
        PORTC=PORTC & 0b11111110;
}

void lightTwoOn(){
        PORTC=PORTC | 0b00000010;
}

void lightTwoOff(){
        PORTC=PORTC & 0b11111101;
}

void lightThreeOn(){
        PORTC=PORTC | 0b01000000;
}

void lightThreeOff(){
        PORTC=PORTC & 0b10111111;
}

void showAlarm(int displayPos){

        setCursorPosition(displayPos);

        printNumber(AH1);
        printNumber(AH2);
        printChar(0b00111010);
        printNumber(AM1);
        printNumber(AM2);

        cursorOff();

}

void setSnoozeTime() {
        clearDisplay();
```

```
        setCursorPosition(0);
        printChar(0b01010011); //S
        printChar(0b01100101); //e
        printChar(0b01110100); //t
        printChar(0b00100000);
        printChar(0b01110011); //s
        printChar(0b01101110); //n
        printChar(0b01101111); //o
        printChar(0b01101111); //o
        printChar(0b01111010); //z
        printChar(0b01100101); //e
        printChar(0b00100000);
        printChar(0b01110100); //t
        printChar(0b01101001); //i
        printChar(0b01101101); //m
        printChar(0b01100101); //e
        printChar(0b00111010); //:

        cursorOn();
        setCursorPosition(64);
        printNumber(0);
        printNumber(0);
        printChar(0b00100000);
        printChar(0b00101000); //(
        printChar(0b01101101); //m
        printChar(0b01101001); //i
        printChar(0b01101110); //n
        printChar(0b00101001); //)
        setCursorPosition(64);
        SNM1=setM1(64);
        SNM2=setM2(65);

        clearDisplay();
}


void setAlarmTime(){
        setCursorPosition(0);
        printChar(0b01010011); //S
        printChar(0b01100101); //e
        printChar(0b01110100); //t
        printChar(0b00100000);
        printChar(0b01100001); //a
        printChar(0b01101100); //l
        printChar(0b01100001); //a
        printChar(0b01110010); //r
        printChar(0b01101101); //m
        printChar(0b00111010); //:

        showAlarm(64);
        cursorOn();
        setCursorPosition(64);
        AH1=setH1();
        AH2=setH2();
        AM1=setM1(67);
        AM2=setM2(68);
        clearDisplay();
        setSnoozeTime();
```

```
	isAlarmOn=1;
	showAlarmActive(1);
}


void alarmOff(){
	summerOff();
	isAlarmOn=0;
	isAlarmRinging=0;
	lightOneOff();
	lightTwoOff();
	lightThreeOff();
	snoozeCount=0;
}


void snooze(){

	if(snoozeCount<3){
		summerOff();
		isAlarmRinging=0;
		switch(snoozeCount){
			case 0:
			lightOneOn();
			break;

			case 1:
			lightTwoOn();
			break;

			case 2:
			lightThreeOn();
			break;
		}
		snoozeCount++;

		int tempTime = M1*10+M2 + SNM1*10+SNM2;
		if (tempTime<60) {
			AH1=H1;
			AH2=H2;
			AM1=tempTime/10;
			AM2=tempTime%10;
		}

		else {
			if (H1<2) {
				if (H2<9) {
					AH1=H1;
					AH2=H2+1;
					tempTime=tempTime-60;
					AM1=tempTime/10;
					AM2=tempTime%10;
				}
				else {
					AH1=H1+1;
					AH2=0;
					tempTime=tempTime-60;
```

```
                                        AM1=tempTime/10;
                                        AM2=tempTime%10;
                                }
                        }
                        else {
                                if (H2<3) {
                                        AH1=H1;
                                        AH2=H2+1;
                                        tempTime=tempTime-60;
                                        AM1=tempTime/10;
                                        AM2=tempTime%10;
                                }
                                else {
                                        AH1=0;
                                        AH2=0;
                                        tempTime=tempTime-60;
                                        AM1=tempTime/10;
                                        AM2=tempTime%10;
                                }
                        }
                }
        }
}



void playGame () {
        clearDisplay();
        cursorOn();
        setCursorPosition(0);
        printChar(0b01010100); //T
        printChar(0b01101111); //o
        printChar(0b00100000);
        printChar(0b01100100); //d
        printChar(0b01100101); //e
        printChar(0b01100011); //c
        printChar(0b01101001); //i
        printChar(0b01101101); //m
        printChar(0b01100001); //a
        printChar(0b01101100); //l
        printChar(0b00111010); //:


        int b0=rand()%2;
        int b1=rand()%2;
        int b2=rand()%2;
        int b3=rand()%2;
        int b4=rand()%2;
        int b5=rand()%2;
        int result = b0*pow(2,0) + b1*pow(2,1) + b2*pow(2,2) + b3*pow(2,3) + b4*pow(2,4) +
b5*pow(2,5);
        setCursorPosition(64);
        printNumber(b5);
        printNumber(b4);
        printNumber(b3);
        printNumber(b2);
        printNumber(b1);
        printNumber(b0);
```

```
	printChar(0b00111010); //:
	printChar(0b00100000);
	printNumber(0);
	printNumber(0);
	setCursorPosition(72);
	res = setM2(72)*10 + setM2(73);

	while (res!=result) {
		setCursorPosition(72);
		printNumber(0);
		printNumber(0);
		setCursorPosition(72);
		res = setM2(72)*10 + setM2(73);
	}
	clearDisplay();
}


void alarmOn()
{
	summerOn();
	isAlarmRinging=1;

	while(1)
	{

		if (soundCounter>2 && isAlarmOn==1 && isAlarmRinging==1) {
			summerOff();
		}
		if (soundCounter>4 && isAlarmOn==1&& isAlarmRinging==1) {
			summerOn();
			soundCounter=0;
		}
		_delay_ms(100);
		//checkAlarm();

			if (H1==AH1 && H2==AH2 && M1==AM1 && M2==AM2 && S1==0 && S2==0 &&
isAlarmOn==1) {
					summerOn();
					isAlarmRinging=1;
			}


		showTime(0);
		if(buttonClicked()&& isAlarmRinging==1){
			snooze();
		}

		if (checkKeypad()==15) {
			playGame();
			alarmOff();
			break;
		}

		if (snoozeCount>2 && isAlarmRinging==1)
		{
			summerOn();
			playGame();
```

```
                        alarmOff();
                        break;

                }

        }

}

void checkAlarm () {
        if (H1==AH1 && H2==AH2 && M1==AM1 && M2==AM2 && S1==0 && S2==0 && isAlarmOn==1) {
                alarmOn();
        }
}


int main (void) {

        startUp();
        setTime();
        timerStart();


        while (1) {
                if (interupptCounter>=1) {


                        showTime(0);

                        checkAlarm();
                        switch (checkKeypad()) {
                                case 10:
                                setTime();
                                break;
                                case 11:
                                setAlarmTime();
                                break;
                                case 12:
                                toggleAlarm();
                                break;

                        }
                        interupptCounter=0;
                }
        }
        return 0;
}
```