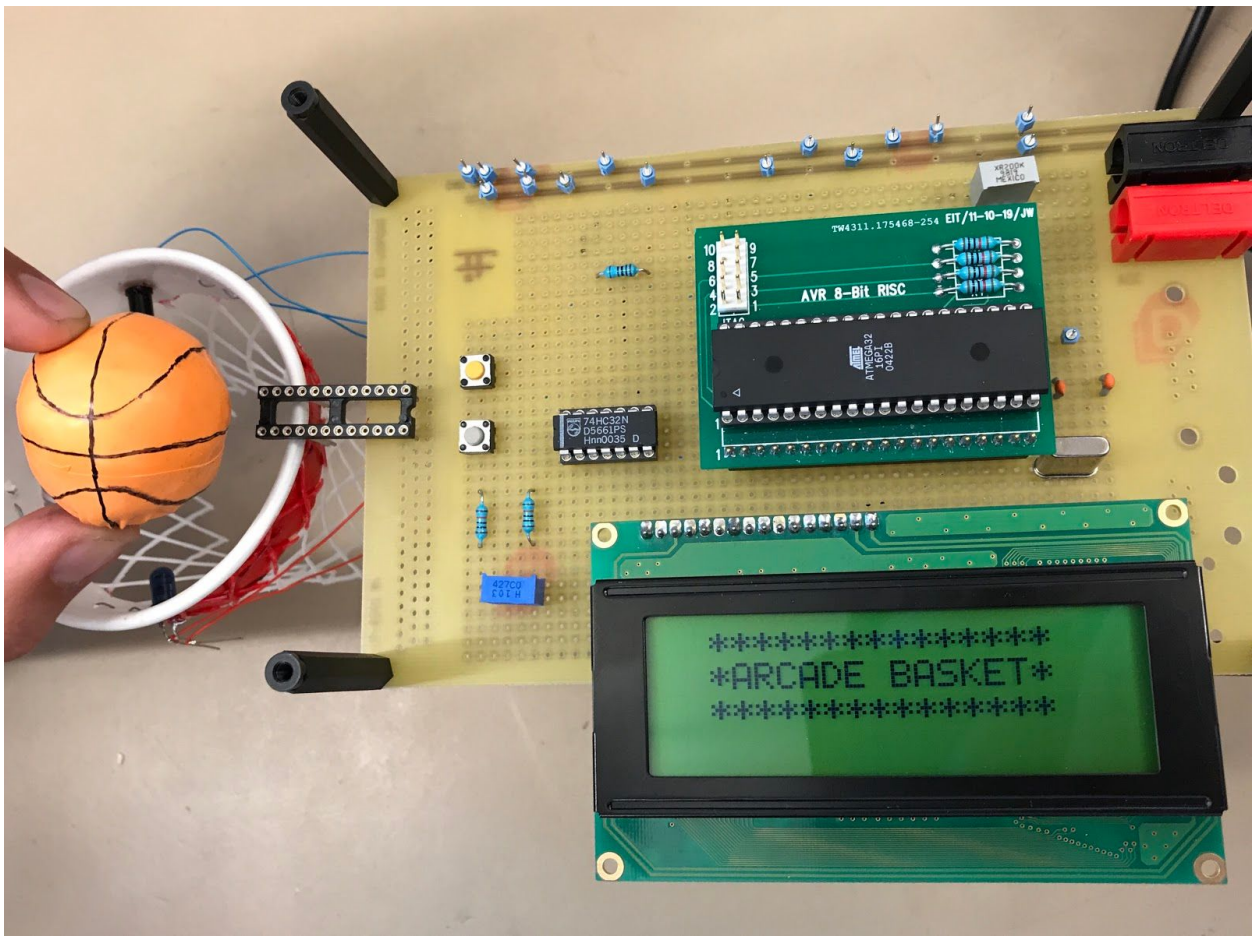
Digitala Projekt:

Arcade Basket



Digitala Projekt, EITF11 - VT17
Oscar Ahlgren, Patrik Lorentsson och Sten Hellberg
Handledare: Bertil Lindvall

Industriell Ekonomi, Lunds Tekniska Högskola
2017-05-22

Abstract

This project is a part of the course Digital Projects at Faculty of Engineering, LTH, at Lund University. The project (and the course) aims to give an understanding for electronic and digital systems and how to construct a digital product. This is accomplished during the project, mainly by gathering knowledge about the hardware and software and their interaction between each other. As our project, we decided to build an arcade basket hoop. The electronic basket hoop offers two game modes, both including registering of hits during time countdown. This report will give an insight into the project and make it possible for a person not involved in the project to continue the project where we finished it.

Innehållsförteckning

1 Inledning

Nedan ges en introduktion till projektet och dess syfte.

1.1 Bakgrund

Det här projektet ingår i kursen Digitala Projekt vid Lunds Tekniska Högskola. Kursdeltagarna fick, med begränsningen att projektet skulle innehålla såväl elektroteknik som digitalteknik, fritt välja en produkt som skulle konstrueras. Under de tolv veckor som projektet sträckte sig över skulle kursdeltagarna i mindre grupper genomgå stegen från idé till prototyp. Nödvändig hårdvara för att kunna bygga en prototyp av produkten erhölls av kursansvarig.

1.2 Syfte

Syftet med det här projektet är att konstruera en arkadbasketkorg, det vill säga en elektroniska basketkorg som räknar poängen på en display. Därigenom är målet att utveckla förståelsen för hårdvara och mjukvara i elektroniska och digitala system och hur hårdvaran och mjukvaran samverkar i produkter. Vidare ska projektet ge kunskaper och färdigheter vad beträffar framtagning av kravspecifikation, ritande av kopplingsschema, informationsinhämtning ur datablad, lödning, programmering i C med mera.

2 Produktbeskrivning

Produkten ska ha en liten basketkorg, vilken ska kunna registrera träffar. Detta ska ske med hjälp av dioder, som sänder ut respektive tar emot IR-ljus. Med hjälp av två knappar ska man kunna välja mellan två spellägen; normalläge (normal mode) och arkadläge (arcade mode). I normalläge gäller det att göra så många träffar som möjligt på 30 sekunder. Arkadläget ska fungera på snarlikt sätt, med skillnaden att man vid var tredje träff får tre sekunders tidstillägg. Träffarna ska, tillsammans med tiden, skrivas ut på en display. De högsta poängen för respektive spelläge ska sparas i spelet och visas på displayen efter en avslutad omgång.

3 Kravspecifikation

3.1 Funktionella krav

3.1.1 Produkten ska ha en basketkorg med diameter i storleksordningen 5-10 cm.
3.1.2 Produkten ska kunna registrera träffar genom att en IR-stråle bryts.
3.1.3 Spelet skall ha 2 spellägen att tillgå.
3.1.4 Produkten ska ha 2 knappar för att kunna navigera genom spelet.
3.1.5 Produkten ska kunna räkna upp tid.
3.1.6 Spelet ska spara highscore för respektive spelläge.

3.2 Kvalitetskrav

3.2.1 Produkten ska missa att registrera en träff vid maximalt 1 gång på 50.
3.2.2 Produkten ska räkna tid med max 1 sekunds felmarginal på 60 sekunder.
3.2.3 Knapptryckningar ska missas att registreras vid maximalt 1 gång på 100.

4 Hårdvara

Komplett kopplingsschema återfinns i appendix A. De hårdvarukomponenter som använts under projektets gång listas nedan.

4.1 Processor

Processorn som använts till vår prototyp är en Atmel Atmega32. Det är en 8 bitars mikrokontroller med 32 kilobyte flashminne. Processorn har 40 pinnar varav 32 är fördelade mellan 4 olika portar, PORT A till PORT D.

4.2 Display

Den LCD display som använts är en 4 raders alfanumerisk display av typen Sharp Dot Matrix, BC2004A-serien. För att styra kontrasten på displayen har vi använt en spänningsregulator.

4.3 Logisk grind

För att styra avbrott från knapparna har en logisk grind kopplats till PORT D. Grinden är av typen Quad 2-Input OR, SN54HC32.

4.4 Fotodiod

För att registrera när ljusstrålen bryts har vi använt en Fotodiod av typen SFH 203 FA.

4.5 IR-diod

För att generera en ljusstråle har en IR-diod av typen TSUS5400 använts.

4.6 Kristall

En extern 16Mhz kristall har använts för att öka precisionen på processorns interna klocka.

4.7 Knappar

För att styra displayen har 2 st. knappar av standardmodell använts, en grå och en gul.

4.8 Resistorer

För IR-dioden och fotodioden har resistorer av storlek 50 Ω respektive ett reglerbart motstånd på 100kΩ. Till knapparna används 2 st.10 kΩ-resistorer.

4.9 JTag

Användes för att föra över mjukvaran till processorn.

5 Mjukvara

Mjukvaran som laddas över till processorn skrivs i programmeringsspråket C. Fullständig kod hittas i appendix B.

Registreringarna av såväl knapptryckningar som brytningar av IR-strålen drivs av externa avbrott. Knapparna är, via en logisk grind (OR), kopplade till samma processorpinne för externa avbrott. Utöver det är varje knapp också kopplad till var sin pinne på A-bussen. Detta innebär att då ett externt avbrott uppstår på grund av en knapptryckning måste signalerna på A-bussen noteras för att avgöra vilken knapp som gett upphov till avbrottet. Fotodioden (som tar emot IR-strålen) är däremot kopplad till en egen processorpinne för externt avbrott. Processorn registrerar spänning under 1 V som low och spänning över 3 V som high. Således kan brytning av IR-strålen tolkas som ett externt avbrott om spänningen över processorpinnen är över 3 V när IR-dioden lyser på fotodioden och under 1 V när IR-strålen bryts.

För att räkna tid mer exakt än om endast processorns egna klocka skulle ha använts har en extern kristall på 16 MHz kopplats in. I syfte att kunna räkna sekunder användes processorns 8-bitars timer. Prescalern sattes till högsta möjliga värde (=1024), vilket gav det eftersökta antalet uppräkningar lika med 15 625 ($(16\,000\,000 \div 1024)$. 8-bitars timern gör 255 uppräkningar per overflow, vilket innebär att den genererar 61,2745098 ($15625 \div 255$) overflow per sekund. Således skrevs koden så att sekundräknaren tickade upp efter att 61,2745098 overflows genererats.

Displayen består av fyra rader, med plats för 20 tecken på varje rad. Varje teckenplats har en unik adress, vilket gör det möjligt att adressera en specifik plats på displayen. Detta görs genom att ange önskad adress till DDRAM när man befinner sig i kommandoläge. Därefter kan man i skrivläge välja vad som ska skrivas ut på displayen genom att ange önskat tecken på B-portarna.

6 Metod

6.1 Planering

Under första veckan av kursen bestämdes vilken konstruktion som skulle göras, en tidsplan för arbetet och krav på den färdiga konstruktionen. Det bestämdes också att alla gruppdeltagare skulle delta i alla delar av projektet och att större delen av byggandet skulle vara klart innan programmeringen påbörjades.

6.2 Hårdvarukonstruktion

Efter planeringen var det första som gjordes att bestämma vilka hårdvarukomponenter som behövdes för att uppfylla de krav som ställts upp. När komponenterna bestämts ritades ett kopplingsschema upp för hand och därefter kopplades komponenterna samman. För att det skulle vara enkelt att göra ändringar i konstruktionen virades trådarna först och löddes senare fast när gruppen var säker på att kopplingen stämde.

6.3 Programmering

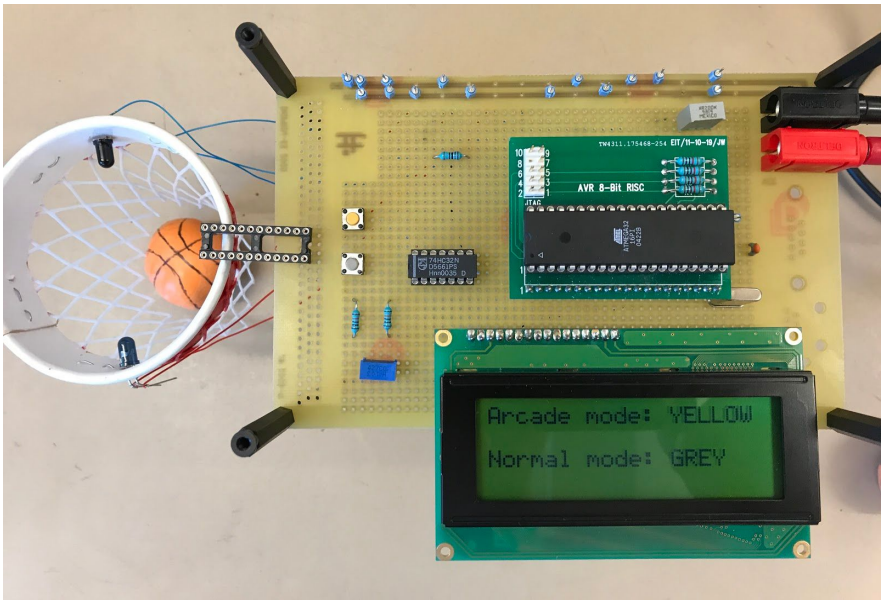
För att inte fastna på detaljnivå i ett tidigt skede av programmeringen var det första som gjordes att identifierades ett antal funktioner som programmet behövde klara av och sedan programmera dessa en i taget. Funktionerna var:

- Att skriva ut text på displayen.
- Att utföra ett specifikt kommando när en specifik knapp tryckts ner.
- Att skriva ut korrekt tid på displayen.
- Att utföra ett kommando när IR-ljuset bröts.

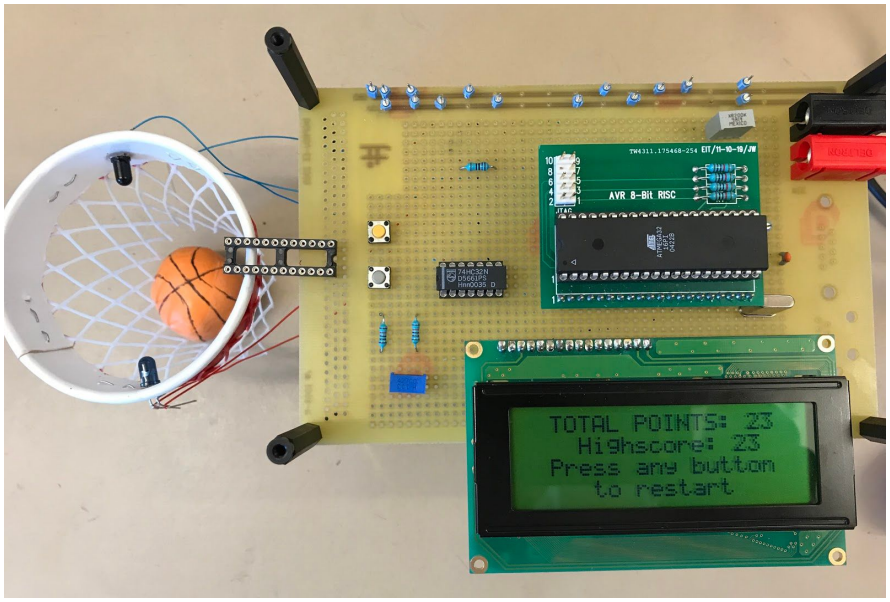
När detta var klart sattes all kod ihop till ett program som fungerade enligt våra krav.

7 Resultat

Projektet resulterade i en småskalig basketkorg som uppfyller alla de krav presenterade i avsnitt 3. Produkten har två olika spellägen (se figur 1) och kan räkna upp poäng allteftersom ljusstrålen bryts, samt hålla koll på det högsta resultatet (se figur 2). Under spelets gång visas korrekt tidtagning och produkten har text som gör programmet lättförståeligt och visuellt tilltalande.



Figur 1: De två spellägena; Arcade Mode och Normal Mode.



Figur 2: Slutsidan efter att ett spel körts.

8 Diskussion

Då vi inför kursen besatt begränsade kunskaper om både elektronik och programmering i C valde vi att begränsa oss till att konstruera en prototyp där mjukvaran utgjorde den stora delen arbetet. Detta eftersom det gav oss en bättre möjlighet att utforma produkten efter vad vi lärt oss under projektets gång. Trots denna begränsade kunskap har projektet gått utmärkt.

Likt många projekt inom ett nytt område tillkom det genomgående motgångar under projektets gång. Allt från små felkopplingar av IR- och fotodiod som gjorde att vi inte kunde generera en ljusstråle att bryta till en icke fungerande JTag som förhindrade oss att överföra vår kod från Atmel till vår processor.

Det stora tidskrävande arbetet i detta projekt har dock varit att lära sig att hämta relevant information ur respektive datablad. Då ingen av oss tidigare hade en vana av C eller olika hårdvaror var det svårt att leta fram information som bestämde motståndsstyrka eller vilken funktion som skulle användas då flera fanns att tillgå.

Vid ett andra försök att ta sig an detta projekt hade vi förhoppningsvis kunnat skala upp produkten för att göra den användarvänligare samt fokuserat mer på att skriva effektiv kod exempelvis genom att använda sig av flaggor för att behandla olika processer istället för delays. Detta hade gjort det enklare för oss och för den som vill fortsätta på produkten att göra förbättringar och utveckla den och vi hade kunnat undvika att behöva byta ut vår ursprungliga Atmega16 processor mot en Atmega32 då vi var i behov av extra minnesplats vid slutkodningen av projektet.

9 Slutsats

Detta projekt har gett oss en ökad förståelse och ödmjukhet inför det arbete som ligger till grund för nya produkter och komplexiteten vid framtagandet. Vikten av att kunna felsöka på mer än ett sätt är något vi blev tvungna att lära oss den hårda vägen då det ofta var väldigt svårt att veta om felet var ett hårdvarufel, JTAG-fel eller ett kodfel.

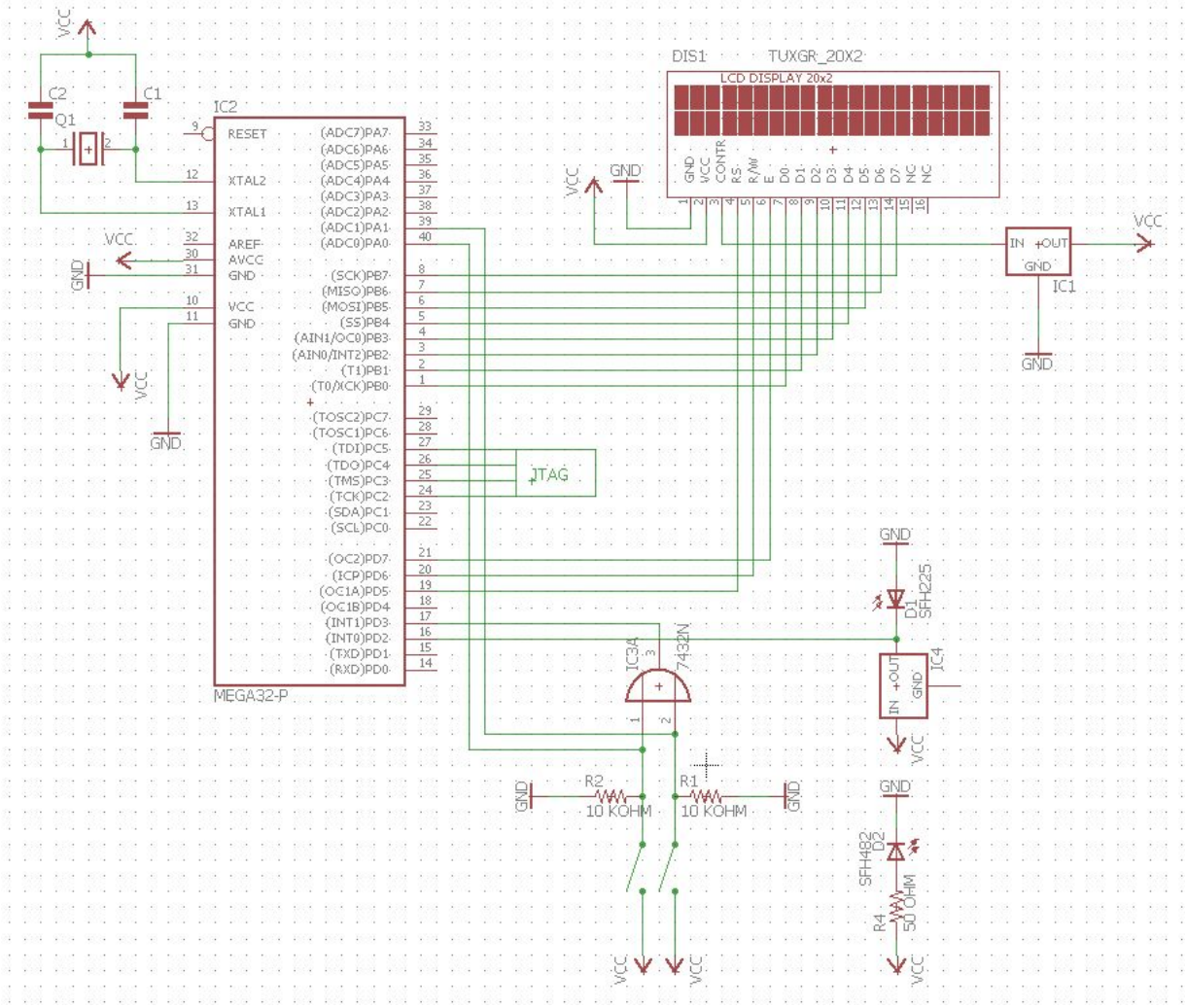
Trots den stora mängd hjälp som krävts från handledarna för att kunna producera en prototyp har arbetet varit väldigt roligt och lärorikt och vi är överlag väldigt nöjda med att kunna presentera vår version av en arkadbasketkorg.

10 Referenser

Datablad för Atmega32 processor:
http://www.atmel.com/images/doc2503.pdf

Datablad för Sharp-Dot-Matrix:
http://www.dema.net/pdf/bolymin/BC2004A-series_VER04.pdf

Appendix A



Figur 1: Kopplingsschema

Appendix B

```
/*
 * ArkadBasket.c
 *
 * Created: 2017-03-24 13:16:38
 * Author : ine13plo
 */

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
int anybuttonpressed=0;
int gamestarted=0;
int irsensor=0;
int points=0;
int boolint=0;
int boolint2=0;
int boolint3=0;
int greyButtonPressed=0;
int yellowButtonPressed=0;
int menustarted=0;
int arcademode=0;
int pointhit=0;
int arcadehighscore=0;
int normalhighscore=0;

void display_cmd(){
	PORTD= 0b10000000;		// RW, RS -> 0, E -> 1
	PORTD= 0b00000000;		// RW, RS -> 0, E -> 0
	_delay_ms(10);
}

void display_setup(){
	DDRA = 0b11111100;
	PORTA = 0b11111100;
	DDRB = 0b11111111;		// Sätter pinnarna för port B till ettor (ut)
	DDRD = 0b11100000;		// Sätter Pinne 19-21 till ettor
	_delay_ms(10);

	//Function Set
	PORTB= 0b00111000;		// Sätter interface datalängd, antal linjer etc. //ändrade här nu har vi fyra
rader
	display_cmd();
	_delay_ms(10);

	//Display on
	PORTB = 0b00001100;		// Inställningar för displayen
	display_cmd();
```

```
	_delay_ms(10);

	//Entry mode set
	PORTB = 0b00000110;
	display_cmd();
	_delay_ms(10);
}

void display_clear(){
	PORTB= 0b00000001;
	_delay_ms(10);
	display_cmd();
}

void write_string(char str[]) {
	int i = 0;
	while( str[i] != '\0') {		// \0 null character in ASCII
	PORTB = str[i];
	_delay_ms(10);
	PORTD= 0b10100000;			// RS -> 1, E -> 1
	_delay_ms(10);
	PORTD= 0b00100000;			// RS -> 1, E -> 0
	i++;
	}
}


/*****************************************************************
 * Interrupt
 */

void setup_interrupt() {
		MCUCR = MCUCR | 0b00001111;
		GICR = GICR | 0b11000000;

		sei();
}

	ISR(INT1_vect){
		anybuttonpressed=1;
		if((menustarted==1)&&(gamestarted==0)){
		switch(PINA){
			case 0b11111101:		//(Av någon anledning sätts PINA2-PINA7 till ettor när programmet
							//startas även fast de inte används.)
				_delay_ms(10);
				greyButtonPressed=1;
				break;
			case 0b11111110:
				_delay_ms(10);
				yellowButtonPressed=1;
```

```
                                        break;
                                }
                        }
                }

                ISR(INT0_vect){
                irsensor=1;
                }



                /*************************************************************
 * KLOCKAN
 */
volatile int count;
double measured_time = 0.0;
double time_in_seconds =0.0;
double total_time_in_seconds=0.0;

volatile double count2;
double measured_time_2 = 0.0;
double time_in_seconds_2 =0.0;

//Overflow Interrupt (för klockan)
                ISR(TIMER0_OVF_vect) {   // TIMER0_OVF_vect
                        count2++;
                        if(gamestarted==1){
                                count++;                // Räknas upp vid varje interrupt. I det här fallet xxx per sekund
                        }
                }

void clock_setup() {
                TCCR0 = TCCR0 | 0b00000101; // Prescaler 1024   (Sida 81,83 i databladet.)
                TIMSK = TIMSK | 0b00000001; // Overflow interrupt aktivt.  (Sida 83 i databladet.)
}


void start_race(){
                count = 0;
                points=0;
}

void starttext(){
                count2 = 0;
                display_clear();
                PORTB = 0b11000000;                                             // Byt rad till rad 2
                _delay_ms(20);
                display_cmd();
                _delay_ms(10);
                write_string("    GET READY");
```

```
                while(count2/61.2745098<3){
                //Inget ska hända i tre sekunder
                }

        display_clear();
        count2 = 0;

        for (int i = 3; i>0 ; i--){
                 PORTB = 0b11000000;                            // Byt rad till rad 2
                 _delay_ms(20);
                 display_cmd();
                 _delay_ms(10);
                 char str[5];
                 sprintf(str, "%d", i);                 //Int till char
                 write_string("       ");
                 write_string(str);

                while(count2/61.2745098<1){
                        //Inget ska hända i en sekund
                }
                count2 = 0;
                display_clear();
        }
}

void main(void){
        display_setup();

        display_clear();

        setup_interrupt();

        clock_setup();

        start_race();

        write_string("  ***************  ");

        // Byt rad till rad 2
        PORTB = 0b11000000;
        _delay_ms(20);
        display_cmd();
        _delay_ms(10);

        write_string("  *ARCADE BASKET*");

        // Byt rad till rad 3
        PORTB = 0b10010100;
        _delay_ms(20);
        display_cmd();
```

```
	_delay_ms(10);
	write_string("  ***************  ");

	while(count2/61.2745098<3){
		//Inget ska hända i tre sekunder
	}

	display_clear();
	write_string(" Press a button to");
	write_string("       start!");
_delay_ms(300);

	while (1) {

		if((anybuttonpressed==1)&&(gamestarted==0)&&(menustarted==0)){
			display_clear();
			menustarted=1;
			write_string("Arcade mode: YELLOW Normal mode: GREY");
		}

		 if((yellowButtonPressed==1)&&(menustarted==1)&&(gamestarted==0)){
			display_clear();

			// Byt rad till pos 44
			PORTB = 0b11000100;
			_delay_ms(20);
			display_cmd();
			_delay_ms(10);
			 write_string("Arcade mode");

			 arcademode=1;
			 starttext();
			 gamestarted=1;
			 anybuttonpressed=0;
			 menustarted=0;
		}

		 if((greyButtonPressed==1)&&(menustarted==1)&&(gamestarted==0)){
			 display_clear();

			  PORTB = 0b11000100;
			  _delay_ms(20);
			  display_cmd();
			  _delay_ms(10);
			  write_string("Normal mode");

			 starttext();
			 gamestarted=1;
			  anybuttonpressed=0;
			  menustarted=0;
```

```
 }

 if(irsensor==1){
            points++;
            irsensor=0;
            pointhit=1;
 }


measured_time=count;
time_in_seconds=measured_time/61.2745098 ;

if(gamestarted==1){
  if(total_time_in_seconds<30){                                    //Slutar skriva ut efter 30 s.
  if(time_in_seconds>=1){                                          //Vad som görs varje sekund
            if(arcademode==1){
                      if((points%3==0) && (pointhit==1)){
                                total_time_in_seconds-=3;
                                PORTB = 0b10010100;                          // Byt rad till rad 3
                                _delay_ms(20);
                                display_cmd();
                                _delay_ms(10);
                                write_string("Extra time!");
                                _delay_ms(10000);
                                _delay_ms(10000);
                                pointhit=0;
                      }
            }

            display_clear();
            char str[5];
            sprintf(str, "%d", points);                  //Int till char
            write_string("Points: ");
            write_string(str);

            time_in_seconds=0.0;
            count=0.0;
            total_time_in_seconds ++;
            char time_str[2];
            dtostrf(total_time_in_seconds, 2, 0, time_str); // Skapar char-array av double (3e parametern
                                                            //anger antal decimaler)
            write_string(" Time: ");
            write_string(time_str);
           }
  }
}


  //Verkar onödigt att ha display_clear i egen metod, men det funkar inte annars.
  if(total_time_in_seconds>=30 && boolint3==0){
```

```
		display_clear();
		boolint3=1;
		boolint2=1;
}

if(boolint2==1 && boolint==0){
		display_clear();
		if((arcademode==1)&&(points>arcadehighscore)){

					arcadehighscore=points;

					for(int i=0;i<7;i++){
					 PORTB = 0b11000000;				// Byt rad till rad 2
					 _delay_ms(20);
					 display_cmd();
					 _delay_ms(10);

					 write_string("    HIGHSCORE!");
					 _delay_ms(2700);
					 display_clear();
					 _delay_ms(2000);
					}
					display_clear();
			}

			 if((arcademode==0)&&(points>normalhighscore)){

					 normalhighscore=points;

					 for(int i=0;i<7;i++){
							PORTB = 0b11000000;			// Byt rad till rad 2
							_delay_ms(20);
							display_cmd();
							_delay_ms(10);

							write_string("    HIGHSCORE!");
							_delay_ms(2700);
							display_clear();
							_delay_ms(2000);
					 }
					 display_clear();
			 }

		char str[5];
		sprintf(str, "%d", points);			//Int till char
		write_string("  TOTAL POINTS: ");
		write_string(str);

		// Byt rad till rad 2
		PORTB = 0b11000000;
```

```
            _delay_ms(20);
            display_cmd();
            _delay_ms(10);
            write_string("    Highscore: ");

            if(arcademode==1){
                        char str[3];
               sprintf(str, "%d", arcadehighscore);              //Int till char
               write_string(str);
            }

            if(arcademode==0){
                        char str[3];
                        sprintf(str, "%d", normalhighscore);                //Int till char
                        write_string(str);
            }


            // Byt rad till rad 3
            PORTB = 0b10010110;
            _delay_ms(20);
            display_cmd();
            _delay_ms(10);
            write_string("Press any button");

            //byt till rad 4
            PORTB = 0b11011001;
            _delay_ms(20);
            display_cmd();
            _delay_ms(10);
            write_string("to restart");
            anybuttonpressed=0;
            boolint=1;
}

if((boolint==1)&&(anybuttonpressed==1)){
            greyButtonPressed=0;
            yellowButtonPressed=0;
            points=0;
            count=0;
            total_time_in_seconds=0.0;
            measured_time=0.0;
            time_in_seconds=0.0;
            boolint=0;
            boolint2=0;
            boolint3=0;
            arcademode=0;
            pointhit=0;
            menustarted=0;
            gamestarted=0;
```

```
                    arcademode=0;
                }
            }
        }
```