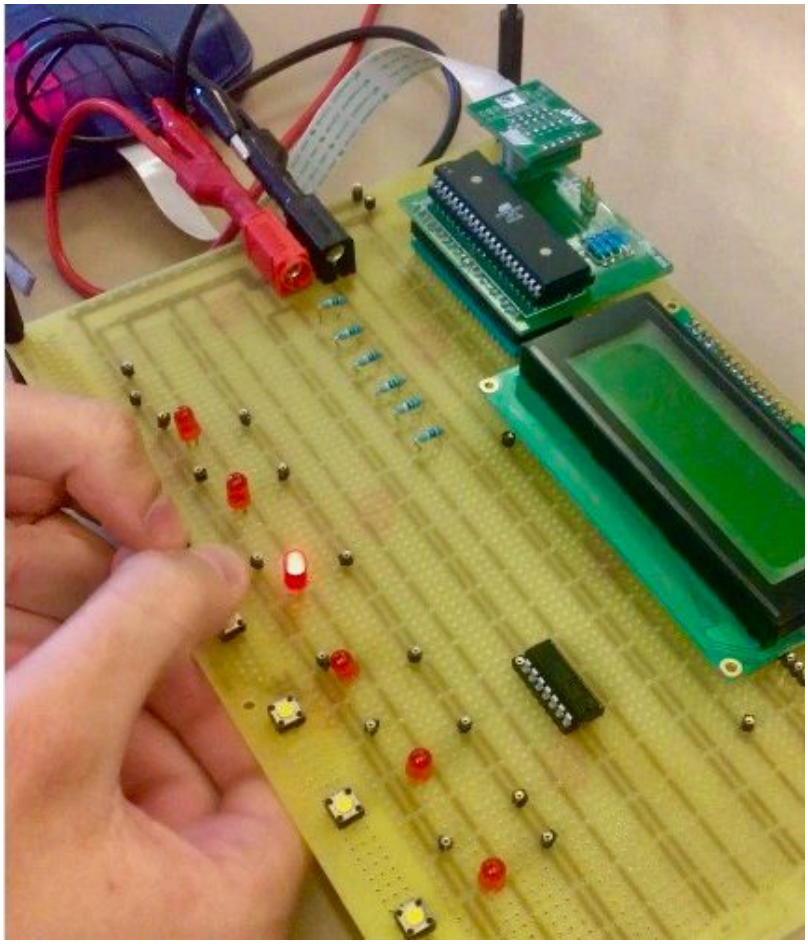
Digitala Projekt EITF11
Projektarbete
Handledare: Bertil Lindvall
2018-05-19

Grupp 9
Hanna Bondesson
Albin Schurmann
Marcus Skog

Reaktionsspel

EITF11



Abstract

This report is an outcome of the Digital Projects course at the faculty of engineering at LTH. In the report we describe how one can build a reaction game using six leds and appurtenant buttons. The main focus of the course has been to understand the integration between hardware and software and this has been accomplished iteratively as the project proceeded. The game is based on the concept of *whac-a-mole* and offers both one player mode and two player mode. The purpose of the report is both to share the making of the prototype but also make it possible for a reader to continue evolving the project.

1. Introduktion

1.1 Bakgrund

Denna rapport är resultatet från ett projektarbete i kursen Digitala Projekt på LTH. Rapporten beskriver framtagningen av ett reaktionsspel och är baserat på konceptet whac-a-mole. Prototypen består i detta fall av sex lysdioder med tillhörande knappar, samt en display som visar poäng och ger spelaren möjlighet att välja mellan två spellägen; en spelare eller två spelare.

1.2 Syfte

Syftet med kursen och därigenom projektet är att skapa förståelse för hur en konstruktionsprocess kan se ut och hur integrationen mellan hård- och mjukvara fungerar. Därutöver ska det ge insikt i flera av produktframtagningens processer, så som ritandet av kopplingsschema, framställningen av en kravspecifikation, lödning, användningen av datablad samt kodning i C. Rapporten syftar också till att möjliggöra för en läsare att vidareutveckla prototypen.

2. Kravspecifikation

2.1 Övergripande

- ❏ Spelet har sex LED-lampor och sex tillhörande tryckknappar samt en display som visar poäng.
- ❏ Vid spelets början kan spelaren välja *one player mode*, *two player mode* eller *high score* genom att trycka på den första, andra respektive tredje knappen från vänster.
- ❏ På high score visas de tre bästa (kortaste) tiderna för one player mode som spelats sedan spelet startades.

2.2 One player mode

- ❏ När spelet startas ska slumpmässigt en lysdiod åt gången lysa och spelaren ska då trycka på tillhörande knapp för att släcka den och få nästa att lysa.
- ❏ Under en omgång ska 10 lysdioder lysa.
- ❏ När en omgång är slut skall total tid visas på displayen (i ms)
- ❏ Spelet ska gå tillbaka till menyn när spelaren trycker på valfri knapp.
- ❏ I det fall att spelaren får den bästa tiden (kortast) sedan spelet startats ska displayen visa “nytt high score” innan den totala tiden visas och high score listan ska uppdateras.

2.3 Two player mode

- ❏ När spelet startas ska slumpmässigt en lysdiod hos vardera spelare lysa åt gången.
- ❏ Lysdioderna som lyser är placerade på samma ställe i förhållande till sin spelare (d.v.s. lysdiod 1 och 4, lysdiod 2 och 5 eller lysdiod 3 och 6 lyser samtidigt).
- ❏ Den spelare som först trycker på rätt knapp ska få poäng och först då lyser nästa två lampor.
- ❏ Under en omgång ska 10 dueller ske.
- ❏ När en omgång är slut ska båda spelarnas resultat visas på displayen.
- ❏ Spelet ska gå tillbaka till menyn när spelaren trycker på valfri knapp.

3. Hårdvara

Processor	ATmega16, en 8-bitars microcontroller med 40 pinnar. Hur komponenterna är kopplade till processorn kan ses i kopplingsschemat i bilaga 1.
Display	LCD display, 4 raders alfanumerisk display av typen Sharp Dot Matrix, BC2004A-serien. För att styra kontrasten på displayen har en potentiometer använts.
Lysdioder	6 stycken, lyser slumpartat en i taget för att visa spelaren vilken knapp denne skall trycka på för att få poäng.
Resistorer	6 stycken à 400 ohm, används för att begränsa strömmen till lysdioderna.
Knappar	6 stycken, en knapptryckning registreras genom att spänningen är 0V när den är nedtryckt och 5V annars.
Logisk grind	CMOS logisk 8-input NAND grind, för att styra avbrott från knapparna. Denna kopplas till PORTD (INT1). Vanligtvis är knapparna satta till ettor, men så fort en trycks ned är det alltså
JTAG	Används för att föra över mjukvaran till processorn.

Tabell 1: hårdvara

4. Mjukvara

Programmet skrevs i AtmelStudio med språket C och för att testa kod och hårdvara användes JTAG. I bilaga 2 återfinns koden för spelet.

För att få lysdioderna att lysa på ett slumpartat sätt har processorns interna 8 bitars timer använts. Attributet *count* ökar varje gång ett overflow från timern genererats. När metoden *randomLight()* anropas, som ska få en av lamporna att lysa, används modulo-operationen för att hitta resten vid division med sex (ty sex lysdioder) och denna avgör vilken lysdiod som kommer att lysa. Prescalern sattes till 64 efter testning. För two player mode skiljer sig algoritmen något.

Registreringen från knapptryckningarna drivs av externa avbrott. Knapparna är kopplade via en logisk grind (NAND) och därefter till samma processorpinne för externa avbrott. Dessutom är de kopplade till var sin pinne på C-bussen. När en knapp trycks ned registreras detta alltså direkt och därefter måste signalerna på C-bussen avläsas för att fastställa vilken knapp det var som gav upphov till avbrottet.

För att registrera tiden användes också processorns egna klocka. Prescalerna sattes då till 1024, högsta möjliga värde, vilket gör att antalet uppräkningar blir 7812.5 per sekund (8 000 000/1024). 8 bitars-timern kan räkna upp till 255 vilket gör att det genereras 30.63 overflows per sekund. Koden skrevs därför så att poängattributet *pointCount* räknas ut genom att dividera antalet overflows, under tiden då en lampa lyst tills dess att rätt knapp trycks ned, med $30.63*10^{-3}$ för att få antalet millisekunder.

Vid användning av displayen används en metod där ett 8-bitars stycke skickas med som argument. I metoden ändras A-portarna enligt argumentet och RS, RW samt E sätts till sina värden i rätt ordning för att displayen ska utföra det önskade. Denna metod underlättar både setup av displayen och de handlingar man vill utföra, exempelvis att rensa displayen, då RS, RW samt E inte behöver ställas in varje gång något ska utföras. Displayen har fyra rader och plats för 20 tecken på varje rad. Vid användning av en kodad skrivmetod kan man skriva exakt vad som ska synas på displayen genom att skriva in önskade tecken som argument.

5. Utförande

5.1 Planering

De första veckorna gick åt att bestämma typen av projekt, alltså någon form av reaktionsspel. Därefter togs en första kravspecifikation fram, att använda som utgångspunkt då kopplingsschemat skulle ritas. Under tiden fördes en dialog med handledare för att se vilken hårdvara som fanns tillgänglig och hur den kunde användas.

När kopplingsschemat godkänts av handledare påbörjades ihopsättningen som efter hand fått uppdateras, då nya idéer tillkommit. Grinden är ett sådant exempel och vi insåg snabbt att vår första tanke om att få hårdvaran klar först inte skulle vara möjlig, utan att vi behövde arbeta med programmeringen och framställandet av hårdvaran iterativt. Programmeringen i C påbörjades så fort den första prototypen var gjord och testningen skedde med hjälp av JTAG. Kravspecifikationen uppdaterades efter hand.

5.2 Konstruktion

Konstruktionen syns i följande bilder. Den sammansattes efter kopplingsschemat i bilaga 1.

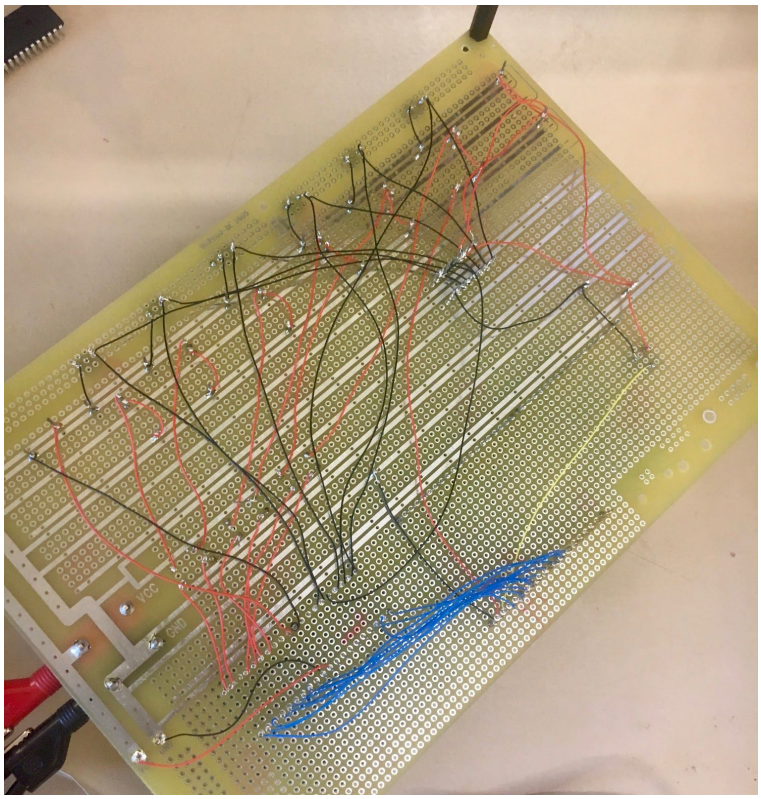


Bild 1: Färdig konstruktion underifrån

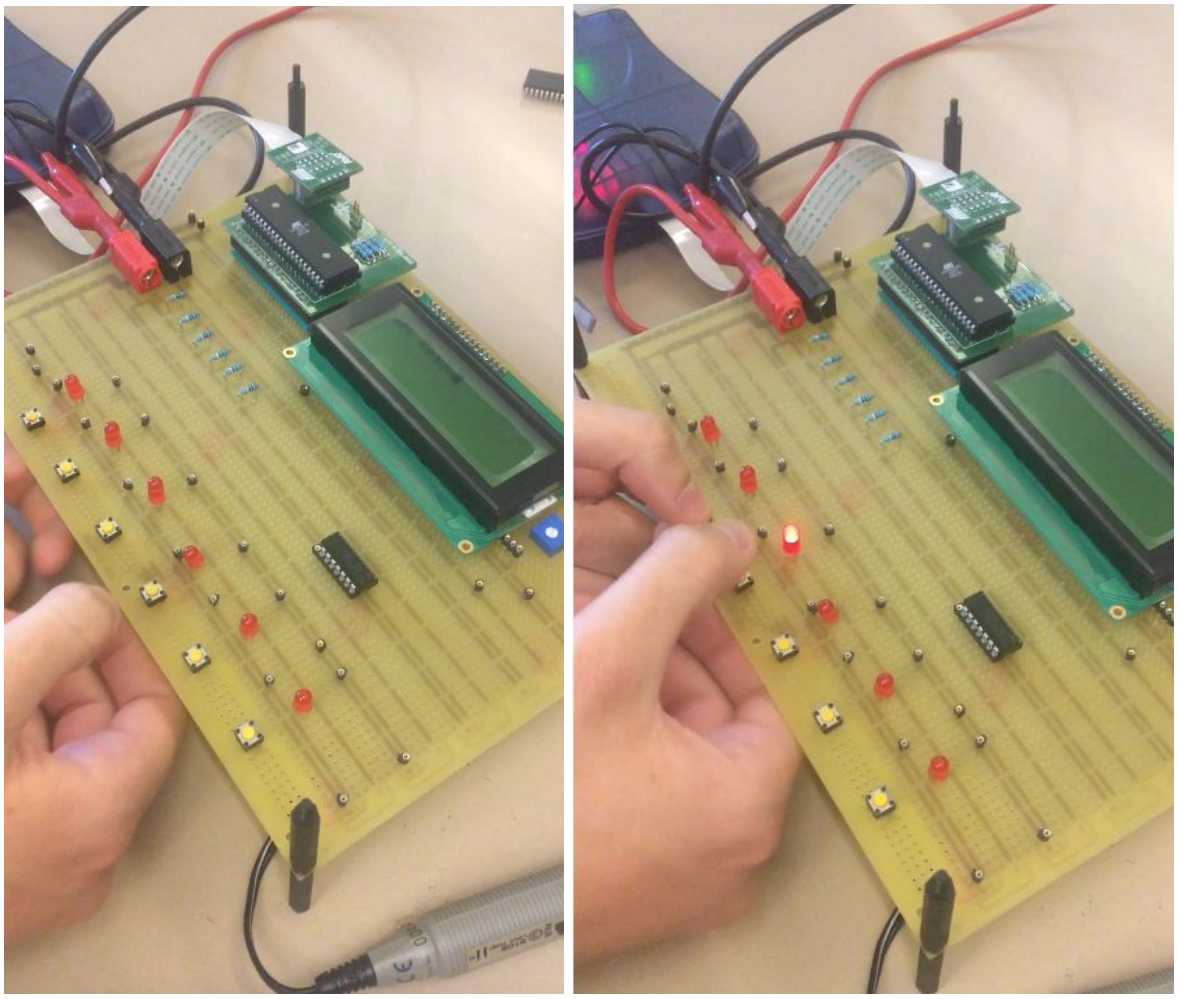


Bild 2 och 3: Färdig konstruktion ovanifrån

5.3 Programmering

Först när hårdvaran testats i AtmelStudio påbörjades programmeringen. Arbetsprocessen delades i princip upp i följande steg:

- Få lamporna att lysa på ett slumpartat vis
- Få knapparna att generera avbrott på rätt sätt (enligt kravspecifikationen)
- Kunna skriva ut text på displayen
- Få timern att fungera

Programspråket C användes och resultatet återfinns i bilaga 2.

6. Resultat

En prototyp av ett reaktionsspel som uppfyller kravspecifikationen i avsnitt 2. Spelet har bland annat två spellägen; one player mode och two player mode. Se bilder nedan.

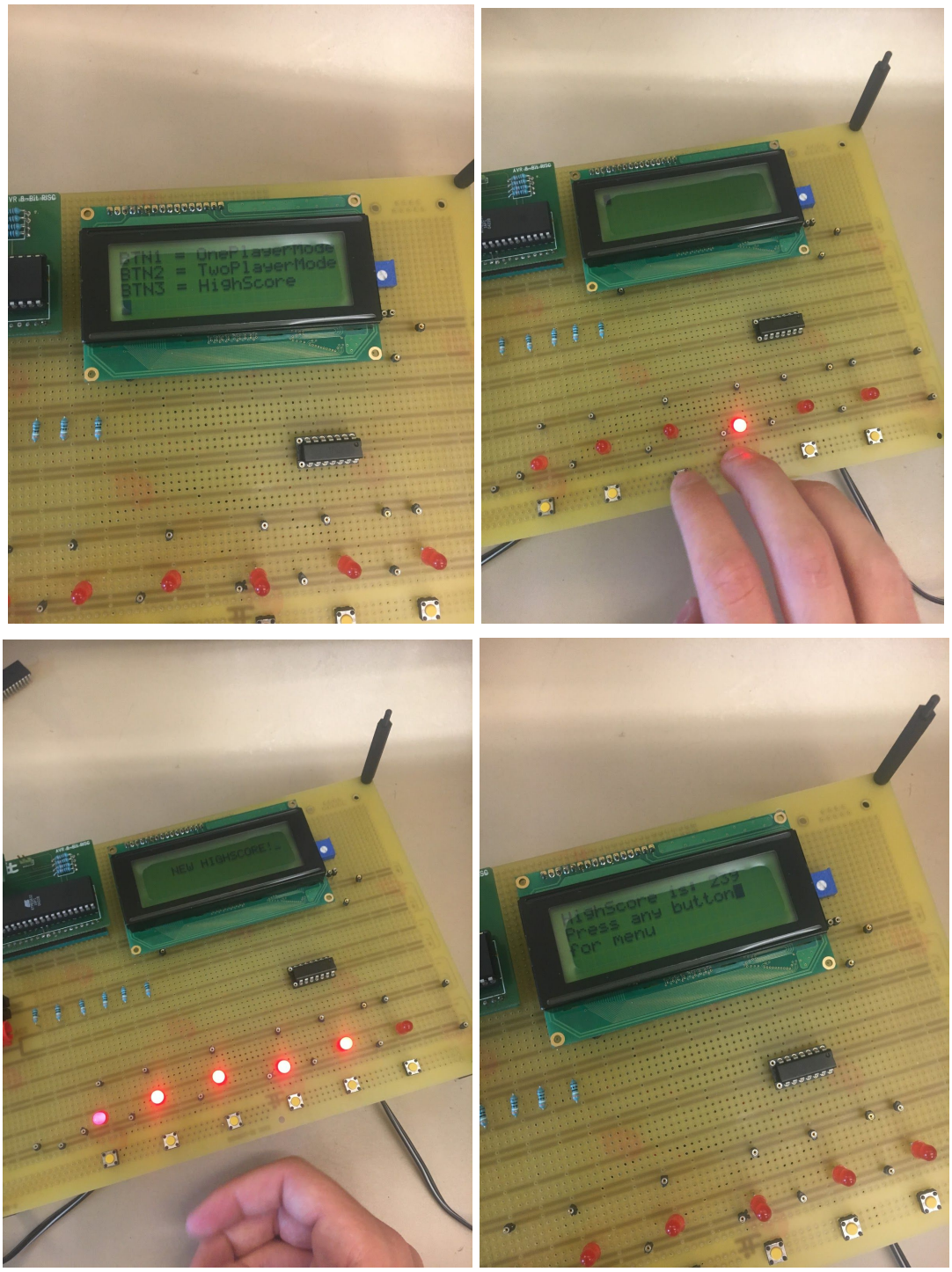


Bild 4-7: Olika spellägen

7. Diskussion

Inför projektet hade vi begränsade kunskaper inom ellära, elektronik och programmering i C och bestämde oss därför för att begränsa oss och göra ett spel där fokus skulle ligga på mjukvaran. Det är således den del som utgjort den stora delen av arbetet. Efter hand som vi blev mer bekväma med både hanteringen av elektroniken och C kunde vi utforma och utvidga projektet.

Till följd av detta har det mest utmanande hela tiden varit att komma igång med de olika delarna i arbetsprocessen.

För hårdvaran gällde det hur komponenterna skulle kopplas samman men också att sätta sig in i hur processorn fungerar och hur databladet skulle användas. I efterhand kan vi se att det hade funnits bättre sätt att koppla samman konstruktionen rent hårdvarumässigt. Som exempel på detta hade det varit mer logiskt att koppla knapparna till VCC och inte GND, så att deras utgångsläge varit 0V och att de blivit ettor (5V) då de tryckts ned. Vad gäller hantverket vid ihopsättningen; lödning och att fästa sladdarna, gick det relativt snabbt att förstå hur det bäst skulle göras.

För programmeringen i C var det själva konfigureringen av alla komponenter och utgångar som upptog det mesta av tiden, samt att förstå sig på hur databladet skulle läsas när kodningen skedde. Efter ett tag fick vi ordning på det och då blev det enklare att lägga till nya funktioner på prototypen.

8. Referenser

Datablad för processorn:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf

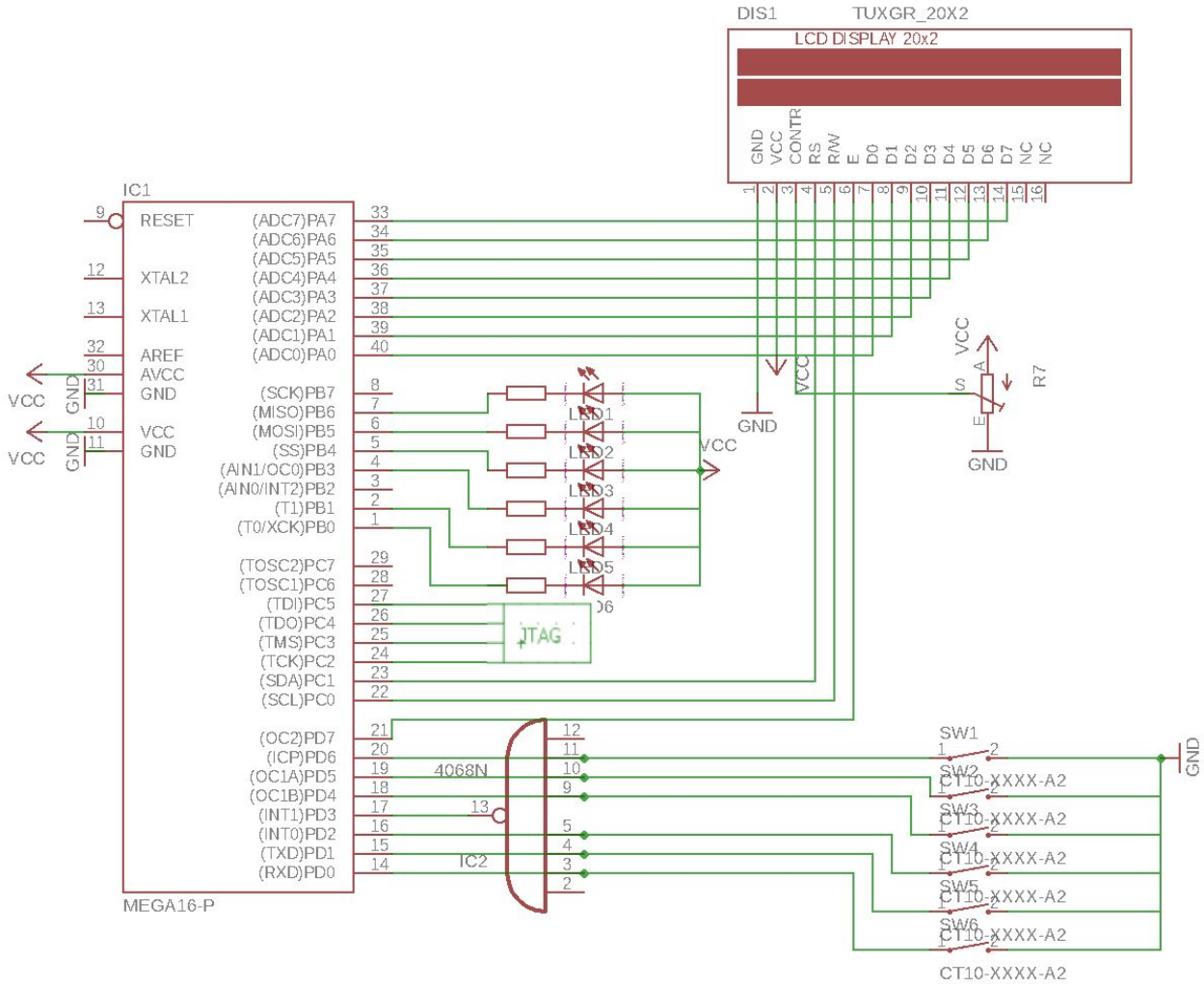
Datablad för grinden:
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Logik/74HC/74HC30.pdf

Datablad för displayen:
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf

Grundläggande om programmeringsspråket C:
http://www.eit.lth.se/fileadmin/eit/courses/eitf11/lecture/Programmera_i_C.pdf

9. Bilagor

Bilaga 1: Kopplingsschema för hårdvaran



Figur 1: Kopplingsschema för hårdvaran. Ritades i programmet Eagle.

Bilaga 2: Användarmanual

Spelets innehåll

- ❏ Spelet har sex LED-lampor och sex tillhörande tryckknappar samt en display som visar poäng.
- ❏ Vid spelets början kan spelaren välja *one player mode*, *two player mode* eller *high score* genom att trycka på den första, andra respektive tredje knappen från vänster.
- ❏ På high score visas de tre bästa (kortaste) tiderna för one player mode som spelats sedan spelet startades.

One player mode

- ❏ När spelet startas lyser 10 lysdioder, en åt gången, och spelaren ska då trycka på tillhörande knapp för att släcka den och få nästa att lysa.
- ❏ När en omgång är slut visas total tid på displayen (i ms) och spelet går tillbaka till menyn först när spelaren trycker på valfri knapp.

Two player mode

De tre knapparna till vänster tillhör spelare ett och de till höger spelare två.

- ❏ När spelet startas ska slumpmässigt en lysdiod hos vardera spelare lysa åt gången.
- ❏ Den spelare som först trycker på rätt knapp får poäng och först då lyser nästa två lampor.
- ❏ En omgång är 10 dueller lång.
- ❏ När en omgång är slut visas båda spelarnas resultat på displayen och spelet går tillbaka till menyn först när spelaren trycker på valfri knapp.

Bilaga 3: Kod

```
/*
* MoleGame123.c
*
* Created: 2018-05-14 09:55:44
* Author : ine15asc
*/

#include <avr/io.h>

#include <avr/io.h>


#define F_CPU           8000000UL

#define BTN_1           0b01111101
#define BTN_2                   0b01111110
#define BTN_3                   0b01110111
#define BTN_4                   0b01101111
#define BTN_5                   0b01011111
#define BTN_6                   0b00111111

#define BT_1            0b11111101
#define BT_2            0b11111110
#define BT_3            0b11110111
#define BT_4            0b11101111
#define BT_5            0b11011111
#define BT_6            0b10111111

#define LIGHT_1     0b00000101
#define LIGHT_2     0b00000110
#define LIGHT_3     0b01000100
#define LIGHT_4     0b00001100
#define LIGHT_5     0b00010100
#define LIGHT_6     0b00100100

int highScoreNumber = 10000;
int lightOn = 0;
```

```
int lightTurnedOn = 1;
int rightButton = 0;
int score = 0;
int gameround = 0;
int buttonPressed = 0;
int currentTime = 0;
int pointCount = 0;
int endTime = 0;
int initiateGame = 1;
int playerOne = 0;
int playerTwo = 0;
int playerOnePoints = 0;
int playerTwoPoints = 0;
int gameMode = 0;
int a = 0;

volatile double count;
volatile double countClock;


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


void setup_interrupt(void);
void setUpPorts(void);
void initTimers(void);


void randomLight(void);
void randomNumber(void);
void lightOff(void);
void allLightOn(void);
```

```
void display_setup(void);
void writeChar(char c);
void display_cmd(char c);
void writeString(char string[]);
void firstMessage();
void initiateGameMessage();
void clearDisplay();
void writePoints();
void twoPlayerRandomLight();
void twoPlayerDisplayScore();
void twoPlayerMode();
void twoPlayerTwoWritePoints();
void twoPlayerOneWritePoints();
void displayMenu();
void celebrateLight();
void highScore();
void checkButton();
void checkButtonTwo();

// **********************Port-setup*******************
void setUpPorts(){

	DDRB = 0b11111011;		// Sätter lamporna utåt
	lightOff();				//(pga vi kopplade tvärtom)
	DDRD = 0b10000000;		// Sätter knapparna inåt och interrupten inåt (display, pinne
21 utåt)
	PORTB = 0xff;
	PORTD = 0b11111011;
	initTimers();			//ställer in timers
	setup_interrupt();		//ställer upp interrupt
	display_setup();

}

// **********************Display*******************
void display_setup(){
	DDRA = 0b11111111; // Sätter pinnarna för port A till ettor (ut)
	DDRC  |= _BV(PC0);
	DDRC  |= _BV(PD1); //sätter pinne 22-23 till ettor (D- är redan satt)
	_delay_ms(10);
```

```
	PORTC &= ~_BV(PC0);
	PORTC &= ~_BV(PC1); //RS 0 & RW 0

	// function set
	//PORTA= 0b00111111; // Sätter interface datalängd, antal linjer etc.
	display_cmd(0b00111111);
	_delay_ms(10);

	//Display on
	//PORTA = 0b00001111; // Inställningar för displayen
	display_cmd(0b00001111);
	_delay_ms(10);

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

// **********************Timers*********************
void initTimers(){
	TCCR0 = TCCR0 | 0b00000011; // Prescaler 64 (Sida 85 i databladet.)
	TCCR2 = TCCR2 | 0b00000111; // Prescaler 1024. Sida 131 i databladet.
	TIMSK = TIMSK | 0b01000001; // Overflow interrupt aktivt. (Sida 83 i databladet.)
}

// **********************Interrupt*********************
void setup_interrupt(){
	GICR   |= (1<<INT0);
	MCUCR  |= (1<<ISC01) | (1<<ISC00);
	sei();
}


// **********************Lysdiod-kommandon*********************
void lightOff(){		//stänger av alla lampor
	PORTB |= 0b11111111;
	lightOn=0;

}
```

```
void allLightOn(){                          //sätter på alla lampor
        PORTB &= ~(0b11111111);
}


void randomNumber(){
        lightTurnedOn= (int) count%6;
}

void randomLight(){
        randomNumber();
        switch(lightTurnedOn){
                case 0: PORTB &= ~LIGHT_1 ; lightOn=1; break;
                case 1: PORTB &= ~LIGHT_2 ; lightOn=2; break;
                case 2: PORTB &= ~LIGHT_3 ; lightOn=3; break;
                case 3: PORTB &= ~LIGHT_4 ; lightOn=4; break;
                case 4: PORTB &= ~LIGHT_5 ; lightOn=5; break;
                case 5: PORTB &= ~LIGHT_6 ; lightOn=6; break;
        }
}

void twoPlayerRandomNumber(){
        lightTurnedOn= (int) count%3;
}

void twoPlayerRandomLight(){
        twoPlayerRandomNumber();
        switch(lightTurnedOn){
                case 0: PORTB &= ~LIGHT_1 ; lightOn=1; PORTB &= ~LIGHT_4 ;   break;
                case 1: PORTB &= ~LIGHT_2 ; lightOn=2; PORTB &= ~LIGHT_5 ;  break;
                case 2: PORTB &= ~LIGHT_3 ; lightOn=3; PORTB &= ~LIGHT_6 ;  break;
        }
}

void celebrateLight(){
        for( a = 0; a < 5; a = a + 1 ){
                PORTB &= ~LIGHT_1;
                _delay_ms(50);
                PORTB &= ~LIGHT_2;
```

```
                _delay_ms(50);
                PORTB &= ~LIGHT_3;
                _delay_ms(50);
                PORTB &= ~LIGHT_4;
                _delay_ms(50);
                PORTB &= ~LIGHT_5;
                _delay_ms(50);
                PORTB &= ~LIGHT_6;
                _delay_ms(50);
                lightOff();
        }
}



// *******Knapparna och interrupten****************

ISR(INT0_vect){
        buttonPressed = 1;
        _delay_ms(20);

        if(initiateGame == 1){
                if(PIND == BTN_1 || PIND == BT_1){
                        gameMode = 1;
                }
                if(PIND == BTN_2 || PIND == BT_2){
                        gameMode = 2;
                }
                if(PIND == BTN_3 || PIND == BT_3){
                        gameMode =3;
                }


        } else{
                if(gameMode == 2){
                        checkButtonTwo();
                } else if(gameMode == 1) {
                        checkButton();
```

```
                } else{

                }
        }
}


void checkButton(){
        switch(lightOn){
                case 1 :
                if(PIND == BTN_1 || PIND == BT_1){
                        lightOn = 0;
                        lightOff();
                        endTime = countClock;
                        rightButton = 1;
                }
                break;
                case 2 :
                if(PIND == BTN_2 || PIND == BT_2){
                        lightOn = 0;
                        lightOff();
                        endTime = countClock;
                        rightButton = 1;
                }
                break;
                case 3 :
                if(PIND == BTN_3 || PIND == BT_3){
                        lightOn = 0;
                        lightOff();
                        endTime = countClock;
                        rightButton = 1;
                }
                break;
                case 4 :
                if(PIND == BTN_4 || PIND == BT_4){
                        lightOn = 0;
                        lightOff();
                        endTime = countClock;
                        rightButton = 1;
```

```
                }
                break;
                case 5 :
                if(PIND == BTN_5 || PIND == BT_5){
                        lightOn = 0;
                        lightOff();
                        endTime = countClock;
                        rightButton = 1;
                }
                break;
                case 6 :
                if(PIND == BTN_6 || PIND == BT_6){
                        lightOn = 0;
                        lightOff();
                        endTime = countClock;
                        rightButton = 1;
                }
                break;
        }
}

void checkButtonTwo(){
switch(lightOn){
        case 1 :
        if(PIND == BTN_1 || PIND == BT_1){
                lightOff();
                rightButton = 1;
                lightOn = 0;
                endTime = countClock;
                playerOne = 1;
                break;
        }
        if(PIND == BTN_4 || PIND == BT_4){
                lightOff();
                rightButton = 1;
                lightOn = 0;
                endTime = countClock;
                playerTwo = 1;
                break;
```

```
        }
        break;
        case 2 :
        if(PIND == BTN_2 || PIND == BT_2){
                lightOff();
                rightButton = 1;
                lightOn = 0;
                endTime = countClock;
                playerOne = 1;
                break;
        }
        if(PIND == BTN_5 || PIND == BT_5){
                lightOff();
                rightButton = 1;
                lightOn = 0;
                endTime = countClock;
                playerTwo = 1;
                break;
        }
        break;
        case 3 :
        if(PIND == BTN_3 || PIND == BT_3){
                lightOff();
                rightButton = 1;
                lightOn = 0;
                endTime = countClock;
                playerOne = 1;
                break;
        }
        if(PIND == BTN_6 || PIND == BT_6){
                lightOff();
                rightButton = 1;
                lightOn = 0;
                endTime = countClock;
                playerTwo = 1;
                break;
        }
        break;
}
```

```
}

void waitForButton(){
	buttonPressed = 0;
	while(buttonPressed == 0){
		writeString("");
	}
	_delay_ms(150);
	buttonPressed = 0;
}

// **********************Timers*********************

//Overflow Interrupt (för klockan)
ISR(TIMER0_OVF_vect){ // TIMER0_OVF_vect
	count++;
}

ISR(TIMER2_OVF_vect) { // TIMER0_OVF_vect
	countClock++;
}



// **********************Display-kommandon*********************
void display_cmd(char c){

	PORTD |= _BV(PD7); // RW, RS -> 0, E -> 1
	PORTC &= ~_BV(PC0); //RW
	PORTC &= ~_BV(PC1); //RS
	PORTA = c;   // skickar c
	PORTD &= ~_BV(PD7); // läser av c när E=0
	_delay_ms(10);
	PORTD |= _BV(PD7); // RW, RS -> 0, E -> 1
}

void writeChar(char c) {
	PORTA = c;
	PORTC |= _BV(PC1);
	PORTC &= ~_BV(PC0);
```

```
	PORTD |= _BV(PD7);
	PORTD &= ~_BV(PD7);  // sätter RS till 1 RW till 0
	_delay_us(50);
}

void writeString(char string[]){
	//display_cmd(0b00000000); // sätter cursorn i början när vi skriver
	int i = 0;
	while(string[i] != '\0'){ // "\0" betyder "null" i ASCII.
		writeChar(string[i]); //Skriver ut bokstaven.
		i++;
	}
}


void firstMessage(){
	writeString("Do you want to play ");
	writeString("Press any button    ");
	writeString("a game? ");
	_delay_ms(2);
}

void initiateGameMessage(){
		display_cmd(0b00000001);
		writeString("Get Ready");
		_delay_ms(1000);
		display_cmd(0b00000001); // clearar
		writeString("3");
		_delay_ms(1000);
		display_cmd(0b00000001);
		writeString("2");
		_delay_ms(1000);
		display_cmd(0b00000001);
		writeString("1");
		_delay_ms(1000);
		display_cmd(0b00000001);
		writeString("GO");
		_delay_ms(500);
		display_cmd(0b00000001);
}
```

```
void writePoints(){
	 char str[5];
sprintf(str, "%d", pointCount); //Int till char
writeString(str);
}

void clearDisplay(){
	display_cmd(0b00000001);
}

void displayScore(){
	writeString("Score: ");
	writePoints();
	writeString("	For Menu");
	writeString("	 Press Any Button");
	_delay_ms(1000);
	initiateGame = 1;
	pointCount = 0;
	gameround = 0;
}

void twoPlayerOneWritePoints(){
	char stri[5];
	sprintf(stri, "%d", playerOnePoints);
	writeString(stri);
}

void twoPlayerTwoWritePoints(){
	char strin[5];
	sprintf(strin, "%d", playerTwoPoints);
	writeString(strin);
}

void twoPlayerDisplayScore(){
	writeString("Player1 Score: ");
	twoPlayerOneWritePoints();
	writeString("    Press any button");

	writeString("    Player2 Score: ");
```

```
        twoPlayerTwoWritePoints();
        _delay_ms(1000);
        buttonPressed = 0;
        gameround = 0;
        playerOnePoints = 0;
        playerTwoPoints = 0;
}

void highScore(){
        if (highScoreNumber == 10000){
                clearDisplay();
                writeString("No one has played   ");
                writeString("button for Menu     ");
                writeString("yet. Press Any");
                _delay_ms(1000);
                _delay_ms(1000);
        }
        else {
                clearDisplay();
                writeString("HighScore is: ");
        char string[5];
        sprintf(string, "%d", highScoreNumber);
        writeString(string);
        writeString("For Menu           ");
        writeString("Press Any Button");
        _delay_ms(1000);
        }
}

void displayMenu(){
        clearDisplay();
        writeString("BTN1 = OnePlayerMode");
        writeString("BTN3 = HighScore    ");
        writeString("BTN2 = TwoPlayerMode");
        _delay_ms(20);
}



// **********************Game*********************
```

```
void runGameRound(){
	while(gameround < 10) {
		randomLight();
		currentTime = countClock;
		rightButton = 0;
		while(rightButton == 0){
			writeString("");
		}
		rightButton = 0;
		pointCount = pointCount + (endTime - currentTime)/0.030637254902;
		gameround++;
		buttonPressed = 0;
		_delay_ms(10);
	}
	if(pointCount < highScoreNumber){
		highScoreNumber = pointCount;
		writeString("                ");
		writeString("                ");
		writeString("    NEW HIGHSCORE!");
		celebrateLight();
		clearDisplay();
	}
	gameMode = 0;
	gameround=0;
}


void twoPlayerMode(){
	while(gameround < 10){
		twoPlayerRandomLight();
		while(rightButton == 0){
			writeString("");
		}
		rightButton=0;
		if(playerOne == 1){
			playerOnePoints++;
		}
		if(playerTwo == 1){
			playerTwoPoints++;
```

```
                }
                playerOne = 0;
                playerTwo = 0;
                rightButton = 0;
                gameround++;
                buttonPressed = 0;
                _delay_ms(100);
        }
        gameMode = 0;
        gameround = 0;

}


// **********************Main********************
int main (void) {
        setUpPorts();
        clearDisplay();

        firstMessage();
        waitForButton();
        displayMenu();
        gameMode = 0;
        while(gameMode == 0){
                waitForButton();
        }


        while(1){
                initiateGame = 0;
                if(gameMode == 1){
                        count = 0;
                        countClock = 0;
                        initiateGameMessage();
                        runGameRound();
                        displayScore();
                }
                if(gameMode == 2){
                        count = 0;
```

```
                countClock = 0;
                initiateGameMessage();
                twoPlayerMode();
                twoPlayerDisplayScore();
            }
            if(gameMode==3){
                highScore();
            }
            waitForButton();
            clearDisplay();
            displayMenu();
            gameMode=0;
            initiateGame=1;
            while(gameMode == 0){
                waitForButton();
            }
            clearDisplay();


        }
}
```