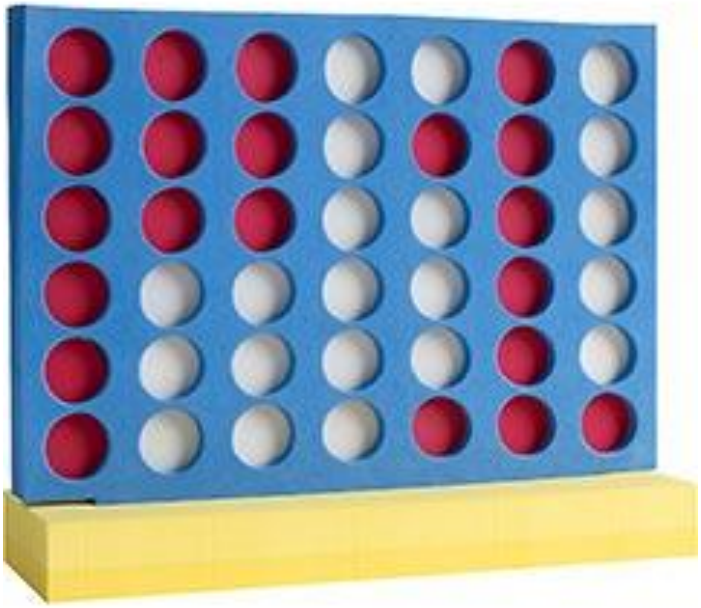
Digitala projekt EITF11

Fyra i rad

Handledare: Bertil Lindvall



Vilhelm Sjölund I-11, Henrik Melin I-10
5/19/2014

Sammanfattning

Denna rapport behandlar ett arbete inom kursen Digitala Projekt, EITF11. I kursen ingår ett projekt där en prototyp av något slag ska byggas och programmeras. Här har en digital version av fyra i rad byggts och programmerats, och nedan presenteras först hur arbetet planerades, sedan hur själva konstruktionen gick till och till sist färdigställandet med programmering.

Innehållsförteckning

Inledning

Under vårterminen 2014 har ett projekt i kursen Digitala projekt genomförts inom teknikprofilen System- och programvaruutveckling. Syftet med kursen är att ge grundläggande förståelse för C-programmering direkt mot hårdvara, grundläggande elektronik samt ge en försmak på hur problemlösning inom elektrisk konstruktion går till. Kursen innefattar nästan uteslutande eget arbete och ger möjlighet att disponera tiden som man själv önskar. Mycket fokus ligger på att på egen hand strukturera upp projektet och se till att man klarar av att hålla den deadline som fastställts.

Från början var planen att bygga ett fyra i rad utan att använda en skärm utan enbart med lysdioder. En hastig överslagsräkning visade orimligheten i detta utförande och ledde till att en LCD-skärm åter blev aktuellt. Kravspecifikationen fick korrigeras för att möta de nya kraven på utrustningen.

Funktionella krav lysdioder (ej del av utförandet)

1. När man trycker på en knapp ska bricka i den färg vars tur det för tillfället är släppas ned i kolumnen som knappen avser och hamna ovanpå eventuellt tidigare lagda brickor.
2. Om en kolumn är full ska inget hända om man trycker på kolumnens knapp.
3. Om en användare får fyra i rad ska spelet meddela att den färgen har vunnit.
4. Om alla rutor fylls utan att någon har vunnit ska spelet meddela detta.
5. Om någon trycker på omstarts-knappen startas spelet om.

Reviderade funktionella krav

1. En LCD-skärm ska tillhandahålla den grafiska representationen av spelet. Det ska tydligt gå att skilja på de två olika spelarnas typer av brickor, och en pil eller liknande indikator ska markera var någonstans man tänker lägga en ny bricka
2. Spelet ska utrustas med fyra knappar. Två stycken för att styra markören i vänster respektive höger riktning, en knapp för att lägga en bricka samt en knapp för att starta om spelet.

Hårdvara

Processor

ATMega 16 microcontroller med 40 pin. Processorn har programmerbart flashminne med 16 kb vilket möjliggör lagring av vårt program.

Skärm

GDM12864HLCM – En bakgrundbelyst LCD-skärm med 64x128 pixlar. Displayen består av två stycken separata 64 x 64 skärmar som tillsammans utgör hela displayen.

Knappar

Fyra stycken knappar används. Knappen är egentligen en sorts strömbrytare som är låg när knappen inte är intryckt och hög när knappen är intryckt.

Resistorer
Sex stycken resistorer används totalt. Fyra stycken används till knapparna för att begränsa strömtillförseln när de är nedtryckta, och de sista två är kopplade till lysdioderna för att de inte ska utsättas för alltför hög spänning. Även en resistor med reglerbart motstånd användes för att öka eller minska skärmens ljusstyrka.

Dioder
Två stycken dioder används för att visa vilken spelares tur det är. De har färgerna röd och blå.

Mjukvara
Programmet skrivs uteslutande i C och för att felsöka och testa koden användes Jtag. Med hjälp av denna kunde varje rad kod prövas separat.

Utförande

Planering
Projektet inleddes läsperiod två i början av mars. Målsättningen var att innan påsk ha en fullt fungerande hårdvarukonstruktion som kunde testas med enkel kod. Efter påsken inleddes programmeringsfasen.

En kravspecifikation utformades över prototypen. Placeringen av hårdvarukomponenterna föll sig naturligt, och någon ingående analys över detta var inte nödvändig. Målet var att uppsättningen skulle falla sig intuitiv. För att uppnå detta ritades en enklare skiss upp över hur det skulle komma att se ut. Två knappar i höjd med varandra representerade höger och vänster navigering, och under dessa, mitt emellan, en knapp för att placera brickorna. Den sista knappen, som används för att starta om spelet, placerades på andra sidan plattan för att inte förvirring skulle uppstå.

Konstruktion
Först studerades databladen för de olika komponenterna och sedan ritades de planerade kopplingsschemana upp i programmet Powerlogic (se bilaga 1). Ritningen gjordes med stor noggrannhet för att undvika framtida komplikationer.

Själva bygget karaktäriserades av något som kan beskrivas som en ”bottom-up”-princip. Med detta menas att de icke avancerade komponenterna monterades först för att i tidigt stadie kunna testa och kontrollera att allting kopplats på rätt sätt. Konstruktionen inleddes således med montering av strömtillförsel, processor och de två lysdioderna. Knapparna monterades enligt den planering som fastställts, se föregående stycke. Slutligen monterades skärmen, vilken skulle visa sig svår att testa och kontrollera, se nedanstående.

Programmering

Tester och kontroll

Under konstruktionsfasen utfördes löpande testning för att kontrollera huruvida komponenterna kopplats på ett korrekt sätt eller inte. Testningen av lysdioderna utgjordes helt enkelt av ett mycket simpelt program som satte den pin de kopplats till som hög eller låg.

Eftersom knapparna, som tids nog skulle styra skärmen, monterades före skärmen i sig behövdes ett annat sätt att testa dessa på. Ett enklare program skrevs med funktionen att om en viss knapp trycks ned så ska en av lysdioderna börja lysa.

Slutligen utfördes tester på skärmen. Det skulle visa sig problematiskt då själva programmeringen av skärmen var relativt avancerad, och svårigheter uppstod med att identifiera om det var fel i den fysiska inkopplingen av skärmen eller om felet låg i själva testningen. När skärmens Write Timing exekverades korrekt visade det sig att skärmen var inkopplad som planerat. De båda skärmhalvorna testades var och en för sig, där enklare figurer ritades in i vardera.

Programmering av spelet

Inledningsvis skrevs de mer generella funktioner som skulle komma att behövas i ett senare skede, som till exempel Write och SetXY. Dessa skrevs för att kunna hantera båda skärmarna, men den ena halvan visade sig vara överflödig. Spelplanen för just detta spel är av relativt kvadratisk struktur vilket inte var förenligt med skärmens rektangulära struktur.

Spelmekanismen grundar sig i en matris som kan innehålla tre olika värden: 0, 1 eller 2. En nolla representerar en tom ruta, alltså där en bricka ännu ej lagts, medan 1 och 2 representerar de två olika spelarnas respektive brickor. Eftersom brickorna inte kan flyttas när de väl är lagda behövs ingen funktion som kontinuerligt letar igenom matrisen för att uppdatera spelplanen, utan då en bricka läggs ritas den ut på skärmen och läggs till i matrisen.

Den största utmaningen i programmeringen var algoritmen som kontrollerar om en spelare har vunnit. Denna skulle kunna skrivas på många olika sätt, och framför allt med olika grad av effektivitet, men de totala antalet platser att gå igenom var så pass få att tidsvinsten inte var värd besväret. Algoritmen kontrollerade istället de olika riktningar för att se om det någonstans låg fyra i rad, med riktningarna upp till ned, höger till vänster, diagonalt nedifrån vänster och upp till höger samt diagonalt nedifrån höger och upp till vänster.

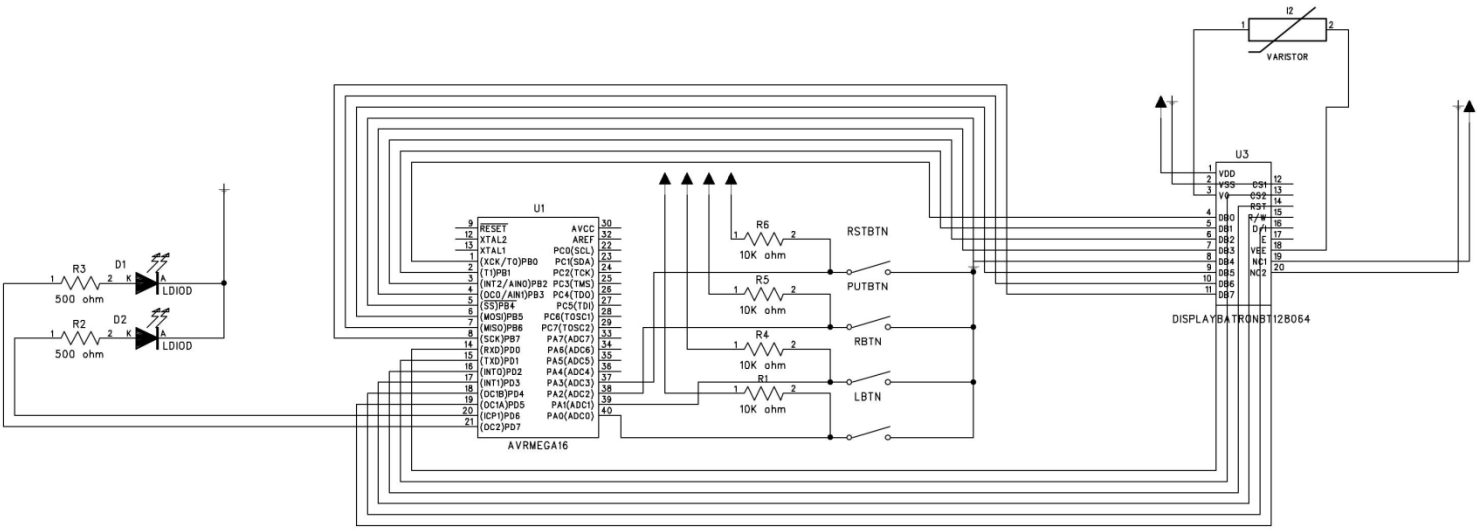
Diskussion

Sammanfattningsvis gick projektet någorlunda smärtfritt. Det hade varit önskvärt att ha haft lite godare förkunskaper inom elektronik, men det var inget som inte kunde plockas upp efterhand som kursen pågick. Att programmera i C var helt nytt men språket är både väldigt logiskt uppbyggt samt har mycket gemensam syntax med Java så detta visade sig vara ett av de minsta problemen.

Att bit för bit bygga upp något ger mycket god förståelse för hur elektroniska komponenter kommunicerar med varandra, och till skillnad från högnivå-programmering kan en byte där en etta förväxlats med en nolla spela stor roll för slutresultatet.

Appendix

Bilaga 1 – Kretsschema PowerLogic



Bilaga 2 – Källkod

```
#include <avr/io.h>
#include<stdio.h>
short int RIGHT;
short int LEFT;
short int PUT;
short int RESET;
#define RSHIGH 0x01
#define RWHIGH 0x08
#include "math.h"
int side;
int matrix[6][7];
#include <util/delay.h>
short int finished;
short int arrowCord;
short int player;
char init;

int main(void)
{
	while(1)
```

```
	{
		DDRA = 0x00;
		DDRB = 0xFF;
		DDRD = 0xFF;
		PORTD = 0x00; //Reset
		init = 0x77;
		createMatrix();
		finished = 0;
		arrowCord = 0;
		side = 1;
		turnOnLCD();
		clearScreen();
		drawArrow(0);
		player = 1;
		PORTD = 0x77;
		startGame();


		PORTB = 0x00;
	}
}

void createMatrix() {
	for (int i = 0; i < 6; i++) {
		for (int j = 0; j < 7; j++) {
			matrix[i][j] = 0;
		}
	}
}

int findPlace() {
	short int currentPlace = 0;
	if (matrix[currentPlace][arrowCord] != 0) {
		return -1;
	}
	while (matrix[currentPlace+1][arrowCord] == 0 && currentPlace < 5) {
		currentPlace++;
	}
	return currentPlace;
}

int checkWin(int x, int y) {
	int stopleft=0;
	int stopright=0;
	int count = 1;
	for (int i = 1; i < 7; i++) {
		if (matrix[x][y] == matrix[x+i][y] && x < 6 && stopright==0) {
			count++;
		} else {
			stopright=1;
		}
		if (matrix[x][y] == matrix[x-i][y] && x > 0 && stopleft==0) {
			count++;
		} else {
			stopleft=1;
		}
	}
	if (count>=4) {
```

```
			return 1;
		}
		count = 1;
		stopleft = 0;
		stopright = 0;
		for (int i = 1; i < 7; i++) {
			if (matrix[x][y] == matrix[x][y+i] && y < 7 && stopright==0) {
				count++;
			} else {
				stopright=1;
			}
			if (matrix[x][y] == matrix[x][y-i] && y > 0 && stopleft==0) {
				count++;
			} else {
				stopleft=1;
			}
		}
		if (count>=4) {
			return 1;
		}
		count = 1;
		stopleft = 0;
		stopright = 0;
		for (int i = 1; i < 7; i++) {
			if (matrix[x][y] == matrix[x+i][y+i] && x < 6 && y < 7 && stopright==0) {
				count++;
			} else {
				stopright=1;
			}
			if (matrix[x][y] == matrix[x-i][y-i] && x > 0 && y > 0 && stopleft==0) {
				count++;
			} else {
				stopleft=1;
			}
		}
		if (count>=4) {
			return 1;
		}
		count = 1;
		stopleft = 0;
		stopright = 0;
		for (int i = 1; i < 7; i++) {
			if (matrix[x][y] == matrix[x+i][y-i] && x < 6 && y > 0 && stopright==0) {
				count++;
			} else {
				stopright=1;
			}
			if (matrix[x][y] == matrix[x-i][y+i] && x > 0 && y < 7 && stopleft==0) {
				count++;
			} else {
				stopleft=1;
			}
		}
		if (count>=4) {
			return 1;
		}
		return 0;
}
```

```
void startGame() {
	while(finished != 1) {
		_delay_ms(800);
		readButton();
	}
}

void readButton() {
	if (((~PINA) & 0x08) == 0x08) {
		if (arrowCord < 6) {
			clearArrow(arrowCord);
			arrowCord++;
			drawArrow(arrowCord);
		}
		if (player == 1) {
			init = 0x77;
			PORTD = 0x77;
		} else {
			init = 0xB7;
			PORTD = 0xB7;
		}
	}

	if (((~PINA) & 0x02) == 0x02) {
		if (arrowCord > 0) {
			clearArrow(arrowCord);
			arrowCord--;
			drawArrow(arrowCord);
		}
		if (player == 1) {
			init = 0x77;
			PORTD = 0x77;
		} else {
			init = 0xB7;
			PORTD = 0xB7;
		}
	}
	if (((~PINA) & 0x04) == 0x04) {
		short int x = findPlace();
		if (x != -1) {
			if (player == 1) {
				drawCircle(x, arrowCord);
				matrix[x][arrowCord] = 1;
				player = 2;
				init = 0xB7;
				PORTD = 0xB7;
				drawArrow(arrowCord);
				if (checkWin(x,arrowCord) == 1) {
					finished = 1;
					clearScreen();
					drawArrow(0);
				}
				_delay_ms(1000);
			}
			else {
				drawEmptyCircle(x,arrowCord);
				matrix[x][arrowCord] = 2;
```

```
                        player = 1;
                        init = 0x77;
                        PORTD = 0x77;
                        drawArrow(arrowCord);
                        if (checkWin(x,arrowCord) == 1) {
                                finished = 1;
                                clearScreen();
                                drawArrow(0);
                        }
                        _delay_ms(1000);
                }
        }
    }
    if (((~PINA) & 0x01) == 0x01) {
        createMatrix();
        finished = 0;
        arrowCord = 0;
        side = 1;
        clearScreen();
        drawArrow(0);
        player = 1;
        init = 0x77;
        PORTD = 0x77;
    }
}

int statusRead() {
    _delay_ms(1);
}

void clearScreen() {
    for (int i = 0; i < 64; i++) {
        for (int j = 0; j < 8; j++) {
            setXY(j,i);
            PORTB = 0x00;
            write();
        }
    }
}

void draw(int x, int y, char data) {
    setXY(x,y);
    PORTB = data;
    write();
}

void drawArrow(int y) {
    y = 8*y;
    draw(0,y,0x00);
    draw(0,y+1,0x00);
    draw(0,y+2,0x00);
    draw(0,y+3,0xFC);
    draw(0,y+4,0xFC);
    draw(0,y+5,0x00);
    draw(0,y+6,0x00);
    draw(0,y+7,0x00);
    draw(1,y,0x00);
    draw(1,y+1,0x01);
```

```
	draw(1,y+2,0x03);
	draw(1,y+3,0x07);
	draw(1,y+4,0x07);
	draw(1,y+5,0x03);
	draw(1,y+6,0x01);
	draw(1,y+7,0x00);
	for (int i = 0; i < 8; i++) {
		draw(i,y,0xFF);
		draw(i,y+7,0xFF);
	}
}

void clearArrow(int y) {
	y = 8*y;
	for (int i = 0; i < 8; i++) {
		for (int j = 0; j < 2; j++) {
			draw(j, y+i,0x00);
			draw(i,y,0x00);
			draw(i,y+7,0x00);
		}
	}
}

void drawEmptyCircle(int x, int y) {
	y = 8*y;
	draw(x+2,y,0x00);
	draw(x+2,y+1,0x18);
	draw(x+2,y+2,0x24);
	draw(x+2,y+3,0x42);
	draw(x+2,y+4,0x42);
	draw(x+2,y+5,0x24);
	draw(x+2,y+6,0x18);
	draw(x+2,y+7,0x00);
}

void drawCircle(int x, int y) {
	y = 8*y;
	draw(x+2,y,0x00);
	draw(x+2,y+1,0x18);
	draw(x+2,y+2,0x3C);
	draw(x+2,y+3,0x7E);
	draw(x+2,y+4,0x7E);
	draw(x+2,y+5,0x3C);
	draw(x+2,y+6,0x18);
	draw(x+2,y+7,0x00);
}

void turnOnLCD() {
	PORTB = 0x3F;
	executeInstruction();
}

void executeInstruction() {
	statusRead();
	if (side == 0) {
		PORTD = init; //Init
		PORTD = 0x26; //Instruction
		PORTD = 0x06; //Execute
```

```
		PORTD = init; //Init
	} else {
		PORTD = init;
		PORTD = 0x25;
		PORTD = 0x05;
		PORTD = init;
	}

}

void write() {
	statusRead();
	if (side == 0) {
		PORTD = init; //Init
		PORTD = 0x36; //Data mode, Chip select 1
		PORTD = 0x16; //Execute
		PORTD = init; //Init
	}
	else {
		PORTD = init; //Init
		PORTD = 0x35; //Data mode, Chip select 2
		PORTD = 0x15; //Execute
		PORTD = init; //Init
	}
}

void setXY(int x, int y) {
	//Sätt x
	PORTB = 0xB8 + x;
	executeInstruction();
	if (y < 64) {
		//Sätt y
		PORTB = 0x40 + y;
		executeInstruction();
	}
	else if (y > 63) {
		//Sätt y
		y = y - 64;
		PORTB = 0x40 + y;
		executeInstruction();
	}
}
```