

Fyra i rad

Sammanfattning

Detta dokument behandlar ett projekt genomfört i kursen Digitala Projekt vid Lunds Tekniska högskola. Syftet med det projekt som avhandlas i denna rapport var att sammanställa en grundläggande digital prototyp och göra denna funktionsduglig genom att programmera tillhörande processor. Målet med detta projekt var att skapa ett fyra i rad – spel. I nedanstående rapport återfinns beskrivning av arbetsprocessen samt en diskussion av tillämpad metodik och erhållna resultat.

Innehåll

Inledning.....	3
Problemformulering.....	3
Kravspecifikation.....	3
Hårdvara.....	4
Mjukvara.....	4
Utförande.....	5
Planering.....	5
Konstruktion.....	5
Programmering.....	6
Resultat och diskussion.....	6
Appendix 1: Kopplingschema.....	7
Appendix 2: Källkod.....	8

Inledning

Det projekt som behandlas i denna rapport utgjorde huvudsaklig arbetsuppgift i kursen Digitala projekt EITF11. Kursen i fråga är uppdelad i två separata verksamhetsperioder fördelat på två läsperioder. Kursen inleddes med teoretisk undervisning representerad av laborationer och föreläsningar under läsperiod ett, varpå studenterna delades in projektgrupper med uppgiften att på egen hand, understödda av handledare, tillämpa förvärvade kunskaper på ett praktiskt problem i form av en digital prototyp av godtycklig art.

Den primära intention grupp 12 hade under utvecklingen av projektets grundläggande koncept var att skapa ett interaktivt spel vilket innefattade programmering av en LCD skärm. Flertalet potentiellt gångbara koncept övervägdes innan beslutet om att programmera ett fyra i rad spel togs.

Problemformulering

I syfte att göra spelet interaktivt togs beslutet att spelarna skulle styra respektive markör med en joystick istället för med ett flertal enklare knappar. Därutöver var prototypen i behov av en resetknapp för att uppnå önskad funktionalitet.

För att kunna genomföra projektet enligt det koncept som utarbetats initialt behövdes därtill ett antal komponenter fundamentala för konstruktion av en fungerande prototyp.

- En processor och en LCD-skärm var nödvändiga för att hantera hårdvaran samt för att kunna visualisera spelplanen.
- AD-omvandlare vilken möjliggör hantering av analog information från spelets joysticks via processorn
- En röd LED-lampa för visa när en spelare vunnit
- Kod för att skapa spelet

Kravspecifikation

De krav som ställts på konstruktionen, utifrån den initiala projekttiden, är följande:

Funktionella krav

- Systemet ska ha en display
- Displayen ska visa en spelplan och en poängräknare för var spelare
 - Spelplanen ska vara ett 4x4 rutnät
 - På displayen ska varje spelare styra en unik symbol
- Systemet ska ha två uppsättningar knappar, en vid var kortsida på displayen
 - Var spelare ska kunna flytta markören upp, ner, höger och vänster samt kunna fästa markören på önskad ruta
- Systemet ska ha en resetknapp
 - Vid knapptryck så ska spelet starta om från spelomgång ett
- Spelet ska köras tre omgångar innan det avslutas

- Fördröjningen mellan knapptryck och händelse på display ska vara så liten som möjligt
- Spelet ska lyda följande regler:
 - En spelomgång avbryts då en spelare har fyra symboler i rad horisontellt, vertikalt eller diagonalt. Poäng registreras och ny spelomgång startas
 - Spelet bryts efter tre spelomgångar
 - Ena lysdiod lyser när spelomgång tre avslutas
 - En redan ifylld ruta får inte skrivas över med en ny symbol
 - Markören får förflyttas över en ifylld ruta
 - Om hela brickan fylls utan att en spelare vunnit så bryts spelomgången och ingen får poäng

Hårdvara

För att konstruera prototypen användes följande hårdvarukomponenter:

- En AVR AT mega32 microprocessor användes för att lagra information och styra övriga komponenter
- LCD-skärmen GDM12864HLCM 128x64 pixlar, användes för att visualisera spelplanen och spelets övriga delkomponenter; markören, symboler, poängställning.
- En variabel varistor användes för att kontrollera den riktning i vilken LCD-skärmen uppvisar bäst ljusstyrka
- Resistorer kopplade till bland annat knapparna
- Ett mönsterkort på vilket alla hårdvarukomponenter monterades
- En strömkälla på 5 V och 5 A

Mjukvara

Vid programmeringen av spelet användes programspråket C. Den kod som skrevs hanterades i programvaran Atmel studio 6.1. Felsökning av såväl hårdvara som mjukvara samt exekvering av koden via processorn genomfördes med hjälp av JTAG.

Programmets basala struktur utgörs av en mainmetod vilken innefattar de metoder och den övriga kod som behövs för att köra en spelomgång. Därutöver innefattar källkoden specifika metoder för bland annat; rita ut symboler, spelplan och poängställning, undersöka om någon av spelarna vunnit innevarande omgång och eller hela spelet.

Därtill innehåller koden attribut vilka utgörs av exempelvis vektorer och matriser som används för att lagra information under spelets gång.

Utförande

Arbetsprocessen kan delas in i ett antal olika delmoment vilka presenteras och beskrivs nedan.

Planering

I projektets första skede övervägdes flertalet idéer avseende det koncept enligt vilket spelet skulle utformas. Efter närmare diskussion föll dock valet på att utveckla ett fyra i rad – spel.

Då projektets typ och primära syfte fastslagits fortskred arbetet med utformning av en kravspecifikation utformad utifrån gruppens initiala idé. I samband med kravspecifikationen utformades en problemformulering i syfte att skapa ytterligare perspektiv på arbetets omfattning samt för att kartlägga den hårdvara som behövdes samt de funktioner som var nödvändiga för att spelet skulle fungera som önskat.

I planeringens slutskede utformades ett kopplingsschema vilket kom att utgöra referensmall för de kopplingar av hårdvaran som utfördes senare i projektet.

Kopplingsschema och kravspecifikation presenterades för och godkändes efter några mindre ändringar av handledare varpå det handfasta arbetet inleddes.

Konstruktion

Med färdigställt kopplingsschema som referens påbörjades konstruktionen av spelet. Den hårdvara som prototypen skulle komma att bestå av fästes vid mönsterkortet. Komponenter som processor, LED-skärm och joystick löddes fast vid mönsterplattan med hjälp av lödpenna och lödtenn. Övriga komponenter behövde ingen annan fästnanordning än de sladdar via vilka de kopplades till kretsens övriga komponenter.

Alla i prototypen ingående komponenter kopplades därefter samman enligt det kopplingsschema som utformats under planeringsfasen. Hårdvaran och komponenternas förbindelser med processorn testades sedan med JTAG. JTAG möjliggör manuell inställning av processorns olika pins och portar vilket underlättar felsökning av genomförda kopplingar.

Vid felsökningen upptäcktes ett begränsat antal fel vilka åtgärdades genom att lödningar gjordes om för att förbättra förbindelser mellan några av kretsens komponenter samt genom att ett fåtal sladdar virades om.

Programmering

Programmeringsarbetet inleddes med att testkod skrevs för varje hårdvarukomponent för att verifiera att dessa fungerade som de skulle. Parallellt med att testmetoder togs fram och användes programmerades övriga nödvändiga metoder som exempelvis metod för att rita ut en individuell pixel på skärmen, metod för att rensa skärmen etcetera. Datablad till LCD-skärm och microprocessor var oombärliga vid programmering av de metoder som användes för att ge grundläggande instruktioner till skärm och övriga komponenter.

Då grundläggande metoder skrivits utökades koden med metoder för att hantera utritning av spelplanen, markören, symboler, poängställning samt med metoder vilka hanterade kommunikationen mellan skärmen och vardera joystick.

Under programmeringsarbetets gång infördes begränsningar anpassade efter önskad funktion av prototypen samt regelverket för fyra i rad.

Resultat och diskussion

Efter en intressant och mycket lärorik arbetsprocess renderade projektet slutligen ett välfungerande, interaktivt och lättanvänt fyra i rad- spel vilket fungerar i enlighet med den kravspecifikation som utformades i planeringsstadiet.

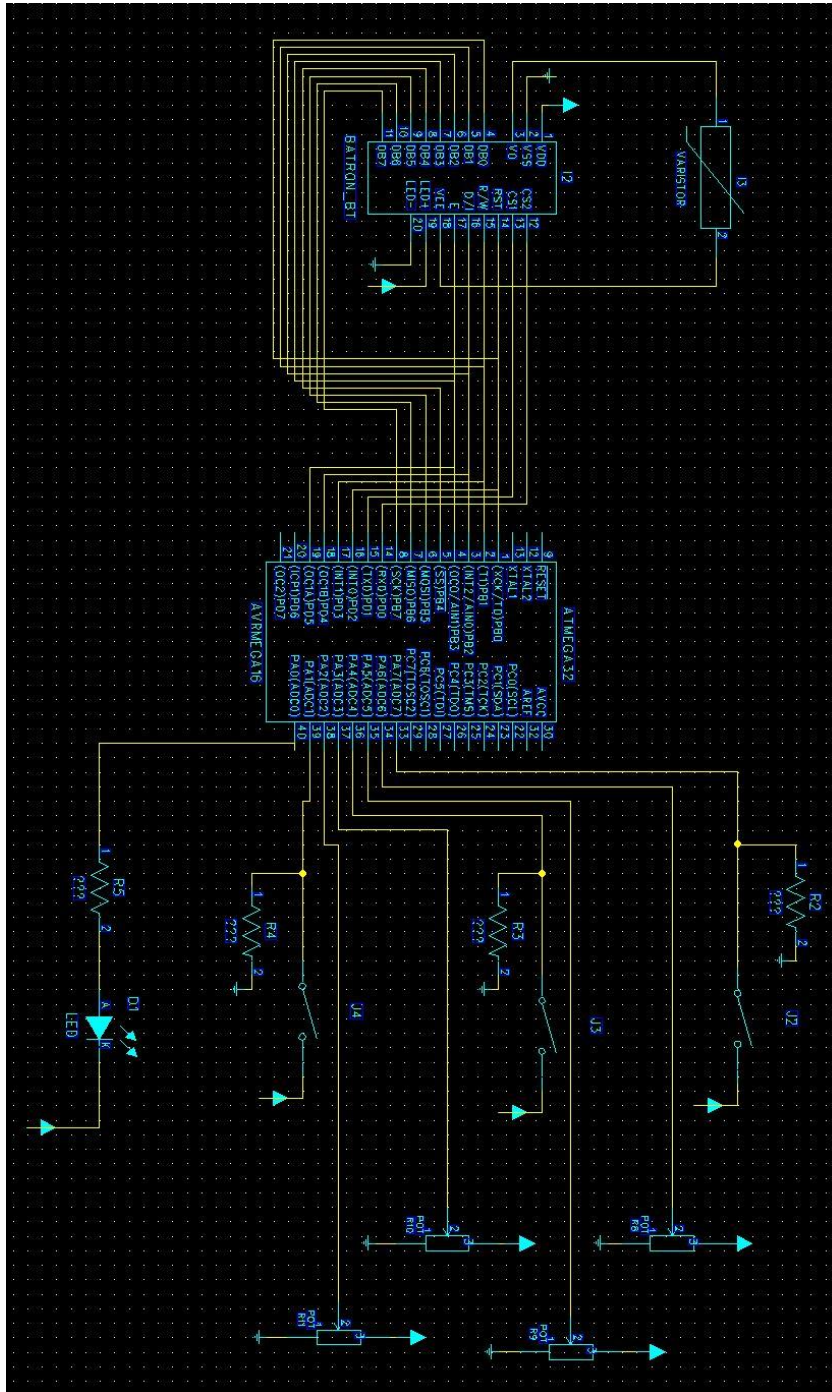
Projektet integrerade programmering och elektroteknik och gav därför en för industriella ekonomer en unik inblick i hur utvecklingen av en digital prototyp från hårdvara till mjukvaruprogrammering och ett fungerande program går till.

Till följd av relativt noggrann planering av arbetet vid projektets början fortskred arbetet utan större avbräck. Ett antal smärre modifieringar genomfördes efter felsökning. Vid programmering av mjukvaran upptäcktes dessutom ett minnesproblem vilket åtgärdades genom att en AT mega16 microprocessor byttes mot en AT mega32 vilken har något större minnesutrymme.

Då fokus låg på att uppnå en välfungerande prototyp lades mindre vikt vid att optimera koden varför några mindre fel kvarstår i mjukvaran. Dessa utgörs av att ena skärmhalvan är förskjuten i förhållande till den andra samt att ett av spelplanens sträck ritas ut två gången. Trots felsökning har anledningen till dessa fel inte kunnat hittas. En optimering av koden skulle innebära att ovan nämnda fel åtgärdas.

I det stora hela fungerar dock spelet väl och enligt kravspecifikationen.

Appendix 1: Kopplingschema



Appendix 2: Källkod

```
    /*  
 * spelet.c  
 *  
 * Created: 2016-04-13 14:17:09  
 * Author: Elin Blomstergren and Victor Sundgren  
 */  
#include <avr/io.h>  
#include <util/delay.h>  
  
unsigned char screen[8][128];  
int board[4][4]; /* 0 = player 0, 1 = player 1, 2 = empty spot */  
int player; /* either player 0 or 1 */  
int wins[2];  
int markerLoc[2];  
  
void setDataDirection();  
void ADC_init();  
void displayOn();  
void play();  
void clearScreen();  
void drawBoard();  
void reDraw();  
void ledOn();  
void ledOff();  
void readJoystickOne();  
void readJoystickTwo();  
void readButton();
```



```
int main()
{
setDataDirection();
ADC_init();
displayOn();
play();
return 0;
}
```

```
void newGame()
{
for(int a = 0; a < 4; a = a + 1){
for(int b = 0; b < 4; b = b + 1){
board[a][b] = 2;
}
}
markerLoc[0] = 0;
markerLoc[1] = 0;
}
```

```
void play()
{
wins[0] = 0;
wins[1] = 0;
player = !player;
newGame();
clearScreen();
drawBoard();
ledOn();
}
```

```
while(1){  
  
if(player){  
readJoystickOne();  
} else {  
readJoystickTwo();  
}  
readButton();  
}  
}
```

```
void putOnBoard()  
{  
int x = markerLoc[0];  
int y = markerLoc[1];  
if(board[x][y] == 2){  
board[x][y] = player;  
player = !player;  
markerLoc[0] = 0;  
markerLoc[1] = 0;  
checkWin();  
reDraw();  
}  
}
```

```
void win(void)  
{  
ledOn();  
_delay_ms(1000);  
ledOff();  
play();  
}
```

```
void checkWin(void)
{
if(board[0][0] == board[1][1] && board[0][0] == board[2][2] && board[0][0] == board[3][3] &&
board[0][0] != 2){
wins[board[0][0]] = wins[board[0][0]] + 1;
newGame();
} else if(board[0][3] == board[1][2] && board[0][3] == board[2][1] && board[0][3] == board[3][0] &&
board[0][3] != 2){
wins[board[0][3]] = wins[board[0][3]] + 1;
newGame();
} else {
for(int a = 0; a < 4; a = a + 1){
if(board[a][0] == board[a][1] && board[a][0] == board[a][2] && board[a][0] == board[a][3] &&
board[a][0] != 2){
wins[board[a][0]] = wins[board[a][0]] + 1;
newGame();
break;
} else if(board[0][a] == board[1][a] && board[0][a] == board[2][a] && board[0][a] == board[3][a] &&
board[0][a] != 2) {
wins[board[0][a]] = wins[board[0][a]] + 1;
newGame();
break;
}
}
}
if(wins[0] == 3 || wins[1] == 3){
win();
}
}
```

```
{  
DDRA = 0b00000001;  
DDRB = 0b11111111;  
DDRD = 0b11111111;  
}
```

void rwLow(void)

```
{  
PORTD &= ~_BV(PD3);  
}
```

void rwHigh(void)

```
{  
PORTD |= _BV(PD3); // sets PD3 to 1  
}
```

void rsLow(void)

```
{  
PORTD &= ~_BV(PD4);  
}
```

void rsHigh(void)

```
{  
PORTD |= _BV(PD4);  
}
```

void eLow(void)

```
{  
PORTD &= ~_BV(PD5);  
}
```

```
void eHigh(void)
```

```
{  
PORTD |= _BV(PD5);  
}
```

```
void selectC1(void)
```

```
{  
PORTD |= _BV(PD0);  
PORTD &= ~_BV(PD1);  
}
```

```
void selectC2(void)
```

```
{  
PORTD |= _BV(PD1);  
PORTD &= ~_BV(PD0);  
}
```

```
void setXAdress(int x)
```

```
{  
eHigh();  
rsLow();  
rwLow();  
PORTB = 0b10111000 | (x/8);  
eLow();  
eHigh();  
}
```

```
void setYAdress(int y)
```

```
{  
eHigh();  
rsLow();  
rwLow();  
PORTB = 0b01000000 | y;  
eLow();  
eHigh();  
}
```

```
void displayOn(void)
```

```
{  
eHigh();  
PORTB = 0b00111111;  
rwLow();  
rsLow();  
PORTD |= _BV(PD2);  
selectC1();  
eLow();  
eHigh();  
  
selectC2();  
eLow();  
eHigh();  
}
```

```
void ledOn(void)
```

```
{  
PORTA |= _BV(PA0); // Sets PA0 to 1  
}
```

```
{  
PORTA &= ~_BV(PA0); // Sets PA0 to 0  
}
```

```
void ADC_init(void) //Initiera ADC
```

```
{  
ADMUX = 0b01000000;  
ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);  
}
```

```
int ADC_read(unsigned char channel)
```

```
{  
ADMUX = channel;  
ADCSRA|=(1<<ADSC);  
while(!(ADCSRA & (1<<ADIF)));  
ADCSRA|=(1<<ADIF);  
return (ADC);  
}
```

```
void readJoystickOne()
```

```
{  
int x = ADC_read(0b01000101);  
int y = ADC_read(0b01000110);  
if(x > 800) {  
if(markerLoc[0] < 3) {  
markerLoc[0] = markerLoc[0] + 1;  
}  
_delay_ms(500);  
reDraw();  
}
```

```
if(x < 200) {  
  
    if(markerLoc[0] > 0) {  
        markerLoc[0] = markerLoc[0] - 1;  
    }  
    _delay_ms(500);  
    reDraw();  
}  
  
if(y > 800) {  
    if(markerLoc[1] > 0) {  
        markerLoc[1] = markerLoc[1] - 1;  
    }  
    _delay_ms(500);  
    reDraw();  
}  
  
if(y < 200) {  
    if(markerLoc[1] < 3) {  
        markerLoc[1] = markerLoc[1] + 1;  
    }  
    _delay_ms(500);  
    reDraw();  
}  
}  
  
void readJoystickTwo()  
{  
    int x = ADC_read(0b01000010);  
    int y = ADC_read(0b01000011);  
    if(x < 200) {  
        if(markerLoc[0] < 3) {  
            markerLoc[0] = markerLoc[0] + 1;
```



```
_delay_ms(500);
```

```
reDraw();
```

```
}
```

```
if(x > 800) {
```

```
if(markerLoc[0] > 0) {
```

```
markerLoc[0] = markerLoc[0] - 1;
```

```
}
```

```
_delay_ms(500);
```

```
reDraw();
```

```
}
```

```
if(y < 200) {
```

```
if(markerLoc[1] > 0) {
```

```
markerLoc[1] = markerLoc[1] - 1;
```

```
}
```

```
_delay_ms(500);
```

```
reDraw();
```

```
}
```

```
if(y > 800) {
```

```
if(markerLoc[1] < 3) {
```

```
markerLoc[1] = markerLoc[1] + 1;
```

```
}
```

```
_delay_ms(500);
```

```
reDraw();
```

```
}
```

```
}
```

```
void readButton()
```

```
{
```

Rapport Digitala Projekt EITF11
Grupp 12
Elin Blomstergren, Victor Sundgren
Handledare: Bertil Lindvall
int buttonval= PINA & 0b10010010;

2016-05-19

```
if(buttonval== 0b00000010) {  
    ledOn();  
    _delay_ms(1000);  
    ledOff();  
    _delay_ms(1000);  
    play();  
} else if(buttonval == 0b00010000) {  
    if(!player) {  
        putOnBoard();  
    }  
} else if(buttonval == 0b10000000) {  
    if(player) {  
        putOnBoard();  
    }  
}  
}
```

```
void reDraw()  
{  
    clearScreen();  
    drawBoard();  
}
```

```
void draw(int x, int y)  
{  
    eHigh();
```

```
int newY;  
if(y < 63){
```

```
selectC1();
```

```
newY = y;
```

```
} else {
```

```
selectC2();
```

```
newY = y - 64;
```

```
}
```

```
setXAdress(x);
```

```
setYAdress(newY);
```

```
rwLow();
```

```
rsHigh();
```

```
screen[x/8][y] = screen[x/8][y] | (1<<(x%8));
```

```
PORTB = screen[x/8][y];
```

```
eLow();
```

```
eHigh();
```

```
}
```

```
void clearScreen(void)
```

```
{
```

```
for(int a = 0; a < 64; a = a + 8){
```

```
for(int b = 0; b < 64; b = b + 1){
```

```
selectC1();
```

```
setXAdress(a);
```

```
setYAdress(b);
```

```
rwLow();
```

```
rsHigh();
```

```
PORTB = 0b00000000;
```

```
eLow();
```

```
eHigh();
```

```
selectC2();  
setXAdress(a);  
setYAdress(b);  
rwLow();  
rsHigh();  
PORTB = 0b00000000;  
eLow();  
eHigh();  
}  
}  
}
```

```
void drawScore(void)  
{  
switch(wins[1]){  
case 0:  
for(int a = 12; a < 24; a = a + 1){  
draw(27, a);  
draw(28, a);  
draw(34, a);  
draw(35, a);  
}  
for(int a = 28; a < 35; a = a + 1){  
draw(a, 11);  
draw(a, 12);  
draw(a, 23);  
draw(a, 24);  
}  
draw(30, 16);
```

```
draw(30, 17);

draw(31, 17);

draw(31, 18);

draw(32, 18);

draw(32, 19);

break;

case 1:

for(int a = 29; a < 35; a = a + 1){

draw(a, 11);

draw(a, 12);

}

for(int a = 13; a < 25; a = a + 1){

draw(31, a);

draw(32, a);

}

draw(29, 23);

draw(30, 23);

draw(30, 24);

break;

case 2:

for(int a = 27; a < 36; a = a + 1){

draw(a, 11);

draw(a, 12);

}

int y = 13;

int x = 28;

while(x < 35 && y < 20){

draw(x, y);

draw(x+1, y);

draw(x+1, y+1);
```

x = x+1;

y = y+1;

}

for(int a = 28; a < 35; a = a + 1){

draw(a, 23);

draw(a, 24);

}

draw(27, 13);

draw(27, 14);

draw(34, 21);

draw(34, 22);

draw(35, 21);

draw(35, 22);

draw(35, 23);

draw(27, 22);

draw(27, 23);

draw(28, 22);

break;

case 3:

for(int a = 28; a < 35; a = a + 1){

draw(a, 11);

draw(a, 12);

}

for(int a = 12; a < 18; a = a + 1){

draw(34, a);

draw(35, a);

}

for(int a = 29; a < 34; a = a + 1){

draw(a, 17);

}

int c = 30;

int d = 19;

while(c < 34 && d < 23){

draw(c,d);

draw(c+1, d);

draw(c+2,d);

c = c + 1;

d = d + 1;

}

for(int a = 27; a < 36; a = a + 1){

draw(a, 23);

draw(a, 24);

}

draw(27, 12);

draw(27, 13);

draw(28, 13);

break;

}

switch(wins[0]){

case 0:

for(int a = 105; a < 117; a = a + 1){

draw(28, a);

draw(29, a);

draw(35, a);

draw(36, a);

}

for(int a = 29; a < 36; a = a + 1){

draw(a, 116);

```
draw(a, 117);

draw(a, 104);

draw(a, 105);
}

draw(33, 111);

draw(33, 112);

draw(32, 110);

draw(32, 111);

draw(31, 109);

draw(31, 110);

break;

case 1:

for(int a = 29; a < 35 ; a = a + 1){

draw(a, 116);

draw(a, 117);

}

for(int a = 104; a < 116; a = a + 1){

draw(31, a);

draw(32, a);

}

draw(34, 105);

draw(33, 105);

draw(33, 104);

break;

case 2:

for(int a = 28; a < 37; a = a + 1){

draw(a, 116);

draw(a, 117);

}

int c = 29;
```



```
while(c < 36 && d < 116){  
draw(c, d);  
draw(c-1, d);  
draw(c-1, d+1);  
draw(c, d+1);  
c = c + 1;  
d = d + 1;  
}  
for(int a = 29; a < 36; a = a + 1){  
draw(a, 104);  
draw(a, 105);  
}  
draw(36, 115);  
draw(36, 114);  
draw(29, 105);  
draw(29, 106);  
draw(29, 107);  
draw(28, 106);  
draw(28, 107);  
draw(36, 105);  
draw(36, 106);  
draw(35, 106);  
break;  
case 3:  
for(int a = 28 ; a < 36; a = a + 1){  
draw(a, 116);  
draw(a, 117);  
}  
for(int a = 111; a < 117; a = a + 1){
```

draw(28, a);

}

for(int a = 29; a < 34; a = a + 1){

draw(a, 110);

draw(a, 111);

}

int x = 30;

int y = 106;

while(x < 34 && y < 110){

draw(x, y);

draw(x-1, y);

draw(x-2, y);

x = x + 1;

y = y + 1;

}

for(int a = 27; a < 36; a = a + 1){

draw(a, 104);

draw(a, 105);

}

draw(36, 115);

draw(36, 116);

draw(35, 115);

draw(35, 114);

draw(36, 114);

break;

}

}

void drawCircle(int x, int y)

```
{  
for(int a = x + 3; a < x + 9; a = a + 1){  
draw(a, y+2);  
draw(a, y+9);  
}  
for(int a = y + 3; a < y + 9; a = a + 1){  
draw(x+2, a);  
draw(x+9, a);  
}  
}
```

```
void drawMarker(int x, int y)  
{  
for(int a = x + 3; a < x + 9; a = a + 1){  
draw(a, y+4);  
draw(a, y+5);  
draw(a, y+6);  
draw(a, y+7);  
}  
for(int a = y + 3; a < y + 9; a = a + 1){  
draw(x+4, a);  
draw(x+5, a);  
draw(x+6, a);  
draw(x+7, a);  
}  
}
```

```
void drawCross(int x, int y)  
{  
for(int a = x + 3; a < x + 9; a = a + 1){
```

```
draw(a, y+5);

draw(a, y+6);

}

for(int a = y + 3; a < y + 9; a = a + 1){
draw(x+5, a);
draw(x+6, a);
}
}

/* draws the board according to matrix board */
void drawBoard(void)
{
/* draw the lines of the board */
for(int a = 4; a < 57; a = a + 13){
for(int b = 36; b < 90; b = b + 1){
draw(a, b);
}
}

for(int a = 4; a < 57; a = a + 1){
for(int b = 36; b < 89; b = b + 13){
draw(a, b);
}
}

for(int a = 0; a < 8; a = a + 1){
for(int b = 0; b < 128; b = b + 1){
screen[a][b] = 0;
}
}

/* fill the board according to the matrix board */
for(int x = 0; x < 4; x = x + 1){
```

```
for(int y = 0; y < 4; y = y + 1){  
    if(board[x][y] == 0){  
        drawCircle(5+13*x, 37+13*y);  
    }  
    if(board[x][y] == 1){  
        drawCross(5+13*x, 37+13*y);  
    }  
    if(markerLoc[0] == x && markerLoc[1] == y){  
        drawMarker(5+13*x, 37+13*y);  
    }  
    }  
    }  
    }  
    for(int a = 0; a < 8; a = a + 1){  
        for(int b = 0; b < 128; b = b + 1){  
            screen[a][b] = 0;  
        }  
    }  
    /* draw current score */  
    drawScore();  
}
```

Rapport Digitala Projekt EITF11
Grupp 12
Elin Blomstergren, Victor Sundgren
Handledare: Bertil Lindvall

2016-05-19