Snake

Institutionen för Elektro- och informationsteknik vid Lunds Tekniska Högskola
EITF11 Digitala Projekt

Handledare: Bertil Lindvall
Författare: Fredrik Wegelid och Svante Pagels, I13

2016-05-23

Abstract

This report is a part of the course Digital Systems, Project Laboratory (EITF11), at the Faculty of Engineering (LTH) at Lund University. The aim of the course is to illustrate how developing work can be done in the industry. The authors have designed, developed, build, debugged and documented a system for playing the game Snake. The result of the project is a physical construction, complete with documentation, where the player is able to enjoy this classic game.

Innehållsförteckning

Inledning

Denna rapport är en dokumentation av ett projekt som genomförts under våren 2016 i kursen Digitala projekt (EITF11) vid Lunds Tekniska Högskola. Syftet med kursen är att illustrera industriellt utvecklingsarbete. Huvuddelen av kursen består i att konstruera, bygga och testa en konstruktion. Författarna av denna rapport har valt en konstruktion för att spela datorspelet Snake.

Snake är ett klassiskt spel från 1976, till stor del känt för sitt framträdande på Nokiatelefoner. Spelet går ut på att styra en orm som ständigt rör sig framåt. På spelplanen finns det alltid ett äpple och målet med spelet är att samla så många av dessa som möjligt, där insamlade äpplen motsvarar spelarens poäng. Spelet fortsätter till dess att ormen krockar med sig själv eller väggen.

Rapporten är uppbyggd i fyra huvudsakliga delar. Den första delen, teoridelen, behandlar kravspecifikation för system samt vilka komponenter som ingår. Därefter kommer en del om projektets genomförande där det förklaras hur processen gått till. Rapportens sista del är resultat och diskussion där konstruktionen presenteras och projektet diskuteras. Bifogat finns även appendix med kopplingsschema för konstruktionen samt källkoden för systemet.

Teori

Kravspecifikation

Arbetet inleddes med att att definiera vilka krav som behövde uppfyllas av både hård- och mjukvara. Kraven delades upp i funktionella och kvalitetskrav och de funktionella kraven delades i sin tur upp i krav för mjukvara och för hårdvara. Syftet med kravspecifikationen är att lägga grunden för kopplingsschema och för att kontinuerligt kunna följa att utveckling av hårdvara och mjukvara uppfyller de krav som ställs på prototypen. Nedan följer kraven som ställts på snakespelet.

Funktionella krav

Hårdvara

- Det ska finnas en display
    - Ormen och äpplet ska visas på displayen
    - Ormen ska befinna sig i en bana, representerad av en fyrkant
    - Antal poäng (insamlade äpplen) ska visas på displayen
- Det ska finnas en fyra knappar för att styra ormen
    - Varje knapp motsvarar en riktning
    - Om spelaren trycker på en av dessa knappar ska ormen vrida sig i den riktningen
        - Om ormen redan rör sig i den riktningen händer ingenting

Mjukvara

- Ormen ska ständigt röra sig framåt
- Om ormen kör in i sin egen kropp eller i väggen är spelet slut
    - Spelet ska kunna startas igen genom att trycka på valfri knapp
- Om ormen kör in i ett äpple ska:
    - Ormens kropp bli en enhet längre
    - Antalet insamlade äpplen uppdateras
    - Ett nytt äpple skapas

Kvalitetskrav

- Fördröjning mellan knapptryckning och respons på skärmen ska inte vara för lång
- Den grafiska displayen ska vara uppbyggd av 128x64 punkter.

Hårdvara

Komponenter

Processor

Processorn som använts till prototypen är en Atmel ATmega32. Det är en 8-bitars mikrokontroller med 32 kB flashminne, 6 kB statisk RAM och 40 olika pins varav 32 är indelade i fyra olika portar och fyra tillför mikrokontrollern med ström. Port B består av pins 1-8 och är databus till LCD. Port D består av pins 14-21 och är kopplade till knappar och logik. Port C består av pins 22-29 och är kopplade till JTAG. Port A består av pins 33-40 och används för att skicka instuktioner till LCD. För mer information kring koppling se appendix A.

LCD

Skärmen som används är en bakgrundsbelyst LCD, GDM12864HLCM. Skärmen har 128x64 pixlar uppdelat på två skärmhalvor. Via databus (Port B på mikrokontrollern) ändras register om 8 bitar i taget.

Knappar

För att styra ormen i spelet används fyra knappar. Knapparna är kopplade till Port B hos mikrokontrollern.

Logik

Till interupt pins i Port B, INT0 och INT1 är även två disjunkta (OR) logiska grindar inkopplade (se appendix A).

JTAG

För att programmera mikrokontrollern via programmet Atmel Studios användes en JTAG.

Resistorer

Fem stycken resistorer användes även i prototypen. Fyra stycken resistorer på 10 kΩ vardera användes till var och en av knapparna och en variabel resistor användes för att justera kontrast på LCD.

Kopplingsschema

Se appendix A.

Metod

Arbetet påbörjades med att bryta ner problemet i beståndsdelar. En första kravspecifikation skrevs och lämnades in till handledaren för projektet. Efter att denna kravspecifikation godkänts ritades ett kopplingsschema som också godkändes av projekthandledaren. Kravspecifikation och kopplingsschema uppdaterades kontinuerligt under projektets gång i de fall då det var nödvändigt.

Material, komponenter och verktygslåda kvitterades därefter ut och det riktiga arbetet kunde påbörjas. Kopplingsschemat följdes och komponentera kopplades och löddes samman till en prototyp. Med hjälp av JTAG och Atmel Studio 6.2 kunde processorn testas och via felsökningsfunktionen (engelska: debugger) erhölls en större insikt i hur de olika portarna reagerade med varandra. Under detta steg i projektet användes datablad för de olika komponenterna i stor utsträckning för att förstå hur processorn skulle kommunicera med övriga komponenter.

När skärmen kunde startas via felsökningsfunktionen påbörjades kodandet och simpla metoder skrevs för att starta och släcka skärmen. Efter att ha förstått hur pixlar tänds och släcks på skärmen skrevs även metoder för att rita eller sudda en viss pixel. Då skärmen kräver att åtta pixlar skrivs samtidigt behövdes en virtuell skärm, det vill säga att information sparas över vilka pixlar som är tända och vilka som är släckta. Det insågs då att den ursprungliga processorn inte hade tillräckligt arbetsminne för att spara all denna information. Efter överläggning med projekthandledaren uppgraderades processorn från ATMega16 till ATMega32.

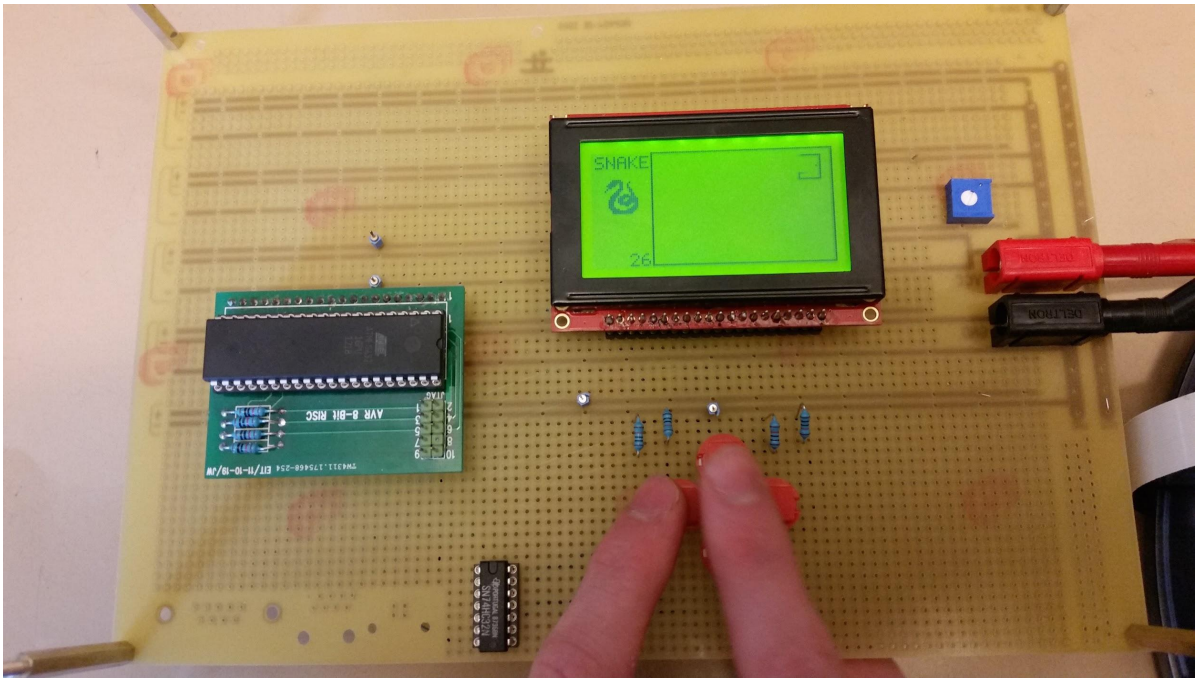
När metoder fanns för att rita på skärmen via ett koordinatsystem påbörjades arbetet med att få knapparna att ge respons till processorn. Här användes på felsökningsfunktionen och datablad flitigt för att första hur så kallade interrupts fungerar i kodskrivandet.

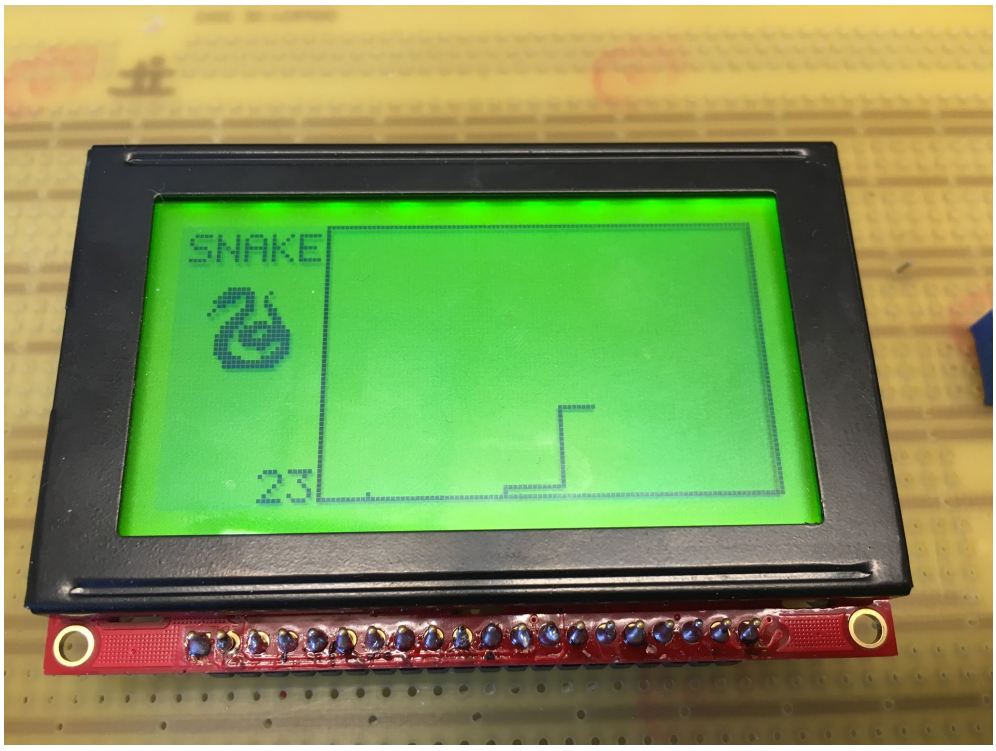
Efter att hårdvaran fungerade som tänkt kunde själva programmet för Snake skrivas. I detta steg användes kravspecifikationen för att se till att samtliga krav som ställdes på spelet uppfylldes. Detta var en iterativ process med mycket testning på den faktiska hårdvaran för att försäkra sig om att spelet fungerade som tänkt. Allt eftersom kunde fler funktioner läggas till. Exempel på dessa är poängräknare och kontroll av självkollision.

Det sista steget i projektet var att testa spelet i stor utsträckning för att se till att samtliga krav var uppfyllda och att det var tillfredsställande att spela.

Resultat

Projektet resulterade i en konstruktion som uppfyller samtliga krav i kravspecifikationen. Spelet fungerar väl och svårighetsgraden är enligt författarna på en lagom nivå. Nedan finns två bilder på konstruktionen i användning.





Diskussion

Trots begränsade kunskaper inom både C-programmering och elektronik, gick projektet utmärkt. Svårgiheter som uppkommit längs vägen har till stor del bestått av svårigheter att förstå, allt från datablad till upplägg och struktur inom C-programmering och Atmel Studios. Varje motgång har däremot lett till lärdom och mot slutet av perioden flöt arbetet på bra.

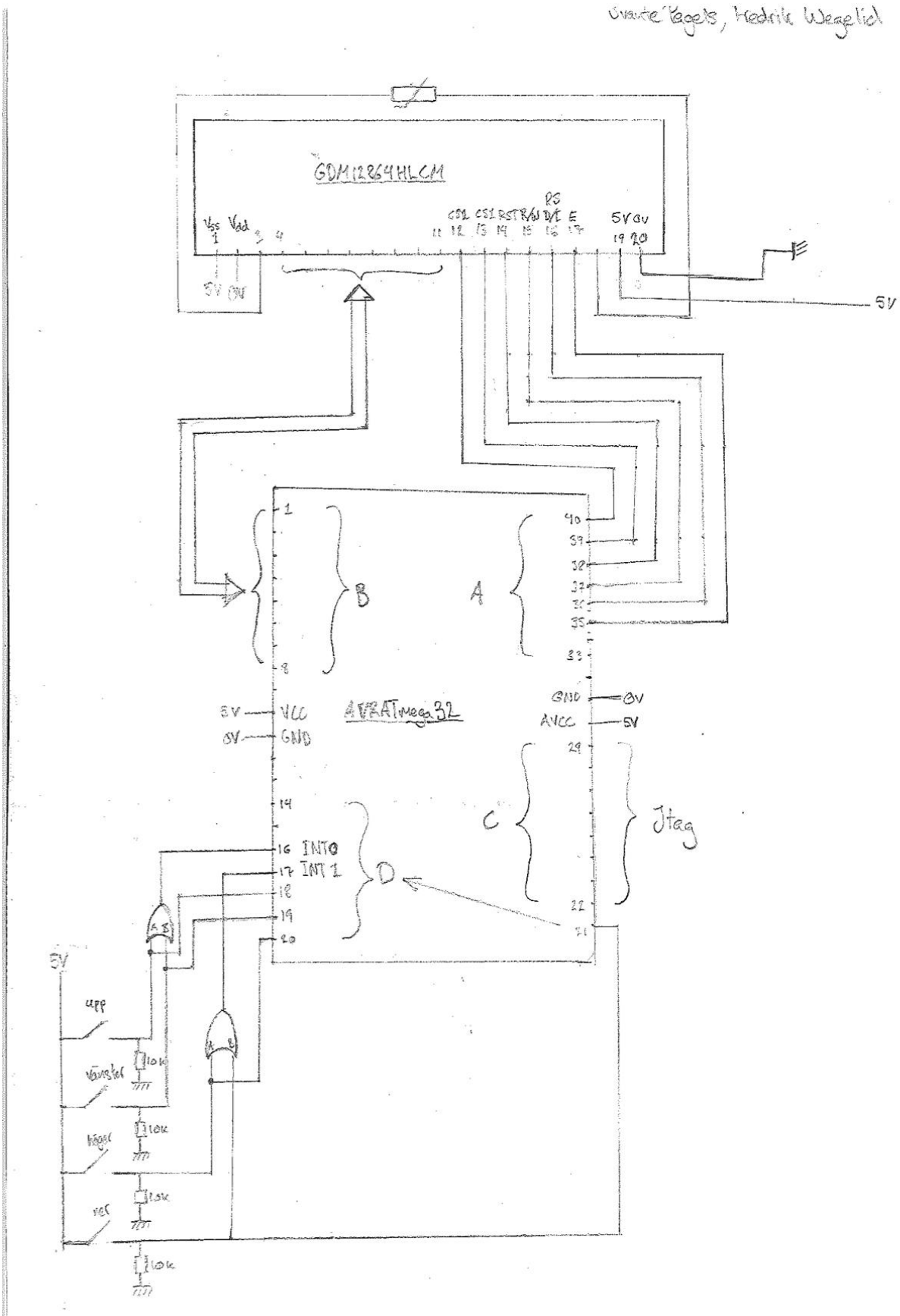
Ett av problemen som uppkommit under arbetet var att mikrokontrollerns ursprungliga arbetsminne på 16 kB inte räckte till för att spara en virtuell bild av skärmen som användes vid uppdatering av skärmen. Detta var ett svåridentifierat problem där lärdomen drogs att man alltid bör hålla reda på vad man sparar i applikationen då man arbetar med ett begränsat RAM.

I övrigt insågs vikten ganska omgående att förståelse av databladen var av största vikt. Många timmar spenderades i Atmel Studios på grund av otillräcklig förståelse av datablad och framförallt LCD och mikrokontrollens funktioner.

Överlag anses ändå en fungerande prototyp vara ett gott resultat och projektgruppen är mycket nöjda med insatsen.

Ett stort tack ska även ges till handledare Bertil Lindvall som stått ut med både dumma frågor och dåliga skämt under projektperioden, tack!

Appendix A - Kopplingsschema



Appendix B - Källkod

```
/*
 * snake.c
 *
 * Created: 2016-04-11 16:01:11
 *  Author: Svante Pagels och Fredrik Wegelid
 */

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

char virtualDisplay [8][128];

char appleLocation[2];
char snakeLocation[2][200];

char snakeLength;
char snakeDirection;
char snakeAlive;

char points;

char gameStarted = 0;

char collision(char x, char y);

int main(void) {
	setDataDirection();
	startDisplay();
	clearDisplay();
	setupBoard();
	setupSnake();
	enableInterrupt();

   while(1) {

		while (gameStarted == 0) {}
		points = 0;
		writePoints();
```

```
                eraseSnake();
                setupSnake();
                newApple();

                while (snakeAlive == 1) {
                        _delay_ms(35);
                        moveSnake();
                }
        }
}

void setupBoard() {
        writeSnake(2, 2);
        drawBoard();
        drawSlytherin(15,7);
        writePoints();
}

void newApple() {
        erase(appleLocation[0], appleLocation[1]);
        char x;
        char y;
        do {
        x = 2 + rand() % 60;
        y = 33 + rand() % 93;
        } while (collision(x,y));
        appleLocation[0] = x;
        appleLocation[1] = y;
        draw(appleLocation[0],appleLocation[1]);
}

void moveSnake() {

        char newX;
        char newY;

        switch (snakeDirection) {
                case 1:
                newX = snakeLocation[0][0];
                newY = snakeLocation[1][0] + 1;
                break;
                case 2:
                newX = snakeLocation[0][0] - 1;
                newY = snakeLocation[1][0];
```

```
                break;
                case 3:
                newX = snakeLocation[0][0];
                newY = snakeLocation[1][0] - 1;
                break;
                case 4:
                newX = snakeLocation[0][0] + 1;
                newY = snakeLocation[1][0];
                break;
        }

        if (newX >= 62 || newX <= 0 || newY >= 127 || newY <=32 || collision(newX, newY) == 1)
{
                _delay_ms(500);
                snakeAlive = 0;
                gameStarted = 0;
        } else {
                erase(snakeLocation[0][snakeLength - 1], snakeLocation[1][snakeLength-1]);

                for (char i = snakeLength - 1; i > 0; i--) {
                        snakeLocation[0][i] = snakeLocation[0][i-1];
                        snakeLocation[1][i] = snakeLocation[1][i-1];
                }

                snakeLocation[0][0] = newX;
                snakeLocation[1][0] = newY;
                drawSnake();

                if (newX == appleLocation[0] && newY == appleLocation[1]) {
                        points++;
                        snakeLength++;
                        newApple();
                        writePoints();
                        drawSnake();
                }
        }
}

char collision(char x, char y) {
        for (char i = 0; i < snakeLength; i++) {
                if (snakeLocation[0][i] == x && snakeLocation[1][i] == y) {
                        return 1;
                }
        }
```

```
	return 0;
}

void setupSnake() {
	snakeLength = 15;
	snakeDirection = 4;
	snakeAlive = 1;
	for (char i = 0; i < 15; i++) {
		snakeLocation[0][i] = 32 - i;
		snakeLocation[1][i] = 80;
	}
	drawSnake();
}

void drawSnake() {
	for (char i = 0; i < snakeLength; i++) {
		draw(snakeLocation[0][i], snakeLocation[1][i]);
	}
}

void eraseSnake() {
	for (char i = 0; i < snakeLength; i++) {
		erase(snakeLocation[0][i], snakeLocation[1][i]);
	}
}

void clearDisplay() {
	for (char y = 0; y < 128; y++) {
		for (char x = 0; x < 64; x++) {
			erase(x,y);
		}
	}
}

void draw(char x, char y) {
	virtualDisplay[x/8][y] |= (1<<(x%8));

	char newY;

	if (y < 64) {
		cs2high();
		cs1low();
		newY = y;
	} else {
```

```
                cs1high();
                cs2low();
                newY = y - 64;
        }

        setX(x / 8);
        setY(newY);

        writeData(x,y);

}

void erase(char x, char y) {
        virtualDisplay[x/8][y] &= ~(1<<(x%8));

        char newY;

        if (y < 64) {
                cs2high();
                cs1low();
                newY = y;
        } else {
                cs1high();
                cs2low();
                newY = y - 64;
        }

        setX(x / 8);
        setY(newY);

        eraseData(x,y);
}

void writeData(char x, char y) {
        rsHigh();
        rwLow();

        PORTB = virtualDisplay[x / 8][y];

        eHigh();
        eLow();
}

void eraseData(char x, char y) {
```

```
	rsHigh();
	rwLow();

	PORTB = virtualDisplay[x / 8][y];

	eHigh();
	eLow();
}

void setX(char x) {
	rsLow();
	rwLow();

	PORTB = 0b10111000 | x;

	eHigh();
	eLow();
}

void setY(char y) {
	rsLow();
	rwLow();

	PORTB = 0b01000000 | y;

	eHigh();
	eLow();
}

void setDataDirection() {
	DDRA = 0b11111111;
	DDRB = 0b11111111;
}

ISR(INT0_vect) {
	if (gameStarted == 0) {
		gameStarted = 1;
		_delay_ms(500);
		} else {
		switch (PIND) {
			case 0b00010110:
			turnUp();
			break;
			case 0b00100110:
```

```
                        turnLeft();
                        break;
                }
        }
}

ISR(INT1_vect) {
        if (gameStarted == 0) {
                gameStarted = 1;
                _delay_ms(500);
                } else {
                switch (PIND) {
                        case 0b10001000:
                        turnDown();
                        break;
                        case 0b01001000:
                        turnRight();
                        break;
                }
        }
}

void turnRight() {
        if (snakeDirection != 3) {
                snakeDirection = 1;
        }
}

void turnLeft() {
        if (snakeDirection != 1) {
                snakeDirection = 3;
        }
}

void turnUp() {
        if (snakeDirection != 4) {
                snakeDirection = 2;
        }
}

void turnDown() {
        if (snakeDirection != 2) {
                snakeDirection = 4;
        }
```

```
}

void startDisplay() {
	PORTB = 0b00111111;

	cs1low();
	cs2low();

	resetHigh();
	rwLow();
	rsLow();
	eHigh();

	cs1high();
	cs2high();

	eLow();
	eHigh();
}

void stopDisplay() {
	PORTB = 0b00111110;

	cs1low();
	cs2low();

	resetHigh();
	rwLow();
	rsLow();
	eHigh();

	cs1high();
	cs2high();

	eLow();
	eHigh();
}

void enableInterrupt() {
	PIND = 0b00000000;
	DDRD = 0b00000000;
	MCUCR |= 0b00001111;
	GICR |= 0b11000000;
	sei();
```

```
}

void cs2high() {
	PORTA |= 0b00000001;
}

void cs2low() {
	PORTA &= 0b11111110;
}

void cs1high() {
	PORTA |= 0b00000010;
}

void cs1low() {
	PORTA &= 0b11111101;
}

void resetHigh() {
	PORTA |= 0b00000100;
}

void resetLow() {
	PORTA &= 0b11111011;
}

void rwHigh() {
	PORTA |= 0b00001000;
}

void rwLow() {
	PORTA &= 0b11110111;
}

void rsHigh() {
	PORTA |= 0b00010000;
}

void rsLow() {
	PORTA &= 0b11101111;
}

void eHigh () {
	PORTA |= 0b00100000;
```

```
}

void eLow() {
	PORTA &= 0b11011111;
}

void writePoints() {
	for (char x = 56; x < 64; x++) {
		for (char y = 20; y < 31; y++) {
			erase(x,y);
		}
	}

	switch (points % 10) {
		case 1:
		draw1(56, 26);
		break;
		case 2:
		draw2(56, 26);
		break;
		case 3:
		draw3(56, 26);
		break;
		case 4:
		draw4(56, 26);
		break;
		case 5:
		draw5(56, 26);
		break;
		case 6:
		draw6(56, 26);
		break;
		case 7:
		draw7(56, 26);
		break;
		case 8:
		draw8(56, 26);
		break;
		case 9:
		draw9(56, 26);
		break;
		case 0:
		draw0(56, 26);
		break;
```

```
        }

        switch (points / 10) {
                case 1:
                draw1(56, 20);
                break;
                case 2:
                draw2(56, 20);
                break;
                case 3:
                draw3(56, 20);
                break;
                case 4:
                draw4(56, 20);
                break;
                case 5:
                draw5(56, 20);
                break;
                case 6:
                draw6(56, 20);
                break;
                case 7:
                draw7(56, 20);
                break;
                case 8:
                draw8(56, 20);
                break;
                case 9:
                draw9(56, 20);
                break;
        }
}

void draw1(char x, char y) {
        for (char i = 0; i < 7; i++) {
                draw(x + i, y + 2);
        }
        draw(x + 1, y + 1);
        draw(x + 6, y + 1);
        draw(x + 6, y + 3);
}

void draw2(char x, char y) {
        draw(x, y + 1);
```

```
	draw(x, y + 2);
	draw(x, y + 3);

	draw(x + 1, y);
	draw(x + 1, y + 4);

	draw(x + 2, y + 4);
	draw(x + 3, y + 3);
	draw(x + 4, y + 2);
	draw(x + 5, y + 1);

	for (char i = 0; i < 5; i++) {
		draw(x + 6, y + i);
	}
}

void draw3(char x, char y) {
	for (char i = 0; i < 5; i++) {
		draw(x, y + i);
	}

	draw(x + 1, y + 3);
	draw(x + 2, y + 2);
	draw(x + 3, y + 3);
	draw(x + 4, y + 4);
	draw(x + 5, y);
	draw(x + 5, y + 4);

	draw(x + 6, y + 1);
	draw(x + 6, y + 2);
	draw(x + 6, y + 3);
}

void draw4(char x, char y) {
	for (char i = 0; i < 5; i++) {
		draw(x + 4, y + i);
	}

	for (char i = 0; i < 7; i++) {
		draw(x + i, y + 3);
	}

	draw(x + 3, y);
	draw(x + 2, y + 1);
```

```
	draw(x + 1, y + 2);
}

void draw5 (char x, char y) {
	for (char i = 0; i < 5; i++) {
		draw(x, y + i);
	}

	draw(x + 1, y);

	for (char i = 0; i < 4; i++) {
		draw(x + 2, y + i);
	}

	draw(x + 3, y + 4);
	draw(x + 4, y + 4);

	draw(x + 5, y);
	draw(x + 5, y + 4);

	draw(x + 6, y + 1);
	draw(x + 6, y + 2);
	draw(x + 6, y + 3);
}

void draw6(char x, char y) {
	draw(x, y + 2);
	draw(x, y + 3);

	draw(x + 1, y + 1);
	draw(x + 2, y);

	for (char i = 0; i < 4; i++) {
		draw(x + 3, y + i);
	}

	draw(x + 4, y);
	draw(x + 4, y + 4);

	draw(x + 5, y);
	draw(x + 5, y + 4);

	draw(x + 6, y + 1);
	draw(x + 6, y + 2);
```

```
	draw(x + 6, y + 3);
}

void draw7(char x, char y) {
	for (char i = 0; i < 5; i++) {
		draw(x, y + i);
	}

	draw(x + 1, y + 4);
	draw(x + 2, y + 3);
	draw(x + 3, y + 2);

	draw(x + 4, y + 1);
	draw(x + 5, y + 1);
	draw(x + 6, y + 1);
}

void draw8(char x, char y) {
	for (char i = 1; i < 4; i++) {
		draw(x, y + i);
		draw(x + 3, y + i);
		draw(x + 6, y + i);
	}

	for (char i = 1; i < 3; i++) {
		draw(x + i, y);
		draw(x + i, y + 4);
		draw(x + 3 + i, y);
		draw(x + 3 + i, y + 4);
	}
}

void draw9(char x, char y) {
	draw(x + 6, y + 1);
	draw(x + 6, y + 2);

	draw(x + 5, y + 3);
	draw(x + 4, y + 4);

	for (char i = 1; i < 5; i++) {
		draw(x + 3, y + i);
	}

	draw(x + 2, y);
```

```
	draw(x + 2, y + 4);

	draw(x + 1, y);
	draw(x + 1, y + 4);

	draw(x, y + 1);
	draw(x, y + 2);
	draw(x, y + 3);
}

void draw0(char x, char y) {
	for (char i = 1; i < 6; i++) {
		draw(x + i, y);
		draw(x + i, y + 4);
	}

	for (char i = 1; i < 4; i++) {
		draw(x, y + i);
		draw(x + 6, y + i);
	}

	draw(x + 4, y + 1);
	draw(x + 3, y + 2);
	draw(x + 2, y + 3);
}

void writeSnake(char x, char y) {
	drawS(x,y);
	drawN(x, y + 6);
	drawA(x, y + 12);
	drawK(x, y + 18);
	drawE(x, y + 24);
}

void drawBoard() {
	for (char y = 32; y < 128; y++) {
		draw(0, y);
		draw(62, y);
	}
	for (char x = 0; x < 63; x++) {
		draw(x, 32);
		draw(x, 127);
	}
}
```

```
void drawSlytherin(char x, char y) {

	//rad 0
	for (char i = 4; i < 9; i++) {
		draw(x, y + i);
	}

	//rad 1
	for (char i = 3; i < 10; i++) {
		draw(x + 1, y + i);
	}
	draw(x + 1, y + 13);
	erase(x + 1, y + 4);

	//rad 2
	for (char i = 2; i < 10; i++) {
		draw(x + 2, y + i);
	}
	erase(x + 2, y + 7);
	draw(x + 2, y + 12);

	//rad 3
	for (char i = 1; i < 10; i++) {
		draw(x + 3, y + i);
	}
	erase(x + 3, y + 6);
	erase(x + 3, y + 7);
	draw(x + 3, y + 12);

	//rad 4
	draw(x + 4, y + 2);
	draw(x + 4, y + 3);
	draw(x + 4, y + 8);
	draw(x + 4, y + 9);
	draw(x + 4, y + 12);
	draw(x + 4, y + 13);

	//rad 5
	draw(x + 5, y + 1);
	draw(x + 5, y + 7);
	draw(x + 5, y + 8);
	draw(x + 5, y + 9);
	draw(x + 5, y + 13);
```

```
draw(x + 5, y + 14);

//rad 6
draw(x + 6, y + 0);
draw(x + 6, y + 6);
draw(x + 6, y + 7);
draw(x + 6, y + 8);
draw(x + 6, y + 13);
draw(x + 6, y + 14);

//rad 7
draw(x + 7, y + 5);
draw(x + 7, y + 6);
draw(x + 7, y + 7);
draw(x + 7, y + 14);
draw(x + 7, y + 15);

//rad 8
draw(x + 8, y + 4);
draw(x + 8, y + 5);
draw(x + 8, y + 6);
draw(x + 8, y + 7);
draw(x + 8, y + 10);
draw(x + 8, y + 11);
draw(x + 8, y + 12);
draw(x + 8, y + 14);
draw(x + 8, y + 15);

//rad 9-19 ifyllnad
for (char dx = 9; dx < 19; dx++) {
	for (char dy = 2; dy < 18; dy++) {
		draw(x + dx, y + dy);
	}
}

//rad 9
erase(x + 9, y + 2);
erase(x + 9, y + 7);
erase(x + 9, y + 17);

//rad 10
erase(x + 10, y + 2);
erase(x + 10, y + 6);
erase(x + 10, y + 11);
```

```
erase(x + 10, y + 12);

//rad 11
erase(x + 11, y + 6);
erase(x + 11, y + 10);
erase(x + 11, y + 11);

//rad 12
erase(x + 12, y + 5);
erase(x + 12, y + 6);

//rad 13
erase(x + 13, y + 5);
erase(x + 13, y + 6);
erase(x + 13, y + 7);
erase(x + 13, y + 14);
erase(x + 13, y + 15);

//rad 14
for (char i = 5; i < 15; i++) {
	erase(x + 14, y + i);
}
draw(x + 14, y + 11);
draw(x + 14, y + 12);

//rad 15
for (char i = 6; i < 14; i++) {
	erase(x + 15, y + i);
}

//rad 16
erase(x + 16, y + 2);
erase(x + 16, y + 8);
erase(x + 16, y + 9);
erase(x + 16, y + 10);
erase(x + 16, y + 11);
erase(x + 16, y + 17);

//rad 17
erase(x + 17, y + 2);
erase(x + 17, y + 3);
erase(x + 17, y + 16);
erase(x + 17, y + 17);
```

```
	//rad 18
	erase(x + 18, y + 2);
	erase(x + 18, y + 3);
	erase(x + 18, y + 15);
	erase(x + 18, y + 16);
	erase(x + 18, y + 17);

	//rad 19
	erase(x + 19, y + 2);
	erase(x + 19, y + 3);
	erase(x + 19, y + 4);
	erase(x + 19, y + 5);
	erase(x + 19, y + 6);
	erase(x + 19, y + 13);
	erase(x + 19, y + 14);
	erase(x + 19, y + 15);
	erase(x + 19, y + 16);
	erase(x + 19, y + 17);

}

void drawS(char x, char y) {
	for (char i = 1; i < 5; i++) {
		draw(x, y + i);
	}

	draw(x + 1, y);
	draw(x + 2, y);

	for (char i = 1; i < 4; i++) {
		draw(x + 3, y + i);
	}

	draw(x + 4, y + 4);
	draw(x + 5, y + 4);

	for (char i = 0; i < 4; i++) {
		draw(x + 6, y + i);
	}
}

void drawN(char x, char y) {
	for (char i = 0; i < 7; i++) {
		draw(x + i, y);
```

```
                draw(x + i, y + 4);
        }

        draw(x + 2, y + 1);
        draw(x + 3, y + 2);
        draw(x + 4, y + 3);
}

void drawA(char x, char y) {
        for (char i = 1; i < 4; i++) {
                draw(x, y + i);
                draw(x + 3, y + i);
        }

        for (char i = 1; i < 7; i++) {
                draw(x + i, y);
                draw(x + i, y + 4);
        }
}

void drawK(char x, char y) {
        for (char i = 0; i < 7; i++) {
                draw(x + i, y);
        }

        for (char i = 0; i < 4; i++) {
                draw(x + 3 + i, y + 1 + i);
                draw(x + 3 - i, y + 1 + i);
        }
}

void drawE(char x, char y) {
        for (char i = 0; i < 7; i++) {
                draw(x + i, y);
        }

        for (char i = 1; i < 5; i++) {
                draw(x, y + i);
                draw(x + 3, y + i - 1);
                draw(x + 6, y + i);
        }

}
```