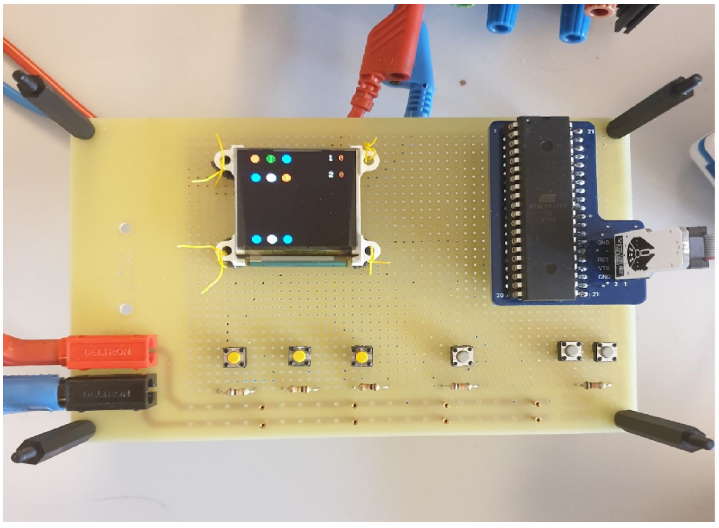
LUNDS UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörhögskolan vid Campus Helsingborg

Mastermindspel



Handledare: Bertil Lindvall, Lars-Göran Larsson
2019 Projektarbete för kursen Digitala System, EITA15
Utförd av: Henrik Brange, Linus Piggott, Christoffer Larsson

Sammanfattning

Detta är en rapport för ett projekt utfört under vårterminen 2019 som del av kursen Digitala System (EITA15) given på Lunds Tekniska Högskola vid Campus Helsingborg. Syftet med projektet var att för studenterna att öva på att samarbeta med att utveckla en produkt utifrån en kravspecifikation. Produkten i fråga är ett spel som kallas Mastermind som går ut på att en spelare ska gissa en kombination av färger i en sekvens. För varje gissning får spelaren reda på hur många av gissningarna var av rätt färg och hur många som var av rätt färg och på rätt plats i sekvensen. Spelet byggdes på ett mönsterkort och består utav en skärm av typ uOLED-128G2, en ATmega1284-mikrokontroller och ett antal knappar.

Abstract

This paper is a report on a project done during the spring term of 2019 as part of the course Digitala System (EITA15) at Lunds Tekniska Högskola in Helsingborg. The purpose of the project was for the students to practice cooperating on developing a product from a required specification. The product in question is a game called Mastermind. The game is played by the player guessing a sequence of colors. The game then responds by letting the player know the number of correct colors in the guessed sequence, as well as the number of correct colors in the correct places in the sequence. The game was built on a circuit board and consists of a uOLED-128G2-screen, an ATmega1284 microcontroller and a set of buttons.

Innehållsförteckning

1. Inledning

Kursen Digitala system, EITA15, vid Lunds Tekniska Högskola, innehåller ett projektmoment där studenterna fördelas i grupper för att skapa en konstruktion relevant till kursen. Studenterna fick själva välja vad för konstruktion de ville skapa och de skrev själva sin kravspecifikation. Utrustning som behövdes tillhandahölls utifrån denna specifikationen. Denna rapport avhandlar ett sådant projekt, specifikt skapandet av ett Mastermind-spel. Mastermind är ett spel där en spelare gissar en kombination av färger i en viss ordning [2]. För varje gissning får spelaren reda på hur många av gissningarna var av rätt färg och hur många som var av rätt färg och på rätt plats i sekvensen. Därefter görs nya gissningar med hjälp av informationen man fått tillbaka. Om spelaren inte har nått fram till rätt färgsekvens inom ett visst antal gissningar förlorar spelaren.

Spelet konstruerades med hjälp av ett mönsterkort, en uOLED-128G2, en ATmega1284 och ett antal knappar och resistorer. Mikrokontrollern programmerades i C med hjälp av Atmel studio.

Syftet bakom detta projektet var att ge studenterna erfarenhet av projektarbete, något som är vanligt förekommande för ingenjörer. Eleverna skulle även göra sig förstådda på relationen mellan hårdvara och mjukvara, informationssökning i diverse tekniska dokument och programspråket C. Som mål är att projektarbetet ska vara färdigt, tillsammans med en tillhörande rapport och hemsida, samt en presentation av projektet.

Ett par anledningar till varför ett Mastermind-spel valdes som projekt är för att det är ett bra sätt att utnyttja en flerfärgad skärm samt att ett spel var av alla gruppmedlemmars intresse.

2. Kravspecifikation

2.1 **Hårdvara:**

- Det ska finnas en display.

  - På displayen ska vi ha ett Mastermind-spel.

- Det ska finnas fyra knappar för att kontrollera vad som händer på skärmen.

  - Tre av knapparna ska användas till att skifta mellan färger för de tre .

  - Den fjärde knappen ska användas för att låsa in kombinationen av färger man gissar.

2.2 **Mjukvara:**

- Kombinationen man ska gissa skall slumpgenereras.

- När man låser in sin nuvarande gissning, så ska programmet ta reda på om gissningen är rätt eller fel.

  - Om gissningen är fel så ska:

    - Skärmen visa hur många färger som finns på rätt plats i en gissning.

    - Skärmen visa hur många färger som finns i den korrekta kombinationen, men på fel plats.

  - Om gissningen är korrekt så ska:

    - Den korrekta kombinationen visas.

3. Hårdvara

3.2.1 **ATmega1284 [1]:**
Denna mikrokontroller av märket Atmel användes för att programmera själva Mastermindspelet. Det är en 8-bitars AVR mikrokontroller med RISC-baserad arkitektur. Mikrokontrollern har en mängd olika funktioner däribland två USART-portar som användes för att kommunicera med skärmen.

3.2.2 **uOLED-128G2 [3]:**
Denna 1.5”, 128x128, flerfärgade OLED-display (Organic Light-Emitting Diod) av 4D Systems användes för att se resultaten av programmeringen som gjordes. Innehåller en egen processor kallad Goldelox som tolkar den information som skärmen mottar via sin UART-ingång.

3.2.3 **Atmel Ice AVR:**
En JTAG (Joint Test Action Group) av Atmel. Användes för att ladda över mjukvara till processorer. Nyttig när man testar ens mjukvara och debuggar.

3.2.4 **Knappar:**
Används för att styra spelet. Fem aktivt höga och en aktivt låg till reseten.

3.2.5 **Resistorer:**
Kopplades till knapparna. Detta för att kunna skapa antingen aktivt höga eller aktivt låga knappar.

4. Mjukvara

Programmet skrevs i programspråket C i Atmel Studio. Atmel Studio kan kompilera koden och föra över det till mikrokontrollerns minne via JTAGen. Har även bra debugging-verktyg. Se bilaga 3 för källkoden.

5. Metod

Till en början så planerades arbetet. Med en bra dialog mellan varann kunde det brainstormas effektivt, för att få fram ett förslag på en prototyp som skulle skapas. Med det gjort kunde det börja skrivas på en kravspecifikation. Lite av kraven till kravspecifikationen kunde man hitta i spelets menade regler.

För att hitta vilka komponenter man skulle använda så kunde man utgå ifrån kravspecifikation. Alla komponenter förseddes av handledarna. När dessa komponenter hade bestämts var det dags att titta i dess tekniska dokumentation [1][3]. Dessa dokument var till bra hjälp när man man behövde få reda på hur komponenterna fungerar och skulle kopplas. Därför var det inga problem att rita ett kopplingsschema efteråt.

Utifrån kopplingsschemat var det dags att koppla ihop komponenterna till konstruktionen. Hårdvaran monterades och kopplades på relevanta platser på ett mönsterkort. Komponenterna sattes fast och kopplades antingen genom lödning eller så virades sladdarna runt komponenternas pinnar med hjälp av en virpistol.

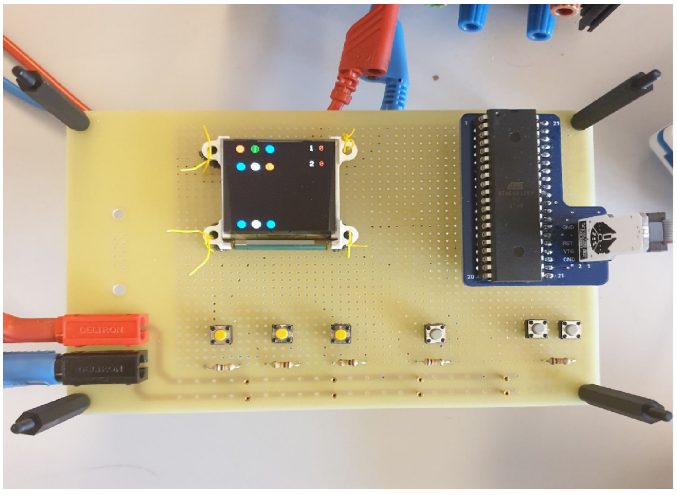
När allting var ihopkopplat var det dags att testa hårdvaran. Detta gjordes genom diverse testprogram i Atmel studio, samt en logikpenna. Till en början hade gruppen problem med kommunikationen mellan mikrokontrollern och skärmen. Det var inte så att konstruktionen var felkopplad, utan mikrokontrollern var inställd på fel klockfrekvens. När det hade ordnats visade det sig att konstruktionen var korrekt kopplad.

Eftersom konstruktionen var korrekt kopplad, var det dags att programmera spelet. Detta tog upp väldigt mycket av gruppens tid. Teknisk dokumentation för hur skärmen kunde programmeras var till mycket bra hjälp [4].

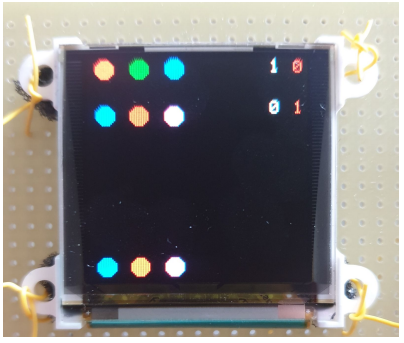
Det första som gruppen bestämde sig att programmera var en funktion som kunde generera en slumpmässig rad med siffror. Det är dessa siffror det är menat att man ska gissa fram, men de är representerade som färger. Därpå programmerade vi så att skärmen kunde rita ut en cirkel. Dessa skulle det behövas många av positionerade på olika ställen för vårt spel. Därefter ville vi se till att vi skulle lyckas byta färger på våra cirklar genom att trycka på knappar. Sedan såg vi till att programmet skrev korrekt siffror beroendes på vad man gissat för färgsekvens. Efter det var spelet i sig färdigt, men gruppen bestämde att programmera en enkel display både för när man förlorade och när man vann.

6. Resultat

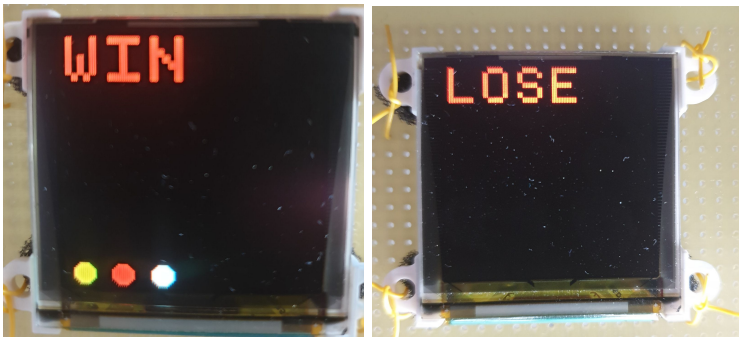
Resultatet av projektet är ett fungerande Mastermind-spel som uppfyller kraven av den angivna kravspecifikationen. En slumpmässig sekvens skapas av programmet. Knapparna kan skifta färg för de olika sekvenspositionerna, och låsa in en spelares gissning och kontrollera om det är korrekt eller inte. Se nedanstående figurer för visuell demonstration av vissa aspekter av konstruktionen.



Figur 1: En spelomgång där två gissningar har gjort och resultaten för dem har presenterats.



Figur 2: Närbild av spelet



Figur 3: Demonstration av hur det ser ut när spelaren antingen vinner eller förlorar.

7. Diskussion

Arbetets utveckling har knappast varit stadig, utan den har varierat väldigt mycket fram och tillbaka. Vissa dagar flöt arbetet på, medans vissa dagar satt man fast och kunde inte komma någon vart. Som när det skulle läras C när medlemmarna är främst vana vid det objektorienterade programspråket Java. Ett annat exempel var när kommunikationen till skärmen inte fungerade pga. att klockfrekvensen på mikrokontrollern var felaktigt inställd. Men när gruppen väl lyckats med att kunna programmera sin skärm så flöt arbetet på väldigt bra tack vare databladen som hittats. Medlemmarna känner sig självsäkra att de hade kunnat lägga till fler funktioner även om det inte krävdes. Till exempel en slags introduktionsdisplay, mer komplex win/lose-display eller fler svårighetsgrader till spelet.

Sammanfattningsvis instämmer gruppen om att arbetet blivit lyckat och att det varit till god nytta. Alla medlemmar var relativt oerfarna med att programmera hårdvara som den i detta projekt, och rent allmänt programspråket C. Men efter många om och men har gruppen en färdig prototyp att visa upp.

8 Källförteckning

[1] Atmel, “8-bit AVR Microcontrollers”, ATmega1284 data sheet, 2016.

Länk: https://www.eit.lth.se/fileadmin/eit/courses/eita15/Laborationer/Datorteknik/lab2/ATmega1284.pdf

[2] Hasbro, “Mastermind Manual”, 2010.

Länk: https://www.hasbro.com/common/documents/dad261421c4311ddbd0b0800200c9a66/819DF9B85056900B10B25C68C95E39E5.pdf

[3] 4D systems, “1.5” microOLED GOLDELOX Display“, uOLED-128G2 data sheet, 2015.

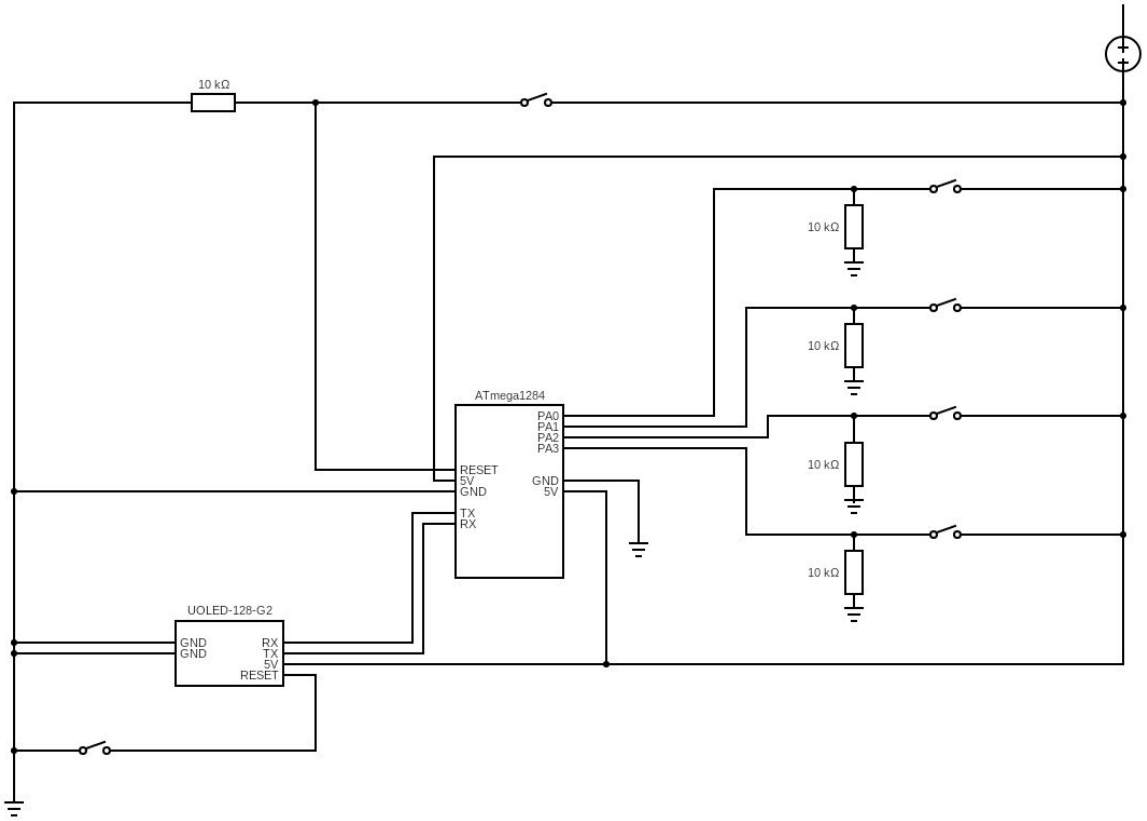
Länk: https://www.components-mart.com/datasheets/4f/UOLED-128-G2.pdf

[4] 4D Labs technical staff, “Goldelox processor serial commands reference manual”, 4D systems, 2017.

Länk: https://www.4dsystems.com.au/productpages/GOLDELOX/downloads/GOLDELOX_serialcmdmanual_R_2_0.pdf

9 Appendix

Bilaga A, Kopplingsschema:



Bilaga B, Knappmanual:

Knapp	Funktion
1	Bytar färg på sekvensposition 1.
2	Bytar färg på sekvensposition 2.
3	Bytar färg på sekvensposition 3.
4	Låser in din gissningssekvens, kontrollerar om du har rätt eller fel och låter spelaren påbörja en ny gissning.
5	Reset till skärmen. Används inte under spelandet.
6	Reset till programmet. Används för att kunna starta om spelet.

Bilaga C, Källkod:

```
/*
 * GccApplication3.c
 *
 * Created: 2019-05-15 15:13:59
 * Author : nat15hbr
 */
 #define F_CPU 8000000UL

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/delay.h>
#include <time.h>

int gameSequence[3]={1,1,1};
int guessSequence[3]={1,1,1};
int result[2];
int guessNbr;
int correctColors;
int correctColorAndPlace;
int win = 0;

void sequenceGenerator(){

	for(int i=0;i<3;i++){
		gameSequence[i] = rand() % 6;
	}

}


void checkSequence(){
	correctColors=0; //RESET
	correctColorAndPlace=0;

	for(int i=0; i<3; i++){

		if(guessSequence[0]==gameSequence[i] ||
guessSequence[1]==gameSequence[i] || guessSequence[2]==gameSequence[i]){
			correctColors++;
		}
	}

	for(int k=0; k<3; k++){
		if(guessSequence[k] == gameSequence[k]){
			correctColorAndPlace++;
```

```
                }
        }

        correctColors = correctColors-correctColorAndPlace;

        result[0] = correctColors;
        result[1] = correctColorAndPlace;
        if(correctColorAndPlace==3){winDisplay();}
        if(guessNbr==5){loseDisplay();}

}

void initMCU(){
        DDRA = 0;
        PCICR |= (1<<PCIE0);
        PCMSK0 |= (1<<PCINT0);
        PCMSK0 |= (1<<PCINT1);
        PCMSK0 |= (1<<PCINT2);
        PCMSK0 |= (1<<PCINT3);
        //PCIFR |= 1; //behövs denna??
}

void USART_Init()
{
UCSR0B = 0x00;
UCSR0C = 0x00;

UBRR0=51; //9600

UCSR0B |= (1<<RXEN0) | (1<<TXEN0);

UCSR0C |= (1<<UCSZ00) | (1<<UCSZ01);
//UCSR0C &= ~(1<<UCSZ02);
//UCSR0B &= ~(1<<USBS0);
}

void USART_Transmit( unsigned char data ) {
        /* Wait for empty transmit buffer */
        while ( !( UCSR0A & (1<<UDRE0)) ) ;
        /* Put data into buffer, sends the data */
        UDR0 = data;
}

unsigned char USART_Receive( void )
{
 /* Wait for data to be received */
```

```
 while ( !(UCSR0A & (1<<7)) )
 ;
 /* Get and return received data from buffer */
 return UDR0;
}

inline void send_string(unsigned char *msg, int len)
{
	for (int i = 0; i < len; i++) {
		USART_Transmit(msg[i]);
	}
}

void drawSequence(){

#define GHOSTWHITE		0xFFDF
#define GREEN		0x0400
#define RED		0xF800
#define BLUE		0x001F
#define MAGENTA		0xF81F
#define YELLOW		0xFFE0

	int colorMSB = 0;
	int colorLSB = 0;

		for(int i=0; i<3; i++){
			int xpos = i*20+10;

			if(guessSequence[i]==1){colorMSB = 0xFF; colorLSB = 0xDF;}
			if(guessSequence[i]==2){colorMSB = 0x04; colorLSB = 0x00;}
			if(guessSequence[i]==3){colorMSB = 0xF8; colorLSB = 0x00;}
			if(guessSequence[i]==4){colorMSB = 0x00; colorLSB = 0x1F;}
			if(guessSequence[i]==5){colorMSB = 0xF8; colorLSB = 0x1F;}
			if(guessSequence[i]==0){colorMSB = 0xFF; colorLSB = 0xE0;}

					//cmd	x	y	rad	color
				unsigned char data[] = {0xFF, 0xCC, 0x00, xpos, 0x00, 120,
0x00, 0x05, colorMSB, colorLSB}; //Draw circle
					send_string(data, sizeof(data));
					USART_Receive();
		}
}

void drawResponseSequence(){

		int colorMSB = 0;
```

```
int colorLSB = 0;
unsigned char bajs[] = {0xff, 0x7f, 0x00, 0x10};
unsigned char text_width[] = {0xff, 0x7c, 0x00, 0x01};
unsigned char text_height[] = {0xff, 0x7b, 0x00, 0x01};
send_string(bajs, sizeof(bajs));
USART_Receive();
send_string(text_width, sizeof(text_width));
USART_Receive();
send_string(text_height, sizeof(text_height));
USART_Receive();

for(int i=0; i<6; i++){
	if(guessNbr==i){
		int help = 0;
		if(guessNbr==1){help = 0;}
		if(guessNbr==2){help = i+1;}
		if(guessNbr==3){help = i+3;}
		if(guessNbr==4){help = i+5;}
		if(guessNbr==5){help = i+7;}

		unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00, 0x0F};
		send_string(move, sizeof(move));
		USART_Receive();

		if(result[0]==0 && result[1]==0){
			unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
			send_string(data, sizeof(data));
			USART_Receive();
			unsigned char text[] = {0x00, 0x06, '0', 0x00};
			send_string(text, sizeof(text));
			USART_Receive();
			unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
			unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
			send_string(move, sizeof(move));
			USART_Receive();
			send_string(data2, sizeof(data2));
			USART_Receive();
			unsigned char text2[] = {0x00, 0x06, '0', 0x00};
			send_string(text2, sizeof(text2));
			USART_Receive();

		}
```

```
                        //result[1] can never equal 3 because that means the player
won which is handled by the checkSequence function!
                        if(result[0]==0 && result[1]==1){
                                unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
                                send_string(data, sizeof(data));
                                USART_Receive();
                                unsigned char text[] = {0x00, 0x06, '0', 0x00};
                                send_string(text, sizeof(text));
                                USART_Receive();
                                unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
                                unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
                                send_string(move, sizeof(move));
                                USART_Receive();
                                send_string(data2, sizeof(data2));
                                USART_Receive();
                                unsigned char text2[] = {0x00, 0x06, '1', 0x00};
                                send_string(text2, sizeof(text2));
                                USART_Receive();
                        }
                        if(result[0]==0 && result[1]==2){
                                unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
                                send_string(data, sizeof(data));
                                USART_Receive();
                                unsigned char text[] = {0x00, 0x06, '0', 0x00};
                                send_string(text, sizeof(text));
                                USART_Receive();
                                unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
                                unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
                                send_string(move, sizeof(move));
                                USART_Receive();
                                send_string(data2, sizeof(data2));
                                USART_Receive();
                                unsigned char text2[] = {0x00, 0x06, '2', 0x00};
                                send_string(text2, sizeof(text2));
                                USART_Receive();
                        }
                        if(result[0]==1 && result[1]==0){
                                unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
                                send_string(data, sizeof(data));
```

```
                    USART_Receive();
                    unsigned char text[] = {0x00, 0x06, '1', 0x00};
                    send_string(text, sizeof(text));
                    USART_Receive();
                    unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
                    unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
                    send_string(move, sizeof(move));
                    USART_Receive();
                    send_string(data2, sizeof(data2));
                    USART_Receive();
                    unsigned char text2[] = {0x00, 0x06, '0', 0x00};
                    send_string(text2, sizeof(text2));
                    USART_Receive();
                }
                if(result[0]==1 && result[1]==1){
                    unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
                    send_string(data, sizeof(data));
                    USART_Receive();
                    unsigned char text[] = {0x00, 0x06, '1', 0x00};
                    send_string(text, sizeof(text));
                    USART_Receive();
                    unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
                    unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
                    send_string(move, sizeof(move));
                    USART_Receive();
                    send_string(data2, sizeof(data2));
                    USART_Receive();
                    unsigned char text2[] = {0x00, 0x06, '1', 0x00};
                    send_string(text2, sizeof(text2));
                    USART_Receive();
                }
                if(result[0]==1 && result[1]==2){
                    unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
                    send_string(data, sizeof(data));
                    USART_Receive();
                    unsigned char text[] = {0x00, 0x06, '1', 0x00};
                    send_string(text, sizeof(text));
                    USART_Receive();
                    unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
```

```
                unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
                send_string(move, sizeof(move));
                USART_Receive();
                send_string(data2, sizeof(data2));
                USART_Receive();
                unsigned char text2[] = {0x00, 0x06, '2', 0x00};
                send_string(text2, sizeof(text2));
                USART_Receive();
            }
            if(result[0]==2 && result[1]==0){
                unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
                send_string(data, sizeof(data));
                USART_Receive();
                unsigned char text[] = {0x00, 0x06, '2', 0x00};
                send_string(text, sizeof(text));
                USART_Receive();
                unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
                unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
                send_string(move, sizeof(move));
                USART_Receive();
                send_string(data2, sizeof(data2));
                USART_Receive();
                unsigned char text2[] = {0x00, 0x06, '0', 0x00};
                send_string(text2, sizeof(text2));
                USART_Receive();
            }
            if(result[0]==2 && result[1]==1){
                unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
                send_string(data, sizeof(data));
                USART_Receive();
                unsigned char text[] = {0x00, 0x06, '2', 0x00};
                send_string(text, sizeof(text));
                USART_Receive();
                unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
                unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
                send_string(move, sizeof(move));
                USART_Receive();
                send_string(data2, sizeof(data2));
                USART_Receive();
```

```
                unsigned char text2[] = {0x00, 0x06, '1', 0x00};
                send_string(text2, sizeof(text2));
                USART_Receive();
        }
        if(result[0]==2 && result[1]==2){
                unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
                send_string(data, sizeof(data));
                USART_Receive();
                unsigned char text[] = {0x00, 0x06, '2', 0x00};
                send_string(text, sizeof(text));
                USART_Receive();
                unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
                unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
                send_string(move, sizeof(move));
                USART_Receive();
                send_string(data2, sizeof(data2));
                USART_Receive();
                unsigned char text2[] = {0x00, 0x06, '2', 0x00};
                send_string(text2, sizeof(text2));
                USART_Receive();
        }
        if(result[0]==3 && result[1]==0){
                unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
                send_string(data, sizeof(data));
                USART_Receive();
                unsigned char text[] = {0x00, 0x06, '3', 0x00};
                send_string(text, sizeof(text));
                USART_Receive();
                unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
                unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
                send_string(move, sizeof(move));
                USART_Receive();
                send_string(data2, sizeof(data2));
                USART_Receive();
                unsigned char text2[] = {0x00, 0x06, '0', 0x00};
                send_string(text2, sizeof(text2));
                USART_Receive();
        }
        if(result[0]==3 && result[1]==1){
```

```
                                        unsigned char data[] = {0xFF, 0x7F, 0xFF, 0xFF};
//Change foreground color
                                        send_string(data, sizeof(data));
                                        USART_Receive();
                                        unsigned char text[] = {0x00, 0x06, '1', 0x00};
                                        send_string(text, sizeof(text));
                                        USART_Receive();
                                        unsigned char data2[] = {0xFF, 0x7F, 0xF8, 0x00};
//Change foreground color
                                        unsigned char move[] = {0xff, 0xe4, 0x00, help, 0x00,
0x11};
                                        send_string(move, sizeof(move));
                                        USART_Receive();
                                        send_string(data2, sizeof(data2));
                                        USART_Receive();
                                        unsigned char text2[] = {0x00, 0x06, '2', 0x00};
                                        send_string(text2, sizeof(text2));
                                        USART_Receive();
                                }
                                //if(result[0]==3 && result[1]==2){} //result[0] can never equal 3
if result[1] equals 2.
                        }
                }

                ////////////////////////////////////////////////////////

                unsigned char data2[] = {0xFF, 0xE4, 0x00, 0x00, 0x00, 0x00}; //Move Cursor
                send_string(data2, sizeof(data2));
                USART_Receive();

                unsigned char data3[] = {0x00, 0x06, 1, 0x00}; //Put character
                send_string(data3, sizeof(data3));
                USART_Receive();

                for(int i=0; i<3; i++){
                        int xpos = i*20+10;

                        if(guessSequence[i]==1){colorMSB = 0xFF; colorLSB = 0xDF;}
                        if(guessSequence[i]==2){colorMSB = 0x04; colorLSB = 0x00;}
                        if(guessSequence[i]==3){colorMSB = 0xF8; colorLSB = 0x00;}
                        if(guessSequence[i]==4){colorMSB = 0x00; colorLSB = 0x1F;}
                        if(guessSequence[i]==5){colorMSB = 0xF8; colorLSB = 0x1F;}
                        if(guessSequence[i]==0){colorMSB = 0xFF; colorLSB = 0xE0;}

                        if(guessNbr==1){
                                        //cmd       x         y          rad       color
```

```
                        unsigned char data[] = {0xFF, 0xCC, 0x00, xpos, 0x00, 0x05, 0x00,
0x05, colorMSB, colorLSB}; //Draw circle
                        send_string(data, sizeof(data));
                        USART_Receive();

                        }
                        if(guessNbr==2){
                                        //cmd     x       y       rad     color
                        unsigned char data[] = {0xFF, 0xCC, 0x00, xpos, 0x00, 0x20, 0x00,
0x05, colorMSB, colorLSB}; //Draw circle
                        send_string(data, sizeof(data));
                        USART_Receive();
                        }
                        if(guessNbr==3){
                                        //cmd     x       y       rad     color
                        unsigned char data[] = {0xFF, 0xCC, 0x00, xpos, 0x00, 0x35, 0x00,
0x05, colorMSB, colorLSB}; //Draw circle
                        send_string(data, sizeof(data));
                        USART_Receive();
                        }

                        if(guessNbr==4){
                                        //cmd     x       y       rad     color
                        unsigned char data[] = {0xFF, 0xCC, 0x00, xpos, 0x00, 0x50, 0x00,
0x05, colorMSB, colorLSB}; //Draw circle
                        send_string(data, sizeof(data));
                        USART_Receive();
                        }

                        if(guessNbr==5){
                                        //cmd     x       y       rad     color
                        unsigned char data[] = {0xFF, 0xCC, 0x00, xpos, 0x00, 0x65, 0x00,
0x05, colorMSB, colorLSB}; //Draw circle
                        send_string(data, sizeof(data));
                        USART_Receive();
                        }
                }
}

void resetResult(){
        result[0]=0;
        result[1]=0;
}

void winDisplay(){
        USART_Transmit(0xFF); //Clear
```

```
        USART_Transmit(0xD7);
        USART_Receive();
        unsigned char text_width[] = {0xff, 0x7c, 0x00, 0x03};
   unsigned char text_height[] = {0xff, 0x7b, 0x00, 0x03};
        send_string(text_width, sizeof(text_width));
        USART_Receive();
        send_string(text_height, sizeof(text_height));
        USART_Receive();
        unsigned char move[] = {0xff, 0xe4, 0x00, 0x00, 0x00, 0x00};
        send_string(move, sizeof(move));
        USART_Receive();
        unsigned char data[] = {0x00, 0x06, 'W', 'I', 'N', 0x00};
        send_string(data, sizeof(data));
        USART_Receive();
        win = 1;

}

void loseDisplay(){
        USART_Transmit(0xFF); //Clear
        USART_Transmit(0xD7);
        USART_Receive();
        unsigned char text_width[] = {0xff, 0x7c, 0x00, 0x03};
        unsigned char text_height[] = {0xff, 0x7b, 0x00, 0x03};
        send_string(text_width, sizeof(text_width));
        USART_Receive();
        send_string(text_height, sizeof(text_height));
        USART_Receive();
        unsigned char move[] = {0xff, 0xe4, 0x00, 0x00, 0x00, 0x00};
        send_string(move, sizeof(move));
        USART_Receive();
        unsigned char data[] = {0x00, 0x06, 'L', 'O', 'S', 'E', 0x00};
        send_string(data, sizeof(data));
        USART_Receive();
}

int main(void)
{
        srand(time(NULL));
        _delay_ms(4000);
        initMCU();
        sei();
        USART_Init();
        USART_Transmit(0xFF); //Clear
        USART_Transmit(0xD7);
        USART_Receive();
```

```
        USART_Transmit(0x00);
        USART_Transmit(0x0C);
        USART_Transmit(0x00);
        USART_Transmit(0x00);
        USART_Receive();
        resetResult();
        sequenceGenerator();

    /* Replace with your application code */
        guessNbr = 0;
        correctColors = 0;
        correctColorAndPlace = 0;
        drawSequence();

        while(1){}

}

ISR(PCINT0_vect) {
        _delay_ms(100);
        int read = PINA;

        if(read == 8){
                guessSequence[0] = (guessSequence[0] + 1) % 6;
                }else if(read == 4){
                guessSequence[1] = (guessSequence[1] + 1) % 6;

                }else if(read == 2){
                guessSequence[2] = (guessSequence[2] + 1) % 6;

                }else if(read==1){
                        guessNbr++;
                        checkSequence();
                        if(guessNbr<5 && win == 0){
                        drawResponseSequence();
                        }
        }
        if(guessNbr<5){drawSequence();}
}
```