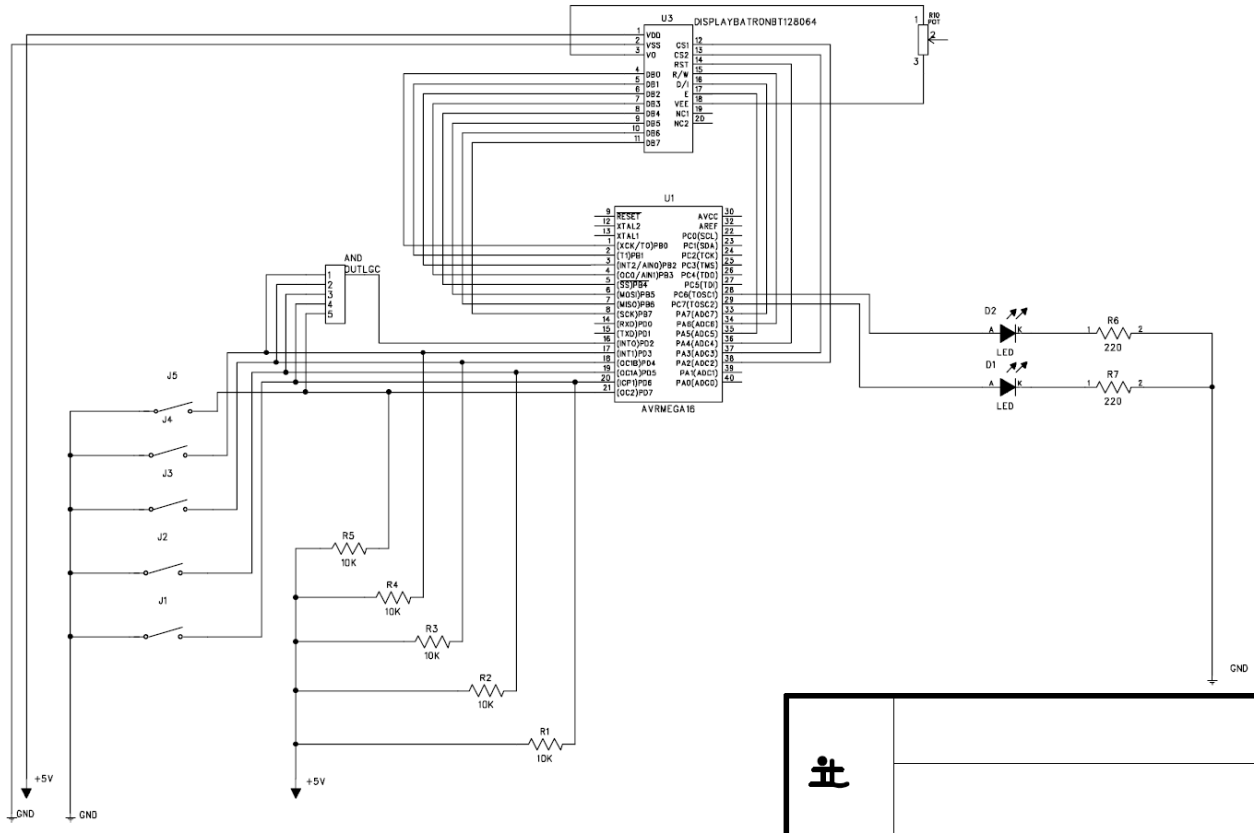
A Circuit diagram



B Source code

```
#include <avr/io.h>
#include <stdint.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <string.h>

// LCD <=> AVR connections:
//
// D:   PORTB
// D/I: PA7
// R/W: PA6
// E:   PA5
// RS:  PA4
// CS2: PA3
// CS1: PA2
//


#include <inttypes.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
```

```
#define Y_MIN 0x40
#define Y_MAX 0x3F
#define _NOP()  asm volatile("nop\n");
#define X_MIN 0xB8
#define X_MAX 0xBF

unsigned short int xPos;
unsigned short int yPos;
unsigned int allMarkers[8][8];
unsigned int aPos;
unsigned int bPos;
unsigned short int drawX = 0xB8;
unsigned short int drawY = 0x42;
unsigned short int player;
unsigned short int skipped;
unsigned short int score1;
unsigned short int score2;

void wait()
{
        int i = 0;
        while(i < 32)
        {
                i++;
        }
}

void toggle()
{
        PORTA = PORTA | 0x20; //Or med "Toggle hög" (1 0 0 0 0 0 0 0 0) => E hög
        PORTA = PORTA & 0xDF; //And med inverterad "Toggle hög" (0 1 1 1 1 1 1 1 1) => E låg //7F
        PORTA = PORTA | 0x20; //Or med "Toggle hög" (Se första steget) => E hög
}

void displayOn()
{
        PORTB = 0x3F;
        PORTA = 0x3C; //9C
        toggle();
}

void displayOff()
{
        PORTB = 0x3E;
        PORTA = 0x3C; //9C
        toggle();
}

void convertGridToCord(int a,int b) {// konverterar en koordinat till pixelkoordinater
        drawX = 0xB8+(0x01*a);
        drawY = 0x42+(0x08*b);
}

void convertCordToGrid(int x, int y){
        aPos = x - 0xB8;
        bPos = (y - 0x42)/0x08;
}

void clearDisplay()
{
        for(int k = 0xB8; k <= (0xB8 + 0x07); k++)
        {
        PORTA = 0x3C; //Markera båda skärmhalvorna //var 9C
        PORTB = k; //Markera den aktuella byten
        wait();
```

```
        toggle();
        PORTB = 0x40; //Markera längst till höger
        wait();
        toggle();
        PORTA = 0xBC; //Markera för utskrift //var DC
        PORTB = 0x00; //Rensa markerade pixlar
        for(int i = 0; i < 64; i++)
                {
        //      wait();
                toggle();
                }
        }
}

void clearScoreDisplay()
{
        for(int k = 0; k <= 16; k++)
        {
        PORTA = 0x38; //Markera båda skärmhalvorna //var 9C
        PORTB = k; //Markera den aktuella byten
        wait();
        toggle();
        PORTB = 0x40; //Markera längst till höger
        wait();
        toggle();
        PORTA = 0xB8; //Markera för utskrift //var DC
        PORTB = 0x00; //Rensa markerade pixlar
        for(int i = 0; i < 64; i++)
                {
        //      wait();
                toggle();
                }
        }
}



void write(int x, int y, int data, int displayScreen)
{
        if(displayScreen == 1)
        {
                PORTA = 0x34; //34
        } else if (displayScreen == 2) {
                PORTA = 0x38; //98
        }
        //wait();
        //toggle();
        PORTB = y; //y
        wait();
        toggle();
        PORTB = x ; //x
        wait();
        toggle();

        if(displayScreen == 1)
        {
                PORTA = 0xB4; //D4
        } else if (displayScreen == 2){
                PORTA = 0xB8; //D8
        }
        PORTB = data;
//      wait();
//      toggle();

        PORTA = 0x30; //90
```

```
        wait();
        toggle();
}

void drawCursor() {
        int x=xPos;
        int y=yPos;
        for(int i=0x00;i<=0x06;i+=0x01){
                if(i == 0x00 || i == 0x06)
                        write(x,y-0x01+i,0xFF,1);
        }



}
void drawBlackMarker(int x, int y) {

        for (int i = 0; i < 5; i++) {
                write(x,y+i,0xBE,1);
        }
}

void drawWhiteMarker(int x, int y) {

        for(int i=0x00;i<=0x04;i+=0x01){
                //write(x,y,0xBE,1); Svart bricka (Y)
                if(i == 0x00 || i == 0x04){
                        write(x,y+i,0xBE,1);
                } else {
                        write(x,y+i,0xA2,1);
                }
        }
}


void drawChar(int x, int y, int num) {
        switch(num) {
                case 0:
                        write(x,y,0x00, 2);
                        write(x,y,0x7C, 2);
                        write(x,y,0x8A, 2);
                        write(x,y,0x92, 2);
                        write(x,y,0xA2, 2);
                        write(x,y,0x7C, 2);
                        break;
                case 1:
                        write(x,y,0x00, 2);
                        write(x,y,0x00, 2);
                        write(x,y,0xFE, 2);
                        write(x,y,0x40, 2);
                        write(x,y,0x20, 2);
                        write(x,y,0x00, 2);
                        break;
                case 2:
                        write(x,y,0x00, 2);
                        write(x,y,0x62, 2);
                        write(x,y,0x92, 2);
                        write(x,y,0x8A, 2);
                        write(x,y,0x86, 2);
                        write(x,y,0x42, 2);
                        break;
                case 3:
                        write(x,y,0x00, 2);
                        write(x,y,0x6C, 2);
                        write(x,y,0x92, 2);
```

```
                write(x,y,0x92, 2);
                write(x,y,0x82, 2);
                write(x,y,0x44, 2);
                break;
        case 4:
                write(x,y,0x00, 2);
                write(x,y,0x08, 2);
                write(x,y,0xFE, 2);
                write(x,y,0x48, 2);
                write(x,y,0x28, 2);
                write(x,y,0x18, 2);
                break;
        case 5:
                write(x,y,0x00, 2);
                write(x,y,0x9C, 2);
                write(x,y,0xA2, 2);
                write(x,y,0xA2, 2);
                write(x,y,0xA2, 2);
                write(x,y,0xE4, 2);
                break;
        case 6:
                write(x,y,0x00, 2);
                write(x,y,0x4C, 2);
                write(x,y,0x92, 2);
                write(x,y,0x92, 2);
                write(x,y,0x92, 2);
                write(x,y,0x7C, 2);
                break;
        case 7:
                write(x,y,0x00, 2);
                write(x,y,0xE0, 2);
                write(x,y,0x90, 2);
                write(x,y,0x8E, 2);
                write(x,y,0x80, 2);
                write(x,y,0x80, 2);
                break;
        case 8:
                write(x,y,0x00, 2);
                write(x,y,0x6C, 2);
                write(x,y,0x92, 2);
                write(x,y,0x92, 2);
                write(x,y,0x92, 2);
                write(x,y,0x6C, 2);
                break;
        case 9:
                write(x,y,0x00, 2);
                write(x,y,0x7C, 2);
                write(x,y,0x92, 2);
                write(x,y,0x92, 2);
                write(x,y,0x92, 2);
                write(x,y,0x64, 2);
                break;
        case 10:        //10 = B
                write(x,y,0x00, 2);
                write(x,y,0x6C, 2);
                write(x,y,0x92, 2);
                write(x,y,0x92, 2);
                write(x,y,0x92, 2);
                write(x,y,0xFE, 2);
                break;

        case 11:        //11 = W
                write(x,y,0x00, 2);
                write(x,y,0xFE, 2);
                write(x,y,0x04, 2);
```

```
                        write(x,y,0x18, 2);
                        write(x,y,0x04, 2);
                        write(x,y,0xFE, 2);
                        break;

                case 12:        //12 = Colon-sign
                        write(x,y,0x00, 2);
                        write(x,y,0x00, 2);
                        write(x,y,0x00, 2);
                        write(x,y,0x6C, 2); //6C
                        write(x,y,0x6C, 2); //6C
                        write(x,y,0x00, 2);
                        break;

                case 13:        //13 = Space
                        write(x,y,0x00, 2);
                        write(x,y,0x00, 2);
                        write(x,y,0x00, 2);
                        write(x,y,0x00, 2);
                        write(x,y,0x00, 2);
                        write(x,y,0x00, 2);
                        break;

                case 14:        //14 = | marker (Active player)
                        write(x,y,0x00, 2);
                        write(x,y,0x7C, 2);
                        break;

                case 15:        //15 = small-space (Not active player)
                        write(x,y,0x00, 2);
                        write(x,y,0x00, 2);
                        break;
                }



/*
                //Conversion chart to flip char-segments
                        3E -> 7C
                        42 -> 42
                        61 -> 86
                        51 -> 8A
                        49 -> 92
                        46 -> 62
                        00 -> 00
                        04 -> 20
                        02 -> 40
                        7F -> FE
                        18 -> 18
                        14 -> 28
                        12 -> 48
                        36 -> 6C
                        41 -> 82
                        22 -> 44
                        39 -> 9C
                        45 -> A2
                        27 -> E4
                        32 -> 4C
                        07 -> E0
                        09 -> 90
                        01 -> 80
                        71 -> 8E
                        26 -> 64
                        10 -> 80
*/
```

```
}

void drawScore() {
        clearScoreDisplay();
        int b1 = score1/10;
        int b2 = score1%10;
        int w1 = score2/10;
        int w2 = score2%10;

        //PLAYER ONE TEXT//////////////////////

        if (player==1) {
                drawChar(16,0,14); //|
                } else {
                drawChar(16,0,15); //small-space
        }

        drawChar(16,0,b2);
        if (b1 >= 1) {
                drawChar(16,0,b1);
                } else {
                drawChar(16,0,13); //space
        }

        drawChar(16,0,12); //colon-sign
        drawChar(16,0,10); //B

        if (player==1) {
                drawChar(16,0,14); //|
                } else {
                drawChar(16,0,15); //small-space
        }
        ////////////////////////////////////////

        drawChar(16,0,13); //Space

        //PLAYER TWO TEXT//////////////////////

        if (player==2) {
                drawChar(16,0,14); //active-marker
                } else {
                drawChar(16,0,15); //small-space
        }


        drawChar(16,0,w2);
        if (w1 >= 1) {
                drawChar(16,0,w1);
                } else {
                drawChar(16,0,13); //space
        }
        drawChar(16,0,12); //colon-sign
        drawChar(16,0,11); //W
        if (player==2) {
                drawChar(16,0,14); //active-marker
                } else {
                drawChar(16,-7,15); //small-space
        }
        ////////////////////////////////////////

}


void drawAllMarkers() {
```

```
        score1 = 0;
        score2 = 0;
        for (int k = 0; k <= 7; k++) {
                for (int l = 0; l <= 7; l++) {

                        if (allMarkers[k][l] == 1) {
                                convertGridToCord(k,l);
                                score1++;
                                drawBlackMarker(drawX, drawY);
                                //      converter(k,l);
                        //      for (int i = 0; i < 5; i++) {
                                        //write(k,l+i,0xBE,1);
                        //              write(drawX,drawY-i,0xBE,1);
                                        //}
                        }
                        else if (allMarkers[k][l] == 2) {
                                convertGridToCord(k,l);
                                score2++;
                                drawWhiteMarker(drawX,drawY);
                        //      for (int i = 0; i< 5; i++) {
                        //              if (i == 0 || i == 4) {
                        //                      write(k,l-1, 0xFF, 1);
                        //              } else {
                        //                      write(k,l-1, 0xC1, 1);
                        //              }
                        //      }
        //              }
                        }
                }
        }
}


void redraw()
{
        //clearDisplay();
        int i;

        for (i=X_MIN; i <= X_MAX ; i+=0x01) {

                int y = Y_MIN;
                int j;
                for (j = 0; j<64; j++){
                        if((j % 8) == 0){
                                write(i, y, 0xFF, 1);
                        } else {
                                write(i, y, 0x80, 1);
                        }
                        y += 0x01;
                }

        }
        drawAllMarkers();
        drawCursor(xPos,yPos);

//      PORTC=0x40;

}

void switchPlayer(){
        if(player == 1){
                player = 2;
        //      PORTC =0x80;
                PORTC=(0xC0 ^ PORTC);//SPEAKERTEST
        } else {
```

```
                player = 1;
                //PORTC =0x40;
                PORTC=(0xC0 ^ PORTC);//SPEAKERTEST
        }
}

int movePossible() {
        int opponent = (player == 1)? 2 : 1;
        for (int a = 0 ; a <= 7 ; a++) {
                for (int b  = 0 ; b <= 7 ; b++) {
                        if (allMarkers[a][b] == 0) {
                                int deltaA;
                                int deltaB;
                                for (deltaA = -1; deltaA <= 1; deltaA++) {
                                        for(deltaB = -1; deltaB <= 1; deltaB++){

                                                if (a+deltaA > 7 || a+deltaA < 0 ||  b+deltaB > 7 ||
b+deltaB < 0 || (deltaA == 0 && deltaB == 0)){
                                                        continue;
                                                }

                                                if (allMarkers[a+deltaA][b+deltaB] == opponent){

                                                        int nextA = a+deltaA;
                                                        int nextB = b+deltaB;

                                                        for(;;){

                                                                nextA += deltaA;
                                                                nextB += deltaB;

                                                                if(nextA > 7 || nextA < 0 || nextB >
7 || nextB < 0){
                                                                        break;
                                                                        // Out-of-bounds, not valid
                                                                }

                                                                if(allMarkers[nextA][nextB] == 0){
                                                                        break;
                                                                        // Empty, not valid
                                                                }
                                                                if(allMarkers[nextA][nextB] ==
player){
                                                                        return 1;
                                                                        // Valid move found
                                                                }
                                                                break;
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }
        return 0;
}



void placeBrick() { //kollar om draget är giltligt och placerar då ut bricka
        int movesPerformed = 0;
        int opponent = (player == 1)? 2 : 1;
        int a = aPos;
        int b = bPos;
        if (allMarkers[a][b] == 0) {
```

```
                int deltaA;
                int deltaB;
                for (deltaA = -1; deltaA <= 1; deltaA++) {
                        for(deltaB = -1; deltaB <= 1; deltaB++){


                                if (a+deltaA > 7 || a+deltaA < 0 ||  b+deltaB > 7 || b+deltaB < 0 ||
(deltaA == 0 && deltaB == 0)){
                                        continue;
                                }


                                if (allMarkers[a+deltaA][b+deltaB] == opponent){

                                        int nextA = a+deltaA;
                                        int nextB = b+deltaB;

                                        for(;;){

                                                nextA += deltaA;
                                                nextB += deltaB;

                                                if(nextA > 7 || nextA < 0 || nextB > 7 || nextB < 0){
                                                        break;
                                                        // Out-of-bounds, not valid
                                                }

                                                if(allMarkers[nextA][nextB] == 0){
                                                        break;
                                                        // Empty, not valid
                                                }
                                                if(allMarkers[nextA][nextB] == player){

while(allMarkers[nextA-=deltaA][nextB-=deltaB] == opponent){
                                                                        allMarkers[nextA][nextB] = player;
                                                                        movesPerformed++;
                                                        }
                                                        break;
                                                        // Valid move found, traverses back and flips
the disks
                                                }



                                        }
                                }
                        }
                }
        }

        if(movesPerformed != 0){
                allMarkers[aPos][bPos] = player;
                skipped = 0;
                switchPlayer();
        }
        drawAllMarkers();
}


void removeCursor() {

        int x = xPos;
        int y = yPos;
        for(int i = 0x00; i <= 0x06; i+= 0x01) {
                if (i == 0x00 || i == 0x06) {
```

```
                        write(x,y-0x01+i,0x80,1);
                }
        }

}

void newGame() {
        xPos = 0xBC;
        yPos = 0x5A;
        aPos = 4;
        bPos = 3;

        memset(allMarkers,0, sizeof allMarkers);//resets allMarkers to zeros
        // Wait a little while the display starts up
        wait();


        //      SREG = 0xFF;

        //PORTC=0x40; //LED GREEN

        PORTC=(0x40 | PORTC);//SPEAKERTEST

        displayOn();
        clearDisplay();
        allMarkers[3][4] = 1;
        allMarkers[4][3] = 1;
        allMarkers[3][3] = 2;
        allMarkers[4][4] = 2;
        player = 1;
        skipped = 0;
/*      allMarkers[5][4] = 1;
        allMarkers[5][3] = 1;
        allMarkers[6][3] = 1;
        allMarkers[6][4] = 1;
        allMarkers[2][3] = 1;
        allMarkers[2][3] = 1;
        allMarkers[2][4] = 1;

*/
        //drawAllMarkers();
        redraw();
}

void checkWinner() {
        if (score1 > score2) {
                for(int i = 0 ; i < 5 ; i++) {
                        PORTC=0x00;
                        _delay_ms(300);
                        PORTC=0x40;
                        _delay_ms(300);
                }
        } else if (score2 > score1) {
                for(int i = 0 ; i < 5 ; i++) {
                        PORTC=0x00;
                        _delay_ms(300);
                        PORTC=0x80;
                        _delay_ms(300);
                }
        } else {
                for(int i = 0 ; i < 5 ; i++) {
                        PORTC=0x00;
                        _delay_ms(300);
                        PORTC=0xC0;
                        _delay_ms(300);
```

```
                }
        }

}

//void cursor(left,right,up,down) {}

ISR(INT0_vect) {

        unsigned short int input = PIND;

        _delay_ms(150);
        _NOP();
        switch(input) {//enter FUNKAR
                case 0xBB:
                        //PORTC = (0x01 | PORTC);
                        placeBrick();
                        drawScore();
                        if(!movePossible()) { //kolla nu movePossible för nästa spelare
                                //skipped++;
                                //if (skipped == 2) {
                                //      PORTC=0xC0;
                        //              _delay_ms(3000);
                        //              checkWinner();
                        //                      newGame();
                //                      } else {
                                                PORTC=0xC0;
                                                _delay_ms(1000);
                                                switchPlayer();
                                                if(!movePossible()){
                                                        PORTC=0xC0;
                                                        _delay_ms(3000);
                                                        checkWinner();
                                                        newGame();
                                                }
                                                drawScore();

                        }
                        break;

                case 0xF3://right Funkar
                        if (yPos <= 0x42) {
                        break;
                        }
                        else
                                removeCursor();
                                yPos-= 0x08;
                                bPos--;
                                drawCursor();
                                //redraw();
                                //cursor(1,0,0,0);
                        break;

                case 0xDB://left FUNKAR
                        if (yPos >= 0x7A) {
                                break;
                        }
                        else
                                removeCursor();
                                yPos+=0x08;
                                bPos++;
                                drawCursor();
                                //redraw();
                        //cursor(0,0,0,1):
                        break;
```

```
                case 0xEB://down FUNKAR
                        if (xPos <= 0xB8) {
                                break;
                        }
                        else
                                removeCursor();
                                xPos -= 0x01;
                                aPos--;
                        //      redraw();
                                drawCursor();
                        //cursor(0,1,0,0);
                        break;

                case 0x7B://up
                        if (xPos >= 0xBF) {
                                break;
                        }
                        else
                                removeCursor();
                                xPos += 0x01;
                                aPos++;
                        //      redraw();
                                drawCursor();
                        //cursor(0,0,1,0);
                        break;
        }

}


int main(void) {
        //INIT
        DDRA = 0xFF;
        DDRB = 0xFF;

        DDRD = 0x00;
        DDRC=0xFF;

        GICR = 0x40;
        MCUCR = 0x02;
        newGame();
        drawScore();

        sei();
        while(1);

}
```