Snake

EITF11 Digitala projekt

Carl Bergman, I12
Johanna Öjeling, I12

Lunds Tekniska Högskola
Institutionen för elektro- och informationsteknik
Handledare: Andreas Johansson, Bertil Lindvall

2015-05-12

Sammanfattning

Den här rapporten behandlar ett projekt som har genomförts i kursen EITF11 Digitala projekt på Lunds tekniska högskola under vårterminen 2015. Studenterna har haft i uppgift att ta fram en fungerande prototyp av valfritt slag och svara för såväl konstruktion som programmering. Den här gruppen har skapat en spelkonsol för spelet Snake och i rapporten beskrivs arbetsprocessen följt av en diskussion kring denna.

Innehållsförteckning

1. Inledning

1.1 Bakgrund

Inom ramen för teknikprofilen System- och programvaruutveckling på civilingenjörsprogrammet i industriell ekonomi vid Lunds tekniska högskola genomförs under vårterminen 2015 kursen EITF11 Digitala projekt, 10 hp.

Syftet med kursen är att illustrera industriellt utvecklingsarbete genom att låta studenterna designa, bygga, testa och felsöka en konstruktion.

Kursen är indelad i två delar, där den första består av föreläsningar och laborationer och den andra av självständigt arbete som ska resultera i en fungerande prototyp.

1.2 Problemformulering

Med beaktande av förkunskaper och intressen faller valet av projekt på att skapa en spelkonsol för spelet Snake.

För att konstruera konsolen behövs en mikroprocessor, en grafisk LCD-skärm, en varistor, tre knappar, tre resistorer, en diod och en logisk grind.

Programmet skrivs i språket C och en JTAG används för att låta mjukvaran kommunicera med hårdvaran.

2. Kravspecifikation

Denna kravspecifikation behandlar konstruktionen av hårdvara och mjukvara för spelet Snake.

2.1 Funktionella krav

1. Systemet ska ha en display.
2. Displayen ska visa en spelplan och en poängräknare.
3. Systemet ska ha en vänsterknapp och en högerknapp.
    1. Vid knapptryckning ska ormen svänga åt det håll knappen representerar.
4. Systemet ska ha en resetknapp.

5. Systemet ska ha en logisk del som tolkar knapptryckningar och visar resultat på skärmen.
6. Systemet ska ha en knapp för att starta om spelet.
    1. Vid knapptryckning ska spelet starta om.

2.2 Kvalitetskrav

1. Displayen ska vara uppbyggd av 128x64 pixlar.
2. Systemet ska ha en tillfredsställande fördröjning mellan knapptryck och uppdatering på skärmen.
3. Systemet ska följa gängse regler för spelet snake.
    1. Ormen ska röra sig framåt hela tiden.
    2. Om ormen kör in i sig själv eller i en vägg ska spelet avslutas.
    3. Om ormen kör in i ett äpple ska spelaren få ett poäng.
    4. Om ormen kör in i ett äpple ska den blir en ruta längre.
    5. Om det inte finns något äpple på spelplanen ska ett äpple läggas till i en tom ruta.

3. Hårdvara

För kopplingsschema, se bilaga 1.

3.1 Processor

Mikroprocessorn AVR AT Mega32 utgör kärnan i konstruktionen. Det inbyggda emuleringsminnet på 32 kB och de 40 pinnarna möjliggör lagring av program respektive kommunikation med ytterligare komponenter.

3.2 Grafisk LCD-skärm

Som display används den enfärgade bakgrundsbelysta LCD-skärmen GDM12864HLCM med 128x64 pixlar. Den består av två separata skärmar på 64x64 pixlar vardera, vilka tillsammans utgör hela displayen.

3.3 Varistor

En variabel resistor används för att justera LCD-skärmens ljusstyrka.

3.4 Knappar

Styrenheter utgörs av tre knappar. Den svarta knappen används för att starta om spelet och de två röda knapparna för att svänga ormen 90 grader åt höger respektive vänster.

3.5 Resistorer

Tre resistorer kopplade till en knapp vardera begränsar strömtillförseln till knapparna då de hålls nedtryckta.

3.6 Dioder

En diod används till att begränsa strömtillförseln till displayen.

3.7 Logisk grind

En logisk OR-grind, vars ingångar utgörs av signaler från de tre knapparna, används till att initiera ett avbrott hos processorn då någon av knapparna trycks ned.

3.8 Mönsterkort

Ett mönsterkort används för montering av samtliga delar samt sammankoppling av dem.

3.9 Strömförsörjning

Till kretsen ansluts en strömkälla på 5 V och 5 A.

4. Mjukvara

Programmet skrivs i språket C i utvecklingsplattformen Atmel Studio 6.1. JTAG används för att ladda och exekvera mjukvaran i processorn samt för att felsöka och testa programmet.

Koden innehåller ett mainprogram som startar en spelomgång. Bland annat finns metoder för att rita ut och radera pixlar på displayen, för att hantera knapptryckningar, med mera. Den intresserade läsaren hänvisas till den bifogade källkoden, se bilaga 2.

5. Utförande

5.1 Planering

I ett första skede utformades problemformulering och kravspecifikation. Med hänsyn till dessa bestämdes de ingående komponenterna och ett kopplingsschema ritades upp.

Under framtagning av kopplingsschemat konsulterades såväl datablad som handledare. Schemat fick under arbetsprocessens gång även justeras till följd av nya upptäckter.

En skiss på hur komponenterna skulle placeras på mönsterkortet togs fram med användarvänligheten för spelaren i åtanke.

5.2 Konstruktion

Efter slutförd planering monterades komponenterna på mönsterkortet och kopplades samman genom lödning och virning av sladdar. Kommunikationen mellan komponenterna och processorn testades med JTAG. Testningen visade att vissa mindre modifieringar, såsom montering av nya komponenter och omkopplingar, behövde göras.

5.3 Programmering

Inledningsvis skapades funktioner för att ge basala instruktioner till LCD-skärmen, såsom skriva ut eller sudda ut en pixel.

Därefter togs algoritmer fram för initiering av en orm samt hur denna skulle bete sig vid användarens knapptryckningar. Begränsningar infördes sedan med hänsyn till de gängse reglerna för spelet Snake.

Slutligen utvecklades spelets layout. Själva spelplanen finns representerad på höger skärmhalva, medan spelets titel Snake och spelarens aktuella poäng framgår på den vänstra skärmhalvan.

6. Resultat

Arbetet mynnade som önskat ut i en fungerande spelkonsol för spelet Snake.

Den uppskattade tidsåtgången stämde ganska väl överens med den faktiska. Det som visade sig vara den största utmaningen var att tyda databladen på ett korrekt sätt samt lista ut hur mjukvaran skulle utformas för att kommunicera med hårdvaran. En spännande utmaning med

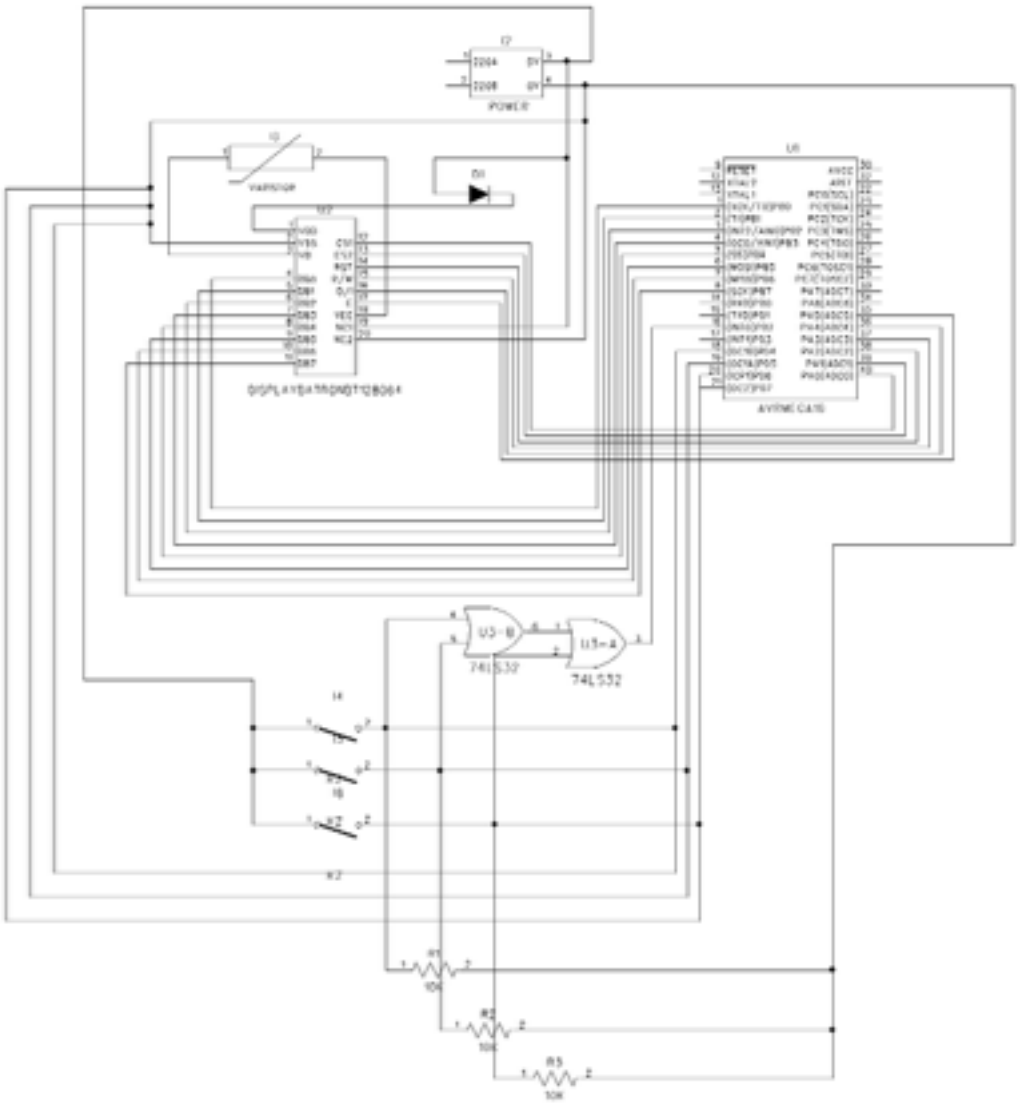
C var även att flera funktioner som vi är vana vid från exempelvis Java saknades. Det har gjort att vi har behövt lära oss mer om bland annat minneshantering, vilket har varit lärorikt.

7. Diskussion och slutsats

Det var intressant att genomföra projektet, då det illustrerade produktutveckling i miniatyr. Projektet gav insikter i hur komplext arbetet med att få hårdvara och mjukvara att samverka är.

8. Bilagor

Bilaga 1 - Kopplingsschema



Bilaga 2 - Källkod

```
/*
 * snake.c
 *
 * Created: 2015-04-07 09:58:47
 * Author: Johanna Öjeling and Carl Bergman
 */

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


char display[8][128];

int points = 0;
int apple[2];
int alive = 1;

// Snake
int length = 3;
int snake[100][2];
int dir = 2; // 1=N, 2=E, 3=S, 4=W

// Other variables
int click = 0;

void main(void)
{
	setDataDirection();
	resetHigh();
	startDisplay();
	startGame();

	play();

	while(1){}
}
```

```
void play() {
	clearDisplay();
	setupSnake();
	createApple();
	setupBoard();
	writeSnake();
	writeScore();

	while(alive) {
		move();
		_delay_ms(500);
	}
}

void restart() {
	length = 3;
	dir = 2;
	points = 0;
	alive = 1;
	play();
}

void move() {

	int newPos[2];
	newPos[0] = getNextX();
	newPos[1] = getNextY();
	if (newPos[0] == apple[0] && newPos[1] == apple[1]) {

		// förläng ormen
		length++;
		for (int i = length - 1; i > 0; i--) {
			snake[i][0] = snake[i-1][0];
			snake[i][1] = snake[i-1][1];
		}

		// Set the head of the snake
		snake[0][0] = newPos[0];
		snake[0][1] = newPos[1];
```

```
            createApple();

            points++;
            writeScore();

      } else if (newPos[0] > 62 || newPos[0] < 1 || newPos[1] > 62 ||
newPos[1] < 1 || selfCollision() == 1) {
            alive = 0;
      } else {

            // Save last position in snake temporarily
            int last[2];
            last[0] = snake[length - 1][0];
            last[1] = snake[length - 1][1];

            // Move snake forward (start from the rear)
            for (int i = length - 1; i > 0; i--) {
                  snake[i][0] = snake[i-1][0];
                  snake[i][1] = snake[i-1][1];
            }

            // Set the head of the snake
            snake[0][0] = newPos[0];
            snake[0][1] = newPos[1];

            // Erase the snake at the position that the snake
            // just left.
            erase(last[0], last[1]);
      }

      drawSnake();
}

int selfCollision() {

      int head[2];
      head[0] = snake[0][0];
      head[1] = snake[0][1];

      for (int i = 1; i < length; i++) {
            if (head[0] == snake[i][0] && head[1] == snake[i][1]) {
```

```
                return 1;
            }
        }

        return 0;
}

void createApple() {
        int newApple[2];

        do {
            newApple[0] = getInt(64);
            newApple[1] = getInt(64);
        } while (validateApple(newApple) == 0);

        apple[0] = newApple[0];
        apple[1] = newApple[1];

        draw(apple[0], apple[1]);
}

int getNextX() {

        int x;

        switch (dir) {
            case 1:
                x = snake[0][0] + 1;
                break;
            case 2:
                x = snake[0][0];
                break;
            case 3:
                x = snake[0][0] - 1;
                break;
            case 4:
                x = snake[0][0];
                break;
        }

        return x;
```

```
}

int getNextY() {

	int y;

	switch (dir) {
		case 1:
			y = snake[0][1];
			break;
		case 2:
			y = snake[0][1] - 1;
			break;
		case 3:
			y = snake[0][1];
			break;
		case 4:
			y = snake[0][1] + 1;
			break;
	}

	return y;
}

int validateApple (int newApple[]) {
	for (int i = 0; i < length; i++) {
		if (snake[i][0] == newApple[0] && snake[i][1] == newApple[1])
{
			return 0;
		}
	}
	if (newApple[0] > 62 || newApple[0] < 1 || newApple[1] > 62 ||
newApple[1] < 1) {
		return 0;
	}
	return 1;
}

void drawSnake() {
	for (int i = 0; i < length; i++) {
		draw(snake[i][0], snake[i][1]);
```

```
      }
}

void setupSnake() {

      snake[0][0] = 32;
      snake[0][1] = 30;

      snake[1][0] = 32;
      snake[1][1] = 31;

      snake[2][0] = 32;
      snake[2][1] = 32;

      drawSnake();

}

void writeScore() {

      int startX = 3;
      int startY = 67;

      int tempPoints = points;
      int nDigits = floor(log10(abs(points))) + 1;

      if (points == 0) {
            nDigits = 1;
      }

      for (int i = 0; i < nDigits; i++) {
            writeInt(startX, startY + i * 8, tempPoints % 10);
            tempPoints = tempPoints / 10;
      }
}

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

```
void resetHigh () {
      PORTA |= (1<<2);
}

void resetLow () {
      PORTA &= ~(1<<2);
}

void eHigh () {
      PORTA |= (1<<5);
}

void eLow () {
      PORTA &= ~(1<<5);
}

void rsHigh () {
      PORTA |= (1<<4);
}

void rsLow () {
      PORTA &= ~(1<<4);
}

void rwHigh () {
      PORTA |= (1<<3);
}

void rwLow () {
      PORTA &= ~(1<<3);
}

void selectChip1 () {
      PORTA &= ~(1<<1);
      PORTA |= (1<<0);
}

void selectChip2 () {
      PORTA &= ~(1<<0);
      PORTA |= (1<<1);
}
```

```
void startDisplay () {

      eLow();
      rsLow();
      rwLow();

      // Set data
      PORTB |= 0b00111111;

      PORTA |= (1<<0);
      PORTA |= (1<<1);

      eHigh();
      eLow();
}

void endDisplay () {

      rsLow();
      rwLow();

      // Set data
      PORTB |= 0b00111110;

      eHigh();
      eLow();
}

void startGame() {
      // Enable interrupt for buttons
      PIND = 0b00000000;
      DDRD = 0b10001011;
      GICR = 0b01000000;
      MCUCR = 0b00000011;
      sei();
}

void setupBoard() {
      // Draw the edge of the board.
      for (int i = 0; i < 64; i++) {
```

```
            draw(0, i);
            draw(63, i);
            draw(i, 0);
            draw(i, 63);
      }
}

void setX(int x) {

      rsLow();
      rwLow();

      PORTB = 0b10111000 | (x/8);

      eHigh();
      eLow();
}

void setY(int y) {

      rsLow();
      rwLow();

      PORTB = 0b01000000 | y;

      eHigh();
      eLow();
}

void draw(int x, int y) {

      int newY;

      if (y < 64) {
            selectChip1();
            newY = y;
      } else {
            selectChip2();
            newY = y - 64;
      }
```

```
	setY(newY);
	setX(x);

	rsHigh();
	rwLow();

	display[x/8][y] = display[x/8][y] | (1<<(x%8));
	PORTB = display[x/8][y];

	eHigh();
	eLow();
}

void erase(int x, int y) {

	int newY;

	if (y < 64) {
		selectChip1();
		newY = y;
	} else {
		selectChip2();
		newY = y - 64;
	}

	setY(newY);
	setX(x);

	rsHigh();
	rwLow();

	display[x/8][y] = display[x/8][y] & ~(1<<(x%8));
	PORTB = display[x/8][y];

	eHigh();
	eLow();
}

void clearDisplay() {
	for (int c = 0; c < 128; c++) {
		for (int r = 0; r < 64; r++) {
```

```
                erase(r,c);
            }
        }
}

int getInt (int limit) {
    int divisor = RAND_MAX/(limit+1);
    int retval;

    do {
        retval = rand() / divisor;
    } while (retval > limit);

    return retval;
}

void writeInt (int x, int y, int n) {

     if (n >= 0 && n <= 9) {
          for (int r = x; r < x + 10; r++) {
               for (int c = y; c < y + 6; c++) {
                    draw(r, c);
               }
          }

          switch (n) {
               case 0:
                    for (int r = x + 2; r < x +8; r++) {
                         erase(r, y + 2);
                         erase(r, y + 3);
                    }
               break;
               case 1:
                    for (int r = x; r < x + 10; r++) {
                         erase(r, y + 2);
                         erase(r, y + 3);
                         erase(r, y + 4);
                         erase(r, y + 5);
                    }
               break;
               case 2:
```

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

            erase(x + 3, y);
            erase(x + 3, y + 1);
            erase(x + 3, y + 2);
            erase(x + 3, y + 3);

            erase(x + 6, y + 2);
            erase(x + 6, y + 3);
            erase(x + 6, y + 4);
            erase(x + 6, y + 5);

            erase(x + 7, y + 2);
            erase(x + 7, y + 3);
            erase(x + 7, y + 4);
            erase(x + 7, y + 5);
break;
case 3:
            erase(x + 2, y + 2);
            erase(x + 2, y + 3);
            erase(x + 2, y + 4);
            erase(x + 2, y + 5);

            erase(x + 3, y + 2);
            erase(x + 3, y + 3);
            erase(x + 3, y + 4);
            erase(x + 3, y + 5);

            erase(x + 6, y + 2);
            erase(x + 6, y + 3);
            erase(x + 6, y + 4);
            erase(x + 6, y + 5);

            erase(x + 7, y + 2);
            erase(x + 7, y + 3);
            erase(x + 7, y + 4);
            erase(x + 7, y + 5);
break;
```

```
case 4:
     for (int r = x; r < x + 4; r++) {
          for (int c = y + 2; c < y + 6; c++) {
               erase(r, c);
          }
     }
     erase(x + 6, y + 2);
     erase(x + 6, y + 3);
     erase(x + 7, y + 2);
     erase(x + 7, y + 3);
     erase(x + 8, y + 2);
     erase(x + 8, y + 3);
     erase(x + 9, y + 2);
     erase(x + 9, y + 3);
break;
case 5:
          erase(x + 2, y + 2);
          erase(x + 2, y + 3);
          erase(x + 2, y + 4);
          erase(x + 2, y + 5);

          erase(x + 3, y + 2);
          erase(x + 3, y + 3);
          erase(x + 3, y + 4);
          erase(x + 3, y + 5);

          erase(x + 6, y);
          erase(x + 6, y + 1);
          erase(x + 6, y + 2);
          erase(x + 6, y + 3);

          erase(x + 7, y);
          erase(x + 7, y + 1);
          erase(x + 7, y + 2);
          erase(x + 7, y + 3);
break;
case 6:
          erase(x + 2, y + 2);
          erase(x + 2, y + 3);

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

```
            erase(x + 3, y + 3);

            erase(x + 6, y);
            erase(x + 6, y + 1);
            erase(x + 6, y + 2);
            erase(x + 6, y + 3);

            erase(x + 7, y);
            erase(x + 7, y + 1);
            erase(x + 7, y + 2);
            erase(x + 7, y + 3);
break;
case 7:
     for (int r = x; r < x +8; r++) {
            erase(r, y + 2);
            erase(r, y + 3);
            erase(r, y + 4);
            erase(r, y + 5);
     }
break;
case 8:
            erase(x + 2, y + 2);
            erase(x + 2, y + 3);

            erase(x + 3, y + 2);
            erase(x + 3, y + 3);

            erase(x + 6, y + 2);
            erase(x + 6, y + 3);

            erase(x + 7, y + 2);
            erase(x + 7, y + 3);
break;
case 9:
            erase(x + 2, y + 2);
            erase(x + 2, y + 3);
            erase(x + 2, y + 4);
            erase(x + 2, y + 5);

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

```
                        erase(x + 3, y + 4);
                        erase(x + 3, y + 5);

                        erase(x + 6, y + 2);
                        erase(x + 6, y + 3);

                        erase(x + 7, y + 2);
                        erase(x + 7, y + 3);
                  break;
            }
      }

}

void writeSnake() {
      int x = 49;
      int y = 67;

      // S
      y = y + 12;
      for (int r = 0; r < 12; r++) {
            for (int c = 0; c < 10; c++) {
                  draw(x + r, y + c);
            }
      }
      for (int i = 0; i < 8; i++) {
            erase(x+2, y+2+i);
            erase(x+3, y+2+i);
            erase(x+4, y+2+i);
            erase(x+7, y+i);
            erase(x+8, y+i);
            erase(x+9, y+i);
      }

      // N
      y = y + 12;
      for (int r = 0; r < 12; r++) {
            for (int c = 0; c < 10; c++) {
                  draw(x + r, y + c);
            }
      }
```

```
for (int r = 0; r < 7; r++) {
      for (int c = r; c < 7; c++) {
            erase(x+r, y+c+1);
            erase(x+11-r, y+8-c);
      }
      //erase(x+11, y+r+2);
      //erase(x, y+r+2);
}

// A
y = y + 12;
for (int r = 0; r < 12; r++) {
      for (int c = 0; c < 10; c++) {
            draw(x + r, y + c);
      }
}
for (int c = 0; c < 6; c++) {
      for (int r = 0; r < 5; r++) {
            erase(x+r, y+2+c);
      }
      for (int r = 0; r < 3; r++) {
            erase(x+7+r, y+2+c);
      }
}

// K
y = y + 12;
for (int r = 0; r < 12; r++) {
      for (int c = 0; c < 10; c++) {
            erase(x + r, y + c);
      }
}
for (int i = 0; i < 6; i++) {
      for (int j = i; j < 3 + i; j++) {
            draw(x + i, y + j);
            draw(x + 11 - i, y + j);
      }
}
for (int i = 0; i < 12; i++) {
      draw(x + i, y + 9);
      draw(x + i, y + 8);
```

```
	}

	// E
	for (int r = 0; r < 12; r++) {
		for (int c = 0; c < 10; c++) {
			draw(x + r, y + c);
		}
	}
	for (int i = 0; i < 8; i++) {
		erase(x + 2, y + i);
		erase(x + 3, y + i);
		erase(x + 4, y + i);

		erase(x + 7, y + i);
		erase(x + 8, y + i);
		erase(x + 9, y + i);
	}
	erase(x+5, y);
	erase(x+5, y+1);
	erase(x+6, y);
	erase(x+6, y+1);

}

// Buttons
ISR(INT0_vect) {

	if (click == 1) {
		return;
	}

	click = 1;
	cli(); // Disable interrupt

	_delay_ms(200);

	switch(PIND) {

		case 0b00010100: // Right button pressed
			if (dir == 4) {
				dir = 1;
```

```
                } else {
                        dir++;
                }
        break;

        case 0b00100100: // Left button pressed
                if (dir == 1) {
                        dir = 4;
                } else {
                        dir--;
                }
        break;

        case 0b01000100: // Reset button pressed
                sei(); // Enable interrupt
                click = 0;
                restart();
        break;
    }

    sei(); // Enable interrupt

    click = 0;
}
```