LUNDS UNIVERSITET
Lunds Tekniska Högskola

Projektrapport – Målgång

Lunds Tekniska Högskola
EITF11 – Digitala Projekt

2016-05-23

Grupp 3

Författare:
Albert Johansson
Ludvig Brisby Jeppsson
Mattias Hamberg

Handledare:
Bertil Lindvall

1 Inledning

Projektet har gått ut på ta fram, montera och programmera en produkt från idé till en fungerande prototyp. Vi har fått ökad kunskap inom framtagningen av digitala produkter vilket involverar framtagande av idé, ritningar, kopplingsschema, lödning, programmering i språket C och mycket mer.

1.1 Syfte

Syftet med projektet är att bygga en målgång som tar reda på vem som först passerar en mållinje mellan två banor samt mäta tidsskillnaden mellan de två tävlande.

2 Produktbeskrivning

Produkten ska bestå av en målgång som är uppdelad i två banor. I varje bana finns en IR-diod och en fototransistor som tillsammans känner av när ett föremål passerar, exempelvis en gummianka eller en leksaksbil. När första föremålet passerar lyser en grön LED vid denna bana samt startas en klocka som sen avslutas när ett föremål passerar i den andra banans målgång. Efter det presenteras tidsskillnaden på en display för användaren. Om användaren sedan önskar att tävla mellan två föremål igen används en återställningsknapp för att starta om systemet.

3 Kravspecifikation

Denna kravspecifikation behandlar konstruktion av hårdvara och mjukvara för en målgång.

3.1 Funktionella krav

1. Systemet ska ha två parallella banor.
2. Systemets banor ska ha varsin mållinje.
3. Systemets banor ska vara minst 10 cm breda.
4. Systemet ska använda sig av processorn AVR Mega 16.
5. Systemets banor ska ha varsin diod.
6. Vinnarbanans diod ska lysa.
7. Tidsskillnad ska visas på en display.
8. Målgång ska mätas mellan två lysdioder och två fototransistorer.
9. Tidsskillnaden ska visas i antal sekunder med 3 decimaler.
10. Systemet ska ha en knapp som återställer målgången (rensar displayen samt släcker vinnarbanans lysdiod).

3.2 Kvalitetskrav

1. Tidsskillnaden ska ha en felmarginal på maximalt 5 procent.
2. Information ska skrivas ut på displayen inom 1 sekund efter att den andra målgången passerats.

4 Hårdvara

Nedan beskrivs alla komponenter som använts i projektet

4.1 Processor

Processorn som används är ATmega16. Den har 40 Input/Output-portar varav 32st är lediga att programmera mot. Den har ett programmerbart minne på 16kB. Hur portarna används kan ses i kopplingsschemat som finns i Appendix.

4.2 Display

För att skriva ut tidsskillnaden mellan de tävlande används en alfanumerisk display av typen SHARP dot-matrix.

4.3 Målgång

Det finns 2st IR-dioder (TSUS5400) som utgör de två målgångarna tillsammans med var sin fototransistor (PT204-6B).

4.4 Vinnardioder

För att indikera vilken av de två tävlande som kommer i mål först finns en grön LED vid varje bana. Enbart den LED vid vinnarens bana börjar lysa.

4.5 Återställningsknapp

Det finns en knapp för att återställa programmet så att det kan göras en ny målgång.

4.6 Hjälpkomponenter

För att justera kontrasten på displayen används en regulator som går att variera och mellan fototransistorerna används en Schmittrigger (74HC14).

4.7 Resistorer

Till IR-dioderna används två stycken 50 Ω-resistorer. Till fototransistorerna används två stycken 22 kΩ-resistorer. Till vinnardioderna används två stycken 390 Ω-resistorer. Till återställningsknappen används en 10 kΩ-resistorer.

5 Mjukvara

Programmet startar med att ställa in display, interrupts, klocka och globala variabler. Funktionaliteten bygger på att interrupts triggas och utför diverse kommando. För detaljerad beskrivning av metoderna hänvisas till kommentarer i källkoden som återfinns i Appendix.

Displayen kräver kod för att skicka högt (1) eller lågt (0) från 3 av portarna (E, RS och R/W). Det skrevs därför metoder för *low* och *high* för de portarna som skickade antingen 0 eller 1. Dessa metoder kräver ett skifte från högt till lågt när det skickas kommando, eller tecken till displayen. För att underlätta utskrifter på displayen skrevs diverse hjälpmetoder, bland annat en för att skriva ut strängar (write_string()) och en för att skriva ut tecken (write_char()).

För att avgöra start av ett lopp används en knapp som är kopplad till en interrupt. När knappen trycks in väntar programmet på att mållinjerna bryts. Detta avgörs med två andra interrupts som triggas då ljusstrålarna mellan diod och fototransistor bryts. Vid den första interrupten sätts en klocka igång. Klockan stannas då den andra ljusstrålen bryts, och tidsskillnaden visas på skärmen.

Klockan fungerar genom att använda en overflow interrupt där en global variabel ökar med ett varje gång det inträffar en overflow interrupt. Denna overflow interrupt inträffar ca 488,28125 gånger per sekund på grund av processorns frekvens på 1 MHz, en prescaler på 8 och en timer på 8 bitar. (1000000 / 2^8 / 8 = 488,28125).

6 Metod

6.1 Planering

Projektet började med att undersöka lämpliga produkter att göra. Med inspiration av tidigare projekt men med viljan att differentiera vårt projekt valde vi att göra en målgång. Först bestämdes funktioner produkten skulle ha vilket slutligen ledde till en kravspecifikation. Därefter bestämdes vilka komponenter som behövdes och slutligen gjordes ett kopplingsschema som finns i Appendix.

6.2 Hårdvarukonstruktion

Efter att ha fått tag i alla komponenter som vi behövde sattes de ihop enligt kopplingsschemat.

6.3 Programmering

När hårdvaran var ihopsatt sattes arbetet igång med mjukvaruutvecklingen. Programmeringsspråket C användes och skrevs i programmet AtmelStudio. För att kunna debugga samt föra över koden till processorn från datorn användes en JTAG.

7 Resultat och diskussion

Likt alla projekt går inte allt enligt ursprunglig plan. Ibland skedde lite felkopplingar mot processorn eller felkopplingar till exempelvis anoden och katoden på dioderna. Då vi kopplat fel till en början på fototransistorna trodde vi inte att IR-dioderna var tillräckligt starka då processorn inte kände av när något bröt målgången mellan IR-dioderna och fototransistorerna. Vi sänkte styrkan på motstånden från 150 Ω till 50 Ω. Då detta inte gjorde någon skillnad la vi till en Schmittrigger mellan fototransistorn och processorn. Även i detta fall fick vi inget utslag när något bröt målgången. Det var då vi upptäckte att fototransistorerna var felkopplade. Genom att leta upp databladet för fototransistorerna lyckades vi sen koppla det rätt och med hjälp av tidigare justeringar fungerade målgångarna utmärkt. Ett annat problem som vi stötte på var att displayens kontrast skulle behöva justeras då den skrev mörka tecken på en mörk bakgrund. Detta löstes relativt enkelt genom att lägga till en justerbar regulator.

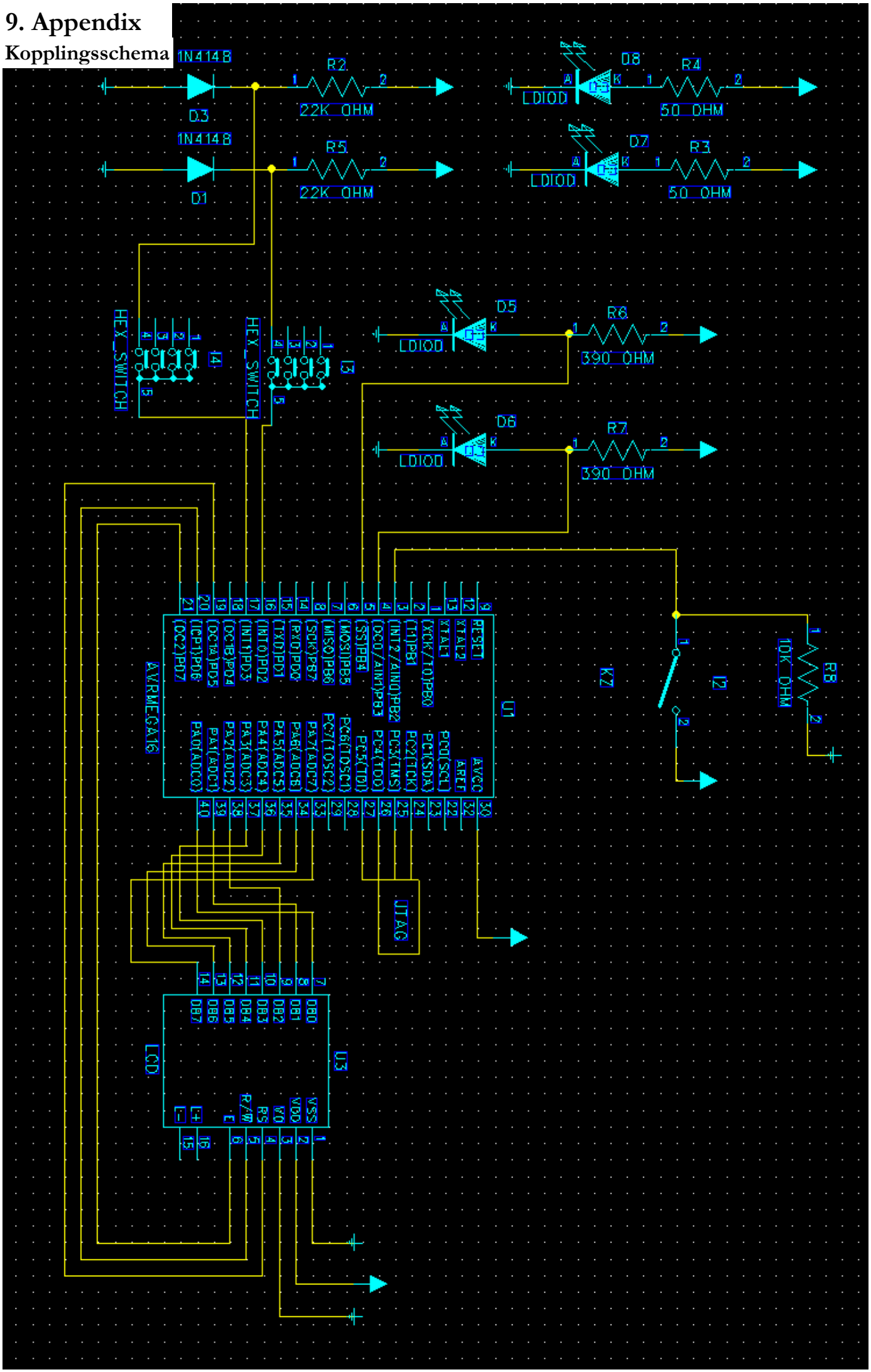
Variationer som diskuterades och som kunde ha implementerats men inte gjordes var bland annat att ha enbart en bana istället för två. Då hade de tävlande kunnat passera samma mållinje och försäkrat om att de åkt lika långt. I vårt fall med två banor kan det variera lite var målgången går. Problemet hade dock blivit att avgöra vilken av de tävlande som vinner om båda passerar ungefär samtidigt, samt att krockar hade kunnat uppstå. En idé som kom fram under denna diskussion var att ha med en riktig kamera för att kunna se ett målfoto. Dock hade det inneburit att behöva införskaffa en kamera med mer avancerad programmering och koppling samt att behöva byta upp till högupplöst grafisk display. Då detta projekt snarare ska fokusera på grunderna i att ta fram en digital produkt sköts idén ner då den var alldeles för avancerad.

8 Slutsats

Genom projektet har vi fått en inblick i en oupptäckt värld kring utvecklande och konstruktion av digitala produkter. Ett område som man ser som komplex och lite skrämmande vid en första anblick men med bra instruktioner från handledare och en hel del tid kan närmast jämföras med hur man börja bygga medlego vid ung ålder. Vi är mycket nöjda med projektets gång och resultatet att vi med i princip noll kunskap om hårdvarukonstruktion kunnat få ihop en prototyp som står sig mot alla krav i den ursprungliga kravspecifikationen.

9. Appendix

Kopplingsschema



```
/*
 * GccApplication3.c
 *
 * Created: 2016-04-22 13:02:52
 *  Author: digpi03
 */

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

int race_active; // 0 ej aktivt, 1 aktivt
int passed_goal_line; // 0 ingen löpare har passerat mållinje, 1 en löpare har passerat
mållinje
int passed_left_goal_line; // 0 ej passerar, 1 passerat
int passed_right_goal_line; // 0 ej passerar, 1 passerat
volatile int count;
double measured_time = 0;


/************************************************************
 * SKÄRMEN
 *
 */
void setup_display() {

    // Function set
    DDRA = 0b11111111;
    _delay_ms(1);
    DDRD = 0b11100000;
    _delay_ms(1);
    highE();
    _delay_ms(1);
    PORTA = 0b00110000;
    _delay_ms(1);
    lowE();
    _delay_ms(1);

    //Display on
    display_command();
    PORTA = 0b00001100;
    _delay_ms(1);
    lowRS();
    _delay_ms(1);
    lowRW();
    _delay_ms(1);
    lowE();
}

// Clear display i databladet
void clear_display() {
    write_cmd(0b00000001); // Clear command
    _delay_ms(1); // behövs för att det ska fungera
}

void welcome_display() {
    write_string("PUSH THE BUTTON TO START.");
}
```

```
/*************************************************************
 * HJÄLPMETODER TILL SKÄRMEN
 *
 */

// Ska alltid användas innan kommando/tecken skickas
void display_command() {
    highE();
    highRW();
    highRS();
}

// Skriva kommando (ej tecken!)
void write_cmd(char c) {
    display_command();
    PORTA = c;
    _delay_ms(1);
    lowRS();
    _delay_ms(1);
    lowRW();
    _delay_ms(1);
    lowE();
}

// Skriva tecken till skärmen
void write_char(char ch) {
    display_command();
    PORTA = ch;
    _delay_ms(1);
    lowRW();
    _delay_ms(1);
    lowE();
}

// Skriva strängar till skärm
void write_string(char str[]) {
    int i = 0;
    while( str[i] != '\0') { // \0 null character in ASCII
        display_command();
        PORTA = str[i];
        _delay_ms(1);
        lowRW();
        _delay_ms(1);
        lowE();
        i++;
    }
}

// Höga och låga E, RS, R/W för displayen
void lowE() {
    PORTD &= 0b01111111;
}
void lowRW() {
    PORTD &= 0b10111111;
}
void lowRS() {
```

```
    PORTD &= 0b11011111;
}
void highE() {
    PORTD |= 0b10000000;
}
void highRW() {
    PORTD |= 0b01000000;
}
void highRS() {
    PORTD |= 0b00100000;
}



/*****************************************************************
 * DIODER
 *
 */
void turn_off_diods() {
    DDRB |= 0b00011000;
    PORTB &= 0b11100111;
}

void light_left_diod() {
    DDRB |= 0b00011000;
    _delay_ms(1);
    PORTB |= 0b00001000;
}
void light_right_diod() {
    DDRB |= 0b00011000;
    _delay_ms(1);
    PORTB |= 0b00010000;
}

/*****************************************************************
 * RACE
 *
 */
void setup_race() {
    race_active = 0;
}

void start_race() {
    clear_display();
    write_string("GET READY");

    _delay_ms(1000);
    clear_display();
    write_char('3');

    _delay_ms(1000);
    clear_display();
    write_char('2');

    _delay_ms(1000);
    clear_display();
    write_char('1');
```

```
    _delay_ms(1000);
    clear_display();
    write_string("GO");
}

void finish_race(char winner) {
    clear_display();
    write_string("WINNER: ");
    write_char(winner);
}

void race_logic(int track) { // track: 0 = höger bana, 1 = vänster bana
    if ( race_active == 1 ) {
        if (passed_goal_line == 0) {
            passed_goal_line = 1;
            start_clock();

            if (track == 0) { // Höger bana
                finish_race('2');
                light_right_diod();
                passed_right_goal_line = 1;
            }

            if (track == 1) { // Vänster bana
                finish_race('1');
                light_left_diod();
                passed_left_goal_line = 1;
            }

        } else if ( ( passed_left_goal_line == 0 && track == 1 ) || ( passed_right_goal_line
        == 0 && track == 0 ) ) {
            stop_clock();
            race_active = 0;
            _delay_ms(1);
            write_string("       DIFF: ");

            double time_in_seconds = measured_time/488.28125; // 488.28125 iom prescaler 8
            och 256, se längre ned

            int size = 7;
            char time_str[size]; // Tillåter max 7 tecken XXX.XXX float. Alltså max 999.999s
            tidsskillnad

            dtostrf(time_in_seconds, size, 3, time_str); //  Skapar char-array av double (3e
            parametern anger antal decimaler)
            write_string(time_str);
            write_string(" s");
        }
    }
}

/*************************************************************
 * INTERRUPT
 *
 */
void setup_interrupt() {

    MCUCSR = MCUCSR | 0b00001111; // Interrupt 0,1 (höger och vänster bana)
```

```
    MCUCSR = MCUCSR | 0b01000000; // Interrupt 2 (knappen)
    GICR = GICR | 0b11100000; // Sätt igång Interrupt 0,1,2

    sei(); // Enable global interrupt
}


// Höger bana
ISR(INT0_vect) {
    race_logic(0); // 0 = höger bana
}

// Vänster bana
ISR(INT1_vect) {
    race_logic(1); // 1 = vänster bana
}

// Knappen
ISR(INT2_vect) {

    if ( race_active == 0 ) {

        turn_off_diods();
        start_race();
        GIFR = 0b11100000; // Rensar Interrupt flags, vilket innebär att alla Interrupts som
        har körts då nedräkningen pågått ej räknas (motverkar tjuvstart)

        measured_time = 0;
        count = 0;
        race_active = 1;
        passed_right_goal_line = 0;
        passed_left_goal_line = 0;
        passed_goal_line = 0;
    }

}

// Overflow Interrupt (för klockan)
ISR(TIMER0_OVF_vect) {
    count++; // Räknas upp vid varje interrupt. I det här fallet 488.28125 per sekund
}


/*************************************************************
 * KLOCKAN
 *
 */
void clock_setup() {
    TCCR0 = TCCR0 | 0b00000010; // Prescaler 8
    TIMSK = TIMSK | 0b00000001; // Overflow interrupt aktivt
}
void start_clock() {
    count = 0;
}
void stop_clock() {
    measured_time = count;
}
```

```
/*************************************************************
 * MAIN_METOD.
 * Programmets kärna
 *
 */
int main(void) {
    setup_display();
    setup_race();
    clear_display();
    welcome_display();
    setup_interrupt();
    clock_setup();
    turn_off_diods();

    while(1) {
    }
}
```