LUNDS UNIVERSITET

LTH

LUNDS TEKNISKA HÖGSKOLA

Digitala System

EITA 15

Kursansvariga: Bertil Lindvall &
Lars-Göran Larsson

AI Bil

Alasadi, Rami
Ayad, Abdelkrim
Salaah, Mohamed Hassan
Ghulam Sarwar, Sami
Sbaie, Abdallah

Sammanfattning

Detta projekt genomfördes vårterminen 2023 i kursen Digital System EITA 15. Syftet med detta projekt är att tillämpa den teoretiska kunskapen som framfördes under kursen genom att konstituera ett fungerande prototyp av en självkörande bil baserad på mikroprocessorn Atmega 1284. Gruppen hade som mål att utveckla en fungerande prototyp av en linje följande robotbil som kan navigera på en förutbestämd bana, identifiera hinder och agera på dessa genom att stanna. I början av projektet skapades ett kretsschema och kravspecifikationen för prototypen bestämdes. Efter att ha fått godkännande av kursansvarig fortsatte gruppen med att samla in det efterfrågade material och sedan bygga och programmera prototypen. Resultatet var en prototyp som levde upp till målsättningen och stämde överens i enlighet med de förbestämda kravspecen. Dessutom skapades en webbplats för att presentera projektet, med information om gruppmedlemmar, en demo, ritningar, källkod och projektets rapport.

Nyckelord

ATMEGA 1284, Självkörande bil, Digitala System, Robotbil

Abstract

This project was carried out in the spring semester 2023 in the course Digital System EITA15. The aim of this project is to apply the theoretical knowledge presented during the course by constituting a working prototype of a self-driving car based on the microprocessor Atmega 1284. The group aimed to develop a working prototype of a line-following robot car that can navigate on a predetermined path, identify obstacles and act on them by stopping. At the beginning of the project, a circuit diagram was created and the requirements specification for the prototype was determined. After receiving approval by the course manager, the group proceeded to collect the requested material and then build and program the prototype. The result was a prototype that lived up to the objective and agreed in accordance with the predetermined requirement specifications. Additionally, a website was created to present the project, with information about team members, a demo, blueprints, source code, and the project's report.

Keywords

ATmega1284, Self-driving car, Digital systems, Robot car.

Innehållsförteckning

1. Inledning

Kursen EITA 15, inom programmet Elektro- och informationsteknik och Datateknik på universitetet, innefattar ett projektarbete som syftar till att ge studenterna praktisk erfarenhet inom fältet. Vårt projekt har fokus på självkörande bilar, vilket är en aktuell och spännande teknologi som väcker intresse över hela världen. Projektet är ett grupparbete där vi har bestämt att bygga och programmera en robot bil som kan följa en linje på en bana.

1.1 Syfte

Syftet med projektet är att tillämpa den kunskap som byggts upp under kursens gång på en prototyp i ett gemensamt grupparbete och därmed fördjupa förståelsen för det som är grundläggande inom ämnet. Genom att arbeta i grupp ska studenterna även utveckla sin förmåga att samarbeta och kommunicera i en projektbaserad miljö. Förväntat resultat är att studenterna ska ha utvecklat en fungerande prototyp i detta fall var en självkörande bil som uppfyller de krav som satts upp i kravspecifikationen och att de ska ha ökat sin kunskap inom området samt utvecklat sin förmåga att samarbeta i grupp.

1.2 Målformulering

Det konkreta och mätbara mål för projektet inkluderar att utveckla en fungerande prototyp som kan navigera på en förutbestämd bana, identifiera hinder och agera på dessa genom att stanna. Målet relaterar direkt till projektets övergripande syfte att tillämpa den kunskap som byggts upp under kursens gång på en prototyp i ett gemensamt grupparbete och därmed fördjupa förståelsen för det som är grundläggande. Genom att uppnå de konkreta målen kommer projektet att bidra till att uppnå detta övergripande syfte.

1.3 Problemformulering

- Hur kan bilen upptäcka och undvika hinder på vägen?
- Hur kan bilen anpassa sin hastighet i svängningar?

1.4 Motivering av examensarbetet

Vi valde att arbeta med en självkörande bil som skulle kunna följa en linje eftersom det var en utmaning som krävde att vi använde våra kunskaper inom digitala system på ett kreativt och praktiskt sätt. Dessutom var det ett intressant och relevant ämne då självkörande bilar har blivit alltmer aktuellt ämne i samhället. Vi ville också ha möjlighet att arbeta med en fysisk prototyp som vi kunde testa och presentera, vilket gjorde projektet mer spännande och givande.

1.5 Avgränsningar

En begränsning till det här projektarbetet är antalet sensorer som användes i bilen. Tanken var att bilen skulle ha tre sensorer, fram, höger och vänster. Men de fanns inte tillgängliga. Därför ändrades planen till endast en sensor på framsidan som skulle hantera ban erkännande.

2. Teknisk bakgrund

I detta avsnittet presenteras tekniska detaljer som behövs för att förstå projektet i detalj och på en djupare nivå. Här presenteras de olika komponenterna, sensorerna och teknikerna som användes både för hård- och mjukvara.

2.1 Hårdvara

Tabellen nedan visar komponenter som användes i projektet.

Tabell 1

Komponent	Funktion
ATMEGA1284	AVR microcontroller, styr andra komponenter och sensorer.
QTR-3A Reflectance Sensor Array	Sensorn känner av den svarta linjen och skickar data till microcontrollern. Placerades på bilens framsida.
DRV8833 Dual H-Bridge Drive	H-bryggan sitter mellan de två motorerna som driver hjulen och microcontrollern. Den kan styra två motorer samtidigt.
HC-SR04 Ultrasonic Sensor	Sensorn används för att mäta avstånd till andra objekt. Placerades på framsidan av bilen.
DC-Motor (2)	För att driva hjulen.
Switch	För att stänga av och sätta på spänningskällan.
Resistor (3)	Begränsar spänningen för komponenter som kräver lägre spänning.
Lampa (3)	Indikerar vilka uppgifter bilen håller på att utföra.
AA Batteri (4)	En spänningskälla för att driva alla komponenter

2.1.1 Reflectance Sensor

Den här fototransistor sensorn användes för att känna av den svarta linjen. På QTR-3A Reflectance Sensor Array sitter tre stycken fototransistorer med separata analoga utgångar. Sensorn behöver ungefär 5V för att drivas. Den ger en 0:a vid erkännande av ett svart underlag och 1:a vid andra ljusa underlag. I mer detalj fungerar sensorn genom att skicka ljus och observera ljuset som reflekteras. Sensorn genererar ljus och har en fotodiod som genererar energi beroende på mängden ljus som har reflekterats.

2.1.2 Dual H-Bridge Drive

En dual H-brygga är en krets som används för att styra riktning och hastighet hos en DC-motor. Den består av två H-bryggkretsar som är sammankopplade och som möjliggör en tvåvägs styrning av två

motorer. Varje H-brygga krets består av fyra transistorer som styrs av en mikrokontroller eller annan styrelektronik som skickar signaler för att slå på eller av varje transistor i H-bryggan.

Genom att kontrollera signalerna som skickas till transistorerna kan dual H-bryggan driva motorn framåt eller bakåt, samt styra hastigheten på motorn genom att variera mängden ström som skickas till den. Detta gör den till ett populärt val för många applikationer, inklusive robotik, automation och elektriska fordon.

2.1.3 Ultrasonic Sensor

En avståndsmätare kan de också kallas. Som namnet antyder så använder den här sensor ultraljud för att kunna mäta avstånd på objekt. På sensorn sitter en transmitter och en receiver. Transmittern har ansvaret att sända ljudvågor med viss frekvens högre än människans frekvens hör zon. Dessa vågor studsar mot objekt och återvänder till sensorn. Då är det receiverns ansvar att fånga upp fånga upp de vågorna.

Programmeringsmässigt ska man använda sig av tid för att kunna mäta avstånd till objekt. Det som händer är att man skickar en 1:a till transmittern, sen väntar man på att få en 1:a på receivern. Med tiden emellan kan man räkna avståndet som vågorna har färdats och då bestämma ett avstånd.

2.2 Mjukvara

En huvuddel av det här projektet är mjukvaran som ska styra hårdvaran som används. För att programmera ATMEGA:an användes Atmel Studio 7, programmeringsspråket som användes var C. En webbsida skulle skapas för projektet, sidan skapades med HTML/CSS i visual studio.

2.2.1 Pulse width modulation (PWM)

På svenska pulsbreddsmodulering, en teknik som används för vissa komponenter för att kunna justera hastighet, styrka, effekten och annat. Tekniken används oftast vid arbete med DC-motorer eller lampor. Tanken är att kunna få ut mer av komponenter som används och få dem att bete sig precis på sättet man önskar programmera.

Pulsbreddsmodulering bygger på att sända icke kontinuerlig ström med korta tidsintervall. Det här tidsintervallet måste vara tillräckligt kort för att komponenten inte ska märka en skillnad mellan kontinuerlig och icke kontinuerlig ström. Det som händer är att effekten hos komponenten sänks och på så sätt minskas hastigheten, styrkan mm.

3 Metod

I den här delen av rapporten presenteras genomförandet av projektet, från planering till kravspecifikation till utmaningar som behövde lösningar.

3.1 Planering

I planeringen av projektet tog gruppen ett gemensamt beslut om att skapa en prototyp som skulle kunna navigera genom en svart linje. För att uppnå detta mål var det nödvändigt att komma överens

om vad som skulle göras och hur gruppen skulle gå tillväga. Detta resulterade i en detaljerad skiss på projektarbetet.

En viktig del av planeringen var att bestämma hur gruppen skulle kommunicera och samarbeta under projektets gång. Detta inkluderade regelbundna möten och användning av digitala verktyg för att underlätta samarbetet.

Byggandet av fordonet var en central del av projektet och krävde noggrann planering och samarbete. Detta inkluderade utformning av kretsschema och programmering av själva bilen.

Rapportskrivning var en viktig del av projektet och krävde att gruppen dokumenterade sina framsteg och resultat. Detta inkluderade även skrivning av en teknisk rapport som beskrev design och funktionalitet av den självkörande bilen. Efter att rapporten var skriven skulle gruppen också fixa en hemsida för att presentera projektet på ett mer visuellt sätt.

Till slut var det dags att presentera projektet och resultatet av arbetet. En muntlig presentation hölls för kursansvarig och övriga studenter där gruppen kunde visa upp sin självkörande bil och redovisa sin tekniska rapport.

3.2 Kravspecifikation

- Bilen ska följa en svart linje med hjälp av en sensor.
- Bilen ska kunna vrida höger och vänster.
- Bilen ska ha en sensor för att upptäcka objekt i vägen. Om det finns objekt så ska bilen stanna. avstånd ca 20-30 cm.
- Bilen ska drivas med fyra AA-batterier.
- Bilen ska ha 3 lampor som berättar vad som pågår i bilen. Varje lampa har sin egen färg.

3.3 Utmaningar

I projektet stötte vi på flera utmaningar som krävde kreativ problemlösning och tålamod. Några av de största utmaningarna inkluderade:

- Byggande - Vi försökte bygga hela fordonets riktning på en gång vilket ledde till vissa problem med att programmera och fick många kortslutningar. Det var viktigt för oss att bryta ner byggandet i mindre delar och bygga upp en komponent efter den andra.

- Programmering - Problemet med att använda ECHO för att mäta avstånd var att tiden blev överfylld när avståndet var långt. För att lösa detta använde vi en prescaler på timer och ignorerade avstånd som var för långa för att ECHO skulle kunna returnera korrekt data.

- Avläsning från sensoren - Vårt sista stora problem var att sensorn var placerad för långt från linjen för att kunna upptäcka den tillförlitligt. Vi behövde ändra sensorns placering och utforska olika lösningar för att förbättra dess prestanda.

Trots dessa utmaningar var det en lärorik upplevelse och vi lyckades övervinna alla hinder för att bygga ett fungerande fordon.

4 Analys

Kravspecifikationen för projektet inkluderade tydliga mål för att säkerställa prototypens funktionalitet och prestation. Genom att följa dessa specifikationer så kunde vi få en klarare bild på hur prototypen skulle se ut. Kravspecifikationen fungerade som en vägledning och referens för gruppen under projektets genomgång. Alla specifikationer som gruppen byggde i projektet uppnåddes.

Vissa problem har stötts på under arbetets genomgång. En bugg som har påträffats i bilens kod påverkade körningshastighet. Detta påverkade i sin tur sensorns förmåga att detektera den svarta linjen. Detta var mest kännbart när bilen svängde höger eller vänster, då den höga hastigheten ledde till att bilen körde utanför banan och bilen kunde inte fortsätta körningen. Detta löstes med att sänka gränshastigheten för bilen genom att utföra ändringar i koden så att bilen stannar inom ramen för banan och därefter så kunde bilen svänga och navigera genom den resterande sträcka av banan.

5 Resultat

Resultatet på det här projektet var en linje följande robotbil. Bilen kunde utföra alla kraven som fanns med i kravspecifikationen. Den kunde följa en svart linje, identifiera höger- och vänstersvängar, upptäcka hinder och stanna så länge hindret finns, drivas med 4 AA-batterier. Bilen hade också tre lampor som indikerar om bilen var på/av, finner linjen/ finner ej linjen och upptäcka hinder/ ingen hinder.

Projektets resultat var också en hemsida konstruerad med HTML/CSS som presenterade information om projektet. Sidan innehöll gruppmedlemmarnas namn, bild på bilen, demo, ritning, källkod och rapport. Den skulle vända sig till folk med ingen teknisk bakgrund.

6 Slutsats

Till slut blev projektet en succé utifrån de olika kravspecifikationer som sattes i förväg. Bilen lyckades uppnå kraven och fungerade som det var tänkt.

6.1 Framtida utvecklingsmöjligheter

Även om bil prototypen fungerar och uppfyller kraven så finns det fortfarande utrymme för förbättring. Det handlar både om att förbättra funktioner som redan finns men också att lägga till tekniker och funktioner för att få en mer kraftfull prototyp.

En sak som kan göra bilen mer kraftfull är att den har tre reflectance sensorer istället för en. På så sätt kan den använda de två sensorerna för att identifiera en sväng tydligare. Då kan sensorn i mitten användas till att hålla bilen rak och inte behöva ingå i identifieringen av en sväng. På så sätt kan bilen känna av banan bättre och tejp bredden spelar mindre roll.

En annan förbättring är att bilen kan backa. Genom att använda PWM kan man få bilen att backa. Den funktionen skulle kunna ge bilen möjlighet att ångra en sväng om vägen har ett slut. Bilen kan då backa och välja en annan väg.

7 Källförteckning

1. Pololu Robotics & Electronics (2022), QTR-3A Reflectance Sensor Array. https://www.pololu.com/product/2456 [2023-05-12]

2. Pololu Robotics & Electronics (2022), QTR-3A Reflectance Sensor Array. https://www.pololu.com/product/2130 [2023-05-12]

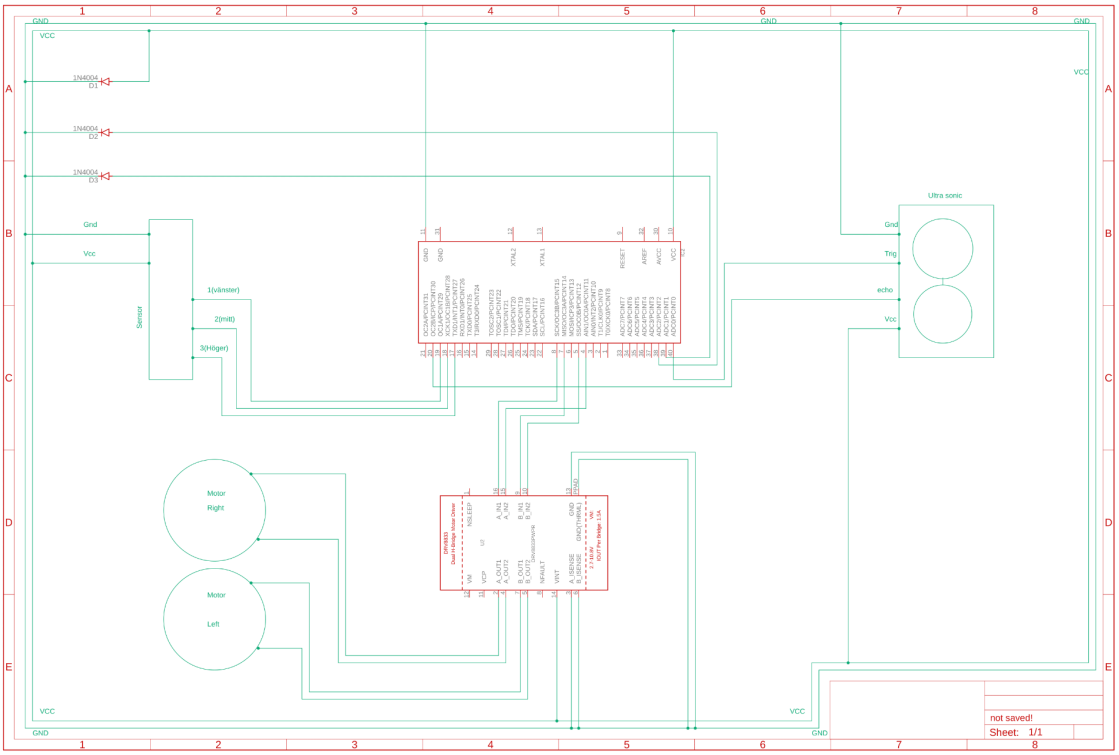
3. Atmel Corporation, “Atmel-42718C-ATmega1284_Datasheet_Complete-10/2016”, Datasheet, 2016 2. Texas Instruments, “DUAL H-BRIDGE MOTOR DRIVER”, Datasheet, 2011

4. Texas Instruments, “DUAL H-BRIDGE MOTOR DRIVER”, Datasheet, 2011

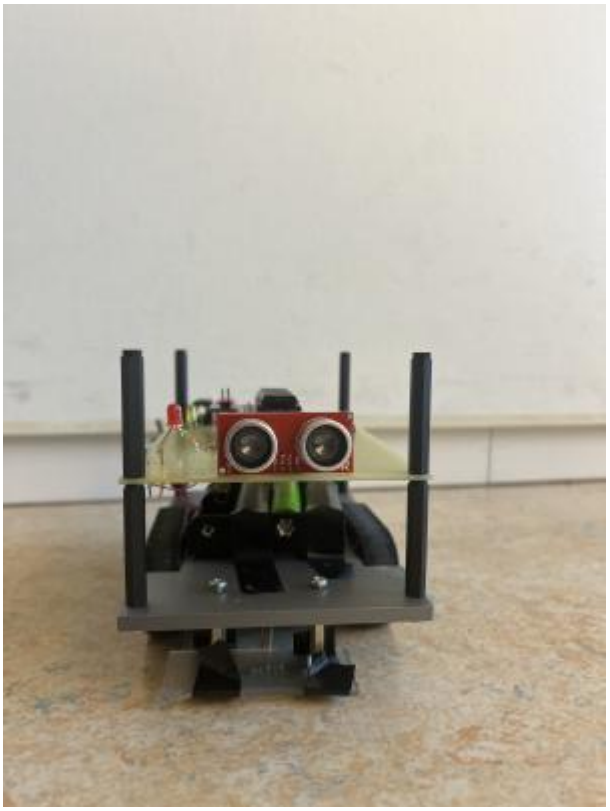
5. MaxBotix (2022), How Ultrasonic Sensors Work. https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work [2023-05-12]

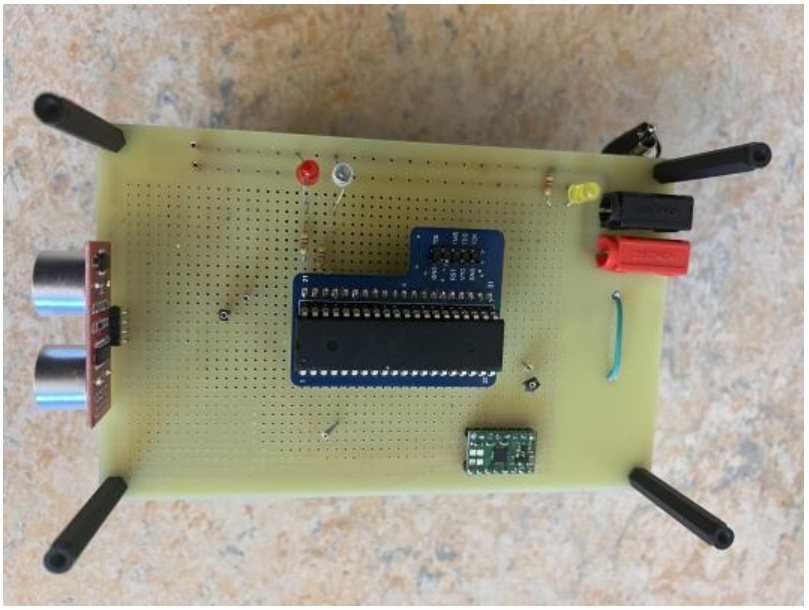
8 Appendix

8.1 Kretsschema



8.2 Prototyp bilder





8.3 Källkod

```
/* CPU freqvensce */
#define F_CPU 8000000UL
 /* Libraries*/
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

/* definition of static variables*/
#define  Trigger_pin    PORTA0  /* Trigger pin */

/* global variables */
long TimerOverflow = 0;
char string[10];
long count;
double distance;
uint8_t right1=0;
uint8_t left1 =0;
uint8_t  turn = 0;
uint8_t  lastturn = 0;

/* functions definition*/
void move(void);
double measureDistance(void);
void OC3ACounter(int speed);
void OC3BCounter(int speed);
void SetSpeedOC3A(int speed);
void SetSpeedOC3B(int speed);
void stop();


/* functions main one*/
int main(void)
{
```

```
    OC3ACounter(400); // set the counter Timers
    OC3BCounter(500); // set the counter timers

    DDRA |= (1 << PORTA0) | (1 << PORTA1) | (1 << PORTA2);      /* Make
trigger pin as output */
    PORTD |= (1 << PORTD6);     /* Turn on Pull-up */
    //DDRB &= ~(1 << PORTB7); // right engine turning left
    //DDRB &= ~(1 << PORTB6); // left engine  turning right


    sei();          /* Enable global interrupt */
    TIMSK1 = (1 << TOIE1);  /* Enable Timer1 overflow interrupts */
    TCCR1A = 0;     /* Set all bit to zero Normal operation */




    while(1)
    {
        distance = measureDistance();  /* distance from ultrasonic
sensor */
        if (distance < 15) /* stop the motor if the object is nearby */
        {
            PORTA |= (1 << PORTA2); // red light
            DDRB &= ~(1 << PORTB7); // right engine turning left
            DDRB &= ~(1 << PORTB6); // left engine  turning right

            PORTA &= ~(1 << PORTA1); // turn of blue light
            _delay_ms(500);
        }

        for (int x = 0; x < 29000; x++) /* for loop to delay the if any
ultra sound jumps back about and for loop delay is about 300ms */
        {
            if(distance >= 15 ) /* if object is in nearby the 20cm move
the move the car forward */
            {
                    move();
            }
            else{ /* otherwise break the loop to not delay */
                break;
            }
```

```
        }



    }
}

/* functions for  measure distance by ultrasonic sensors*/
double measureDistance(void){
    /* Give 10us trigger pulse on trig. pin to HC-SR04 */
    PORTA |= (1 << Trigger_pin);
    _delay_us(10);
    PORTA &= (~(1 << Trigger_pin));

    TCNT1 = 0;  /* Clear Timer counter */
    TCCR1B = 0x41;  /* Capture on rising edge, No prescaler*/
    TIFR1 = 1<<ICF1;    /* Clear ICP flag (Input Capture flag) */
    TIFR1 = 1<<TOV1;    /* Clear Timer Overflow flag */

    /*Calculate width of Echo by Input Capture (ICP) */

    while ((TIFR1 & (1 << ICF1)) == 0);/* Wait for rising edge */
    TCNT1 = 0;  /* Clear Timer counter */
    TCCR1B = 0x01;  /* Capture on falling edge, No prescaler */
    TIFR1 = 1<<ICF1;    /* Clear ICP flag (Input Capture flag) */
    TIFR1 = 1<<TOV1;    /* Clear Timer Overflow flag */
    TimerOverflow = 0;/* Clear Timer overflow count */

    while ((TIFR1 & (1 << ICF1)) == 0);/* Wait for falling edge */
    count = ICR1 + (65535 * TimerOverflow); /* Take count */
    /* 8MHz Timer freq, sound speed =343 m/s */
    return (double)count / 466.47;
}

/* functions for move the motors to follows the lines */
void move(void)
{
    PORTA &= ~(1 << PORTA2);
    PORTA |= (1 << PORTA1);
    if (( ( (PIND&(1 << PORTD5)) >> PORTD5) == 1 ) & (( (PIND&(1 <<
PORTD3)) >> PORTD3) == 1 ) & (( (PIND&(1 << PORTD4)) >> PORTD4) == 1))
    {
        DDRB &= ~(1 << PORTB7); // right engine turning left
```

```
        DDRB &= ~(1 << PORTB6); // left engine  turning right

        SetSpeedOC3A(390);
        SetSpeedOC3B(490);

        int rightTurn = 0;
        int right1 = 0;
        while ( (( (PIND&(1 << PORTD3)) >> PORTD3) == 1 ) & (turn ==
0)) // höger
        {
            DDRB = 0b01000000;
            rightTurn = 1;
            right1 = 1;

        }


        int leftTurn = 0;
        int left1 = 0;
        while ((( (PIND&(1 << PORTD5)) >>PORTD5) == 1) & (turn == 1))
// vänster
        {
            DDRB = 0b10000000;
            left1 = 1;
            lastturn = 1;

        }


        turn++;

        if (turn > 1 & lastturn ==1)
        {
            turn = 0;
        }


    }else if (( ( (PIND&(1 << PORTD5)) >> PORTD5) == 0 ) & (( (PIND&(1
<< PORTD3)) >> PORTD3) == 0 ) & (( (PIND&(1 << PORTD4)) >> PORTD4) ==
0))
    {
        if (right1)
        {
```

```
            while (( ( (PIND&(1 << PORTD5)) >> PORTD5) == 0 ) & ((
(PIND&(1 << PORTD3)) >> PORTD3) == 0 ) & (( (PIND&(1 << PORTD4)) >>
PORTD4) == 0))
            {
                DDRB = 0b01000000;
            }
        }else if (left1)
        {
            while (( ( (PIND&(1 << PORTD5)) >> PORTD5) == 0 ) & ((
(PIND&(1 << PORTD3)) >> PORTD3) == 0 ) & (( (PIND&(1 << PORTD4)) >>
PORTD4) == 0))
            {
                DDRB = 0b10000000;
            }
        }

        if (!left1 & !right1)
        {
            DDRB = 0b00000000;
            PORTA &= ~(1 << PORTA1);// lamp blue turn off
        }
        right1 =0;
        left1  =0;

    }
    else{

        PORTA &= ~(1 << PORTA1);// lamp blue turn off
        PORTA |= (1 << PORTA1);  // turn on blue lamp
        SetSpeedOC3A(490);
        SetSpeedOC3B(590);
        // hålla sig i linje
        if( ( ( (PIND&(1 << PORTD5)) >> PORTD5) == 1 ) & (( (PIND&(1 <<
PORTD3)) >> PORTD3) == 0 ) & (( (PIND&(1 << PORTD4)) >> PORTD4) == 0)
){ // gå fram
            DDRB = 0b11000000;
            right1=0;
            left1 =0;
        }
        if(( (PIND&(1 << PORTD3)) >> PORTD3) == 1 ){ // höger
            DDRB = 0b01000000;
            right1 = 1;
        }
```

```
        if(( (PIND&(1 << PORTD4)) >> PORTD4) == 1){ // vänster
            DDRB = 0b10000000;
            left1 = 1;
        }
        PORTA &= ~(1 << PORTA1);// lamp blue turn off
  }
}

/* functions for Counter timer 3A */
void OC3ACounter(int speed){


    OCR3B = 1023; // top value
    OCR3A = speed;
    TCCR3A |= (1 << COM3A1) | (1 << COM3A0) | (1<<WGM32) | (1<<WGM31) |
(1 << WGM30);
    TCCR3B |= (1 << CS30);
}

/* functions for Counter Timer 3B */
void OC3BCounter(int speed){


    OCR3B = speed; // top value 400 to 0
    TCCR3A |= (1 << WGM33) | (1 << WGM31) | (1 << COM3B1) ;
    TCCR3B |=   (1 << CS30);
}

/* functions for interrupt */
ISR(TIMER1_OVF_vect)
{
    TimerOverflow++;    /* Increment Timer Overflow count */
}

/* functions for change the Counter Timer width */
void SetSpeedOC3A(int speed){
    OCR3A = speed;
}

/* functions for Change counter Timer pulse width*/
void SetSpeedOC3B(int speed){
    OCR3B = speed;
}
```

```
/* functions To stop all counter Timers*/
void stop(void){
    DDRB = 0x00;

    OCR0A, OCR0B, OCR3B, OCR3A = 0;


    TCCR0A ,TCCR0B ,TCCR3A ,TCCR3B = 0;
}
```
