Radiostyrd Bil

2023-05-18

Digitala System Projekt – EITA15

Grupp 18 medlemmar:

Kais Atta Torkmani, Josef Sajit, Leonard Lundberg, Peter Kyngäs Wallin, Hoang Anh Quan Vo, Hoang Minh Quan Vo

Handledare:

Bertil Lindvall, Lars-Göran Larsson

Abstract

This report provides a comprehensive overview of the intricate process involved in constructing a radio-controlled car. The culmination of six weeks of dedicated brainstorming, meticulous designing, coding and rigorous testing by a team of six members, this product is the complete integration of their diverse skill and expertise. We utilized two Atmega 1284 Microcontrollers and two 433 Mhz Transceivers as the primary components of the final product. These components are used for controlling and receiving signals respectively.

1. Inledning

1.1 Bakgrund

I kursen EITA15 Digitala System har studenter ett projekt där de ska bygga en digital produkt. Vår grupp har valt att bygga en radiostyrd bil. Projektet har som mål att konstruera en prototyp med basen på mikroprocessorn ATmega1284.

1.2 Syfte

Syftet med detta projekt har varit att erhålla en djupare insikt i principerna bakom elektroniska apparaters funktion, samt att utforska utvecklingsprocessen och programmering i C-språket. Genom att bygga en radiostyrd bil i enlighet med de specifikationer som presenteras i kommande avsnitt har projektet ämnat att ge en fördjupad förståelse.

Syftet med denna rapport är att presentera för läsaren en detaljerad överblick över framstegen i konstruktionen av den radiostyrda bilen. Rapporten kommer att beskriva den använda hård- och mjukvaran, identifiera och redogöra för de utmaningarna som uppstod samt beskriva de lösningsstrategierna som tillämpades. Rapporten syftar också till att ge en ingående inblick i arbetsprocessen och hur den utvecklades över tid.

1.3 Målformulering

Målet med detta projekt är att skapa en enkel radiostyrd bil som kan köra framåt, bakåt och rotera både höger och vänster. Man ska även, utan större problem, kunna köra bilen längs en testbana som består av några svängar och hinder.

1.4 Problemformulering

- Hur ska arbetet fördelas?
- Hur ska konstruktionen se ut?
- Vilka komponenter ska användas i konstruktionen?
- Vilka program ska användas?
- Hur ska den programmeras?

1.5 Avgränsningar

Eftersom detta endast är ett test av gruppens förmåga att använda olika komponenter ihop samt att slå ihop hårdvara med mjukvara så är utseendet inte högt prioriterat. Det finns även begränsningar i vilka komponenter som kan användas, och därmed är inte konstruktionen helt optimal.

2. Teknisk bakgrund

2.1 Kravspecifikationer

- Bilen ska kunna köra rakt.
- Bilen ska kunna backa.
- Bilen ska kunna svänga både höger och vänster.
- Bilen ska kunna styras med hjälp av kontrollen via radiovågor.

2.2 Hårdvara

- Två dc-motorer användes. Motorerna användes för att driva bilen genom att omvandla elektrisk energi till mekanisk rörelse.

- En spänningsregulator 7805CV: Denna komponenten användes för att stabilisera och reglera spänningen till olika delar av systemet. Den bidrog till att säkerställa en korrekt och konstant spänning av 5v för de olika elektroniska komponenterna.

- Två 433 MHz transceivers. En transceiver installerades i bilen och den andra i kontrollenheten. Dessa komponenter möjliggjorde en trådlös kommunikationslänk mellan bilen och kontrollenheten. De användes för att skicka samt ta emot styrkommandon och signaler.

- Fyra hjul. Bilens rörelse och manövrering åstadkoms genom användningen av fyra hjul.

- Elva batterier. För att förse bilen och kontrollenheten med tillräcklig strömförsörjning användes totalt elva batterier, fyra till kontrollen och sju till bilen.

- Fyra knappar för styrning av bilen via kontrollenheten.

- Två kretskort till både bilen samt kontrollenheten. Dessa kretskort fungerade som centrala enheter för att styra och koordinera olika funktioner i systemet. Kretskorten till både bilen samt kontrollenheten innehöll en Atmega1284 mikrokontroller, som användes för att hantera data och styra bilens rörelse.

- En L298N H-brygga. Denna komponenten användes för att styra och reglera motorerna. H-bryggan tillåter att motorerna kan rotera i olika riktningar och styras med precision.

- 3 lysdioder installerades även för att övervaka strömmen och signalerna, dessa lysdioder gav visuell återkoppling om status och funktion av systemet.

2.3 Mjukvara

För att möjliggöra programmering av processorer i både bilen och kontrollen har datorprogrammet Atmel Studio 7 använts. Dessutom användes programmet Fusion 360 för att skapa kopplingsschemat för systemet. Koden utvecklades i programspråket C, och överföring av kod från dator till bilen utfördes med hjälp av en J-Tag enhet.

För att sammanfatta systemets funktionalitet och styrning, drivs bilen av en While-loop som kontinuerligt väntar på styrkommandon från kontrollenheten. I kontrollens kod har varje knapptryckning tilldelats en specifik kod, till exempel representerar kode “B_LEFT” en vänsterriktning. När en knapptryckning sker skickas en signal till processorn som identifierar vilken port som aktiverats. Via transcievern kan koden sedan överföras för att utföra det kommandot som blivit tilldelat från kontrollen.

3. Metod

3.1 Planering

Innan konstruktionen kunde påbörjas behövdes plan, design samt fördelning av arbetet bestämmas. Då gruppen består av sex medlemmar behövdes en konstruktion som skulle medföra tillräckligt med arbete så att alla kunde delta. Det poängterades att en radiostyrd bil skulle fungera utmärkt, då man behöver konstruera både en bil och en kontroll. På detta sätt kunde arbetet enkelt delas upp så att tre medlemmar arbetade med kontrollen och resten med bilen. Gruppen kom sedan överens om de olika krav som ställdes på bilen och kunde därefter börja fundera på hur den skulle se ut samt vilka komponenter som behövdes.

Då bilen ska kunna köra i många olika riktningar behövdes något sätt att svänga bilen. Första idén var att använda en servomotor som kunde kontrollera i vilken riktning framhjulen stod. Problemet med detta var att bilen kunde bli svårmanövrerad i små utrymmen. I stället bestämdes det att bilen skulle rotera genom att driva ena sidan av bilen framåt samtidigt som den andra sidan drivs bakåt. Med denna metod kan bilen rotera höger och vänster utan att flytta sig från den punkt den befinner sig på. För att bilen skulle bli mer stabil bestämdes det också att larvfötter skulle användas, då dessa ger mer yta som får kontakt med golvet. Dessa larvfötter behövde drivas på något sätt och därav kom idén att använda två motorer som kunde rotera i vardera riktning. Motorerna skulle sedan kontrolleras av en h-brygga.

För att ge instruktioner till bilen behövdes en separat konstruktion, en kontroll. Då bilen endast ska köra framåt, bakåt eller rotera höger och vänster så passade det med en knapp för varje funktion, totalt fyra knappar. Kontrollen måste även kunna få kontakt med bilen så att bilen vet hur den ska köra. En transceiver åt vardera konstruktion löste detta problem. Då en knapp trycks ned ska en siffra skickas från kontrollen till bilen. Denna siffra bestämmer hur bilen ska köra. Ett betyder att den ska åka framåt. Två betyder bakåt. Tre, rotera vänster. Fyra, rotera höger. Om ingen knapp trycks ned så ska bilen inte köra alls och då skickas en nolla.

3.2 Montering av hårdvara

När planeringen var färdig och komponenterna hade hämtats kunde konstrueringen börja. Eftersom komponenterna i sig inte var väldigt komplicerade förutom transcievern, bestämdes det att de skulle kopplas direkt till mikrokontrollern, för att sedan med den kunna testa komponenterna var för sig. Det användes både lödning och direkt koppling med kablar för att koppla ihop alla delar, vilket tog upp en stor del av tiden då kablar gick sönder ibland och behövdes bytas ut. För att testa komponenterna behövdes det först skrivas några få rader enkel programmering, som enbart testade funktionen hos de olika delarna.

3.3 Programmering av mjukvara

För att programmera de båda komponenterna behövdes först en exakt plan för hur bilen ska agera samt hur den ska styras av kontrollen. Då det bestämdes att siffror skulle skickas av kontrollen och sedan avkodas av bilen blev själva programmeringen ganska okomplicerad. Om en knapp skickar signal till mikrokontrollern så skickas en siffra från mikrokontrollern till transceivern. Vilken siffra beror på vilken knapp som detekterats. Då dessa moment har tagits upp i liknande format tidigare i kursen fanns redan en viss kunskap i hur man hanterar dessa problem. Det behövdes många tester för att få allting att fungera som det skulle. C-språket användes i programmeringen, som tog plats i Atmel Studio 7.

4. Analys

H-bryggan som användes visade sig ha ett märkbart spänningsfall då motorerna skulle drivas. Då detta var fallet behövdes fler batterier. Men då transceivern och mikrokontrollern har en maxgräns på cirka 5 V, behövdes spänningen dämpas något innan den nådde dessa komponenter. Detta löstes med en spänningsregulator som klarar av att omvandla upp till 35 V ner till 5 V.

Det uppmärksammades även att om bilen var igång utan att kontrollen sattes på så tog transceivern emot störningar som fick bilen att röra sig. Detta löstes genom att sätta ett slags lås innan kör-funktionerna kunde nås i källkoden. Bilen väntar nu på att kontrollen skickar numret 136 innan den accepterar de andra siffrorna som driver bilen.

Knapparna behövde även resistans seriekopplat efter sig till jorden för att se till att signalen från knappen verkligen var noll när knappen ej var nedtryckt. Annars kunde de ge felaktiga värden.

5. Resultat

Resultatet blev en fullt fungerande radiostyrd bil som kan åka framåt, bakåt och rotera både vänster och höger. Den kan manövrera sig runt en bana utan problem. Signalerna går fram tydligt och snabbt och bilen agerar väldigt precist. Allting fungerar som önskat.

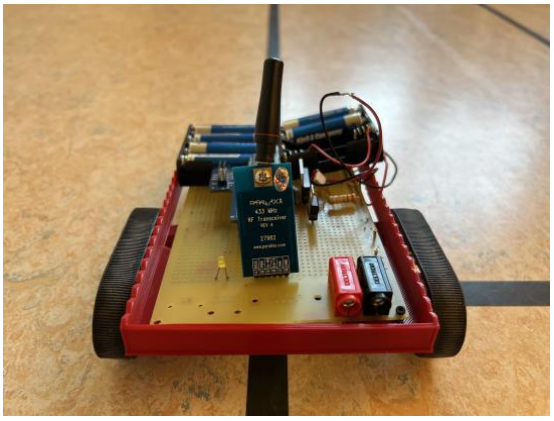


Bild 1. Färdig bil bakifrån

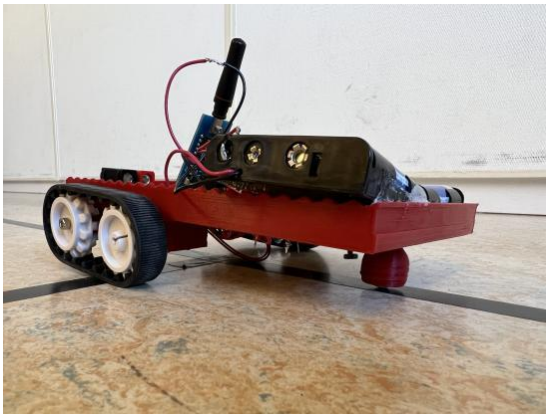


Bild 2. Färdig bil från sidan

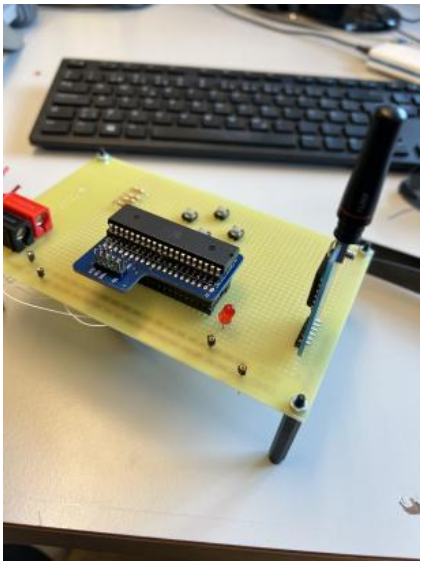


Bild 3. Kontrollen

6. Slutsats

Alla krav som ställdes i början blev uppfyllda. Den kör som den ska och klarar att köra en bana utan problem. Förkunskaper inom dessa områden har varit till stor hjälp, även om det inte alltid har räckt till. Projektet visade vikten av att välja lämpliga komponenter och verktyg för att uppnå önskade funktioner och prestanda. Sammanfattningsvis illustrerar detta projektet den tekniska kompetensen och förmågan att använda moderna verktyg för att bygga en radiostyrd bil. Genom att använda sig av olika komponenter, programmeringsspråk och trådlös kommunikation har en elektronisk apparat skapats, det i sin tur har gett en fördjupad förståelse för elektronik, programmering och systemintegration.

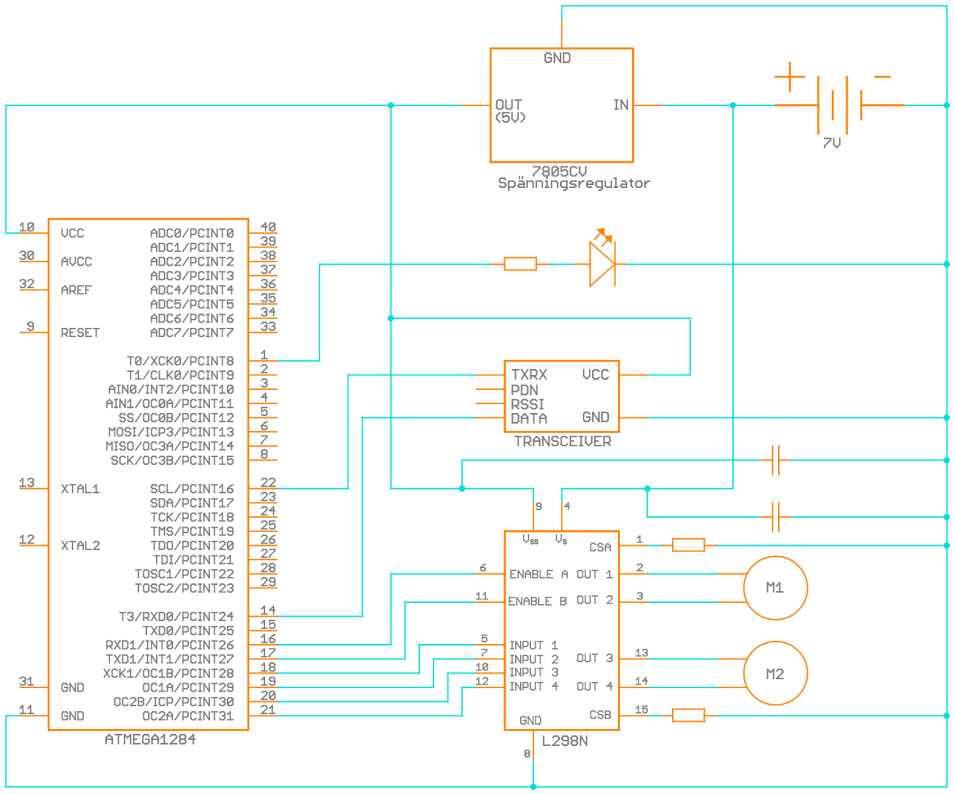
7. Källförteckning

[1] Atmel, *8 bit AVR microcontrollers,* ATmega1284, datablad, 2016
https://www.eit.lth.se/fileadmin/eit/courses/datablad/Processors/ATmega1284.pdf
[2] STMicroelectronics, *DUAL FULL-BRIDGE DRIVER,* L298N, datablad, 2000
https://www.elfa.se/Web/Downloads/t_/en/ofL298N_647108_dat_en.pdf
[3] STMicroelectronics, *Positive voltage regulators,* L78xx-L78xxC, datablad, 2008

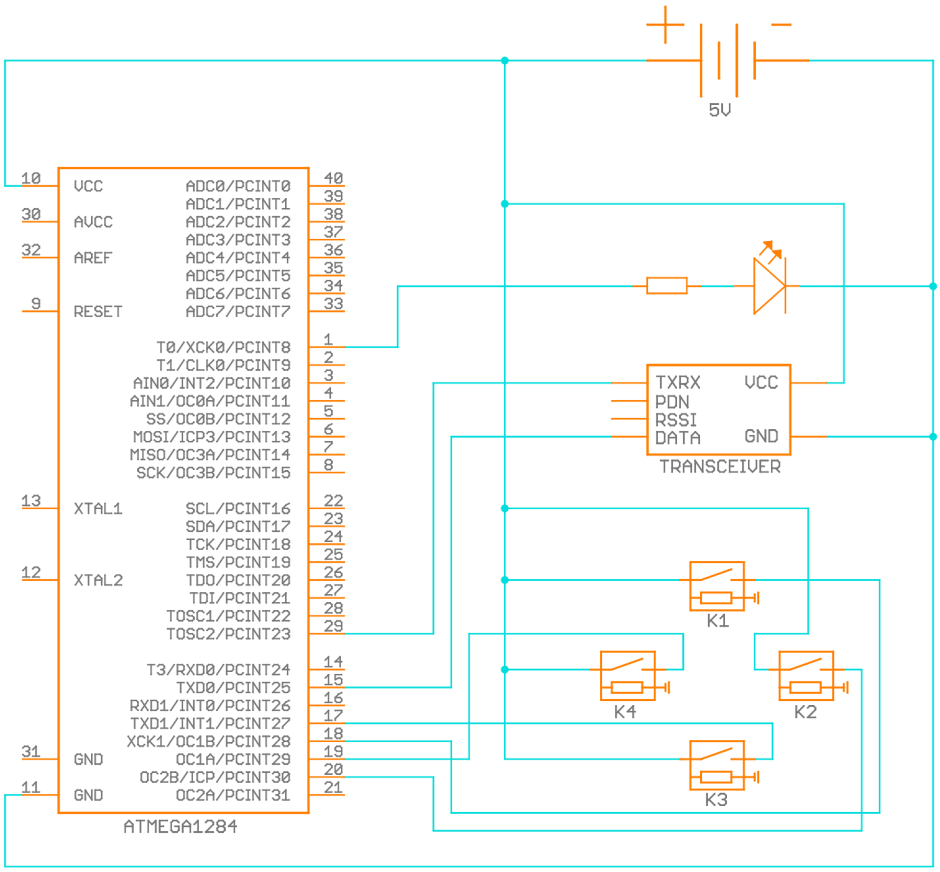
https://www.elfa.se/Web/Downloads/xc/_e/teL78xx-L78xxC_e.pdf

8. Appendix

Bilaga 1: Kopplingsschema för bilen



Bilaga 2: Kopplingsschema för kontrollen



Bilaga 3: Källkod för kontroll

```
#include <avr/io.h>
/*
* Definierar alla knappars pins och register så
* att koden blir lättare att förstå.
*/
#define B_UP (PIND & (1 << 4))
#define B_DOWN (PIND & (1 << 3))
#define B_LEFT (PIND & (1 << 5))
#define B_RIGHT (PIND & (1 << 6))

void usart0_init();
void pin_init();
void usart0_transmit(uint8_t);
```

```
uint8_t button_read_reliably_up();
uint8_t button_read_reliably_down();
uint8_t button_read_reliably_left();
uint8_t button_read_reliably_right();

int main(void)
{
	usart0_init();
	pin_init();
	while (1)
	{
		/*
		* Skickar en nyckel (136) som "låser upp" bilen
		*/
		usart0_transmit(136);
		/*
		* Om någon av knapparna är nedtryckta så skickas
		* en siffra (1 = fram, 2 = bak, 3 = vänster, 4 = höger).
		* Är ingen knapp nedtryckt skickas 0. Siffran avkodas
		* sedan i bilens processor. En LED-lampa tänds även
		* så länge ett knapptryck registreras.
		*/
		if (button_read_reliably_up() != 0)
		{
			PORTA = 1;
			usart0_transmit(1);
		} else if (button_read_reliably_down() != 0)
		{
			PORTA = 1;
			usart0_transmit(2);
		} else if (button_read_reliably_left() != 0)
		{
			PORTA = 1;
			usart0_transmit(3);
```

```
            } else if (button_read_reliably_right() != 0)
            {
                  PORTA = 1;
                  usart0_transmit(4);
            } else
            {
                  PORTA = ~1;
                  usart0_transmit(0);
            }
      }
}

/*
* Initiering av alla pins på ATMEGA1284
*/
void pin_init()
{
      DDRA |= 1;
      DDRC |= (1 << 7);
      DDRD |= 0;
      PORTC = (1 << 7);
}

/*
* Initiering av USART
*/
void usart0_init()
{
      UCSR0B |= 0x08;
      UCSR0C |= 0x03;
      UBRR0 = 103;
}

/*
```

```
* Sänder data
*/
void usart0_transmit(uint8_t data)
{
	while (!(UCSR0A & (1<<UDRE0)));
	UDR0 = data;
}

/*
* button_read_reliavly_up(_down,_left,_right)
* kollar om en knapp är nedtryckt och skickar
* isåfall en etta. Annars skickar den en nolla.
*/
uint8_t button_read_reliably_up()
{
	if (B_UP)
	{
		_delay_ms(60);
		return (B_UP != 0);
	}
	return 0;
}

uint8_t button_read_reliably_left()
{
	if (B_LEFT)
	{
		_delay_ms(60);
		return (B_LEFT != 0);
	}
	return 0;
}

uint8_t button_read_reliably_right()
```

```
{
	if (B_RIGHT)
	{
		_delay_ms(60);
		return (B_RIGHT != 0);
	}
	return 0;
}

uint8_t button_read_reliably_down()
{
	if (B_DOWN)
	{
		_delay_ms(60);
		return (B_DOWN != 0);
	}
	return 0;
}
```

Bilaga 4: Källkod för bil

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

uint8_t usart0_receive();
void usart0_init();
void pin_init();
void forward();
void reverse();
void turn_left();
void turn_right();
void stop();

int main(void)
```

```
{
	pin_init();
	usart0_init();
	uint8_t key = 0;
	uint8_t press_recieved = 0;
	while (1)
	{
		PORTD |= (1 << 2);
		PORTD |= (1 << 3);
		key = usart0_receive();
		/*
		* Kollar om nyckeln har tagits emot och startar
		* endast då. Detta för att undvika störningar.
		*/
		if (key == 136) {
			press_recieved = usart0_receive();
			/*
			* Läser av press_recieved som innehåller det som
			* tagits emot av kontrollen. 1 = fram, 2 = back,
			* 3 = vänster, 4 = höger. Allt annat leder till stopp.
			* En LED-lampa tänds även så länge bilen kör i någon
riktning.
			*/
			if (press_recieved == 1)
			{
				PORTA = 1;
				forward();
			} else if (press_recieved == 2)
			{
				PORTA = 1;
				reverse();
			} else if (press_recieved == 3)
			{
				PORTA = 1;
```

```
                        turn_left();
                  } else if (press_recieved == 4)
                  {
                        PORTA = 1;
                        turn_right();
                  } else
                  {
                        PORTA = ~1
                        stop();
                  }
            }
      }
}

/*
* Initering av USART
*/
void usart0_init() {
      UCSR0B |= 0x10;
      UCSR0C |= 0x03;
      UBRR0 = 103;
}

/*
* Initiering av alla pins på ATMEGA1284
*/
void pin_init() {
      DDRA |= 0X01;
      DDRC |= 0x01;
      DDRD |= 0;
      PORTD |= (1 << 3);
      PORTD |= (1 << 2);
      PORTC |= 0;
}
```

```
/*
* Tar emot data och returnerar denna
*/
uint8_t usart0_receive()
{
	while (!(UCSR0A & (1 << RXC0)));
	return UDR0;
}

/*
* Skickar signal till h-bryggan som
* i sin tur skickar signal till motorerna
* som för bilen framåt.
*/
void forward()
{
	PORTD |= (1 << 7);
	PORTD &= ~(1 << 6);
	PORTD |= (1 << 4);
	PORTD &= ~(1 << 5);
}

/*
* Skickar signal till h-bryggan som
* i sin tur skickar signal till motorerna
* som sedan för bilen bakåt.
*/
void reverse()
{
	PORTD |= (1 << 5);
	PORTD &= ~(1 << 4);
	PORTD |= (1 << 6);
	PORTD &= ~(1 << 7);
```

```
}

/*
* Skickar signal till h-bryggan som
* i sin tur skickar signal till motorerna
* som sedan vänder bilen åt vänster.
*/
void turn_left()
{
      PORTD &= ~(1 << 7);
      PORTD &= ~(1 << 6);
      PORTD |= (1 << 5);
      PORTD |= (1 << 4);
}

/*
* Skickar signal till h-bryggan som
* i sin tur skickar signal till motorerna
* som sedan vänder bilen åt höger.
*/
void turn_right()
{
      PORTD |= (1 << 6);
      PORTD |= (1 << 7);
      PORTD &= ~(1 << 4);
      PORTD &= ~(1 << 5);
}

/*
* Skickar signal till h-bryggan som
* i sin tur skickar signal till motorerna
* som sedan stoppar bilen.
*/
void stop()
```

```
{
        PORTD &= ~(1 << 6);
        PORTD &= ~(1 << 7);
        PORTD &= ~(1 << 5);
        PORTD &= ~(1 << 4);
}
```