Projektrapport Digitala system EITA15

“Bilen som inte kan krocka”

Gruppmedlemmar: Oscar Skarler, Jonathan Persson, Erik Schiemann och Mohammed Abou Naasa

vt 2019

Sammanfattning

Inom kursen Digitala system har vi genomfört ett projektarbete, där syftet var att tillverka en prototyp, som är baserad på mikroprocessorn AT Mega 1284. Vi valde att bygga en bil med en ultraljudssensor placerad fram på bilen. Målet var att få bilen att köra rakt fram tills den upptäckte ett hinder/en vägg framför sig, då svänga åt sidan och sedan fortsätta i den nya riktningen. Projektet genomfördes genom att göra en planering, välja ut lämpliga komponenter, rita ett kopplingsschema, programmera processorn, bygga ihop bilen, testa och revidera. Resultatet av arbetet blev en bil som kunde köra med bestämd hastighet rakt fram, eller upptäcka hinder via ultraljudssensorn. Projektgruppen har förutom det praktiska arbete med bilen lärt sig att programmering kan ta längre tid än planerat, att det är viktigt att stämma av planeringen med samtliga gruppmedlemmar och att fler hållpunkter i en planering kan vara bra att ha.

Innehållsförteckning

1 Inledning

Detta projekt genomförs inom kursen Digitala system. I projektet görs en valfri prototyp, kopplingsschema ritas, programmering görs, prototypen byggs samt testas. Felsökning och revideringar testas, och hela projektet redovisas genom en skriftlig rapport, en hemsida samt muntlig redovisning. Som prototyp valdes en självkörande bil med en avståndsmätare i form av ultraljudssensor.

1.1 Syfte

Syftet med detta arbete är att tillverka en prototyp av något slag, som är baserad på mikroprocessorn AT Mega 1284. Prototypen ska fås att fungera med hjälp av programmering. Projektarbetet är också en träning i att genomföra en uppgift i en mindre grupp, och att kunna redovisa ett gemensamt projekt.

1.2 Målformulering

Projektgruppen valde att bygga en bil, som när den kör framåt, ska kunna upptäcka något större föremål framför sig och då svänga åt sidan, för att fortsätta i denna riktning. För att upptäcka föremål framför bilen, användes en avståndsmätare i form av en ultraljudssensor. Målet är att kunna göra en ritning, bygga ihop komponenter enligt ritningen, programmera mikroprocessorn så att bilen kan köra framåt och upptäcka hinder och då svänga åt sidan.

1.3 Problemformulering

Prototypen, bilen, ska kunna:

1. Startas då power switchen slås på.
2. Bilen kör tills den upptäcker en vägg/hinder, via ultraljudssensorn.
3. Bilen ska kunna svänga åt sidan, och fortsätta i den nya riktningen.

1.4 Motivering av projektarbetet

Valet av att bygga en bil med inte allt för många, olika funktioner gjordes utifrån gruppmedlemmarnas kompetens, intresse och tidsramen för projektet inom kursen.

1.5 Avgränsningar

Avståndsmätaren (HC-sr04) kan fästas på en servomotor för att få en större vinkel på avståndsmätningen. Projektgruppen valde dock att inte göra detta, utan fäste mätaren direkt på bilens front.

2 Teori

Följande komponenter användes för att bygga bilen:

ATmega 1284
En 8-bitars mikroprocessor som programmerades för att få övriga komponenter att utföra önskade funktioner.

DRV8833 Dual Motor Driver Carrier
H-brygga som användes för att styra motorerna. Bryggan har två ingångar/utgångar per motor, vilka kan styras var för sig. Motorerna kan fås att snurra åt olika håll beroende på om signalen som skickas till bryggan är låg/hög eller hög/låg, vilket gör att man kan styra bilen.

HC-sr04
En ultraljudssensor som användes för att bestämma avstånd. Sensorn sattes fast längst fram på bilen. Sensorn fungerar genom att skicka ut ljudvågor och registrerar de vågor som studsar tillbaka, varpå tidsskillnaden används för att beräkna avståndet.

JTAG
Användes för att föra in kod i processorn.

Lawicel Zumo Chassis Kit
Bilens chassi med 2 stycken inbyggda DC-motorer, för att driva bilen framåt.

Strömkälla
Fyra stycken 1,5 V batterier användes för att förse komponenterna med ström.

Strömbrytare
En strömbrytare för att slå på och av strömmen.

Mjukvara
Programmering gjordes med hjälp av Atmel studio 7, i programmeringsspråk C (se appendix). Programmeringen gjordes stegvis; motorerna drivs genom pulsbreddsmodulering (PWM), där bredden på vågorna ger spänningens storlek och därmed hastigheten på bilen. Ultraljudssensorn startar när den får en puls och samtidigt startar en timer. Timern stoppar när sensorn skickar tillbaka en signal då den tagit emot den studsande ljudvågen. Tiden används för att beräkna avstånd till ett hinder/vägg.

3 Metod

Projektarbetet har omfattat olika faser. Först gjordes en planering, där idén till att bygga någon slags bil bestämdes. En kravspecifikation på komponenter togs fram, och efter att ha fått information via datablad, så ritades ett kopplingsschema. Viss revidering gjordes efter handledarens förslag; vissa komponenter byttes ut och nytt kopplingsschema ritades (se appendix).

Efter att kopplingsschemat godkänts, så påbörjades konstruktionen. Samtliga komponenter sattes fast på ett mönsterkort. Kopplingen mellan komponenterna gjordes med koppartråd som virades runt pinnarna. Två hål gjordes i mönsterkortet, som sattes fast på chassits två gängade metallstavar. Avståndssensorn sattes fast längst fram på bilen.

Sedan kodades processorn i programspråk C. Genom kunskaper från kursens tidigare laborationer och genom att söka information om respektive komponent, så programmerade projektgruppen stegvis för att få bilen att uppfylla målbeskrivning.

4 Resultat

Komponenterna byggdes ihop enligt kopplingsschemat, programmeringen genomfördes och bilen testades ett antal gånger. Kodningen tog mer tid är vad projektgruppen hade planerat, men gruppen lyckades få bilen att köra framåt; det går att bestämma hastigheten på motorerna med hjälp av pulsbreddsmodulering. Ultraljudssensorn skickar och tar emot ljudvågor, och signalen till processorn hanteras med hjälp av avbrott, och avståndet beräknas. Sensorn fungerar för sig, bilen kör framåt för sig, men båda delar har inte fåtts att fungera samtidigt.

5 Slutsats

Efter detta projekt kan vi konstatera att bilen kan:
Starta då power switchen slås på.
Bilen kan köra med en bestämd hastighet.
Bilen kan upptäcka en vägg/ett hinder via ultraljudssensorn, och då räkna ut avståndet.
Bilen kan inte köra och använda ultraljudssensorn samtidigt.
Bilen kan ännu inte svänga åt sidan.

5.2 Framtida utvecklingsmöjligheter

Projektgruppen valde att inte fästa avståndsmätaren (HC-sr04) på en servomotor. Detta hade gruppen kunnat göra för att för att få en större vinkel på avståndsmätningen, så att bilen upptäcker hinder även på stora infallsvinklar. Vid stora infallsvinklar kommer inte ljudvågorna att studsa tillbaka och registreras om sensorn sitter fast. Gruppen valde nu att fästa mätaren direkt på bilens front.

När det gäller arbetsprocessen, har gruppen lärt sig att det behövs hållpunkter där processen utifrån tidsplaneringen checkas av, så att alla gruppmedlemmar har hunnit med. Om man inte har hunnit med, är det viktigt att kunna be om hjälp så fort som möjligt, så att övriga i gruppen kan hjälpa till med arbetsuppgifter.

6 Källförteckning - datablad

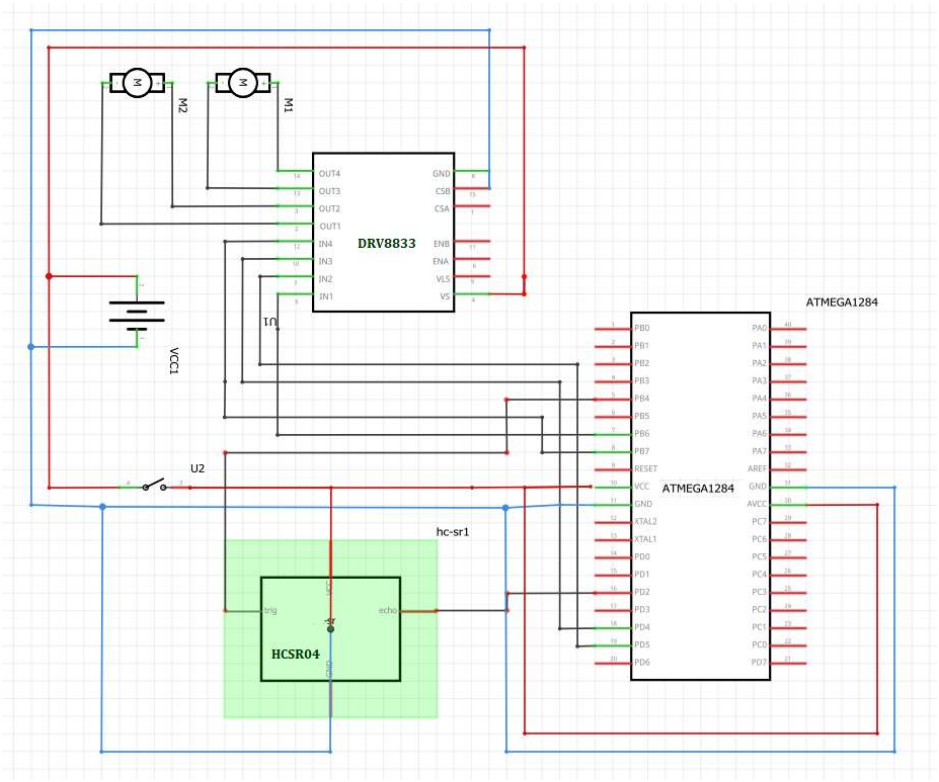
AT Mega 1284: https://www.microchip.com/wwwproducts/en/ATmega1284

H-brygga: https://www.pololu.com/product/2130

Ultraljudssensor: https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf

7 Appendix

Kopplingsschema för bilen:



Kod:

PWM:

```
#include <avr/io.h>
void timer3_init(){
    TCCR1A |= (1 << WGM11) | ( 1 << COM1A1 );
    TCCR1B |= (1 << WGM13) | (1 << WGM12 ) | (1 << CS12) | (1 << CS10);
```

```
// ERSÄTT KODEN HÄR NEDAN MED TCCR3x

    TCCR2A |= (1 << COM2A1) | (1 << WGM20) | (1 << WGM21);
    TCCR2B |= (1 << WGM22) | (1 << CS20) | (1 << CS21) |(1 << CS22);
}
void set_pulse(uint16_t verb){
    OCR1A |= verb;
          OCR1B |= verb;

}
void set_period(uint16_t adj){
    ICR1 |= adj;
}




int main(void)
{
    timer3_init();
    set_pulse(156);
    set_period(200);
    DDRD |= (0b00000000);
    /* Replace with your application code */
    while (1)
    {
    }
}
```

SENSORN:

```
 #define F_CPU 16000000UL
 #include <avr/io.h>
 #include <util/delay.h>
void ul_init();
uint16_t us_measurement();
void timer3_init();
void set_pulse(uint16_t pulse);
void set_period(uint16_t period);

uint16_t dist;

int main(void) {
     ul_init();
     timer3_init();
     set_period(0xFFFF);
      while (1) {
          dist = us_measurement();
          set_pulse(dist);
           _delay_ms(60);
```

```
          }
           }
void ul_init(){
    DDRB |= (1 << PINB4);
    }

uint16_t us_measurement(){
   PORTB |= (1 << PINB4);
    _delay_us(10);
    PORTB &= ~(1 << PINB4);
     while(!(PIND & (1 << PIND2)));
     TCCR2B |= (1 << CS20);
     TCNT2 = 0;
      while((PIND & (1 << PIND2)));
      TCCR2B &= ~(1 << CS20);
       return TCNT2;
       }
void timer3_init() {
     DDRB |= (1 << PORTB6);
     TCCR3A |= (1 << COM3A1) | (1 << WGM31);
     TCCR3B |= (1 << WGM33) | (1 << WGM32) | (1 << CS30);
      }
void set_pulse(uint16_t pulse){
    OCR3A = pulse;
    }
void set_period(uint16_t period){
   ICR3 = period;
}
```