Digitala projekt

Henrik Lundstedt d02hl@efd.lth.se
Gustav Darpö d02gd@efd.lth.se

2006-05-22

Abstract

Our goal with this project was to build a robot, capable of avoiding obstacles which intersects its route. Parts of the construction consist of LEGO, mainly for the steering and chassi. The rest is made up of analog electronics on a circuit board with an AVR ATmega16 processor as the main component. Two LEGO motors are used for drivning the robot. In order to observe obstacles the robot is equipped with an infrared distance meter. The sensor is mounted on a servo which sweeps the area in front of the vehicle. This approach makes it possible for the robot to observe objects not only straight ahead but also to the left and to the right. Based on the results of the measurements of the IR-sensor the motors are assigned appropriate control power. The final result is a robot which manages to avoid obstacles quite good. Problems can sometime arise when the robot for some reason gets too close to an object. This is due to the natural noise that arise when measuring the IR-signal where the distance is very short. In conclusion the robot behaves pretty much as expected and given more time further improvements can be done.

Contents

1 Inledning

Målsättningen med vårt projekt var att bygga ett fordon som kan ta sig fram utan att krocka med hinder som påträffas på vägen. Uppgiften består av tre huvuddelar: design, bygge och programmering. I design-fasen görs en planering av hur konstruktionen ska se ut, hur de olika delarna ska samverka och hur de ska kopplas samman. Bygget innebär i huvudsak konstruktion av hårdvaran i analog elektronik men även de rent mekaniska delarna som till exempel banddriften i vårt fall. Slutligen krävs programmering av processorn för att logiskt sammanbinda de olika delarna i konstruktionen.

2 Teori

2.1 Processor

Vi använde en AVR ATmega16 som processor för vår robot. Det är en 8-bitars mikrokontroller med RISC-arkitektur och en klocka på upp till 8 MHz. Den har 1 kb internt SRAM samt 16 kb Self-Programmable Flash. AVR ATmega16 är som sagt inte enbart en processor utan en mikrokontroller dvs. en komplett dator som fungerar självständigt utan behov av påbyggnader. Processorn har 40 pinnar varav 32 kan användas för in- och utmatning av data. Den har även inbyggd timer och 10-bitars AD-omvandlare. Till roboten använde vi en 8-bitars timer, AD-omvandlaren och ca 10 procent av minnet. Timern ställde vi in så att vi hade en upplösning på 1/20 ms, vilket var tillräckligt för att positionera servon med stor noggrannhet. AD-omvandlaren genererar ett avbrott när en omvandling är klar.

2.2 Avståndsmätare

Avståndsmätaren användes för att mäta avstånd till objekt roboten bör undvika. Den klarar avstånd upp till 80 cm vilket var fullt tillräckligt för algoritmen roboten använder för att undvika hinder. Avstånd mäts genom att en IR-sändare skickar infrarött ljus, startar en timer och tar emot ljuset som studsar tillbaka. Timern stoppas och avståndet kan bestämmas. Avståndsmätaren har två sladdar för strömförsörjning och en för en analog signal som representerar avståndet till det objekt den pekar på. Signalen varierar omvänt linjärt med avståndet till objektet, och är som störst när objektet är nära. Vi upptäckte att när avståndet till objektet var alltför nära introducerades allvarligt brus som var så stort att undvikningsalgoritmen havererade. Vi blev därför tvungna att undvika att köra så nära väggar att detta problem uppenbarade sig.

2.3 Servo

För att svepa fram och tillbaka med avståndsmätaren i en vinkel framför roboten, använde vi en servo. Utöver strömförsörjning har en servo enbart en sladd för styrning. Genom den sladden skickas pulser till servon för att positionera den i olika vinklar. Pulslängden brukar variera från 1-2 ms motsvarande 0-120 grader. Vinklar däremellan fås genom en linjär mappning av 1-2 ms. Pulserna har en period av 10-30 ms, för att servon aktivt ska reglera vinkeln och inte hamna i viloläge.

3 Konstruktion och genomförande

Konstruktionen består av två huvuddelar. Till att börja med en bottenplatta byggd i LEGO som innehåller två LEGO-motorer, kopplade via kugghjul till drivband som används för att köra och styra roboten. Ovanpå denna detta vilar kopplingsplattan med elektronik i form av processor och tillhörande in/ut-enheter.

Arbetet med konstruktionen utfördes etappvis. Till att börja med anslöts en motor till processorn och ett enkelt program skrevs för att experimentera med pulsning av motorn, det vill säga köra den i olika hastigheter. På samma sätt utvigades programmet med nya tester allteftersom nya enheter anslöts. När slutligen alla enheter fungerade tillfredställande skrevs en algoritm för att styra roboten.

Grunden i den färdiga konstruktionen utgörs av en AVR ATmega16-processor. Till denna finns kopplad i första hand de tidigare nämnda lego-motorerna för styrning. Styrning sker genom att en av motorerna körs långsammare än den andra, alternativt körs i motsatt riktning. Genom banddriften kan roboten svänga runt på stället utan att behov av svängrum. Mjuka svängar kan göras genom att köra motorerna med olika hastighet. För att åstadkomma pulsning av motorerna och för att bestämma körriktning används en så kallad H-brygga (L298). Till den kan man koppla två motorer och för varje motor bestämma om den ska vara på eller av samt körriktning. Genom att pulsvis slå på och av motorerna och variera pulslängden kan hastigheten regleras.

För att att undvika hinder krävs funktionalitet för att upptäcka sådana. Detta är implementerat i form av av en avståndsmätare som använder sig av IR-ljus. Avståndsmätaren finns monterad på ett servo i främre delen av roboten som kontinuerligt sveper över området framför. Algoritmen för att styra roboten är uppbyggd kring denna svepning. Med jämna intervall vrids servot ett litet steg och en avläsning av IR-sensorn utförs. AD-omvandlaren genererar ett avbrott när en omvandling är klar, och servon positioneras i en ny vinkel. Totalt sju steg från vänster till höger görs under en svepning. Därefter noteras i vilken position servot befann sig när den påträffade det närmaste förmålet samt avståndet till det. Med hjälp av denna information beräknas styrsignalen, det vill säga pulslängden för respektive motor. Är avståndet väldigt kort och om vinkeln indikerar att objektet upptäcktes föröver vrids roboten maximalt. I annat fall sker en pulsning som bestäms av avståndet. Även vinkeln i vilket förmålet befinner sig påverkar: En liten vinkel innebär en kraftigare vridning och tvärtom.

Flera av enheterna i konstruktionen är avsedda för en matninsspänning på 5.0 V. Eftersom batterispänningen ofta ligger däröver används en spänningsregulator för att reglera ner spänningen till processorn samt in/ut-enhter. Motorerna körs däremot på batterispänningen för maximal effekt. Kopplingsschemat för hela konstruktionen visas nedan.

4 Resultat och slutsats

Vi prövade många olika algoritmer för objektundvikning innan vi bestämde oss för att använda en bestående av ett flertal if-satser. Innan dess prövade vi bl.a. att interpolera fram pulsningar till motorerna, vilket visade sig onödigt samt för krävande för att producera ett förutsägbart programflöde (pga avbrott). Roboten undviker de flesta hinder så snabbt den upptäcker dem, men om den klantar sig och kommer för nära ett hinder så vänder den på stället. Genom denna nödvändning kör den sällan in i hinder. Det finns tillfällen då roboten kör in i hinder men de beror till största delen på bristningar i informationsinsamling:

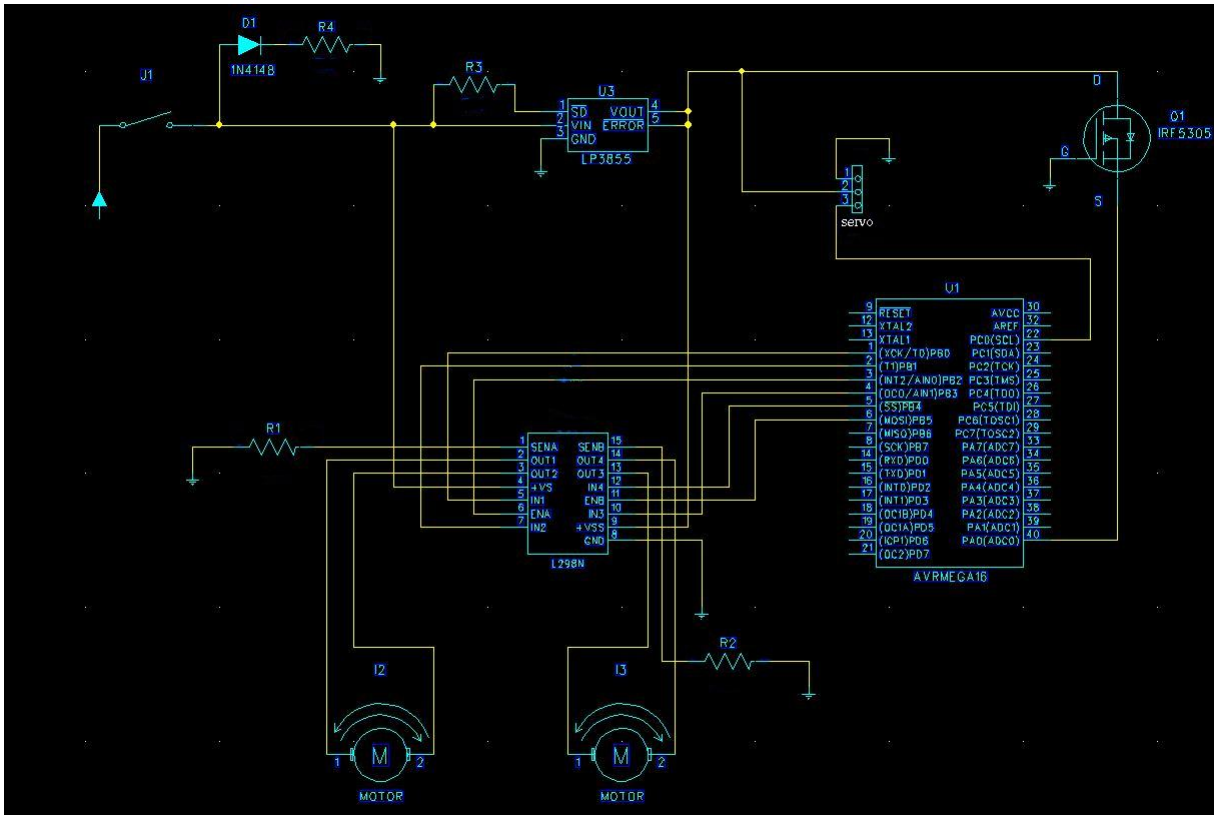

Figure 1: Kretsschema över roboten

Om roboten snabbt vänder kan den missa att läsa in föremål eftersom servons hastighet är begränsad. Den kan även slå i objekt med bakpartiet vid vändningar eftersom den inte har någon sensor där. Tunna föremål missar roboten ofta för att den har begränsat med riktningar som avstånds-kontrolleras vid varje svepning.

Som sagt finns en del brister trots en ganska robust algoritm för att undvika föremål. Problemen ligger snarare i hårdvaran och sammanfattas här:

Hastigheten på servot begränsar hur snabbt evalueringar, av vad roboten ser, kan göras. En svepning tar sådan tid att bilden av robotens omgivning blir felaktig, för att den hinner åka en bit mellan två succesiva evalueringar. Fler avståndsmätare hade delvis kunnat lösa detta problem. Information om hur mycket roboten verkligen svängde och förflyttades hade varit värdefullt för en noggrannare analys.

Arbetet har fortskridit relativt smärtfritt. Det största problemet har nog varit att fästa sladdarna till avståndsmätaren utan att de gick av som resultat av servons vridningar.

Vi tänkte även implementera styrning av roboten via en vanlig TV-fjärrkontroll, men då vi fick stora problem med att fastställa knapparnas kodning hann vi inte göra det. Fjärrkontrollen var av okänt märke så vi kunde inte samla information om pulslängder m.m. vilket var nödvändigt för att bestämma kodningen.

Sammanfattningsvis är vi nöjda med vad vi åstadkommit och vi anser att undvikningsalgoritmen är bra nog med tanke på den hårdvara roboten använder. Algoritmen kan finslipas efter mer testning, men den fungerar tillfredsställande.

5 Källkod