REU 2002
Machine Intelligence Laboratory
University of Florida, Gainesville, FL

Final Report

SARbot

Vincent Au-Yeung
vauyeung@sfu.ca

Table of Contents

1 Abstract

SARbot is an autonomous robot capable of navigating a variety of dry, outdoor environments. It uses sonar to perform obstacle avoidance and GPS to navigate to programmed coordinates. It’s purpose is to serve in a land based Search and Rescue mission when inhospitable conditions make sending live personnel into the field dangerous. Inherently, this platform has military applications but may also be used for civilian operations such as searching for lost hikers.

2 Executive Summary

SARbot was initially conceived as an autonomous robotic vehicle designed to perform search and rescue functions in situations that would otherwise be deemed too risky for human undertaking or in cases where personnel availability is lacking.

SARbot is constructed in such a way as to give it the ability to traverse various terrain (gravel, paved road, grass, mud, etc) outdoors while also having enough power to propel it up steep gradients and over obstacles. The tank tread drive system accomplishes this and consists of 3 pulley wheels to support each tread. Both treads consist of an inverted urethane timing belt that is mounted under tension and driven by friction between the belt and the driving pulley wheel. The front pulley wheel increases the contact area with which SARbot can use to grip and climb over obstacles.

Obstacle avoidance is accomplished by 2 sonar modules that each continuously sweep 90 degrees for a 180 degree coverage of approaching obstacles. The sonar allows for more accurate distance measurements to be made than with IR sensors and is immune to colors and heat, thus making it suitable for outdoor use under the sun. Navigation is accomplished using an OEM GPS module that outputs positional information at a rate of 1Hz, and using this together with some computation enables SARbot to find its heading and thus to navigate to a destination GPS coordinate without the use of a compass. SARbot is programmed to navigate to a pre-programmed GPS coordinate, then will return to its start position upon reaching its destination.

SARbot is controlled by the Atmel Atmega163 microcontroller resting on an STK500 development board. The micro is ISP programmed and the software is written in C.

3 Introduction

There are two parts to carrying out a search and rescue mission: the first is to search out a target and the second is to bring that target back to a safe environment. SARbot was originally intended to fulfill these requirements, but the numerous challenges faced with the construction of SARbot ate away at time, and my focus was shifted away from giving all of these features to SARbot and risking a non-working robot to focussing on the core functions of SARbot.

This led to the idea of employing a more localized search mechanism upon arrival at a GPS location to be scrapped as well as the idea of having some kind of simulated rescue action performed to also be discarded. The first of these non-implemented features was conceived to be accomplished using an audible sound emitting beacon to signify the target to be searched for and a microphone designed to sweep 360 degrees listening for this beacon in order to guide SARbot towards it. The latter feature was originally planned to be implemented as some kind of robot arm with a magnet on its end which would pick up the target (which also supposedly had an attached magnet) upon finding it via the audio sweep. It would then bring the rescued object back to the mission's start location, thereby completing the rescue operation.

However, the final version of SARbot navigates to a pre-programmed GPS coordinate and returns to its start location upon arrival at its destination, with the idea that the stranded target would be able to do some searching on its own (for SARbot) and SARbot would carry a payload of survival and communications equipment on its mission to be handed over to and used by the target.

This paper will detail the design and construction of SARbot and present the various challenges and difficulties encountered. A complete set of captioned pictures of SARbot can be found by visiting http://www.sfu.ca/~vauyeung/REU02/all_pic/indexall_pic.html, which complements this report by allowing the reader to visualize my descriptions, for merely reading this report may likely leave the reader a little confused.

4 Integrated System

SARbot can be divided into 5 subsystems: the frame and body, the drive system, the obstacle avoidance system, the navigation system and the microcontroller code. Each of these subsystems was independently worked on in the order listed and likewise decreased in difficulty.

The frame and body is the physical platform that supports everything else and had to be sturdy enough to support 20lbs of load, which is why I decided on making the frame of aluminum and to fill in the gaps with balsa wood.

The drive system was just as challenging because I had to devise a way to mount the motors to the frame and to also create a housing that would allow me to couple the motor shaft to the pulley wheel's drive shaft. This was done using aluminum as well and the housing was given volume by using aluminum standoffs, thereby making the drive system open and easily accessible while also reducing some of the robot's weight. The inverted timing belts were mounted over the 3 pulleys and the tension was made adjustable by the turn of 2 nuts on the front arm of SARbot supporting the upper front pulleys.

The mechanism for enabling the two sonar modules to perform their controlled sweep was devised using a single servomotor and a Lego based mechanism for translating linear movement into rotational motion. The sonar modules themselves were directly controlled by the mega163 and independently powered by their own voltage regulator.

The navigation system consisted of an OEM GPS module with an attached active antenna in such a small package it was tough for me to find a way to connect wires to it because the only connection interface provided for it was a ribbon cable so small that only a single strand of wire of the standard stranded ribbon cable wire could be soldered onto it. The GPS system had to also be powered with its own 3.3V regulator and because the mega163 runs at 5V, a level shifter was also used on the GPS serial output line. An LCD display was used to help in the debugging of the GPS module because the only UART interface on the microcontroller was already taken up by the GPS module.

The microcontroller was programmed in C in such a way as to get the individual components to work independently before combining all their operations and making them work with each other for the final control code of SARbot.

5 Mobile platform

This was my first design task of the project, and proved to be the most challenging since I set out to build quite an unusual mechanical platform utilizing material I had never before worked with. Nevertheless, even though my background is in computer engineering, I had a lot of fun and learned the most from working on this part of the project. I tremendously appreciated the hands-on approach to mechanical design and the tangible, satisfying results of spending your entire day in the machine shop.

First of all, I needed a sturdy enough design to support the components required to power my 448oz-in motors that run at 24V and draw a combined 2A. My batteries were sealed lead acid weighing 6 lbs and I had to build a motor housing to couple the motor shaft to the drive shaft. This necessitated the construction of a platform that could not be entirely composed of balsa wood (the standard material used in all the smaller robotics projects) because that would yield too flimsy a body. Instead, I used scrap aluminum I found lying around the lab to make SARbot's frame and filled in the gaps with balsa wood. The frame was put together using nuts and bolts (with locking washers), and I'm sure these added a good pound or two to SARbot because I used so many of them. It would have been more weight friendly to rivet the frame together or even weld it, but I did not have access to those options.

Each of the two front arms (supporting the horizontal bar that holds the 2 front pulleys) is made of a threaded rod covered with an aluminum tube which was held in place by drilling through the tube/rod assembly and hammering in a locking pin. The bottom end of the threaded rod screws into another much larger diameter aluminum rod which in turn directly attaches onto SARbot's frame with the use of machine screws. This larger diameter rod (painted green) that attached to SARbot's frame was chosen because I could easily drill screw holes into it at different angles due to the surface being uniformly round and thus making it simple to setup the drill bit perpendicular to the surface tangent. Had I used a rectangular rod I would have had to find out how to start the drill hole at an angle without breaking the bit. The pictures on my website (http://www.sfu.ca/~vauyeung/REU02/all_pic/indexall_pic.html)show this entire assembly in better detail than I can ever put into words.

The front arm pulleys are held in place with shaft collars whereas each of the middle pulleys mount onto a 3/8" shaft held in place using two shaft collars, a flanged sleeve bearing, and a regular sleeve bearing. The drive train assembly was a little more complex because I had to create a motor housing that would allow me to couple the motor shaft to the drive shaft as well as fix the free-spinning pulley wheel to the shaft so that its rotation could be controlled by the motor. All this can be seen on my website.

Finally, an inverted urethane timing belt was mounted on the three pulley wheels on each side of SARbot and the tension was adjusted by turning the two nuts located on the two front arms. This completed the toughest of the mechanical work I had to do for SARbot.

The real lesson I learned from building the mobile platform was that mechanical work is not as forgiving as poorly written software; one can always rewrite and re-compile software whereas a single drilling mistake you make will either ruin your part or will make it end up looking pretty ugly if you find a way to work around the mistake (I had a few of the latter happen. Can you spot it in the pictures?).

6 Actuation

I used two motors (one for the right tread and one for the left) to drive the rear pulley wheels which ran the timing belts like tank tracks. The motors were fairly powerful, rated at 448oz-in with a maximum speed of 47rpm, and each draw a maximum continuous current of 1.08A at 24V. I used two 12V SLA batteries in series to power the motors and a National Semiconductor LMD18200T PWM H-bridge motor driver to control each motor. The steering was accomplished using zero radius turning whereby SARbot would pivot around its center point by having the treads turn in opposite directions. This method of steering allowed SARbot to turn on all the surfaces I had intended it to, from coarse grass to gravel to pavement. The timing belt increased the surface contact area with the ground, which gave SARbot more traction for climbing a slope, descending a slope, and climbing over obstacles like exposed tree roots and curbsides.

The National motor drivers I used were rated at being able to supply a continuous current of 3A and peak at 6A, which was plenty for my motors. Somebody had previously printed a circuit board for this chip and had some left over so I used two of them for SARbot. These motor drivers used the standard BJT H-bridge design for switching the direction of current flow to the motors. The digital logic interface allowed me to directly control the motor driver from the micro's output pins. The speed of the DC gearmotor was controllable by sending a PWM into the 'enable' pin of the driver, which, based on the PWM's duty cycle, controlled the amount of current sourcing the motor and hence affected the motor's speed. The direction control was accomplished via another pin which accepted either a logic high or logic low. The motor driver was also heat-sinked to prevent overheating although this wasn't really necessary for my motors. I should also note that when the motors are running under typical conditions and occasionally switch directions, this affects all the other circuits drawing power from the same batteries in an unpredictable way. The best way to ensure proper operation of these other circuits would have been to provide a completely separate, isolated, power supply to them. However, I found difficulties in implementing that because the motor driver circuit required a ground reference for the logic input and this had to be common with the micro's ground reference, otherwise the motor driver would have had problems decoding the input logic levels. For SARbot, all I did was to use voltage regulators for every component that required power, and had a single ground common to every electronic component used.

The other type of motor I used was a standard servomotor, and I designed a mechanism that would only require one of these motors to control the rotation of both sonar modules (you can see this best in the pictures on my website, so I will not attempt to explain the operation in this paper). The idea behind using a single motor was that I could make both sonar modules rotate in a symmetrical fashion and to also reduce the power consumption. The servomotor operates on the basis of controlling the motor shaft's position by controlling the duty cycle of the input PWM. This allowed me to accurately determine where a potential obstacle in front of SARbot was in order to properly steer away from it (since I would know the angle the sonar would be pointed in when I sent a ping to make a distance measurement).

7 Sensors

There were two types of sensors used in SARbot: sonar and GPS. The sonar modules allowed SARbot to avoid obstacles whereas the GPS module allowed it to navigate outdoors based on information fed to it from the GPS module. Both these sensors worked independent of each other although the sonar modules were accurately positioned by the servomotor.

Both sonar modules were Devantech SRF04 ultrasonic rangers specified to be capable of measuring obstacles from 3cm to 3m away. The advantage of using sonar over IR rests in its capability to operate outdoors without adverse effects from the surrounding environment since it generates a 40KHz audio ping way beyond human hearing. Unless an object is capable of absorbing or greatly distorting this signal, the only errors that may present themselves would be in the form of electrical delays and whatever effect a cheap transducer may have on the quality of the ping transmitted and received. Apart from the robustness inherent in an ultrasonic ranging device, the other greatest advantage lies in the long range it can measure. Of course, a nice upgrade to using sonar for obstacle avoidance would lie either in the incorporation of a vision system or some kind of laser ranging device, both of which would have caused me to blow my budget by a fair deal. Nevertheless, operating the sonar was pretty straightforward: raise the sonar input line high for 20us, bring it down (the ping is sent on the falling edge of the input line), start a counter and listen for the output echo line to go high, which triggers an external interrupt on the rising edge. When the micro is interrupted, read the counter and use the speed of sound to calculate a distance. This was the basic operational flow of both sonar modules.

The OEM GPS module I used was the RGM2101 by Royaltek and includes an active antenna built on top of the GPS engine board. The entire module is sized at just under 2" wide and long, and under 1" high. This made the placement easy but connecting headers to it difficult because it was supplied with an extremely small cable meant for machine soldering (that's what you get when you buy OEM). The GPS module, by default, sends out NMEA 0183 standard messages serially at 4800bps, updating at a rate of 1Hz, so I didn't have to write code to transmit data to setup the GPS module. All I did was to setup the UART on the micro to receive these messages and I'd analyze the messages I wanted (by filtering the headers), then used this data to navigate SARbot. One intermediary I did have to implement was a level shifter because the GPS module runs at 3.3V whereas the micro operates at 5V.

I had also wanted to use a digital compass to help SARbot navigate but that didn't play out because my budget didn't leave room for one nor did my pre-occupations with other problems help. I did find out when it finally came time for me to write SARbot's navigation code that having a digital compass would have helped tremendously by reducing the amount of computation I had to put the micro through, not to mention the improvement in accuracy it would also have provided. SARbot navigated based on three pieces of information: the current heading provided by the GPS module, the current GPS coordinate, and the destination coordinate. If a change in heading was required to steer SARbot in the right direction towards the destination GPS coordinate, I would calculate the required adjustment to the current heading using the three pieces of information and some trigonometry. I did run into some problems with my program code being too large and not fitting into the flash on the mega163, but through some optimizations and tweaking I was finally able to just barely squeeze the code in.

8 Behaviors

SARbot's behavior is very straightforward: collision avoidance by sonar has highest execution priority when an obstacle is detected within a buffer zone in front of SARbot, otherwise if there is a valid GPS reading SARbot will attempt to adjust its heading (if necessary) to steer it towards its destination. If there is no valid GPS reading, SARbot will just do collision avoidance, and if there are no obstacles to avoid, it will just go straight.

One important note to make about SARbot is that when going "straight", SARbot will inevitably shift about 1 ft to the left for every 10 ft traveled. This is a result of many factors including non-identical motors (and motor driver circuits), and non-identical drive trains for the left and right side. I did not attempt to compensate for this because it is time-variant and also depends on a whole slew of other factors like temperature, humidity, etc. and is also the subject of a Ph.D thesis, which, needless to say is well beyond the scope of my implementation.

SARbot has two simultaneously running sonar modules; one scans the left side and the other the right. Due to the length of time it takes to move a sonar into position (using the servomotor) and the time it takes to send the ping and wait for the echo, I was only able to get each sonar to "ping" two different positions every second (or a bit under a second). One of the positions was directly in front of SARbot and the other at 45 degrees (the diagonal corner). Fortunately, the sonar ping spreads out in a conical fashion with a cone angle of about 30 degrees, which provides me with sufficient coverage to detect any potential obstacle when pinging at only 2 different positions with each sonar.

If at any moment SARbot detects an obstacle within a foot of it (such as somebody suddenly walking in front of it), SARbot will react by either immediately turning away from it or it will move backwards about 3 ft and look to the left and right for somewhere to turn. The decision of what to do depends on where it detects the obstacle(s) and how close each obstacle is. In the "normal" collision avoidance mode (when an obstacle is detected by the sonar within 6 ft in front of SARbot, but no obstacle is within 1 ft), SARbot will use the readings from the 4 sonar positions (diagonal left, front left, front right, diagonal right) to determine the best direction and angle to turn. If this algorithm does not yield a result indicating the best path to take (maybe because all four sonar positions indicate the presence of an equally close obstacle), then the servo will rotate both sonar modules 90 degrees from the front so that they are positioned to view directly left and right of SARbot, looking for more room to maneuver. This algorithm successfully prevents SARbot from ever getting stuck in a corner or other "trap" because SARbot will always react based on the best obstacle information it obtains from a 180 degree field of view in front of it. There is only randomness in the decision making process when it is in a situation where it detects a symmetrical obstacle pattern on its right and left sides.

When not running the higher priority collision avoidance routine, and the GPS module is outputting valid data, SARbot will navigate by adjusting its heading as described in Section 7. The algorithm used to find the angle to turn in order to adjust SARbot's heading makes use of some simple trigonometry that I will not attempt to describe here. However, I must re-iterate how beneficial a digital compass would have been. This is because SARbot moves at a rather slow pace, and with the error inherent in the GPS data (of typically a few meters), the heading information provided by the GPS module ends up being somewhat inaccurate, especially since SARbot's forward travel is not quite "straight". Nevertheless, if a large margin of error was accounted for in determining when SARbot had reached its destination, SARbot could successfully navigate to its destination, and upon reaching it re-program its destination so that it would now be headed back to its start position.

9 Conclusion

Although SARbot was initially a little too ambitious, once I started getting things rolling along by ordering parts and working in the machine shop, I realized that I had to set a realistic, preliminary core goal, and if time permitted I would seek an extended goal. I accomplished this preliminary goal with SARbot and as most projects go, the extended, more ambitious goal was not even sought. I managed to get SARbot to navigate to a GPS coordinate and return back to its start coordinate (given I accounted for a large margin of error in the GPS coordinates). Collision avoidance also worked fine although I did have to tilt the sonar modules upwards by about 20 degrees, otherwise running SARbot on outdoor, hilly terrain would not have worked properly (it would have erroneously detected obstacles).

I am particularly proud of my mechanical work and the fact that SARbot drew a lot of praise for being a good looking robot. I learned that making a robot good looking is about as important as making it functional, since an ugly piece of junk that does miracles will still not draw as much attention as a stellar looking robot that just sits there and does nothing :)

As I mentioned several times already, a digital compass would have been the best possible upgrade for SARbot. Given a bigger budget, I would have also liked to use a real tank tread drive system as found on actual tanks instead of inverting timing belts and running them under tension. In fact, you may even classify SARbot as a lawn mower because it does a heck of a good job tearing up grass as it runs over it. It might even be good competition for the pizza mower project (inside joke).