

Date: April 21, 2008

Written report

Autonomous Navigation and Obstacle Avoidance Vehicle

EEL 5666: Intelligent Machines Design Laboratory
Spring 2008 1st Written Report

Student Name: Kyuhyong You

Instructor: Dr. Eric M. Schwartz
Dr. A. Antonio Arroyo

TA : Adam, Mike, Sara

University of Florida

Table of Contents

1. Abstract.....	3
2. Executive Summary.....	3
3. Introduction.....	3
4. Integrated System	4
5. Mobile Platform.....	5
6. Actuation.....	5
7. Sensors	6
8. NavigationAlgorithm.....	10
9. Behaviors	11
10. Experimental Layout and Results	12
11. Conclusion	13
12. Documentation	13
13. Appendices.....	14
14. Reference	15

1. Abstract

My robot is a mobile platform of any kind of exploration robot so that it can navigate through designated waypoints while trying to avoid obstacles. And finally comes back to the starting point or another place. To minimize the time and money for building a new mobile platform, it is based on commonly used scale RC model car with some modification.

2. Executive Summary

For many years, navigation of the robot has been the most basic and yet most challenging stuff in developing a mobile robot. It is because moving the robot to the required place is getting more important as they become mobile. For example, if hundreds of researchers developed a bomb deactivation robot with great sensors and arms but it cannot go to the target by it, how can we evaluate the robot as useful? This shows how navigation is important in developing a mobile robot. Especially for outdoor application, utilizing the GPS receiver became a significant breakthrough in developing mobile robot. With an affordable GPS receiver and better accuracy, now everyone can develop mobile robot much easier than before.

However, since the GPS is nothing more than just providing global coordinates of the current location and heading information, it is required to implement a hardware and software system that can navigate the robot. So this project will cover comprehensively from building a hardware system of mobile robot and complete navigation system with obstacle avoidance.

This article will discuss issues in developing autonomous GPS navigation based on RC car. First, I will introduce hardware components including the car, motor driver, GPS system and proximity sensors, and show how they are organized with micro controller. Then, bring some idea about the navigation and problem solving techniques. And finally analyze the result of testing and discussions.

3. Introduction

The concept of my robot starts from very basic requirement about the robot itself. How to make the robot can have mobility? Extensive development in electronics and software enabled robots in nowadays can do almost anything we can imagine and its boundary will be expanded. However most of robots don't have enough mobility thus far and even some of mobile robots have to depend on inefficient mobile platforms. First of all, they are too slow. It is obvious that 4 legs or biped cannot be developed to be more efficient and fast enough than wheeled vehicle. If we can have fully autonomous and independent vehicle then it will change our life dramatically.

Primary objective on this project is designing a car based robot that can navigate through GPS coordinates automatically. The robot is based on regular RC car and entire components including microprocessor and sensors are on the same base. More details about the platform and system will be discussed first then go through each components. And will talk about behavior of the robot and experimental result followed by conclusions.

4. Integrated System

As we can imagine a driver in a car use his or her eyes to sense the road ahead and makes decision with the brain then give a command to the hand and legs to control the car, this robot has almost the same components. The robot is composed of four components as microcontroller, sensors, servos and motor.

Fallowing diagram shows how things work with the robot.

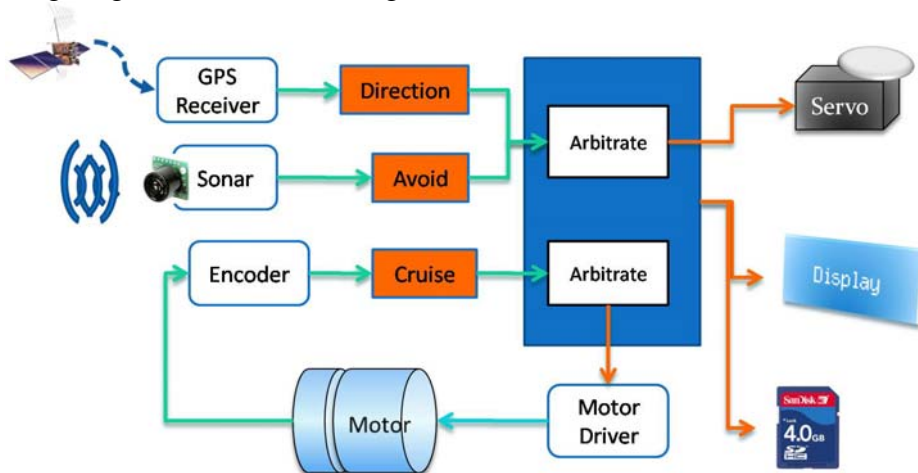


Fig 1 Behavior Control Diagram

Microcontroller, which is almost equivalent to brain of a driver, will process data, make decision and give signal to its actuators. For this project, HC12 16-bit microprocessor is selected.

For the sensor part, ultrasonic sensors will be attached as two in front and one in rear to detect obstacles and measure the distance from them. To make the sensors more functional and flexible, each sensor will be attached on a small servo. And GPS sensor will be placed on the top of the platform in order to prevent interference with other objects.

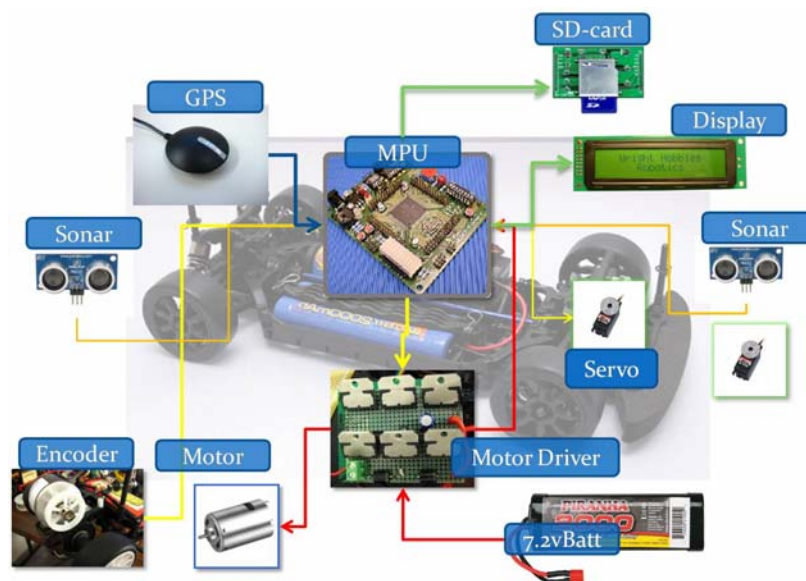


Fig 2 Overall Hardware Integration

5. Mobile Platform

Since this robot is based on a regular RC model car, physical dimension and mechanical specifications are limited to a given RC car.



Specifications	
Length	378.6mm
Width	188mm
Wheel base	257mm
Motor	RS-540SH
Drive	Rear 2WD

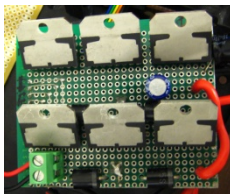
6. Actuation

1. Motor



RS-540SH, generic brushed DC motor is attached at the rear section of the platform and produce main power to the car. DC motor especially for this kind of cheap motors are known as very difficult to control because it draws so much electricity from the circuit board that the microcontroller become unstable. To prevent such kind of harmful effect from running motor, it is highly recommended to separate the source of electric power.

Motor Driver



An array of six L298N Dual-full-bridge motor drivers is used for the motor driver. According to the datasheet of L298N, each driver can draw 4A, hence by combining them into 6, it can produce up to 24A theoretically.

PID control

To control the DC motor more precisely and reliably, PID control is required.

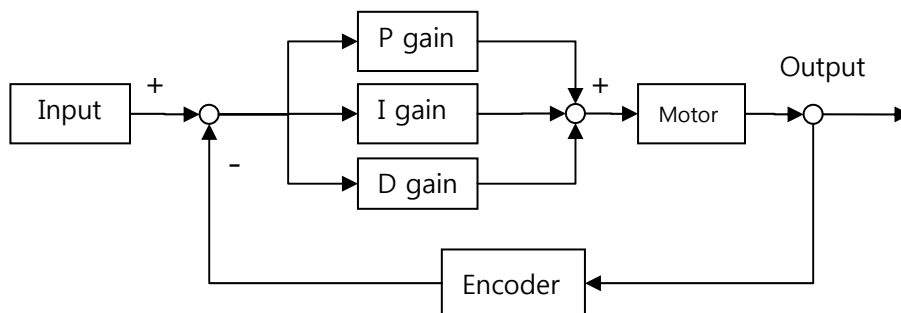


Fig 3 A block diagram of a PID controller

An incremental encoder with resolution of 1440 ppr(Pulse Per Revolution) is connected to the main spur gear which drive main shaft. This encoder will

measure the rotational speed of the shaft by counting the pulse at a specific timing. By comparing the error between the input and output, the controller can determine if it is going to increase or decrease the power of the motor.

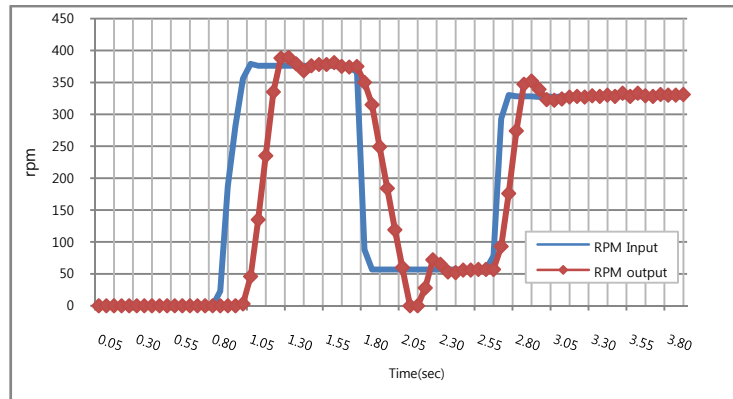


Fig 4 PID control result

2. Servo

High speed HS-5925MG digital servo is mounted in front section of the car to change the direction.

7. Sensors

Some proximity sensors and encoder are used for this project. Proximity sensors are essential for obstacle avoidance and encoder for motor control. In order to minimize cost and time, each sensor is evaluated in terms of suitability, affordability and feasibility. The point of choosing sensor is about doing more with less. I tried to minimize requirement of sensors.

1. Obstacle Avoidance

Obstacle avoidance requires a robot can detect or measure the distance from the object in front or in the course. A robot or micro controller can perform maneuvers to avoid obstacles according to the measured signal from those sensors. Therefore it is easy to say that better sensor gives better outputs.

This table shows some common sensors that can be used for the robot.

Type	Distance	Accuracy	Signal	Price(EA)	Out door
IR sensor	0.1-0.3m		Analogue	\$10~15	No
SONAR	0-6.4m	±10~	Analogue RS-232 PWM Timing	\$25-40	Yes
Bump switch	1.0-5.0cm		On/Off	\$0.50-1.00	Yes
Vision Sensor	0-10m+		RS-232 Digital	\$120-?	Yes
Laser	0-1,000m	±1.5mm	Digital	\$500-?	Yes

Some sensors like IR sensors are very common in obstacle avoidance and/or proximity measurements; however it works poorly in outdoor application. So I excluded this type of sensor.

	IR	Sonar	Bump	Vision	Laser
Suitability	1	5	2	5	4
Affordability	5	4	5	3	1
Feasibility	5	5	5	2	2
Total	11	14	12	10	7

1:Very poor, 2:Poor, 3:Middle, 4:Good 5:Very good

Table 1 My sensor evaluation


i. Ultrasonic sensor (SONAR)

SONAR or Ultrasonic sensor is very common type of sensor detecting object and measuring distance.

Two ultrasonic sensors are mounted in front of the car and one in rear. Front sensors are mainly used for detecting obstacles and measuring the distance between the car and the objects. To help microcontroller finding the way more easily, these sensors can be tilted around the front area by small servos.

Maxbotix LV-EZ1

I choose EZ1 from Maxbotix for the project because this is the cheapest of all kind and it also provides various types of output signal including PWM, RS-232 and Analogue voltage. One of the good things about using analogue output is that the output is proportional to every inch. So the user does not have to worry about handling timer interrupt and calculating distance with the flight time of ping.

	Specification	
	Vcc	5v
	Range	0-6.45m
	Output	RS232 Serial Analogue – 10mV/in PWM – 147uS/in
	Price	\$25.00 x 2

2. Encoder



Stegmann LD-20 Incremental Encoder	
Resolution	1440 ppr
Vo	5v
Type	5p Optical

CONN. PIN	WIRE COLOR	FUNCTION
1	CLEAR	DRAIN
2	RED/BLK	M OUTPUT
3	WHT	A OUTPUT
4	DRN	B OUTPUT
5	BLK	GND
6	RED	+5V

The encoder has an output of 3 channels as A, B, and M. Channel A and B detects different phase of pulse so that by looking at the phase of each channel, the controller can determine whether the motor turns forward or backwards.

Fig 5 Pin out of the encoder

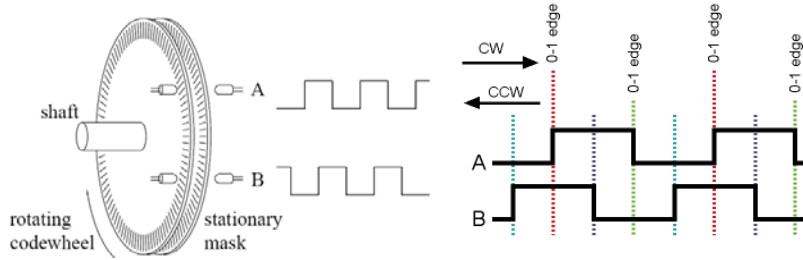


Fig 6 Encoder Signal Diagram

In order to count the number of pulses comes from the encoder, I plugged the channel A into pulse accumulator pin in 68HC12 board. Since the HC12 controller has only one pulse accumulator channel, I have to connect channel B to one of the input capture timer channels and increase the number of count on every capture interrupts.


This can be done simply by enabling PACTL and initializing PACN32 which is 32-bit counter, automatically increase number for every pulse.

PACTL_PAEN = 1;	// Enable Pulse Accumulator
PACTL_PEDGE = 1;	// Rising edge
PACN32 = 0;	// Pulse Accumulator Counter 2,3 initialize

3. GPS Navigation

Currently there are hundreds of different kinds of GPS receivers in the market.

i. GPS Receiver

	GlobalSat BU-353 USB Mouse GPS (From BuyGPS.com)
Chipset	SiRF Star III (firmware version 3.1.1 with WAAS Support turned on by default)
Accuracy Position	10 meters, 2D RMS 5 meters, 2D RMS, WAAS enabled
Reacquisition	0.1 sec., average
Power consumption	80mA
Baud rate	4,800 bps
Main power input	4.5V ~ 6.5V DC input
Output message	NMEA 0183 GGA, GSA, GSV, RMC, VTG, GLL
Physical Characteristics	Dimension: 53mm diameter, 19.2mm height

GPS data analysis
PARSE GPRMC DATA

```
$GPRMC,194205.000,A,2938.5724,N,08220.8584,W,0.37,78.34,010308.,*21
```

- (1) (2) (3) (4) (5) (6) (7) (8) (9)
- (1) Current Time: hhmmss.ss
 - (2) Validity: A- AOK,
 - (3) Latitude: DDMM.SS.ss
 - (4) North South
 - (5) Longitude: DDMM.SS.ss
 - (6) East West
 - (7) Speed : Knots
 - (8) Heading: 0, 360: north
 - (9) Date today

Reading and Parsing the Data

Processing the GPS data is composed of two parts as receiver and parser. Receiver is simply an interrupt service routine that stores data into certain variable. Since the interrupt is set very quickly, the code should be as small as possible to avoid delayed interrupt problem.

Every time the controller receives a character through a SCI channel, an interrupt is set and store it to a global variable. If it finds a character '\$' which means the starting of GPS code, then the controller compare next 5 characters with 'GPRMC' which I'm looking for. If the data contains GPRMC code, a flag is set and the main program calls the Parsing function.

```
interrupt void SCI0_ISR(void) { /* SCI interrupt service routine */
    unsigned char comp[5];
    temp = SCI0SR1; //dummy read(clear flag)
    temp = SCI0DRL; //read data
    if(gpsData.GPS_Received == 1) return;
    switch(gpsData.state){
        case 0:
            if(temp == 0x24){ //wait for receiving '$'
                PORTB_BIT6 = PORTB_BIT6^1;
                gpsData.ptr = 0;
                gpsData.state = 1;
            }
            break;
        case 1:
            gpsData.Temp[gpsData.ptr++]=temp;
            if(temp == 0x2a){ // IF '*' is found
                if(strncmp(gpsData.Temp,"GPRMC",5)==0){
                    // Compare if "GPGGA" received
                    gpsData.GPS_Received =1;
                } else gpsData.GPS_Received = 0;
                gpsData.state = 0;
            }
            break;
    }
}
```

If the gps receive flag is set, the parser routine is called and start analyzing the code. Since the data is separated by commas, strtok function defined in <stdio.h> is conveniently used to parse the data out of one string.

```

void Parse_GPS(void){
    char *strHead_RMC,*strTime_RMC,*strAOK_RMC,*strLat1_RMC,      strLat2_RMC,
        *strNS_RMC,      *strLon1_RMC,*strLon2_RMC,      *strEW_RMC,      *strSpd_RMC,
        *strHDG_RMC;
    strHead_RMC = strtok(gpsData.Temp, ",");
    strTime_RMC = strtok(0, ",");
    strAOK_RMC = strtok(0, ",");
    strLat1_RMC = strtok(0, ".");
    strLat2_RMC = strtok(0, ",");
    strNS_RMC = strtok(0, ",");
    strLon1_RMC = strtok(0, ".");
    strLon2_RMC = strtok(0, ",");
    strEW_RMC = strtok(0, ",");
    strSpd_RMC = strtok(0, ",");
    strHDG_RMC = strtok(0, ",");
}

```

8. NavigationAlgorithm

1. Navigation

- i. How the navigation system works?

The idea of finding designated waypoint and following the heading was based on very simple geometry. If I know where I am and where my heading is then I can determine whether I should turn to right or left. In this project, the robot can figure out its location in the global coordinate system from GPS signal. However the GPS does not provide anything about the directional information.

This kind of navigation system is similar to how a pilot maneuvers the airplane towards the waypoint.

Variable Name	Description
cHDG	Current Vehicle Heading
gHDG	Goal Heading (atan2(dLat, dLong))
dHDG	Heading Differential
LOC	Current Location
gPoint	Goal Point

Goal Heading represents the angle between current location and next way point measured from the north pole. This can be obtained from arctangent of

difference of latitude and longitude between the two.

$$gHDG = \tan^{-1} \left(\frac{dLat}{dLon} \right)$$

$$dAngle = gHDG - cHDG$$

The dAngle which is directly related to control the servo to align the robot can be calculated simply by subtracting current heading from goal heading.

At first, this simple algorithm worked fine with some waypoints, however it turns out that only this process could be confusing when the target is in the north and the robot is heading toward the north. This is due to the specialty of the North pole. Simply because that the north pole can be represent either way as 0 degree or 360 degree and this makes the robot confused. These problematic cases are discussed below.

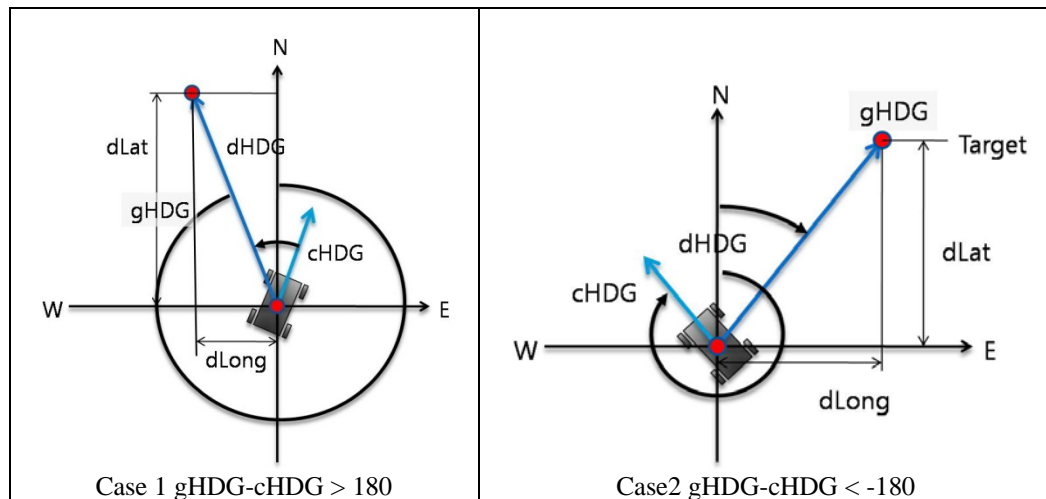


Fig 7 Exceptional Cases in GPS navigation

The first case shows when the robot is almost aligned to the target but the current angle is way smaller than the goal heading. At this moment, the robot will turn opposite direction and keep confused and never get to the target.

The second case is an opposite case of the first one.

In either case, the robot becomes disoriented and would never get to the target in the experiment. I solved this problem by adding some code here.

$$\text{If case 1: } dHDG = gHDG - 360 - cHDG$$

$$\text{If case 2: } dHDG = 360 - cHDG + gHDG$$

This works great with navigation and the robot wouldn't be confused again.

9. Behaviors

1. Navigation Mode

i. Initialization

Boot up the GPS module and find the current coordinate.

ii. Set the course

Load designated waypoints into the list. Select the first waypoint and turn to the direction.

- iii. Go to the waypoint
Start traveling to the next waypoint. Once it gets to the waypoint, set the waypoint and the next waypoint as its new starting point and end point respectively.
Repeat until it reaches the final destination.
- 2. Obstacle Avoidance
 - i. Obstacle detection
 - ii. Decide whether the object is in left or right.
 - iii. Correct actuator signal.

10. Experimental Layout and Results

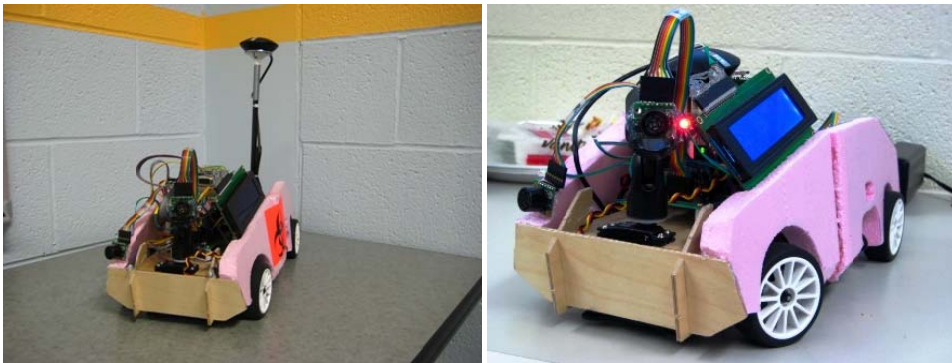


Fig 8 Overall Layout

GPS Navigation Field test

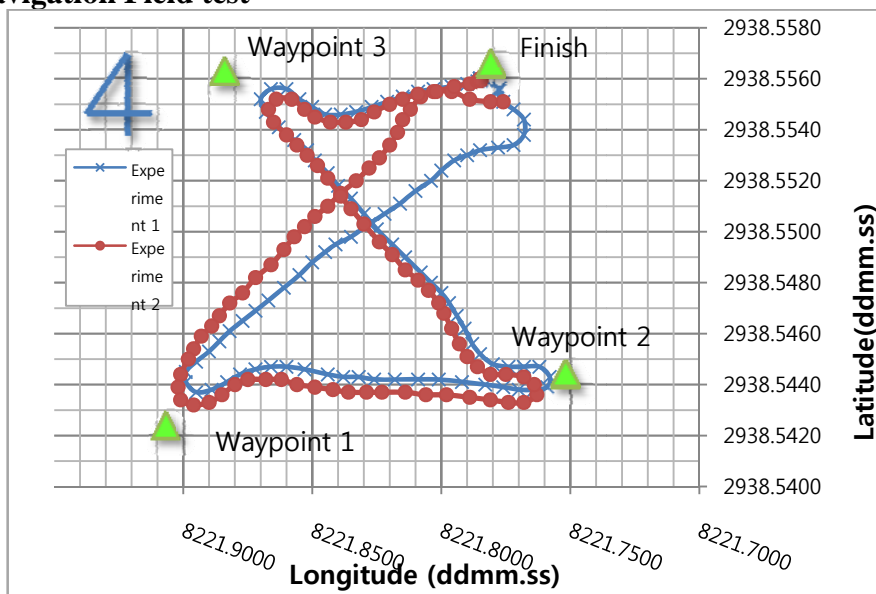


Fig 9 Experimental Data of GPS Navigation

Actual GPS navigation test conducted in one of the empty parking lot in campus. I conducted two experiments with same starting point and without any obstacles in the field. Each point is recorded on the embedded SD-card module whenever a GPS signal received by MCU. Data is saved as a text file and processed using spreadsheet software. No filtering and smoothing has been made during data processing. Notice that a distance from the starting point to the first waypoint is approximately 20 meters.

This experiment shows how the robot accurately can go through the waypoints. Initial instability of tracking can be observed and this is caused by disorientation at the beginning part of navigation. This problem happens when the robot is stopped for a while and the GPS could not calculate orientation of the robot. I expect attaching a compass module to the robot should help a lot to determine orientation of the robot.

11. Conclusion

Before choosing this project, I have researched what students who took this class had previously created in the internet and it inspired me to go with GPS navigation. This decision was made simply because that nobody actually made successful robot with this theme. Another reason is that I was interested in autonomous GPS navigation. So, receiving the data would be the highest goal in my project

Processing the GPS signal and controlling the robot is much difficult than I initially anticipated. It took me about one full month just to get the reliable GPS data from the receiver. I started writing a data processing code right after receiving the GPS receiver and I thought to be almost successful but it gets the data on and off. I was almost desperate when time passed more than two weeks and almost giving up. One significant breakthrough was made when I got some suggestions and source code from a researcher in Korea who I added as a reference. Thanks to his work, I could change my crappy GPS receiving source code and finally can grab the data I needed.

But this was just a beginning of the whole project. Implementing the navigation system was a big challenge and demanding work since it requires lots of actual tests. Moving the robot according to a navigation algorithm is not just simple as making software. I was literally exhausted after countless testing of the robot in the field. I had to perform the test in the campus parking lot during late night when people drive their car out of the place to avoid accident.

In sum, I was satisfied with the result that my robot performed with limited number of components. Actually this project was done with just a RC car, controller and a GPS module. This robot doesn't even have any accelerometers or gyros which is very common in mobile robot! Although I still think I should have applied digital compass I think keeping the hardware as simple as possible enabled me to accomplish this project.

12. Documentation

13. Appendices

1. Budget

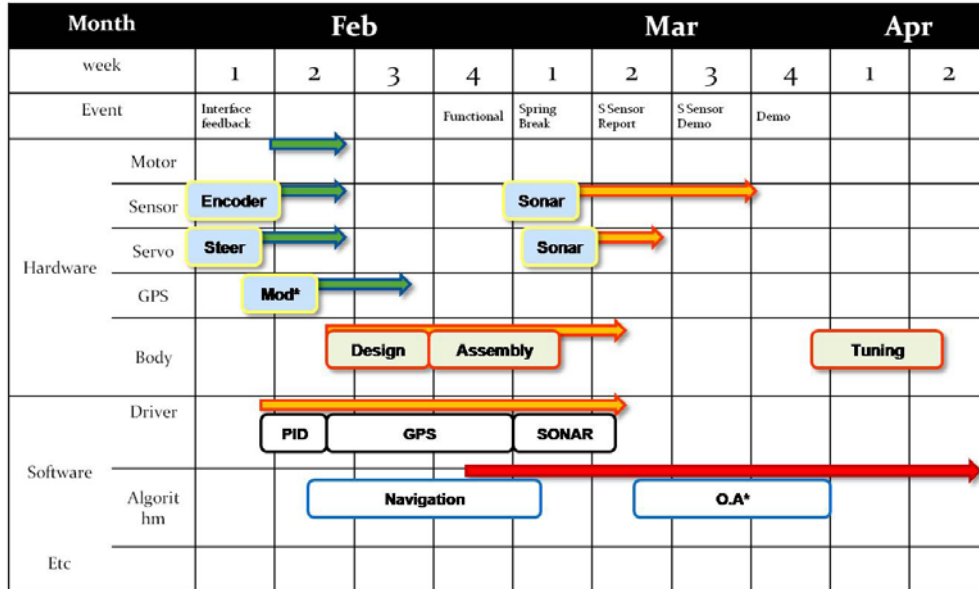
Item	Desc	Price	Qty	Sub Total	Remark
RC car		\$ -	1	\$ -	Pre owned
GPS receiver	GlobalSat BU0252 USB Receiver	\$ 36.95	1	\$ 45.95	Buygps.com
LCD module	20x4 Character, Blue Backlight	\$ 4.99	1	\$ 9.99	ebay
Sonar	Maxbotix LV-EZ1	\$ 24.95	2	\$ 53.20	Sparkfun.com
Encoder	Stegman Sick Oprical Encoder	\$ 28.00	1	\$ 34.00	ebay
SD-COM	SD type memory module	\$ 30.78	1	\$ 30.78	Eleparts.co.kr
Motor Driver		\$ -	1	\$ -	Pre owned
Servo Motor	Hitec HS-81 Micro(Sonar)	\$ 16.99	1	\$ 16.99	
	Hitec 9645MG (Steering)	\$ -	1	\$ -	Pre owned
Battery	7.2v NiCD 1800mAh	\$ 13.99	1	\$ 13.99	Hobbytown USA
MCU	HC12 T-Board	\$ -	1	\$ -	Pre owned
BDM	Debugger	\$ -	1	\$ -	Pre owned
Total				\$ 204.90	

Table 2 Total Budget

*Pre owned items are excluded from the budget

2. Project Schedule

Following Table shows development schedule of the project. I divided the project as Hardware part and software part. Each arrow represents the duration of each task.



Remarks: *Mod: Modification of the GPS module
O.A: Obstacle Avoidance

Table 3 Project schedule

14. References

- [1] Steven F. Barrett, "Embedded Systems Design and Applications with the 68HC12 and HCS12", pp.346-349, 2005
- [2] Lee Junseok, GPS NMEA code parsing source code,
- [3] Jason, GPS navigation, "<http://www.ke4nyv.com/navigation.htm>"
- [4] Joseph L.Jones, "Mobile Robots: Inspiration to Implementation", AK Peters, 1999