AUTOMATING VEHICLE NAVIGATION

Raphael Ahmed
Student ID: 06310045

Md.Intekhabul Hafiz
Student ID: 05101021

Department of Computer Science and Engineering

January 2009

BRAC University, Dhaka, Bangladesh

DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of Supervisor

Signature of Author

ACKNOWLEDGMENTS

First of all, all praise belongs to Allah, without His help we wouldn't have been able to do anything. We would then like to acknowledge the continuous support of our supervisor Dr. A. K. M. Abdul Malek Azad. We would also like to thank Syed Saiful Islam, Munshi Asadullah, Annajiat Alim Rasel, Mohammad Asaduzzaman Al Faruq and Ahmed All Amin for their enthusiasm and interest in our thesis that kept us motivated throughout the thesis period.

ABSTRACT

Technological advancements in one sector integrate with other sectors resulting into generation of further developments. GPS is the output from researches for better navigation systems, which is now used by field researchers who travel to uncharted distant places so that they don't get lost and of course know where they are going.

Researchers have now brought up a new dimension to our lifestyle with the introduction of unmanned vehicles on every possible surface: from air to land and above and under water. Unmanned vehicles are no more innovative characters in Sci-fi thrillers, but a reality and it is possible using GPS.

Our area of interest for the thesis is the implementation of GPS in an unmanned vehicle that would take the input from a user in terms of latitude and longitude and then make its way to the destination automatically by creating a path and maintaining it till it reaches there.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Introduction

Land, marine and air navigation systems have seen many advances with the advancements of technologies such as DECCA Navigator System [1], LORAN [2] and many more with the latest and the most accurate to date being the Global Positioning System or GPS [3] in short.

With the introduction of this service to the civil population, researches resulted into commercial units offering navigation aids for automobiles, ships and aircrafts.

1.2 Project Overview

The vehicle consists of two motors, responsible for backward and forward movement and right and left rotation respectively, a microcontroller interfaced with a GPS device and user input interface. The user inputs the desired destination in terms of longitude and latitude. The microcontroller reads the data from GPS device and creates a path to the final destination. It then gives necessary signal to the motors. As the vehicle starts to move, the microcontroller keeps on taking feedback from the GPS device making necessary corrections by constantly sending signal to the motors. The process is continued till the vehicle reaches the destination. The block diagram in the figure below illustrates the entire process.

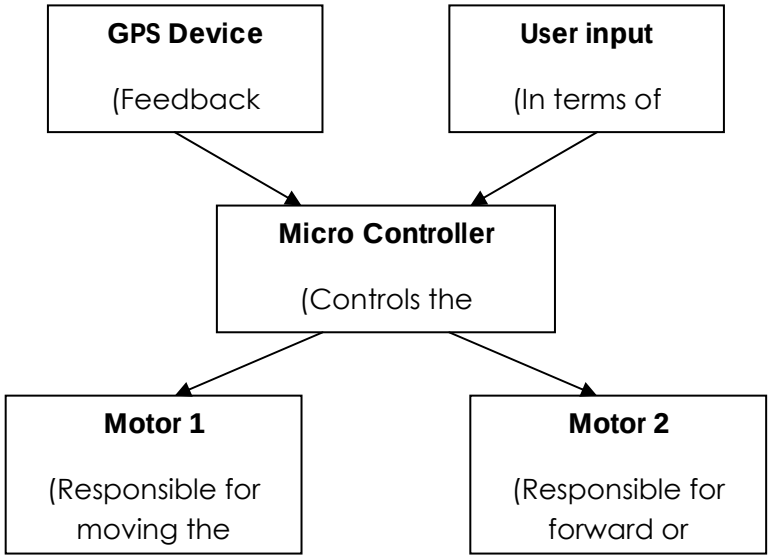

Fig. 1.1 System Overview

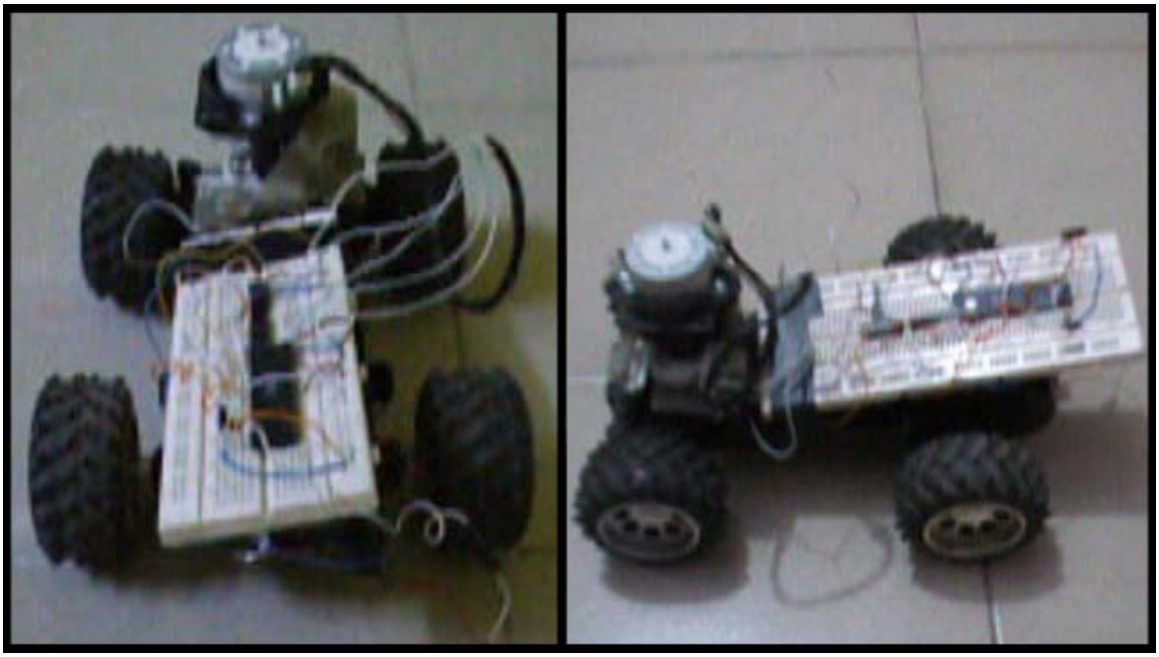

Fig. 1.2 Pictures of the car with embedded system

1.3 Global Positioning System

It is a system by which an object's position can be pin pointed. A GPS [4] receiver's job is to locate four or more of the satellites revolving the planet, figure out the distance to each, and use this information to deduce its own location. This operation is based on a simple mathematical principle called trilateration.

A GPS receiver calculates the distance to GPS satellites by timing a signal's journey from satellite to receiver. As it turns out, this is a fairly elaborate process. At a particular time (let's say midnight), the satellite begins transmitting a long, digital pattern called a pseudo-random code. The receiver begins running the same digital pattern also exactly at midnight. When the satellite's signal reaches the receiver, its transmission of the pattern will lag a bit behind the receiver's playing of the pattern.

The length of the delay is equal to the signal's travel time. The receiver multiplies this time by the speed of light to determine how far the signal traveled. Assuming the signal traveled in a straight line, this is the distance from receiver to satellite. **Differential GPS** (DGPS) [4] helps correct some errors. The basic idea is to gauge GPS inaccuracy at a stationary receiver station with a known location. Since the DGPS hardware at the station already knows its own position, it can easily calculate its receiver's inaccuracy. The station then broadcasts a radio signal to all DGPS-equipped receivers in the area, providing signal correction information for that area. In general, access to this correction information makes DGPS receivers much more accurate than ordinary receivers.

1.4 Micro-Controller

The micro-controller used in the system is ATMEL's AT89C51. The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer [5] with 4Kbytes of Flash programmable and erasable read only memory (PEROM) [6]. The device is

manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set and pin-out. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

Fig. 1.3 IC: AT89C51

CHAPTER 2: THE OVERALL SYSTEM

2.1 System Data Flow

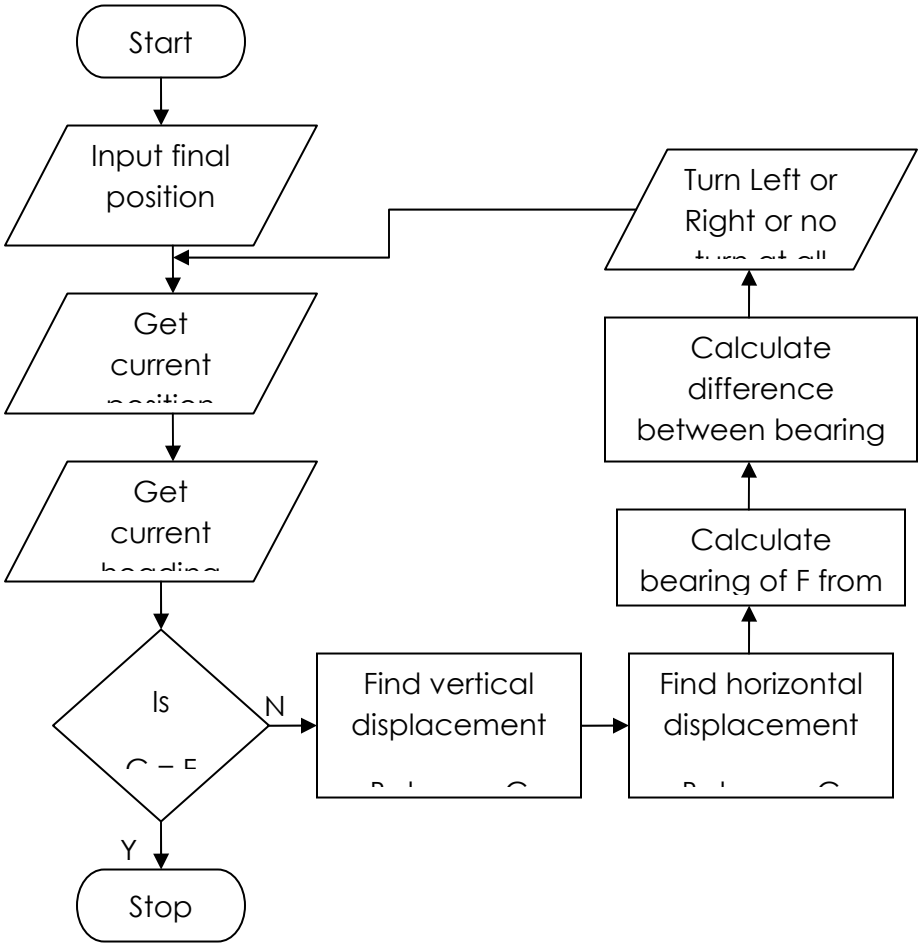

Fig. 2.1 System data flow

This is how the overall system works:

1. First it takes input from the user in terms of longitude and latitude.
2. Then it takes reading of the current position of the vehicle from the GPS device.

3. Then it takes reading from the electronic compass, which gives the current heading.
4. It then checks whether the current position is equal to the final destination. If it is yes then it stops and if it is no then it goes to the next step.
5. At this point it finds the vertical displacement between current and final destination.
6. Then it finds the horizontal displacement between current and final destination.
7. Using the values of vertical and horizontal displacement, it calculates the bearing of final destination from the current position.
8. It then calculates the difference between bearing of the final destination from the current position and the direction the vehicle is currently heading.
9. Based on the values found the system decides in which direction should the vehicle move or should it move at all. The entire process from step 2 is repeated till the vehicle reaches the destination.

2.1.1 Calculating vertical displacement

This is how the system calculates the vertical displacement. Figure 2.2 shows the flowchart of the algorithm.

1. It checks if the current location is in the northern hemisphere. Based on the decision, it takes any of the two paths.
2. Assuming the current location was in the northern hemisphere, it then checks whether the final destination is also in the northern hemisphere. There again based on the reading it takes any of the two paths.
3. Assuming the final position is also in the northern hemisphere, it then checks if the current latitude is equal to the final latitude. Based on the decision, it takes any of the two paths.
4. Assuming the current is not equal to final, it then check if the destination latitude is greater than the current latitude. Based on the decision, it takes any of the two paths.

5. Assuming the final is greater than current, it then calculates the difference between final and current and this gives the vertical displacement. It then goes on to the next step which is to find out the horizontal displacement.

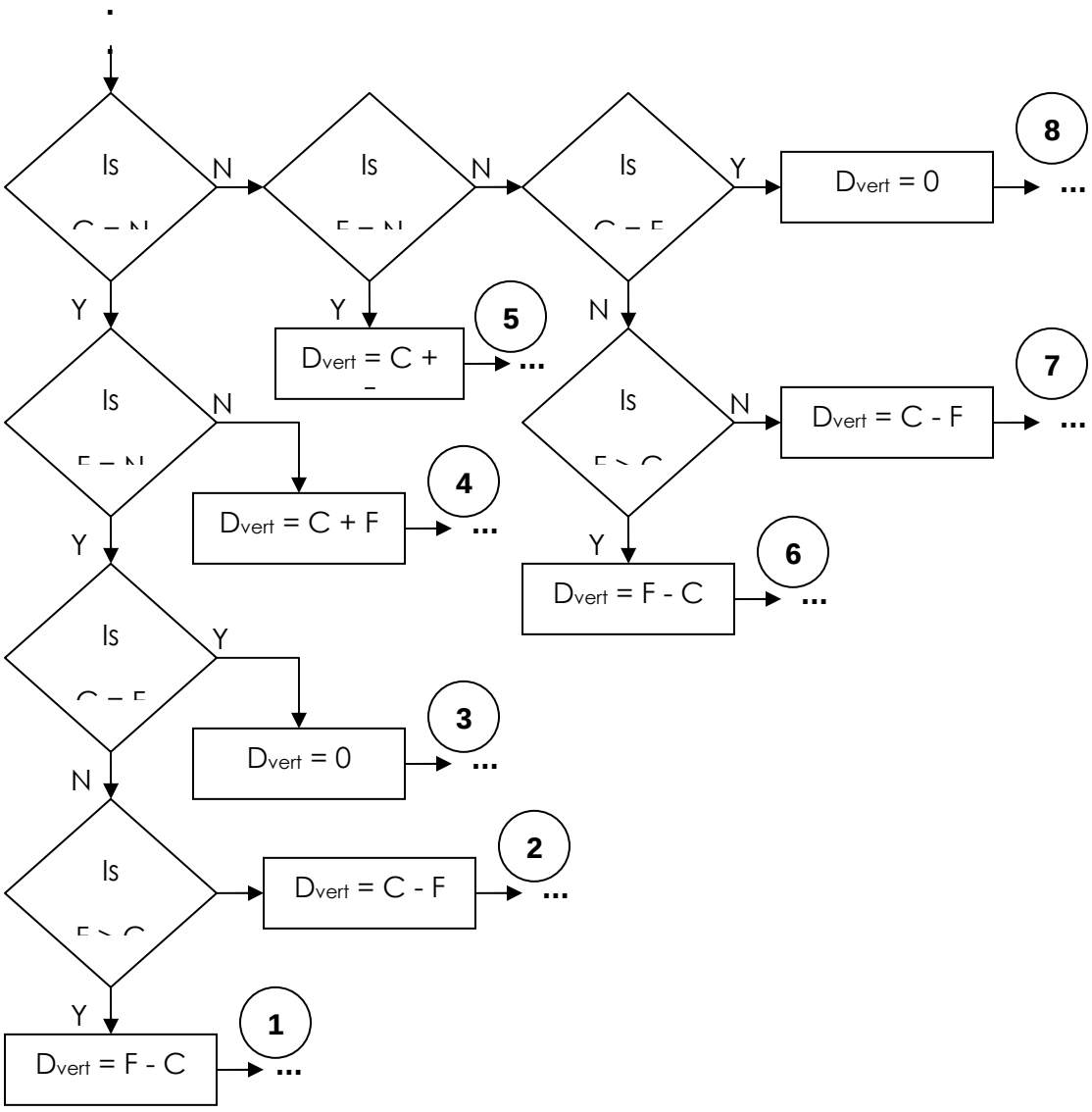

Fig. 2.2 Vertical displacement calculation flowchart

2.1.2 Calculating horizontal displacement

This is how the system calculates the horizontal displacement. Figure 2.3 shows the flowchart of the algorithm.

1. It checks if the current location is in the eastern hemisphere. Based on the decision, it takes any of the two paths.
2. Assuming the current location was in the eastern hemisphere, it then checks whether the final destination is also in the eastern hemisphere. There again based on the reading it takes any of the two paths.
3. Assuming the final position is also in the eastern hemisphere, it then checks if the current longitude is equal to the final longitude. Based on the decision, it takes any of the two paths.
4. Assuming the current is not equal to final, it then check if the destination longitude is greater than the current longitude. Based on the decision, it takes any of the two paths.
5. Assuming the final is greater than current, it then calculates the difference between final and current and this gives the horizontal displacement. It then goes on to the next step which is to calculate the bearing of the final destination from the current location.

From the algorithm of finding the vertical displacement, we find that there are eight possible results. Also, the horizontal displacement algorithm gives eight possible results. Since the algorithm of finding horizontal displacement is applied immediately after the vertical displacement algorithm, thus that leaves us with sixty four different combinations.

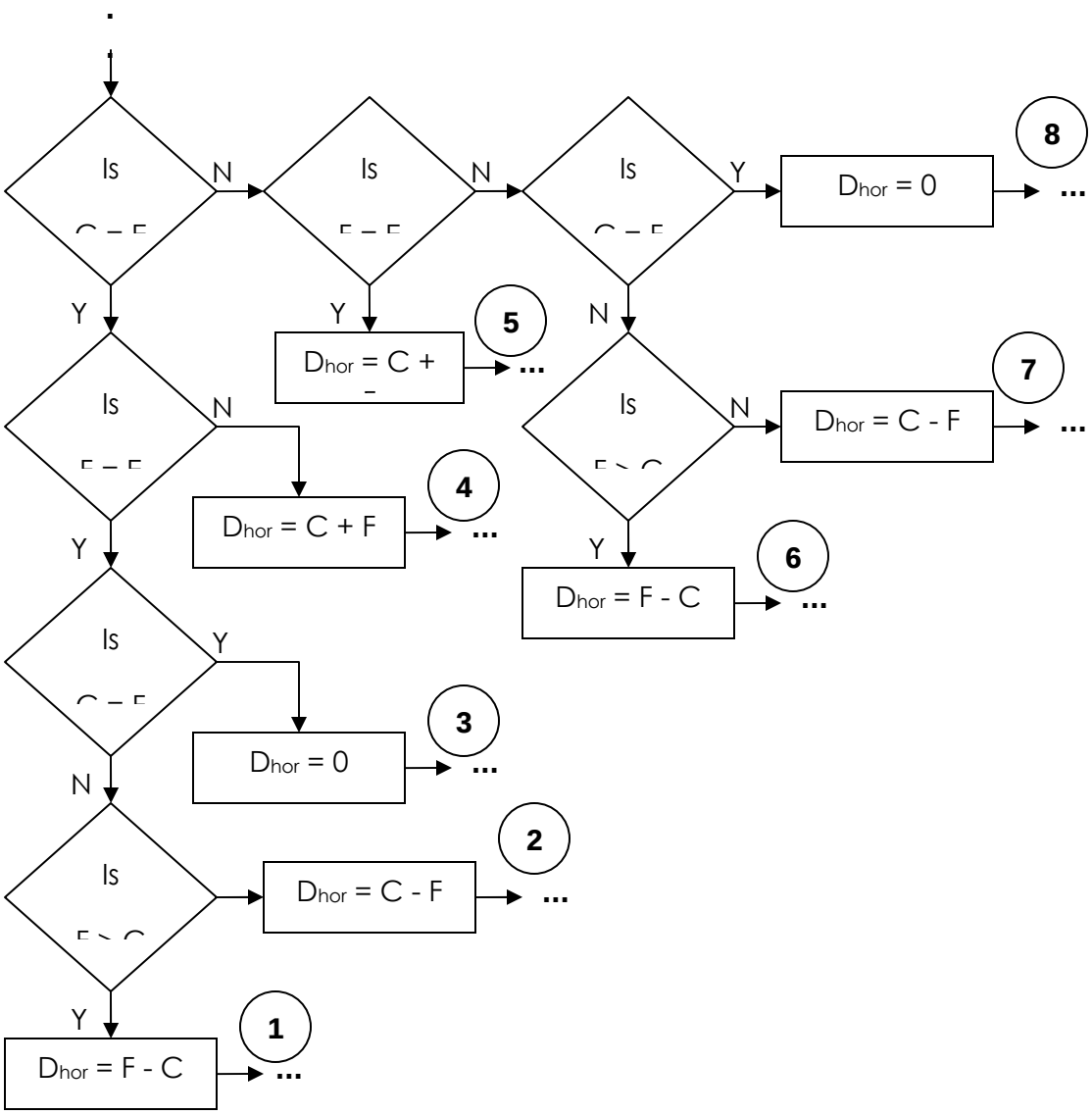

Fig. 2.3 Horizontal displacement calculation flowchart

Putting these sixty four different combinations in a table we found out that different formulae need to be used to calculate the bearing of the final destination from the current location. Table 2.1 to 2.8 shows the results of the bearings.

Table 2.1

Calculation of bearing (C=N,F=N,C≠F F>C)

1							
C=N F=N C≠F F>C							
1	2	3	4	5	6	7	8
C=E F=E C≠F F>C	C=E F=E C≠F F<C	C=E F=E C=F	C=E F≠E	C≠E F=E	C≠E F≠E C≠F F>C	C≠E F≠E C≠F F<C	C≠E F≠E C=F
B_1	B_2	B_3	B_2	B_1	B_2	B_1	B_3

Table 2.2

Calculation of bearing (C=N, F=N, C≠F, F<C)

2							
C=N F=N C≠F F<C							
1	2	3	4	5	6	7	8
C=E F=E C≠F F>C	C=E F=E C≠F F<C	C=E F=E C=F	C=E F≠E	C≠E F=E	C≠E F≠E C≠F F>C	C≠E F≠E C≠F F<C	C≠E F≠E C=F
B_7	B_8	B_5	B_8	B_7	B_8	B_7	B_5

Table 2.3

Calculation of bearing (C=N, F=N, C=F)

3							
C=N F=N C=F							
1	2	3	4	5	6	7	8
C=E F=E C≠F F>C	C=E F=E C≠F F<C	C=E F=E C=F	C=E F≠E	C≠E F=E	C≠E F≠E C≠F F>C	C≠E F≠E C≠F F<C	C≠E F≠E C=F
B_4	B_6	STOP	B_6	B_4	B_6	B_4	STOP

Table 2.4

Calculation of bearing (C=N, F≠N)

4							
C=N F≠N							
1	2	3	4	5	6	7	8
C=E F=E C≠F F>C	C=E F=E C≠F F<C	C=E F=E C=F	C=E F≠E	C≠E F=E	C≠E F≠E C≠F F>C	C≠E F≠E C≠F F<C	C≠E F≠E C=F
B_7	B_8	B_5	B_8	B_7	B_8	B_7	B_5

Table 2.5

Calculation of bearing (C≠N, F=N)

5							
C≠N F=N							
1	2	3	4	5	6	7	8
C=E F=E C≠F F>C	C=E F=E C≠F F<C	C=E F=E C=F	C=E F≠E	C≠E F=E	C≠E F≠E C≠F F>C	C≠E F≠E C≠F F<C	C≠E F≠E C=F
B_1	B_2	B_3	B_2	B_1	B_2	B_1	B_3

Table 2.6

Calculation of bearing (C≠N, F≠N, C≠F, F>C)

6							
C≠N F≠N C≠F F>C							
1	2	3	4	5	6	7	8
C=E F=E C≠F F>C	C=E F=E C≠F F<C	C=E F=E C=F	C=E F≠E	C≠E F=E	C≠E F≠E C≠F F>C	C≠E F≠E C≠F F<C	C≠E F≠E C=F
B_7	B_8	B_5	B_8	B_7	B_8	B_7	B_5

Table 2.7

Calculation of bearing (C≠N, F≠N, C≠F, F<C)

7							
C≠N F≠N C≠F F<C							
1	2	3	4	5	6	7	8
C=E F=E C≠F F>C	C=E F=E C≠F F<C	C=E F=E C=F	C=E F≠E	C≠E F=E	C≠E F≠E C≠F F>C	C≠E F≠E C≠F F<C	C≠E F≠E C=F
B_1	B_2	B_3	B_2	B_1	B_2	B_1	B_3

Table 2.8

Calculation of bearing (C≠N, F≠N, C=F)

8							
C≠N F≠N C=F							
1	2	3	4	5	6	7	8
C=E F=E C≠F F>C	C=E F=E C≠F F<C	C=E F=E C=F	C=E F≠E	C≠E F=E	C≠E F≠E C≠F F>C	C≠E F≠E C≠F F<C	C≠E F≠E C=F
B_4	B_6	STOP	B_6	B_4	B_6	B_4	STOP

The formulae for finding the bearings, B_1 to B_8 are given in equations 2.1 to 2.8.

$$B_1 = \tan^{-1} (D_{hor} / D_{vert}) \quad (2.1)$$

$$B_2 = 360^o - \tan^{-1} (D_{hor} / D_{vert}) \quad (2.2)$$

$$B_3 = 0^o \quad (2.3)$$

$$B_4 = 90^o \quad (2.4)$$

$$B_5 = 270^o \quad (2.5)$$

$$B_6 = 180^o \quad (2.6)$$

$$B_7 = 180^o - \tan^{-1} (D_{hor} / D_{vert}) \quad (2.7)$$

$$B_8 = 180^o + \tan^{-1} (D_{hor} / D_{vert}) \quad (2.8)$$

2.1.3 Calculating front wheel rotation angle

This is how the system decides the front wheel rotation. Figure 2.4 shows the flowchart of the algorithm.

1. It checks if the current heading is equal to bearing of final destination from current position. If it is Yes then there is no rotation and it takes new reading from GPS
2. Here it checks if heading is greater than bearing.
3. Assuming the Heading to be greater than bearing, it then calculates the difference between heading and bearing.
4. It then checks if the difference is greater than 180^{o}.
5. Assuming the difference to be greater than 180^{o} It checks if the wheels were turned right previously.
6. Assuming that the wheels were not turned, it then turns the wheels right.

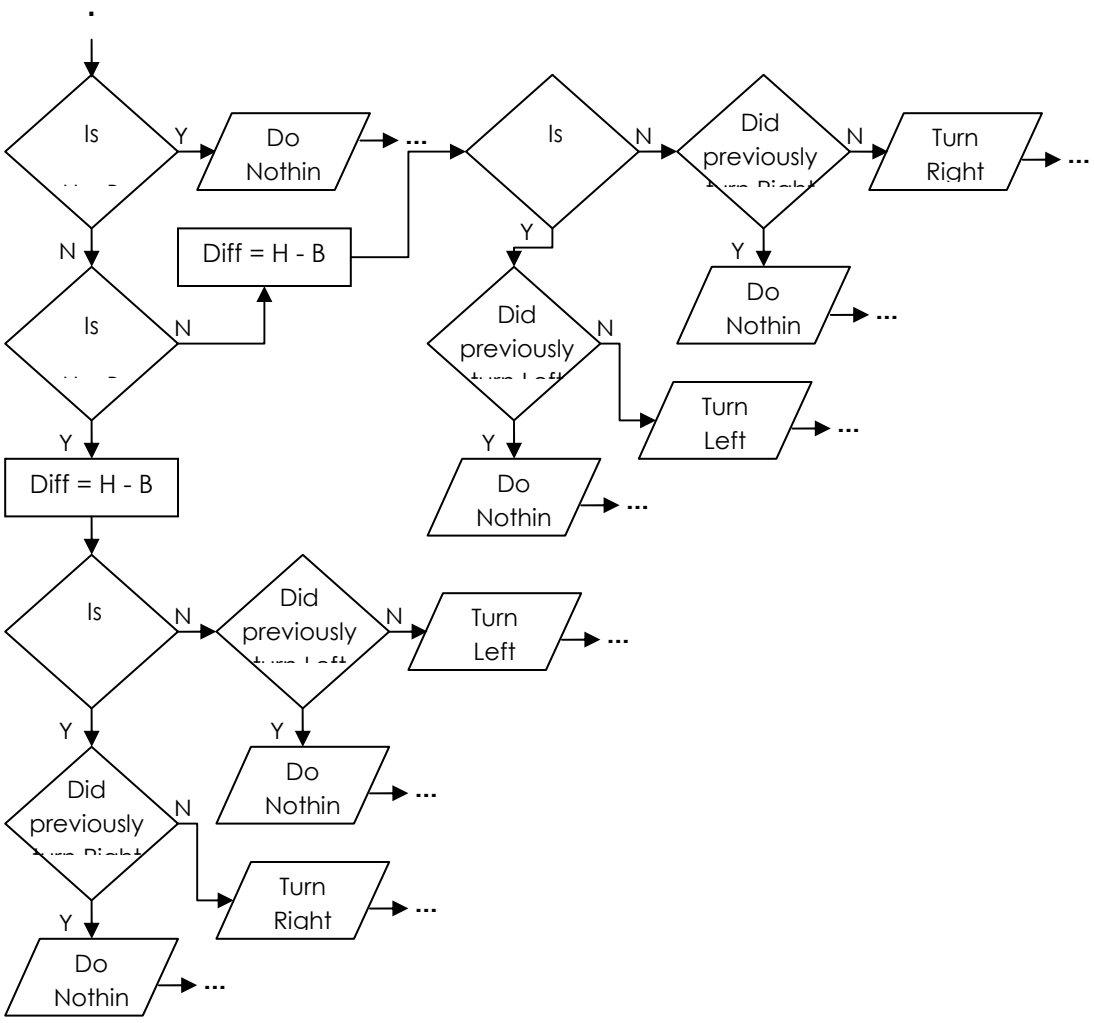

Fig 2.4 Calculating front wheel rotation angle

2.1.4 Determining the turn angle (alternate):

To account the turning arc for a vehicle trying to make a turn the following is another solution. The arc of turn is to be considered as it will cause a final displacement slightly different from the desired displacement. Thus, in order to determine the arc of displacement, we need to find the optimum angle for which a minimum optimal arc is going to be traversed by the vehicle.

The angle found by the previous step is denoted by the red arc and the angle to which the wheel must be turned is denoted by the green arc. This is the angle that we need to find which will maintain a uniform arc to the destination B.

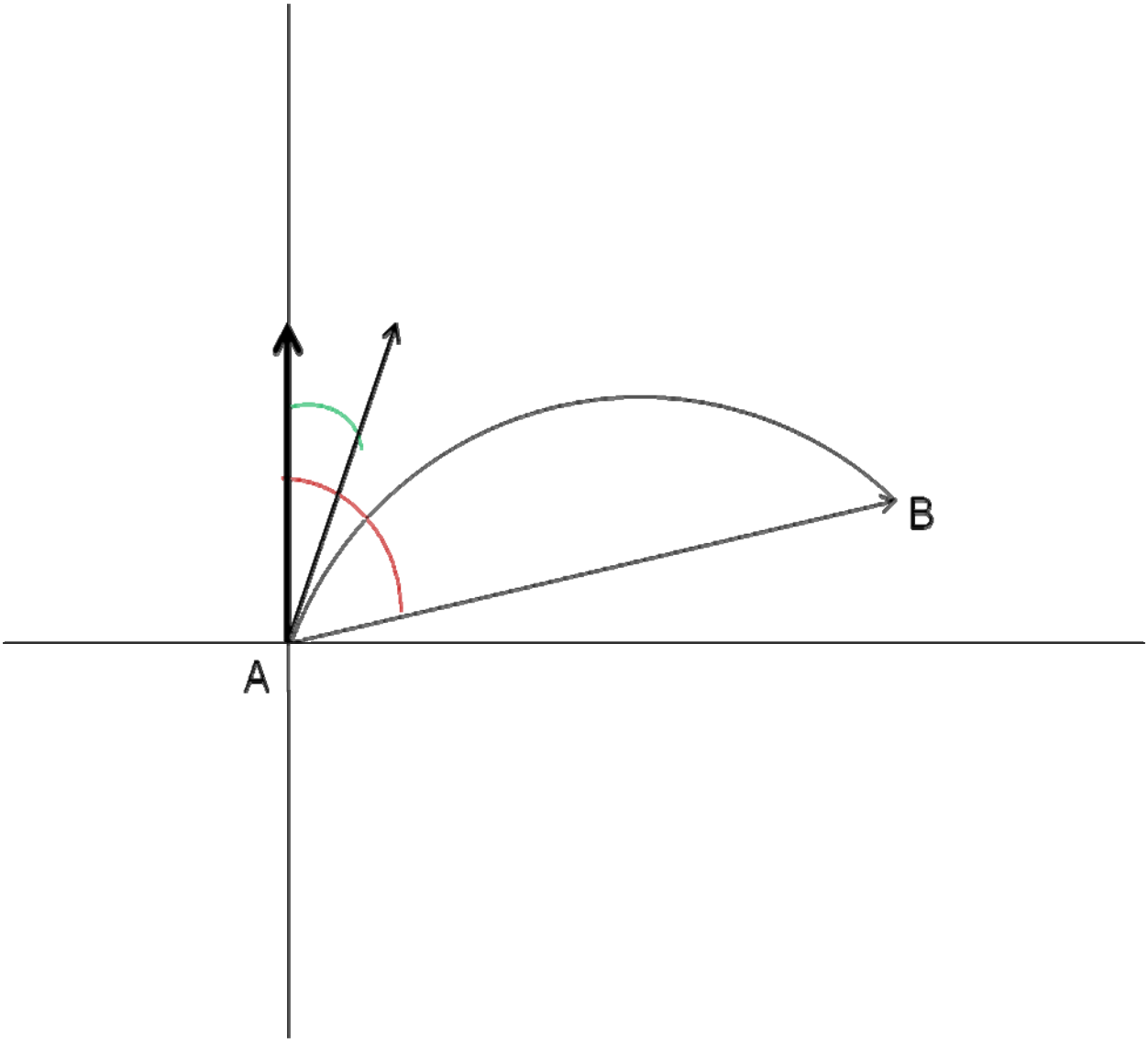

Fig. 2.5 Turning arc

2.1.4.1 Finding the arc length

The arc length [7] can be found given the initial position, heading and the final destination using the formula:

$$\varphi = \tan^{-1}\left[\frac{-(x_2 - x_1)(\tan(90 - \theta)) + (y_2 - y_1)}{((y_2 - y_1)\tan(90 - \theta)) + (x_2 - x_1)}\right] \quad (2.9)$$

$$length\ of\ Arc = \varphi\frac{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}}{\sin\varphi} \quad (2.10)$$

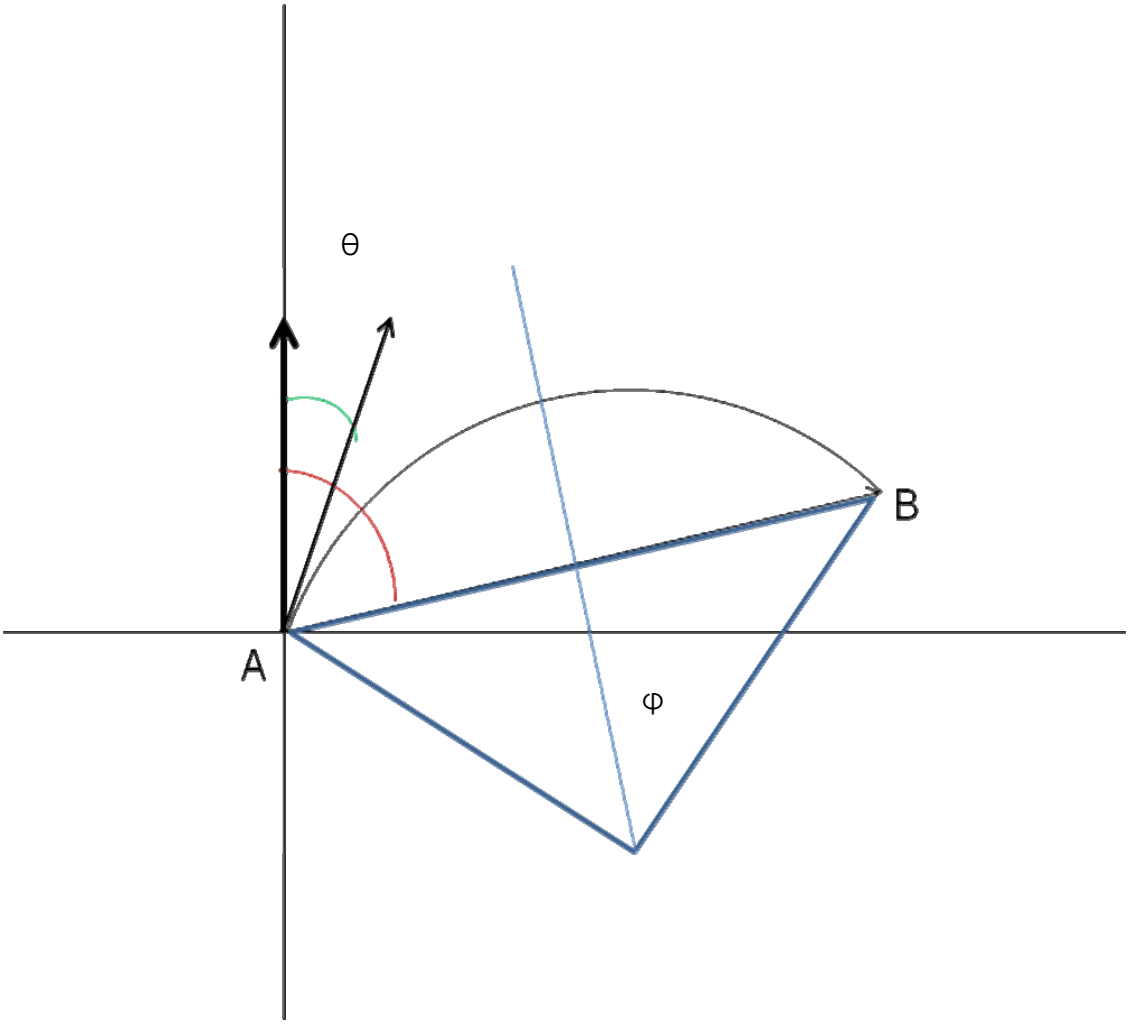

Fig. 2.6 Determining the arc length

2.1.4.2 Finding the optimum wheel turn angle

As we are now equipped with the value of theta, we need to find the optimum angle to which the wheel must be turned to traverse the arc accurately. Now, this depends on many factors like, vehicle size and shape, displacement of the wheels and the forward thrust power. So, we have to find an optimal angle that will be least affected by all of these factors. Thus, an optimal angle would be T% of arc length, where value of T would depend on the above factors.

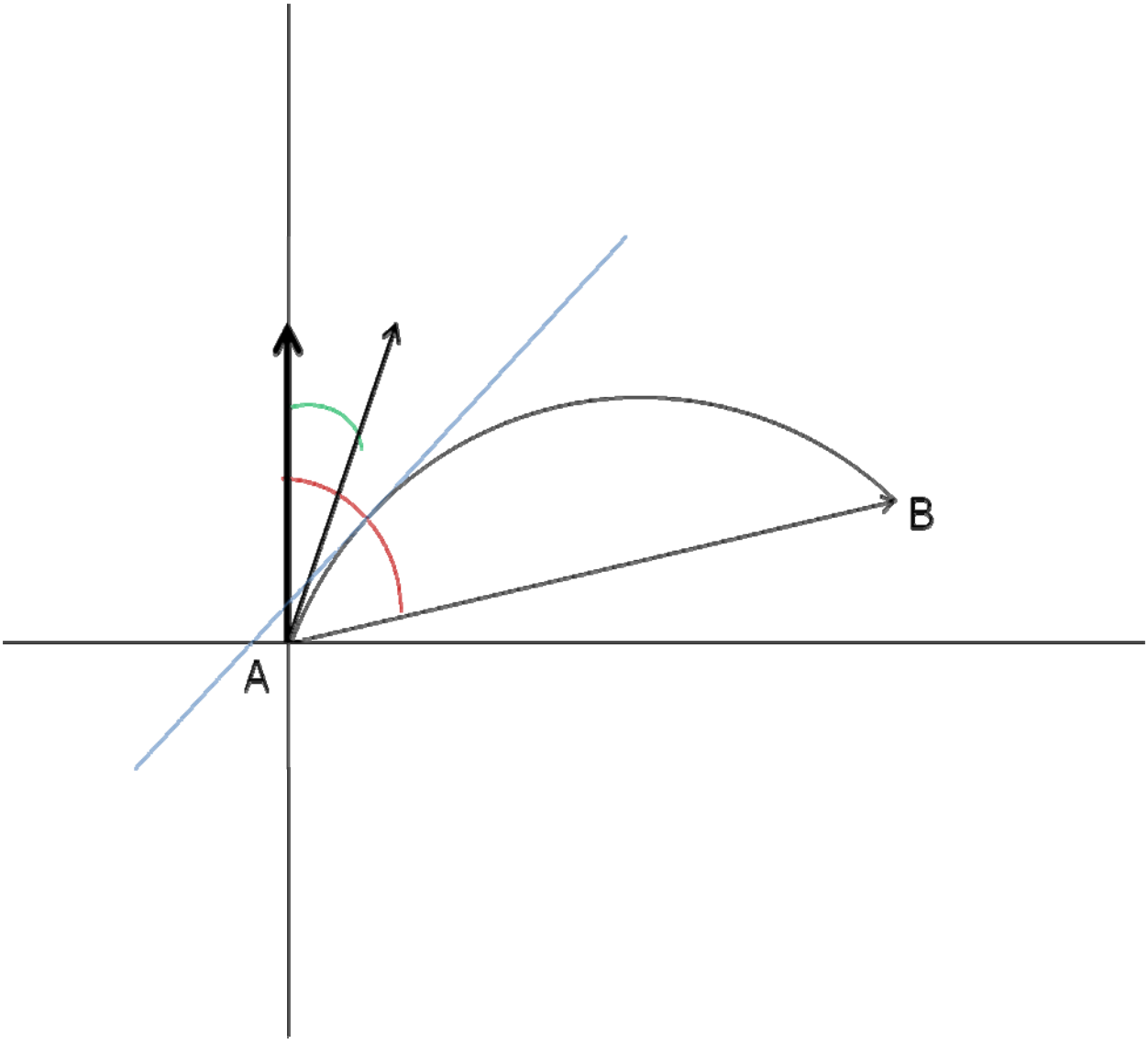

Fig. 2.7 Determining the optimum turn angle

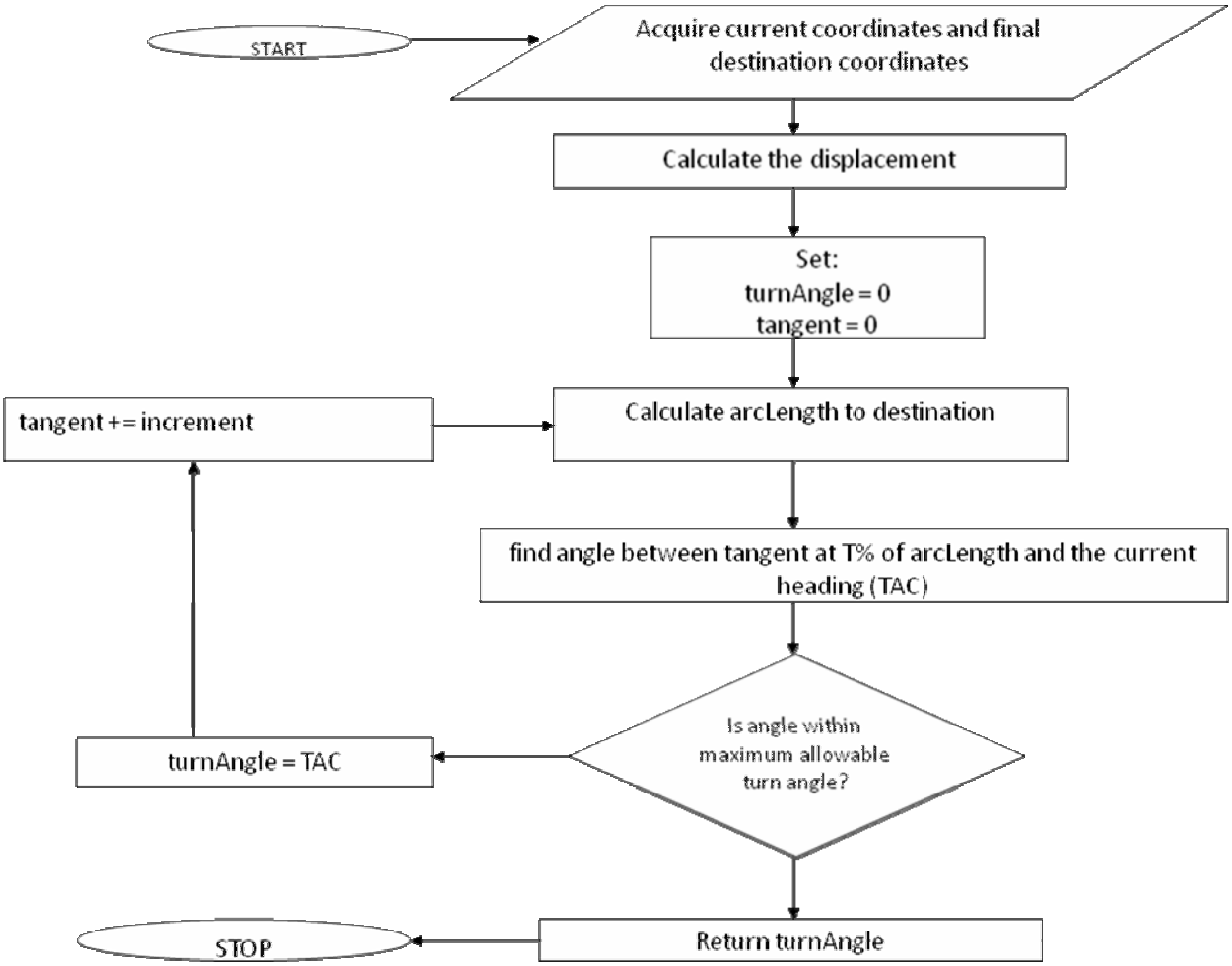

Fig. 2.8 Flowchart for determining the turn angle

The flowchart above shows the flow of data for finding the optimum turn angle.

2.1.5 Applying the alternate

Now, this method can be used to straighten the vehicle to align with the final heading. This is illustrated in the following diagrams.

In the first one, the slope of tangent to the arc at the end point is compared with the slope of line from that point to the point B. If the slopes are equal then the wheels are straightened and the car moves forward heading straight towards heading B.

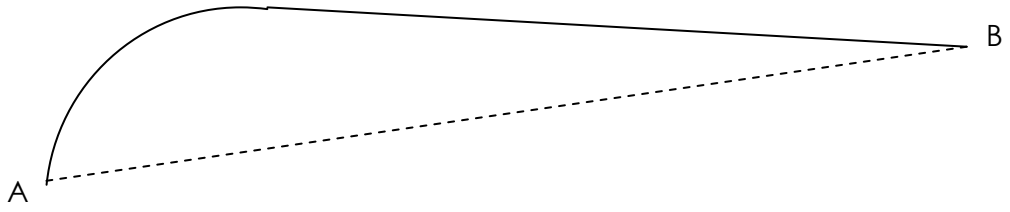

Fig. 2.9 Aligning the vehicle heading with single arc

The figure below is a dynamic approach to aligning the vehicle to the destination bearing.

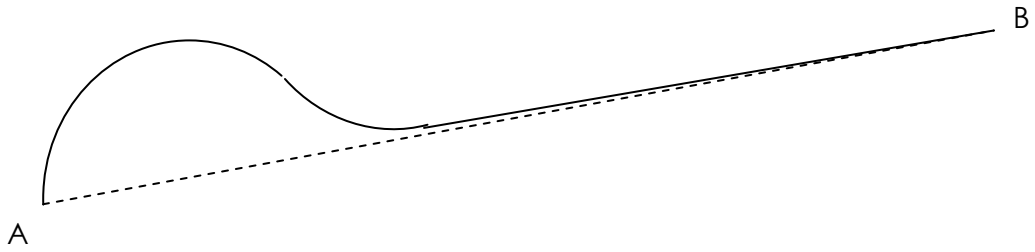

Fig. 2.10 Aligning the vehicle heading with multiple arcs

CHAPTER 3: MOTORS AND DRIVERS

For our thesis we used a stepper motor to control the front wheel rotation and we used a normal DC motor for the rear wheels for forward and backward movement.

To control the stepper motor we used IC: ULN2003 and to control the DC motor we used IC: L293D. Figure 3.1 illustrates how stepper motor can be controlled using ULN2003. Figure 3.2 illustrates how the DC motor can be controlled using L293D and figure 3.3 shows the circuit diagram of the entire embedded system.

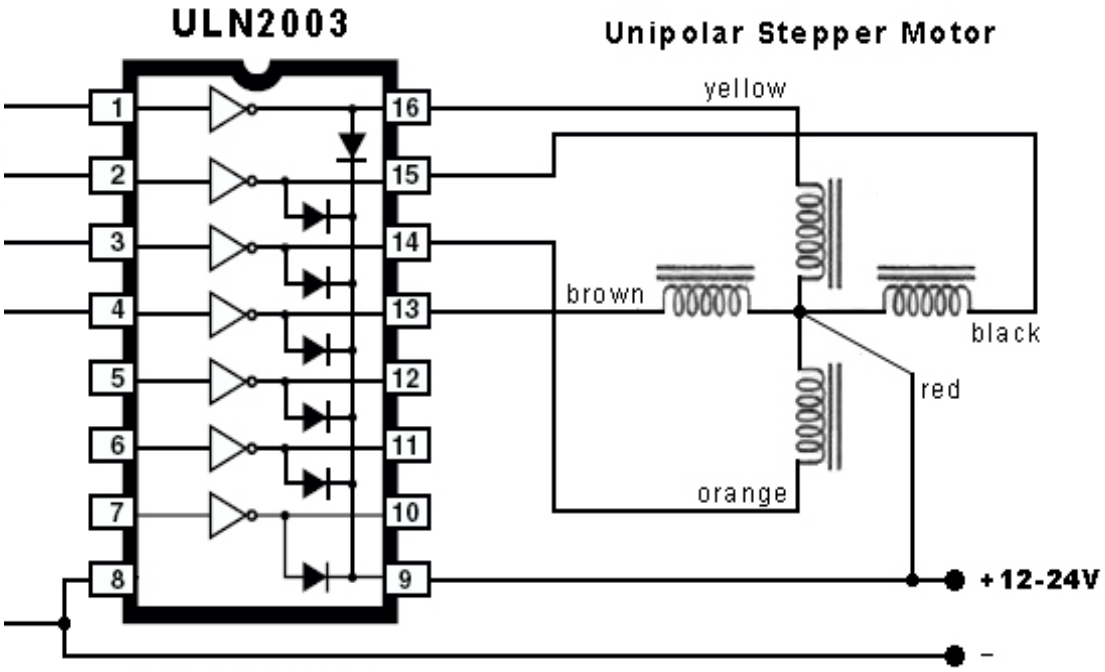

Fig. 3.1 Controlling unipolar stepper motor using ULN2003

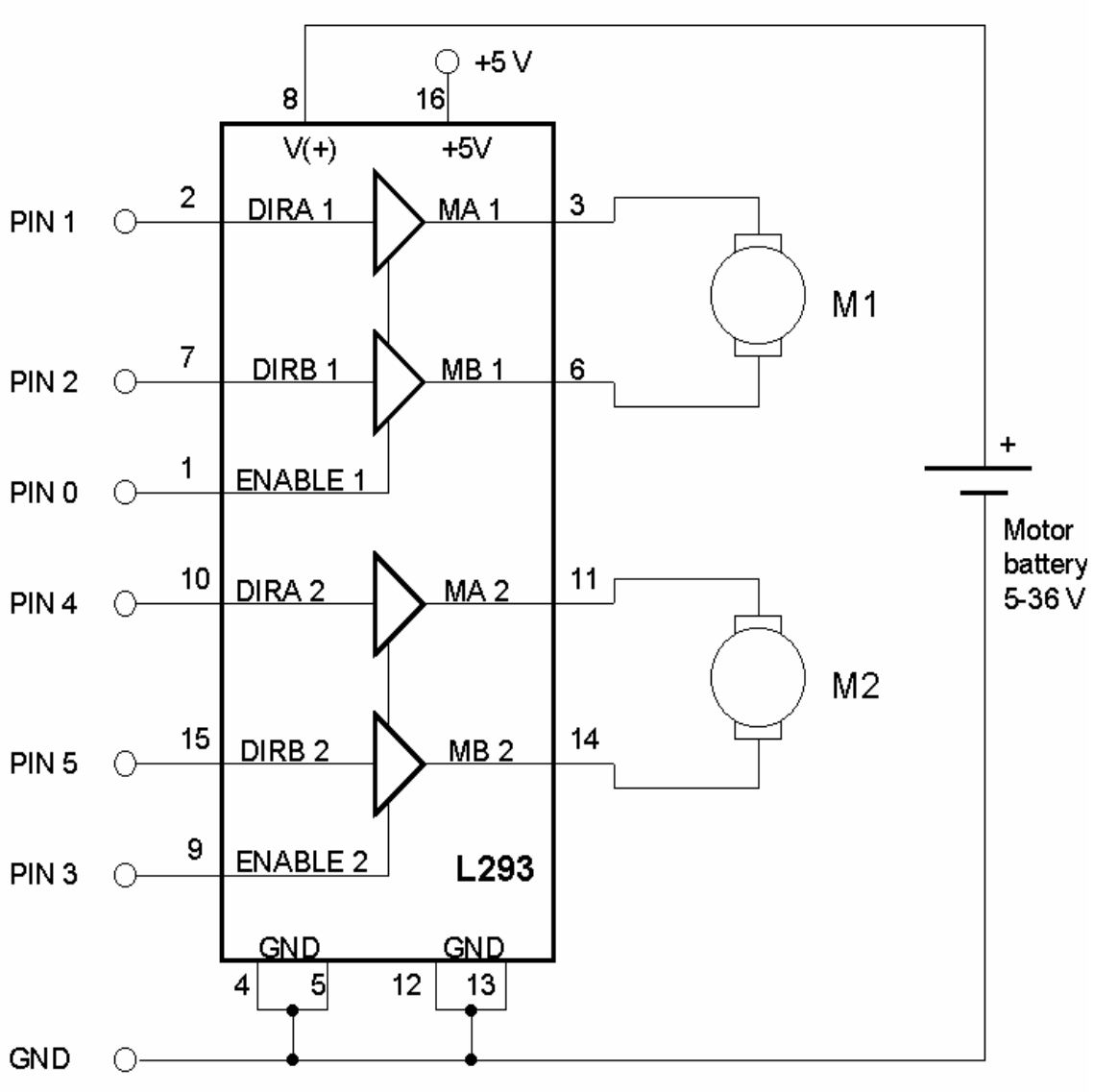

Fig. 3.2 Controlling DC motor using L293D

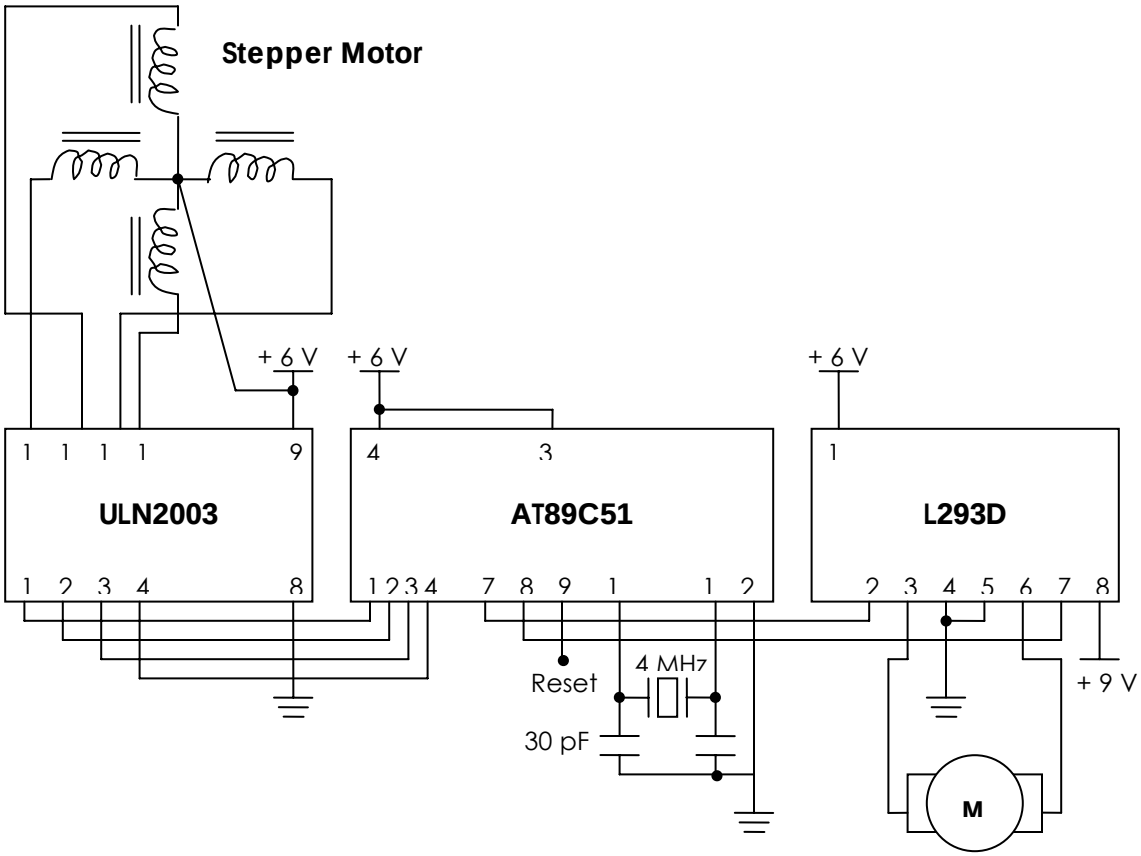

Fig. 3.3 Circuit diagram of entire embedded system

CHAPTER 4: PROGRAMMER

4.1 TOP 2005+ Universal Programmer

To burn our programs into the microcontroller, we used a universal programmer manufactured by the Chinese company TOPWIN. The programmer comes with installation CD with the software and a user manual. It has USB interface and very easy to use.

We wrote our programs in C and compiled and converted to hex files, which then we downloaded into the microcontroller using the universal programmer and the software that came with it.

Fig. 4.1 Universal Programmer

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

The Automation system was designed to work acquiring the coordinate data from a GPS receiver. The system would run according to the actual algorithms described in the respective sections. However, during test sessions, factors like vehicle size, weight, displacement of front and rear wheels, movement environment like surface friction, skidding of wheels etc affected the actual path traversed by the vehicle. Proper GPS receiver was not acquired during the course of the development, which had made it impossible to carry out any tests that involved the data from a GPS receiver.

5.2 Future Work

Thus, the system still requires to be worked upon to be made fully active. Among all the other developments, we would suggest the use of obstacle avoidance techniques and relevant sensors if the vehicle is to be used in a dynamic environment where the obstructions are not known before hand. To use a vehicle equipped with our automating system on regular traffic roads, the system would need an integrated map which could assist it following the road exactly and it would also need integration with the existing traffic system.

APPENDICES

Appendix A Vehicle Movement in Multidirections

```
unsigned int DATA;
void delay(unsigned int x){     /* x * 1ms delay */
      unsigned int k, l;
      for(k=0;k<x;++k){
            for(l=0;l<50;++l);
      }
}
void Sleep(unsigned int x){     /* x * 1ms delay */
      unsigned int k, l;
      for(k=0;k<x;++k){
            for(l=0;l<50;++l);
      }
}
void Out32(unsigned int DATAa, unsigned int pval){
      DATAa=0;
      P1=pval;
}
void stop (void){
            unsigned int pval = 0x00;
```

```
            P1=pval;
}
void left (void){
      unsigned int delayTime = 20;
      unsigned int m_angle=10;
      unsigned int angle = (m_angle/(3.75))*96;
      unsigned int i=0;
      while (i<angle){
            switch(i%4){
                  case 0:
                        Out32(DATA,0x08); delay(delayTime);
                        ++i;break;
                  case 1:
                        Out32(DATA,0x04); delay(delayTime);
                        ++i;break;
                  case 2:
                        Out32(DATA,0x02); delay(delayTime);
                        ++i;break;
                  case 3:
                        Out32(DATA,0x01); delay(delayTime);
                        i++;break;
                  default:
                        ++i;
            }
```

```
	}

}

unsigned int Inp32(unsigned int x){
	x=P1;
	return x;
}

void left1 (unsigned int m_angle){

	unsigned int delayTime = 2;
	unsigned int angle = (m_angle/(3.75*2))*96;
	unsigned int i=0;
	unsigned int currentState=0;
	while (i<angle){
		currentState=Inp32(DATA);
		if((currentState & 0x10) == 0x10){
			switch(i%4){
			case 0:
				Out32(DATA,0x18); Sleep(delayTime);
				++i;break;
			case 1:
				Out32(DATA,0x14); Sleep(delayTime);
```

```
                ++i;break;
        case 2:
                Out32(DATA,0x12); Sleep(delayTime);
                ++i;break;
        case 3:
                Out32(DATA,0x11); Sleep(delayTime);
                i++;break;
        default:
                ++i;
        }
}

else if((currentState & 0x20) == 0x20){
        switch(i%4){
        case 0:
                Out32(DATA,0x28); Sleep(delayTime);
                ++i;break;
        case 1:
                Out32(DATA,0x24); Sleep(delayTime);
                ++i;break;
        case 2:
                Out32(DATA,0x22); Sleep(delayTime);
                ++i;break;
        case 3:
```

```
                Out32(DATA,0x21); Sleep(delayTime);
                i++;break;
        default:
                ++i;
        }
}

else{
        switch(i%4){
        case 0:
                Out32(DATA,0x08); Sleep(delayTime);
                ++i;break;
        case 1:
                Out32(DATA,0x04); Sleep(delayTime);
                ++i;break;
        case 2:
                Out32(DATA,0x02); Sleep(delayTime);
                ++i;break;
        case 3:
                Out32(DATA,0x01); Sleep(delayTime);
                i++;break;
        default:
                ++i;
        }
```

```
		}
	}

}



void right1 (int m_angle){
	/*	m_angle/=2; */
	unsigned int delayTime = 2;
	unsigned int angle = (m_angle/(3.75*2))*96;
	unsigned int i=0;
	unsigned int currentState=0;
	while (i<angle){
		currentState=Inp32(DATA);
		if((currentState & 0x10) == 0x10){
			switch(i%4){
			case 3:
				Out32(DATA,0x18); Sleep(delayTime);
				++i;break;
			case 2:
				Out32(DATA,0x14); Sleep(delayTime);
				++i;break;
			case 1:
				Out32(DATA,0x12); Sleep(delayTime);
```

```
            ++i;break;
        case 0:
            Out32(DATA,0x11); Sleep(delayTime);
            ++i;break;
        default:
            ++i;
        }
    }

    else if((currentState & 0x20) == 0x20){
        switch(i%4){
        case 3:
            Out32(DATA,0x28); Sleep(delayTime);
            ++i;break;
        case 2:
            Out32(DATA,0x24); Sleep(delayTime);
            ++i;break;
        case 1:
            Out32(DATA,0x22); Sleep(delayTime);
            ++i;break;
        case 0:
            Out32(DATA,0x21); Sleep(delayTime);
            ++i;break;
        default:
```

```
                ++i;
            }
        }

        else{
            switch(i%4){
            case 3:
                Out32(DATA,0x08); Sleep(delayTime);
                ++i;break;
            case 2:
                Out32(DATA,0x04); Sleep(delayTime);
                ++i;break;
            case 1:
                Out32(DATA,0x02); Sleep(delayTime);
                ++i;break;
            case 0:
                Out32(DATA,0x01); Sleep(delayTime);
                ++i;break;
            default:
                ++i;
            }
        }
    }
}
```

```
void right (void){
	int m_angle=10;
	unsigned int delayTime = 20;
	unsigned int angle = (m_angle/(3.75))*96;
	unsigned int i=0;
	while (i<angle){
		switch(i%4){
			case 3:
				Out32(DATA,0x08); Sleep(delayTime);
				++i;break;
			case 2:
				Out32(DATA,0x04); Sleep(delayTime);
				++i;break;
			case 1:
				Out32(DATA,0x02); Sleep(delayTime);
				++i;break;
			case 0:
				Out32(DATA,0x01); Sleep(delayTime);
				++i;break;
			default:
				++i;
		}
	}
```

```
}

void forward (void){
	unsigned int forwardTime=3000;
	Out32(DATA,0x10);
	Sleep(forwardTime);
	stop();
}

void reverse (void){
	unsigned int reverseTime=3000;
	Out32(DATA,0x20);
	Sleep(reverseTime);
	stop();
}

void main(){
	unsigned char c[]="SLFFRFFSBBRFFFLF0T";
	unsigned int i=0;
	unsigned int ext=0;
	P1=0;
	P2=0;
	P3=0;
	P0=1;
```

```
        while (c[i]!='T'){
                switch (c[i]){
                case 'S': stop(); break;
                case 'L': left(); break;
                case 'R': right(); break;
                case 'F': forward(); break;
                case 'B': reverse(); break;
                case 'T': stop(); ext=1; break;
                default: stop(); break;
                }

                if(ext==1) break;
                ++i;
        }
        while(1){
                P1=0x00;
        }

}
```

Appendix B Turn Angle Calculation and Implementation

```
double turnDegree (double x1, double y1, double x2, double y2){
        double minArcLength = 0.0 ;
```

```
	double ArcLength = 0.0;
	double maxArcAngle = 0.0;
	double arcAngle = 26.0;
	double decrement = 0.5;
	double centralAngle = 0.0;
	double phi = 0.0;
	double theta = 26.0;
	double X = 0.0;
	double Y = 0.0;
	Y = y2-y1;
	X = x2-x1;
	while (arcAngle>0.0){
		phi = atan( ((x1-x2) * tan(90-theta) + Y)/(Y * tan(90-
theta) + X));
		centralAngle = 2.0*phi;
		ArcLength = phi * sqrt( Y*Y + X*X ) / (sin ( phi ) );
		if(minArcLength > ArcLength){
			minArcLength = ArcLength;
			maxArcAngle = arcAngle;
		}
		arcAngle = arcAngle - decrement;
	}
	return maxArcAngle;
}
```

```
void processRun (void){
	unsigned char c[] = "abcdefghijklmnopqrstuvwxyz";
	unsigned char move[] = "0,0,2,2";
	unsigned int i = 0;
	unsigned int ext = 0;
	double turnAngle = 0;
	double x1;
	double x2;
	double y1;
	double y2;
	unsigned char *tok;
	unsigned char token[] = "abcd";
	tok = strtok (move, " ,\t\n");
	if (tok != NULL) {
		strcpy(token,tok);
		x1 = (double)atof(token);
		tok = strtok (NULL, " \t\n");
	}

	if (tok != NULL) {
		strcpy(token,tok);
		y1 = (double)atof(token);
		tok = strtok (NULL, " \t\n");
```

```
}
if (tok != NULL) {
	strcpy(token,tok);
	x2 = (double)atof(token);
	tok = strtok (NULL, " \t\n");
}
if (tok != NULL) {
	strcpy(token,tok);
	y2 = (double)atof(token);
	tok = strtok (NULL, " \t\n");
}



turnAngle = turnDegree (x1,y1,x2,y2);
rightAngle(turnAngle);
strcpy(c,"SFF");



P1=0;
P2=0;
P3=0;
P0=1;
while (c[i]!='T'){
	switch (c[i]){
```

```
            case 'S': stop(); break;
            case 'L': left(); break;
            case 'R': right(); break;
            case 'F': forward(); break;
            case 'B': reverse(); break;
            case 'T': stop(); ext=1; break;
            default: stop(); break;
            }

            if(ext==1) break;
            ++i;
        }
}
```

REFERENCES

[1] Wikipedia. (2008). *Decca Navigator System* [online]. [Accessed 13 September 2008]. Available from World Wide Web: <http://en.wikipedia.org/wiki/Decca_Navigator_System>.

[2] Wikipedia. (2008). *LORAN* [online]. [Accessed 13 September 2008]. Available from World Wide Web: <http://en.wikipedia.org/wiki/LORAN>.

[3] Wikipedia. (2008). *Global Positioning System* [online]. [Accessed 13 September 2008]. Available from World Wide Web: <http://en.wikipedia.org/wiki/GPS>.

[4] Howstuffworks. (2008). *Global Positioning System* [online]. [Accessed 13 September 2008]. Available from World Wide Web: < http://adventure.howstuffworks.com/gps1.htm >

[5] Atmel. (2008). *High performance micro-controllers (89C51)* [Online]. [Accessed 13 September 2008]. Available from World Wide Web: < http://www.atmel.com/dyn/resources >

[6] Atmel. (2008). *8-bit Microcontroller with 4K Bytes Flash (89C51)* [Online]. [Accessed 16 October 2008]. Available from World Wide Web: < http://www.atmel.com/dyn/resources/prod_documents/doc0265.pdf >

[7] Mathworks. (2008). *Arcs and chords* [online]. [Accessed 18 November 2008]. Available from World Wide Web: < http://www.mathworks.com/ >

[8] G. Lachapelle, B. Townsend, H. Gehue, M. E. Cannon. (1993). GPS versus Loran-C for vehicular navigation in urban andmountainous areas. *Proceedings of the IEEE-IEE, Vehicle Navigation and Information Systems Conference, 1993. p456-459.*

[9] David Mark Bevly. (2001). High Speed, Dead Reckoning, And Towed Implement Control For Automatically Steered Farm Tractors Using GPS. *A dissertation submitted to the Department of Mechanical Engineering and the Committee on Graduate Studies of Stanford University.*

[10] J. Stephen and G. Lachapelle. (2001). Development and Testing of a GPS Augmented Multi-Sensor Vehicle Navigation System. *The Journal of Navigation. 54 (2), p297-319.*

[11] Salah Sukkarieh, Eduardo M. Nebot and Hugh F. Durrant-Whytte. (1999). A High Integrity IMU/GPS Navigation Loop for Autonomous Land Vehicle Applications. *IEEE transactions on Robotics and Automation. 15 (3), p572-578.*

[12] Chang-Sun Yoo, Lee-ki Ahn. (2003). Low cost GPS/INS sensor fusion system for UAV navigation. *Digital Avionics Systems Conference, 2003. DASC '03. The 22nd. 2 (3), p8.A.1 - 8.1-9.*

[13] S. H. Kwak, J. B. McKeon, J. R. Clynch, and R. B. McGhee. (1992). Incorporation of Global Positioning System into Autonomous Underwater Vehicle Navigation. *Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology, 1992. AUV '92. p291-297.*

[14] Qiuping Wu, Zhongyu Gao, and Yongliang Wang. (2002). Study on GPS/DR/MM integrated navigation system for vehicle based on DSP. *IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions. 2, p1591-1595.*

[15] David Hohman, Thomas Murdock, Edwin Westerfield, Thomas Hattox, and Thomas Kusterer. (2000). GPS roadside integrated precision positioning system. *Position Location and Navigation Symposium, IEEE 2000. p221-230.*

[16] Yongyi Zhao, Bo Song, and Jin Li. (2007). A Map Matching Algorithm in GPS-based Car Navigation System*. Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. 1, p77-80.*

[17] Youngsheng Wang, Xiangpeng Li, and Yong Huang. (1996). Navigation system of pilotless aircraft via GPS. *Aerospace and Electronic Systems Magazine, IEEE. 11 (8), p16-20.*