Navigator III

EEL5666 IMDL

Summer 2000

Charles C. Mullins III

Sponsored by: ChuckIII.com

Table of Contents

ABSTRACT

Navigator III is a fully autonomous self-navigating boat. The platform for this robot was built from scratch using wood and fiberglass. The boat was designed to take on the shape of a pontoon boat. The reason for using this design is because if it's great stability in all weather conditions. The boat is powered by two 30 pound thrust Minn Kota trolling motors. The boat was controlled using a 500Mhz Sony Vaio laptop, and a MSCC11 Single-Chip Controller board. Navigation was accomplished using a Rand Mcnally Street Finder GPS (Global Positioning System) unit. Navigator III can be used to aid scuba divers in their efforts to reach a predetermined GPS location. It can also be used as a rescues boat or surf buddy.

EXECUTIVE SUMMARY

Navigator III is a twin motor fully autonomous self-navigating boat. Through the use of a Rand Mcnally Street Finder GPS (Global Positioning System) unit, it will be able to self navigate. A 500 Mhz Sony Vaio laptop with Visual Basic is used to communicate with the GPS unit through the serial port. The GPS must also be connected to a ps2 port from where it can draw its power. Once turned on the GPS will begin trying to acquire a latitude and longitude. It immediately starts sending data back to the VB program where it is parsed for relevant data. Once the latitude and longitude of the current position is acquired the VB program will wait to be told where to go. Once a destination latitude and longitude are entered the program will turn the motors on and begin to navigate to the appropriate location. The motors are controlled by motor driver circuits and a MSCC11 Single-Chip Controller board. Visual Basic communicates with the MSCC11 through a second serial port and orders the manipulation of the motor driver circuits. The motors will be both turned on in order to achieve a strait course. To make a turn the motor on the side wishing to be turn towards will be turned off, or in reverse. When the motor is put in reverse the boat will make a much sharper turn then when just turned off.

INTORDUCTION

The motivation behind Navigator III is the desire to scuba dive a site with out having to swim to it. Using a robot to tow scuba divers to a location a few hundred yards away from shore would be of great value. As a scuba diver I have had to battle the surf and swim hundreds of yards to arrive at my destination. Many of the times only to find that I have swam off course. Using a GPS (Global Positioning System) unit would give the ultimate accuracy. Incorporating this into a boat, which was small and portable, would allow the every day diver to use it.

I started the design of the platform by looking for a stable boat. Normal V-haul boats have a tendency to flip in heavy surf. I decided to go with a pontoon style boat to achieve the greatest stability possible. The pontoon on either side of the boat made for a great location to store the batteries needed to power the motors. By having the batteries in the pontoons it would help to stabilize the boat and further prevent it from flipping in heavy surf.

To the best of my knowledge know IMDL student has been able to successfully incorporate GPS into their robot. For this reason I scoured the Internet for Information about affordable GPS units. I found three units that would attach to a laptop via serial port. These units ranged from $79 - $159. The reason I feel using GPS can be successful

in my case as appose to others who have tried is the fact that I do not need a high degree of accuracy. When dealing with a large body of water an error of 100 ft would be of little significance. Many students talk of installing GPS into a remote controlled car where they would need accuracy down into the 5-foot range, which would be nearly impossible. This report will begin with a general discussion of the integrated system used in Navigator III. It will then go on to discuss the platform used. Next we will look at actuation, sensors, and then the behaviors of the robot. Finally we will discuss the results, and future plans for the robot.

INTEGRATED SYSTEM

Navigator III uses a 500 Mhz Sony Vaio laptop for it's main processing power. The laptop connects to the GPS unit via a serial port. The GPS unit is powered with a PS2 port. The unit sends data as soon as it receives power. Using visual basic, the computer poles for data about once every sec. The computer then parses the data received from the GPS and determines the directions the motors will go. To control the motor the VB program will send a char via a serial port connection to the MSCC11 Single-Chip Controller board. A single char represents each direction the motors will be able to accomplish. The chars used are as follows:

1 = Forward, both motors forward

2 = Left Turn, left motor off right motor forward

3 = Right Turn, right motor off left motor forward

4 = Left turn in place, left motor reverse right motor forward

5 = Right turn in place, right motor reverse left motor forward

6 = Reverse, both motors reverse

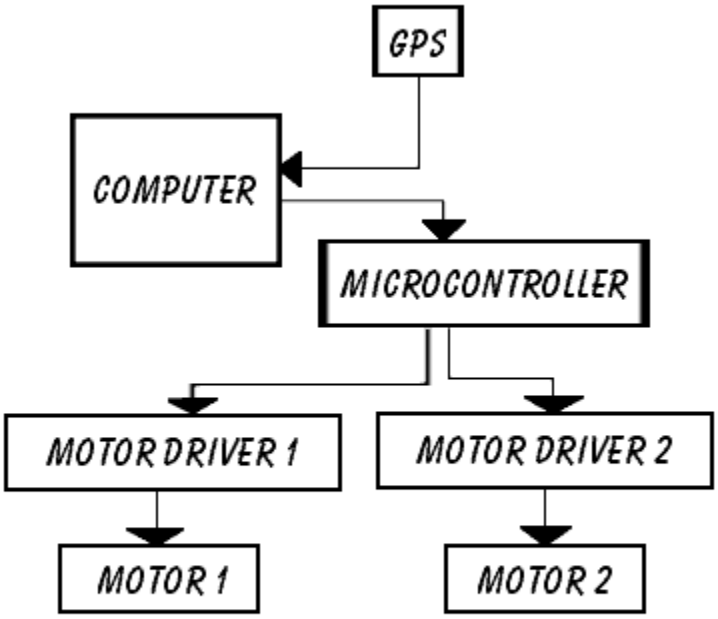
7 = Stop, both motors off

Anything else = Stop, both motors off

The MSCC11 Single-Chip Controller board using C interprets these motor commands. This program tells the MSCC11 to turn on and off the different ports needed to accomplish these tasks. An example command would be "2", a left turn. An if structure is used to determine which command was received, and then it would send 0x60 to PORTC.

What this says is to set pins C5 and C6 high. C5 tells the motor to turn on. C6 tells the motor to go forward or reverse. IN this case since C6 and C5 are high, the motor is on and goes forward.

Below is a simple block diagram of the interaction of the different components used.



MOBILE PLATFORM

The platform is a pontoon boat. This platform was chosen because it would allow for the greatest stability in a wide variety of conditions. With the batteries located in the center of each pontoon it provided a low center of gravity. There is a middle compartment between the two pontoons. This proved to be a great place to store the electronics. It kept the electronics as far away from the water as possible. A tongue and grove was used to seal the hull and the top of the boat.

This proved to be a great way of water proofing the boat from above. The hull was preliminarily constructed with wood, which made for a nice skeleton. Most of the seams were then filled with a spray liquid foam found at most hardware stores. This foam expanded and made for a great surface to shape the boat. Once the boat was shaped, fiberglass was laid and resin was added to make a waterproof hull. In the rear of the boat there were two wheels attached to make the boat easy to transport out of water. This turned out to be one of the best and worse ideas. It was a great idea because the boat

ended up weighing over 360lbs with all of the equipment added. It would have been nearly impossible to move with out them. The negative aspect was the fact that it was incredibly hard to make the hull waterproof with the wheels located up inside the rear of the hull.

SENSORS

The main sensor for this project was the GPS (Global Positioning System) unit. Three different GPS units were purchased. The first 2, one by Sony and the other by earthmate did not output data without first receiving some kind of data. Since I didn't know what data to input In order to receive an output, I didn't use these. These two units were $139 and $149 respectively. The third unit purchased was the Rand Mcnally Street Finder GPS. This unit was only $99 with a deluxe version of mapping software, and only $79 without mapping software. This unit output the following standards once power is provided to the unit.

GPGGA - Global Positioning System Fix Data

$GPGGA ,060616,2937.5223,N,08221.5158,W,1,06,0.91,36.1,M,-31.1,M,,*7A

TIME - GMT (UTC), 6 hours- 6 minutes- 16 seconds
LATTITUDE - 29 degrees and 37.5223 minutes
LATTITUDE - "N"orth or "S"outh
LONGITUDE - 82 degrees and 21.5158 minutes
LONGITUDE - "E"ast or "W"est
QUALITY - 0 = No fix, 1 = GPS fix, 2 = Differential fix
SATELLITES - 0-12 Acquired
HORIZONTAL DILUTION OF PERCISION - inaccuracy of acquired position, 0.91
ALTITUDE ABOVE SEA LEVEL - 36.1 meters
GEOIDAL SEPARATION - height of Geoid above the WGS_84 ellipsoid, in meters
DIFFERENTIAL UPDATE - Time in seconds since last differential update, Null when not used
DIFFERENTIAL REFERENCE STATION ID - from 0000-1023, Null when not used

GPRMC - Recommended minimum specific GPS/Transit data

$GPRMC ,060616,A,2937.5223,N,08221.5158,W,0.000,0.0,240700,4.4,W*79

TIME - GMT (UTC), 6 hours- 6 minutes- 16 seconds
DATA VALIDITY - "A" valid data, "V" unreliable data
LATTITUDE - 29 degrees and 37.5223 minutes
LATTITUDE - "N"orth or "S"outh
LONGITUDE - 82 degrees and 21.5158 minutes

LONGITUDE - "E"ast or "W"est
SPEED - 0 knots
COURSE - 000 degrees
DATE - 24th day 7th month of year 00
MAGNETIC VARIATION - 4.4 degrees between true and magnetic North
DIRECTION OF MAGNETIC VARIATION - "E"ast or "W"est

GPGSA - GPS DOP and active satellites

$GPGSA ,A,3,03,31,15,19,01,11,,,,,,,1.81,0.91,1.56*0C

DATA VALIDITY - "A" valid data, "null" unreliable data
3D - "2"D or "3"D navigation, 3D indicates altitude availability
SATELLITES - 12 fields, list satellites aquired
DILUTION OF PRECISION - 3 fields

GPGSV - Satellites in view

$GPGSV ,3,1,11,19,82,082,36,31,63,042,41,27,55,324,00,08,33,319,00*71

NUMBER OF GPSV RECORDS - 1-3
SEQUENCE NUMBER OF RECORD - 1-3
VISIBLE SATELLITES - number of satellites above the horizon, not necessarily acquired
  SATELLITE IDENTIFICATION NUMBER - 0-31
  ELEVATION OF SATELLITE - in degrees
  AZIMUTH OF SATELLITE - 0=North, 90=East, 180=South, 270=West, in degrees
  SIGNAL QUALITY - >30 is a usable signal, 60 is the best quality signal
  Repeats to Satellite ID Number - 4 satellites per record

PRWIZCH - Rockwell Zodiac Proprietary Channel Information

$PRWIZCH ,00,0,03,7,31,7,15,7,19,7,01,7,22,2,27,2,13,0,11,7,08,0,02,0*4C

SATELLITE IDENTIFICATION NUMBER - 0-31
SIGNAL QUALITY - 0 low quality - 7 high quality
Repeats 12 times

This unit gives an array powerful data. The standard I chose to incorporate was the $GPGGA. This standard as seen above gives TIME, Latitude, longitude, quality, number of satellites, horizontal dilution of precision, altitude, geoidal separation, differential update, and differential reference station id. Since this unit did not use differential GPS a few of the fields were ignored. The fields used for this project were Latitude, longitude, and number of satellites. Once 4 satellites are acquired the GPS will start to provide actual data. The accuracy of the data seemed to be about + - 0.0025 in fair conditions.

BEHAVIORS

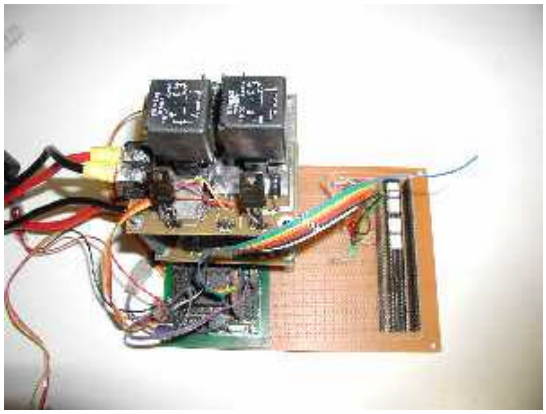
The robot was programmed to have 2 behaviors. The first one was to go to a predetermined location and return. This was accomplished quite easily with a simple program. If the boat had not yet reached its target within the specified accuracy it would continue moving closer and closer. Once the boat reached its desired coordinates it would turn around and return.

The second behavior was to zig zag between to set lines. The lines were either a latitude line or a longitude line. Once the boat at reached it's target line it would turn around and go to it's other target line. This would be done back and forth in a zigzag motion for an allotted time.

EXPERIMENTAL LAYOUT AND RESULTS

Each component was built separately and able to run independent of the other components. The first component was the GPS unit. It was connected via the serial port to the laptop and data was extrapolated via Visual Basic. The same Visual Basic program was able to independently give the MSCC11 command to control the motor. The MSCC11 was then able to receive commands and determine which motors needed to be turned on or off. The MSCC11 was also able to independently control the motors. These functions could then be incorporated to give a finished control system.

The results were not as desired. Due to the over utilization of long wires the control system became a cobweb. Once the wires were reduced and the boards were mounted together the control system became usable.



CONCLUSION AND FUTURE WORK

The conclusion reached for this project is that GPS is a great way of being able to control navigation when high accuracy is not required. With the evolution of GPS units it has become very affordable, and I for see a lot of individuals incorporating it into their projects. One thing to remember is that it is only good for large-scale devices. A remote control car using GPS to navigate around a room or tight course is an impractical expectation. GPS requires a direct line of sight and can be obstructed with trees and even sometimes clouds.

In the future the plan is to incorporate sonar obstacle avoidance. This was originally going to be implemented in this version of Navigator III but do to time constraints was scraped. Sonar will also be used as a beacon to follow scuba divers around while they are diving.

REFERENCES:

GPS NMEA info - http://sites.waldonet.net.mt/christian/GPS.html

GPS Unit Compatibility - http://www.gpsy.com/compatibility.html

NMEA-0183, common messages - http://homepages.ihug.co.nz/~jbaran/nmea0183.htm

Rand McNally GPS receiver with the Street Finder Deluxe 2000 - *Review by John Galvin.* - http://www.gpsnuts.com/myGPS/GPS/Hardware%20reviews/RandMcNally/rand_mcnally_gps.htm

Practical Visual Basic 6 - Bob Rselman & Richard Peasley, Que September 1999

Mekatronix - http://www.mekatronix.com/

APPENDIX

```
/*****************************************************
Motor Driver
Chuck Mullins
Version 2.0
*****************************************************/



#include <tjbase.h>


void main(void)
{
char temp;
  init_serial();
  DDRC = 0xff;
  PORTC = 0x00;
  while(1)
  {

    temp =  get_char();

 /*******************************************************
 Left forward = left motor PORT C 2
 Right forward = right motor PORT C 1
 Left reverse = left motor PORT C 4
 Right reverse = right motor PORT C 3

 1 = Forward
 2 Left Turn
 3 Right Turn
 4 Left turn in place
 5 Right turn in place
 6 Reverse
 7 Stop
 *******************************************************/

  if (temp == 49)
  {
  PORTC = 0x66;     /* Forward */
  }
  else if (temp == 50)
  {
  PORTC = 0x60;     /* Left Turn */
  }
  else if (temp == 51)
  {
  PORTC = 0x06;     /* Right Turn */
  }
  else if (temp == 52)
  {
  PORTC = 0x64;     /* Left Turn in place */
  }
  else if (temp == 53)
  {
  PORTC = 0x46;     /* Right Turn in place*/
  }
```

```
  else if (temp == 54)
  {
  PORTC = 0x44;     /* Reverse */
  }
  else if (temp == 55)
  {
  PORTC = 0x00;     /* Stop */
  }
  else
  {
  PORTC = 0x00;   /* Stop on Errors*/
  }

}
}
```

```
/*****************************************************
 Navigator Compass Driver
 Chuck Mullins
```

```
 Version 1.0
 ******************************************************/



#include <tjbase.h>


void main(void)
{
char temp;
int dir;
int num;

  init_analog();
  init_serial();

  DDRC = 0xff;

  // For Debuging With LEDs
  PORTC = 0xff;
  //


  while(1)
  {

     if (analog(0))
      {
        dir = 1;
      }
    if (analog(1))
      {
        dir = 2;
      }
    if (analog(4))
      {
        dir = 3;
      }
    if (analog(0))
      {
        dir = 4;
      }


      if (dir == 1)
      {
  write("North");
      }
      else if (dir == 2)
      {
      write("East");
      }
      else if (dir == 3)
      {
      write("South");
      }
```

```
      else if (dir == 4)
      {
      write("West");
      }
      put_char(13);
      put_char(10);
   temp =  get_char();
}}
```

```
/*****************************************************
 Visual Basic GPS and Motor driver
```

```
 Chuck Mullins
 Version 2.3
 ******************************************************/


Private Sub Command1_Click()
MSComm2.PortOpen = True
MSComm1.PortOpen = True
Label3.Caption = "GPS Connected"




End Sub

Private Sub Command2_Click()

MSComm1.PortOpen = False
End Sub


Private Sub Command3_Click()
Dim temp As String
Dim temp1 As String
Dim newtemp As String
Dim newtemp2 As String
Dim newtemp3 As String
Dim x As Integer
Dim y As Integer
Dim z As Integer
Dim n As Integer
Dim m As Integer
Dim l1 As Long
Dim l2 As Long
Dim l3 As Long
Dim l4 As Long

Rem new stuff
Rem
Dim lat As String
Dim lon As String
Dim sat As String

Dim lngResult1 As Long
Dim lngResult2 As Long
Dim lngResult3 As Long
Dim strMyString As String
Dim strFind As String
Dim lngStart As Long
Dim strlen As Long
Rem
Rem new stuff


n = 0
```

```
j = 1
y = 3500
strlen = 200

temp1 = ""
test = 0
Text4(n).Text = ""

Do
Do
temp = MSComm1.Input
Text4(n).Text = temp
Do
x = x + 1
Text3.Text = x
Dummy1 = DoEvents()
Loop Until x > y
x = 0
strMyString = temp
strFind = "$GPGGA"
lngResult1 = InStr(strMyString, strFind)
lngResult2 = 100
lngResults3 = lngResults2 - lngResults1 + 1
If Len(strMyString) > lngResults3 Then
 newtemp = Mid(strMyString, lngResult1, lngResult2)
End If
strMyString = newtemp
strFind = "$GPGSA"
lngResult1 = 15
lngResult2 = InStr(strMyString, strFind)
lngResult2 = lngResult2 - 26
If lngResult2 > 0 Then
newtemp = Mid(strMyString, lngResult1, lngResult2)
Text4(n).Text = newtemp

If Len(newtemp) > 23 Then

lngResult1 = 1
lngResult2 = 9
newtemp2 = newtemp
newtemp3 = newtemp
lat = Mid(newtemp, lngResult1, lngResult2)
Text1.Text = lat
l1 = 1
l2 = 9
newtemp2 = Right(newtemp2, 32)
lon = Mid(newtemp2, l1, l2)
Text2.Text = lon
newtemp2 = Right(newtemp2, 17)
sat = Mid(newtemp2, 1, 1)
Label5.Caption = sat & " Satellites Acquired"
Label4.Caption = "Satellites Acquired"
Shape1.FillColor = &HFF00&
Shape1.Visible = True
Else
Label4.Caption = "Acquiring Satellites"
Label5.Caption = "0 Satellites Acquired"
```

```
Shape1.FillColor = &HFF&
Shape1.Visible = True
End If
Dummy1 = DoEvents()

n = n + 1
End If
Loop While n < 10
n = 0
Loop While m < 10



End Sub

Private Sub Command4_Click()
Dim tjtemp As String
Dim v As Long


tjtemp = Text5.Text
Text5.Text = ""
Rem MSComm2.Output = tjtemp

End Sub

Private Sub Command5_Click()
Dim GoLat As Double
Dim GoLong As Double
Dim CLat As Double
Dim CLong As Double
Dim TLat As Double
Dim TLong As Double
Dim Dir As Double
Dim LatDev As Double
Dim LongDev As Double
Dim NLatDev As Double
Dim NLongDev As Double
Dim CDir As Double
Dim NDir As Double
Dim CC As String
Dim C As String



LatDev = 0.001
LongDev = 0.001
NLatDev = 0 - LatDev
NLongDev = 0 - LongDev
Rem CDir = Current Direction
CDir = 1

GoLat = Text6.Text
GoLong = Text7.Text
CLat = Text1.Text
CLong = Text2.Text
```

```
TLat = GoLat - CLat
TLong = GoLong - CLong
Text8.Text = TLat
Text9.Text = TLong

Rem Direction
Rem 0 = At Location 0 0
Rem 1 = North   + 0
Rem 3 = East    0 -
Rem 5 = South   - 0
Rem 7 = West    0 +
Rem 2 = North-East  + -
Rem 4 = South-East  - -
Rem 6 = South-West  - +
Rem 8 = North-West  + +

If (TLat > LatDev) Then
    If (TLong > LongDev) Then
      Dir = 8
      Text10.Text = "8"
    ElseIf (TLong < NLongDev) Then
      Dir = 2
      Text10.Text = "2"
    Else
      Dir = 1
      Text10.Text = "1"
     End If
ElseIf (TLat < NLatDev) Then
    If (TLong > LongDev) Then
      Dir = 6
      Text10.Text = "6"
    ElseIf (TLong < NLongDev) Then
      Dir = 4
      Text10.Text = "4"
    Else
      Dir = 5
      Text10.Text = "5"
    End If
Else
     If (TLong > LongDev) Then
      Dir = 7
      Text10.Text = "7"
    ElseIf (TLong < NLongDev) Then
      Dir = 3
      Text10.Text = "3"
    Else
      Dir = 0
      Text10.Text = "0"
    End If
End If
Rem 1 = Forward
Rem 2 = Left turn
Rem 3 = Right turn
Rem 4 = Left turn in place
Rem 5 = Right turn in place
Rem 6 = Reverse
Rem 7 = Stop
```

```
NDir = Dir - CDir
Rem CC = Counter Clock Wise
CC = "2"
Rem C = Clock Wise
C = "3"
If (Dir <> 0) Then
If (NDir > 0) Then
  If (NDir > 4) Then
    MSComm2.Output = CC
    Text5.Text = "++"
  Else
    MSComm2.Output = C
    Text5.Text = "+-"
  End If
ElseIf (NDir < 0) Then
  If (NDir > -4) Then
    MSComm2.Output = CC
    Text5.Text = "-+"
  Else
    MSComm2.Output = C
    Text5.Text = "--"
   End If
Else
    MSComm2.Output = "1"
    Text5.Text = "00"
End If
Else
MSComm2.Output = "7"
End If
Dummy1 = DoEvents()
End Sub
```