GPS controlled navigation Bot

(A prototype for a smart wheelchair)

Amit Mehta (06D07020)

Prashant Saxena (06D07021)

Aditya Padmawar (06D07023)

Supervisor: L. R. S.

Course Instructor: Vivek Agarwal

Introduction:

The aim of our project is to build a prototype for the concept of a smart wheelchair. The prototype is a small bot capable of navigating to a destination given by the user, based on calculations it makes using the source and destination GPS coordinates.

Design Approach:

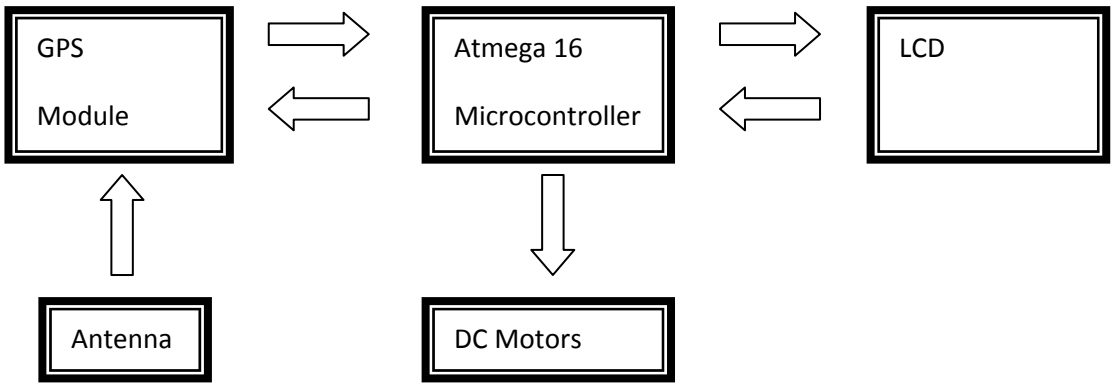
The size of the bot is about 12 cm x 18 cm (without wheels). It is driven by 2 motors with an rpm of 100m each. The bot is mounted with a GPS module, microcontroller and an LCD. The GPS module receives the GPS coordinates of the source and sends it to the microcontroller. The microcontroller is an Atmega 16L chip. It makes some calculations based on the GPS coordinates of the source and the destination (provided by the user) and finally calculates the total distance to be moved and at what angle to the north direction should it move. This is communicated to the DC motors via the driver ICs and the bot moves towards the destination coordinates.

Project stages:

- Receiving the desired data from GPS module and interfacing the microcontroller with the GPS module.
- Building a mechanical system for the prototype of wheelchair.
- Displaying the GPS module's data strings on a text LCD.
- Parsing the NMEA sentences and computing the distance and angle.
- Interfacing microcontroller with motor driver ICs and moving the bot as per the need.
- PCB designing & Combining all the stages into a final product.

Circuit Design:

1. **Hardware Aspect:**



1.1 GPS Module:

The GPS module in use is an iWave product using a Sirf Star III chipset. It is capable of operating over a small range of supply voltages: 3.1 to 3.6 volts. The GPS module on switching on tries to get a fix with at least 3 satellites which takes about 20 – 40 seconds. after which it starts showing NMEA protocol strings which are standard for all GPS units. Out of several NMEA string formats, this module shows GGA, GSA, GSV, RMC, GLL, VTG. It updates its data (all these strings) at rate of 1 Hz.

These strings provide different data like satellites in view, date and time besides the geographic location. We select the GPRMC sentence from the data which is the "Recommended Minimum" sentence. It looks somewhat like this:

$GPRMC,204606.000,A,1908.0704,N,07254.2662,E,0.9,307.78,250409,,,*67

All these strings are sent to the microcontroller via serial - USART communication at a baud rate of 4800. The transmission pin of GPS module is connected to the D0 (Rx pin of microcontroller).

The precision in co-ordinates of up to 2-3 meters is essential for this project and this module has an accuracy of that level in open space.

1.2 Microcontroller

The microcontroller used by us is the Atmega 16L. It is capable of operating in a wide range of supply voltage: 2.7 to 5.5 volts.

The atmega receives the NMEA sentences by calling the function USART_Receive(). Then, according to the logic fed in the microcontroller, it parses the string provided and extracts Latitude and Longitude. The value of Latitude and Longitude is displayed on a text LCD. Using initial values of Latitude and Longitude required calculations are performed to calculate distance and angle with the north.

Another reason for choosing ATMEGA16L was that if we need to give any signal to the GPS via transmission pin of microcontroller, the voltage level is 5V in ATMEGA 16, while it is 3.3 V in ATMEGA16L, which is below maximum voltage that can be tolerated at any of the GPS pins.

1.3 Motors:

For this part of the project we are using simple DC motors with 100 rpm. A low rpm value is required in order to have sufficient value of torque to carry the whole assembly and battery weight.

We have used L298 motor driver IC for running the motors. It needs a logic power supply of 5V and supply voltage of 12V (which is given by the lead-acid battery). The input pins of the motor driver are connected through Port A of microcontroller.

In the initial part of the project we have tested stepper motors for high precision in distance moved by the bot and angle by which we can rotate the bot in desired direction. We have tested these motors on A3967 as well as L298 motor driver ICs. However due to results varying with different power sources we had to revert back to using DC motors. We will incorporate higher precision handling with stepper motors in the later part of the project.

Schematic:

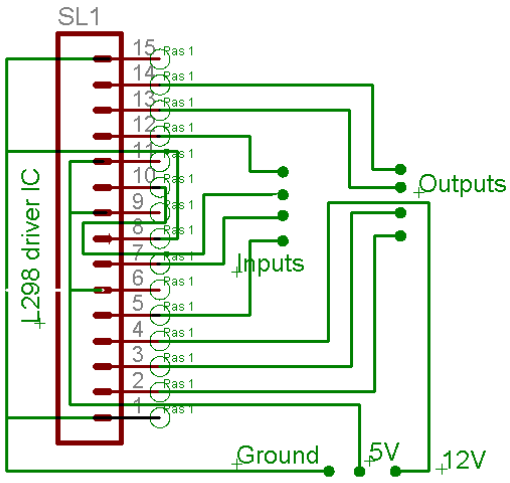


Figure1: Motor driver (L298) circuit

1.4 LCD:

We are using a JHD162A 16x2 text LCD for the display of source - destination co-ordinates, distances other messages. The LCD is connected to the port C of microcontroller. 3 pins(C0, C1, C2) are used for control signals and 4 pins (C4, C5, C6, C7) are used as data pins.

We have tried using the text LCD at port D (leaving pin D0 which is used for UART communication with GPS). But since for using LCD we have to declare this port as output port, as result of which we can no longer use the special function of D0 pin (Receiver for UART). Hence LCD has to be used at port D.

Schematic:

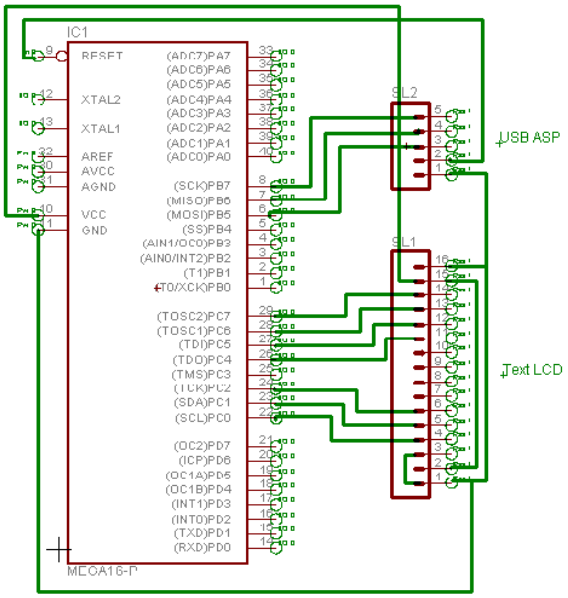


Figure 2: LCD and USB ASP programmer circuit

1.5 USB-ASP:

For programming the microcontroller we are using standard USB-ASP programmer which is a ISP (In-System Programmer).

1.6 GPS antenna:

A standard GPS antenna AA105 (make toaglas) has been used for this application. The gain provided by the antenna is well within the range of what is required by the GPS (28+/- 2 dB).

1.7 Power supply:

We are using a 12V lead-acid battery. The power consumption mainly lies in driving the motors. Power consumption by other units is:

GPS <111.5mW

LCD <16.5mW

Outlook of the final PCB:

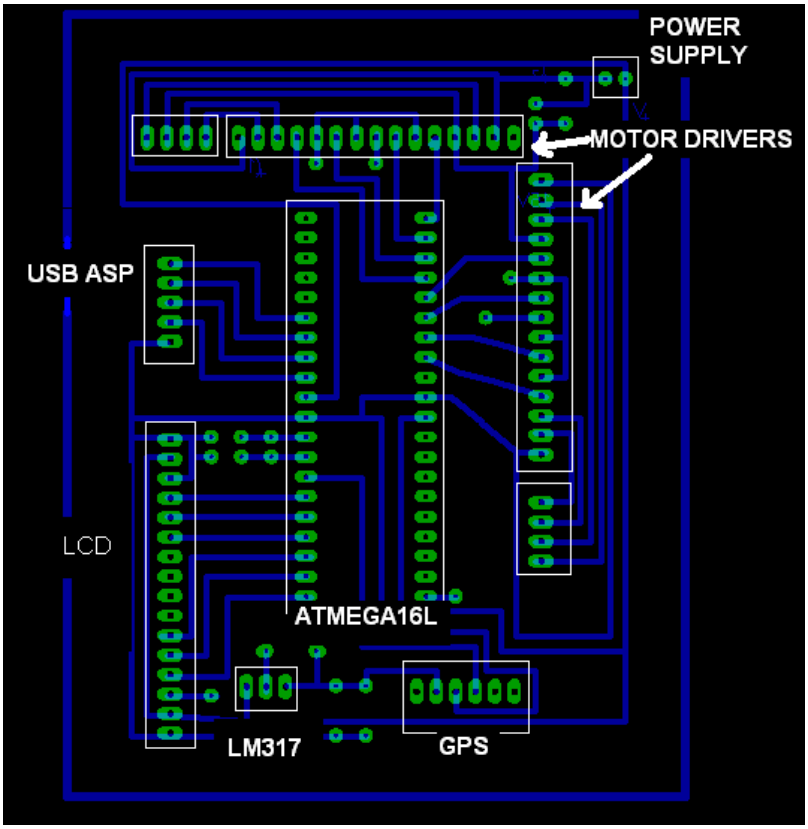


Figure 3: All components connected to microcontroller

2. Software Aspect:

2.1 Parsing of GPS Data

Once the USART_Receive function is called it takes in the serial port values bit by bit. We use a for loop to store the first 500 characters at each call of this function and store these into an array. Over this array we perform a check whether a $GPRMC has come inside this array somewhere.

$GPRMC,204606.000,A,1908.0704,N,07254.2662,E,0.9,307.78,250409,,,*67

Upon encountering the characters RMC in sequence, we then parse the full sentence, taking out values of Latitude, Longitude and velocity. These values are taken out depending on the no. of commas we encounter after RMC because these values are generally enclosed within 2 commas.

What we get for latitude is a string 1908.0704N. This string is converted into a double 1908.0704 by the C function atof( ). The GPS coordinates are then converted into degree decimal form so as to calculate distance and angle.

2.2 Calculation of distance and angle:

To calculate the distance between points on earth with given latitude and longitude, people generally use a theorem called The Great Circle Theorem. However, in our

particular case, the distances involved would rarely exceed 3 km. So we use an approximation of the spherical geometry and trigonometry calculation by the Great Circle Theorem, known as Flat Earth Approximation. This approximation is valid for distances less than 10 miles and the fractional error by this is proportional to $\left(\frac{d}{R}\right)^2$

Using this approximation, we can convert points on the spherical earth onto an X-Y coordinate system with reasonable accuracy of distances:

Let source coordinates received from GPS unit be slt and slg for latitude and longitude respectively. Similarly, coordinates of destination are dlt and dlg.

Now, the NMEA format has a peculiar format of showing the coordinate value. It is actually 100*degrees + minutes. So, we need to convert these coordinates into decimal degree form like:

Coordinate received from GPS: $1908.0658 = 19^0 + (08.0658)' = 19.134433^0$

So, we convert all these coordinates into this format and name them sltdeg, slgdeg, dltdeg and dlgdeg.

To calculate the x and y coordinate distances between these points using the flat earth approximation, we use the formulae:

X distance: $R*\pi/180*(dlgdeg-slgdeg)*\cos(sltdeg)$ (1)

Y distance: $R*\pi/180*(dltdeg-slgdeg)$ (2)

where R = Radius of earth = 6378.1 km

So, the total distance can be calculated by the simple formula:

$D^2 = X^2 + Y^2$ (3)

Also the bearing of the destination from source point (angle subtended with the North) can be calculated using:

$B = \mathrm{mod}(\mathrm{atan2}((x,y),2\pi))$ (4)

3. Test Procedure:

For the test we are assuming there are no obstacles in the path of the bot and the path from source to destination is a linear one.

We require the co-ordinates of latitude and longitude of destination. This we are providing with hard coding (we will program the microcontroller before the test run to enter co-ordinates of our choice). The LCD will display the co-ordinates of source, destination, distance to be travelled and angle of rotation, etc.

The bot has to be oriented towards North initially. The microcontroller will calculate the angle by which it will rotate on the spot. Then the bot will move in a straight direction towards the destination until it reaches within its range of 2-3 meters.

Test Results:

1) For our room in hostel 13, co-ordinates received from GPS:

   1908.0666N, 7254.2729E

   co-ordinates received from google earth:

   1908.0658N, 7254.2760E

This signifies sufficient amount of accuracy in presence of a clear signal.

2) See Appendix 1 for the GPS strings received on the hyperterminal of a computer. The image shows typical data strings sent by GPS in standard NMEA format.

4. Conclusion and further improvement:

Major part of our initial goal has been achieved. The system is able to detect its own co-ordinates and based upon desired destination we are able to move in that direction with sufficient accuracy. We conclude this kind of system is feasible and implementable in the given environment and can show improved performances with some modifications.

However there are some issues which couldn't be resolved till the end. We have to set the system facing North initially since it can't detect its initial orientation. This problem can be solved with use of a digital/electronic compass. But its price is too high to be included in this project. Some more changes are needed to be done in order to convert this into a product.

As a part of our EDL-II project we are planning to add features like obstacle detection, accurate menuvere and a LCD screen showing current location of the user on a map.

Since a smart wheelchair's initial purpose was to simplify transportation in a familiar & known environment, like day-to-day work inside a college campus. We will be having a complete map of the paths which user requires a set of destination (co-ordinates of which are stored inside system already). Then the user will simply have to select the

destination number like “#5 - Main Building” or “#7 – Main Gate”, and the wheelchair should be able to transport the user to that location. In order to proceed in this direction we can also work on the node-based algorithm for movement of bot on a non-linear trajectory.

References

1. *www.atmel.com/dyn/resources/prod_documents/doc2466.pdf*
2. www.fischl.de/**usbasp**/
3. www.**datasheet**catalog.com/**datasheets**_pdf/L/2/9/8/**L298**.shtml
4. www.net**wavegps**.com/
5. www.wikipedia.org

Appendix

6. Data strings given by GPS on hyper-term