GPS NAVIGATION SYSTEM

GROUP D5
Apoorv Tiwari (05D07001)
Chetankumar Phulpagare(05D07013)
Kedar Chandrayan(05D07030)

EDL Stage II
Date: 29th November 2008

DESCRIPTION

The objective of our project is to build a GPS navigation system that can be used for simple consumer use. Over the previous semester we built a system which consists of a GPS module, a display element (color LCD), a storage element (SD card), a USB interface to communicate with a computer and a user interface. As the user moves from place to place the corresponding location is displayed on a map on the lcd. The map has been pre-downloaded into the system by the user. Also a travel log is kept which can be accessed later. New Features that have been added to the device are:

1. The device besides showing the path taken by the user on the LCD screen can also navigate for the user, by asking for a destination and pointing the way
2. It would save the travel log on the SD card plugged into the device.
3. The user can download the map of the area on the SD card through the USB interface provided. Also the user can zoom in and out of the map making the device easier to use.
4. A keypad has been added to the device for user convenience.
5. Packaging of the device into a convenient size on a single PCB

SPECIFICATIONS

Hardware

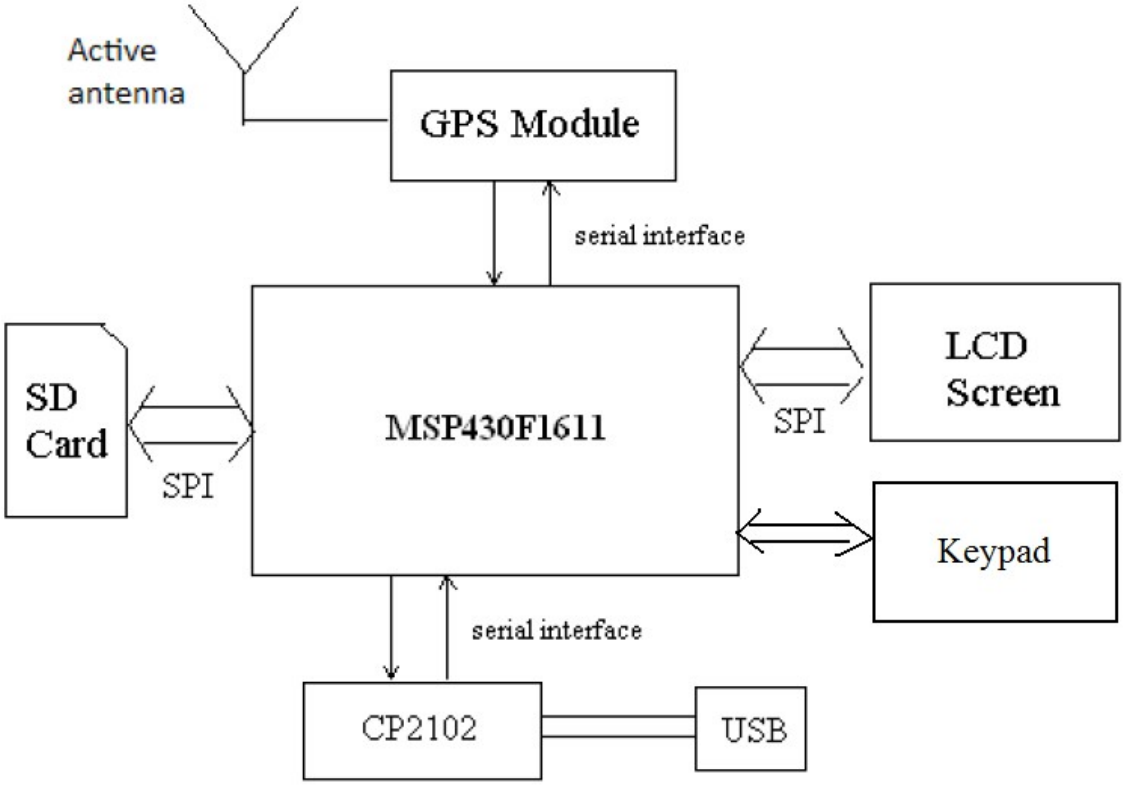
- GPS Module: ZX4125 with active antenna
- Microprocessor: MSP430F1611 (2 UART with SPI mode)
- Sandisk SD card (2 GB)
- USB to UART bridge controller: CP2102
- Graphic LCD screen: Nokia 6610

Software

- Reading serial data from the GPS module and storing the log in the SD card.
- Reading the map from the SD card and displaying it on the LCD.

- Plotting the points using the coordinates received from the GPS.
- Identifying the current location and classifying it as a “node”.
- Navigation from one node to another.
- Downloading map via USB interface and saving it on SD card.
- Responding to commands issued by the user through the keypad.
- Zoom in/zoom out , scroll functions for the map.

Block Diagram:



Progress Plan:

- **Identifying and working on a new LCD:** Since for a user friendly navigational aid, the display element (lcd screen) should be large enough to provide a convenient user experience. We felt that our existing LCD(**Nokia 6610**) is not large enough and started looking for suitable alternatives. We had looked at the **Nokia 9200** and the **Nokia E61** LCD last semester and had also considered using a **Nokia 6600** LCD, however that was not possible due to lack of availability of proper documentation for these LCDs. (one reason why we worked with Nokia 6610 lcd was that its information was properly documented and was easily available)This semester, we decided to use a **Sparkfun**

LCD 08335 and connector. But lack of a suitable controller meant that detailed pixel by pixel colour coding and other driver options needed to be written which would have needed a lot more time than 1 semester. Thus we decided to stay with the nokia 6610 lcd and improve other features of the device.

- **Choosing our microcontroller:** Our hardware needs a controller that has at least 2 USARTs. Since we decide to use another LCD we needed more processing power to make the higher resolution (and also requiring more computing resources) run at a decent speed. Thus we decided to get another faster microcontroller preferring an ARM controller. We decided to go for a **SAM 9 Series** processor. However since the new LCD was not employed and the older one worked sufficiently well with our existing controller, we decided to continue with the **MSP430F1611** chip. The microcontroller is run at **8MHz** clock speed.

- **Other Hardware Elements:** We did not need much change in the other hardware that our device has. The external USB controller was the **CP2102** chip by **Silabs** which had an internal EPROM as well as an **inbuilt voltage regulator** to independently act as a **USB to UART Bridge Controller**. Interfacing the USB chip with the MSP was through a simple UART serial interface with a baud rate of 9600.We also designed a separate PCB for the CP2102 chip based on the circuit obtained from its datasheet. The chip drivers for windows (**Virtual COM Port driver**) was obtained from the Silabs website.
  The memory element used was an **SD Card**. A 2 GB card was chosen as in there are some compatibility issues between systems using SD cards of capacity 1 GB and below and those using 2 GB and above. So as we wanted an option to expand our memory capacity at a later date, we selected a 2 GB SD card. The SD card has been interfaced with the MSP using an **SPI interface** using a power supply of 3.3v.
  A **Keypad** was added to make user interfacing easier(earlier user instructions were entered through the comp.)

- **Adding Software Features:** To make our device more user friendly newer features needed to be added. Features added were:
  - A navigation software: It calculates its present position/asks the user for the source position and describes the path to the destination.
  - Map features: To somewhat compensate for the lack of a large enough lcd,zoom in zoom out features were implemented. Also a scroll feature was implemented.

- A primitive was implemented. We tried to implement a FAT16 file system but it did not work properly. So we settled for a less complex system.

- **Designing External Dimensions for the Device:** We approached a person working for a design firm and got a rough design for the whole product keeping in mind all the dimensions and components and their relative location to each other. The design tried to keep in mind the aesthetics of the product while also ensuring ease of use and proper functional coherence for the different components in the device.

- **Designing a Final PCB:** The final PCB was then designed keeping in mind the external design already made and the physical limitations of the components and the device itself. This PCB was then made and tested.

Detailed Descriptions Along with Testing Procedures:

Hardware Testing

UART Testing:

- MSP was programmed using JTAG programmer through parallel port.

- MSP UART were tested using sample codes provided on TI website.

- Then MSP <-> GPS interface was tested by using an **LED** as indicator. The LED was made to **toggle** when "$GPRMC..." messages were received.

- Once CP2102 PCB was made, its on-chip voltage regulator was used to power MSP430 and other elements in the circuit except GPS module.

- Before connecting the CP2102 PCB to desktop, USB drivers should be installed so that Virtual COM port is created when we connect CP2102 using USB cable.

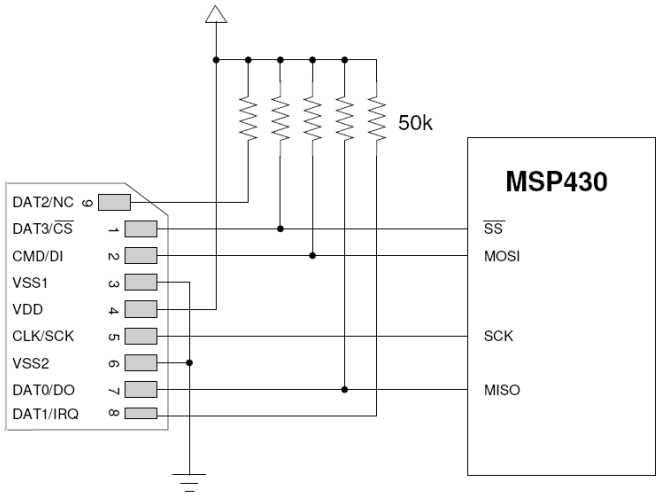
- For testing the USB side of CP2102 we short the Rx Tx pin and check for echo of character in hyper terminal (or minicom in Linux).

- For testing the UART side of CP2102 we sent a char via USB (hyper terminal) to MSP, then incremented it and then sent it to the desktop.

- Then we developed s/w for the interface: Desktop ←USB→ CP2102 ← UART → MSP430. Then we developed the interface: MSP430 ← UART → GPS Module

- We also tested simultaneously the above two interfaces on different UARTs of MSP, thus were able to see the GPS coordinates on comp hyper-terminal screen through USB interface.

SD Card testing:

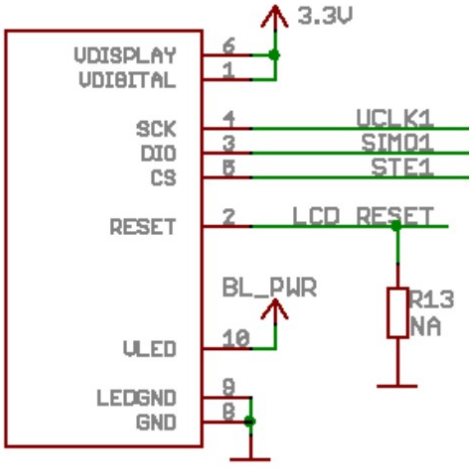


C library for interfacing SD card with MSP430 was available on a link given in references. It had functions for initialization of SD card, block-read and block-write. We tested these functions and verified using Card Reader on Laptop. For verifying that a proper read/ write has taken place we used the “dd” command available in Linux.
Synopsis of dd command:

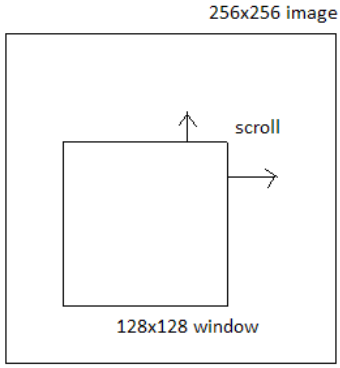
$ dd bs=512 count=num_of_blks seek=output_offset skip=input_offset if=input_file_path of=output_file_path
e.g.
$ dd bs=512 count=1 skip=512 if=/dev/mmcblk0 of=/tmp/mmc

We also tried different implementations of FAT16 file system on SD card but none worked.

LCD testing:

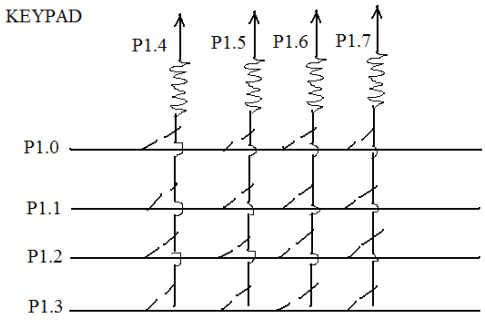


We got a sample code for the LCD interfacing with MSP430 family controller from olimex.com.
Sample code worked as expected. We played with the menu pointer defined in the code. We changed the shape and colour of the pointer to a rectangle.
Different values for LCD commands were tested such as “ACCESSCTRL” that changes orientation of axis on LCD. Then various colour patterns were tried.
Different images including IITB map (132x132) were shown on LCD. Image raw data was stored on SD card and converted to required colour format for LCD using MSP.
Since LCD is small, roads on map could not be seen clearly. So a higher resolution map was to be shown. Same map with higher resolution (256x256) was used. Only a part of this map that can be seen through a 128x128 window was displayed on LCD and this window could be scrolled in all four directions to view the complete map.
This enables us to see any high resolution map on same LCD.



We also developed functions to show ASCII chars with desired foreground and background colour on LCD.

Keypad Testing:



Originally we started with a 4X4 keypad which had a layout like this:

E00E	D00E	B00E	700E
0	4	8	C
E00D	D00D	B00D	700D
1	5	9	D
E00B	D00B	B00B	700B
2	6	A	E
E007	D007	B007	7007
3	7	B	F

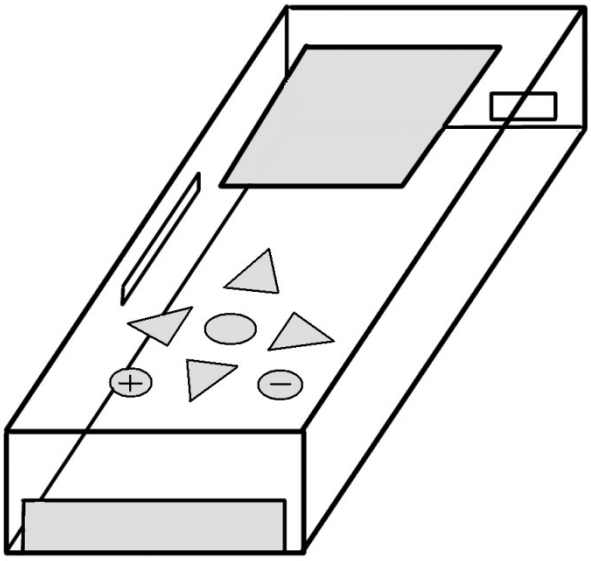
After this, since we did not need 16 buttons, it was reduced to 9 buttons as below:

```
// |===|===|===|
// |  0  |  U  |  1  |
// |===|===|===|
// |  L  | CR | R  |
// |===|===|===|
// |  +  |  D  | -  |
// |===|===|===|
```

External Package Design:

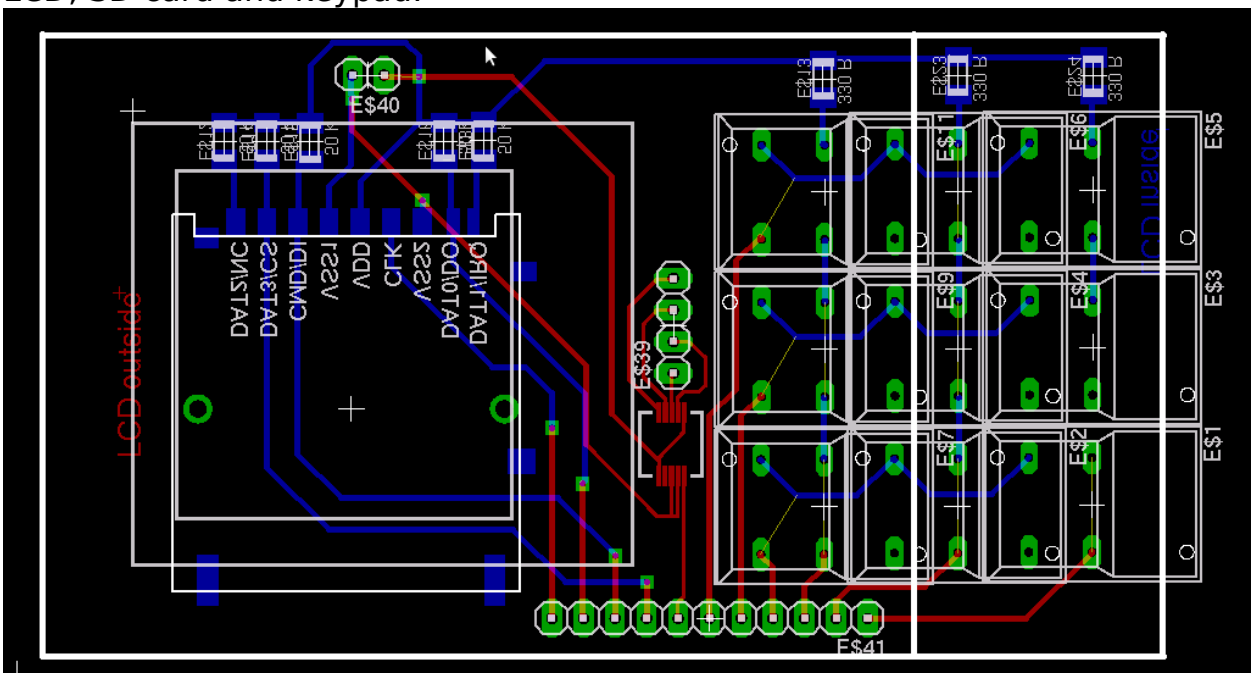
Before designing the final PCB, we first decided the external casing specifications and then designed the PCB in order to meet them. The design for the external package is:

Body dimensions: 90x50x30
LCD window dim: 30x30
USB 'A' type female opening: 8x14
SD Card Slot: 28x3
Antenna connector opening: circular diameter = 9
Battery: 35x10

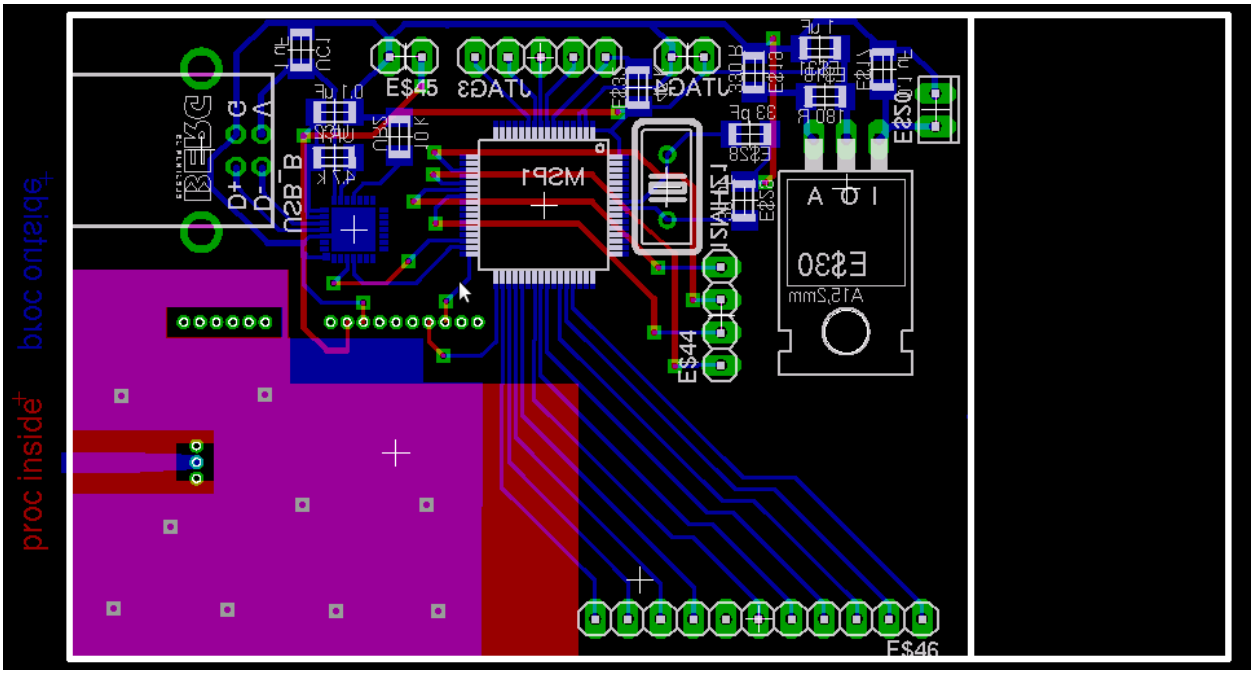


Final PCB Design:
The final design consists of 2 double layered PCBs held back to back. First board has LCD and keypad laid out in a fashion we should have on a mobile handheld device. Other board carries the processor and other required components. SD card can be inserted in or out sidewise into the device. The PCB sizes are made according to the external casing size.

LCD, SD card and keypad:



Processor and other components



Software

Navigation:
This is one of the most important aspects of this device and is the one feature that makes it the most useful. The whole map is divided into a **series of nodes** showing important locations(likes hostels departments etc.). The user inputs his present location (or its read directly from the map by the device) and enters the location he wants to travel to. The device provides instructions in terms of which node to travel to next until the destination is reached.

To calculate which node to travel to next in order to obtain the shortest possible path, we use distributed graphs. Each **sub-graph** is either a **line** or a **loop**(all sub-graphs are so chosen). To reach a node from another one in a **line** graph, there can be only **one direction** of travel. To reach one node from another in a **loop** graph, there are 2 possible paths, one in either direction. (eg the hostels and sac form a loop around the main playing field). This direction is decided by the cost functions taken according to the distances between adjacent nodes. Thus within a sub-graph the optimized paths are calculated.

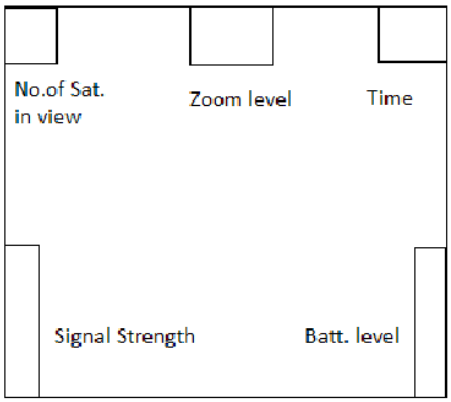
To go from 1 sub-graph to another, the person must go through some common nodes called **key nodes.** Thus, to optimize travel over the whole graph, optimization is done in each sub-graph and then over key nodes and the whole graph.

Some of the defined nodes are:
Hostels(1 to 13)
Departments
Main Building
Convocation Hall
Convocation Hall Turn
Library
Library Turn
Swimming Pool
BJC Turn
SOM
Canara Bank
Kresit
Main Gate
YP Gate

LCD Functions:
Zoom In/Out: The user can vary the detail level of any part of the map at will. Zoom in Zoom out have been implemented using the simplest method possible.
We have stored images of all zoom levels in SD card and read the appropriate image accordingly. This enables for a richer user experience and a convenient easy to use product. The current zoom levels of the IIT Bombay map are:
I. 256x256
II. 512x512
III.768x768
IV.1024x1024
The zoom level is visible on the screen.

Scroll: The **scroll function** (up, down, right, left) has also been implemented. The user can go to any part of the map he wants to.

Screen Layout:



File System
File System Details are as follows:
block 1:
1. no of files [2 bytes]
2. entry corresponding to each file
    a. File ID [4 bytes]
    b. File name [max 12 chars]
    c. Starting block of file. [4 bytes]
    d. File Size [4 bytes]

Files now present:
1. font8x16.img [file ID=1]
2. Menu.img [file ID=2]
3. iitb.map [file ID=3]

Starting block of the file may contain detailed information about the file, depending on the file type or just the data in the file. file ID = 0 implies the file does not exists. File entries are stored in this table in increasing order of file IDs. When a file is to be deleted, just the file ID for this file is made 0. The file iitb.map stores maps of resolutions 128, 256, 512 and 1024.

File System Functions:
Open File: locates the file ID and reads the data
Close File: Close the file
Delete File: Deletes the data as well as the file ID

Reading Data from USB to the SD Card using Matlab:
Since CP2102 is usb-to-serial bridge, it's virtual serial port could be accessed through matlab. Since it is very easy to do image manipulations in matlab, it was possible to take any image, convert it to format suitable for our purpose and then download it on the device.

Purpose of the usb interface:
1. to download map on the sd card from desktop.

2. to view the log files that are stored in the sd card

Handshakes:
First of all the user will press a button on the keypad which will tell the device to enter in COM mode. After entering into COM mode it will poll for a command from the desktop.
There will be some standard set of commands to start some particular form of data transfer. I will number them as C0, C1, C2, etc.
Notation: Desktop (D) <USB> MSP (M) <SPI> SD

Gather information about the SD data:
First D sends C0 to M, which is in COM mode. M gathers the information about the SD card memory allocation and passes this info to D. I think maintaining the current memory info (that is which file in the SD card is stored where) can be loaded at the SD_init itself and then we can update this info in the RAM of M. The format of data which will be transferred to D will as follows:
numOfFiles'$'filename1'$'startBlkAdd1'$'EndBlkAdd1'$filename2.....EOF
numOfFiles: 2 bytes
filename: 16 bytes
startBlkAdd: some bytes
'$' is our special character which will not be allowed in the file name. This will be standard procedure: M enters in COM mode--> D sends C0 to M and M responds by giving the memory info to D in the format described above. After receiving EOF D stops reception of data, M again enters in the COM mode.

Write data to SD:
For this we implement a command C1. D sends C1 to M which was in COM mode polling for commands. After this D sends the start block addr and the number of blocks to be written. M then polls for these many number of blocks to recieve data and do blockwise write to SD. After tx of each block the matlab code on D should wait for some time. In the meantime, M writes the block to the proper block address.

Read from SD:
For this we will implement a command C2. D sends C2 to M which was in COM mode polling for commands. After this D (matlab code) enters in polling state expecting the number of blocks first and then data.

Update SD info:
For this we will implement a command C3. D sends C3 to M which was in COM mode polling for commands. After this M expects the update due to the last write which is written by M to the first block of the SD. An update is necessary after every C1.

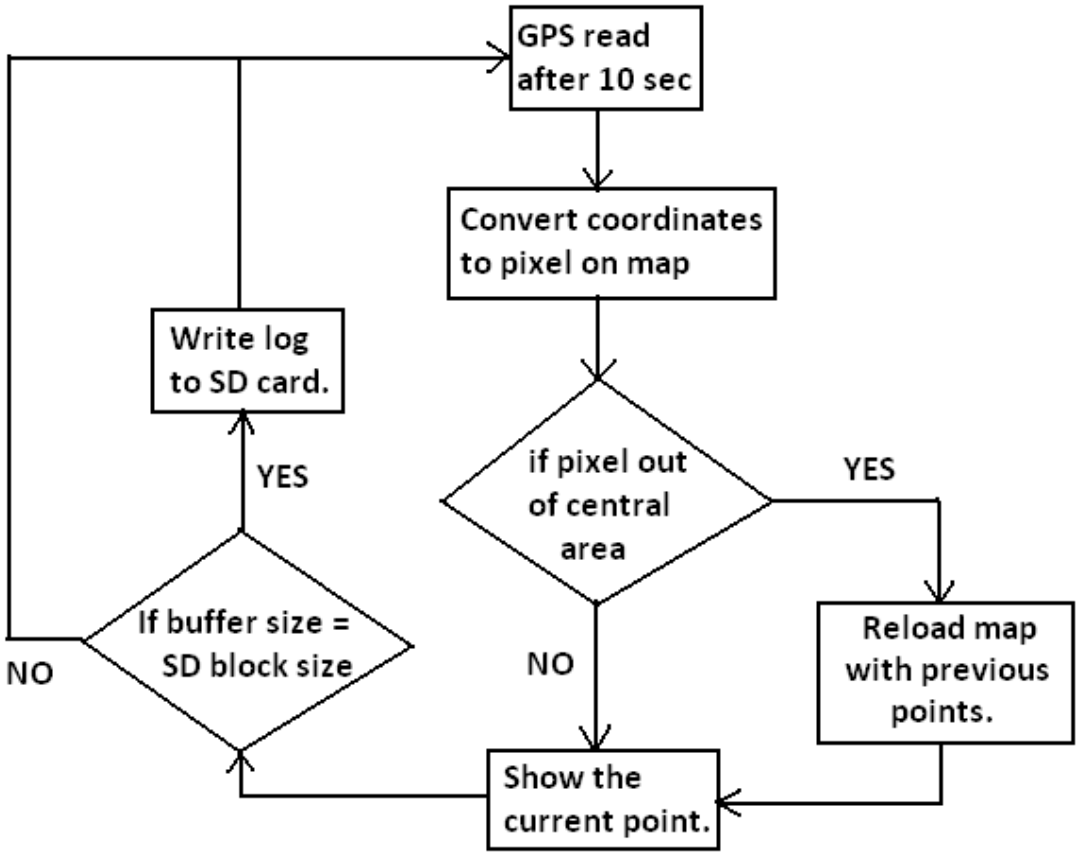
Having done this, we can look at SD card as if directly connected to D and as if M doesn't exist in between. -- ie. prepare a capsule matlab code!

DOWNLOAD of map:
First send C0. find where the sufficient amount of memory is available, then issue C1 then provide the necessary protocol info and the map data and then C3 and then necessary protocol data.

READ LOG file:
simple enough. just C2

Assumptions:
1. All our files are of integer number of block length.
2. File names will have proper extension to specify that it is a map file or a log file etc.

To display the location of the person on the map, the following algorithm was worked out:



Conclusion:

We have tried to build on the work done in the previous semester and make our navigator into a more complete product. Functional though as it is, the product suffers from a drawback: the LCD is too small to make a good commercial product. Although this device can be marketed as a low cost (about Rs. 3500 – 4000) solution, it needs to have a better LCD and a faster processor to be a huge commercial success. To add further value to this, other features like voice instructions and colour coded LEDs can be used to enhance user experience. With a large number of calibrated maps along with these added features, this low cost device will have the potential to become commercially successful.

References:

SD Card:

www.cs.ucr.edu/~amitra/**sdcard**/Additional/**sdcard_appnote**_foust.pdf

LCD:

http://www.olimex.com/dev/msp-4619lcd.html

http://dev.ivanov.eu/projects/msp430-4619lcd_sample/index.html

http://www.opencircuits.com/SFE_Footprint_Library_Eagle

FAT:

http://www.digitalspirit.org/file/index.php/obj-download/docs/fat/appnote_fat16.pdf