PROJECT REPORT

on

GPS BASED VIRTUAL FENCING

By

Joel Sequeira

Mainuddin Shaikh

Nathan Peirera

Teja Naik

Under the Guidance of

Prof. Reena Fernandes

(Assistant Professor, ETC Department)

ELECTRONICS AND TELECOMMUNICATION ENGINEERING

Padre Conceição College of Engineering, Goa

GOA UNIVERSITY

2012-2013

Department of Electronics and Telecommunications

Padre Conceição College of Engineering

Verna, Goa-India

CERTIFICATE

This is to certify that this dissertation work entitled

"GPS Based Virtual Fencing"

is a bonafide record of the work done by

Joel Sequeira	PRN.: 200806087
Mainuddin Shaikh	PRN.: 200806111
P. Nathan Peirera	PRN.: 201007355
Teja Naik	PRN.: 200806093

In the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

In

ELECTRONICS AND TELECOMMUNICATIONS ENGINEERING

Under

GOA UNIVERSITY

Project Guide	Head of ETC Dept	Principal	External
Mrs. Reena Fernandes	Dr. K R Pai	Dr. Luis Mesquita	Examiner

ACKNOWLEDGEMENT

We would like to thank all the people who were a part of this project and who gave their unending support from the time the project idea was conceived.

We would like to express our sincere gratitude to our Project Guide as well as our mentor Prof. Reena Fernandes at Padre Conceição College of Engineering, Verna, Goa , for channeling our efforts, supervising and providing the vital inputs which helped us immensely in this project.

We would also like to thank the Head of Department of ETC, Dr.K.R.Pai and the staff of ETC department for their neverending encouragement and guidance.

ABSTRACT

For our final design project, we chose to build a GPS Based Virtual Fencing System. Essentially, the GPS Based Virtual Fencing System is used to create a virtual fence for the user.

We utilized a microcontroller (Atmega32A), GPS module (*i*Wave), a 4X4 keypad, a 16X2 LCD and a GSM module (SIM300).

Numerous projects on GPS have been done throughout the years for various applications, but the idea of using the concept of GPS to create a virtual fence is being implemented for the first time. Nothing can be more intuitive than creating a fence virtually as compared to building a physical one.

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations

GPS – Global Positioning System

GSM– Global System for Mobile Communications

LCD – Liquid Crystal Display

UDR – USART Data Register

NMEA – National Marine Electronics Association

CONTENTS

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

The main rationale behind this project was that it seemed to be a fun and interesting idea to implement. Sources of inspiration initially came from previous projects, specifically the numerous GPS based projects over the years.

Suppose we wish to safe guard any object of a certain value, we can effectively do so by virtually fencing it by using this kind of a system. In this system, we restrict a device within a boundary limited by GPS co-ordinates calculated by the fence parameters specified by the user. Hence if the device crosses the virtual fence, we would get a notification in terms of a message containing the current location of the device. Knowing the current position of the device, we can alert the police and retrieve it.

This project mainly consists of the following parts:
Microcontroller (Atmega32), GPS Module (*i*Wave), 4X4 keypad, 16X2 LCD, GSM Module (SIM300).

The microcontroller arbitrates all the commands and control over the system and acts as the Master interacting with all the other devices. The GPS Module continuously provides the current location of the device. The 4X4 keypad and 16X2 LCD are used for entering the fence parameters and interacting with the system administrator.

1.1 WHY ATMEGA32A?

Our main aim in this project is to create a virtual Fence around an object by using the dimensions of the fence as specified from the user and converting these in terms of latitudes and longitudes. This, along with comparing the co-ordinates as obtained from the GPS module and generating an alarm via the GSM module when the fence is crossed, has to be done in real time. This requires high speed processing and controllability as is accomplished using an Atmega32A microcontroller. Hence we chose Atmega32A over other microcontrollers due to availability of developments tools and the microcontroller chip.

1.2 WHAT MAKES ATMEGA32A UNIQUE?

Atmega32 has the following features useful in our project:

- Advanced RISC Architecture
- 131 Powerful Instructions (Most Single-clock Cycle Execution)
- 32 × 8 General Purpose Working Registers.
- 32Kbytes of In-System Self-programmable Flash program memory
- 1024Bytes EEPROM
- 2Kbytes Internal SRAM
- Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- Programmable Serial USART
- Programmable Watchdog Timer with Separate On-chip Oscillator
- 32 Programmable I/O Lines
- Operating speed 0 - 16MHz
- The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, and interrupt system to continue functioning.

1.3 BLOCK DIAGRAM OF ATMEGA32A

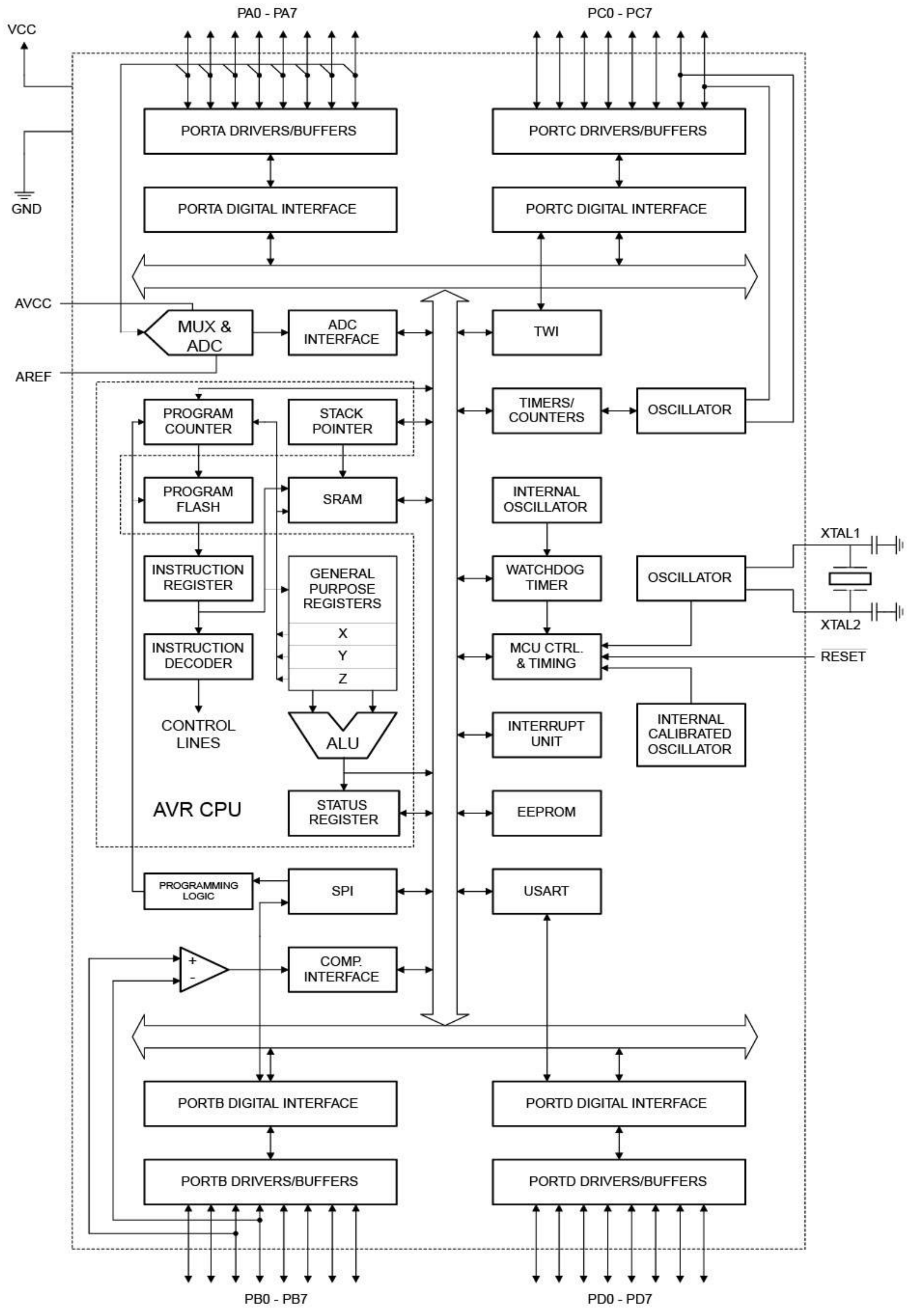


Figure 1: Block Diagram of Atmega32A

1.4 Objectives of the Project:

1) Creation of a virtual fence which would involve:

- GPS interfacing with microcontroller to provide geographical location.
- GSM interfacing with microcontroller to provide a notification via SMS when the object moves out of it’s virtual fence.

2) Theft recovery by monitoring the location of stolen goods, which would require:

- Being able to create the fence dynamically at any location on the Earth irrespective of hill, slope, steep areas etc.,
- Ability to increase the boundary distance dynamically if needed.

CHAPTER 2

GLOBAL POSITIONING SYSTEM (GPS)

2.1 GPS MODULE

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on the Earth where there is an unobstructed line of sight to four or more GPS satellites. The system provides critical capabilities to military, civil and commercial users around the world. It is maintained by the United States government and is freely accessible to anyone with a GPS receiver.

It is a U.S. space-based radio navigation system that provides reliable positioning, navigation, and timing services to civilian users on a continuous worldwide basis -- freely available to all.

- For anyone with a GPS receiver, the ***system will provide location with time***. GPS provides accurate location and time information for an unlimited number of people in ***all weather, day and night, anywhere in the world.***
- The GPS is made up of ***three parts***: ***satellites orbiting the Earth; control and monitoring stations on Earth; and the GPS receivers*** owned by users.
- Each GPS receiver then provides ***three-dimensional location*** (***latitude, longitude, and altitude***) ***plus the time***.

2.2 WORKING OF GPS:

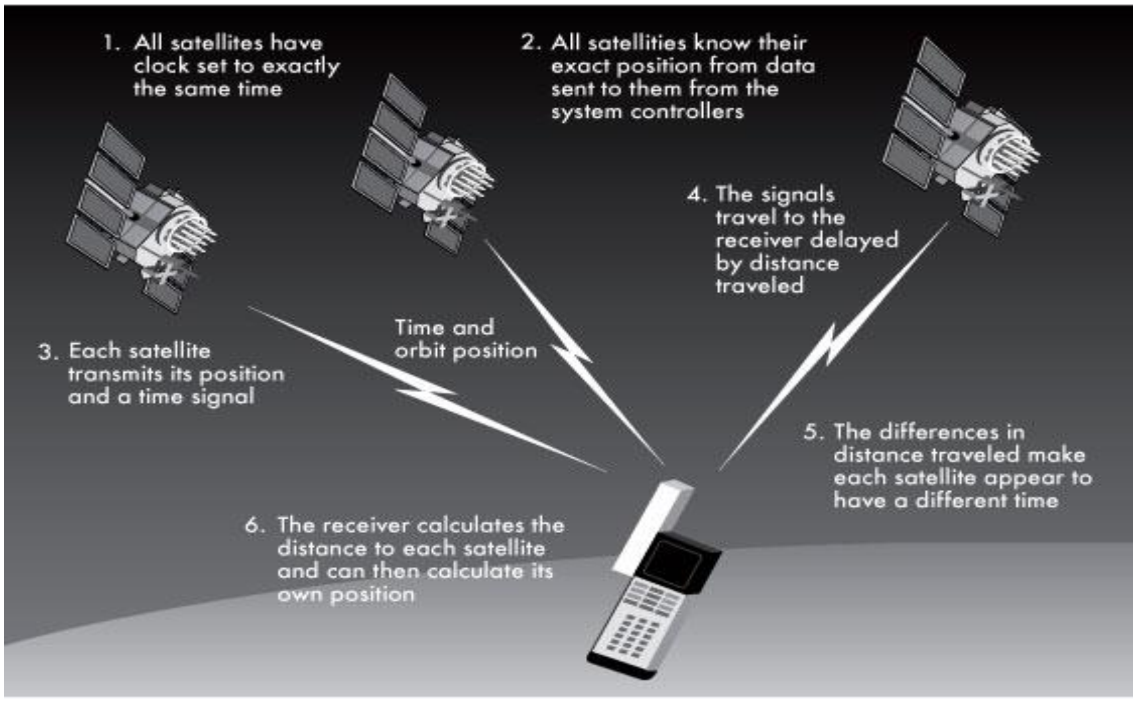

1. All satellites have clock set to exactly the same time
2. All satellites know their exact position from data sent to them from the system controllers
3. Each satellite transmits its position and a time signal
Time and orbit position
4. The signals travel to the receiver delayed by distance traveled
5. The differences in distance traveled make each satellite appear to have a different time
6. The receiver calculates the distance to each satellite and can then calculate its own position

Figure 2: Working of GPS

2.3 GPS RECEIVER (*i*Wave):

Features:

- 65 channels to acquire and track satellites simultaneously
- Industry-leading TTFF speed(time to first fix)
- Tracking sensitivity reaches -161 dBm
- quick cold start
- Integral LNA(low noise amplifier) with low power control
- SBAS (Satellite based augmentation system) (WAAS9*Wide Area Augmentation System)*/EGNOS(*European Geostationary Navigation Overlay Service)*) capable
- Cold start 29 sec under clear Sky
- Hot start 1 sec under clear Sky
- Accuracy 5m CEP(*Circular Error Probability)*
- Operable at 3.6V-6V
- Both of RS232 and UART interface at CMOS level
- Small form factor of 32 mm W x 32 mm Lx 8 mm H
- Mountable without solder process

Applications:

- Personal Positioning
- Automotive and Marine Navigation
- Automotive Navigator Tracking
- Emergency Locator
- Geographic Surveying

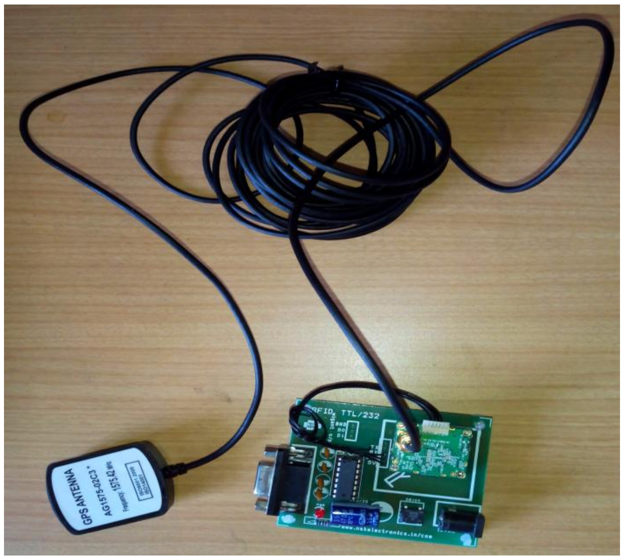
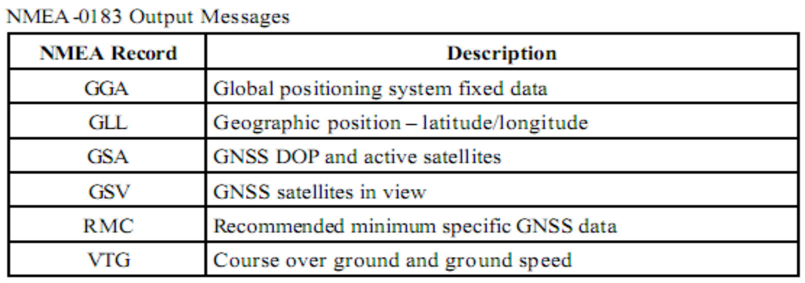


Figure 3: Photo of the GPS receiver

2.4 Extracting GPS Data:

GPS uses the (National Marine Electronics Association) NMEA 0183 protocol for communication. NMEA 0183 messages use the ASCII character set and have a defined format. Each message begins with a $ (hex 0x24) and ends with a carriage return and line feed (hex 0x0D 0x0A). For Our application, we are only interested in the latitude, longitude and time. All this is provided in the GPGGA message string. GP denotes Global positioning and GGA denotes Global Positioning System fixed data.
An example GPGGA string is shown below:
$GPGGA,100156.000,2650.9416,N,07547.8441,E,1,08,1.0,442.8,M,-42.5,M,,0000*71


NMEA -0183 Output Messages

NMEA Record	Description
GGA	Global positioning system fixed data
GLL	Geographic position – latitude/longitude
GSA	GNSS DOP and active satellites
GSV	GNSS satellites in view
RMC	Recommended minimum specific GNSS data
VTG	Course over ground and ground speed

Table 1: NMEA -0183 Output Messages

Procedure for data extraction:

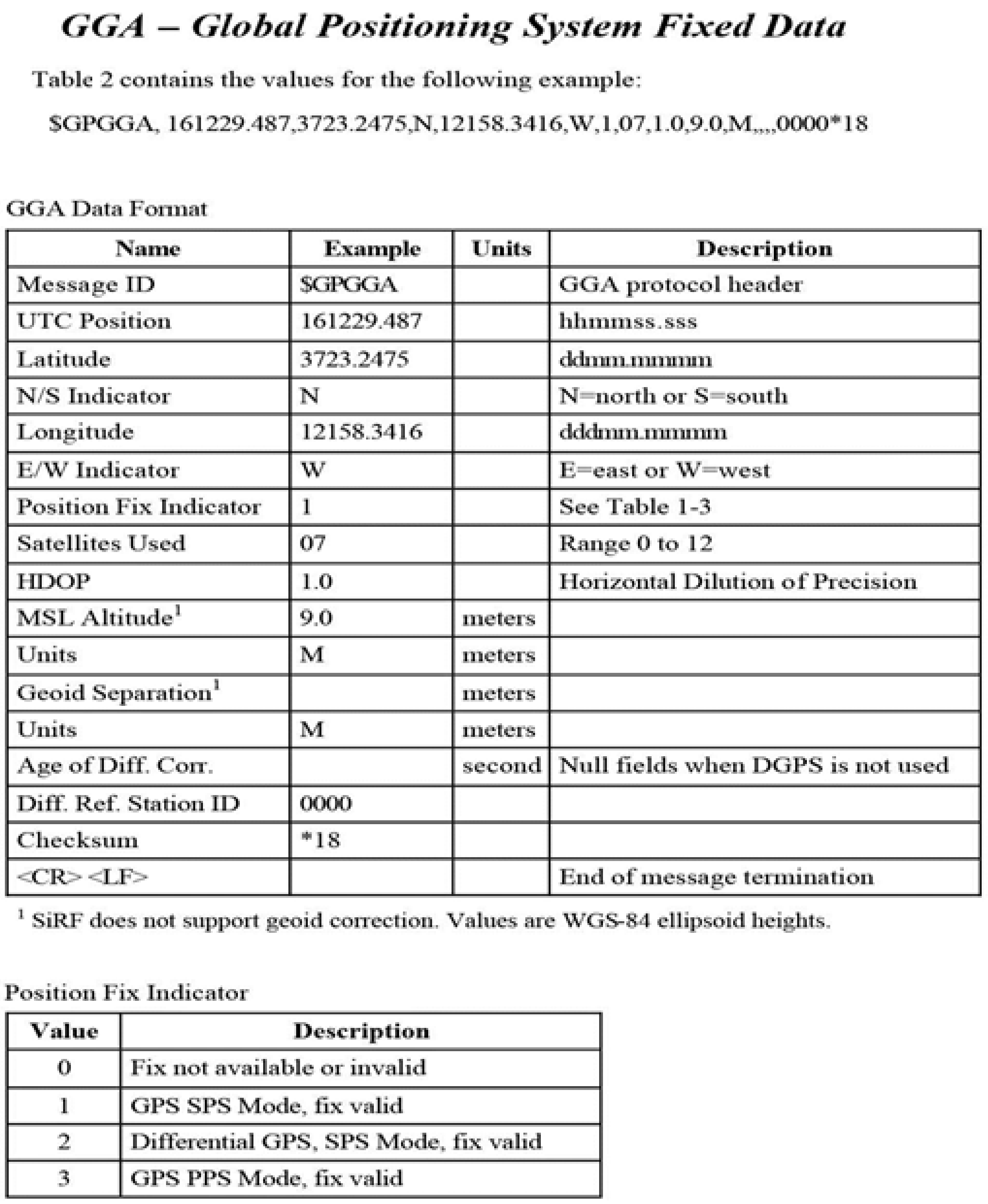
The procedure that we used to extract the latitude and longitude information from the GPS module using $GPGGA string and display it on a LCD is as follows:

1) Get data in USART Data Register (UDR) and check weather that data is equal to $. If the data matches go to step(2) else get a new data.
2) Get data byte by byte and check if the received byte is equal to GPGGA.
3) If the step (2) matches completely then go to step (4) else go back to step(1).
4) Leave first comma and wait till second comma (since we not looking for time).
5) Start taking data in an array lati_value[ ] till the next comma.
6) Get latitude direction in lati_dir.
7) Do the same for longitude.
8) Display the values on LCD and go back to step (1).

An example string has been given and explained below:
$GPGGA,100156.000,2650.9416,N,07547.8441,E,1,08,1.0,442.8,M,-42.5,M,,0000*71

1) A string always start from '$' sign
2) GPGGA: Global Positioning System Fix Data
3) ',' Comma indicates the separation between two values
4) 100156.000 : GMT time as 10(hr):01(min):56(sec):000(ms)
5) 2650.9416,N: Latitude 26(degree) 50(minutes) 9416(sec) NORTH
6) 07547.8441,E: Longitude 075(degree) 47(minutes) 8441(sec) EAST

7) 1 : Fix Quantity 0= invalid data, 1= valid data, 2=DGPS fix
8) 08 : Number of satellites currently viewed.
9) 1.0: HDOP
10) 442.8,M : Altitude (Height above sea level in meter)
11) -42.5,M : Geoids height
12) ,,: DGPS data
13) 0000 : DGPS data
14) *71 : checksum



GGA – Global Positioning System Fixed Data

Table 2 contains the values for the following example:

$GPGGA, 161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M,,,,0000*18

GGA Data Format

Name	Example	Units	Description
Message ID	$GPGGA		GGA protocol header
UTC Position	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Position Fix Indicator	1		See Table 1-3
Satellites Used	07		Range 0 to 12
HDOP	1.0		Horizontal Dilution of Precision
MSL Altitude[1]	9.0	meters	
Units	M	meters	
Geoid Separation[1]		meters	
Units	M	meters	
Age of Diff. Corr.		second	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*18		
<CR> <LF>			End of message termination

[1] SiRF does not support geoid correction. Values are WGS-84 ellipsoid heights.

Position Fix Indicator

Value	Description
0	Fix not available or invalid
1	GPS SPS Mode, fix valid
2	Differential GPS, SPS Mode, fix valid
3	GPS PPS Mode, fix valid

Table 2: GPGGA data format

CHAPTER 3

GLOBAL SYSTEM FOR MOBILE COMMUNICATIONS (GSM)

3.1 GSM MODULE(SIM300)

GSM (Global System for Mobile Communications, originally Groupe Spécial Mobile), is a standard set developed by the European Telecommunications Standards Institute (ETSI) to describe protocols for second generation (2G) digital cellular networks used by mobile phones.
The GSM standard was developed as a replacement for first generation (1G) analog cellular networks, and originally described a digital, circuit switched network optimized for full duplex voice telephony. This was expanded over time to include data communications, first by circuit switched transport, then packet data transport via GPRS (General Packet Radio Services) and EDGE (Enhanced Data rates for GSM Evolution or EGPRS).Further improvements were made when the 3GPP developed third generation (3G) UMTS standards followed by fourth generation (4G) LTE Advanced standards.
We made use of AT commands to communicate with the GSM.

Features:

- This GSM modem is a highly flexible plug and play quad band GSM modem
- Direct and easy integration to RS232.
- Supports features like Voice, Data/Fax, SMS, GPRS and integrated TCP/IP stack.
- Control via AT commands(GSM 07.07,07.05 and enhanced AT commands)
- Use AC – DC Power Adaptor with following ratings · DC Voltage : 12V /1A
- Current Consumption in normal operation 250mA, can rise up to 1Amp while transmission.

Applications:

- Designed for global market, SIM300 is a Tri-band GSM/GPRS engine
- SIM300 can fit almost all the space requirements in your applications, such as smart phone, PDA phone and other mobile devices.
- Can be used to send sms, calling, etc.

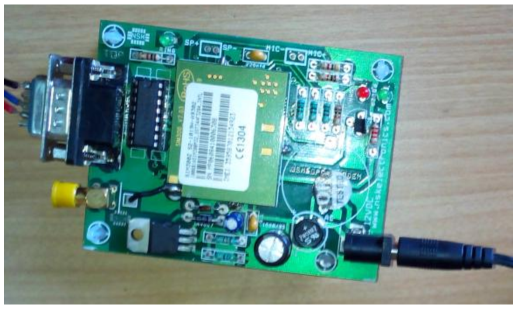


Figure 4: Photo of the GSM module

3.2 AT Commands:

AT commands are instructions used to control a modem. AT is the abbreviation of Attention. Every command line starts with "AT" or "at". That's why modem commands are called AT commands. Many of the commands that are used to control wired dial-up modems, such as ATD (Dial), ATA (Answer), ATH (Hook control) and ATO (Return to online data state), are also supported by GSM/GPRS modems and mobile phones. Besides this common AT command set, GSM/GPRS modems and mobile phones support an AT command set that is specific to the GSM technology, which includes SMS-related commands like AT+CMGS (Send SMS message), AT+CMSS (Send SMS message from storage), AT+CMGL (List SMS messages) and AT+CMGR (Read SMS messages).

Note that the starting "AT" is the prefix that informs the modem about the start of a command line. It is not part of the AT command name. For example, D is the actual AT command name in ATD and +CMGS is the actual AT command name in AT+CMGS. However, some books and web sites use them interchangeably as the name of an AT command.
There are two types of AT commands: basic commands and extended commands. Basic commands are AT commands that do not start with "+". For example, D (Dial), A (Answer), H (Hook control) and O (Return to online data state) are basic commands. Extended commands are AT commands that start with "+". All GSM AT commands are extended commands. For example, +CMGS (Send SMS message), +CMSS (Send SMS message from storage), +CMGL (List SMS messages) and +CMGR (Read SMS messages) are extended commands.

In this Project, we only require to send an SMS via the GSM module when the device crosses the virtual fence boundaries to serve as an indication to the user that the item has been stolen. This is effectively done using the following AT commands.
Sending SMS :
1) Test Command:
AT+CMGS=?
Response : OK
2) If text mode i.e.(+CMGF=1):
AT+CMGS=<da><CR>   ; <da> denotes destination adresss(i.e. mobile phone number)
text is entered
>message<CTRL-Z>
e.g.AT+CMGS="9860329813"<CR> Hello! How are You? <CTRL-Z>

CHAPTER 4

USER INTERACTING DEVICES

(4x4 Matrix Membrane Keypad and 16X2 LCD)

4.1 4X4 Matrix Membrane Keypad:

This 16-button keypad provides a useful human interface component for microcontroller projects.

Figure 5: 4X4 Matrix Membrane Keypad

How it Works?

Matrix keypads use a combination of four rows and four columns to provide button states to the host device, typically a micrcontroller. Underneath each key is a pushbutton, with one end connected to one row, and the other end connected to one column. These connections are shown in the figure below.

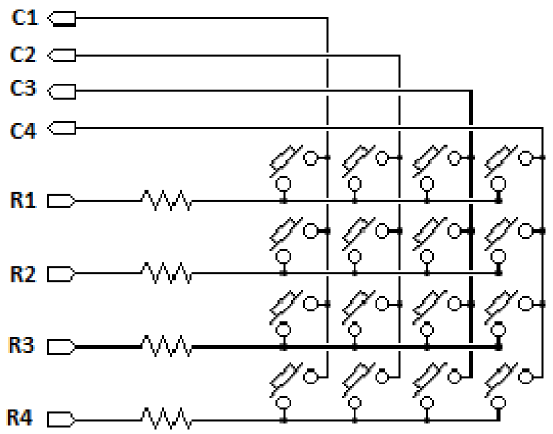


Figure 6: 4X4 Matrix Keypad Connections

In order for the microcontroller to determine which button is pressed, it first needs to pull each of the four rows (R1-R4) either low or high one at a time, and then poll the states of the four columns (C1-C4). Depending on the states of the columns, the microcontroller can detect which button is pressed.

4.2 16X2 LCD:

LCD (Liquid Crystal Display) screen is an electronic display module and finds a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits.

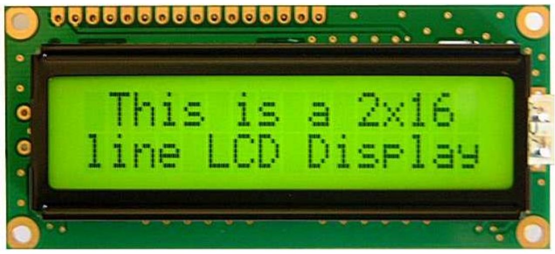
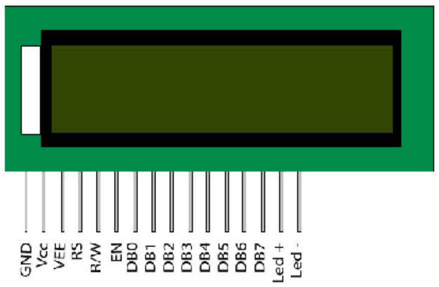
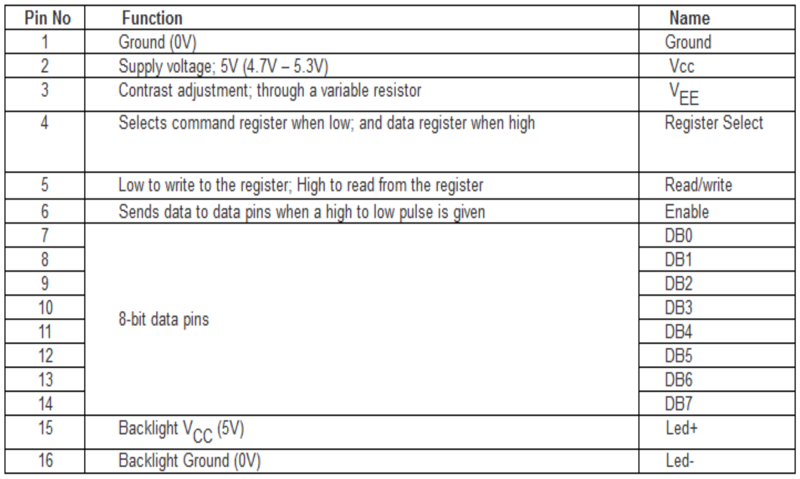


Figure 7: 16X2 LCD

These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on. We use the LCD in the 8-bit data mode.
A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.
The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. The figure below shows the pins of the LCD with the table next to it stating their corresponding descriptions.

16X2 LCD Pin Descriptions:





Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

Figure 8: 16X2 LCD Connections

CHAPTER 5

SYSTEM WORKING

5.1 SYSTEM BLOCK DIAGRAM:

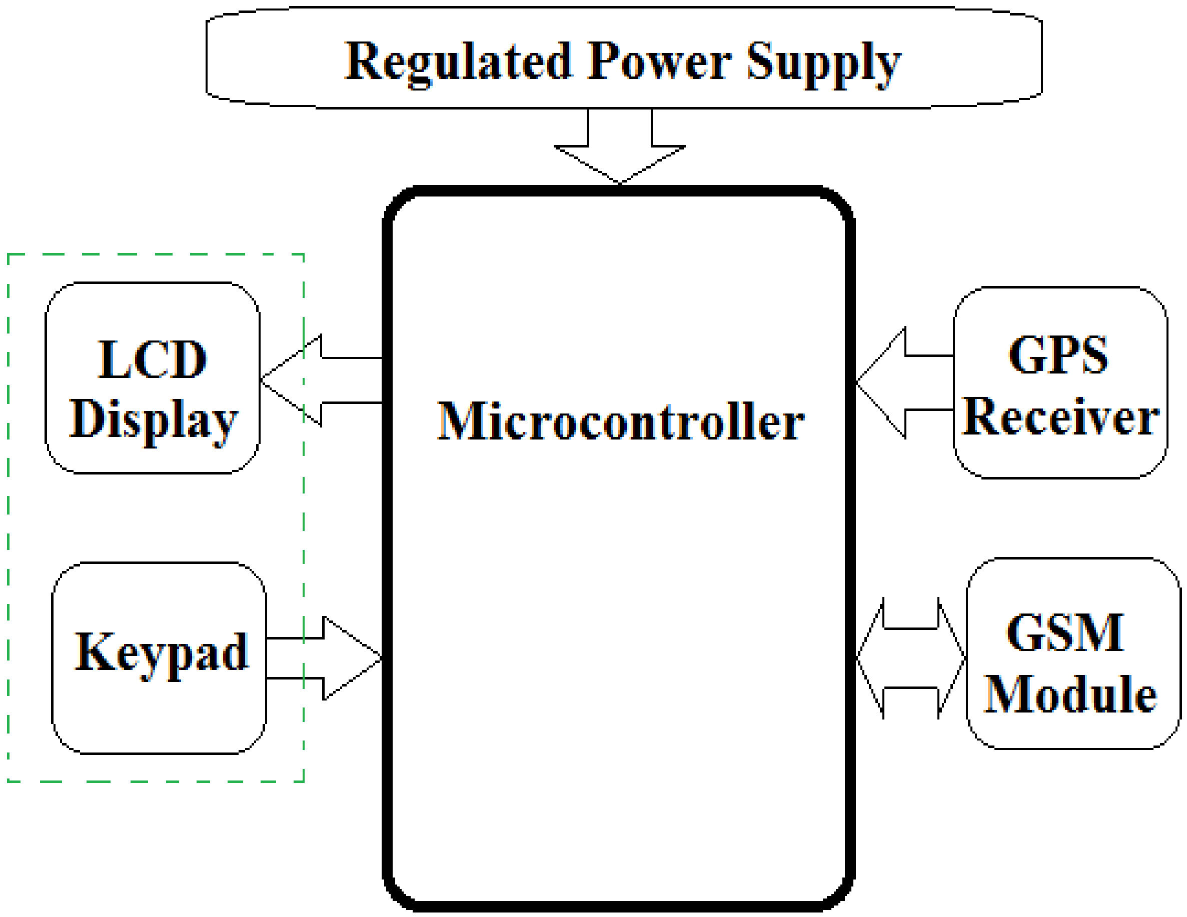


Figure 9: System Block Diagram

EXPLANATION:

The block diagram consists of a microcontroller (Atmega32A) which is interfaced to a GPS module (*i*Wave), a GSM module (SIM300) via an RS232 connection; a 4X4 keypad, a 16X2 LCD and other circuits which serve as indicators as per the application.

The keypad and LCD can be disconnected from the system once the virtual fence dimensions have been specified by the user until future re-configuration.

First, the LCD displays a choice to the user whether he would wish to re-configure the fence or use previously entered dimensions of the fence. Depending on the user's choice, the system would proceed as follows.

When indicated by the user that the fence needs to be re-configured, the LCD would then display to the user whether he would like to enter the fence parameters in the units of meters (0 to 999) or kilometres (0 to 999). Depending on the user's choice the fence parameters would be stored in the microcontroller.

The LCD would next indicate to the user to enter the mobile number of the mobile phone that the system would send the SMS to, when the virtual fence is crossed. The microcontroller would then compute the dimensions of the fence considering the User (GPS Fence system) as the centre point of a square and convert these in terms of latitudes and longitudes.

Next, the microcontroller would constantly compare these co-ordinates with those received from the GPS unit. In case a value beyond the virtual boundary would occur, it would send an indication to the user via the GSM that the device has travelled beyond its specified boundary and is currently located at so and so position.

In case the user initially chose to continue using previously entered values for the virtual fence, the system would proceed normally without asking the user to re-enter the dimensions of the virtual fence.

Hence this system would effectively serve as an efficient means to create and monitor a fence around an object virtually and hence help safeguard it.

The above explanation is also shown in the following section in terms of a flowchart for better understanding.

5.2 Flowchart of the system:

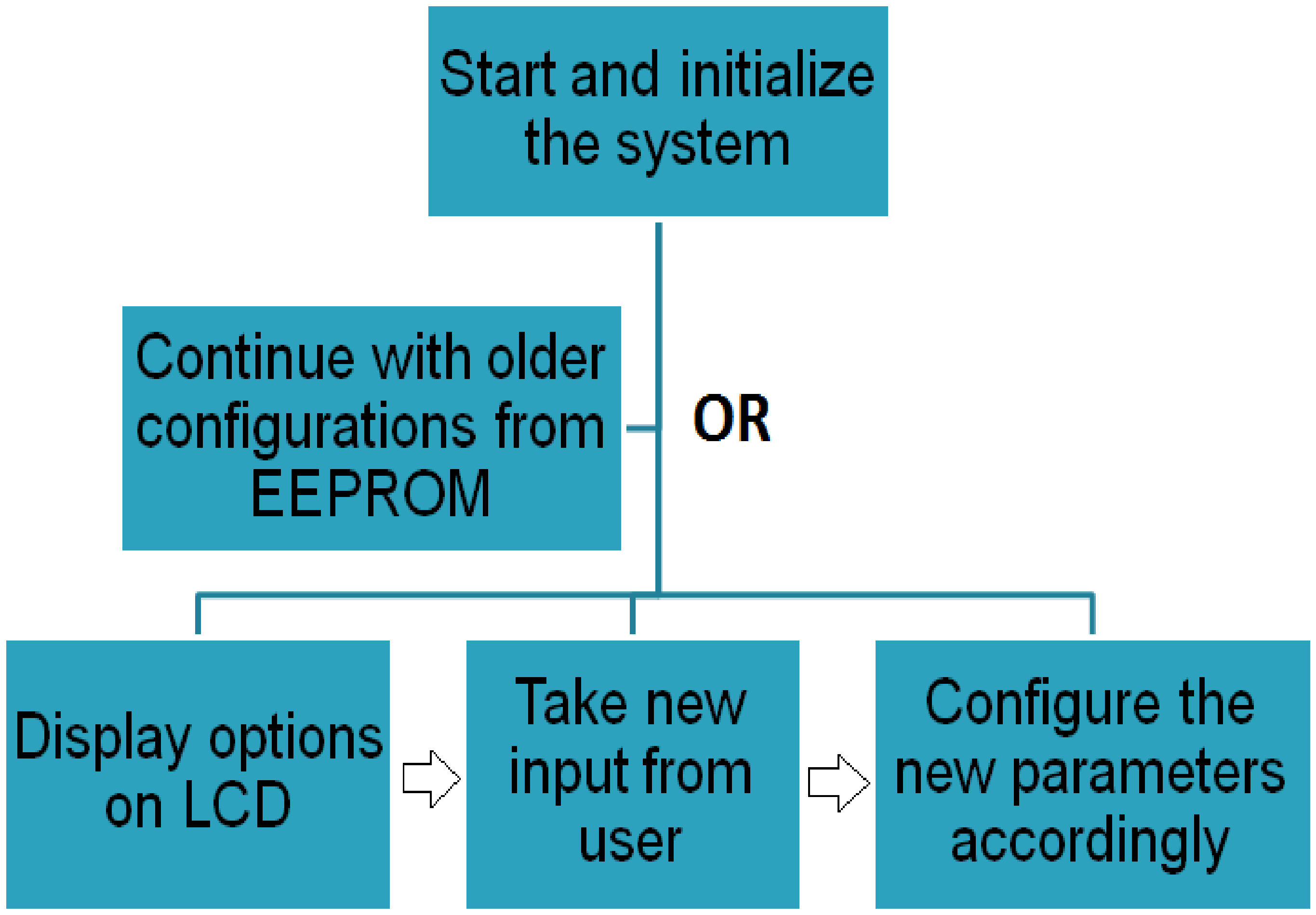

Start and initialize the system
Continue with older configurations from EEPROM
OR
Display options on LCD
Take new input from user
Configure the new parameters accordingly

Figure 10: System Flowchart

5.3 Fencing Algorithm:

Consider the navigational World map as shown in the figure below.

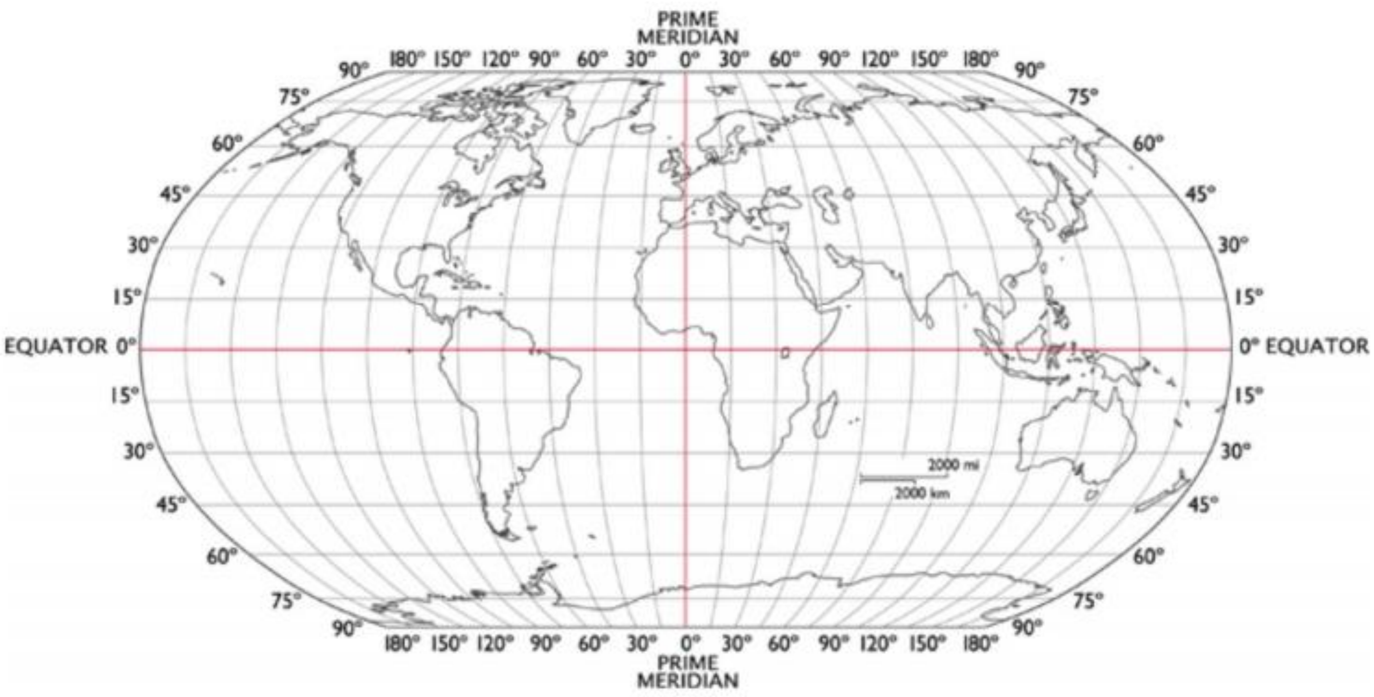


Figure 11: Navigational World Map

The World map is divided in terms of

- latitudes (horizontal, parallel lines) ranging from 0 to 90 on either side of the equator, and in terms of
- longitudes (vertical, non-parallel lines) ranging from 0 to 180 on either side of the prime meridian.

For simplicity, let us assume that

<u>1 (degree) of latitude = 111 km (kilo-meters) approx.</u> throughout the entire map.
The same relationship in terms of km is also considered for longitudes although it differs significantly in this case when nearing the poles.
Hence, we obtain that,
1’ (minute) = 1.85 km and
1” (second) = 0.03083 km.
Therefore, 1” = 30.83 m (meters).

To better understand the creation of a virtual fence, consider a simplified figure of the geographic coordinates of the Earth as shown in the figure below.

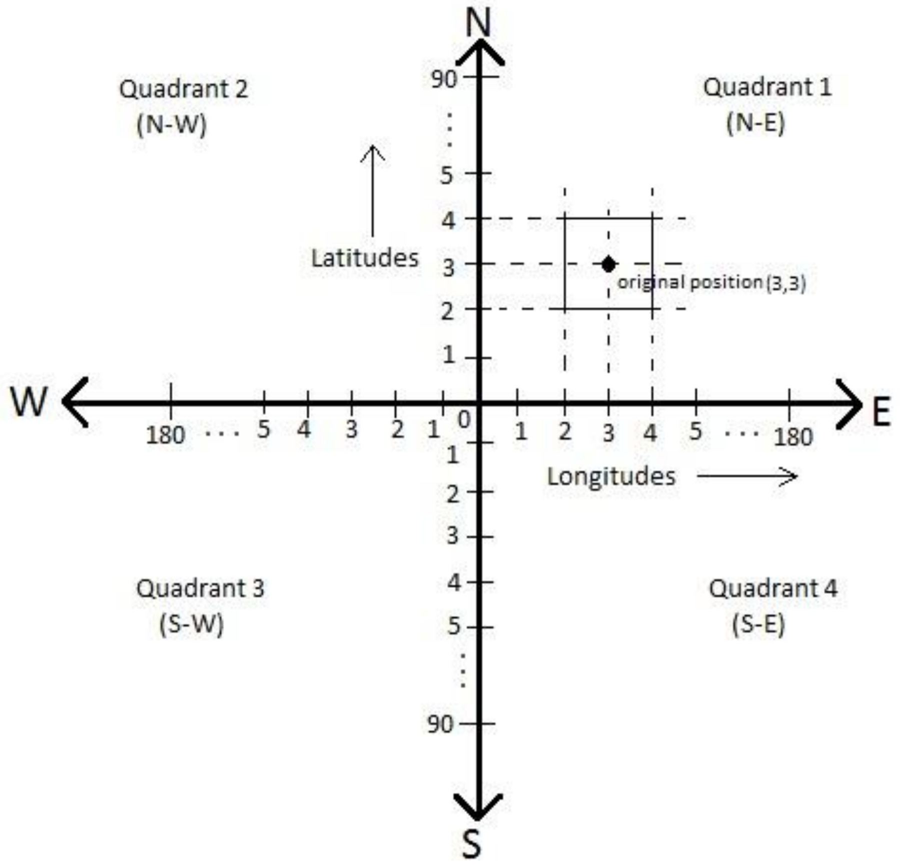


Figure 12: Simplified figure of the geographic coordinates of the Earth

Consider the original position (while creating the virtual fence initially) of the object as shown in the figure. Using the parameters entered by the user, the microcontroller will compute the virtual fence parameters as shown and constantly monitor the object's current location. It will compare the current location with that of the initial location and check whether the object has crossed the area of the boundary limited effectively by the co-ordinates as seen in the figure. In case it does cross the boundary, it will send notification to the user via GSM module.

The worst case of this computational problem would be when the initial position of the object is at a position where the boundary limits would extend into more than a single quadrant. This is effectively taken care of by using the positional information provided by the GPS module in terms of North-South, East-West with every latitude and longitude information received as is seen in the GPS message GPGGA typical format explained in the previous section on Extracting GPS data.

5.4 Circuit Diagram:

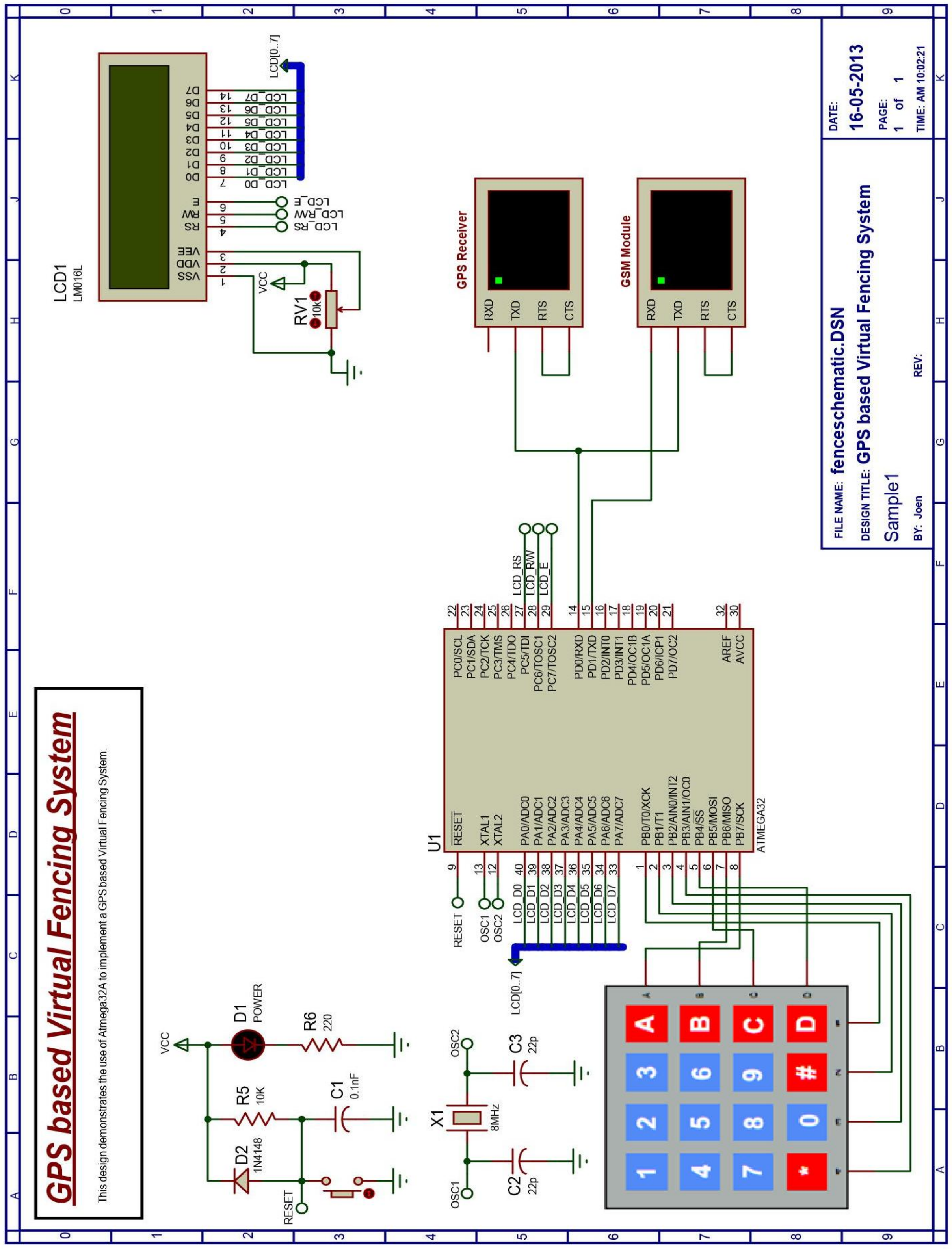

GPS based Virtual Fencing System
This design demonstrates the use of Atmega32A to implement a GPS based Virtual Fencing System.
LCD1
LM016L
RV1
10k
GPS Receiver
GSM Module
U1
ATMEGA32
D1
POWER
R6
220
R5
10K
C1
0.1nF
D2
1N4148
RESET
X1
8MHz
C2
22p
C3
22p
OSC1
OSC2
LCD[0..7]
FILE NAME: fenceschematic.DSN
DESIGN TITLE: GPS based Virtual Fencing System
Sample1
BY: Joen
REV:
DATE: 16-05-2013
PAGE: 1 of 1
TIME: AM 10:02:21

Figure 13: System Circuit Diagram

5.5 System Example:

The figure below shows an example of how the system would work.

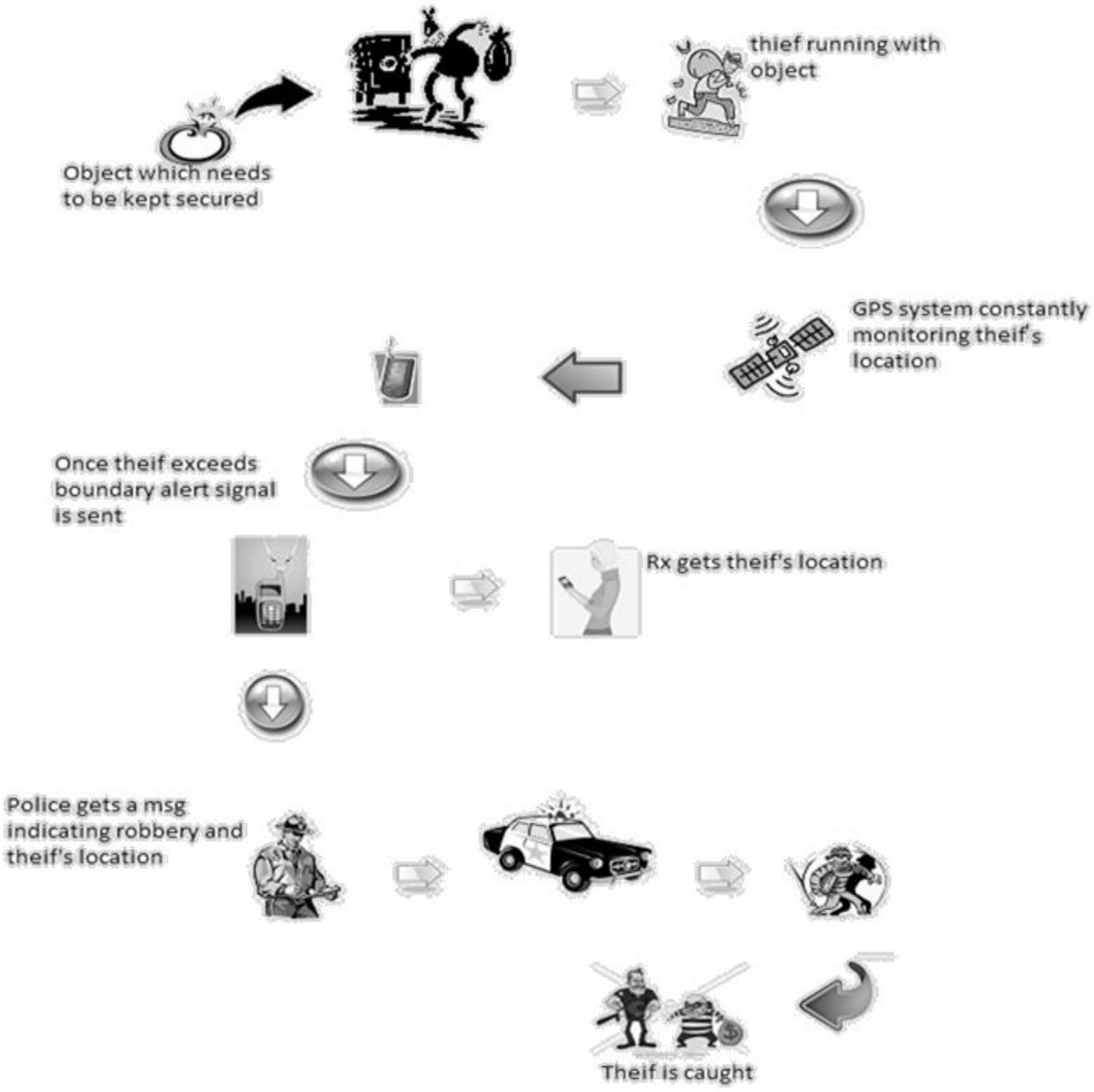

Object which needs to be kept secured
thief running with object
GPS system constantly monitoring theif's location
Once theif exceeds boundary alert signal is sent
Rx gets theif's location
Police gets a msg indicating robbery and theif's location
Theif is caught

Figure 14: System Example

CHAPTER 6

APPLICATIONS AND SOFTWARES

6.1 APPLICATIONS:

- Theft detection and recovery of stolen goods.
- The size of the kit can be reduced to embedded chip and made easy for usage.
- It can be used in boarder alerting system for military, marine surveying and land surveying with appropriate modifications.
- To restrict animals/pets/criminals to certain areas, like restricting elephants to forests and preventing them from crossing dangerous boundaries by subjecting a shock using muscle stimulator circuit.
- Since the project result is mainly based on the GPS signals, by using a better quality of GPS receiver for more accuracy results in accurate and spontaneous applications.
- Also the battery supply can be enhanced by using rechargeable batteries with good battery backup.

6.2 Cost of the project:

List of components:

• Microcontroller Atmega32	Rs 250
• GPS receiver module	Rs 1800
• GSM module	Rs 1600
• 16X2 LCD Display	Rs 130
• 4X4 membrane keypad	Rs 400
• Connecting wires, program burners, testing boards, other circuit components, etc.	Rs 1500
Total =	Rs 6000 approx.

So 6000/number of group members(4), therefore each group member contributed Rs 1500 approx., Hence the project was FEASIBLE in terms of Cost.

6.3 Software used:

The following softwares were used in the making of this project:
For writing program and simulating the code:
Atmel AVR Studio 6.0, Labcenter Electronics Proteus 7.10 VR studio
For Burning the Code into the Microcontroller:
SinaProg ver. 1.3.5.6 via AVR Dude(USBasp) in USB mode.
For testing the GPS Receiver and the GSM module:
Trimble Studio, Hyperterminal

CONCLUSION:

The Project was successfully completed and submitted at the due date for submission. The Project idea had to be modified quite a number of times due to various factors such as cost, hardware limitations, etc. The hardware that we used consisted of lower cost GPS modules which were not capable of obtaining a GPS position fix under obstructed line of sight and hence we managed to design and implement only a small working system demonstrating the concept of virtual fencing.
Wireless implementation of the system could not be implemented due to high current requirement of the GSM and GPS modules that were used. This could be overcome in future implementations of this kind of a system by making use of the GPS receiver and GSM module of very small size and lower current requirement.
The Idea of virtual fencing could easily be implemented on a mobile phone which already has the GPS receiver and GSM modules integrated into it. By writing/modifying a simple GPS tracker program and including the GPS fencing algorithm into it, such a system could help in mobile theft retrieval. We were not able to implement such a system on a mobile phone due to "Mobile phone programming" elective being introduced to us only in the last eight semester. Hence we were short on time to try to implement such a system on a mobile phone.

The following motives were achieved after the completion of the project:
A GPS based virtual fencing system was implemented. The system was built as a model to demonstrate the idea of the concept of a virtual fence.
The system was seen to be working fine in an open environment having a proper line of sight for the GPS receiver to the satellites.

The entire setup was tested and the minimum fence range was found to be 20mts around the object. We also propose, that the above idea could be used in various applications for example in theft detection and retrieval of mobile phones or any other valuable item.

We also understood the basic concept of using a common ground when connecting two circuits which use different power supplies. Also we were able to understand the basic working of the microcontroller and programming an AVR microcontroller in assembly language throughout the duration of the project. During the course of the project we strengthened our fundamentals in electronics and interfacing different devices.

APPENDIX

BIBLIOGRAPHY

References

- http://geography.about.com/od/geographictechnology/a/gps.htm
- http://support.radioshack.com/support_tutorials/gps/gps_tmline.htm
- http://electronics.howstuffworks.com/gadgets/travel/gps.htm
- http://en.wikipedia.org/wiki/Global_Positioning_System
- www.csiro.au/science/Virtual-Fencing-Project.html
- http://www.developershome.com/sms/atCommandsIntro.asp

GPS Based Virtual Fencing

Joel Sequeira, Mainuddin Shaikh, Nathan Pereira and Teja Naik
Padre Conceicao College of Engineering, Verna-Goa

E-mail: Jomanate1213@gmail.com
Contact Numbers: +919860329813, +917875048943, +919405331510, +918698684912

Abstract— **Global Positioning System (GPS) has taken rapid strides with the proliferation of satellites and integration with other systems. GPS can track down the location and velocity of an object anywhere on the globe. A man on the street can use his cell phone with GPS capability to know his whereabouts and to select the best possible route to his destination.**

We chose to build a GPS based Virtual Fencing System that essentially creates a virtual fence around an object and continuously monitors and reports its location when it crosses the fence.

I. Introduction

A fence demarcates an area, setting its boundary limits. The virtual fence has no brick and mortar walls. Yet it has within its perimeter a specified area defined by the geographical latitude and longitude. It uses the GPS to specify the area limits and also to modify them as desired. The virtual fence can have wide applications. It can be used to trace stolen and lost items and prevent valuable artifacts from leaving the country.

II. System Block Diagram

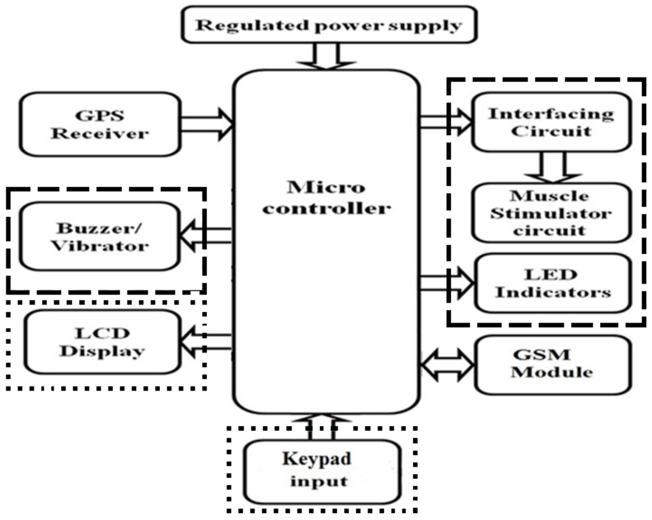


Figure (1) System Block Diagram

The block diagram consists of a microcontroller (capable of high speed processing in real time) which is interfaced to a GPS module, a GSM module; a keypad, a LCD and other circuits which serve as indicators as per the different applications. The other circuits are shown enclosed in dashed lines in the figure (1). The keypad and LCD can be disconnected from the system once the virtual fence dimensions have been specified by the user until future re-configuration. This is shown using dotted lines enclosing the keypad and LCD. Hence the system would essentially require only the microcontroller, the GPS module and the GSM (Global system for mobile Communications) module to be placed on the object around which we wish to create the fence, at all times.

III. System Working

Initially, when setting up the virtual fence around an object to which this system is going to be attached, the user would be asked to enter the boundary parameters of the virtual fence via the keypad-LCD interface. As an example consider the case of the user being asked to enter the radius (circle). The microcontroller would then compute the dimensions of the fence considering the User (GPS Fence system) as the center point of the figure (circle) and convert these in terms of latitudes and longitudes.

Next, the microcontroller would constantly compare these co-ordinates with those received from the GPS unit. In case a value beyond the virtual boundary would occur, it would send an indication to the user via the GSM that the device has travelled beyond its specified boundary and is currently located at so and so position.

Hence this system would effectively serve as an efficient means to create and monitor a fence around an object virtually and hence help safeguard it.

The following explains the working and basic requirements (for this system) of the different modules used in this system (GPS module and GSM module and microcontroller).

A. GPS Module

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The system provides critical capabilities to military, civil and commercial users around the world. It is maintained by the United States government and is freely accessible to anyone with a GPS receiver. The GPS Module follows the NMEA protocol.

NMEA 0183 messages use the ASCII character set and have a defined format. Each message begins with a $ (hex 0x24) and end with a carriage return and line feed (hex 0x0D 0x0A)

An example string has been given and explained below:

$GPGGA,100156.000,2650.9416,N,07547.8441,E,1,08,1.0,442.8,M,-42.5,M,,0000*71

A string always start from ‘$’ sign. GPGGA: Global Positioning System Fix Data. ‘,’ Comma indicates the separation between two values. 100156.000 : GMT time as 10(hr):01(min):56(sec):000(ms). 2650.9416,N: Latitude 26(degree) 50(minutes) 9416(sec) NORTH. 07547.8441,E: Longitude 075(degree) 47(minutes) 8441(sec) EAST. 1 : Fix Quantity 0= invalid data, 1= valid data, 2=DGPS fix. 08 : Number of satellites currently viewed. 1.0: HDOP. 442.8,M : Altitude (Height above sea level in meter). -42.5,M : Geoids height. ,,: DGPS data. 0000 : DGPS data. *71 : checksum

Hence for Our system, we are mainly interested in the latitudes and longitudes positional information of the object which is being continuously monitored by the microcontroller unit.

B. GSM Module

In our system, a GSM module basically capable of sending SMS (short message service) to the user when the device around which the fence had been created has crossed its boundary is used.

C. Microcontroller

A microcontroller capable of fast response to meet real time applications was chosen. Care was also taken that it was capable of asynchronous communication and compatible with the GPS and GSM modules used.

IV. Fencing ALGORITHM

In this section, a typical algorithm for creating a virtual fence as used in this system is explained.

Consider the navigational World map as shown in figure (2).

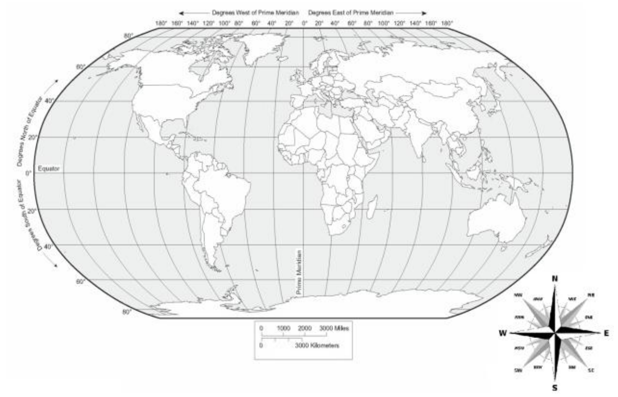


Figure (2) World Map

The World map is divided in terms of latitudes (horizontal, parallel lines) ranging from 0° to 90° on either side of the equator, and in terms of longitudes (vertical, non-parallel lines) ranging from 0°° to 180° on either side of the prime meridian. For simplicity, let us assume that 1° (degree) of latitude = 111 km (kilo-meters) approximately throughout the entire map. The same relationship in terms of km is also considered for longitudes although it differs significantly in this case when nearing the poles. Hence, we obtain that 1’ (minute) = 1.85 km and 1” (second) = 0.03083 km. Therefore, 1” = 30.83 m (meters).

To better understand the creation of a virtual fence, consider a simplified figure of the World map as shown in figure (3).

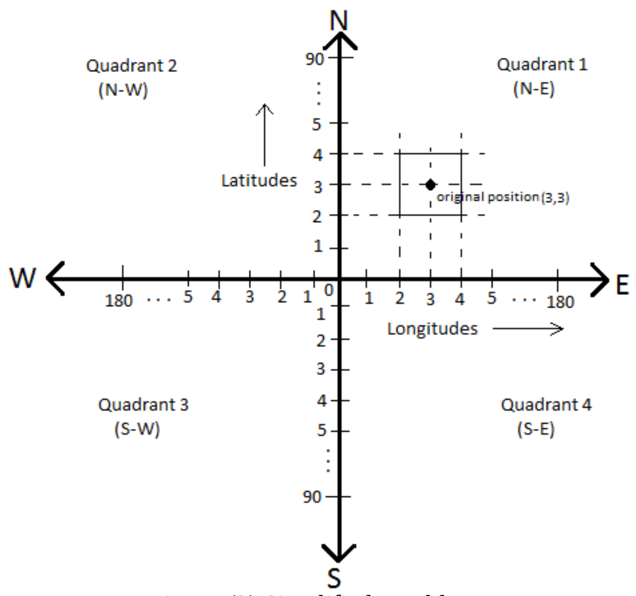


Figure (3) Simplified World Map

Consider the original position (while creating the fence initially) of the object as shown in the figure (3). Using the parameters entered by the user, the microcontroller will compute the virtual fence parameters as shown and constantly monitor the object’s current location. It will compare the current location with that of the initial location and check whether the object has crossed the area of the boundary limited effectively by the co-ordinates as seen in the figure (3). In case it does cross the boundary, it will send notification to the user via GSM module.

The worst case of this computational problem would be when the initial position of the object is at a position where the boundary limits would extend into more than a single quadrant. This is effectively taken care of by using the positional information provided by the GPS module in terms of North-South, East-West with every latitude and longitude information received as was seen previously in the GPS message GPGGA typical format.

V. Applications and Future Scope

This system can be used in border alerting system for military, marine surveying and land surveying. It can also be used to restrict animals/pets/criminals to certain areas, like restricting elephants to forests and preventing them from crossing dangerous boundaries by subjecting a shock using muscle stimulator circuit. Could be used extensively for prevention and theft detection of stolen goods especially devices like mobile phones which already have the GPS and GSM modules embedded in it.

Future Scope:

The size of the kit can be reduced to embedded chip and made easy for usage. Since the project result is mainly based on the GPS signals, by using a better quality of GPS receiver for more accuracy results in accurate and spontaneous applications. Also the battery supply can be enhanced by using rechargeable batteries with good battery backup.

References

[1] "Why Did the Department of Defense Develop GPS?" Trimble Navigation Ltd. http://www.trimble.com/gps/whygps.shtml#0. Retrieved 2010-01-13.

[2] "A Guide to the Global Positioning System (GPS) - GPS Timeline". Radio Shack. http://support.radioshack.com/support_tutorials/gps/gps_tmline.htm. Retrieved 2010-01-14.

[3] Daly, P.. "Navstar GPS and GLONASS: global satellite navigation systems". IEEE. http://ieeexplore.ieee.org/iel1/2219/7072/00285510.

[4] The Global Positioning System by Robert A. Nelson Via Satellite, November 1999

[5] http://www.howstuffworks.com

[6] http://en.wikipedia.org/wiki/Gps

[7] http://www.csirType equation here.o.au/science/Virtual-Fencing-Project

The following code example shows how to read the UCSRC Register contents.

Assembly Code Example[1]

```
USART_ReadUCSRC:
  ; Read UCSRC
  in  r16,UBRRH
  in  r16,UCSRC
  ret
```

C Code Example[1]

```
unsigned char USART_ReadUCSRC( void )
{
  unsigned char ucsrc;
  /* Read UCSRC */
  ucsrc = UBRRH;
  ucsrc = UCSRC;
  return ucsrc;
}
```

Note: 1. See "About Code Examples" on page 6.

The assembly code example returns the UCSRC value in r16.

Reading the UBRRH contents is not an atomic operation and therefore it can be read as an ordinary register, as long as the previous instruction did not access the register location.

19.11 Register Description

19.11.1 UDR – USART I/O Data Register

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR Register location. Reading the UDR Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDRE Flag in the UCSRA Register is set. Data written to UDR when the UDRE Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use read modify write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

19.11.2 UCSRA – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	RXC	**TXC**	**UDRE**	**FE**	**DOR**	**PE**	**U2X**	**MPCM**	**UCSRA**
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXC: USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

- **Bit 6 – TXC: USART Transmit Complete**

This flag bit is set when the entire frame in the transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

- **Bit 5 – UDRE: USART Data Register Empty**

The UDRE Flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE Flag can generate a Data Register empty Interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the transmitter is ready.

- **Bit 4 – FE: Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. i.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

- **Bit 3 – DOR: Data OverRun**

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

- **Bit 2 – PE: Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the parity checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

- **Bit 1 – U2X: Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART receiver that do not contain address information will be ignored. The transmitter is unaffected by the MPCM setting. For more detailed information see "Multi-processor Communication Mode" on page 164.

19.11.3 UCSRB – USART Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXCIE: RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC Flag. A USART Receive Complete Interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

- **Bit 6 – TXCIE: TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC Flag. A USART Transmit Complete Interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

- **Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE Flag. A Data Register Empty Interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

- **Bit 4 – RXEN: Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and PE Flags.

- **Bit 3 – TXEN: Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the transmit Shift Register and transmit Buffer Register do not contain data to be transmitted. When disabled, the transmitter will no longer override the TxD port.

- **Bit 2 – UCSZ2: Character Size**

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character Size) in a frame the receiver and transmitter use.

- **Bit 1 – RXB8: Receive Data Bit 8**

RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.

- **Bit 0 – TXB8: Transmit Data Bit 8**

TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDR.

19.11.4 UCSRC – USART Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

The UCSRC Register shares the same I/O location as the UBRRH Register. See the "Accessing UBRRH/ UCSRC Registers" on page 165 section which describes how to access this register.

- **Bit 7 – URSEL: Register Select**

This bit selects between accessing the UCSRC or the UBRRH Register. It is read as one when reading UCSRC. The URSEL must be one when writing the UCSRC.

- **Bit 6 – UMSEL: USART Mode Select**

This bit selects between Asynchronous and Synchronous mode of operation.

Table 19-4. UMSEL Bit Settings

UMSEL	Mode
0	Asynchronous Operation
1	Synchronous Operation

- **Bit 5:4 – UPM1:0: Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the PE Flag in UCSRA will be set.

Table 19-5. UPM Bits Settings

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

- **Bit 3 – USBS: Stop Bit Select**

This bit selects the number of Stop Bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 19-6. USBS Bit Settings

USBS	Stop Bit(s)
0	1-bit
1	2-bit

- **Bit 2:1 – UCSZ1:0: Character Size**

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

Table 19-7. UCSZ Bits Settings

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 – UCPOL: Clock Polarity**

This bit is used for Synchronous mode only. Write this bit to zero when Asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

Table 19-8. UCPOL Bit Settings

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

19.11.5 UBRRL and UBRRH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The UBRRH Register shares the same I/O location as the UCSRC Register. See the "Accessing UBRRH/ UCSRC Registers" on page 165 section which describes how to access this register.

- **Bit 15 – URSEL: Register Select**

This bit selects between accessing the UBRRH or the UCSRC Register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

- **Bit 14:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the 8 least significant bits of the USART baud rate. Ongoing transmissions by the transmitter and receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

19.12 Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Table 19-9. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see "Asynchronous Operational Range" on page 163). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Table 19-9. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	**f_{osc} = 1.0000 MHz**				**f_{osc} = 1.8432 MHz**				**f_{osc} = 2.0000 MHz**			
	U2X = 0		**U2X = 1**		**U2X = 0**		**U2X = 1**		**U2X = 0**		**U2X = 1**	
	UBRR	**Error**	**UBRR**	**Error**	**UBRR**	**Error**	**UBRR**	**Error**	**UBRR**	**Error**	**UBRR**	**Error**
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max [1]	62.5 kbps		125 kbps		115.2 kbps		230.4 kbps		125 kbps		250 kbps	

1. UBRR = 0, Error = 0.0%

Table 19-10. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 3.6864 MHz				f_{osc} = 4.0000 MHz				f_{osc} = 7.3728 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max [(1)]	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps	

1. UBRR = 0, Error = 0.0%

Table 19-11. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max [1]	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

1. UBRR = 0, Error = 0.0%

Table 19-12. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

	f_{osc} = 16.0000 MHz			
	U2X = 0		**U2X = 1**	
Baud Rate (bps)	**UBRR**	**Error**	**UBRR**	**Error**
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	-3.5%	16	2.1%
230.4k	3	8.5%	8	-3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max [(1)]	1 Mbps		2 Mbps	

1. UBRR = 0, Error = 0.0%