National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

MPS PROJECT

GPS INTEGRATION FOR NUST SHUTTLE SERVICE

GROUP MEMBERS

1) Syed Muhammad Abubakar	337385
2) Abdur Rehman	337668
3) Asad Ahmed	331881

DEDICATIONS

This project is dedicated to our parents,
for their constant support, love and encouragement,
our ambition to eventually become an efficient Electrical Engineer in the future
and bring improvement to the Nust Shuttle Service by this project.
We would also like to dedicate this project to our instructors Sir Arbab Latif and Sir Abrar-ul-Haq.

ACKNOWLEGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful,
Alhamdulillah, all praises to Allah for the strength He has given and His blessing in completing this Project.
We would also like to thank the seniors and our classmates who guided us through the course of this project and gave us the motivation to complete this journey.

ABSTRACT

The shuttle service that operates in NUST consists of 7-8 shuttle buses roaming around NUST (approx. 2km x 2km) from 9-5 with no specific route and no tracking system. Which means that no shuttle is kept under observation while in operation that makes it very obstruse for the students to use. GPS integration is important for effective fleet management. Without GPS integration we cannot keep the drivers accountable and emergency handling becomes a big challenge. By integrating GPS we can monitor the current vehicle's address, it's speed and direction. There can be immediate alerts for emergencies and hourly/daily reports. Most importantly, by the integration drivers are aware of the monitoring, and managers know exactly where each vehicle is. GPS tracking also allows you to identify problems before they occur, saving time and money for the company.

Furthermore, the students will be able to see the location of the shuttles and the nearest shuttle available for their planned destination. They will also be able to see the estimated arrival time of the shuttles so that they don't miss it in any case and can be ready on time.

TABLE OF CONTENTS

Chapter 1: Introduction

1.1 Overview of Project

This project can be divided into 3 modules:

1) **GPS coordinates transmission**: This module is the handling of the location string from the GPS modem and transmitting it to the microcontroller. The string format that the GPS modem uses is NMEA and it is transmitted as is.

2) **Calculations on the coordinates:** Next, the GPS coordinates are extracted from the string of NMEA format and they are transmitted onto the mobile application using Bluetooth module.

3) **Receiving and displaying the coordinates**: The GPS coordinates are received by an android application and the real-time change in position with the current latitude and longitude is displayed.

We planned on using the GSM module for the communication between the microcontroller and mobile, the android application for this task was successfully built but unfortunately the SIM could not be configured so we used the Bluetooth module.
We also planned on using ATmega32 as the microcontroller and also made the program for the task but we could not find a programmer and our alternative method of programming the chip proved unsuccessful due to which we are now using Arduino microcontroller.

1.2 Block Diagram of Complete System

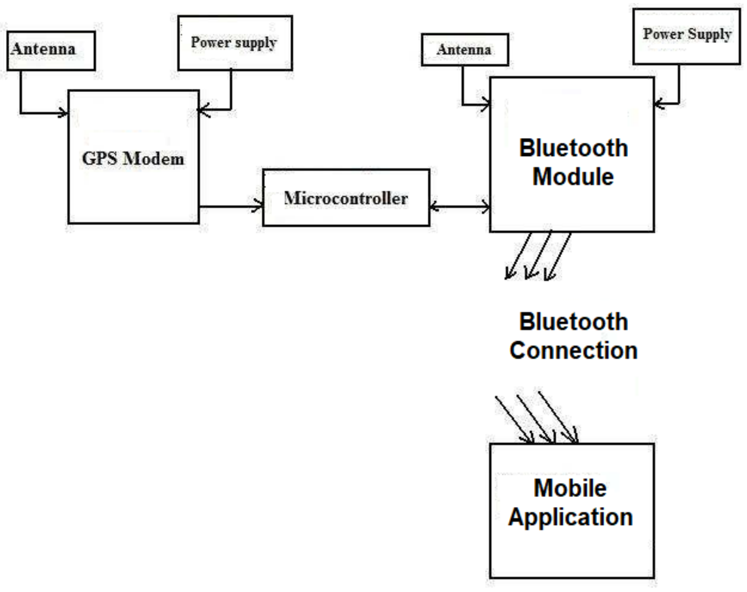


Figure 1: Block Diagram

1.3 Clear Work Division

Abdur Rehman:

Worked on the transmission of GPS coordinates from GPS modem to the microcontroller and the display of coordinates to LCD.

Asad Ahmed:

Worked on the transmission from microcontroller to the mobile application via Bluetooth (AND GSM module which was not successful).

Syed Muhammad Abubakar:

Worked on receiving the coordinates from the microcontroller to the mobile application (created a custom application for the task). Also helped on the extraction of coordinates from the NMEA string.

1.4: Literature Review:

An article was used as a reference for our project. In this article, gsm and gps module had been interfaced with ATmega microcontroller. Tracking has now been a recent trend followed everywhere. This process helps us to collect details and at the same time prevent robbery of devices being tracked. The project 'GPS and GSM based Vehicle Tracking System' which employs a microcontroller as its chief component is mostly implemented to keep track of vehicles in recent times. The 'GPS and GSM based Vehicle Tracking System' project utilizes a GSM modem as a replacement for one of the GPS devices to ensure a two-way communication process. The GSM modem and SIM card combination employ the same technique as a normal cell phone to implement the tracking process. Link to article:

https://www.electronicwings.com/avr-atmega/gps-module-interfacing-with-atmega1632

Chapter 2: Design Methodology

2.1 Problem Statement

The Shuttle Buses in NUST have no kind of navigation or tracking system integrated on them which leaves it prone to security and safety threats. Seeing this, we made this project to improve the shuttle service and make it safer and more convenient to use. By our model, we can track the bus where ever it is on the campus and so it can be monitored easily for quick response to emergency situations. Furthermore, the students can also know about the current location of the shuttle bus and will be able to plan their trip accordingly instead of waiting in the scorching heat to find a bus going in the opposite direction.

2.2 Code

FOR ATMEGA32:

```
/*
   NUST Shuttle Tracking System                 MPS Project
       Group Members:
                     1.     Abdur Rehman
                     2.     Syed Muhammad Abubakar
                     3.     Asad Ahmed
*/

#define F_CPU 1000000UL                  /* Define CPU Frequency e.g. here 8MHz */
#include <avr/io.h>               /* Include AVR std. library file */
#include <util/delay.h>                  /* Include Delay header file */
#include <avr\interrupt.h>

#define LCD_Dir  DDRB                    /* Define LCD data port direction */
#define LCD_Port PORTB                   /* Define LCD data port */
#define RS PB0                           /* Define Register Select pin */
#define EN PB1                           /* Define Enable signal pin */

unsigned char Lat[10];     //array of characters to store the Latitude     =
{'3','3','4','0','.','0','6','4','1','6'}
unsigned char Long[11];    //array of characters to store the Longitude =
{'0','7','3','0','6','.','4','0','3','8','8'}
unsigned char Latitude_in_deg[11]; //array of characters to store the Latitude
unsigned char Longitude_in_deg[11];      //array of characters to store the Longitude
unsigned char* Lat_deg = &Latitude_in_deg[0];
unsigned char* Long_deg = &Longitude_in_deg[0];
unsigned char Data[21];
unsigned char* d = &Data[0];

unsigned char received_char = 0;
int Latitude_Index = 0;
int Longitude_Index = 0;
int CommaCounter = 0;
bool IsItGGAString = false;
```

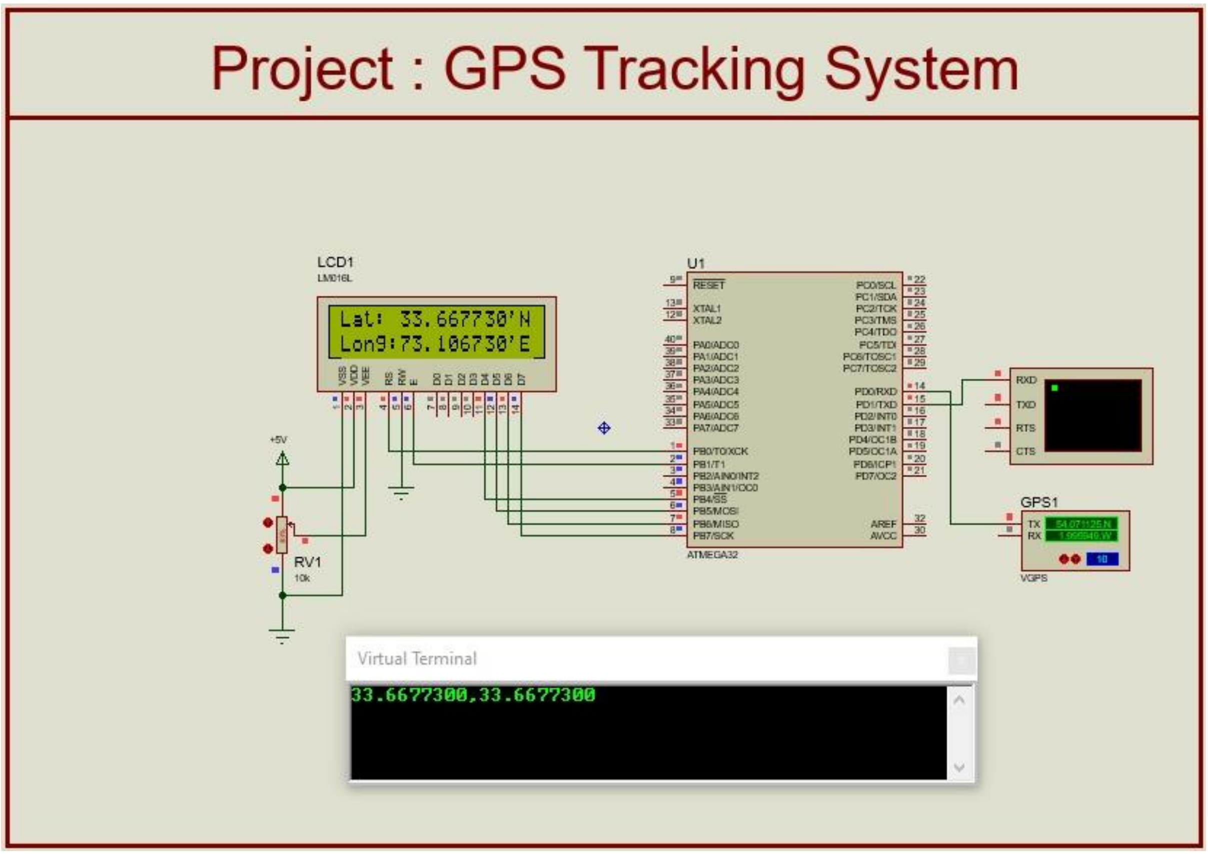
```
unsigned char GGA_CODE[3];        //array of characters to identify the GGA string


void LCD_Command( unsigned char cmnd )
{
	LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0); /* sending upper nibble */
	LCD_Port &= ~ (1<<RS);		/* RS=0, command reg. */
	LCD_Port |= (1<<EN);	/* Enable pulse */
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);

	_delay_us(200);

	LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);  /* sending lower nibble */
	LCD_Port |= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);
	_delay_ms(2);
}

void LCD_Char( unsigned char data )
{
	LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0); /* sending upper nibble */
	LCD_Port |= (1<<RS);	/* RS=1, data reg. */
	LCD_Port|= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);

	_delay_us(200);

	LCD_Port = (LCD_Port & 0x0F) | (data << 4); /* sending lower nibble */
	LCD_Port |= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);
	_delay_ms(2);
}

void LCD_Init (void)			/* LCD Initialize function */
{
	LCD_Dir = 0xFF;			/* Make LCD port direction as o/p */
	_delay_ms(20);			/* LCD Power ON delay always >15ms */

	LCD_Command(0x02);		/* send for 4 bit initialization of LCD  */
	LCD_Command(0x28);              /* 2 line, 5*7 matrix in 4-bit mode */
	LCD_Command(0x0c);              /* Display on cursor off*/
	LCD_Command(0x06);              /* Increment cursor (shift cursor to right)*/
	LCD_Command(0x01);              /* Clear display screen*/
	_delay_ms(2);
}

void LCD_String (char *str)		/* Send string to LCD function */
{
	int i;
	for(i=0;str[i]!=0;i++)		/* Send each char of string till the NULL */
	{
		LCD_Char (str[i]);
	}
}

void LCD_String (unsigned char *str)		/* Send string to LCD function */
{
	int i;
	for(i=0;str[i]!=0;i++)		/* Send each char of string till the NULL */
	{
		LCD_Char (str[i]);
	}
}

void LCD_String_xy (char row, char pos, char *str)	/* Send string to LCD with xy
position */
{
	if (row == 0 && pos<16)
	LCD_Command((pos & 0x0F)|0x80);	/* Command of first row and required
position<16 */
	else if (row == 1 && pos<16)
	LCD_Command((pos & 0x0F)|0xC0);	/* Command of first row and required
position<16 */
	LCD_String(str);		/* Call LCD string function */
}

void LCD_Clear()
{
	LCD_Command (0x01);		/* Clear display */
	_delay_ms(2);
	LCD_Command (0x80);		/* Cursor at home position */
}

int to_int (char count){
	if (count=='0') return 0;
	else if (count=='1') return 1;
	else if (count=='2') return 2;
	else if (count=='3') return 3;
	else if (count=='4') return 4;
	else if (count=='5') return 5;
	else if (count=='6') return 6;
	else if (count=='7') return 7;
	else if (count=='8') return 8;
	else if (count=='9') return 9;
	else return 0;
}

void lat_to_deg(){
	long long int lat_int = 0;
	for (int i = 0;i < 10;i++){
		if (i != 4){
			lat_int = (lat_int*10) + to_int(Lat[i]);
		}
		else{
```

```
                //do nothing
            }
        }
        int deg = lat_int/10000000;
        long int min = (lat_int - (deg*10000000));
        min = min / 60;
        //putting the value into an array of string
        int temp = 0;
        for(int i = 0; i < 10;i++){
            if(i < 2){
                deg = deg*10;
                temp = deg/100;
                deg = deg % 100;
                Latitude_in_deg[i] = '0' + temp;
            }
            else if(i == 2){
                Latitude_in_deg[i] = '.';
            }
            else{
                min = min*10;
                temp = min/100000;
                min = min%100000;
                Latitude_in_deg[i] = '0' + temp;
            }
        }
}

void long_to_deg(){
        long long int long_int = 0;
        for (int i = 0;i < 11;i++){
            if (i != 5){
                long_int = (long_int*10) + to_int(Long[i]);
            }
            else{
                //do nothing
            }
        }
        int deg = long_int/10000000;
        long int min = (long_int - (deg*10000000));
        min = min / 60;
        //putting final value into array
        int temp = 0;
        for(int i = 0; i < 10;i++){
            if(i < 2){
                deg = deg*10;
                temp = deg/100;
                deg = deg % 100;
                Longitude_in_deg[i] = '0' + temp;
            }
            else if(i == 2){
                Longitude_in_deg[i] = '.';
            }
            else{
                min = min*10;
```

```
                        temp = min/100000;
                        min = min%100000;
                        Longitude_in_deg[i] = '0' + temp;
                }
        }
}

void Display_on_LCD()
{
        LCD_Clear();
        LCD_String("Lat: "); // Write string on 1st line of LCD
        LCD_String(Lat_deg);
        LCD_String_xy (0, 14,"'N");
        LCD_Command(0xC0);          //Go to 2nd line
        LCD_String("Long:"); //Write string on 2nd line
        LCD_String(Long_deg);
        LCD_String_xy (1, 14,"'E");
}

void Convert_to_Deg()
{
        lat_to_deg();
        long_to_deg();
}

void Make_Data_String()
{
        for(int i = 0; i < 10;i++){
                Data[i] = Latitude_in_deg[i];
        }
        Data[10] = ',';
        for(int i = 11; i < 21; i++){
                Data[i] = Longitude_in_deg[i];
        }
}

void USARTInit()
{
        UBRRL = 0xC;
        UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);
        UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
}

void USARTWriteChar(char data)
{
        while(!(UCSRA & (1<<UDRE)));
        UDR=data;
}

void USART_StringTransmit(unsigned char* s)
{
        for(int i = 0; i < 21 ; i++){
                USARTWriteChar(s[i]);
        }
```

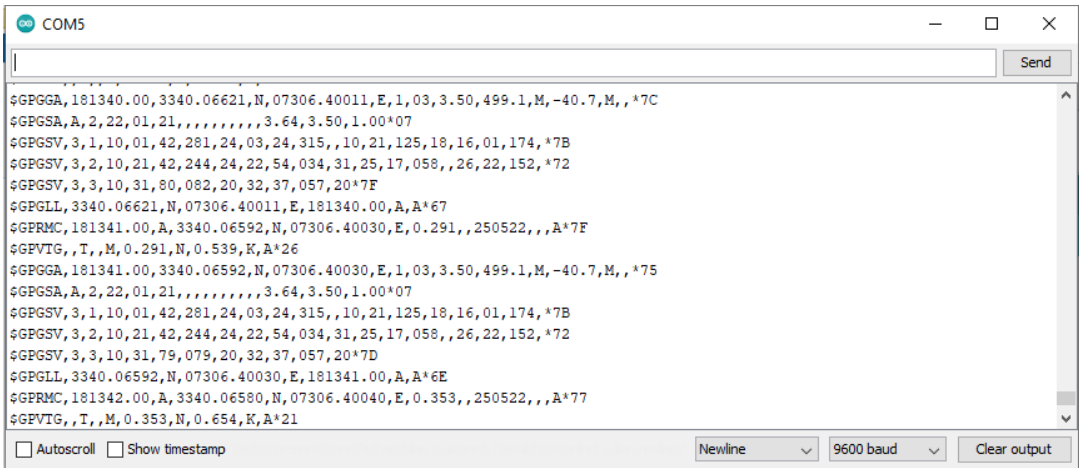
```
}

ISR(USART_RXC_vect)
{
	char received_char = UDR;

	if(received_char =='$'){			// check for '$'
		Latitude_Index = 0;
		Longitude_Index = 0;
		CommaCounter = 0;
		IsItGGAString = false;
	}
	else if(IsItGGAString == true){		// if true save GGA info. into buffer
		if(received_char == ',' ){
			CommaCounter++;	// increment comma counter
		}
		if(CommaCounter == 2){
			Lat[Latitude_Index++] = received_char;
		}
		if(CommaCounter == 4){
			Long[Longitude_Index++]= received_char;
		}
	}
	else if(GGA_CODE[0] == 'G' && GGA_CODE[1] == 'G' && GGA_CODE[2] == 'A'){	//
check for GGA string
		IsItGGAString = true;
		GGA_CODE[0] = 0; GGA_CODE[1] = 0; GGA_CODE[2] = 0;
	}
	else{
		GGA_CODE[0] = GGA_CODE[1];  GGA_CODE[1] = GGA_CODE[2]; GGA_CODE[2] =
received_char;
	}
}

int main()
{
	LCD_Init();			//Initialization of LCD
	_delay_ms(4000);	//wait for GPS Initialization
	USARTInit();
	sei();

	while(1){
		Convert_to_Deg();
		Display_on_LCD();
		Make_Data_String();
		//USART_StringTransmit(d);
		_delay_ms(1000);
	}
}
```

2.3 Simulation



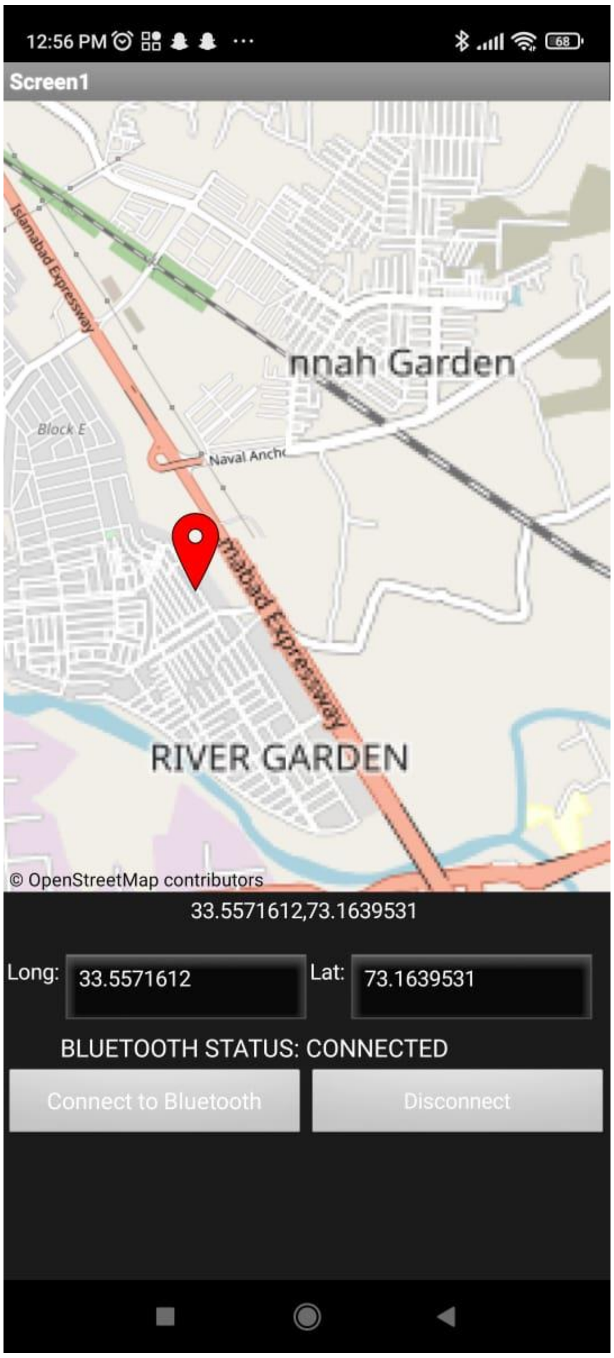
```
33.6677300,33.6677300
```

2.4 GPS input string:



```
$GPGGA,181340.00,3340.06621,N,07306.40011,E,1,03,3.50,499.1,M,-40.7,M,,*7C
$GPGSA,A,2,22,01,21,,,,,,,,,,3.64,3.50,1.00*07
$GPGSV,3,1,10,01,42,281,24,03,24,315,,10,21,125,18,16,01,174,*7B
$GPGSV,3,2,10,21,42,244,24,22,54,034,31,25,17,058,,26,22,152,*72
$GPGSV,3,3,10,31,80,082,20,32,37,057,20*7F
$GPGLL,3340.06621,N,07306.40011,E,181340.00,A,A*67
$GPRMC,181341.00,A,3340.06592,N,07306.40030,E,0.291,,250522,,,A*7F
$GPVTG,,T,,M,0.291,N,0.539,K,A*26
$GPGGA,181341.00,3340.06592,N,07306.40030,E,1,03,3.50,499.1,M,-40.7,M,,*75
$GPGSA,A,2,22,01,21,,,,,,,,,,3.64,3.50,1.00*07
$GPGSV,3,1,10,01,42,281,24,03,24,315,,10,21,125,18,16,01,174,*7B
$GPGSV,3,2,10,21,42,244,24,22,54,034,31,25,17,058,,26,22,152,*72
$GPGSV,3,3,10,31,79,079,20,32,37,057,20*7D
$GPGLL,3340.06592,N,07306.40030,E,181341.00,A,A*6E
$GPRMC,181342.00,A,3340.06580,N,07306.40040,E,0.353,,250522,,,A*77
$GPVTG,,T,,M,0.353,N,0.654,K,A*21
```

2.5 Android Application



33.5571612,73.1639531

Long: 33.5571612 Lat: 73.1639531

BLUETOOTH STATUS: CONNECTED

Chapter 3: Hardware Implementation

3.1 Detail of Sub Modules:

- HC-05 Bluetooth Module
- 16x2 LCD module
- ATmega32A
- NEO 6M GPS Module
- Jumper wires
- Lithium ion batteries
- Breadboard power supply
- Breadboards

NEO 6M GPS Module:

The NEO-6M GPS module is a well-performing complete GPS receiver with a built-in 25 x 25 x 4mm ceramic antenna, which provides a strong satellite search capability. With the power and signal indicators, you can monitor the status of the module.

Features:
1) A complete GPS module with an active antenna integrated, and a built-in EEPROM to save configuration parameter data.
2) Built-in 25 x 25 x 4mm ceramic active antenna provides strong satellite search capability.
3) Equipped with power and signal indicator lights and data backup battery.
4) Power supply: 3-5V; Default baud rate: 9600bps.
5) Interface: RS232 TTL

<u>In our project the GPS module communicated using USART with the ATmega32A. It's baud rate was changed to 4800bps.</u>

16x2 LCD module:

LCDs (Liquid Crystal Displays) are used for displaying status or parameters in embedded systems.

LCD 16x2 is a 16 pin device which has 8 data pins (D0-D7) and 3 control pins (RS, RW, EN). The remaining 5 pins are for supply and backlight for the LCD.

<u>In our project this was used to display the coordinates directly from the ATmega32A</u>

ATmega32A:
The ATmega32A is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega32Aachieves throughputs approaching 1 MIPS per MHz allowing the system designed to optimize power consumption versus processing speed.

<u>In our project this was used to receive NAEMA strings from GPS, Extract the coordinates, decode them, send them to LCD, Make string, and Send it to Bluetooth module.</u>

HC-05 Bluetooth Module:
HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC. HC-05 Bluetooth module provides switching mode between master and slave mode which means it able to use neither receiving nor transmitting data. Any serial data from 9600 to 115200 bps can be communicated with the device.

<u>In our project, it was used to transmit GPS coordinates received serially from the ATMega32A to the mobile application via Bluetooth communication.</u>

Android Mobile Application:

3.2 Issues / Results/ Observations

During our project we encountered several issues, some were fixed whereas for some we relied on alternative options. Some of these issues were:

- The GSM module: the sim was not catching network signals and not working even after PTA verification. This was a major setback for us since our goal was to use SMS communication. We were unsuccessful in fixing the error and making the SIM work so we chose to use the Bluetooth communication.
- Android Application: the application was working perfectly for the SMS communication but when it was modified for Bluetooth communication, the real-time receiving of coordinates gave some unexpected error which was fixed after multiple attempts.
- GPS coordinates: the GPS modem provided its output in NMEA string but we required the coordinates in latitude and longitude. Although we successfully converted the NMEA to our desired format later, but it proved to be a hurdle in the progression of our project.
- The application we developed could only be downloaded on android mobiles which proved to be a big hurdle in debugging since the developer of the app owned an ios device.

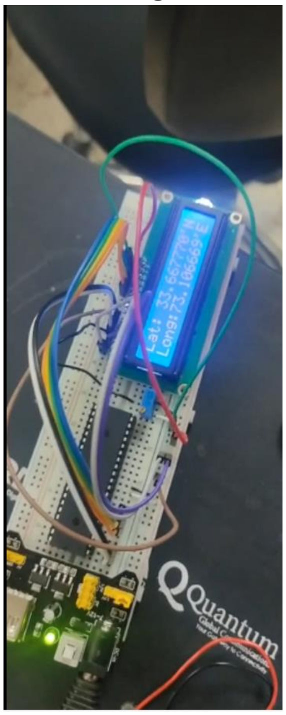
While coding was not so cumbersome, same cannot be said for hardware implementation. The most difficult task was to burn the code. Due to unavailability of Super Max Pro chip burner, we had to burn our ATmega32 using Arduino uno. We were unable to download Arduino code on the Uno board. This was done several times however after some successful attempts the code started behaving irrationally and no longer burned the ATmega32. So we had to switch to Arduino UNO for our hardware implementation.

Final Results:

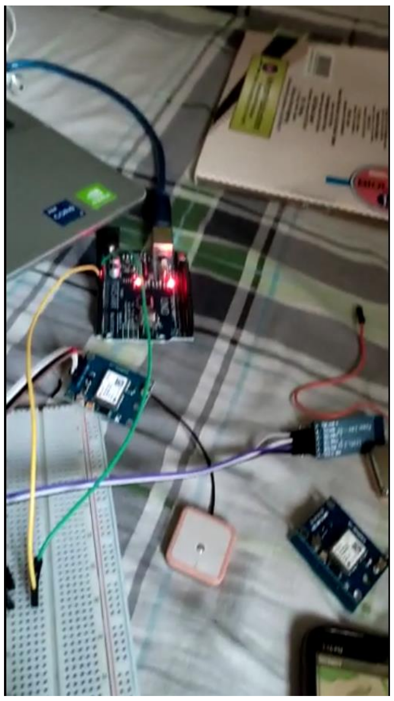
Even with all these setbacks we were able to bring our project to working condition. During the demonstration only the code for GPS was not burnt into the ATmega32A due to load shedding. But we demonstrated that is the array of coordinates is hardcoded into the ATmega32A our system is able to decode them, make them into a string and send the string via Bluetooth to the Android phone where the app show the location on pointer.

3.3 Hardware Pictures:

LCD working with ATmega32:



Complete project:



Video Links:

Working of LCD:

https://drive.google.com/file/d/1ca9AgZfVinSSTmlmoZDdLJWDLuE5OaIr/view?usp=sharing

Working Project with Arduino UNO:

https://drive.google.com/file/d/1TbVYuuiD0u6UGnZjbnFVyQ1sC7KkJB2E/view?usp=sharing

Chapter 4: Conclusions

4.1: Project Applications

1) **NUST Shuttle Service**
   NUST require maximum security, especially in the light of recent events occurring in Pakistan. For this, our GPS integration will be extremely useful since it would give us the location of the shuttles inside NUST and report suspicious behavior.

2) **Careem**
   Careem is currently using android to android GPS communication which is prone to hacking and most control is with the driver and so Careem can use our module for GPS tracking to increase safety.

3) **Bykea**

4) **Swvl**

5) **Surveillance**
   Parents can use our module to keep track of their children's vehicles and get notified if the vehicle leaves a certain premises so as to know where and when their child is at all times.

4.2: Future Recommendations

- First improvement would be to debug the GPS module and test it. The code is written.
- Then to integrate the GSM module which we were unsuccessful in doing. The GSM module would increase the range of the communication between the device and android application by several folds and we would be able to remotely view the location of our device.
- The mobile application can be upgraded to view the speed of the vehicle and the path to destination. (This can be done by developing an application on paid platforms, since the calculations are complex)