EE318 Electronic Design Lab, Project Report, EE Dept, IIT Bombay, April 2009

GPS Tracker

Group No: B11

B.V. Sesha Pavan Srinadh (06007038) <bsrinadh@iitb.ac.in>

Mayank Manjrekar (06007036) <mayankm@iitb.ac.in>

Koneru Vivek Shishir (06007033) <k.vivekshishir@iitb.ac.in>

Guide: Prof. Girish Kumar

ABSTRACT

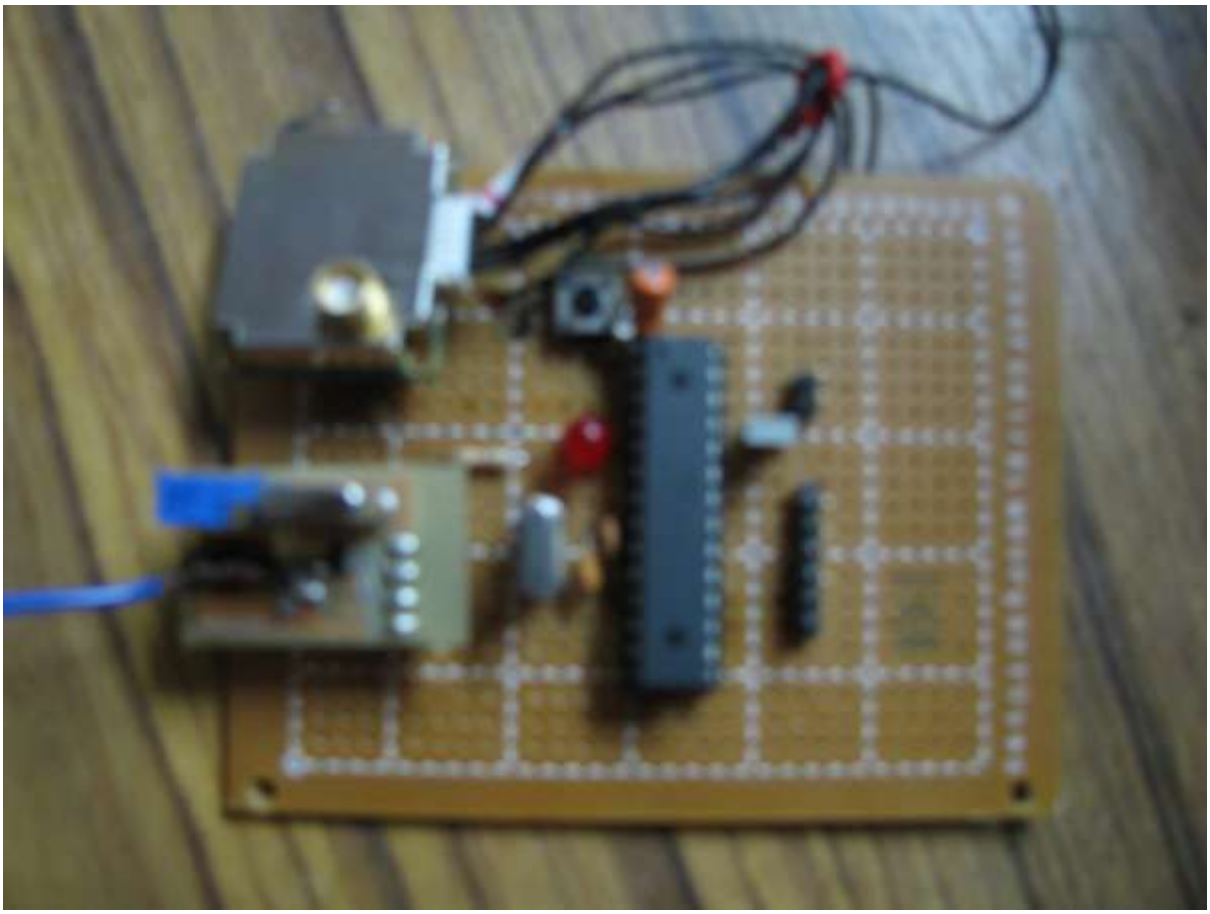
The project is aimed to integrate a GPS module with a Transmitter/Receiver module and display the coordinates on a LCD for tracking applications. The coordinates of a location is got using a GPS module. The data from the GPS module is sent over an RF channel to the receiver module situated at the tracking centre. At the tracking centre the data received is processed and displayed on an LCD screen. We have successfully implemented the idea. The technical details of this project follow later.

CONTENTS

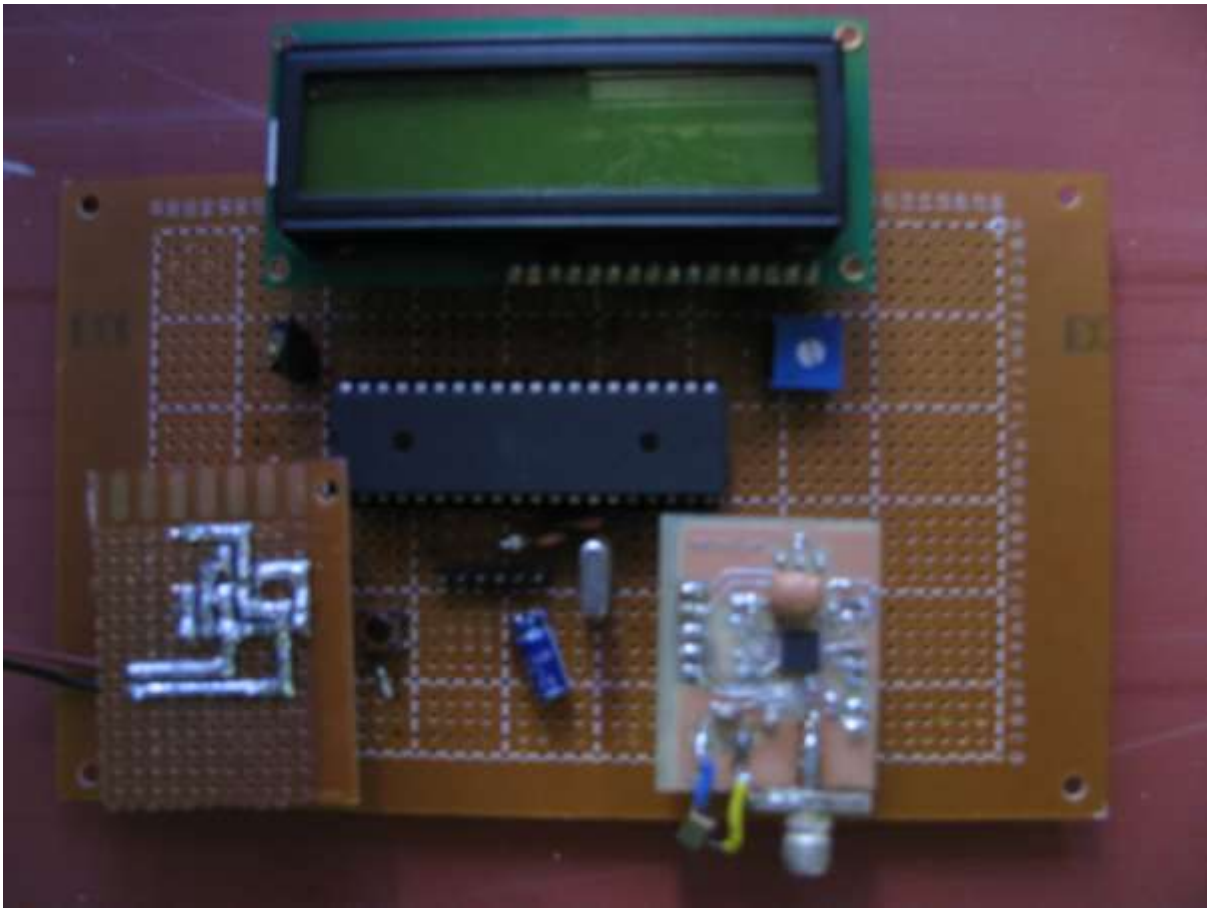
1. INTRODUCTION

The GPS tracker project consists of two modules: Transmitter module and Receiver module. The transmitter module consists of a GPS module (iWave) integrated to a MAX1472 transmitter through Atmega8l microcontroller. The receiver module consists of a MAX1473 receiver integrated to an LCD through Atmega16. The transmitter module receives the coordinates from the GPS module and transmits the data at 433 MHz. The receiver module receives this data and the coordinates are displayed on the LCD screen.

Following are the photographs of the two modules:

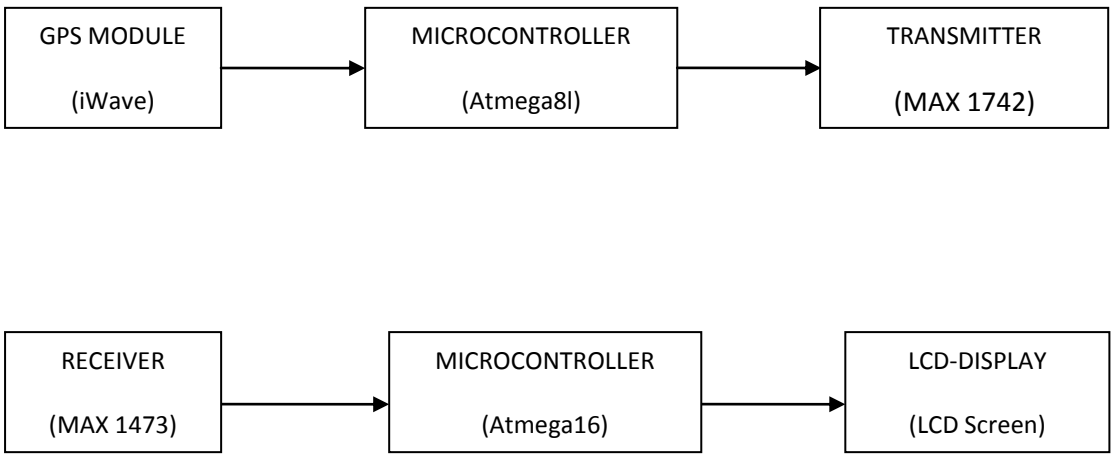


Transmitter Module



Receiver Module

2. BLOCK DIAGRAM



3. MAIN COMPONENTS

1. Transmitter Module:

- **GPS Module:** We are using an iWave GPS Module (iW-PRDLT-UM-01) to get the required coordinates. It uses NMEA protocol. We are using this module because of its reliability, speed and operating voltage. It has a very fast start up time of around 45 seconds and has an operating voltage of 3.3V which is compatible with our transmitter module [1].

- **Atmega8L:** The Atmega8 is a high performance low power AVR 8 bit microcontroller with the following features [2]:

  1. Advanced RISC architecture with 130 powerful instructions and 32 x 8 general purpose working registers.
  2. 8K bytes of in system self programmable flash memory, 512 bytes EEPROM, 1KB internal SRAM.
  3. Programmable serial USART.
  4. Low operating voltages: 2.7-5.5V (Atmega8L).

  The GPS module runs on a voltage range of 2.7 to 3.6 V, hence to avoid stepping down the voltage within the circuit we used the low voltage Atmega8L so that both components can be directly supplied by a single voltage source. As the device is portable power has to be conserved and hence Atmega8L is used.

- **MAX 1472:** The MAX1472 is a crystal-referenced PLL based ASK (Amplitude Shift Keying) transmitter module with the following features [4]:

  1. Has two operating frequencies, 315MHz and 433MHz.We are operating at 433 MHz.
  2. 2.1V to 3.6V Single-Supply Operation.

  The circuit used for the transmitter module is:

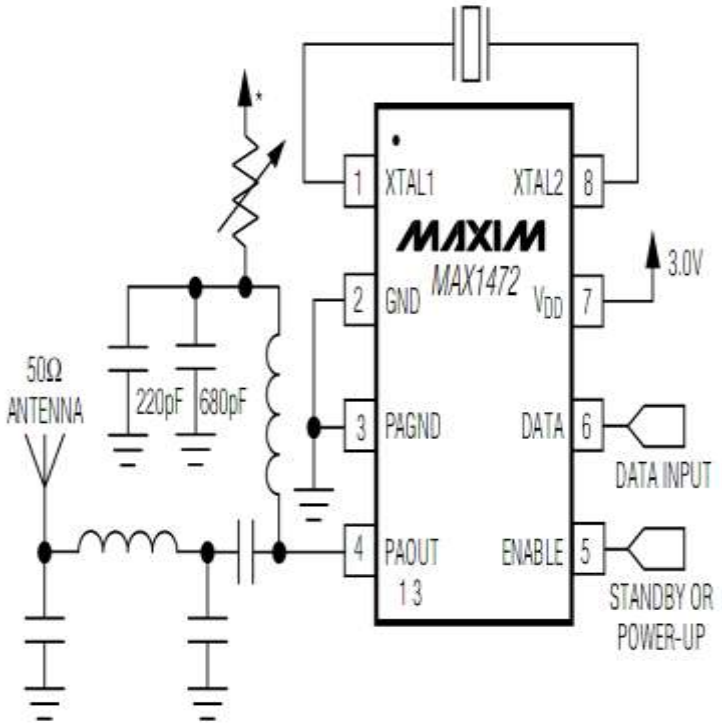


Figure 1: Circuit used in transmitter module

2. Receiver Module:

- **MAX 1473:** The MAX 1473 is a low-power CMOS superheterodyne receiver for receiving ASK (Amplitude Shift Keying) data, with the following features[5]:

1. Has two operating frequencies, 315MHz and 433MHz.We are operating at 433 MHz.
2. Operates from Single 3.3V or 5.0V Supplies
3. Selectable Image-Rejection Centre Frequency
4. The chip consists of a low-noise amplifier (LNA), a fully differential image-rejection mixer, an on chip phase-locked-loop (PLL) with integrated voltage controlled oscillator (VCO).

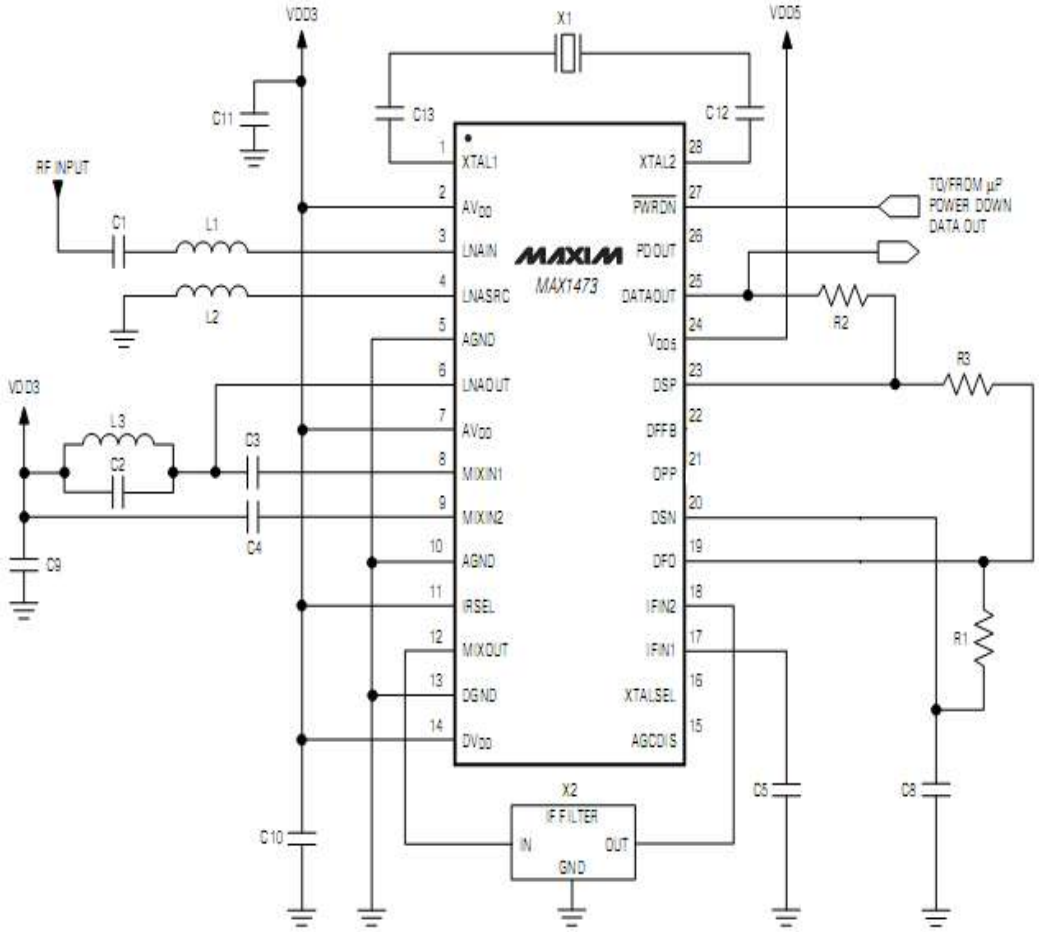


Figure 2: Circuit used in receiver module

➢ **Atmega16:** The Atmega16 is a high performance low power AVR 8 bit microcontroller with the following features [3]:

1. Advanced RISC architecture with 130 powerful instructions and 32 x 8 general purpose working registers.
2. 16K bytes of in system self programmable flash memory, 512 bytes EEPROM, 1KB internal SRAM.
3. Programmable serial USART.
4. Operating voltage of 4.5-5.5V.

➢ **LCD:**

4. HARDWARE DESIGN

The schematic for the transmitter module PCB is given below:

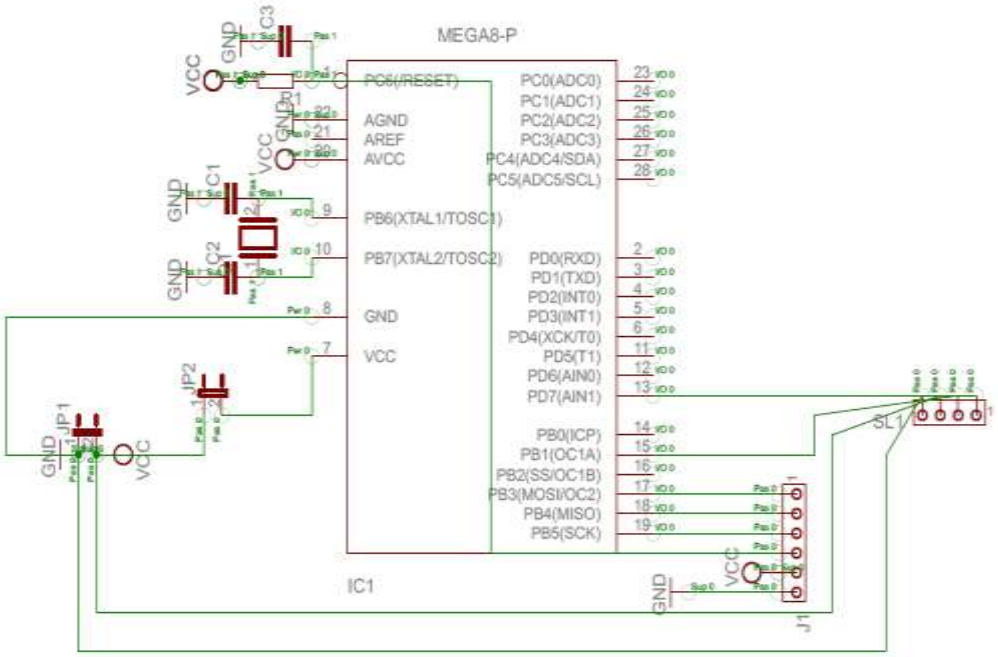


Figure 3: Transmitter Module Schematic

The schematic for the receiver module PCB is given below:

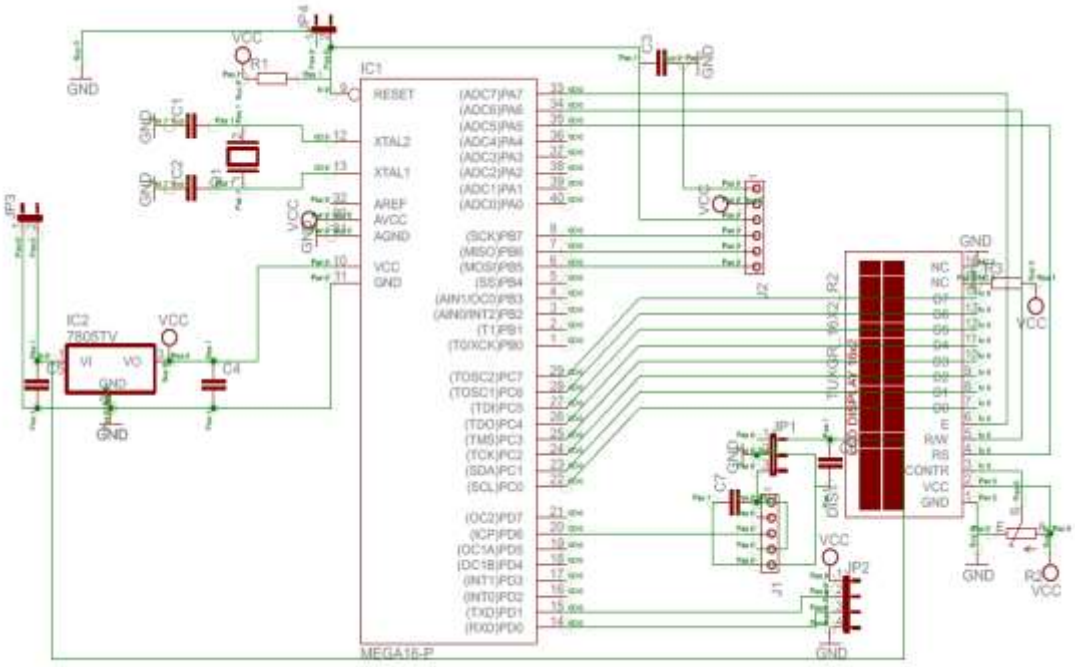


Figure 4: Receiver Module Schematic

5. SOFTWARE DESIGN

1. The first part of software design is responsible for taking in the data from the GPS module and storing only the required data and removing the unwanted data. The following is an example of the type of data received by the microcontroller from the GPS module.

```
$GPGGA,063608.720,1907.8976,N,07255.0095,E,1,03,5.1,62.8,M,-62.8,M,,0000*4E<CR><LF>
$GPGSA,A,2,22,18,19,,,,,,,,,,6.0,5.1,3.2*31<CR><LF>
$GPGSV,3,1,10,22,46,033,33,18,30,069,46,19,21,281,35,14,57,347,32*7F<CR><LF>
$GPGSV,3,2,10,31,45,179,31,21,24,142,27,06,22,207,,03,21,249,26*78<CR><LF>
$GPGSV,3,3,10,30,08,109,,09,01,036,*76<CR><LF>
$GPRMC,063608.720,A,1907.8976,N,07255.0095,E,0.06,204.83,280409,,,A*6A<CR><LF>
$GPVTG,204.83,T,,M,0.06,N,0.1,K,N*08<CR><LF>
```

   The microcontroller checks for the string with “GPGGA” and stores the whole string which contains “GPGGA”.

2. The next part of the software design deals with Manchester coding the string received from the GPS module. Manchester coding uses the transitions of the signal as bits as shown in Figure 3.

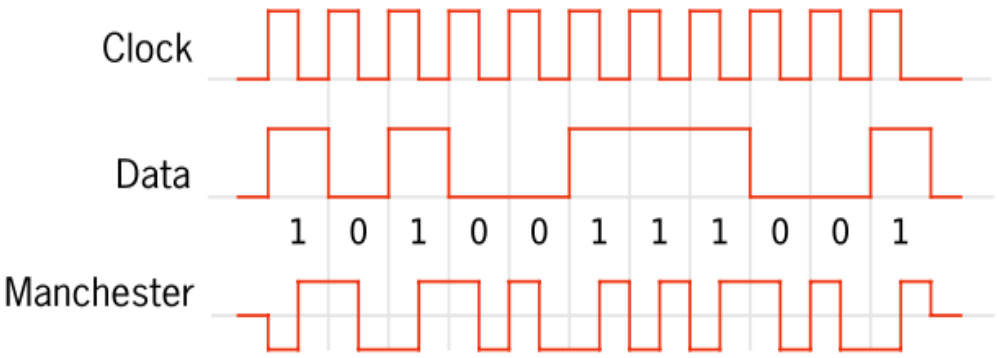


Figure 3: Manchester Coding

After the string is Manchester coded it is sent to the transmitter which transmits the data at 433 MHz.

3. Now the microcontroller at the receiver end has to take the data from the receiver and decode it.

4. After the received data is decoded then the microcontroller collects the required data (Time, Latitude and Longitude) from the string and then displays it on an LCD.

6. TESTING PROCEDURE

- **GPS Module:** The GPS module was tested by connecting it to the computer through a MAX232 circuit. The output was seen through HyperTerminal and the coordinates were verified.

- **Microcontroller:** We tested the microcontroller's USART by using the code given at the end of the report and checking the output on a computer through HyperTerminal. The experiment was successful implying that the USART code of the microcontroller was working correctly.

- **MAX1472 (Transmitter):** After soldering all the components onto the PCB we tested the transmitter module using a spectrum analyser. The spectrum analyser gave a distinct peak at 433 MHz, which is the operating frequency of the transmitter. This indicates that the transmitter is working properly.

- **MAX1473 (Receiver):** After soldering all the components onto the PCB we tested the receiver module by sending a 433 MHz signal using a signal generator and checking the output on a spectrum analyser. We initially had lots of problem while testing the receiver module. Therefore we tested the module stage by stage and corrected the mistakes. After making all the changes we tested the final output using a spectrum analyser and saw a distinct peak at 433 MHz, indicating that the receiver module is working correctly.

- **Manchester Coding:** We tested our Manchester coding by giving an input and checking the correctness of the output on an oscilloscope. We sent many different inputs to confirm that Manchester coding was working. After this we wrote the code for decoding the Manchester coding and tested it by checking that the input to Manchester coding and output after decoding are the same.

- **Transmitter-Receiver Pair:** We tested whether the transmitter and receiver pair was working by transmitting a simple square wave through the transmitter and seeing the output at the receiver end. The output was matched to the input thereby confirming that the transmitter-receiver pair is working.

- **LCD:** The LCD was tested by using a simple hello code (displaying hello on the LCD).

7. DAY TO DAY PROGRESS

26-01-09 Day 1: We read about different types of GPS modules and receivers and transmitters. Finally we decide to use CW20 GPS module and MAX 1472/73 receiver and transmitter respectively due to their simplicity.

27-01-09 Day 2: We procured the required microcontrollers and the transmitter and receiver IC's.

02-02-09 Day 3: We got the PCB's for the transmitter and receiver and also procured a GPS module.

03-02-09 Day 4: We got the GPS antenna and started reading up on the GPS module, transmitter and receiver.

09-02-09 Day 5: We tested the USART of both the microcontrollers. Both were working fine.

10-02-09 Day 6: We tested GPS module in the lab. It was not giving any data as the antenna was not receiving any data inside the lab. So we tested the GPS module at home. It was giving the required data.

24-02-09 Day 7: We soldered the transmitter IC on the PCB and tested it using a Spectrum analyser. There was a distinct peak at 433 MHz, which is the required output.

02-03-09 Day 8: We soldered the receiver IC on the PCB and tested it by sending a 433 MHz signal using a signal generator and seeing the output on a spectrum analyser. The required output was not got, so we started testing the receiver stage by stage.

03-03-09 Day 9: We tested the receiver stage by stage and found the problem. We tried many methods to correct it but were unsuccessful.

09-03-09 Day 10: As we were unable to correct our problem in our receiver we got the receiver of a previous batch and have tested it and compared it with our receiver. Our receiver was still not working so we decide to make another PCB and solder another IC, as the IC looks to be faulty.

16-03-09 Day 11: We have got the PCB for a USBASP. We have soldered the components on the PCB. We also tested the GPS module in the lab. Coding for the transmitter microcontroller interface was also started.

17-03-09 Day 12: We procured a new receiver PCB and soldered all the components onto the PCB.

23-03-09 Day 13: We tested the new receiver stage by stage using a spectrum analyser and an oscilloscope. The receiver testing was successful.

24-03-09 Day 14: We read up on Manchester coding and finished code for Manchester coding. We also tested the Manchester coding giving sample inputs and checking the output on an oscilloscope.

30-03-09 Day 15: We started and completed the transmitter side coding, i.e. receiving data from GPS module and Manchester coding the data and sending the data.

31-03-09 Day 16: While testing the transmitter code the GPS module stopped working, so we tried to find out what was wrong with it, but we were unsuccessful.

06-04-09 Day 17: As the GPS module was not working we ordered a new GPS module from iWave Systems Technologies Pvt. Ltd... We also started code for our receiver.

07-04-09 Day 18: We completed the code for Manchester decoding and tested this code and our testing was successful. We then completed the rest of the receiver code.

20-04-09 Day 19: We tested the new GPS module and integrated it with our transmitter module. We tested this setup by seeing the data on computer using HyperTerminal.

21-04-09 Day 20: We tried integrating both the transmitter and receiver modules (by direct connection, not through wireless) but had some problems. We tried debugging the code.

22-04-09 Day 21: We continued our debugging process by changing the transmitter and receiver codes one at a time.

23-04-09 Day 22: We finally got the transmitter and receiver modules to work together. We now connected the transmitter and receiver IC's to their respective modules.

24-04-09 Day 23: We tested our completed modules through wireless medium and successfully received the required data at the tracking centre.

25-04-09 Day 24: We wrote the code for LCD display and connected it to the receiver module and tested the entire setup.

26-04-09 Day 25: We completed the PCB designs for both the transmitter and receiver modules. We started soldering the components onto the PCB.

27-04-09 Day 26: We completed the PCB's and tested the modules and both were working correctly.

8. RESULTS

Both transmitter and receiver modules are made on a pcb and are working correctly. We were able to receive the coordinates sent by transmitter and display the coordinates on an LCD display.

9. APPLICATIONS

The product can be used for any general tracking applications like tracking a car or bus. The transmitter module can be placed with the object we need to track and receiver module can be placed at a tracking centre. The transmitter updates the tracking centre regularly about the location of the object.

10. IMPROVEMENTS SUGGESTED

The project can be improved if the data received at the receiver module can be interfaced to computer through USB interface and then plotted on a map, thereby showing the path taken by the object being tracked.

The range of transmitter and receiver pair can be increased by adding a power amplifier which is left as future work.

11. ACKNOWLEDGEMENTS

We would like to thank the instructors for their valuable guidelines and help. We are very grateful to Prof. Girish Kumar, Mr. Ravindra Kashyap (antenna lab) Mr. Piyush (antenna lab) for help in making and debugging the RF circuits. Also, we would like to thank Raghavendra Bhushan and other TAs for their help. The lab technicians were always very cooperative.

12. REFERENCES

[1] Datasheet of GPS module, http://www.iwavesystems.com/IPs/iW-GPS.pdf

[2] Datasheet of Atmega8L,
http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf

[3] Datasheet of Atmega16,
http://www.atmel.com/dyn/resources/prod_documents/doc2466.pdf

[4] Datasheet of MAX1472 transmitter,
http://datasheets.maximic.com/en/ds/MAX1472.pdf

[5] Datasheet of MAX1473 receiver,
http://datasheets.maximic.com/en/ds/MAX1473.pdf

[6] Datasheet of MAX232, http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf

13. SOURCE CODE ATTACHMENT

Source Code for Transmitter Module:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include<util/delay.h>
#include <string.h>
#define TIME 0xff
#define F_CPU 4000000// Clock Speed
#define BAUD 4800
#define MYUBRR (F_CPU/16/BAUD)-1
#define tbi(x,y) x ^=_BV(y)


void USART_Init( unsigned int ubrr)
{
        /* Set baud rate */
        UBRRH = (unsigned char)(ubrr>>8);
        UBRRL = (unsigned char)ubrr;

        /* Enable receiver and transmitter */
        UCSRB = (1<<RXEN)|(1<<TXEN);

        /* Set frame format: 8data, 1stop bit */
        UCSRC = (1<<URSEL)|(3<<UCSZ0);
} // end of USART_Init
```

```
void USART_Transmit( unsigned char data)
{
	/* Wait for empty transmit buffer */
	while ( !( UCSRA & (1<<UDRE)) )
	;
	/* Put data into buffer, sends the data */
	UDR = data;
}

unsigned char USART_Receive(void)
{
	/* Wait for data to be received */
	while (!(UCSRA & (1<<RXC)));
	/* Get and return received data from buffer */
	return UDR;
};

void receive(char* rec)
{		int i,len;
		rec[0]=0xaa;
	may:
	for(;USART_Receive()!='$';);
		rec[1]=USART_Receive();
	if(rec[1]!='G') goto may;
		rec[2]=USART_Receive();
	if(rec[2]!='P') goto may;
		rec[3]=USART_Receive();
	if(rec[3]!='G') goto may;
		rec[4]=USART_Receive();
	if(rec[4]!='G') goto may;
		rec[5]=USART_Receive();
	if(rec[5]!='A') goto may;
		rec[6]=USART_Receive();
	if(rec[6]!=',') goto may;

	for(i=7;rec[i-1]!=0x0d;i++) rec[i]=USART_Receive();
		rec[i]=USART_Receive();
	if(rec[i]!=0x0a) goto may;
	rec[i+1]='\x00';
	tbi(PORTD,PD2);
	/*len=strlen(rec);
	for(int k=0;k<20;k++)
 {
  for(int j=0;j<len;j++) USART_Transmit(rec[j]);
  _delay_ms(10);
 }*/
}
```

```
void init(void)
{	//DDRB=0;
	DDRB=0x01;
	PORTB=0x01;
	DDRD=_BV(PD2);
	USART_Init(MYUBRR);

}

int main(void)
{
	int len;
	char* hello;
	char* rec2="mayankfsf\x00";
	init();

	while(1)
	{
		receive(hello);

		len=strlen(hello);
		for(int k=0;k<20;k++)
		{
			for(int j=0;j<len;j++) USART_Transmit(hello[j]);
			_delay_ms(10);
		}

	}
}
```

Source Code for Receiver Module:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include<util/delay.h>
#include<string.h>
#define TIME 0xff
#define F_CPU 4000000// Clock Speed
#define BAUD 4800
#define MYUBRR (F_CPU/16/BAUD)-1
#include<inttypes.h>

#define sbi(x,y) x |=_BV(y)
#define cbi(x,y) x &= ~(_BV(y))
```

```
#define tbi(x,y) x ^=_BV(y)
#define is_high(x,y) ((x & _BV(y)) == _BV(y))
#define is_low(x,y) ((x & _BV(y)) == 0)

#define RS PD6
#define RW PD5
#define EN PD7
#define data PORTC


void USART_Init( unsigned int ubrr);
unsigned int flag;

void wait_lcd(void)
{
	flag=0xff;
	while(flag)
	{
		cbi(PORTD,EN);
		cbi(PORTD,RS);
		sbi(PORTD,RW);
		DDRC=0x00;
		data=0x00;
		sbi(PORTD,EN);
		sbi(PORTD,EN);
		flag=PINC;
		flag=PINC;
		flag&=0x80;
	}
	DDRC=0xff;
	cbi(PORTD,EN);
	cbi(PORTD,RW);
}

void lcd_init(void)
{
	cbi(PORTD,RW);
	cbi(PORTD,RS);
	data=0x38;
	sbi(PORTD,EN);
	sbi(PORTD,EN);
	cbi(PORTD,EN);
	wait_lcd();
	cbi(PORTA,RS);
	data=0x0e;
	sbi(PORTD,EN);
	sbi(PORTD,EN);
	cbi(PORTD,EN);
	wait_lcd();
	cbi(PORTD,RS);
```

```
	data=0x04;
	sbi(PORTD,EN);
	sbi(PORTD,EN);
	cbi(PORTD,EN);
	wait_lcd();
}

void clr_lcd(void)
{
	cbi(PORTD,RS);
	data=0x01;
	sbi(PORTD,EN);
	sbi(PORTD,EN);
	cbi(PORTD,EN);
	wait_lcd();
}

void write_lcd(unsigned char val)
{
	sbi(PORTD,RS);
	data=val;
	sbi(PORTD,EN);
	sbi(PORTD,EN);
	cbi(PORTD,EN);
	wait_lcd();
}

void set_cursor(unsigned int pos)
{
	cbi(PORTD,RS);
	data=pos;
	sbi(PORTD,EN);
	sbi(PORTD,EN);
	cbi(PORTD,EN);
	wait_lcd();
}


void init(void)
{
	DDRC=0xff;
	  USART_Init(MYUBRR);
	DDRD=0x00;
	DDRD|=_BV(RW)|_BV(EN)|_BV(RS);
}


void USART_Init( unsigned int ubrr)

{
```

```
        /* Set baud rate */
        UBRRH = (unsigned char)(ubrr>>8);
        UBRRL = (unsigned char)ubrr;

        /* Enable receiver and transmitter */
        UCSRB = (1<<RXEN)|(1<<TXEN);

        /* Set frame format: 8data, 1stop bit */
        UCSRC = (1<<URSEL)|(3<<UCSZ0);
} // end of USART_Init

void USART_Transmit( unsigned char data1)
{
        /* Wait for empty transmit buffer */
        while ( !( UCSRA & (1<<UDRE)) )
        ;
        /* Put data into buffer, sends the data */
        UDR = data1;
}


unsigned char USART_Receive(void)
{
        /* Wait for data to be received */
        while (!(UCSRA & (1<<RXC)));
        /* Get and return received data from buffer */
        return UDR;
};

void receive(char* rec)
{
  int i;
        rec[0]=0xaa;
        may:
        if(USART_Receive()!='G') goto may;
        if(USART_Receive()!='P') goto may;
        if(USART_Receive()!='G') goto may;
        if(USART_Receive()!='G') goto may;
        if(USART_Receive()!='A') goto may;
        if(USART_Receive()!=',') goto may;
        for(i=1;rec[i-1]!=',';i++) rec[i]=USART_Receive();
        rec[i]=' '; i++;
        for(;rec[i-1]!=',';i++) rec[i]=USART_Receive();
        rec[i]=' '; i++;
        for(;rec[i-1]!=',';i++) rec[i]=USART_Receive();
        rec[i]=' '; i++;
        for(;rec[i-1]!=',';i++) rec[i]=USART_Receive();
        rec[i]='\x0d';
```

```
	rec[i+1]='\x0a';
	rec[i+2]='\x00';
	for(;USART_Receive()!=0x0d;i++);
	if(i>70) goto may;
	if(USART_Receive()!=0x0a) goto may;
}

int main(void)
{
	int i,len;
	char* rec;
	init();
	lcd_init();
	lcd_init();
	clr_lcd();

	while(1)
	{
		receive(rec);
		len=strlen(rec);
		set_cursor(0x80);
		for(i=0;i<16;i++) write_lcd(rec[i]);
		set_cursor(0xc0);
		for(;i<len;i++) write_lcd(rec[i]);

	}
}
```