Longitude & Latitude Display System Using GPS & Microcontroller

Name – Abhiraj Rathore

Enroll no. – 101353

Supervisor – Prof. Vivek Sehgal

May, 2014

Submitted
In
Fulfillment of Degree of Bachelor of Technology
DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

AND

INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY ,

WAKNAGHAT.

CERTIFICATE

This is to certify that the work entitled **Latitude & Longitude Display System using GPS & Microcontroller"** submitted by **Abhiraj Rathore (101353)** in partial fulfillment for award for degree of Bachelor of Technology in Information Technology of JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY has been carried out under my supervision. This work has not been submitted partially or wholly to any other University for any award of this or any other degree.

Prof. Vivek Sehgal
(Associate Professor)
Department of Computer Science Engineering and Information Technology
Jaypee University of Information Technology
Waknaghat

Acknowledgement

"It is not possible to prepare a project without the assistance & Encouragement of other people. This one is certainly no exception."

On the very outset of this report, we would like to extend our sincere & heartfelt obligation towards all the personages who have helped us in this endeavor. Without their active guidance, help, cooperation & encouragement, we would not have made headway in the project.

We would like to show our greatest appreciation to **Prof. Vivek Sehgal.** We feel motivated every time we get his encouragement. For his coherent guidance throughout the tenure of the project, we feel fortunate to be taught by **Prof. Vivek Sehgal**, who gave us his unwavering support. Besides being our mentor, he taught us that there is no substitute for hard work.

We will be always in debt of **Prof. Punit Gupta** for providing us his timely help and guidance.

We owe our heartiest thanks to **Brig. (Retd.) S.P. Ghrera** (HOD, CES/IT Department) who has always inspired us to take initiatives and showed us the path for achieving our goal.

In the light of new developments and recent findings, we devote the task that was asked from us at Jaypee University of Information Technology to **" Latitude & Longitude Display System using GPS & Microcontroller"**.

Abhiraj Rathore(101353)

Table of Contents

CHAPTER 1 - INTRODUCTION

What is a MICROCONTROLLER?

A **microcontroller** is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems.

Looking Inside the MICROCONTROLLER?

- Read Only Memory (ROM)

Read Only Memory (ROM) is a type of memory used to permanently save the program being executed. The size of the program that can be written depends on the size of this memory. ROM can be built in the microcontroller or added as an external chip, which depends on the type of the microcontroller. Both options have some disadvantages. If ROM is added as an external chip, the microcontroller is cheaper and the program can be considerably longer. At the same time, a number of available pins is reduced as the microcontroller uses its own input/output ports for connection to the chip. The internal ROM is usually smaller and more expensive, but leaves more pins available for connecting to peripheral environment. The size of ROM ranges from 512B to 64KB.

- Random Access Memory (RAM)

Random Access Memory (RAM) is a type of memory used for temporary storing data and intermediate results created and used during the operation of the microcontrollers. The content of this memory is cleared once the power supply is off. For example, if the program performes an addition, it is necessary to have a register standing for what in everyday life is called the “sum” . For that purpose, one of the registers in RAM is called the "sum" and used for storing results of addition. The size of RAM goes up to a few KBs.

- Electrically Erasable Programmable ROM (EEPROM)

The EEPROM is a special type of memory not contained in all microcontrollers. Its contents may be changed during program execution (similar to RAM), but remains permanently saved even

after the loss of power (similar to ROM). It is often used to store values, created and used during operation (such as calibration values, codes, values to count up to etc.), which must be saved after turning the power supply off. A disadvantage of this memory is that the process of programming is relatively slow. It is measured in miliseconds.

- Special Function Registers (SFR)

Special function registers are part of RAM memory. Their purpose is predefined by the manufacturer and cannot be changed therefore. Since their bits are physically connected to particular circuits within the microcontroller, such as A/D converter, serial communication module etc., any change of their state directly affects the operation of the microcontroller or some of the circuits. For example, writing zero or one to the SFR controlling an input/output port causes the appropriate port pin to be configured as input or output. In other words, each bit of this register controls the function of one single pin.

- Program Counter

Program Counter is an engine running the program and points to the memory address containing the next instruction to execute. After each instruction execution, the value of the counter is incremented by 1. For this reason, the program executes only one instruction at a time just as it is written. However…the value of the program counter can be changed at any moment, which causes a "jump" to a new memory location. This is how subroutines and branch instructions are executed. After jumping, the counter resumes even and monotonous automatic counting +1, +1, +1…

- Central Processor Unit (CPU)

As its name suggests, this is a unit which monitors and controls all processes within the microcontroller and the user cannot affect its work. It consists of several smaller subunits, of which the most important are:

- **Instruction decoder** is a part of the electronics which recognizes program instructions and runs other circuits on the basis of that. The abilities of this circuit are expressed in the "instruction set" which is different for each microcontroller family.
- **Arithmetical Logical Unit (ALU)** performs all mathematical and logical operations upon data.
- **Accumulator** is an SFR closely related to the operation of ALU. It is a kind of working desk used for storing all data upon which some operations should be executed (addition, shift etc.). It also stores the results ready for use in further processing. One of the SFRs, called the Status Register, is closely related to the accumulator, showing at any given time the "status" of a number stored in the accumulator (the number is greater or less than zero etc.).

- Oscillator

Even pulses generated by the oscillator enable harmonic and synchronous operation of all circuits within the microcontroller. It is usually configured as to use quartz-crystal or ceramics resonator for frequency stabilization. It can also operate without elements for frequency stabilization (like RC oscillator). It is important to say that program instructions are not executed at the rate imposed by the oscillator itself, but several times slower. It happens because each instruction is executed in several steps. For some microcontrollers, the same number of cycles is needed to execute any instruction, while it's different for other microcontrollers. Accordingly, if the system uses quartz crystal with a frequency of 20MHz, the execution time of an instruction is not expected 50nS, but 200, 400 or even 800 nS, depending on the type of the microcontroller!

- Timers/Counters

Most programs use these miniature electronic "stopwatches" in their operation. These are commonly 8- or 16-bit SFRs the contents of which is automatically incremented by each coming pulse. Once the register is completely loaded, an interrupt is generated!

If these registers use an internal quartz oscillator as a clock source, then it is possible to measure the time between two events . If the registers use pulses coming from external source, then such a timer is turned into a counter.

- Watchdog timer

The Watchdog Timer is a timer connected to a completely separate RC oscillator within the microcontroller.

If the watchdog timer is enabled, every time it counts up to the program end, the microcontroller reset occurs and program execution starts from the first instruction. The point is to prevent this from happening by using a special command. The whole idea is based on the fact that every program is executed in several longer or shorter loops.

If instructions resetting the watchdog timer are set at the appropriate program locations, besides commands being regularly executed, then the operation of the watchdog timer will not affect the program execution.

If for any reason (usually electrical noise in industry), the program counter "gets stuck" at some memory location from which there is no return, the watchdog will not be cleared, so the register's value being constantly incremented will reach the maximum et voila! Reset occurs!

- **Interrupt** - electronics is usually more faster than physical processes it should keep under control. This is why the microcontroller spends most of its time waiting for something to happen or execute. In other words, when some event takes place, the microcontroller does something. In order to prevent the microcontroller from spending most of its time endlessly checking for logic state on input pins and registers, an interrupt is generated. It is the signal which informs the central processor that something attention worthy has happened. As its name suggests, it interrupts regular program execution. It can be generated by different sources so when it occurs, the microcontroller immediately stops operation and checks for the cause. If it is needed to perform some operations, a current state of the program counter is pushed onto the Stack and the appropriate program is executed. It's the so called interrupt routine.

- **Stack** is a part of RAM used for storing the current state of the program counter (address) when an interrupt occurs. In this way, after a subroutine or an interrupt execution, the microcontroller knows from where to continue regular program execution. This address is cleared after returning to the program because there is no need to save it any longer, and one location of the stack is automatically availale for further use. In addition, the stack can consist of several levels. This enables subroutines' nesting, i.e. calling one subroutine from another.

There most commonly used Microcontroller in the world today

ATMEL AVR

The **AVR** is a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to one-time programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time.

Basic families

AVRs are generally classified into following:

- **tinyAVR** — the ATtiny series
  - 0.5–16 kB program memory
  - 6–32-pin package
  - Limited peripheral set
- **megaAVR** — the ATmega series
  - 4–512 kB program memory
  - 28–100-pin package
  - Extended instruction set (multiply instructions and instructions for handling larger program memories)
  - Extensive peripheral set
- **XMEGA** — the ATxmega series
  - 16–384 kB program memory
  - 44–64–100-pin package (A4, A3, A1)
  - Extended performance features, such as DMA, "Event System", and cryptography support.
  - Extensive peripheral set with ADCs

Microchip PIC

PIC is a family of modified Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1650,originally developed by General Instrument's Microelectronics Division. The name PIC initially referred to "**Peripheral Interface Controller**"' now it is "**PIC**"' only.[4][5]

PICs are popular with both industrial developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability.

The PIC architecture is characterized by its multiple attributes:

- Separate code and data spaces (Harvard architecture).
- A small number of fixed length instructions
- Most instructions are single cycle execution (2 clock cycles, or 4 clock cycles in 8-bit models), with one delay cycle on branches and skips
- One accumulator (W0), the use of which (as source operand) is implied (i.e. is not encoded in the opcode)
- All RAM locations function as registers as both source and/or destination of math and other functions.[6]
- A hardware stack for storing return addresses
- A small amount of addressable data space (32, 128, or 256 bytes, depending on the family), extended through banking
- Data space mapped CPU, port, and peripheral registers

.

Philips LPC

LPC is a family of 32-bit microcontroller integrated circuits by NXP Semiconductors (formerly Philips Semiconductors). The LPC chips are grouped into related series that are based around the same 32-bit ARM processor core, such as the Cortex-M4F, Cortex-M3, Cortex-M0+, or Cortex-M0. Internally, each microcontroller consists of the processor core, static RAM memory, flash memory, debugging interface, and various peripherals. The legacy LPC families were based on the 8-bit 80C51 core.[2] As of February 2011, NXP had shipped over one billion ARM processor-based chips.

Motorola's Freescale 68HC11

The **68HC11** (**6811** or **HC11** for short) is an 8-bit microcontroller (µC) family introduced by Motorola in 1985.[1] Now produced by Freescale Semiconductor, it descended from the Motorola 6800 microprocessor. It is a CISC microcontroller. The 68HC11 devices are more powerful and more expensive than the 68HC08microcontrollers, and are used in barcode readers, hotel card key writers, amateur robotics, and various other embedded systems. The MC68HC11A8 was the first MCU to include CMOS EEPROM.

Difference between Microcontrollers & Microprocessors

Microprocessor	Microcontroller
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

Microcontroller Used

The Microcontroller used as a part of this project is **ATMEL's AtMega 328P.**

ATMEGA 328P MICROCONTROLLER

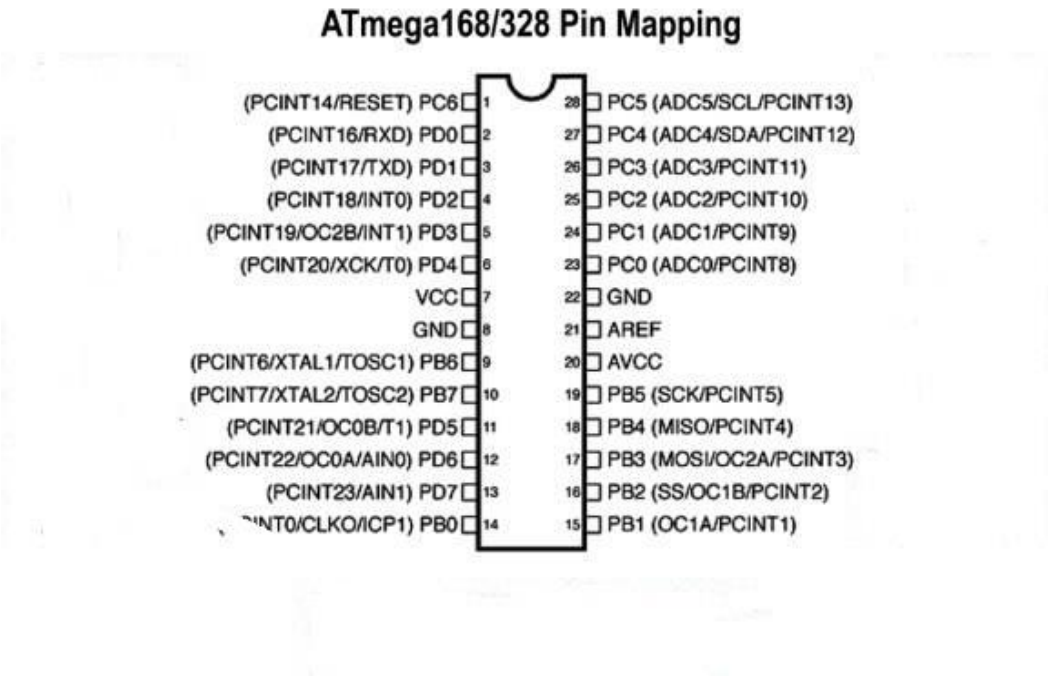
ATMEGA 328P

The **ATmega328P** is a single chip micro-controller created by Atmel and belongs to the mega series.
The high-performance Atmel 8-bit AVR RISC-based microcontroller combines

- Flash memory - 32 KB ISP
- EEPROM - 1 KB
- SRAM - 2 KB
- 23 general purpose I/O lines,
- 32 general purpose working registers
- 3 flexible timer/counters with compare modes, internal and external interrupts,
- Serial Programmable USART
- A byte-oriented 2-wire serial interface
- SPI serial port

- 6-channel 10-bit A/D converter
- Internal oscillator
- Software selectable power saving modes.

The device operates between 1.8-5.5 volts. By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed.



What is a GPS?

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.[1] 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.

A GPS receiver calculates its position by precisely timing the signals sent by GPS satellites high above the Earth. Each satellite continually transmits messages that include:

- the time the message was transmitted and,
- satellite position at time of message transmission.

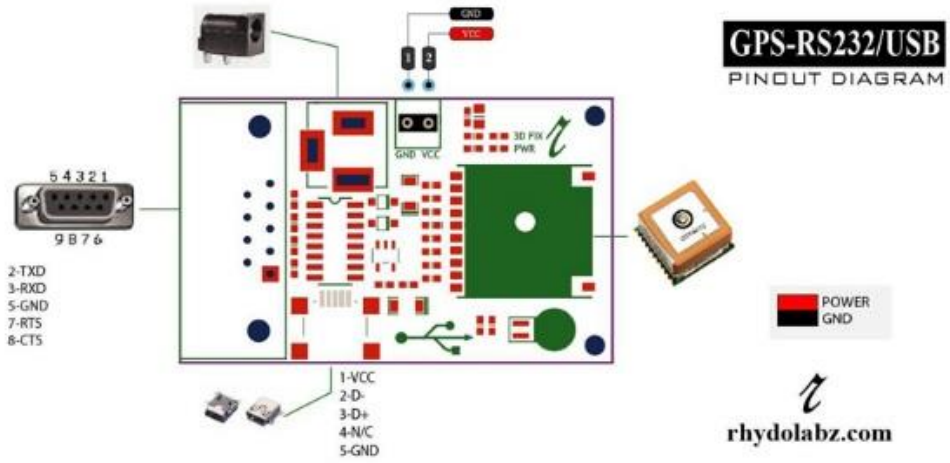
The receiver uses the messages it receives to determine the transit time of each message and computes the distance to each satellite using the speed of light. Each of these distances and satellites' locations defines a sphere. The receiver is on the surface of each of these spheres when the distances and the satellites' locations are correct. These distances and satellites' locations are used to compute the location of the receiver using the navigation equations. This location is then displayed, perhaps with a moving map display or latitude and longitude; elevation or altitude information may be included, based on height above the geoid (e.g. L80M39).

Reciever Used

The Reciever used as a part of this project is **L80M39**

Features:

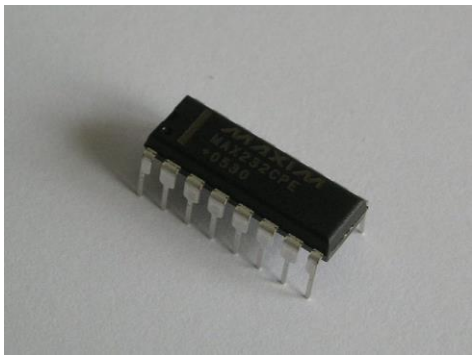
- MediaTek MT3329 Chipset, L1 Frequency, C/A code, 66 Channels
- RS232 Interface via DB9 Connector
- USB MiniB type connector for USB interface and Power
- 9 VDC supply @ 55 mA (typical)
- Data output Baud rate: 9600 bps(Default)
- Standard NMEA0183 output format
- Low Power Consumption: 55mA @ acquisition, 40mA @ tracking
- High Sensitivity, -165 dBm, TCXO Design , superior urban performances
- Position Accuracy: <3.0M 2D-RMS
- DGPS (WAAS/EGNOS/MASA/GAGAN) Support
- Multi-path Compensation ; E-GSM-900 Band Rejection
- Cold Start is Under 36 seconds (Typical)
- Warm Start is Under 34 seconds (Typical)
- Hot Start is Under 1 second (Typical)
- Max. Update Rate : 10Hz (Default: 1 Hz)



Pin Diagram GPS Module L80M39

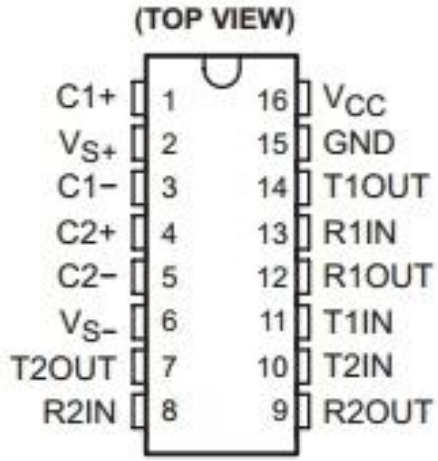
MAX232

The **MAX232** is an IC, that converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits. The MAX232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS signals.



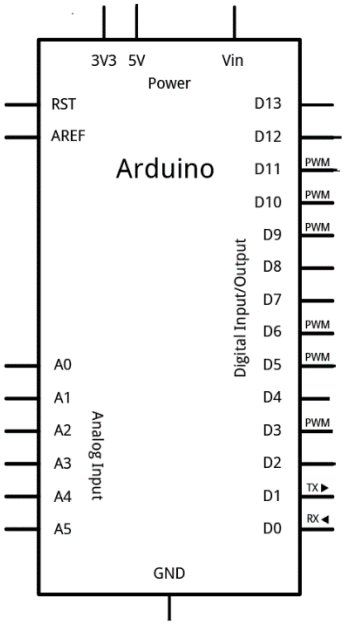
The drivers provide RS-232 voltage level outputs (approx. ± 7.5 V) from a single + 5 V supply via on-chip charge pumps and external capacitors. This makes it useful for implementing RS-232 in devices that otherwise do not need any voltages outside the 0 V to + 5 V range, as power supply design does not need to be made more complicated just for driving the RS-232 in this case.

The receivers reduce RS-232 inputs (which may be as high as ± 25 V), to standard 5 V TTL levels. These receivers have a typical threshold of 1.3 V, and a typical hysteresis of 0.5 V.



PROGRAMMER

The programmer used is **ARDUINO UNO BOARD.** The Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



Pin diagram Arduino Uno

CHAPTER 2 –PREVIOUS STUDY

As this project was altogether new and different, it needed the study of basics of Hardware, circuits. Thus I had to go through the basic libraries for coding in Arduino, Implemented the basic circuits & programs and then went through with the Project .

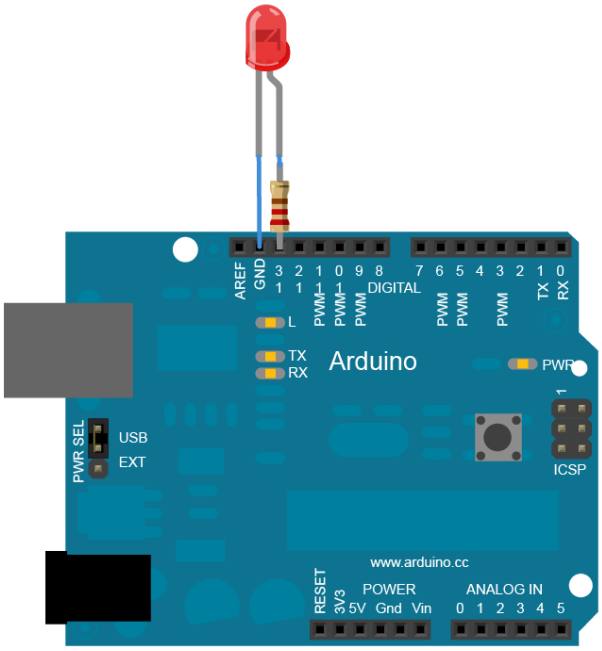
The basic programs implemented were

Blinking an LED

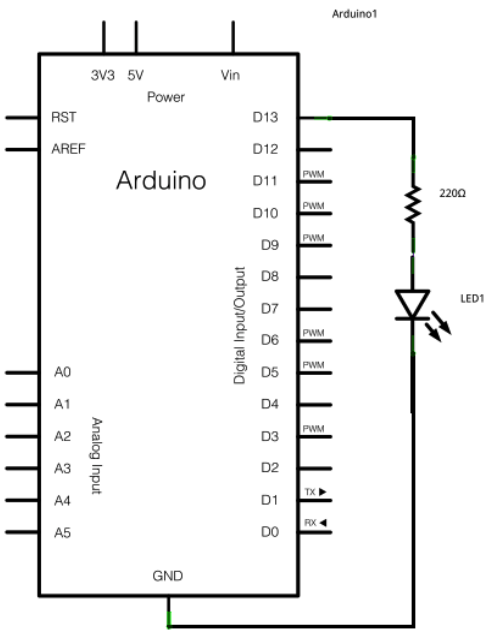
Hardware Required

- Arduino Uno Board
- LED'S
- AtMega 328P

Circuit



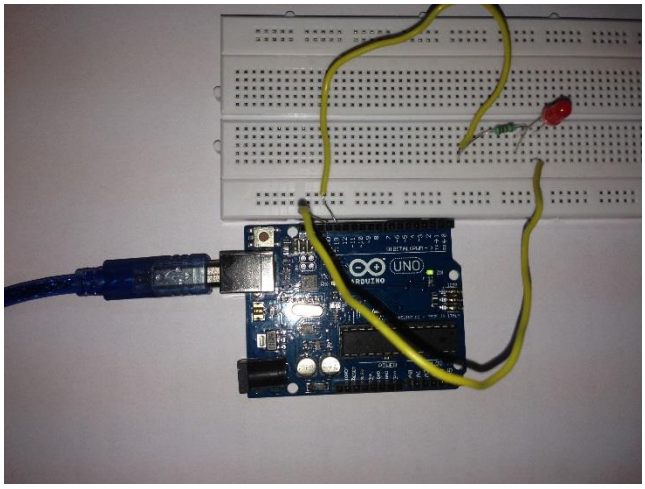
Schematic



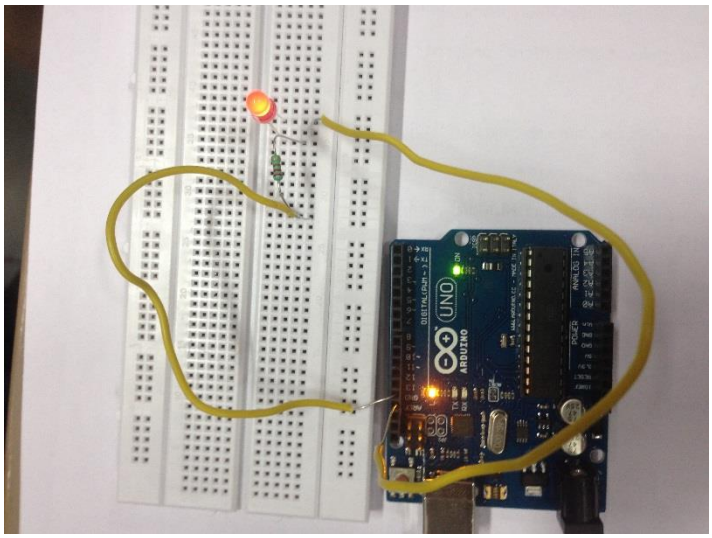
Code

```
 #include <Blink.h>
int led = 13;
void setup()
{
   pinMode(led, OUTPUT);
}
void loop()
{
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Results



ORIGNAL CIRCUIT



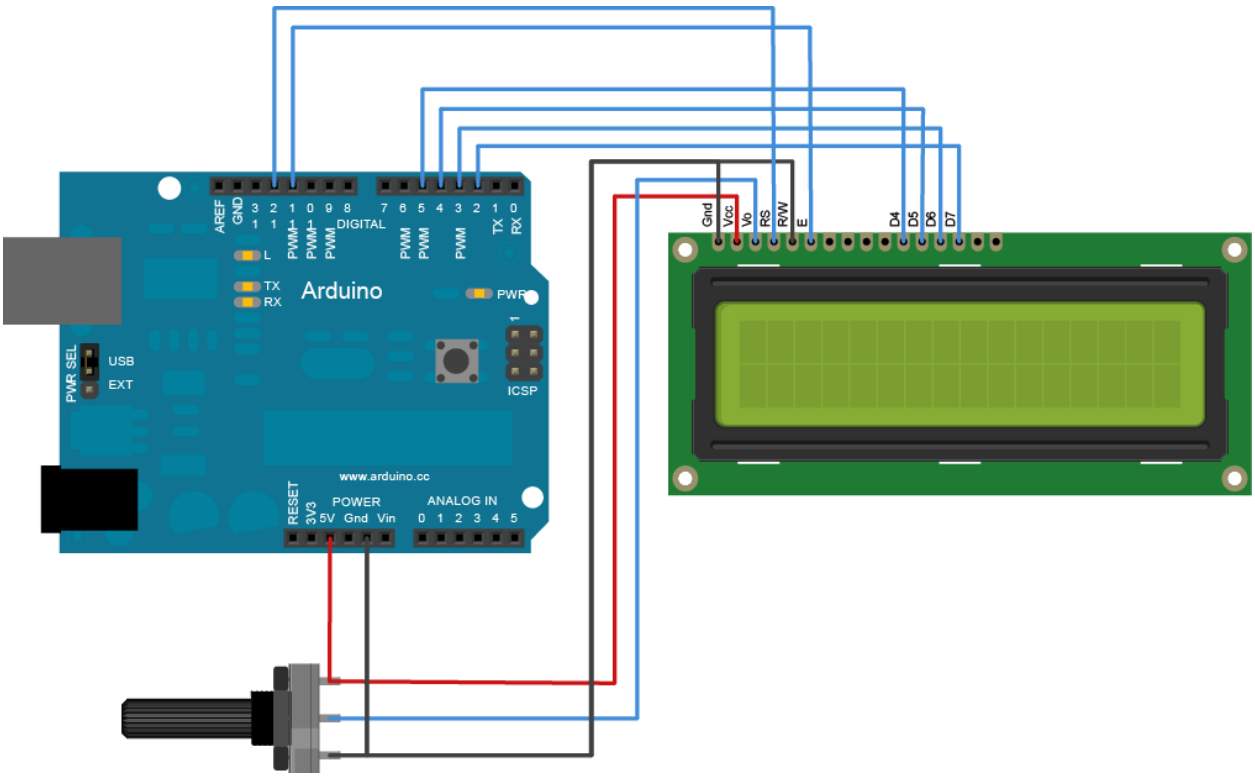
BLINKING LED

Displaying On a LCD Screen

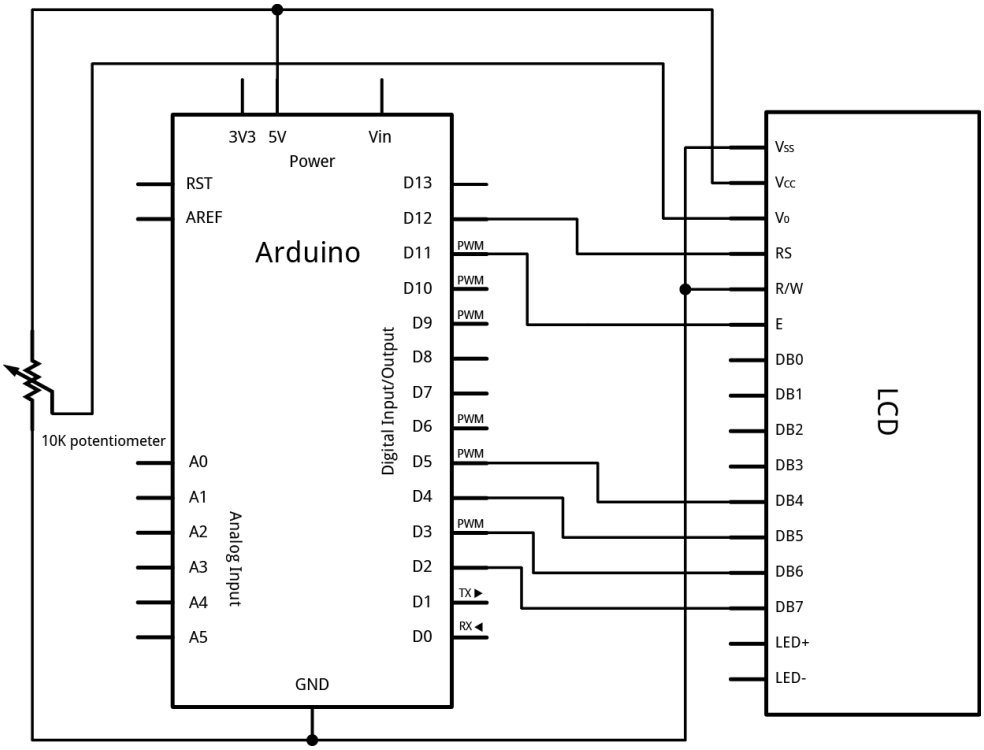
Hardware Required

- Arduino Board
- LCD Screen
- Pin headers to solder to the LCD display pins
- 10k Variable Resistance
- Breadboard
- Hook-up wire

Circuit



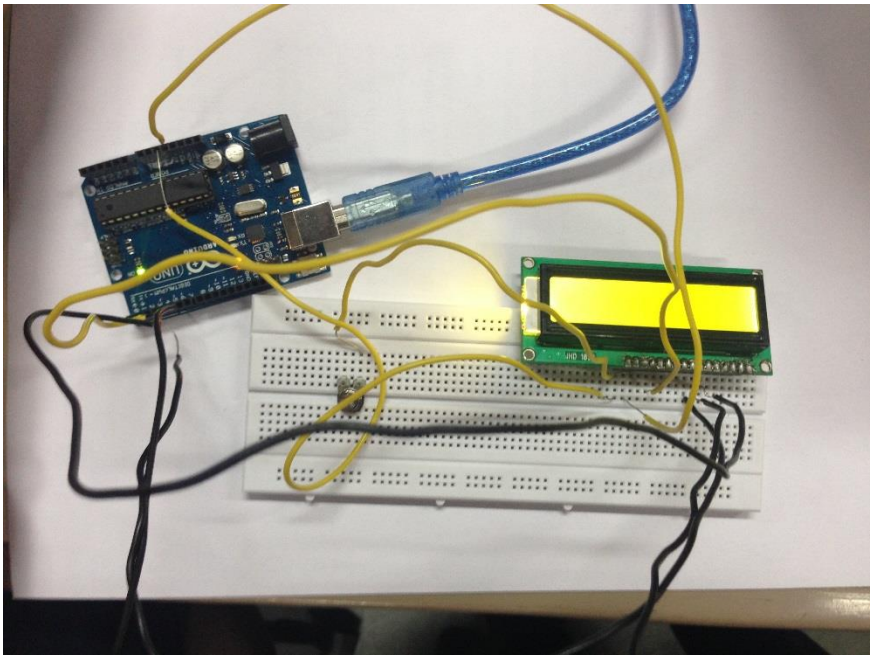
Schematic



Code

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  lcd.begin(16, 2);

  lcd.print("hello, world!");
}

void loop()
{
  lcd.setCursor(0, 1);
  lcd.print(millis()/1000);
}
```

Results



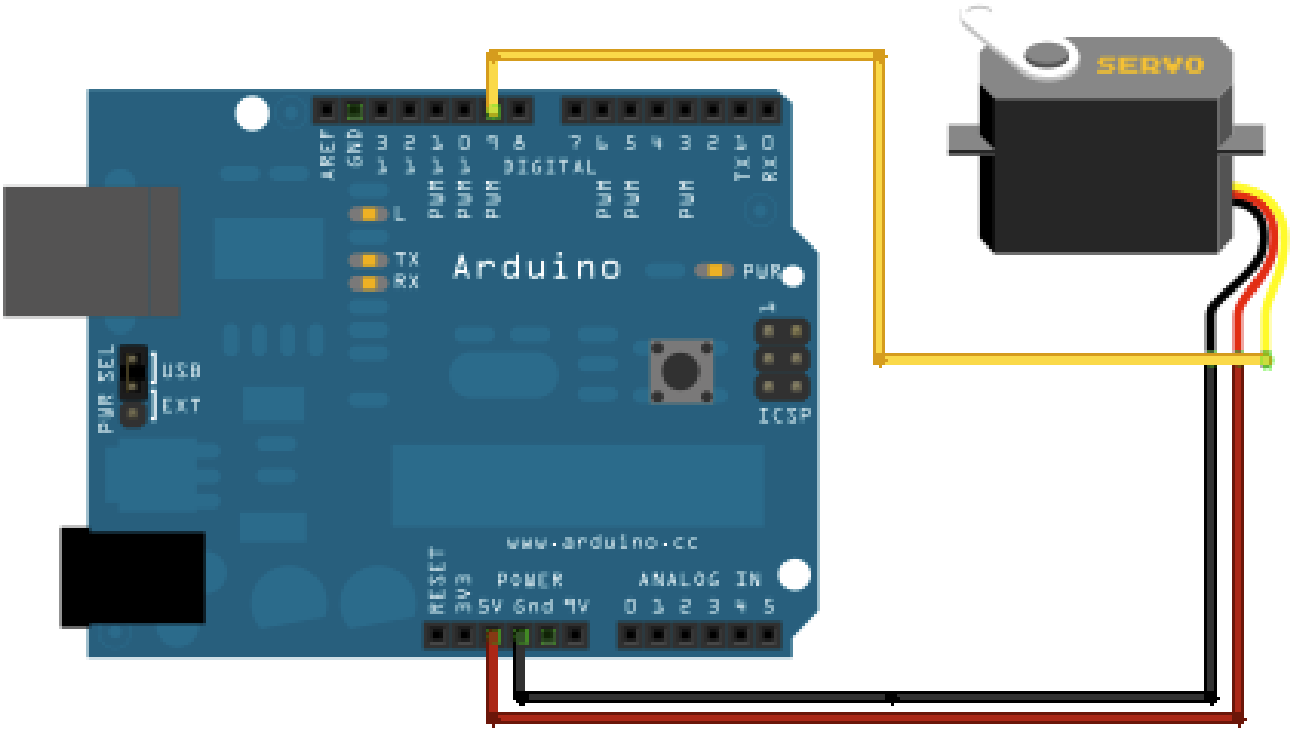
DISPLAYING ON A LCD

Interfacing Servo Motor with Arduino Uno

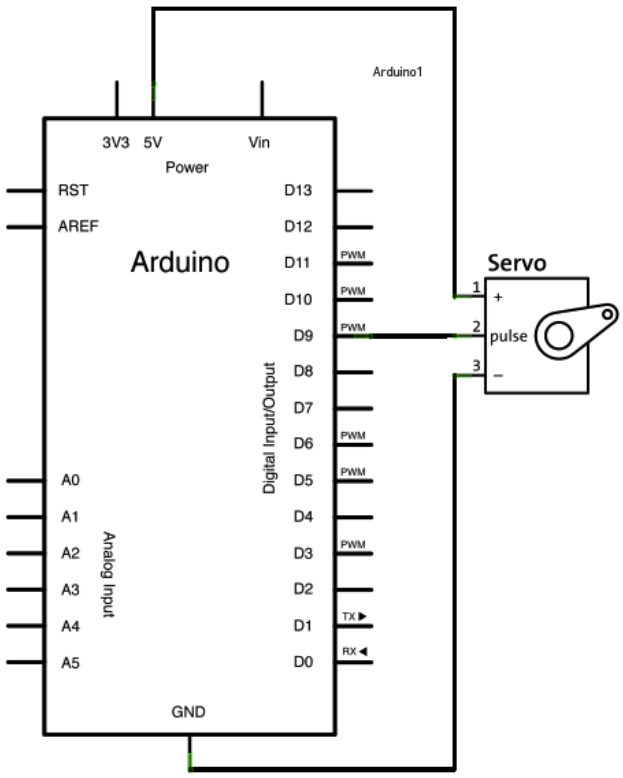
Hardware Required

- Arduino Uno Board
- Servo Motor
- Hook Up Wires

Circuit



Schematic



Code for Arduino

```
#include <Servo.h>

Servo servo1; Servo servo2;


void setup() {

  pinMode(1,OUTPUT);
  servo1.attach(14); //analog pin 0
  //servo1.setMaximumPulse(2000);
  //servo1.setMinimumPulse(700);

  servo2.attach(15); //analog pin 1
  Serial.begin(19200);
  Serial.println("Ready");

}
```

```
void loop() {

  static int v = 0;

  if ( Serial.available()) {
    char ch = Serial.read();

    switch(ch) {
      case '0'...'9':
        v = v * 10 + ch - '0';
        break;
      case 's':
        servo1.write(v);
        v = 0;
        break;
      case 'w':
        servo2.write(v);
        v = 0;
        break;
      case 'd':
        servo2.detach();
        break;
      case 'a':
        servo2.attach(15);
        break;
    }
  }

  Servo::refresh();

}
```

Processing Code

```
import processing.serial.* ;


int gx = 15;
int gy = 35;
int spos=90;

float leftColor = 0.0;
float rightColor = 0.0;
```

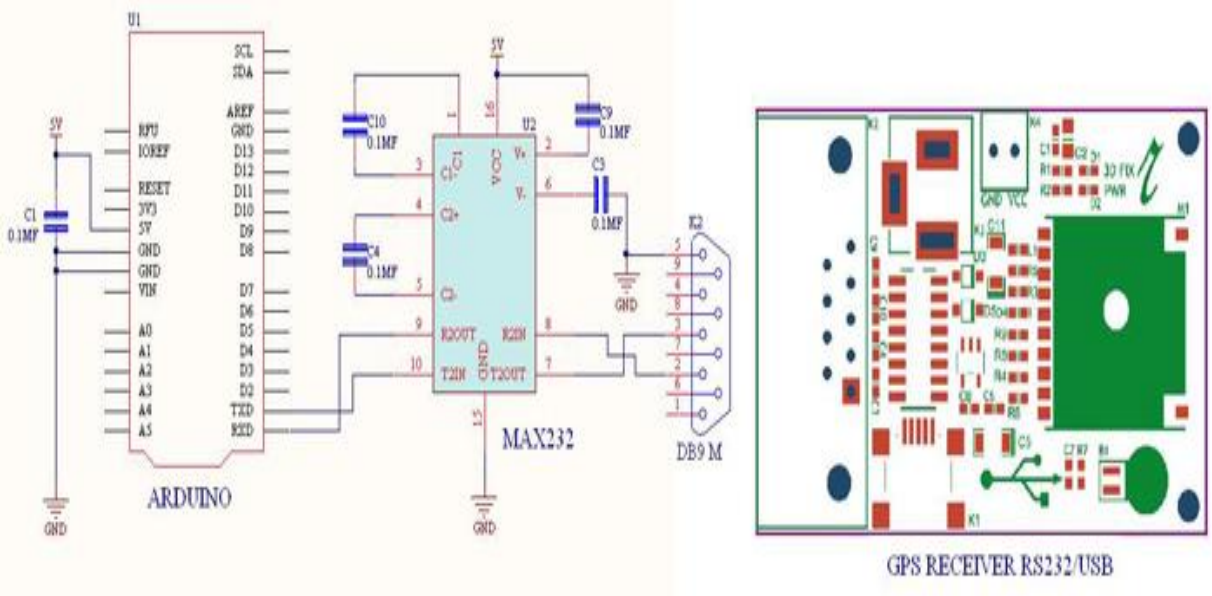
```
Serial port;
void setup()
{
  size(720, 720);
  colorMode(RGB, 1.0);
  noStroke();
  rectMode(CENTER);
  frameRate(100);

  println(Serial.list());

  port = new Serial(this, Serial.list()[1], 19200);
}

void draw()
{
  background(0.0);
  update(mouseX);
  fill(mouseX/4);
  rect(150, 320, gx*2, gx*2);
  fill(180 - (mouseX/4));
  rect(450, 320, gy*2, gy*2);
}

void update(int x)
{
  spos= x/4;
  port.write("s"+spos);
  leftColor = -0.002 * x/2 + 0.06;
  rightColor =  0.002 * x/2 + 0.06;
  gx = x/2;
  gy = 100-x/2;

}
```

Chapter 3: INTERFACING L80M39 GPS MODULE WITH ARDUINO UNO

Hardware Required

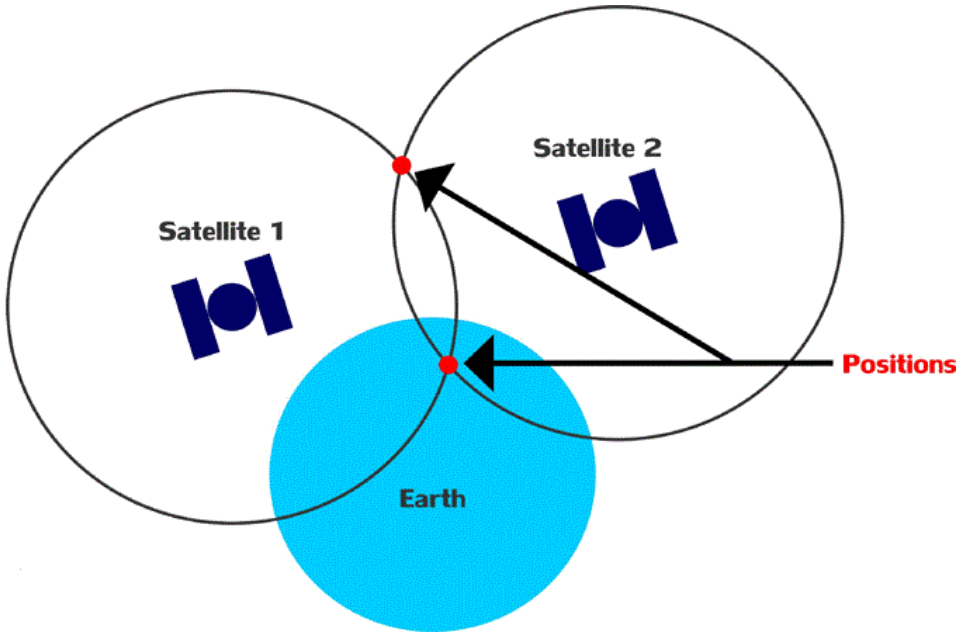
- Arduino Uno Board
- GPS Reciever Module L80M39
- Hook Up Wires

Schematic



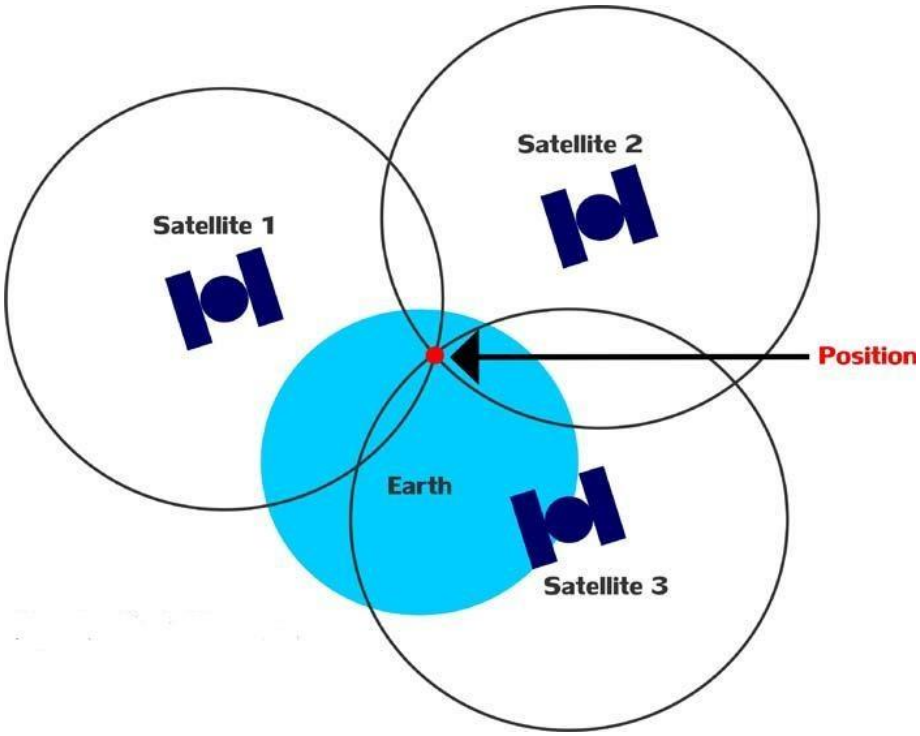
Working

Let me tell you, if we want an object location on the earth, GPS receiver have to connect with minimum 3 Satellite at the point of time. Now i explain this in a simple way by the help of fig 1.1:



In Fig 1.1, I am trying to symbolize 'Earth with a Sky Blue Circle', 'Satellites with Dark Blue', 'Satellites radius with Black outline circle' and the 'position with small Red circle'.

So let us assume we are standing on the earth with GPS Receiver and receiver is connected with two satellites- Satellite 1 and Satellite 2 at the time of instant. Both satellites have its own radius and after connecting with GPS Receiver, both radius will intersect with two points. This means we got two positions of us with respect to satellite at the same point of time which is practically not possible. So we will not able to fix our fix position in this case.



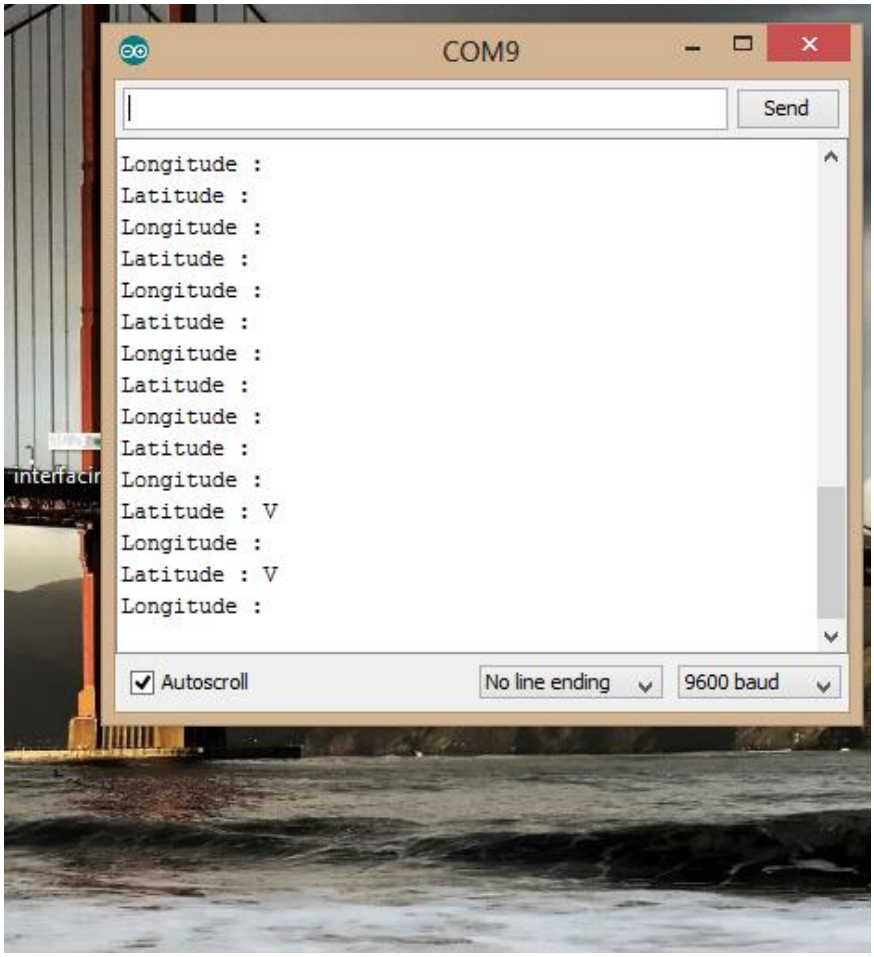
Problem solved. If our GPS receiver is connect with minimum 3 satellite at point of time, we got three radius by Satellite 1, Satellite 2 and Satellite 3, and all three radius will intersect in a common point and that common intersect point is our fix location on the earth with respect to satellites, that I meant in the starting of this section. I hope now you will be able to understand the concept

Code

```
int Gpsdata;
 unsigned int finish =0;
 unsigned int pos_cnt=0;
 unsigned int lat_cnt=0;
 unsigned int log_cnt=0;
 unsigned int flg    =0;
 unsigned int com_cnt=0;
 char lat[20];
 char lg[20];

  void Receive_GPS_Data();


 void setup()
 {
  Serial.begin(9600);
 }


 void loop()
 {
    Receive_GPS_Data();
    Serial.print("Latitude : ");
    Serial.println(lat);
    Serial.print("Longitude : ");
    Serial.println(lg);
    finish = 0;pos_cnt = 0;

 void Receive_GPS_Data()
 {
  while(finish==0){
    while(Serial.available()>0){
      Gpsdata = Serial.read();
      flg = 1;
     if( Gpsdata=='$' && pos_cnt == 0)
       pos_cnt=1;
     if( Gpsdata=='G' && pos_cnt == 1)
       pos_cnt=2;
     if( Gpsdata=='P' && pos_cnt == 2)
       pos_cnt=3;
     if( Gpsdata=='R' && pos_cnt == 3)
       pos_cnt=4;
     if( Gpsdata=='M' && pos_cnt == 4)
       pos_cnt=5;
```

```
      if( Gpsdata=='C' && pos_cnt==5 )
        pos_cnt=6;
      if(pos_cnt==6 &&  Gpsdata ==','){
        com_cnt++;
        flg=0;
      }

      if(com_cnt==3 && flg==1){
       lat[lat_cnt++] =  Gpsdata;
       flg=0;
      }

      if(com_cnt==5 && flg==1){
        lg[log_cnt++] =  Gpsdata;
        flg=0;
      }

      if( Gpsdata == '*' && com_cnt >= 5){
        com_cnt = 0;
        lat_cnt = 0;
        log_cnt = 0;
        flg     = 0;
        finish  = 1;

     }
    }
  }
}
```

Results using Serial Monitor



CHAPTER 4 – FUTURE SCOPE

Like the Internet, GPS is an essential element of the global information infrastructure. The free, open, and dependable nature of GPS has led to the development of hundreds of applications affecting every aspect of modern life. GPS technology is now in everything from cell phones and wristwatches to bulldozers, shipping containers, and ATM's.

GPS boosts productivity across a wide swath of the economy, to include farming, construction, mining, surveying, package delivery, and logistical supply chain management. Major communications networks, banking systems, financial markets, and power grids depend heavily on GPS for precise time synchronization. Some wireless services cannot operate without it.

GPS saves lives by preventing transportation accidents, aiding search and rescue efforts, and speeding the delivery of emergency services and disaster relief. GPS is vital to the Next Generation Air Transportation System (NextGen) that will enhance flight safety while increasing airspace capacity. GPS also advances scientific aims such as weather forecasting, earthquake monitoring, and environmental protection.

Finally, GPS remains critical to U.S. national security, and its applications are integrated into virtually every facet of U.S. military operations. Nearly all new military assets -- from vehicles to munitions -- come equipped with GPS.

CHAPTER 5 – CONCLUSION

Through this Project , I got the basic idea of working with Microcontrollers and its applications. Global Positioning System is used for Navigational Purposes. This project taught me about GPS as well as interfacing it with Microcontrollers. The working & function of components like MAX 232 and Arduino Uno board were also understood really well. Thus this Project enriched my knowledge and was found to be very beneficial.

CHAPTER 6 – TOOLS & TECHNIQUES USED

Proteus - It is a software for microprocessor simulation, schematic capture, and printed circuit board (PCB) design. It is developed by Labcenter Electronics. It combines the ISIS schematic capture and ARES PCB layout programs to provide a powerful, integrated and easy to use suite of tools for professional PCB Design.

Arduino IDE - The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a "sketch".

Fritzing - Fritzing is an open source software initiative to support designers and artists ready to move from physical prototyping to actual product. It was developed at the University of Applied Sciences of Potsdam. The software is created in the spirit of Processing and Arduinoand allows a designer, artist, researcher, or hobbyist to document their Arduino-based prototype and create a PCB layout for manufacturing.

CHAPTER 7 – REFERNCES

- **[1]Edward A. Lee and Sanjit A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach**

- **[2]Sangiovanni-Vincentelli**, A., **Zeng**, H., **Di Natale**, M., **Marwedel**, **Embedded Systems Development**

WEB REFRENCES

- **http://learn.parallax.com/KickStart/28500**

- **http://www.rhydolabz.com/wiki/?p=229**

- **http://en.wikipedia.org/**

- **http://www.arduino.cc/**