Global Positioning System using Arduino UNO

DWITI KRUSHNA DAS

Abstract: Ever since the dawn of advanced technology, we as inquisitive beings try to find our own position with respect to others. When the globe was become smaller because of faster transportation & knowledge of geography. In today's world for navigation, we require GPS to locate some particular place. With electronics getting smaller & smaller, GPS devices come in almost all Portable Devices like phones, Tablet, Electronic Key, & are also used in Cars, Buses, Ships and Airplane. In this project we are going to make a GPS device which will display the coordinates on an LCD.

Aim of the Project(s):

To design a GPS (Global Positioning System) device using Arduino UNO & display the coordinates of the position on an LCD.

Instruments Required:

Arduino UNO, Bread Board, GPS Module (Royaltek REB-4216, Receiver: SIM28ML, Antenna: GPS-01, Frequency range: L1 = 1575.42 MHz), LCD (659M10, 16×2 Display), Wires, Pins, USB, Power Source (portable), a Computer with the Arduino Software, [Bluetooth Module, if required].

Theory & Principle:

How does a GPS Works?

The Global Positioning System concept is based on precision & accuracy of *time*. The satellites carry very stable atomic clocks that are synchronized to each other and to ground clocks. Any drift from true time maintained on the ground is corrected daily. Likewise, the satellite locations are monitored precisely. GPS receivers have clocks as well—however, they are not synchronized with true time, and are less stable. GPS satellites continuously transmit their current time and position. A GPS receiver monitors multiple satellites and solves equations to determine the exact position of the receiver and its deviation from true time. At a minimum, four satellites must be in view of the receiver for it to compute four unknown quantities (three position coordinates and clock deviation from satellite time).

Trilateration/Triangulation (for Geographical Coordinates)

Let us consider that we are standing somewhere on Earth with at least three satellites (part of GPS Network) in the sky above us. If we know how far away, we are from satellite A (refer to the Figure: 1), then we know we can be located somewhere on the red circle {representing a sphere} (circle represents the spherical range of the scan by the satellite at the centre). If we do the same for satellites B and C, we can work out our location by seeing where the three circles intersect. This is just what the GPS receiver does, basically it receives the coordinate time & the position from the satellite.

The more satellites there are above the horizon the more accurately our GPS unit can determine where we are. Additional satellites help in synchronizing the time accurately.

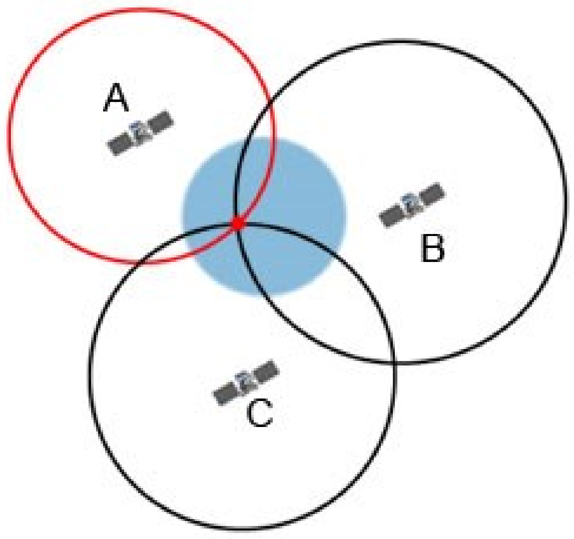

Figure 1: Triangulation by Satellites for Positioning in GPS

The GPS signal contains three different bits of information — a **pseudo random code**, **almanac data** and **ephemeris data**.

Pseudo Random Code is simply an I. D. code that identifies which satellite is transmitting information. You can often view this number on your GPS unit's satellite information page, the number attached to each signal bar identifies which satellites it's receiving a signal from.

1. **Almanac Data** is data that describes the orbital courses of the satellites. Every satellite will broadcast almanac data for EVERY satellite. Your GPS receiver uses this data to determine which satellites it expects to see in the local sky. It can then determine which satellites it should track. With Almanac data the receiver can concentrate on those satellites it can see and forget about those that would be over the horizon and out of view. Almanac data is not precise and can be valid for many months.

2. **Ephemeris Data** is data that tells the GPS receiver where each GPS satellite should be at any time throughout the day. Each satellite will broadcast its OWN ephemeris data showing the orbital information for that satellite only. Because ephemeris data is very precise orbital and clock correction data necessary for precise positioning, its validity is much shorter. It is broadcast in three six second blocks repeated every 30 seconds. The data is considered valid for up to 4 hours but different manufacturers consider it valid for different periods with some treating it as stale after only 2 hours.

GPS Satellites and Relativity

GPS satellites have atomic clocks on board to keep accurate time. General and Special Relativity however predict that differences will appear between these clocks and an identical clock on Earth due to time dilation (different non-inertial frames).

General Relativity predicts that time will appear to run slower under stronger gravitational pull – the clocks on board the satellites will therefore seem to run faster than a clock on Earth. Moreover, Special Relativity predicts that because the satellites' clocks are moving relative to a clock on Earth, they will appear to run slower.

The whole GPS network has to make allowances for these effects, this is one of the important proofs that Relativity has a real impact on our daily lives.

Circuit Design:

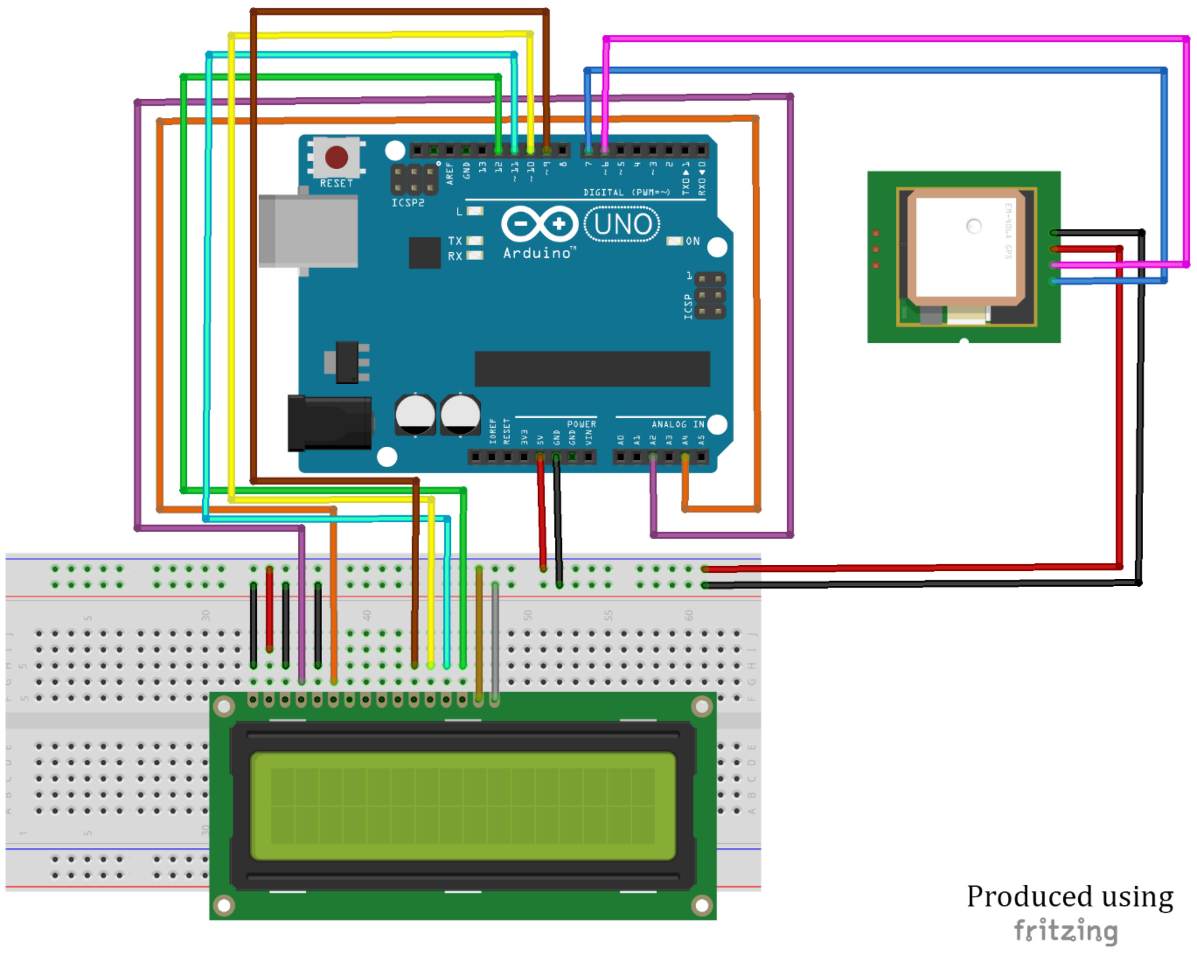

Figure 2: Circuit Design for the Project (made using **Fritzing** Software)

Connections:

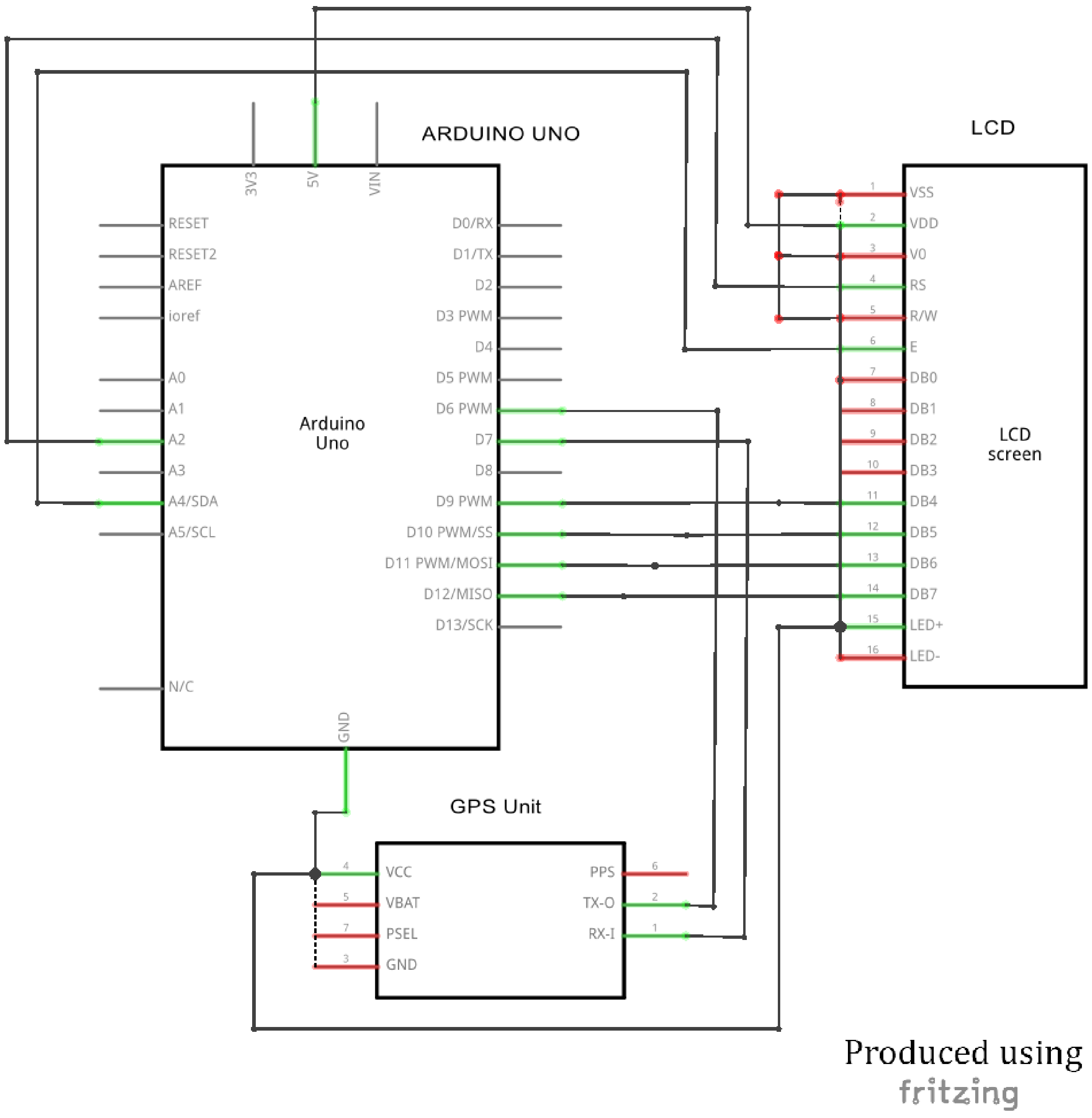

Figure 3: Wiring Connections, Schematic Diagram (made using **Fritzing** Software)

Methodology & Experimental Setup:

i) First of all, I designed the circuit according to the requirement, that is a GPS Receiver (+ Antenna) & LCD (Liquid Crystal Display) is connected to Arduino UNO.

ii) Then according to the circuit design, the equipment(s): Arduino UNO, GPS Module (Royaltek REB-4216, Receiver: SIM28ML, Antenna: GPS-01, Frequency range: L1 = 1575.42 MHz), LCD (659M10, 16×2 Display), Bread Board, Wires, Pins, USB, Power Source (portable)) are connected.

iii) The GPS is first verified using a test code just to check the receiver & verifying the data.

iv) Then the code is written fully to separate out the information required, i.e., Latitude & Longitude, and then the information is relayed to the LCD.

v) The program is verified, (there were some errors which were sorted out with the help of Internet Forums & Books), then is uploaded to the Arduino.

vi) The Arduino along with the whole setup is taken out in open for better signal strength. A power bank is used as the power source.

vii) Then the coordinates are received & is observed on the LCD.

The Experimental setup looked like the following:

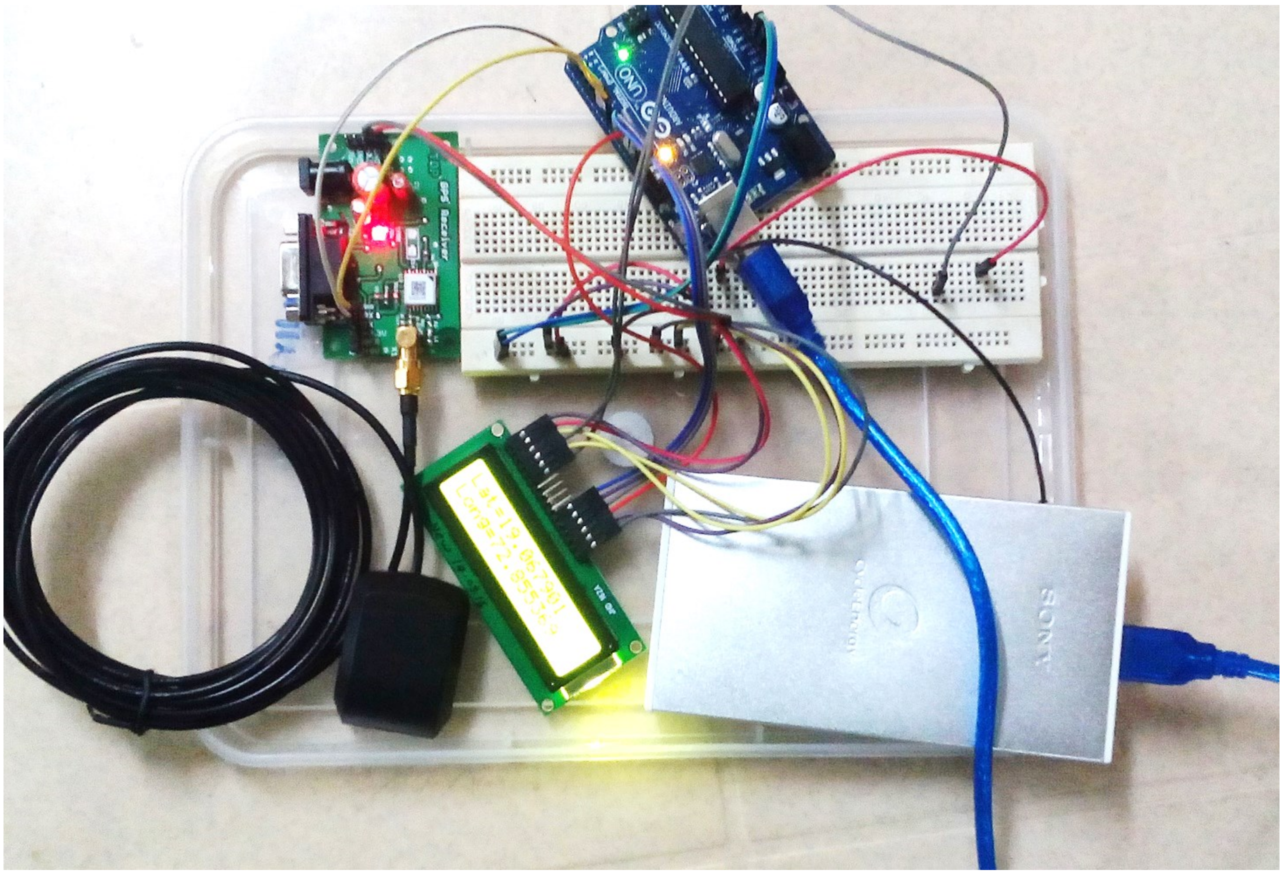

***Figure 4**:* Experimental Setup (Photograph of Actual setup)

Program (Using ARDUINO Software):

The Source Code & some of the brief explanation of commands used are given in following lines:

```
/*

 Global Positioning System (GPS) & Displaying the coordinates on LCD

  The circuit connections:

 * LCD RS pin to digital pin A2
 * LCD Enable pin to digital pin A4
 * LCD D4 pin to digital pin 9
 * LCD D5 pin to digital pin 10
 * LCD D6 pin to digital pin 11
 * LCD D7 pin to digital pin 12
 * LCD R/W pin to ground

 * GPS Pin TX (Data Transfer) to digital pin 6
 * GPS Pin RX (Data Receiver) to digital pin 7

 * Ends to +5V and ground

 The Original Program can be found in Internet of Things Blog

 The Libraries were found in GitHub Repositories

 The Program is modified by DWITI KRUSHNA DAS

 */


#include <SoftwareSerial.h> //Including the SoftwareSerial Library
#include <LiquidCrystal.h>  //Including the LCD Library

#include "TinyGPS.h"       // Including the TinyGPS Library for the GPS
from local folder

TinyGPS gps;  // defining a funtional varible

int unoRxPin = 6; // connected to TX pin of the GPS
int unoTxPin = 7; // connected to RX pin of the GPS
SoftwareSerial ss(unoRxPin, unoTxPin);    // Putting the RX & TX Pins to
digital pins through library files

LiquidCrystal lcd(A2, A4, 9, 10, 11, 12);

long startMillis;     // Defining the time taken to Search the signal
long secondsToFirstLocation = 0; // Initializing the searching time to 0
sec
```

```
void setup()
{
  Serial.begin(9600);   // Initializing the Serial Monitor
  ss.begin(9600);   // Initializing the LCD
  lcd.begin(16,2); // Columns, Rows
  lcd.clear();  // start with a blank screen
  startMillis = millis();     // Initializing the Time taken to search
Signal
}

void loop()
{
  bool newData = false;   // Getting the present status of the Data
recieved
  unsigned long chars = 0;
  unsigned short sentences, failed;

  // For one second GPS data is parsed and some key values is reported
  for (unsigned long start = millis(); millis() - start < 1000;)
  {
    while (ss.available())    // Checking if the data is received
    {
      int c = ss.read();    // Reading the data if available
      ++chars;
      if (gps.encode(c)) // reading & verifying the new data
        newData = true;
    }
  }

  if (newData)
  {
    // We have a locked location so output the Latitude / Longitude and
time to acquire the data
    if(secondsToFirstLocation == 0){
      secondsToFirstLocation = (millis() - startMillis) / 1000;
    }

    lcd.clear();  // Start LCD with a blank screen

    float flat, flon;   // defining floating variable for isolating the
Lat & Long from the Raw Data
    unsigned long age;
    gps.f_get_position(&flat, &flon, &age);
    lcd.setCursor(0,0);         // set cursor to column 0, row 0 (the
first row)
    lcd.print("Lat=");
    lcd.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6); //
Printing Latitude on LCD
    lcd.setCursor(0,1);         // set cursor to column 0, row 1 (the
second row)
    lcd.print("Long=");
    lcd.print(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);  //
Printing Longitude on LCD
  /*
    lcd.setCursor(0,2);
    lcd.print("Acquire Time=");
    lcd.print(secondsToFirstLocation); // Printing the Time elapsed to
acquire the location on LCD
    lcd.print("s");
    */
  }

  if (chars == 0){
    // Error Message, in case no data is received or wrong connection
    lcd.setCursor(0,0);           // set cursor to column 0, row 0 (the
first row)
    lcd.print("No GPS: Check Wiring!");
  }
  else if(secondsToFirstLocation == 0){
    // If it has received some chars but not yet got a fix then it'll
indicate searching and elapsed time
    lcd.clear();  // start with a blank screen
    long seconds = (millis() - startMillis) / 1000;     // counting the
seconds elapsed
    lcd.setCursor(0,0);           // set cursor to column 0, row 0 (the
first row)
    lcd.print("Searching ");
    for(int i = 0; i < seconds % 4; ++i){
      lcd.print(".");
    }

    lcd.setCursor(0,1);
    lcd.print("Elapsed time:");
    lcd.print(seconds);
    lcd.print("s");
  }
}
```

Observations & Result:

During the initial stages the GPS did not receive any data, most probably due to weak or no signal from the satellites.
But finally, it worked. I was working in my Hostel whose coordinates according to a plugin integrated with Google Maps of my Phone are as follows:

Latitude: 19.068134 N
Longitude: 72.855018 E

The Device (made in the project) shows the coordinates obtained as:

Latitude: 19.067758 N
Longitude: 72.855583 E

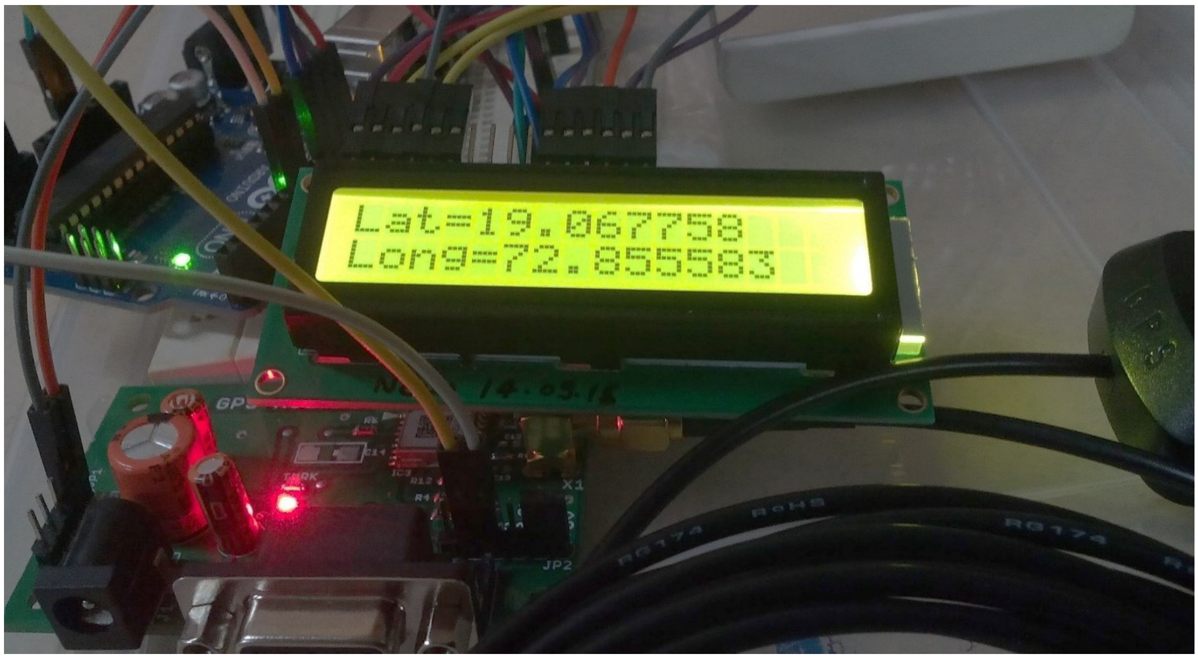

***Figure 5**:* Device showing the GPS Coordinates

Percentage Error in the determination of coordinates using my device (considering better accuracy of the GPS of my Phone):

In determination of Latitude:
∴ The percentage error is,
For % error = (19.068134 - 19.067758 /19.068134) × 100 = 0.001971 %

In determination of Longitude:
∴ The percentage error is,
For % error = (72.855583 - 72.855018 /72.855018) × 100 = 0.000775 %

We can clearly see that the difference between the data from the device & phone is less than 0.001%, so therefore we can say that the GPS device made is quite accurate for general local navigation purposes.

Thus, the project was successful in making a GPS device.

Discussion:

There were few programming / hardware related errors I encountered, which I shall discuss.

Error "**avrdude stk500_getsync(): not in sync resp=0x30 error**"

This problem or error can come due to various reasons like wrong connections or incompatible drivers & ports, etc. But in my case, it came due to my initial connection of the RX & TX pins of GPS directly to the Pin 0 & 1 of Arduino UNO respectively. This connection will not allow proper uploading of the program to Arduino while being connected to GPS Receiver through those pins. As because Arduino UNO is a small board, it doesn't have dual RX & TX ports by default (although the digital pins can be assigned through programming & that's what I did.)

So, I designated Arduino Digital Pin 6 to RX & Pin 7 to TX through the following lines in the code

```
int unoRxPin = 6; // connected to TX pin of the GPS
int unoTxPin = 7; // connected to RX pin of the GPS
SoftwareSerial ss(unoRxPin, unoTxPin);    // Putting the RX & TX Pins
to digital pins through library files
```

This step solved the problem.

Error "**redefinition of void setup()**"

This problem arose because of two programs being uploaded at once due to opening of 2 .ino files in the same window (in separate tabs), i.e., both programs were in same folder. Also, there were conflicting libraries.

At initial stage I tried displaying the coordinates on the Serial Monitor, but due to low signal (I was inside a 6-Storey Building) it couldn't receive proper information. So, I got my power bank & went outside to get the signal. And that worked & the LCD displayed the coordinates as shown in the photograph above.

Inference:

From the experience & knowledge gained during the project was very good. The satisfaction of making a successfully working & with quite accuracy GPS Device is really great.

I can say that the Project was successful.

References:

- Arduino Official Website (http://playground.arduino.cc/) [For Basic Idea]
- Exploring Arduino by Jeremy Blum [For Basic techniques]
- Internet of Things Blog by David Houlding (http://davidhoulding.blogspot.in/) [For Code]
- GitHub (https://**github**.com/) [For the libraries]
- Arduino Forum [for referring the errors]
- Wikipedia [General Information]