A Report on

GPS BASED UNMANNED VEHICLE

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE - PILANI
K.K. BIRLA GOA CAMPUS

Submitted in partial fulfilment for the course

Design Oriented Project (INSTR F376)

Semester II, 2015-2016

Under the guidance of
Dr. Anita Agrawal

By
Charanjit Nayyar
2012A8PS334G

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI, GOA CAMPUS

ZUARINAGAR, GOA – 403726

CERTIFICATE

This is to certify that **Mr. Charanjit Nayyar** bearing **ID No. 2012A8PS334G** has submitted the project report in partial fulfilment of the course **Design Oriented Project (INSTR F376)** during the academic year **2015-2016** for the requirements of the **Design Oriented Project course**.

Date: 30/04/2016 **Instructor In-charge**

ACKNOWLEDGEMENT

I would like to express my sincere thanks to my course instructor, Dr. Anita Agrawal for providing me with an opportunity to do a design oriented project under her guidance and for constantly encouraging and guiding my work from time to time.

TABLE OF CONTENT

ABSTRACT

Use of unmanned surface, air and underwater vehicles is becoming increasingly important lately. It helps us discover and explore areas where it is difficult for humans to survive due to harsh climate etc. In this project we navigate a surface bot with the help of Global Positioning System (GPS). GPS coordinates of the destination are entered and the bot successfully tries to reach it. Besides GPS, various other components such as magnetometer, motors, Bluetooth etc. are interfaced with Arduino microprocessor to achieve the goal.

INTRODUCTION

This project was aimed to design a small self-navigated GPS navigated robot. The robot would get GPS coordinates of the final destination by the user wirelessly and navigate itself using an algorithm designed by our group.

The final design of the robot is a tri-wheeled toy car having two wheels at the back which are connected to two low power DC motors and a supporting roller ball having 360 degree freedom of motion.

COMPONENTS USED

For GPS navigation, one component is indispensable – GPS module to acquire GPS coordinates of its current location. For motion of the bot motors are required. Now either a stepper motor could be used or a low power DC could be used. A stepper motor will definitely give a better control and accuracy to the model but it would increase the complexity of the code. Though it could help in reducing the inaccuracies associated with the present location estimation of the bot because the GPS module used has an error range of 2-3 meters. Two 60 rpm low power DC motor have been used.

In our earlier designs we used USB to UART connector to give GPS coordinates to the bot. This was then changed to using a Bluetooth module to communicate with the bot.

9 volt DC batteries have been used to power the bot.

Three different approaches have been made by the team to get to the direction of motion of the bot-

1. Using a gyroscope sensor.
2. Mathematical approach using GPS coordinates.
3. Using magnetometer sensor.

The following table shows all the components used throughout the course of this project-

S.No.	Name	Power rating
1.	Arduino Mega	7-12V
2.	DC motor	9 V
3.	Bluetooth module	3.3V
4.	GPS module	3.3V
5.	DC power battery	9 V
6.	Gyroscope (mpu 6050)	5V
7.	Magnetometer	3.3V
8.	Motor Driver board	9-12 V

DC Motor

Two low power DC motor have been used. There were several options in the geared motor.

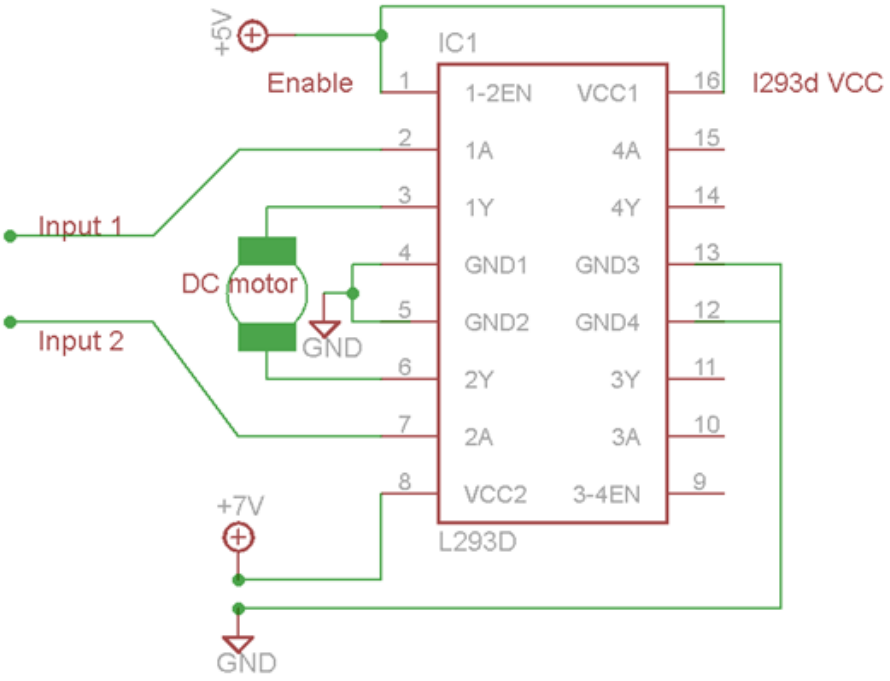
A motor having a higher rpm rating was also available, but the higher rpm motor working on same power rating would be able to deliver lesser torque.

Due to the weight of the components used (especially batteries) the motors need to be have higher torque.

Motor Driver

L293d motor driver board has been used in the project to drive the two motors. L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. It is a 16-pin IC which can control a set of two DC motors simultaneously in any direction.

The following figure shows the circuit diagram of a typical L293d motor driver



It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. As you know voltage need to change its direction for being able to rotate the motor in clockwise or anticlockwise direction, hence H-bridge IC are ideal for driving a DC motor.

In a single l293d chip there two h-Bridge circuit inside the IC which can rotate two dc motor independently. Due its size it is very much used in robotic application for controlling DC motors. Given below is the pin diagram of a L293D motor controller.

There are two Enable pins on l293d. Pin 1 and pin 9, for being able to drive the motor, the pin 1 and 9 need to be high. For driving the motor with left H-bridge you need to enable pin 1 to high. And for right H-Bridge you need to make the pin 9 to high. If anyone of the either pin1 or pin9 goes low then the motor in the corresponding section will suspend working. It's like a switch.

Bluetooth module

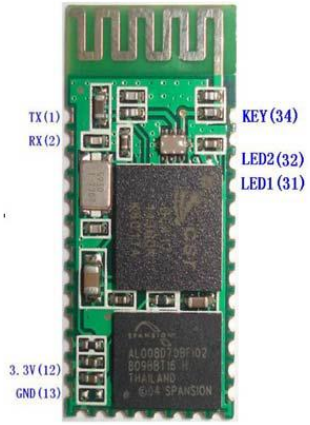
HC-05 serial Bluetooth module has been used as an interface between the Arduino Mega development board and the computer.

The following table shows some features about the Bluetooth module (HC-05)

Power supply	3.3 V, 50 Ma
Bluetooth protocol	V2.0 EDR
Frequency	2.4 GHz ISM Band
Modulation	GFSK(Gaussian Frequency Shift Keying)
Emission power	< 4dB, Class 2
Sensitivity	< -84 dBm at 0.1% BER
Speed	Asynchronous: 2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1Mbps

Here are some features of HC-05 serial Bluetooth module-

- **Master role**: have no function to remember the last paired slave device. It can be paired to any slave device. In other words, just set AT + CMODE=1 when out of factory. If you want HC-05 to remember the last paired slave device address like HC-06, you can set AT+CMODE=0 after paired with the other device.
- **Pairing:** The master device can not only make pair with the specified Bluetooth address, like cell-phone, computer, slave device automatically.
- Default communication baud rate 9600, 4800-1.3 M is settable.



GPS Module

ROBOKITS GPS 02 module has been used on the bot.

The following table shows some specifications of the GPS module

Chipset	MTK MT3329
Frequency	1575.42MHz
CPU	ARM7EJ-S
Dimensions(l*b*h)	16*16*6 mm
Position accuracy	Without aid: 3m 2D-RMS
Update Rate	1 Hz
Signal Output	8 data bits ,no parity, 1 stop bit
UART Interface	TTL level serial port
Voltage	3 V
DC current	<30 mA at 3 V

A GPS requires catching the signal from 4 satellites to estimate a correct location coordinate. For a practical accuracy for around 2-3 meters radius, connecting with 6-7 satellites is required.

A lot of calculations are required to be done by the ARM processor connected in the module.

The power required the signal enhancing antenna added with the power required by the heavy calculations for the ARM processor results in a lot of power consumption by the GPS module.

Gyroscope module

The gyroscope used in the project is MPU-6050 which is a product of invensense technologies. The gyroscope integrated 6-axis Motion Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. With its dedicated I2C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis Motion Fusion™ output.

The following table shows features of the chip

VDD	2.375V-3.46V
VLOGIC	1.71 V- VDD
Serial interface supported	I2C
FIFO Buffer	1024

MICROCONTROLLER- PROGRAMMING PLATFORM

A microcontroller is a small computer (SoC) on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips

There are several types of microcontrollers and programming platforms that could be used-

- ARM core processors (many vendors)
- ARM Cortex-M cores are specifically targeted towards microcontroller applications
- Atmel AVR (8-bit), AVR32 (32-bit), and AT91SAM (32-bit)
- Cypress Semiconductor's M8C Core used in their PSoC (Programmable System-on-Chip)
- Freescale Cold Fire (32-bit) and S08 (8-bit)
- Freescale 68HC11 (8-bit), and others based on the Motorola 6800 family
- Intel 8051, also manufactured by NXP Semiconductors, Infineon and many others
- Infineon: 8-bit XC800, 16-bit XE166, 32-bit XMC4000 (ARM based Cortex M4F), 32-bit TriCore and, 32-bit Aurix Tricore Bit microcontrollers[18]
- MIPS
- Microchip Technology PIC, (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24), (32-bit PIC32)
- NXP Semiconductors LPC1000, LPC2000, LPC3000, LPC4000 (32-bit), LPC900, LPC700 (8-bit)
- Parallax Propeller
- PowerPC ISE
- Rabbit 2000 (8-bit)
- Renesas Electronics: RL78 16-bit MCU; RX 32-bit MCU; SuperH; V850 32-bit MCU; H8; R8C 16-bit MCU
- Silicon Laboratories Pipelined 8-bit 8051 Microcontrollers and mixed-signal ARM-based 32-bit microcontrollers
- STMicroelectronics STM8 (8-bit), ST10 (16-bit) and STM32 (32-bit)
- Texas Instruments TI MSP430 (16-bit), MSP432 (32-bit), C2000 (32-bit)
- Toshiba TLCS-870 (8-bit/16-bit)

We chose to use Arduino Mega 2560 which is based on ATmega 2560 microcontroller.

The ATmega640/1280/1281/2560/2561 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 ATmega640/1280/1281/2560/2561 achieves throughputs approaching 1 MIPS per MHz allowing the system designed to optimize power consumption versus processing speed.

A UART connection is given on Arduino development board which can be connected to USB of a computer to upload a code and receive data.

Arduino is a software company, project, and user community that designs and manufactures computer open-source hardware, open-source software, and microcontroller-based kits for building digital devices and interactive objects that can sense and control physical device.

The Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, 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.

The following table shows specifications of Arduino Mega

Microcontroller	ATmega2560
Operating Voltage	5 V
Input Voltage(recommended)	7-12 V
Input Voltage(limit)	6-20 V
Digital I/O Pins	54(of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Length	101.52 mm
Width	53.3 mm
Weight	37 g

SUGGESTED DESIGNS AND ALGORITHMS

The basic design of which all the designs are subsets was to get GPS coordinates using GPS module. The final destination where the bot has to reach was given to the bot via Bluetooth or by directly connecting the microcontroller to the computer using USB to UART connector (in earlier designs).

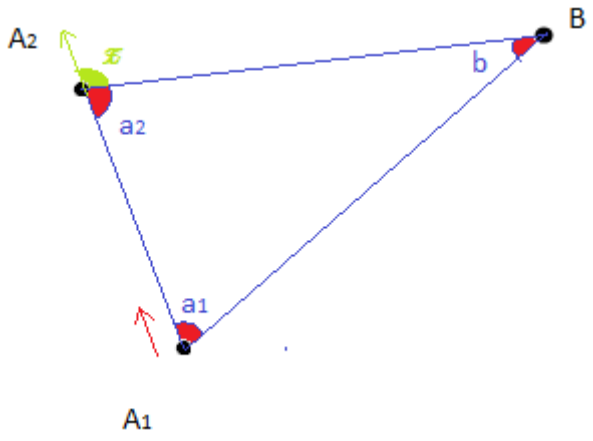
The motion and the speed of the two motors is controlled by changing the PWM output voltage of the pwm pins on the Arduino development board which are connected the motor driver board.

Four pwm pins are connected to motor driver board, when the polarity is changed the motor starts rotating in opposite direction.

Path finding algorithm

This is algorithm can be used to find the path that the bot has to follow to reach the destination coordinates given input by the user. This algorithm can be implemented in two ways-

1. Without using gyroscope
2. Using gyroscope
3. Using magnetometer



Without using gyroscope

The bot successfully receiving GPS coordinates can navigate to the final destination using this algorithm.

Let the position of the bot at say time t is A1 and is moving in direction A1-A2. Now after time t1 the bot reaches a position A2.

The final location where the bot has to reach is say B.

So after acquiring coordinates of A2, the microcontroller has stored three GPS coordintes-A1, A2 and B.

These three point, as shown in the figure, form a triangle. By knowing the 3 coordinates, angle a1, a2 and b can be found.

If angle x is positive we rotate the bot towards right direction, if the angle x is negative- we rotate the bot to left.

x>0 => rotate the left wheel (hence turn right)

x<0 => rotate the right wheel (hence turn left)

After rotating, the bot again moves forward gets a new A2 coordinate and the previous coordinate A2 becomes the new A1.Angle x is again calculated. This process the repeated over and over until either the angle x becomes zero or the bot reaches final destination.

Algorithm using gyroscope sensor

The gyroscope sensor (MPU 6050) can calculate the relative angle to the position after which the code has been uploaded on the microcontroller.

In this algorithm, the GPS receives coordinate A1 and A2, forms the triangle A1A2B.It calculates the angle x and rotates the bot until the gyroscope shows angle x as zero.

After nulling the angle x, the bot moves forward and reaches the final destination.

This helps in reducing the distance travelled by the bot. One thing must be taken into consideration- the gyroscope sensor does not provide the actual angle, i.e. It can only provide relative change in the angles.

If the bot is reset or a new code is uploaded, it would not show the same angle even on keeping it in a particular orientation.

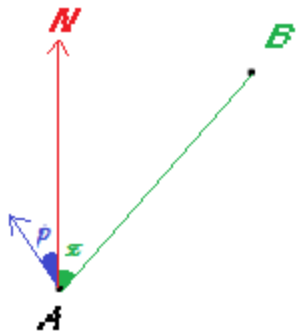
Algorithm using magnetometer

A magnetometer sensor module can calculate the absolute real angle orientation of the bot. It is able to do so by measuring the magnetic field strength of earth at any location.

The microcontroller first estimates the present direction (angle with respect to zero). Then it calculates the angle it should rotate if it has to point in the direction of the destination GPS coordinate. Then it calculates that to point in the required direction, the fastest way would be by rotating clockwise or anti-clockwise.

Once all these calculations are done, the microcontroller enables the motor driver board.

The following figure explains the above mentioned algorithm.



A is the present location coordinate of the bot, **B** is the destination coordinate of the bot. **N** indicates the direction of magnetic north pole. **p** is the angle between magnetic north of the direction of bot. **x** is the angle between magnetic north of destination coordinate.

If p+x > 0

Then rotate left wheel with 9 V

And, rotate right wheel with 4.5 V

If p+x < 0

Then rotate right wheel with 9 V

And, rotate left wheel with 4.5 V

Giving the wheel 4.5 V instead of zero gives the bot a smooth motion else it would take sharp turn and could lead to errors.

MY AREA OF WORK

With regard to my contributions to the project, the areas that I have majorly worked upon are:

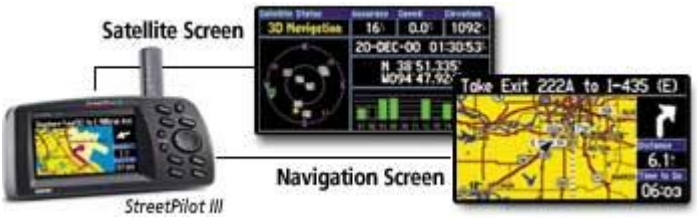
- GPS interfacing
- Magnetometer
- IR sensor

As the other team members will be focusing on discussing the latter two topics in greater depth, this report shall majorly discuss the GPS functionalities.

GPS – Overview

The Global Positioning System (GPS) is a satellite-based navigation system made up of a network of 24 satellites placed into orbit by the U.S. Department of Defense. GPS was originally intended for military applications, but in the 1980s, the government made the system available for civilian use. GPS works in any weather conditions, anywhere in the world, 24 hours a day.

GPS satellites circle the earth twice a day in a very precise orbit and transmit signal information to earth. GPS receivers take this information and use trilateration to calculate the user's exact location. Essentially, the GPS receiver compares the time a signal was transmitted by a satellite with the time it was received. The time difference tells the GPS receiver how far away the satellite is. Now, with distance measurements from a few more satellites, the receiver can determine the user's position and display it on the unit's electronic map.



A GPS receiver must be locked on to the signal of at least 3 satellites to calculate a 2-D position (latitude and longitude) and track movement. With four or more satellites in view, the receiver can determine the user's 3-D position (latitude, longitude and altitude). Once the user's position has been determined, the GPS unit can calculate other information, such as speed, bearing, track, trip distance, distance to destination, sunrise and sunset time and more.

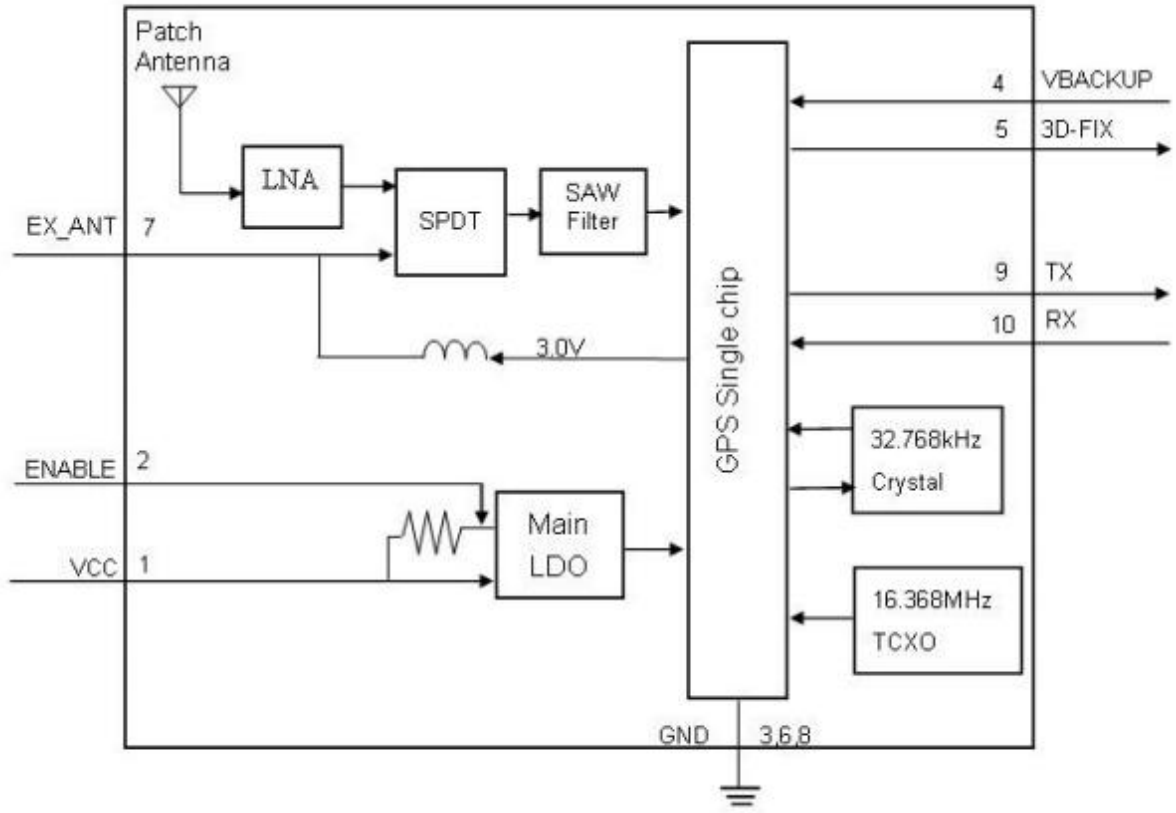
The GPS module used in our system is a revision of PA6 POT (Patch on Top) GPS Module with an extra embedded function for external antenna I/O and comes with automatic antenna switching function and short circuit protection. This POT GPS receiver uses internal patch antenna and will automatically detect and switch over to external antenna for GPS reception once a connection to external antenna is made. In addition, the module itself is protected from short circuits to the external antenna. PA6E is high in position and speed accuracy as well as being high in sensitivity with excellent tracking capabilities in urban conditions. The GPS chipset inside the module is powered by MediaTek Inc., and can support up to 66 channels. The module is suitable for every GPS-related application such as:

- Fleet Management/Asset Tracking
- LBS (location-based service) and AVL system
- Security system
- Hand-held device for personal positioning and travel navigation

Key Specifications:

- Position accuracy : <3m (without aid)
- Update Rate : 1 Hz
- Reacquisition time : < 1s
- Hot start : 1s
- Cold start : 35s
- Sensitivity (tracking): -165 dBm
- Signal output : 8 data bits, no parity, 1 stop bit
- Baud rate : 9600 bps
- UART interface : TTL level serial port

System Block



Key features for selection:

- High Sensitivity: Up to -165 dBm tracking, superior urban performances
- Low Power Consumption: 48mA @ acquisition, 37mA @ tracking
- Protection from Short Circuits to External Antenna (3V power supply or lower)
- Max. Update Rate: up to 10Hz (Configurable by firmware)

We also use an external patch antennae in tandem with our module so as to increase the satellite-finding capabilities of our module.

The GPS module transmits data by means of an **NMEA protocol**. The National Marine Electronics Association (NMEA) has developed a specification that defines the interface between various pieces of marine electronic equipment. The standard permits marine electronics to send information to computers and to other marine equipment. GPS receiver communication is defined within this specification. Once a GPS module is powered, NMEA

data is sent out of a serial transmit pin (TX) at a specific baud rate and update rate.
To configure the GPS module, you will need to also connect the RX pin of the GPS to the TX pin of the microcontroller. It is common for the microcontroller to parse the NMEA data. Parsing is simply removing the chunks of data from the NMEA sentence so the microcontroller can do something useful with the data. For example, the microcontroller might need to read only the altitude of your GPS.

*GPGGA,235317.000,4003.9039,N,10512.5793,W,1,08,1.6,1577.9,M,-20.7,M,,0000*5F*

Instead of dealing with all of this text, the microcontroller can parse the GPGGA sentence and end up with only the altitude (in meters):**1577**

The Arduino platform can parse NMEA data easily with the help of the **TinyGPS** library.

TinyGPS is designed to provide most of the NMEA GPS functionality an Arduino user would want – position, date, time, altitude, speed and course – without the large size that seems to accompany similar bodies of code. To keep resource consumption low, the library avoids any mandatory floating point dependency and ignores all but a few key GPS fields.

Usage

To use, simply create an instance of an object like this:

```
#include "TinyGPS.h"
TinyGPS gps;
```

Feed the object serial NMEA data one character at a time using the **encode()** method. (TinyGPS does not handle retrieving serial data from a GPS unit.) When **encode()** returns "true", a valid sentence has just changed the TinyGPS object's internal state. For example:

```
#define RXPIN 3
#define TXPIN 2
SoftwareSerial nss(RXPIN, TXPIN);
void loop()
{
  while (nss.available())
  {
    int c = nss.read();
    if (gps.encode(c))
    {
      // process new gps info here
    }
  }
}
```

FUTURE PLAN

The following things can be added:

- Obstacle detection to help detect and avoid an obstacle: We can incorporate either IR sensors or UV sensors to help the bot detect and correspondingly avoid path obstacles, by rerouting around such objects.
- Upgrading to a different micro-controller for eg. PIC microcontroller: The Arduino, while being a very helpful microcontroller in terms of its in-built libraries, is limited in terms of programing additional functionalities, which could be performed on higher-end boards.
- Better management of power supply: A major problem with our current model is its excessive current drawn from DC battery sources. This problem could either be tackled by introducing a higher voltage Li-Po battery, introducing rechargeable batteries, or efficient power consumption reduction at various modules where deemed excessive and can be reduced without affecting the vehicle performance.

BIBLIOGRAPHY

- http://garagelab.com/profiles/blogs/gps-guided-autonomous-robot
- https://www.sparkfun.com/pages/GPS_Guide
- http://www.engineersgarage.com/embedded/arduino/arduino-gps-interfacing-project-circuit
- Datasheets of various components used.