Path Planning for Mobile Robotics

By

Rishabh Dev Yadav

(Roll No. : 2015208)

Supervisor

Professor Vijay Kumar Gupta

Professor Aparajita Ojha

Mechanical Engineering Discipline

PDPM Indian Institute of Information Technology, Design and Manufacturing Jabalpur

Final Report

Certificate

This is to certify that the project topic, "**Path planning for Six Wheeled Multi Terrain Robot**", submitted by **Rishabh Dev Yadav** (Roll No. 2015208) in fulfilment of the PBI (Project Based Internship), during the degree of **Bachelor of Technology** in **PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur** is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted elsewhere to any other university/institute for the award of any other degree.

Prof. Vijay Kumar Gupta
Discipline of Mechanical Engineering.

PDPM Indian Institute of Information

Technology, Deign and Manufacturing, Jabalpur, Madhya Pradesh.

Prof. Aparajita Ojha
Discipline of Computer Science Engineering.

PDPM Indian Institute of Information

Technology, Deign and Manufacturing, Jabalpur, Madhya Pradesh.

Acknowledgement

Robotics is an interdisciplinary branch in which knowledge of hardware as well as software both are required. One should have experience in Mechanical, Electronics as well as Computer science.

I am very thankful to my supervisor Professor **Vijay Kumar Gupta**, Department of Mechanical Engineering, and Professor **Aparajita Ojha** PDPM IIITDM Jabalpur, for their continuous support and guidance throughout my work. As a student of mechanical engineering engineering student, it was very challenging to work with electronic components and circuit. So, a proper guidance is always required.

In addition to this, Mechatronics lab of institute helps me every time by proving a proper workplace and electronic components. I am also very thankful to everyone who have helped me directly or indirectly in PBI.

Date: …/…/…… **Rishabh Dev Yadav**

Abstract

Mars rover is the one was successful robot, which was able to move in unknown area with all kind of terrain. So, in addition to this an Six wheeled Multi terrain robot was designed for autonomous rover, which have a combination of caterpillar differential steering system and an high torque servo steering mechanism.

The GPS module and IMU helps the robot to move towards it target while ultrasonic sensors were used for obstacle detection and avoidance. The robot will continuously move toward its target latitude and longitude and avoid the obstacles in its local known area. GPS module give us targeted heading and distance where magnetometer will give current heading. The difference will decide left or right turn of robot.

Since with help of GPS robot can reach a goal in perimeter of 3m only. after reaching in periphery ultrasonic sensor is used to detect the distance and angle of target pole. And then robot move toward that pole with help of ultrasonic senor only.

Later on, Tangent bug approach is tried to follow for obstacle avoidance in its local known area. Tangent bug method is quite simple and finds the shortest possible path. At the end robot was able to reach its target pole successfully and also able to avoid the single obstacle detected in its path.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mobile robotics is field of engineering in which we study the autonomous control of robot. Six wheeled multi-terrain robot-based rocker and bogie mechanism contains six wheeled and having a capability to two kind of turning differential turning as well as High torque motorized steering system that allow robot a better turning. The aim to move the robot in all kind of terrain and uneven surface. Such robot is very useful in remote location or unknown environment where robot need o rover automatically at every king of surface. due to this, Mars rover or curiosity rover was launched by NASA was able to move on mars, was also based on rocker bogie mechanism.

1.1 Literature Review

Shang [1] discussed a 6 wheeled robot with an active adaptive suspension system on it and it contain a geared body with differential system. The robot has a rocker and bogie which is able to move on rough surface.

Moore [2] proposed six wheeled robot with omnidirectional wheel, which is able to move in all direction. He also discussed the autonomous path planning for path planning. Cortner [3]present a six wheeled robot which has a steering control system using motor for all kind of terrain. Nagatani [4] propsed a six wheeled robot which is able to move uneven surface and inclined plane automatically. The robot was able to move automatically according to different kind of surface.

Dung Duong Quoc[5] explained a way for fusion of accelerometer and gyroscope using complimentary filter for calculation of roll, pitch and yaw. He also discussed the Low pass filter and high pass filter for accelerometer and gyroscope.

Professor Othman bin Sidek [6] explained Kalman filtering technique is employed to fuse data obtained from gyroscope and accelerometer in an IMU. Simulations are carried out to find the effect of measurement noise and process noise on
estimation error.

Pengfei gui [7] compares complimentary and Kalman filter for MEMS based IMU and calculated eulers angle in different condition.Sebastian O.H. Madgwick[8] discussed a Madgwick Quaternion based AHRS system which was bale to calculate Euler angle as well as linear acceleration.

Sławomir ROMANIUK [9] explained the fusion of GPS and IMU in which he explained GPG works in absolute frame of reference while IMU work in local frame of reference. So the vale of IMU needs to be convert in NED frame of reference.

Ray-Shine-Run [10] discussed a autonomous robot using GPs and magnetometer and discussed its result. Ajeet Gaur [11] also made a robot using GPs and IMU for autonomous rover.
N. Nath [12] explained a tangent bug approach using ultrasonic senor and its limitation. Results of a simulation of this algorithm in the ideal scenario of the robot being equipped with infinite resolution range sensors are presented.

In the NPTEL lecture of control engineering by IITM it has been explained in detailed about the IMU and their working and error correction. Also, complimentary filter and its result has been discussed there in discrete form.

scottlobdell.me, this is an robotics hobbyist, who has explained the Quaternion Update AHRS system and also explained his project. lukagabric.com is a robotics hobbyist who has build GPS guided autonomous rover robot.

1.2 Aim of Project

The aim is to develop a method for the autonomous rover of SW-MTR in unknown area where robot will move to a targeted position while detecting and avoiding the obstacle it gets in front of it. The unknown will be multi terrain and obstacle could be static or dynamic. The main focus is to use low computation power and low power consumption. For such planning, various sensors and electronic components like GPS, GY87, ultrasonic sensor, motor driver, servo motor has been studied individually. Few mechanical design constraint and their alternative solutions has been proposed.

1.3 Organization of the Project

Chapter 1 discusses general information and introduction and short literature review of the project.

Chapter 2 discusses mechanism design for proposed robot. Parameters considered for the robot design and robot structure are discussed. Advantages of proposed robot design in terms of flexibility and terrain ability discussed.

Chapter 3 is about Different kind of possible movement and its manual controlling via Arduino IDE serial monitor and Play station remote has been discussed in chapter 3.

In chapter 4, detailed sensor fusion algorithms has been discussed. GY87, NMEA parsing of GPS and PID control has been described. Kalman filter, Complimentary filter and Quaternion AHRS method has been discussed in detail.

Chapter 5, discusses the path planning of robot towards its targeted latitude and longitude. Few problems like GPS position in accuracy and their alternative solution has been discussed.

Chapter 6 is short description of obstacle avoidance where robot was able to detect and void a single obstacle successfully.

At the end conclusion and future has been discussed and appendix is attached for additional detail.

Chapter 2

Manual Control of Robot

Six wheeled multi terrain robot was able to move on all kind of surface so, it was very useful for mobile robotics for surveillance in unknown area where robot can move automatically. Such robot is very useful in remote location or unknown environment where robot need o rover automatically at every king of surface.

2.1 Kind of Motion Involved

To understand the movements of robot, consider all 3 wheels of left side as single left wheel and all 3 wheels of right side as single right wheel. Now maintain the speed and direction of left side motors A and B equal to each other, similarly for right side motors.

- Linear motion, it will provide
  - Forward motion
  - Backward motion
- Rotational motion
  - Clockwise rotation about its vertical axis.
  - Anticlockwise about its vertical axis.
- Differential turning due to speed difference in left and right side motors.
  - Turning left
  - Turning right
- Steering using DC servo motor mounted on top of body using worm-spur gear
  - Steering Clockwise
  - Steering Anticlockwise

2.2 Vector Representation of Movements

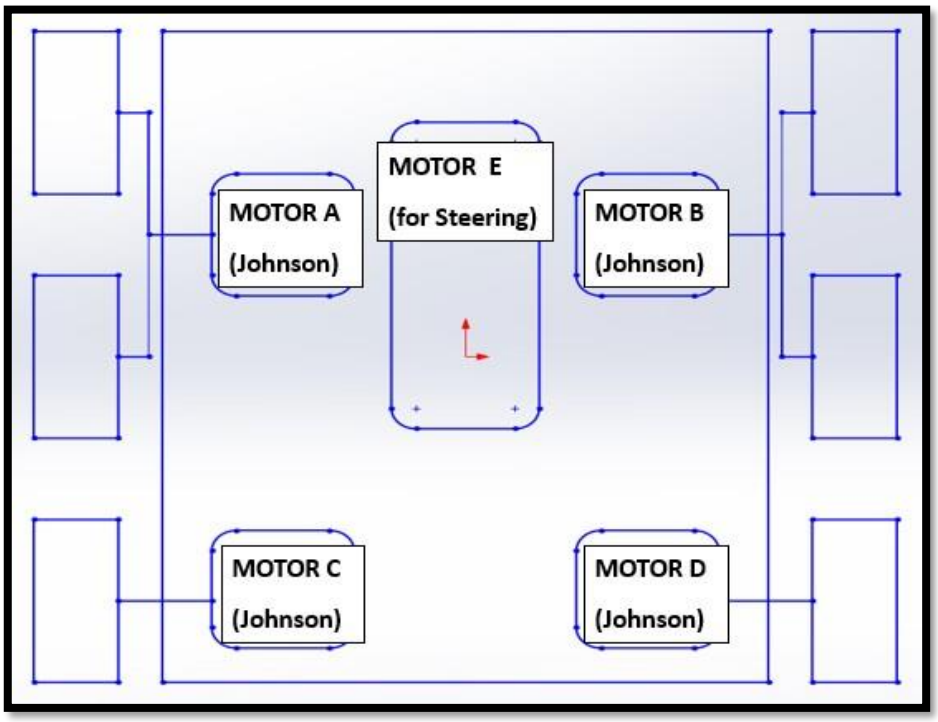


Fig 2.1 SW-MTR motors and wheel representation

Table 2.1 Vector representation chart for various movements

	MOVEMENTS	MOTOR A	MOTOR B	MOTOR C	MOTOR D	MOTOR E
1.	FORWARD	↑	↑	↑	↑	-
2.	BACKWARD	↓	↓	↓	↓	-
3.	CLOCKWISE	↑	↓	↑	↓	-
4.	ANTICLOCKWISE	↓	↑	↓	↑	-
5.	FORW. LEFT TURN	↑	↑	↑	↑	-
6.	FORW. RIGHT TURN	↑	↑	↑	↑	-
7.	LEFT STEERING	-	-	-	-	↑
8.	RIGHT STEERING	-	-	-	-	↓

2.3 Manual Controlling of Robot

2.3.1. Control with Arduino IDE Serial Monitor

All these motions can be controlled by serial monitor of Arduino IDE.

Table 2.2 Keyword to control different motion by Arduino IDE Serial Monitor

S. No.	Keyword	Motion
1	F	FORWARD
2	B	BACKWARD
3	L	LEFT TURN BY CATERPILLAR DIFFERENTIAL
4	R	RIGHT TURN BY CATERPILLAR DIFFERENTIAL
5	C	CLOCKWISE ROTATION ABOUT Y AXIS
6	A	ANTI-CLOCKWISE ROTATION ABOUT Y XIS
7	X	LEFT STEERING
8	Y	RIGHT STEERING
9	Z	STOP STEERING
10	S	STOP ALL KIND OF MOVEMENT

2.3.2 Control with Arduino and PlayStation Remote

Arduino PS shield has been used to control the robot. For manual control PS easy to control. A more Convenient method is control robot using PlayStation. Few addition features will be:

- Two different command of motion can be used simultaneously. For eg: backward and left steering.
- Overall speed of robot can be controlled by single press of button.
  - Triangle button will increase PWM of motors upto 255, simultaneously.
  - Cross button will decrease PWM of motors upto 100, simultaneously ❖ Differential turning angle can be adjusted according to sharpness of turn.
  - PS_RIGHT1 will increase the difference between left and right motors up to 100pwm.
  - PS_RIGHT2 will decrease the difference between left and right motors by 50 PWM

✞ Control of DC servo motor is more convenient using joystick.

Table 2.3 PS Buttons and their function in controlling of the robot

S. No.	Play Station Button	Function
1.	PS2_UP	FORWARD
2.	PS2_DOWN	BACKWARD
3	PS2_RIGHT_1	INCREASE TURNING ANGLE
4	PS2_RIGHT_2	DECREASE TURNING ANGLE
5	PS2_LEFT	TURN LEFT
6	PS2_RIGHT	TURN RIGHT
7	PS2_CROSS	DECREASE MOTOR SPEED
8	PS2_TRIANGLE	INCREASE MOTOR SPEED
9	PS2_JOYSTICK_RIGHT_X	STEERING LEFT/RIGHT(WORM SECTOR)
10	PS2_ JOYSTICK_LEFT_X	CLOCK/ANTICLOCKWISE ROTATION
11	NO BUTTON	STOP ALL MOTION

Fig 2.2 Buttons' detail of Play Station Remote

Table 2.4 Default name of different buttons in Arduino Library

PS2_SELECT	PS2_JOYSTICK_LEFT	PS2_JOYSTICK_RIGHT	PS2_START
PS2_LEFT	PS2_RIGHT	PS2_UP	PS2_DOWN
PS2_LEFT1	PS2_LEFT2	PS2_RIGHT1	PS2_RIGHT2
PS2_TRIANGLE	PS2_CIRCLE	PS2_CROSS	PS2_SQUARE

2.4 Current Drawn on Different Slope

The robot moves according to serial monitor command. The main disadvantage of serial monitor control is only one command is executed by robot. Hercules 8Amp motor driver has been used. Motor A and C controlled by one motor driver and motor B and D are controlled by one motor driver. The overall current dawn by robot has been noted down for different PWM and slope. A constant 12V DC has been supplied. The friction between the wheel and surface will play vita role. Here a wooden ply has been used as surface for all data. The current drawn fluctuate continuously.

For slope=0 degree, following data has been observed:

Table 2.5 PWM vs Current Drawn Table

	PWM	**CURRENT VARIATION**	**AVERAGE CURRENT**
1.	255	0.63A-0.75A	0.69A
2.	200	0.54A-0.65A	0.59A
3.	150	0.50A-0.62A	0.54A
4.	100	0.42A-0.54A	0.46A
5.	50	0.28A-0.32A	0.30A

For PWM=255(max), following data has been observed:

Table 2.6 Inclined slope vs Current Drawn

	SLOPE	CURRENT VARIATION	AVERAGE CURRENT
1.	8 degrees	0.72A-0.88A	0.82A
2.	15 degrees	0.86A-1.06A	0.98A
3.	20 degrees	0.98A-1.14A	1.02A
4.	23 degrees	1.04A-1.24A	1.12A
5.	28 degrees	1.17A-1.32A	Unable to climb

The above current drawn table shows that current drawn by robot increases on increase the load on robot. The current drawn increases as the PWM increases. Also drawn increases as the slope increases. The max current drawn is 1.34A when robot is unable to move due to obstacle. The maximum inclined slope is 27 degree, above it robot is unable to climb on slope.

Chapter 3

Inertial Navigation system

3.1 GY87: Inertial Measurement Units (IMU)

IMU 9DoF Fusion is a 9 Degrees of Freedom sensor board created using Invensense's MPU6050 and Honeywell's HMC5883L. Orientation sensors have become common these days with increasing use in Robotics and RC Applications. These are used for object stabilization as well as detecting the orientation of the object in real time in three-dimensional space.

- **MPU6050:** MPU6050 sensor module is complete 6-axis Motion Tracking Device. It combines 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor all in small package. Also, it has additional feature of on-chip Temperature sensor. It has I2C bus interface to communicate with the microcontrollers.

- **HMC5833L:** The HMC5883L 3-Axis Compass module can measures magnetic

  fields in three directions: X, Y, and Z. It can also be used as a simple compass to find the earth's magnetic north.

- **BMP180:** The BMP180 Breakout is a barometric pressure sensor with an I^2C

  ("Wire") interface. Barometric pressure sensors measure the absolute pressure of the air around them. This pressure varies with both the weather and altitude.

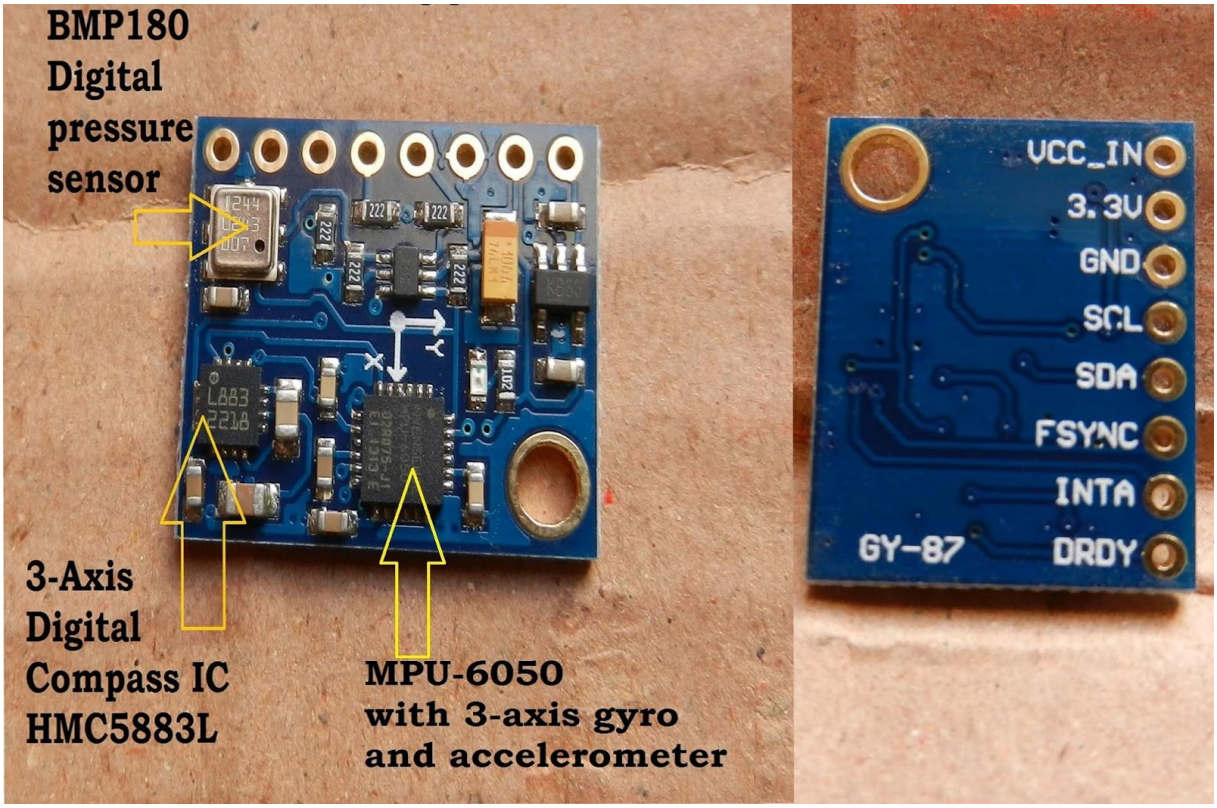


Fig 3.1 GY 87 IMU module

Accelerometer give us acceleration in ax, ay, az and gyroscope give us gx, gy, gz and magnetometer give us mx, my, mz.

Accelerometer give is linear acceleration. Accelerometer can measure roll and pitch only. Since a constant gravity acceleration always work downward so it must be removed but sometime this is used to a reference. Accelerometer is unable to measure yaw. Accelerometer is very susceptible to very small changes in environment, even in static condition it observe minor vibrations. So an accelerometer is passed through Low pass filter to remove "jitter", a high frequency undesirable data. So, for very low speed roll and pitch value by accelerometer is more accurate in compare to gyroscope.

Gyroscope measure angular velocity. Gyroscope can measure roll, pitch and yaw. To calculate roll, pitch and yaw using gyroscope, the value obtained by sensor need to integration. Actually, integration lead to summing of the small error and result in "drift". So gyroscope value need to be passed through Low pass filter to cut the low frequency noise. So, gyroscope is accurate in computing the highspeed dynamics.

$$\dot{\omega}x\_measured = \dot{\omega}x\_true + noise$$

Magnetometer used to measure earth magnetic field strength in x,y,z direction in gauss. But before using magnetometer, always calibrate it because it gets affected by its environment magnetism. Take median of x,x,z value over a small period of time where calibrated and apply offset to get calibrated values. Magnetometer is very importation in computation of yaw and orientation because it provide reference using earth's magnetic field.

"So accelerometer + gyroscope can give us accurate roll and pitch.

Magnetometer + gyroscope can help us in calculating yaw."

Formula to calculate roll, pitch and yaw using accelerometer Ax, Ay and Az is give by:

$$\rho = \arctan\left(\frac{A_X}{\sqrt{A_Y^2 + A_Z^2}}\right)$$

$$\phi = \arctan\left(\frac{A_Y}{\sqrt{A_X^2 + A_Z^2}}\right)$$

$$\theta = \arctan\left(\frac{\sqrt{A_X^2 + A_Y^2}}{A_Z}\right)$$

3.2 SENSOR FUSION

1. Complimentary Filter
2. Kalman Filter
3. Quaternion Update AHRS method

3.2.1 Complementary filter

Accelerometer gives slow moving data while gyroscope is very accurate for fast moving data. So, accelerometer is useful when there is a static condition and gyroscope is useful when there is an dynamic condition. Pass the accelerometer data through low pass filter and gyroscope data through high pass filter and overall result by combining both will give final value which will be accurate in static and dynamic conditions. The sum of LPF and HPF, the frequency response add up to 1.

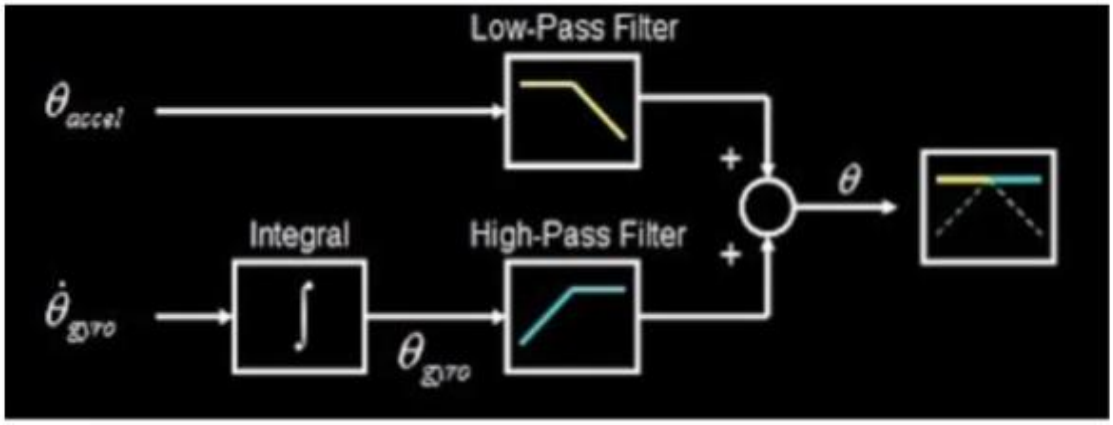


Fig 3.2 Block Diagram of Complimentary Filter [13]

Equation for low-pass filter:

y[n]=(1-alpha)*x[n] + alpha*y[n-1] //use this for angles obtained from accelerometers

x[n] is the pitch/roll/yaw that you get from the accelerometer

y[n] is the filtered final pitch/roll/yaw which you must feed into the next phase of your program

Equation for high-pass filter:

y[n]=(1-alpha)*y[n-1] + (1-alpha)*(x[n]-x[n-1]) //use this for angles obtained from gyroscopes

x[n] is the pitch/roll/yaw that you get from the gyroscope

y[n] is the filtered final pitch/roll/yaw which you must feed into the next phase of your program

n is the current sample indicator.

A way of implementing a complementary filter:

angle = (1-alpha)*(angle + gyro * dt) + (alpha)*(acc)

If alpha is high the final value depends upon accelerometer data while if value of alpha will be low then final value depends on gyroscope. Generally alpha is around 0.8-0.9

3.2.2 Kalman Filter

The Kalman Filter is also known as a Linear Quadratic Estimator. It work in two steps Predict and Update. It helps in fusion of two sensor data giving same value.

System Dynamics

A linear state space form of discrete time :

$$\vec{x}_{t+1} = \mathbf{A} \cdot \vec{x}_t + \mathbf{B} \cdot \vec{u}_t$$

$$\vec{y}_{t+1} = \mathbf{C} \cdot \vec{x}_{t+1}$$

Where $\vec{x}_t$ is the system's state vector

$\vec{u}_t$ is the input vector **A**, **B**, and **C** are the system matrices

To remove the drift of gyro bias at every moment in time, the accelerometer position estimation will be used. And it will be subtracted at regular interval of time, giving us combine data of accelerometer and gyroscope data.

State vector, input vector, and measurement vector:

$$\vec{x}_t = \begin{bmatrix} \hat{\phi}_t \\ b_{\hat{\phi}_t} \\ \hat{\theta}_t \\ b_{\hat{\theta}_t} \end{bmatrix}$$

$$\vec{u}_t = \begin{bmatrix} \dot{\phi}_{G_t} \\ \dot{\theta}_{G_t} \end{bmatrix}$$

$$\vec{z}_t = \begin{bmatrix} \hat{\phi}_{Acc_t} \\ \hat{\theta}_{Acc_t} \end{bmatrix}$$

Where b_{ϕ_t} is the gyro bias at associated with our estimate $\hat{\phi}$. This leads us to our general state-space form:

$$\vec{x}_{t+1} = \begin{bmatrix} 1 & -\Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \vec{x}_t + \begin{bmatrix} \Delta t & 0 \\ 0 & 0 \\ 0 & \Delta t \\ 0 & 0 \end{bmatrix} \vec{u}_t$$

$$\vec{y}_{t+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \vec{x}_{t+1}$$

Kalman Filter Equations

After defining the system in state-space form, put it into the Kalman Filter equations. These can be put into two groups: *Prediction* and *Update*.

Prediction

$$\vec{x}_{t+1} = \mathbf{A} \cdot \vec{x}_t + \mathbf{B} \cdot \vec{u}_t$$

$$\mathbf{P} = \mathbf{A} \cdot \mathbf{P} \cdot \mathbf{A}^T + \mathbf{Q}$$

Update

$$\tilde{y}_{t+1} = \vec{z}_{t+1} - \mathbf{C} \cdot \vec{x}_{t+1}$$

$$\mathbf{S} = \mathbf{C} \cdot \mathbf{P} \cdot \mathbf{C}^T + \mathbf{R}$$

$$\mathbf{K} = \mathbf{P} \cdot \mathbf{C}^T \cdot \mathbf{S}^{-1}$$

$$\vec{x}_{t+1} = \vec{x}_{t+1} + \mathbf{K} \cdot \tilde{y}_{t+1}$$

$$\mathbf{P} = (\mathbf{I} - \mathbf{K} \cdot \mathbf{C}) \cdot \mathbf{P}$$

$\mathbf{K}$ is known as the Kalman gain, which is optimally chosen depending on what the filter thinks is giving a more reliable estimate of the state – the measurement or the system dynamics model.

$\mathbf{P}$ is the error covariance matrix and is initially set by us and then updated by the Kalman Filter. If we are not sure if our guess of the initial state is correct, we will increase the values in this matrix.

$\mathbf{Q}$ and $\mathbf{R}$ are covariance matrices of the process and measurement noise respectively.

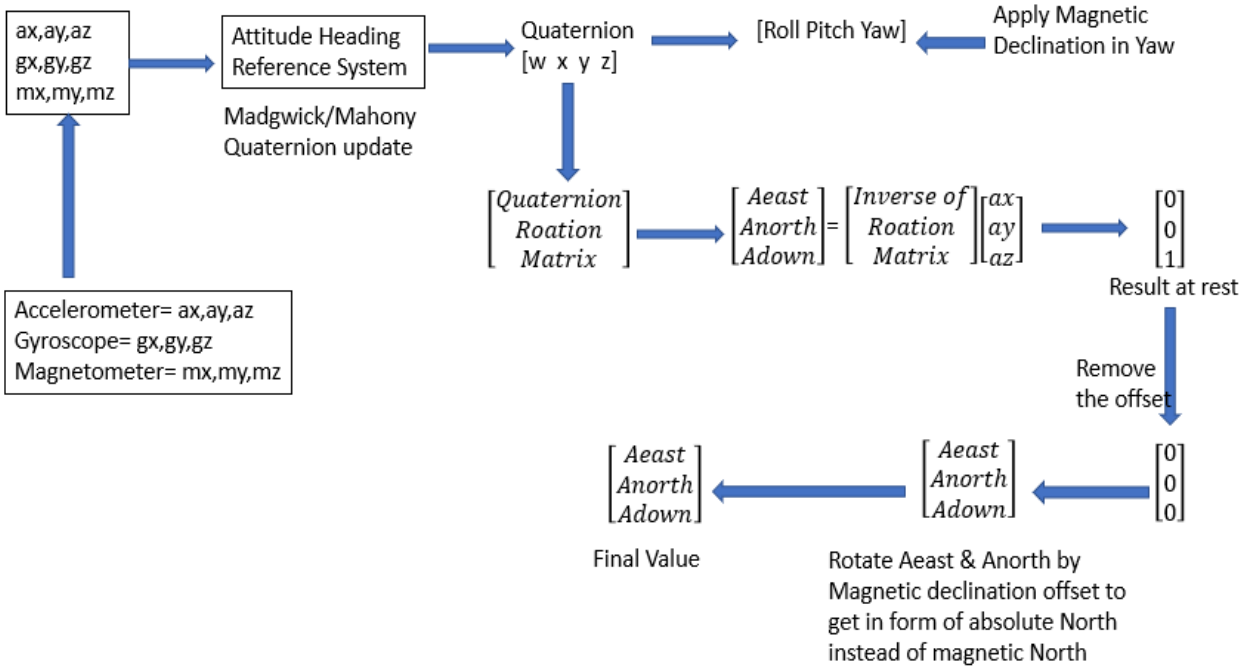
Generally, $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$ are diagonal matrices.

3.2.3 Quaternion AHRS Method

IMU give value in local frame of reference and this method has many advantages over Kalman and Complimentary filter. It avoids the Gimbal lock in 3D rotation also compatible for discrete system. It can give roll pitch yaw as well linear acceleration in absolute frame of reference (NED, north east and down). Madgwick or Mahony filter is used to get Quaternion values. [14]

Mahony is more accurate for very low processor, but Madgwick can be more appropriate for with 9 DOF systems at the cost of requiring extra processing power.

Here, Mahony fusion algorithm is used since it most relevant with 9 DOF where magnetometer is present. Both Madgwick and Mahony AHRS is developed for IMU which work very good with the discrete system. Here Both filters are used as a black box and predefined libraries has been downloaded from github.[27]



Flowchart explaining to get roll, pitch and yaw and linear acceleration in absolute frame of reference

3.3 Result

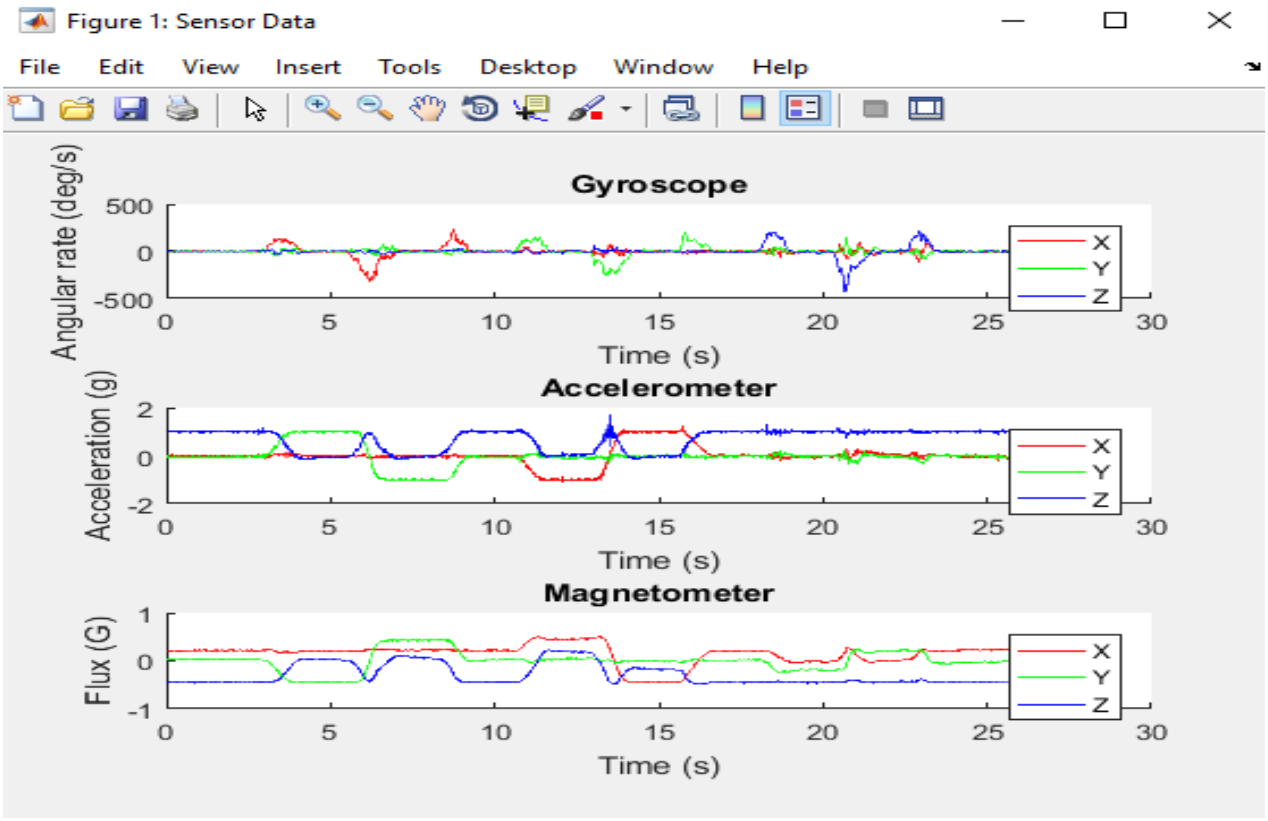


Fig 3.3 Sensor value from gyroscope, accelerometer and magnetometer

Fig 3.3 show raw value of ax, ay, az by accelerometer, gx, gy, gz by gyroscope, mx, my, mz by magnetometer. The raw values in x,y,z in local frame by accelerometer, gyroscope and magnetometer has been stored in excel sheet by TeraByte software from serial monitor of Arduino IDE using GY87 IMU sensor and values are plotted in MATLAB when sensor is rotated. Gyroscope provide angular rate, accelerometer provide linear acceleration and magnetometer measure earth magnetic field.

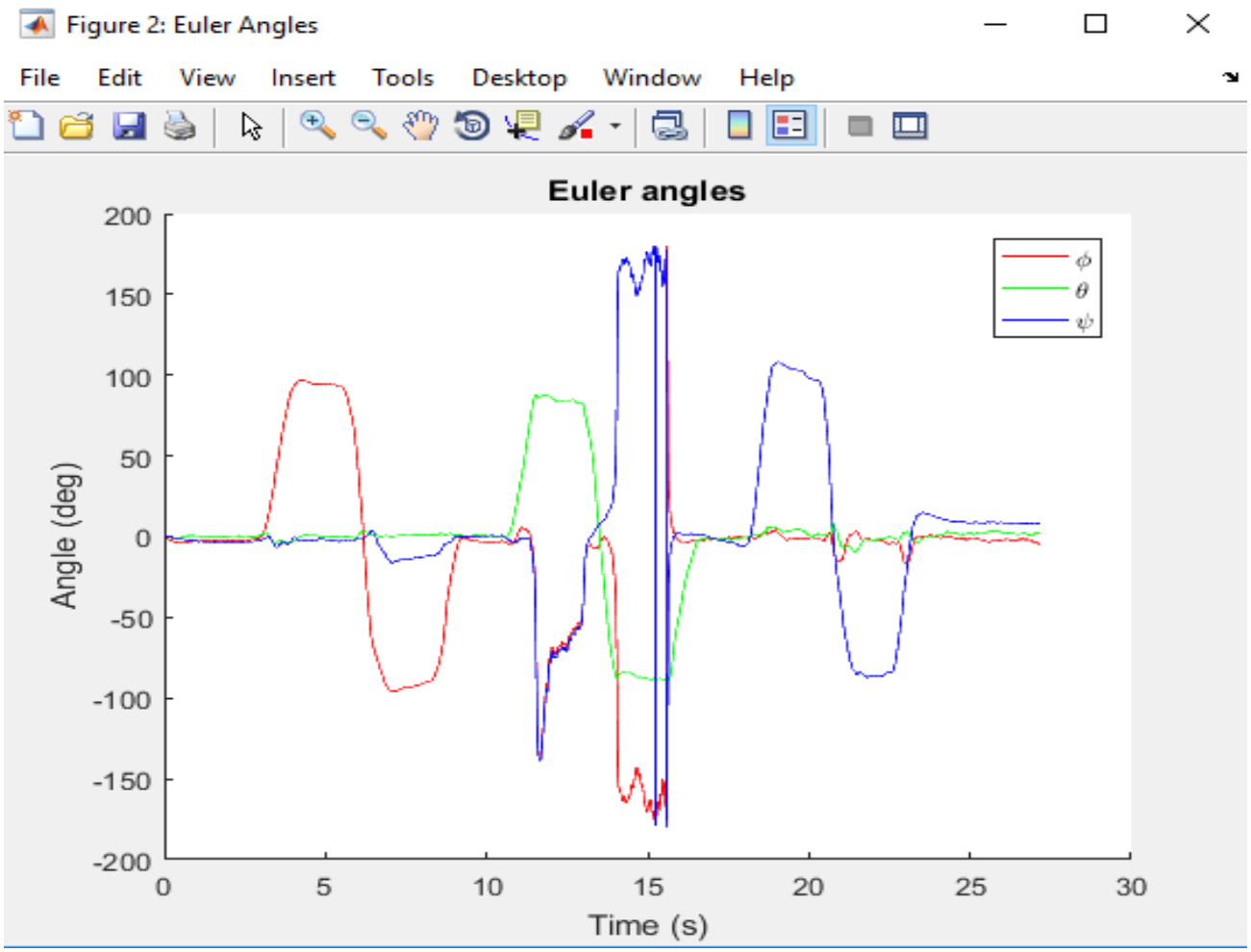


Fig 3.4 Value of Roll, Pitch and Yaw from Sensor data

Fig 3.4 shows the plot of Euler's angle in 3 directions (x, y, z) plotted in MATLAB using the raw data of accelerometer gyroscope and magnetometer shown in Fig 3.3. A continuous fluctuation in roll pitch and yaw can observed when no filter is applied in it.

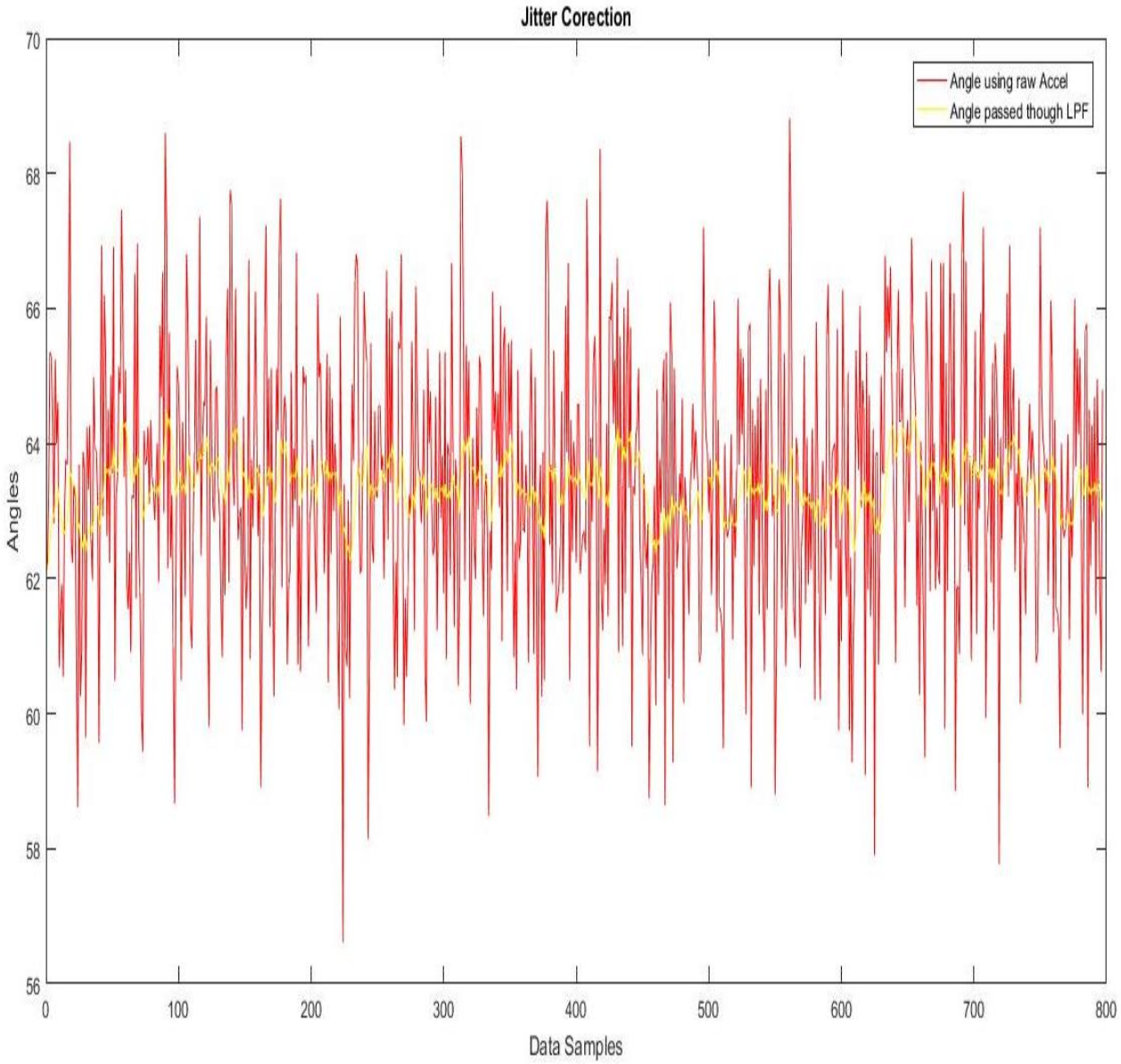


Fig 3.5 Jitter removal from accelerometer data

Fig 3.5 is shows Jitter removal from accelerometer angular angle(roll) calculation using Low Pass Filter method in MATLAB. Accelerometer is very susceptible to very small changes so it values continuously deviates from to original value and unable to give us stable data. Red lines show when Accelerometer is in static position and raw values changes very rapidly even without any kind of disturbance. Yellow line are filtered value after passing raw values though Low pass filter. By changing the “alpha” in LPF formula we can tune is according to the fast or slow movement.

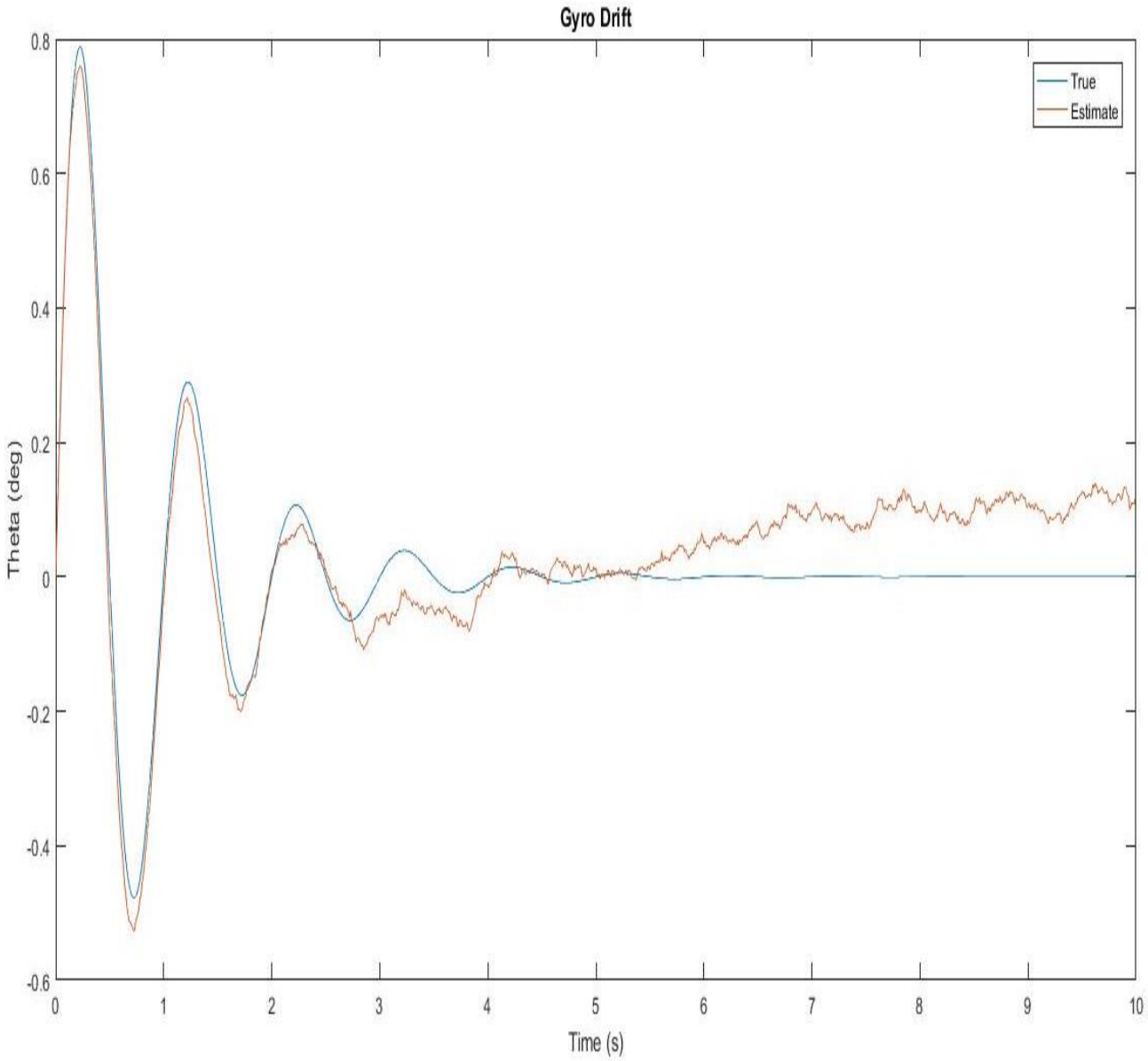


Fig 3.6 Agular Drift in gyro during angular movement

Fig 3.6 is plot between raw value(brown) and filtered value(blue) of angular displacement using magnetometer. Drift occurs due to integration of noise present in the gyroscope data. Since gyroscope give us rate of angular change so by integrating the rate with respect to time we can calculate the angle. As observed from above graph, after 3 second when gyroscope is in static condition, the brown line(without HP filter) start to deviate from blue line(with LP filter) due to continuous summation of noise present in gyroscope.

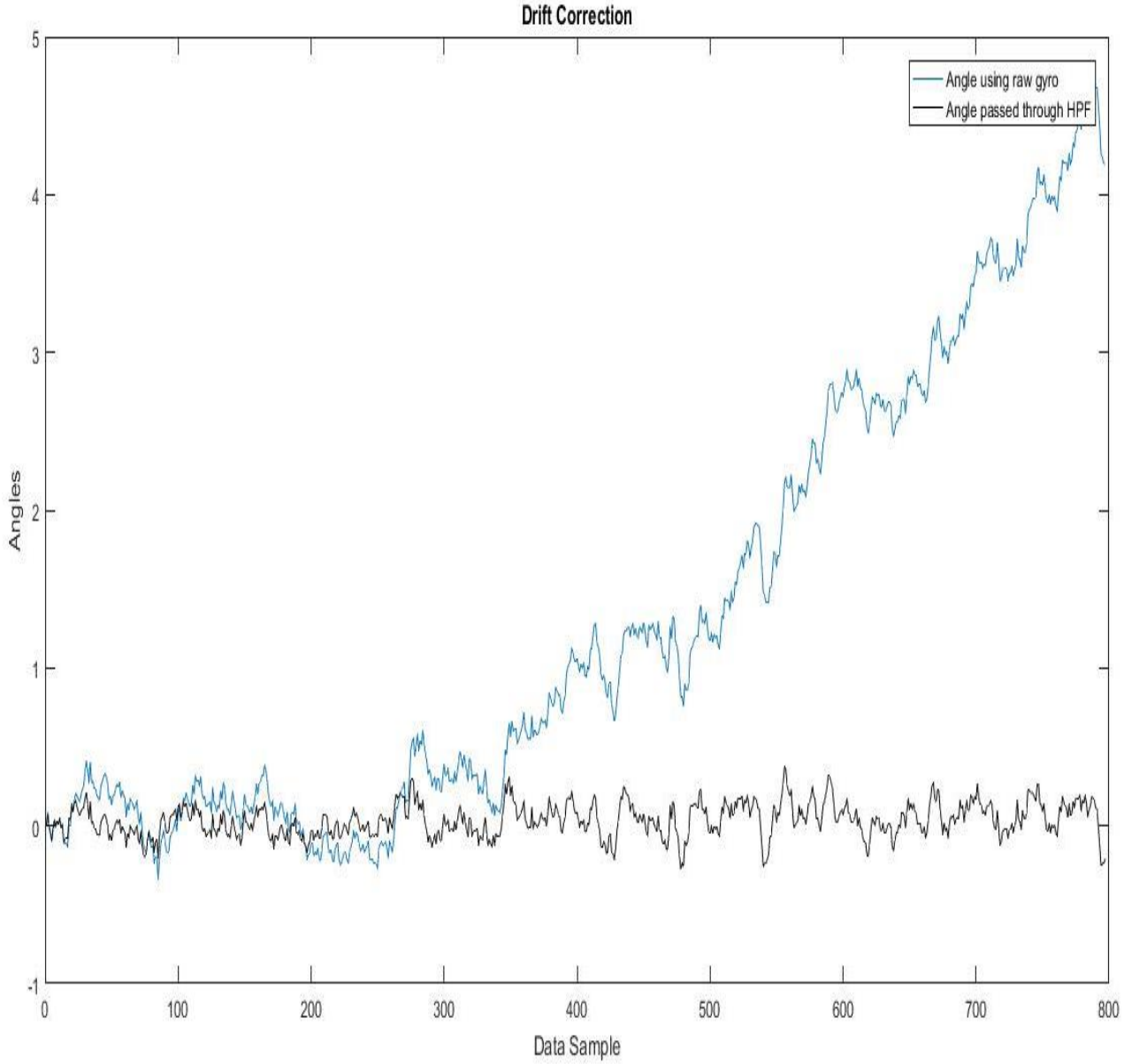


Fig 3.7 Drift correction when gyro is stable.

In fig 3.7 shows the angular displacement obtained by integrating the raw value of gyroscope in blue line start to deviate from the actual angular displacement even when gyroscope is in static condition. This is known as drift caused due to continuous integration of noise present in the gyroscope sensor. This can solve by passing the angular displacement angle by High pass filter. Black line shows the filtered angular displacement when gyroscope is in static condition.

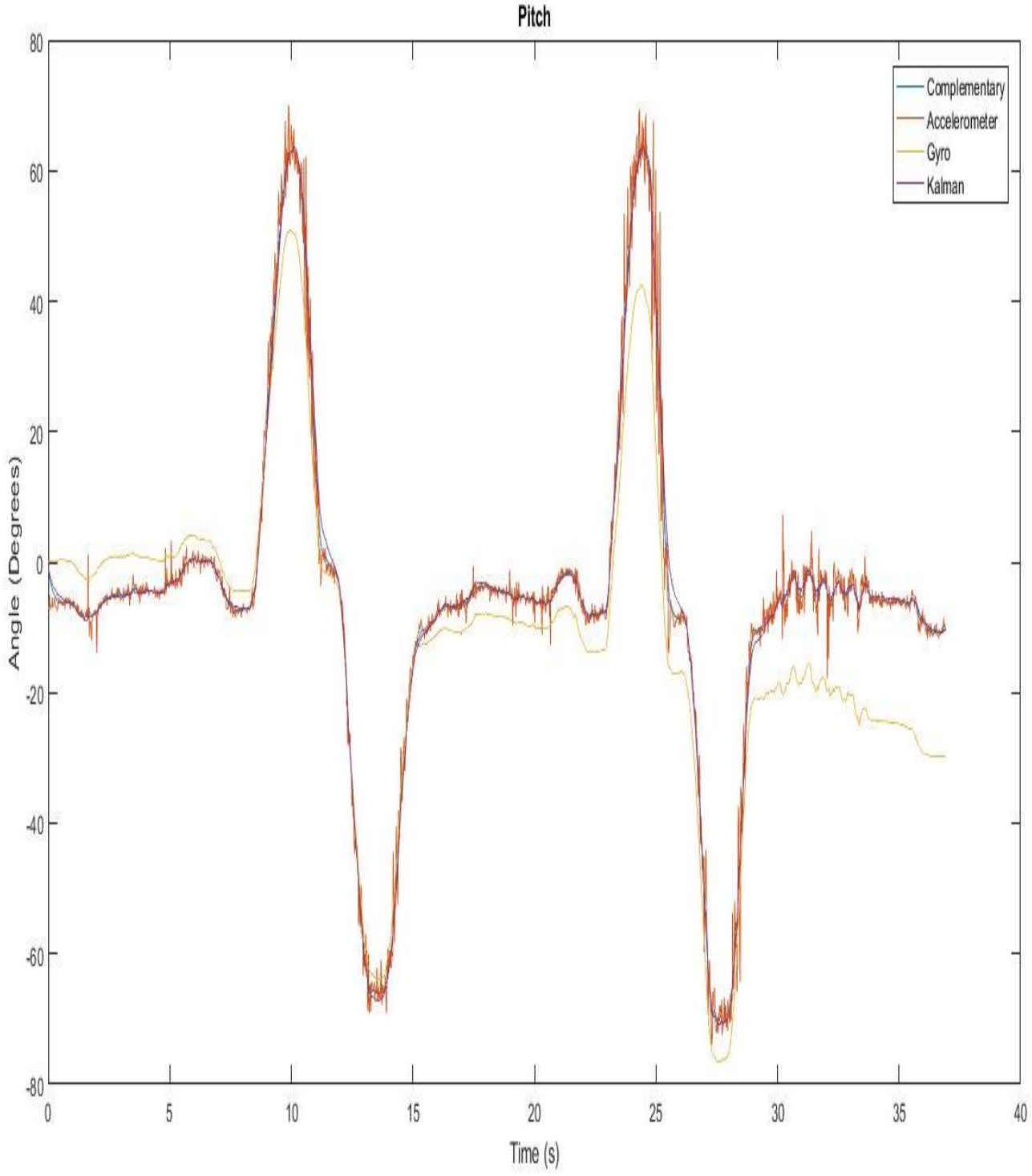


Fig 3.8 Roll angle using Accelerometer, Gyroscope, Kalman and Complimentary

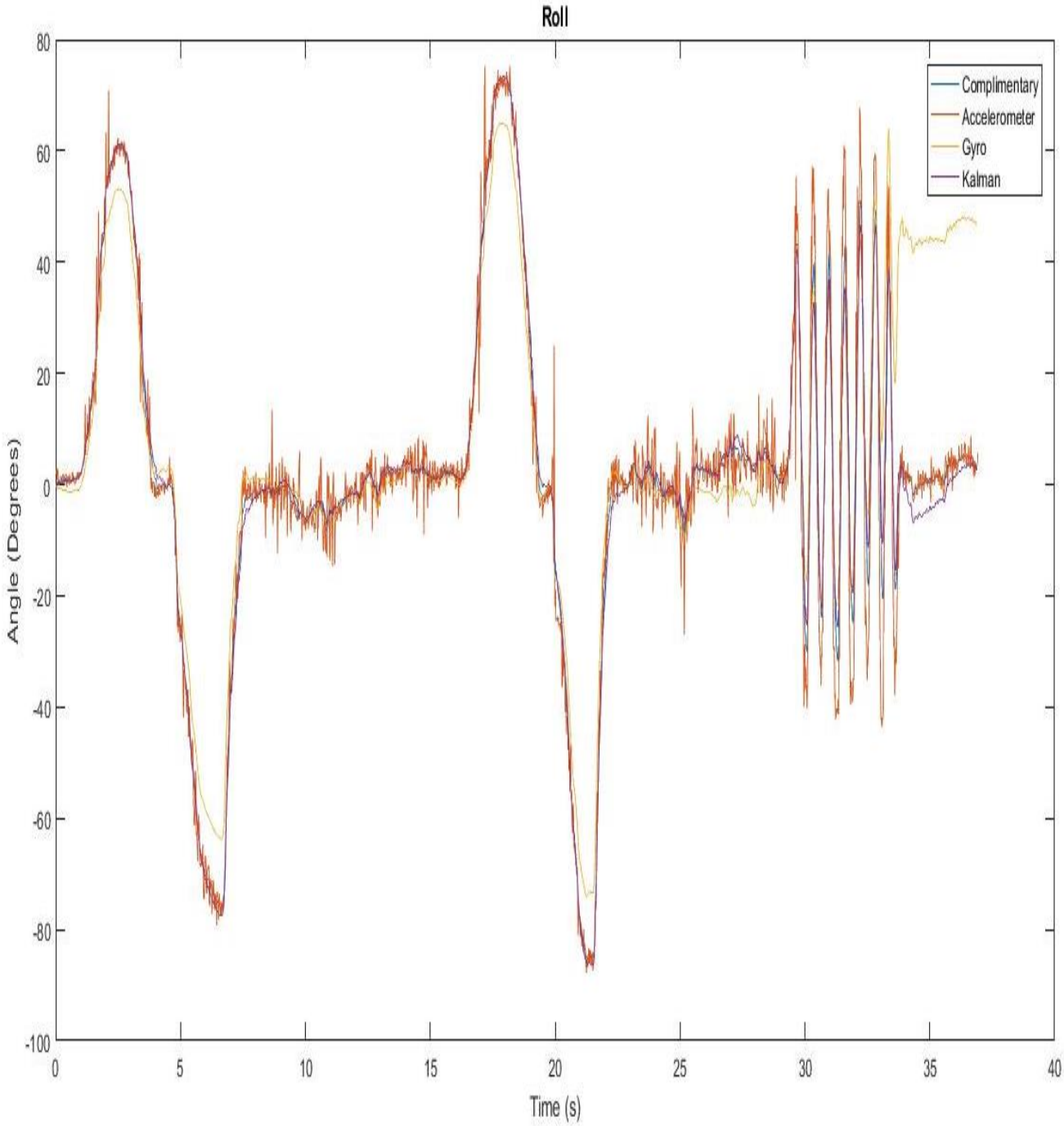


Fig 3.9 Pitch angle using Accelerometer, Gyroscope, Kalman and Complimentary

In fig 3.7 and fig 3.8 angle is plotted by 4 techniques. Blue one angle calculated using complimentary filter, by sensor fusion of accelerometer ang gyroscope, in which roll angle is quite stable and free from the continuous fluctuation. Accelerometer value is passed by LPF and gyroscope value is passed by HPF.

Red one is accelerometer data, angular displacement can be calculated only by accelerometer data but there is continuous fluctuation in value, called jitter. Also, accelerometer is not very good in detecting very rapid movement of sensor.

Yellow one is angular displacement using only gyroscope value. Gyroscope is unable to detect very slow movement of the sensor, just opposite of the accelerometer. Also Yellow line is continuously causes deviation in actual value due to drift.

Violet line is angle drawn using Kalman filter, which fuses the accelerometer and gyroscope. Kalman filter values are very very close to Complimentary filter data. Since Kalman filter fuses the value of accelerometer and gyroscope so during rapid movement of sensor gyroscope values are more reliable and during slow movement of sensor accelerometer data are more reliable.

3.4 NEO GPS 6M

The Global Positioning System (GPS) is a satellite-based navigation system which consist of the 24 satellites moving around the earth. GPS is free of cost and works 24 hours> in bad weather and inside a room it may unable to give the signal. So it will better to use it in open.

GPS satellites move around the Earth twice a day in a precise and send signals towards the earth. GPS devices receive these signals and decode them to use the information and trilateration to calculate a user's exact location. For 2D positioning at least 3 satellites n view are required where for 3D position(latitude, longitude and altitude) at least 4 satellite are required. NEO GPS module give us raw data which can be converted into readable form by NMEA parsing. Tiny GPS++ library has been used for NMEA parsing.

3.5 GPS Inaccuracy

Here is two websites GPS.gov and gpsinindia which clearly states that they have position inaccuracy for a common civilians for the security purpose.

The accuracy of the GPS signal is identical for both the civilian GPS service (SPS) and the military GPS service (PPS). Civilian SPS broadcasts on only one type of frequency 1575.42 MHz, whereas the military PPS uses two 1575.42 MHz and 1227.60 MHz. Which show the military GPS receiver collect better signals from GPS, because there is no loss of signal due to

ionospheric correction. But our mobile shows a stable GPS location because they fuse the information of cellular signals, Wi-Fi router and magnetometer.

There used to be a deliberate inaccuracy added to the civilian signals for security purposes, but that was removed in 2000 by US government. But still 4m-5m inaccuracy is there. See Wikipedia: Error Analysis

3.6 Selection of GPS module

Various GPS module which compatible with the boards like Arduino. Many manufacturers and designers use GPS modules as below: Ublox, Sirf Star III and MTK GPS. The craftsmanship and appearance of Ublox are better. But the cost is higher. Even in the severe environment with plenty of containers and weak GPS signal, "NEO-M8U or NEO-M8L" still has good performance. NEO-M8U Test

Spark fun and Adrafruit GPS module are also in option. Here are the comparison parameter :

1. Update Rate

2. Number of channels

3. Accuracy

4. Antennas

5. Power Consumption

UBlox NEO-M8N GPS Module is easily available in India with the price of 1,400 Rupee. NEO-M8 series is the latest one. UBlox F9 platform is new generation chipset who are claiming for centimeter accuracy but they are in the development phase. M8 and F9 series are fusing Dead reckoning and Real Time Kinematics by combining GNSS such as GPS, GLONASS, Galileo, and BeiDou.

An additional advantage of Ublox M8N product is, It has it own library with NMEA parsing for Arduino. Currently, ublox-neo-6m-gps-module has been used. GPS position accuracy improvement techniques are Differential GNSS, Real Time Kinematics GPS and fusing GNSS, which is discussed in appendix.

3.7 Position update by GPS and accelerometer fusion

Accelerometer have a high frequency that means it has high accuracy but due to double integration of acceleration causes the has Drift in actual value. Note that for fusion with GPS, the acceleration must be in North, East and Down frame of reference. Kalman is best suited filter for fusion of sensor values obtained from different sensors.

Whereas, GPS works on low frequency and have low accuracy as compared to IMUs but GPS provide us absolute position (NED) &absolute velocity in global frame of reference which help for positioning of robot in globe anywhere.

Kalman filter is used to fuse the position and velocity obtained from accelerometer to update GPS latitude and longitude.

Kalman Filter: 1. Predict: accelerometer

2. Update: GPS

Variables Used in Kalman:

A: State transition matrix

B: Control matrix

u: Control vector

Q: Process Variance (acceleration)

R: Measurement Variance (GPS)

Z: Measurement Vector (Position Velocity) for GPS

Note:

1. Variance in accelerometer is calculated by calibrating and taking sample putting it at rest.

2. Variance in GPS can be found by manufacturer specification because if we will put GPS at rest it will result GPS lock.

3. Velocity of GPS is calculated by Doppler Effect

4. Make sure GPS and accelerometer providing position and velocity in North, East and Down frame of reference.

General Kinematics for accelerometer:

$P_f = P_i + (v*\Delta t) + (0.5*a*\Delta t*\Delta t)$
$V_f = V_i + 0 + (a*\Delta t)$

$$\begin{bmatrix} P_f \\ V_f \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_i \\ V_i \end{bmatrix} + \begin{bmatrix} \Delta t * \Delta t * 0.5 \\ \Delta t \end{bmatrix} *\text{Acceleration}$$

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} \Delta t * \Delta t * 0.5 \\ \Delta t \end{bmatrix}$$

u=[a]

$$z = \begin{bmatrix} Position_{GPS} \\ Velocity_{GPS} \end{bmatrix}$$

For any direction North/East/Down

Step 1: Prediction:

$$\begin{bmatrix} P_{k(acc)} \\ V_{k(acc)} \end{bmatrix} = [A] \begin{bmatrix} P_{k-1(acc)} \\ V_{k-1(acc)} \end{bmatrix} + [B][u]$$

Step2: Update:

$$\begin{bmatrix} P_{k(updated)} \\ V_{k(updated)} \end{bmatrix} = \begin{bmatrix} P_{k-1(updated)} \\ V_{k-1(updated)} \end{bmatrix} + \begin{bmatrix} Kalman \\ Gain \\ Matrix \end{bmatrix} \left(\begin{bmatrix} P_{k-1(GPS)} \\ V_{k-1(GPS)} \end{bmatrix} - [C] \begin{bmatrix} P_{k-1(acc)} \\ V_{k(-1acc)} \end{bmatrix} \right)$$

Chapter 4

Navigation and path finding algorithm

For autonomous navigation in unknown area, especially on uneven terrain GPS is best suited sensor. Because it gives global position with unique coordinate. It is free of cost, and easy to use as compared to other techniques like SLAM, image processing or 3d mapping of environment.

4.1 Components Used

Following sensors are used for the autonomous rover of robot. GPS will decide the movement toward the goal while magnetometer will act as feedback device for the heading correction.

4.1.1 GPS module:

To locate the position of robot in unknown environment, GPS give us latitude and longitude of robot. Targeted distance and Target Heading can be calculated by knowing the Goal's and current's latitude and longitude.

4.1.2 IMU:

9 DOF MEMS based IMU contains accelerometer, gyroscope and magnetometer which can give us current heading as well as roll pitch and yaw of the robot which helps us to know the current orientation of robot.

4.1.3 Ultrasonic Senor:

For detection and avoidance of obstacles faced during the autonomous rover of the robot. IT is used as distance measuring device which can tell the distance of obstacle from the robot. Ultrasonic sensor are also used in wall following.

4.2 Calculation of Orientation

a. Current orientation of robot

It is very important to know the in what direction robot is heading. To reach to the goal the robot should always aimed toward final coordinate. Electronic compass has been used to know the heading of robot. HMC5883L has been used for this purpose.

b. Target Heading to robot and Target Distance

Target is predefined orientation in which robot should move. GPS module will give us current and final coordinates and by using simple formula we can find the distance and target heading of the robot.

4.3 Path finding algorithm without Obstacle

Path finding is in continuation to the overall maneuvering and motion of the bot. It is a computational job performed by the on-board computers based on a set of specific algorithms. The overall path of robot will be saved in an array which will contain the latitude and longitude through which robot will have to move. The final latitude and longitude will be already saved in the microcontroller. The robot has on boards GPS which will continuously update the current latitude and longitude. Between any two coordinates the robot will continuously calculate the Target distance and it will move till the target distance will be zero. The target heading will be calculated using GPS module using current and final latitude and longitude. And current heading will be calculated using Magnetometer. The robot will continuously compare the current heading and target heading. If by the way, the difference between target an current heading will be different then robot will have to take a turn. For positive difference take a clockwise turn while for a negative difference it takes a anticlockwise turn. PID control system has been applied for this turning. Heading difference will be the error of system, and more will be the difference more will be the error and robot turning angle will increase.

4.3.1 Pseudo code:

Step1: Store the order of path with longitude and latitude in an array with GPS sensor.

Step2: Target heading is measured using GPS sensor. And current heading is measured using Electronic Compass at every instant of time.

Step3: The difference in angle, i.e. heading error will decide its turn, straight/left/right.

Step 4: Check Targeted Distance. And if Target distance is zero, stop the robot.

Step 5: Rotate the servo motor having an ultrasonic sensor on it, from 0-180 degree and calculate desired and distance of targeted pole.

Step 6: move toward the target pole

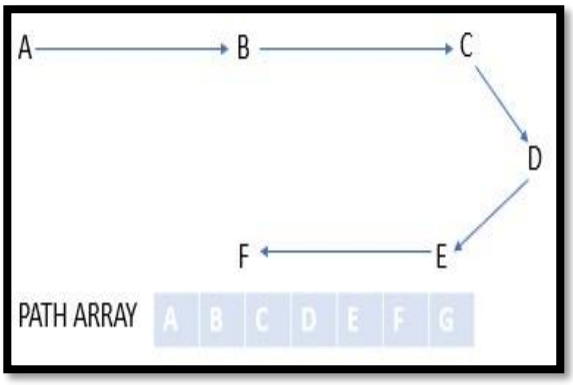


Fig 4.1 Order of Path array

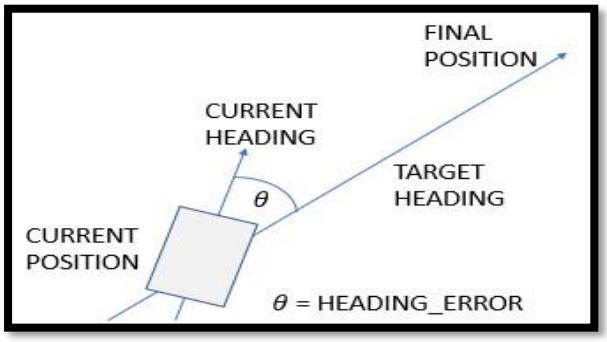


Fig 4.2 Heading Difference

```
void loop
        Current_Heading=Read_instant_Compass_Orientation;
        Target_Heading=Read_instant_GPS_Orientation;
        Calculate_Desired_Turn();
        Calculated_Desired_Distance();
```

```
            If( Desired_Distance =0)
                    STOP;
                     Roate_servo_motor();
                     Read_Distance_of_TargetedPole_by_Ultrasonic();
                     Read_Angle_of_targetedPole();
                     Move_toward_Pole();
End
Calculate_Desired_Turn()
      Heading_error= Target_Heading- Current_Heading;
      If Heading_error=0
            Turn_Direction=Straight
      If Heading_error<0
            Turn_Direction=Left
      If Heading_error>0
            Turn_Direction=Right
End
```

4.4 GPS position inaccuracy

Due to the GPS position inaccuracy, robot can reach at target latitude and longitude having an error of 3-4m. So to travel large distance GPS is used and most of distance will be covered using GPS only.

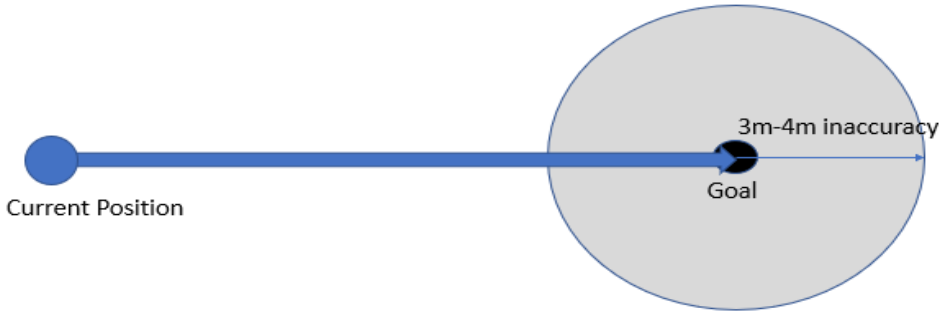


Fig 4.3 GPS position inaccuracy of 3-4m of radius at final goal

First of all, GPS can give its value only in open atmosphere. Never used it inside room or building. And GPS takes few minutes to give it value and take a small time for the saturation of value. Number of satellites also play very crucial role. At least 4 satellite are required to get the value of latitude and longitude. The robot can reach in perimeter of 3m radius of its goal.

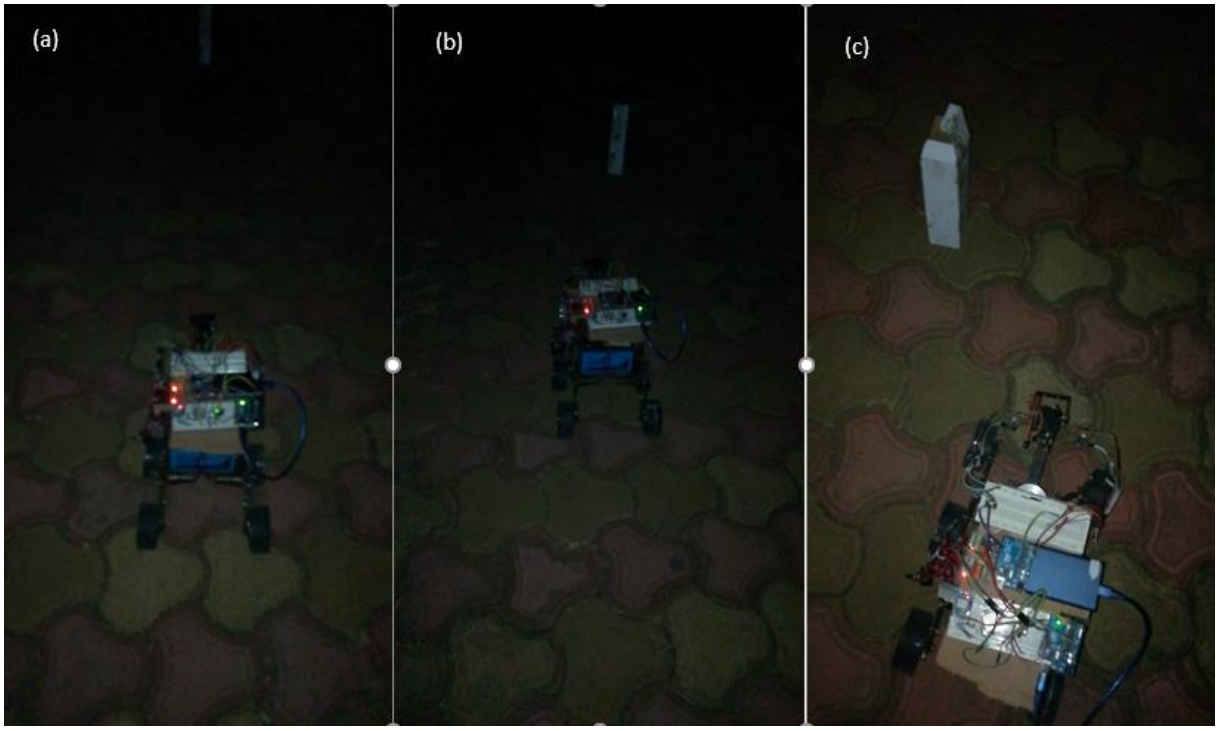


Fig 4.4 Robot does not reach target by using only GPS

In Fig 4.4 (a) and (b) robot is continuously heading toward its goal but in (C) robot stop and does not reach to targeted pole. So a 1.5m of distance is not covered by robot.

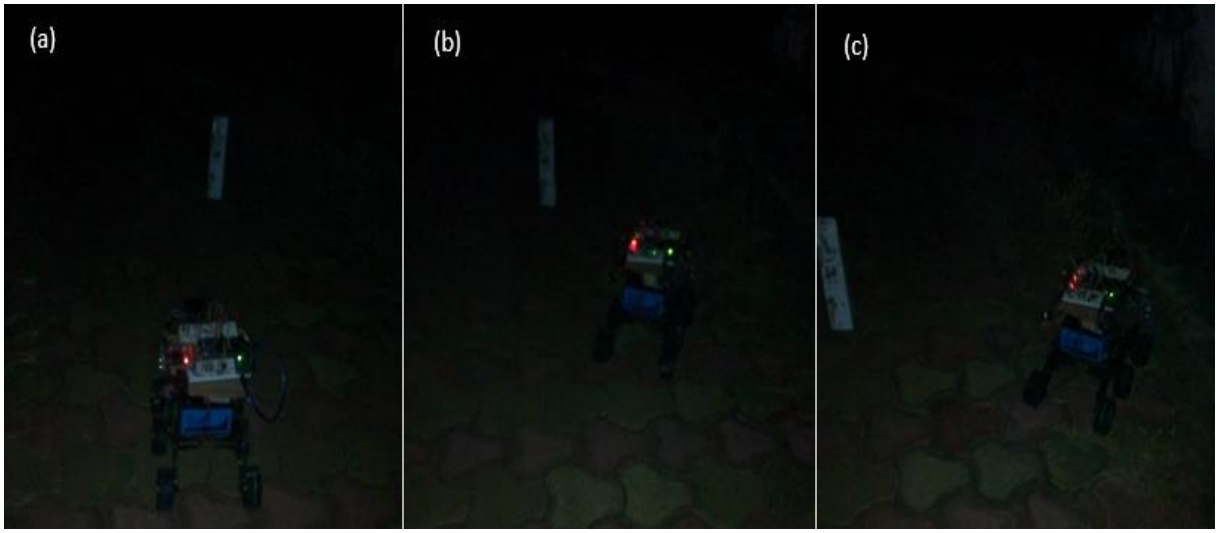


Fig 4.5 Sudden Drift in distance by accelerometer

Fig 4.5 (a) shows robot reach near the targeted pole but in fig (b) it suddenly starts to deviate and in (c) it completed changed it path away from the targeted pole. This happens due to accelerometer drift when acceleration obtained by accelerometer is twice integrated to get the distance to be covered using accelerometer only. During the deviation of path, distance is calculated only with help of accelerometer, GPS data are not working near the targeted pole.

4.5 Solution to reach the Target

For heading difference set a tolerance of 10 degree. -5 to 5 degree in which robot will move continuously forward. For target heading set a tolerance of 2 m for its goal position. Apply angle declination in magnetometer while calculating target heading.

So the robot will continuously move toward the target latitude and longitude while keep updating its heading difference and target distance. But due to GPS in accuracy and Tolerance of 2m has been provided it will stop in the periphery of goal.

At the goal we will place and small 10*10*20 cm cubical object. The front side of robot will contain a servo motor and having a Ultrasonic sensor on it. When robot will reach in the periphery of 2m-3m the servo motor rotate and take reading of the distance using ultrasonic sensor. As the object is detected it will save the angle and distance from robot to which pole is located. According to angle it will take a turn in that angle using magnetometer.

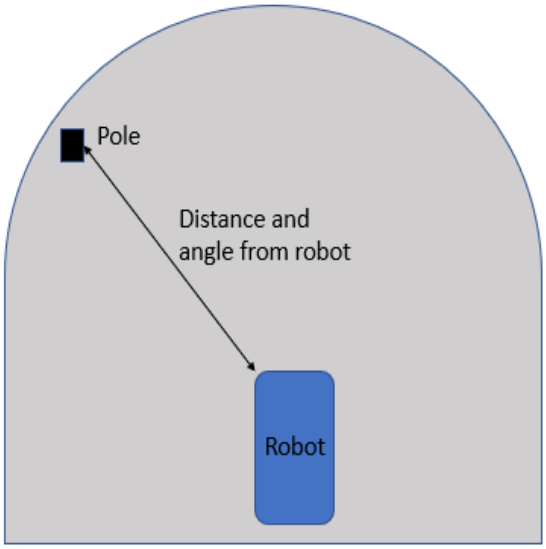


Fig 4.6 Final Position Estimation of Targeted Pole

The two main constraint of robot is:

1. Unable to take sharp turn. The min radius formed is 125 m using differential drive.
2. Unable to rotate on its own axis on rough surface.

So every time for a sharp turn robot has to come backward to maintain a sufficient distance for turning. Then robot will start turning. It will move till the front ultrasonic sensor will read distance 0cm.To take a sharp turn, the robot first move backward so that enough distance is maintained. Then it start differential turning. Such turning can be seen while parking a car.

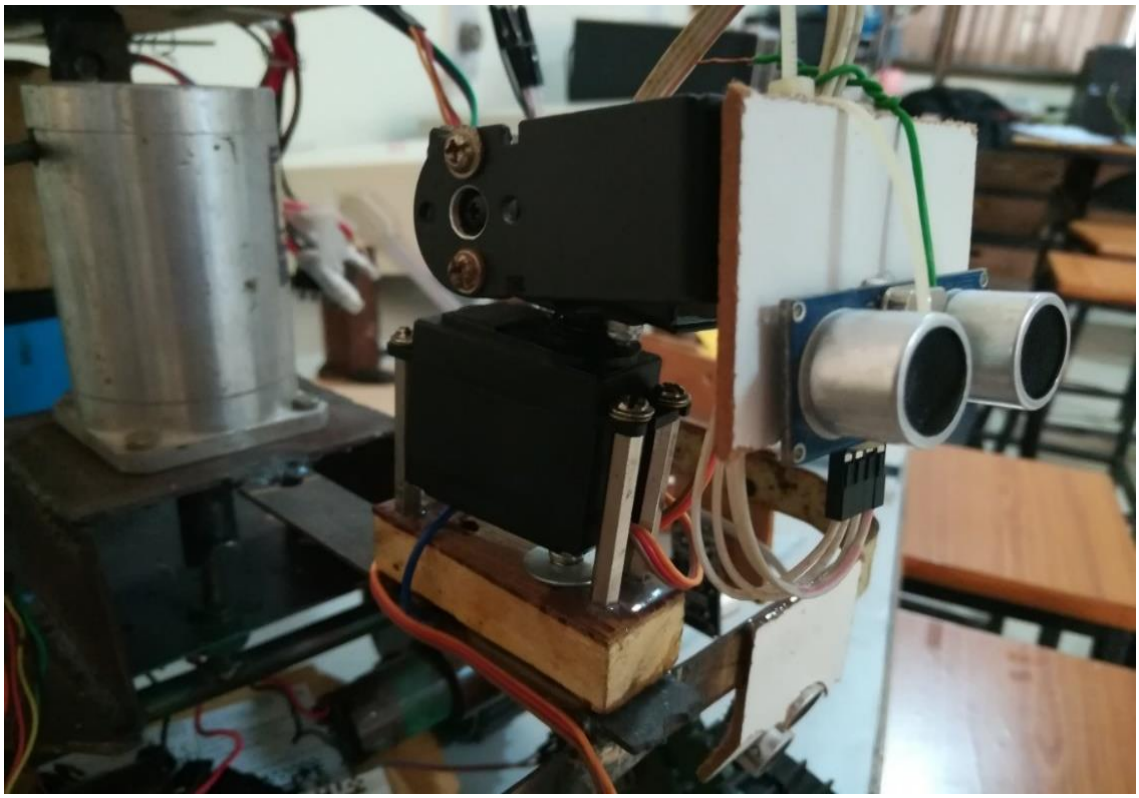


Fig 4.7 Ultrasonic Sensor mounted on Servo motor

4.6 Discussion

Overall working of robot can be understood by this chart. In which first targeted latitude and longitude is saved in robot and robot will cover most of the distance by GPS and magnetometer. After reaching in diameter of 2-3m, robot will stop. And ultrasonic sensor mounted on robot will calculate desired angle and desired distance and start moving toward pole with help of ultrasonic sensor.

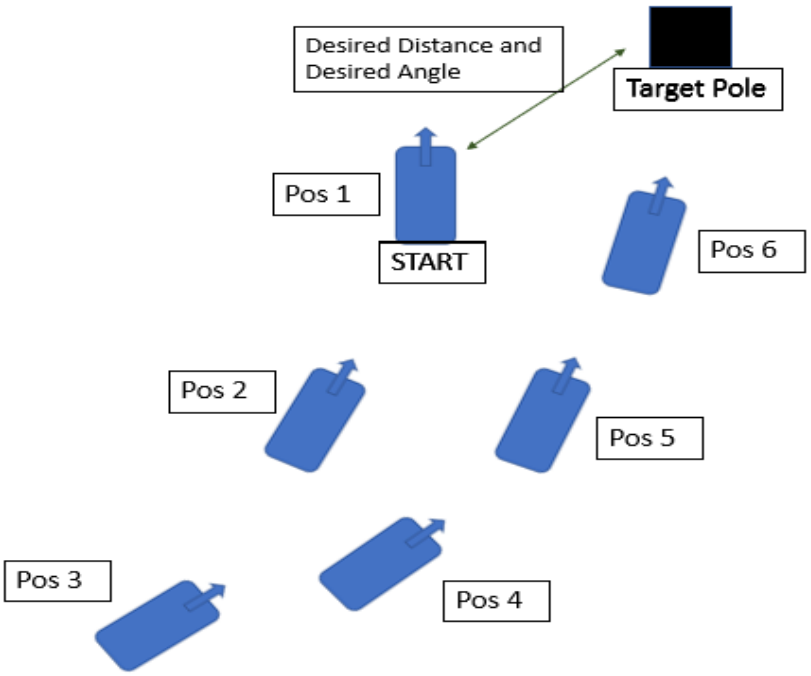


Fig 4.8 Method to take sharp turn is first robot come reverse back and take maximum turn.

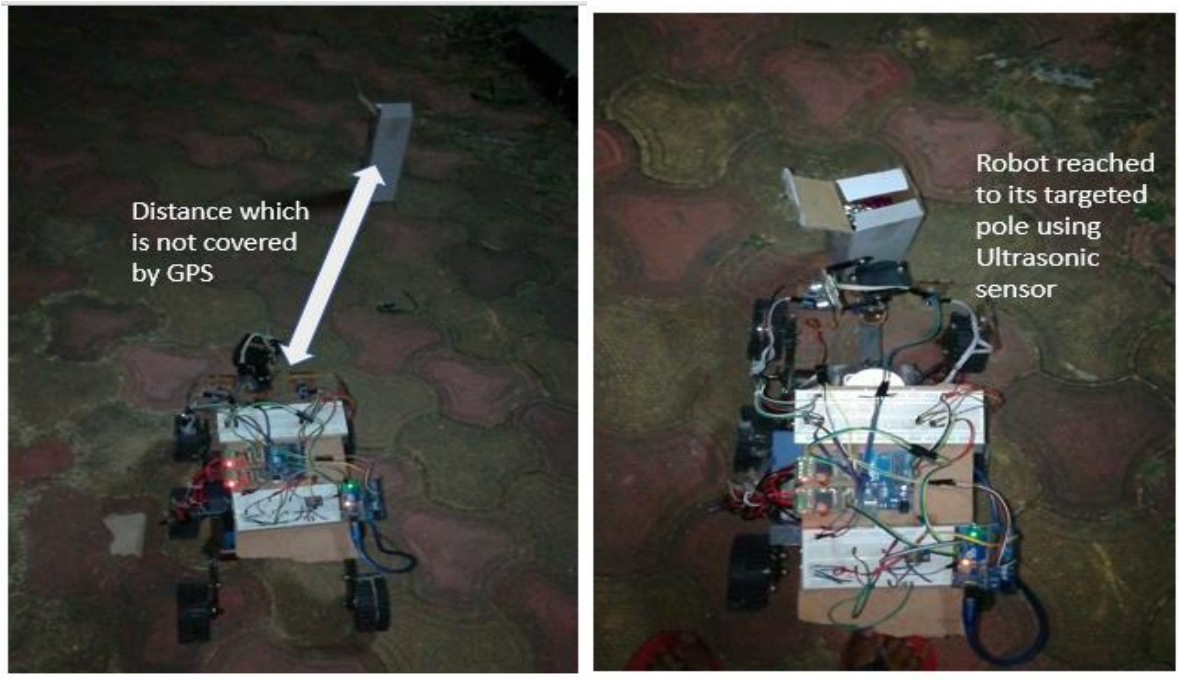
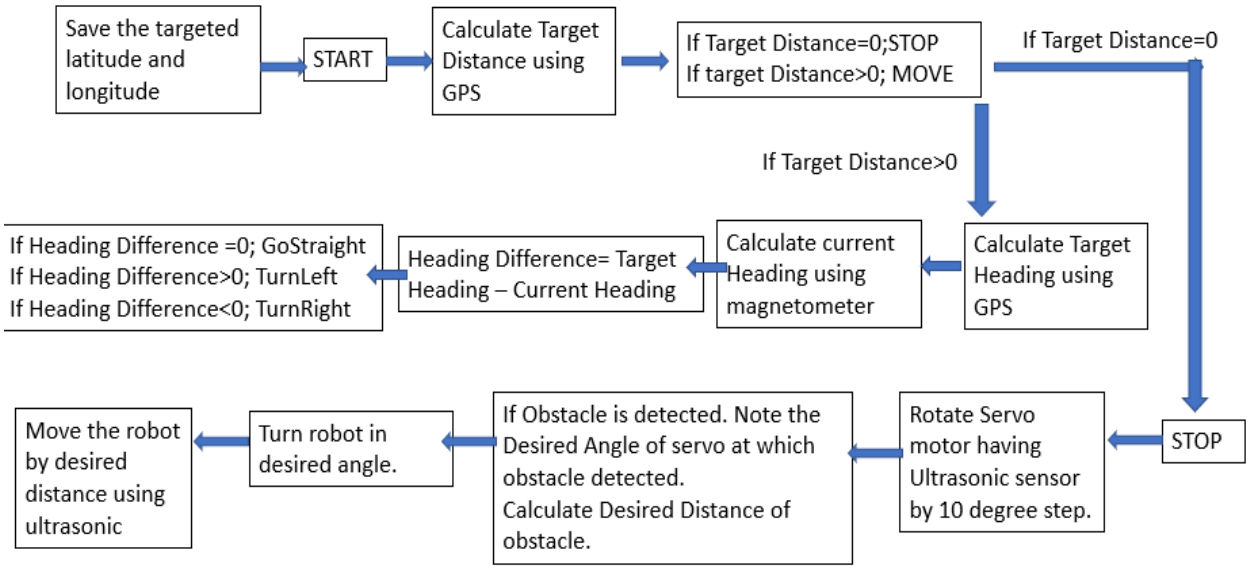


Fig 4.9 a) The robot reached in the periphery of 2m of targeted goal, it is calculated desired angle and desired distance by rotating the servo motor having ultrasonic sensor.

Fig 4.9 b) Robot covered the targeted latitude and longitude.

A final working flowchart has been shown below in which most of the distance is covered by GPS guided by magnetometer but near the goal GPs value will not be obtained and then only with help of ultrasonic sensor robot covered the remaining distance.



4.7 Conclusion

By proposed method robot was able to move toward its goal. In fig 4.9 first robot will reach in the periphery of 2m-3m radius and it will stop. With the help of ultrasonic sensor mounted on servo motor, it will rotate 0 to 180 degree and detect the position of pole in that periphery. Once the obstacle is detected, robot will calculate the desired angle and distance from the pole and move toward it. To take sharp turn, first come backward and then start differential turning.

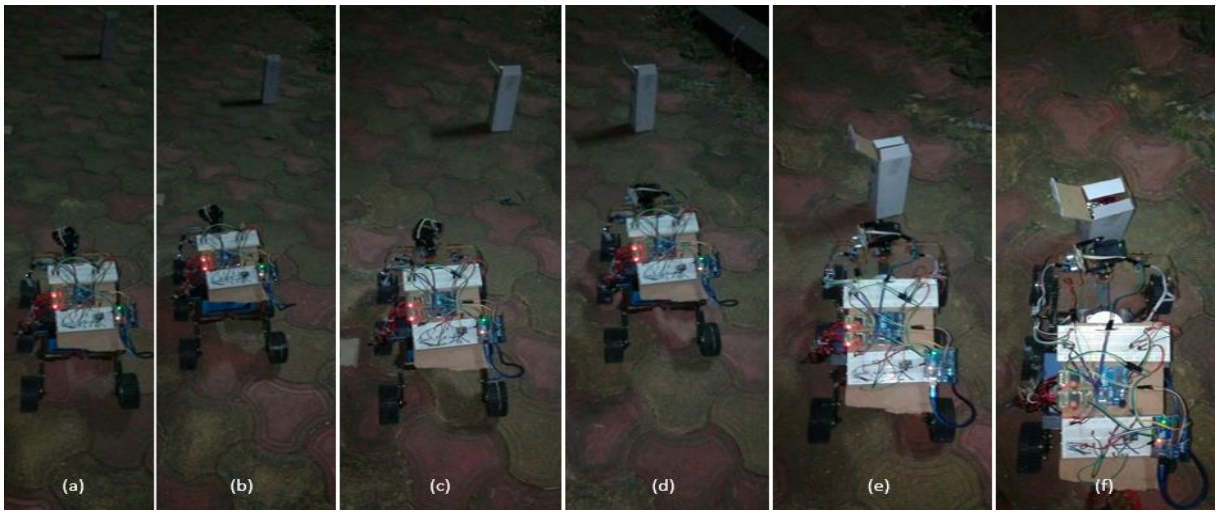


Fig 4.10 Robot heading toward target Lat and Long then moving toward desired pole

[2018-11-11 14:21:07.!	GPS Lat	GPSLong	TargetDistance	TargetHeading	CurrentHeading	HeadingDifference
[2018-11-11 14:21:07.!	23.178642	80.025955	7	141	144	-3
[2018-11-11 14:21:07.!	23.178642	80.025955	7	141	144	-3
[2018-11-11 14:21:07.!	23.178642	80.025955	7	141	144	-3
[2018-11-11 14:21:07.!	23.178642	80.025955	7	141	144	-3
[2018-11-11 14:21:07.!	23.178642	80.025955	7	141	144	-3
[2018-11-11 14:21:07.!	23.178642	80.025955	7	141	144	-3
[2018-11-11 14:21:07.!	23.178642	80.025955	7	141	144	-3
[2018-11-11 14:21:07.!	23.178642	80.025955	7	141	144	-3
[2018-11-11 14:21:08.7	23.178642	80.025962	7	146	144	2
[2018-11-11 14:21:15.:	23.178642	80.025962	7	146	144	2
[2018-11-11 14:21:15.:	23.178642	80.025962	7	146	144	2
[2018-11-11 14:21:17.7	23.178642	80.025962	7	146	144	2
[2018-11-11 14:21:20.7	23.178642	80.025955	7	141	144	-3
[2018-11-11 14:21:23.7	23.178642	80.025955	7	141	144	-3
[2018-11-11 14:21:26.7	23.17864	80.025955	7	141	145	-4
[2018-11-11 14:21:29.7	23.178636	80.025962	6	144	146	-2
[2018-11-11 14:21:32.7	23.178632	80.02597	6	148	146	2
[2018-11-11 14:21:35.7	23.178628	80.02597	5	147	147	0
[2018-11-11 14:21:38.7	23.178623	80.02597	5	144	148	-4
[2018-11-11 14:21:41.7	23.178617	80.025978	4	147	148	-1
[2018-11-11 14:21:44.7	23.178613	80.025978	3	140	151	-11
[2018-11-11 14:21:46.7	23.178611	80.025978	3	140	150	-10
[2018-11-11 14:21:49.7	23.178607	80.025985	2	145	148	-3
[2018-11-11 14:21:52.7	23.178604	80.025985	2	143	150	-7
[2018-11-11 14:21:55.7	23.178598	80.025985	stop			

Table 4.1 Test Drive 1 Lat and Long covered during Moving toward goal

	GPS LAT	GPSLong	TargetDistance	TargetHeading	CurrentHeading	HeadingDifference
[2018-11-11 14:29:12.444	23.178636	80.02597	6	150	144	6
[2018-11-11 14:29:12.444	23.178634	80.02597	6	149	143	6
[2018-11-11 14:29:12.444	23.178632	80.02597	6	148	144	4
[2018-11-11 14:29:12.444	23.178632	80.025962	6	142	144	-2
[2018-11-11 14:29:12.772	23.178632	80.025962	6	142	144	-2
[2018-11-11 14:29:15.791	23.178634	80.025962	6	143	143	0
[2018-11-11 14:29:18.779	23.178634	80.025962	6	143	145	-2
[2018-11-11 14:29:21.780	23.17863	80.025962	6	141	143	-2
[2018-11-11 14:29:24.793	23.178628	80.02597	5	147	145	2
[2018-11-11 14:29:27.776	23.178627	80.02597	5	146	145	1
[2018-11-11 14:29:30.786	23.178623	80.025978	4	150	147	3
[2018-11-11 14:29:33.786	23.178617	80.025978	4	147	149	-2
[2018-11-11 14:29:36.792	23.178611	80.025978	3	140	148	-8
[2018-11-11 14:29:39.780	23.178607	80.025978	3	134	149	-15
[2018-11-11 14:29:41.790	23.178607	80.025978	3	138	148	-10
[2018-11-11 14:29:43.792	23.178604	80.025978	3	134	146	-12
[2018-11-11 14:29:45.779	23.178602	80.025978	2	137	145	-8
[2018-11-11 14:29:47.774	23.178598	80.025978	2	140	145	-5
[2018-11-11 14:29:49.796	23.178598	80.025978	2	137	143	-6
[2018-11-11 14:29:51.780	23.178596	80.025985	STOP			

Table 4.2 Test Drive 2 Lat and Long covered during Moving toward goal

[2018-11-11 14:41::	GPSLat	GPSLong	TargetDistance	TargetHeading	CurrentHeading	HeadingDifference
[2018-11-11 14:41::	23.178621	80.02597	5	143	142	1
[2018-11-11 14:41::	23.178619	80.025962	5	135	141	-6
[2018-11-11 14:41::	23.178617	80.025962	5	134	141	-7
[2018-11-11 14:41::	23.178617	80.025962	5	134	142	-8
[2018-11-11 14:41::	23.178615	80.025962	5	129	142	-13
[2018-11-11 14:41::	23.178613	80.025955	5	122	142	-20
[2018-11-11 14:41::	23.178615	80.025962	5	129	140	-11
[2018-11-11 14:41::	23.178613	80.025962	4	127	139	-12
[2018-11-11 14:41::	23.178613	80.02597	4	134	138	-4
[2018-11-11 14:41::	23.178617	80.02597	4	140	138	2
[2018-11-11 14:41::	23.178613	80.02597	4	134	139	-5
[2018-11-11 14:41:4	23.178613	80.025962	4	127	139	-12
[2018-11-11 14:41:4	23.178617	80.02597	4	140	138	2
[2018-11-11 14:41:4	23.178613	80.025978	3	140	138	2
[2018-11-11 14:41:5	23.178613	80.025985	3	151	139	12
[2018-11-11 14:41:5	23.178611	80.025993	3	165	142	23
[2018-11-11 14:41:5	23.178606	80.026	2	180	146	34
[2018-11-11 14:41:5	23.178594	80.025993				

Table 4.3 Test Drive 3 Lat and Long covered during Moving toward goal

Chapter 5

Obstacle Avoidance

Obstacle avoidance is one of the most important aspects of mobile robotics. Without it robot movement would be very restrictive and fragile. The aim of robot is for obstacle avoidance in its locally known area by onboard ultrasonic sensors. Various methods for obstacle avoidance has been studied and few of them are: Bug method, Potential Field Method, Vector Field Histogram and Bubble band technique.

5.1 Tangent Bug Algorithm

Among these the tangent bug algorithm would perform better than the other, simpler, bug algorithms. It seems to generally be a more intelligent solution to the problem and for many cases takes a shorter path than the other bug implementations. So the approach is for Tangent Bug algorithm. However, It has been not implanted successfully.

Tangent Bug work on two approach:

1. Go to goal behavior
2. Boundary following method

Since Target GPS latitude and longitude was already known to robot and it was successfully able to move toward the goal, so the first step was completed successfully. A motion-to-goal behavior as long as way is clear or there is a visible obstacle boundary point that decreases heuristic distance. A boundary following behavior invoked when heuristic distance increases.

So to complete the task it is very important to make a boundary following robot or wall following robot. 4 ultrasonic Sensors were placed on robot, 2 in front of robot and 1 in left side

and 1 in right side. The aim of providing two ultrasonic sensor in front of robot is so that it can cover whole width of robot either one ultrasonic sonic is insufficient.

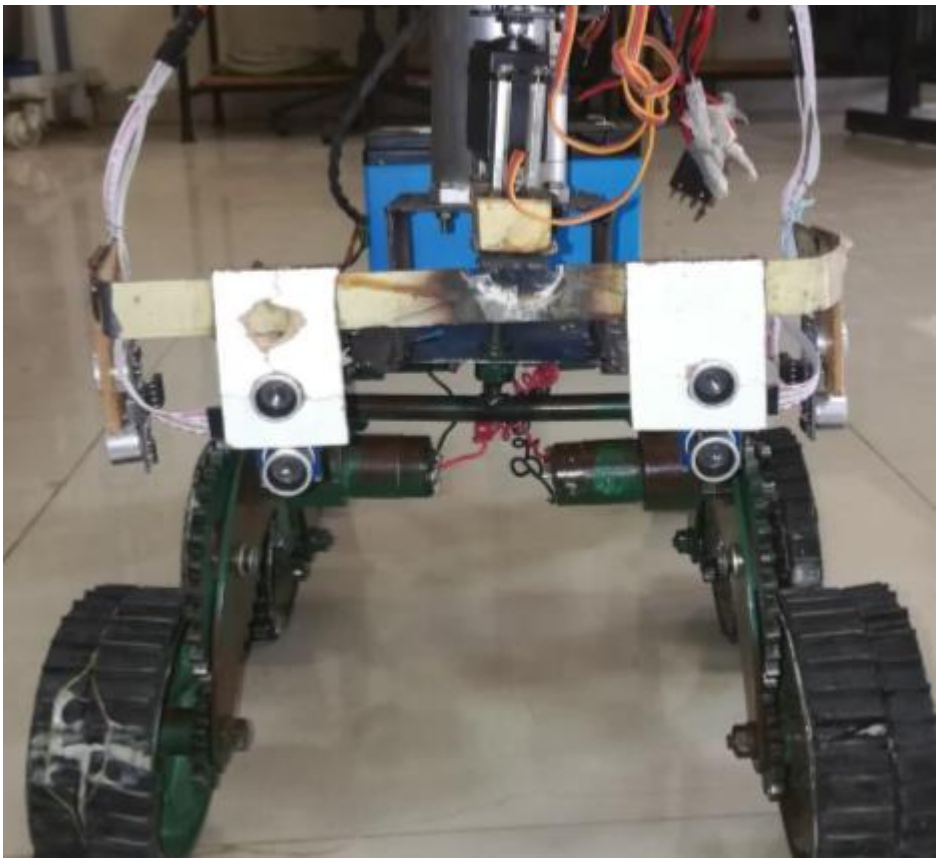


Fig 5.1 Position of 4 Ultrasonic sensors, 2 in front and 1 in both sidewise.

A simple Left wall following algorithm:

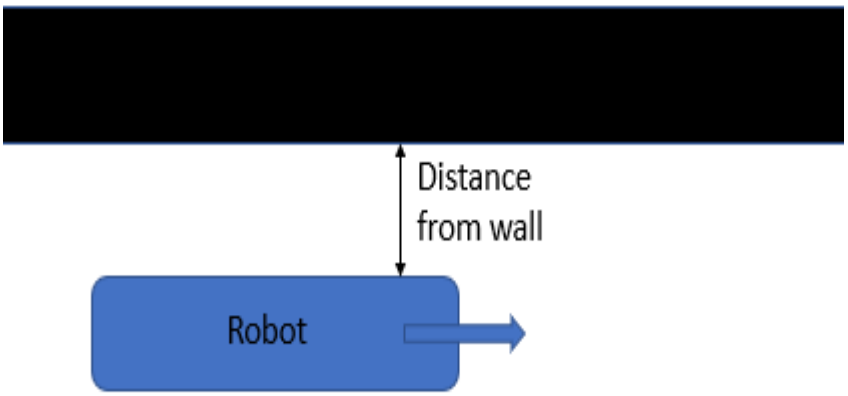


Fig 5.2 Wall following robot

```
SetpointDistance = 20 cm

WallDistance= ReadLeftUltrasonicSensor

If(SetpointDistance=WallDistance)

        moveStraight

If(SetpointDistance>WallDistance)

        turnRight

If(SetpointDistance>WallDistance)

        turnLeft
```

5.2 Problem during Wall following

Following are the disadvantages of Ultrasonic sensor:
It is very sensitive to variation in the temperature. It has more difficulties in reading reflections from a)soft, b)curved, c)thin and d)small objects. Ultrasonic sensor doesn't give accurate result if the obstacle is not exactly in front of it.

Turning limitation of robot: Since min radius that robot can form is 125 cm. It means the robot is unable to take sharp using differential turn if obstacle came in front of it. Another option is rotate the robot on its own axis. And it works quite well on smooth tiles. But on rough surface like concrete, the semicircular concrete shape rocker get rotated by 180 and linkage get stuck.

Due to these two major limitation Tangent bug algorithm was not implement successfully. But wall following robot (left and right) works pretty well on tile where robot can rotate on its vertical axis. Following type of single obstacle wall following method has been developed.

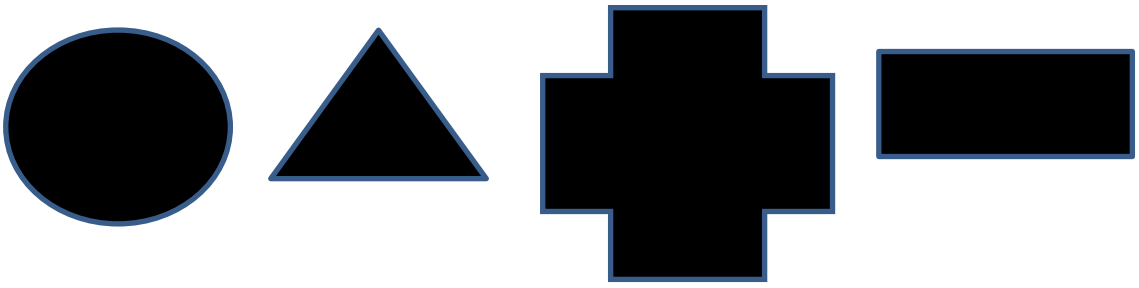


Fig 5.3 Different shape of obstacle for wall following

5.3 Proposed Solution for Wall Following

The robot should have an sharp azimuth distance sensor with infinitesimal angular resolution, so that it can read the discontinuities of obstacles. To increase the steering angle (more than 15 degree) by high torque servo motor mounted on the robot, which can result in sharp turning.

5.4 Obstacle avoidance for Single Obstacle

A simple obstacle avoidance algorithm has been following. So robot will move toward its target latitude and longitude in unknown area. While moving forward if any obstacle(single) is encountered in front of it, then the ultrasonic sensor mounted on servo motor will rotate from 0 to 180 degree and find the angle at which there is no obstacle. And then robot will approach that angle where there is no obstacle. Once robot will crosses the obstacle it, then again follow the Go To Gol mode using GPS and magnetometer.

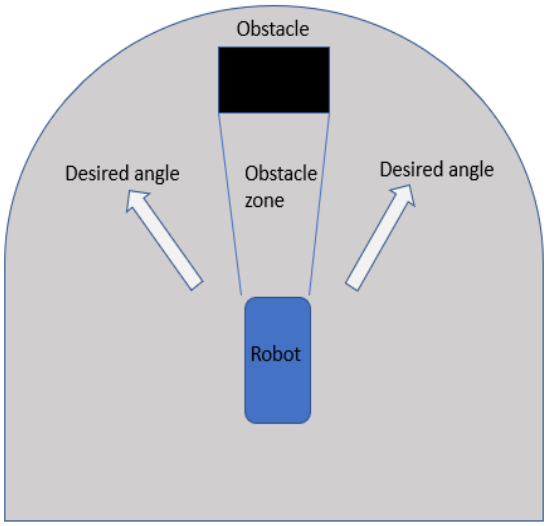


Fig 5.4 Desired angle calculation for obstacle avoidance

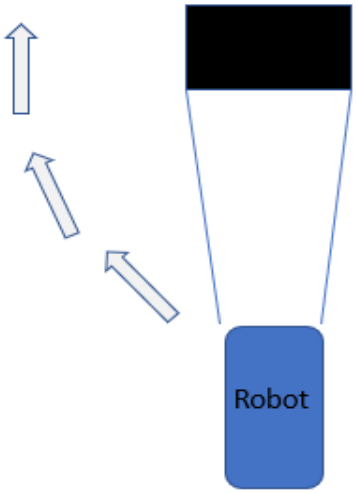


Fig 5.5 Path followed for obstacle avoidance

5.5 Result

The aim of obstacle avoidance was to make a robot based on Tangent bug algorithm but it was not implanted successfully due to limitation of Ultrasonic sensor and problem in sharp turning. However, wall following robot works successfully on smooth surface where rotation of robot on its axis was possible. In addition to that robot was able to move towards its goal on a rough surface and if any obstacle was encountered in front of its robot will take a turn and cross the obstacle passing beside to it.

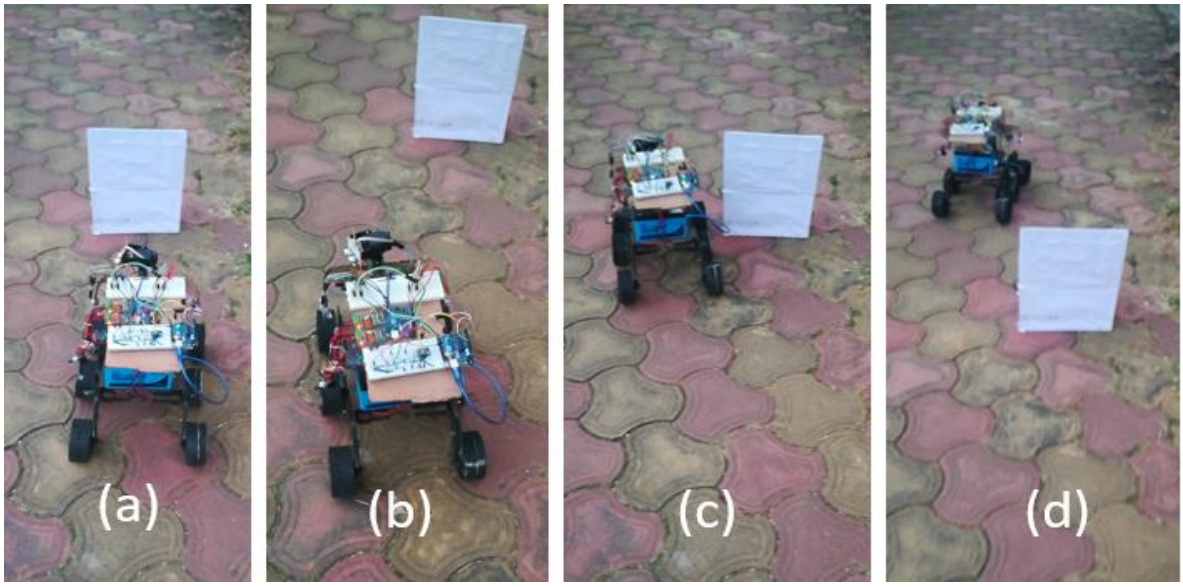


Fig 5.6 Obstacle detection and Obstacle avoidance by the robot. Robot passes by the left side of obstacle.

In fig 5.6 robot finds the obstacle in front of it then it finds the way with no obstacle by rotating ultrasonic sensor mounted on servo motor. To avoid the obstacle, first the robot will come backward and then take differential turn. As surface is rough so rotation is not possible. After obstacle avoidance robot will again move toward it goal.

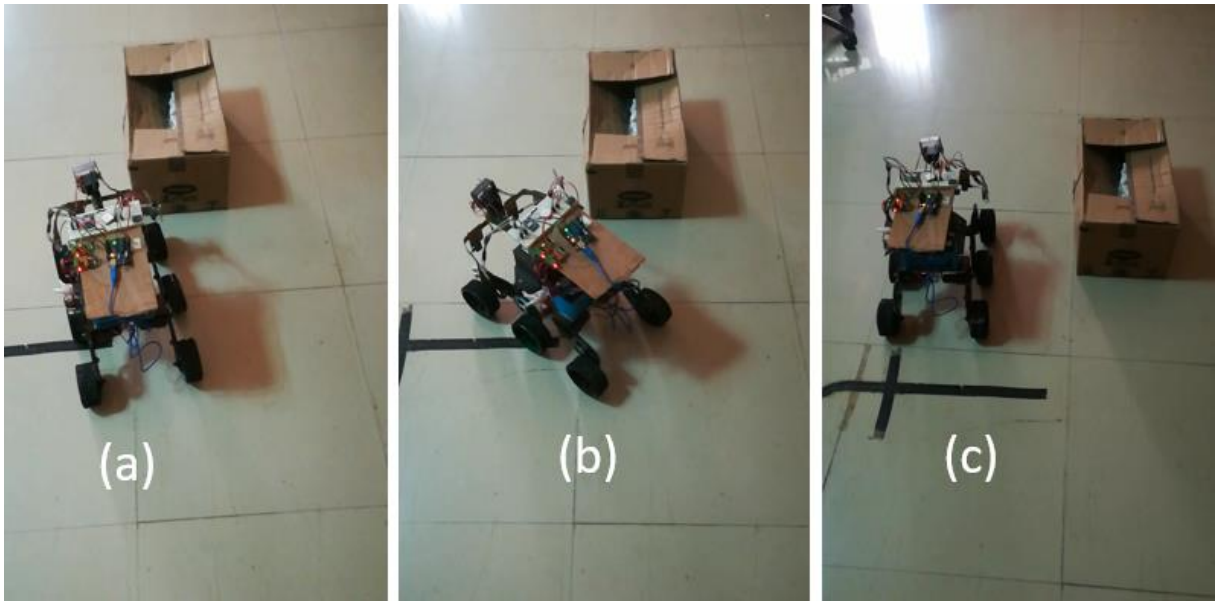


Fig 5.7 Obstacle detection and avoidance when turning is possible on smooth surface.

In fig 5.7 robot is able to detect the obstacle by front ultrasonic sensor and finds the desired way where no obstacle is there using ultrasonic sensor mounted on servo motor. To take turn robot simple rotate about its vertical axis(anticlockwise here) and the move forward. When robot passes from side of obstacle, robot take turn in clockwise to regain same heading followed by it before the obstacle detection.

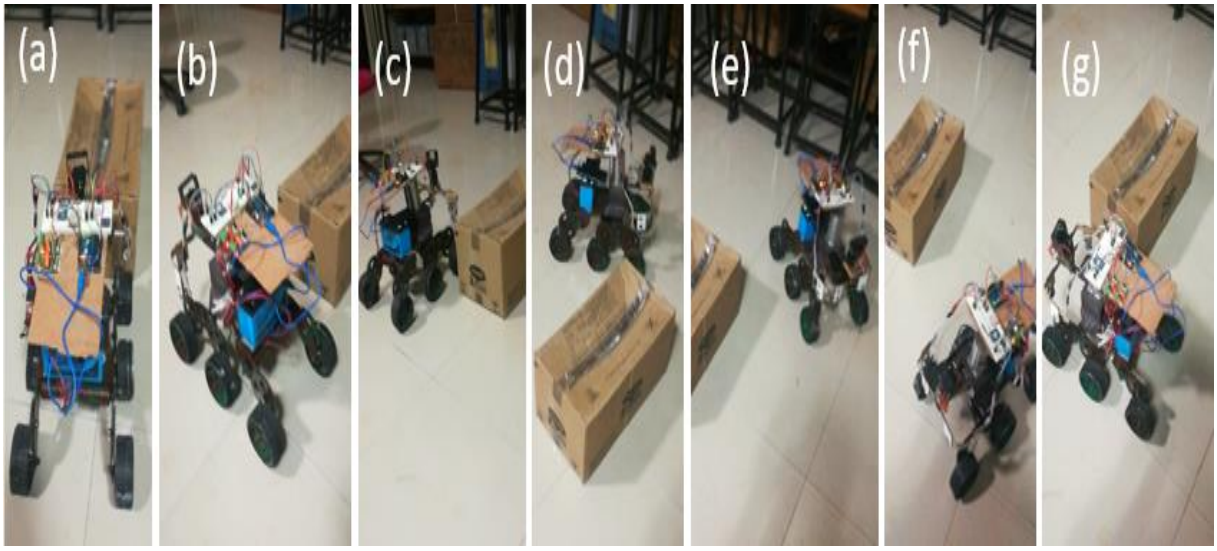


Fig 5.8 Wall following around a cubical box

In fig 5.8, robot is going forward and when a cubical shape obstacle is placed in front of it it starts rotating around it. The surface is very smooth, so at sharp edges robot rotate clockwise about its vertical axis until side sensor read the wall. And algorithm works very fine with circular and triangular object too. But same method is unable to implant for rough surface specially at sharp edges where robot is unable to rotate about is vertical axis, where its semicircular rocker structure rotates by 180 degree.

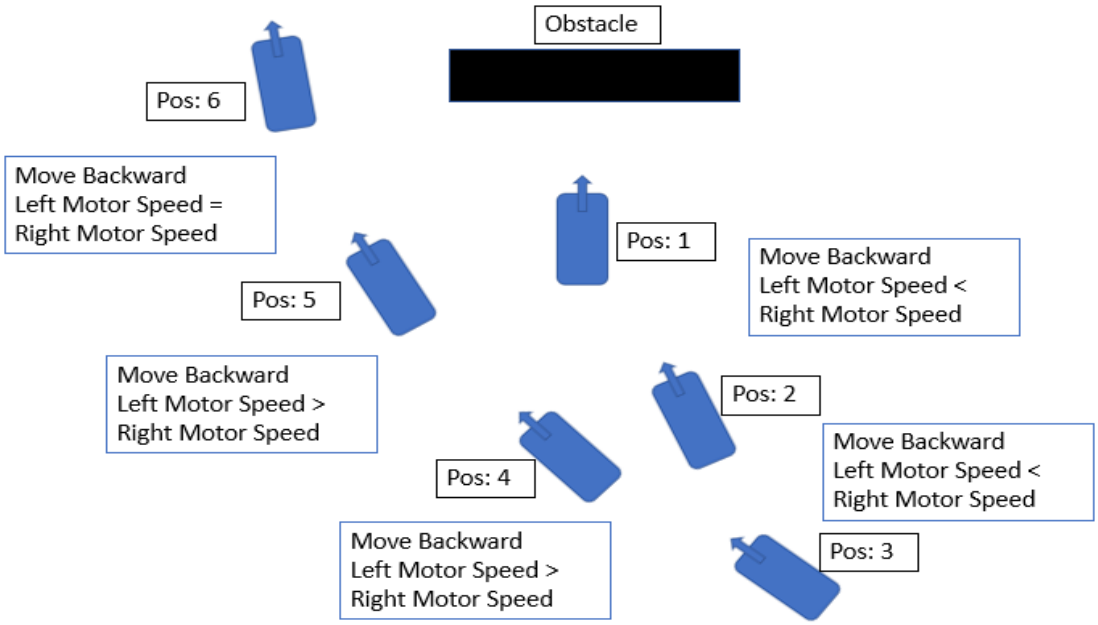


Fig 5.9 Avoiding the obstacle by taking back then turn right

Fig 5.9 explain the method followed in fig 5.6. where robot first come back using differential turn and then again take differential turn for forward motion in opposite direction.

Chapter 6

Conclusions and Future Scope

In this report path planning for Six-Wheeled Multi Terrain Robot (SW-MTR) has been developed. A new designed based on semi-circular rocker mechanism has been proposed which helps the SW-MTR to move on uneven terrains. The rocker mechanism also helps in reduction in torque requirements for the robot while manoeuvring in uneven terrain.

For obstacle and terrain detection and avoidance sensors such as ultrasonic, current sensors and IMUs are employed. GPS has been employed to detect the location of the SW-MTR. Manual controlling via Arduino serial monitor and Play station remote has been done successfully.

The robot was able to move to its target latitude and longitude using GPS module and magnetometer. GPS value will decide targeted distance and target heading, then magnetometer will calculate the current heading and according to positive or negative error, the robot will turn left or right. After reaching in the periphery of goal, an ultrasonic senor mounted on servo motor will rotate from 0-180 degree. And it will detect the position of a target pole mounted there. And cover desired angle and distance to touch that pole.

In addition to that Tangent bug algorithm was aimed to build but not implanted successfully. But a single obstacle avoidance method was adopted by robot while continuously moving toward its goal.

Since for real time path planning method, a fast-single board and a sharp distance sensor will be required in future to implant tangent bug method. the robot needs to be tested on a uneven path. Also, obstacle avoidance method need to be improve so that it can detect and avoid multiple obstacle in static and dynamic condition.

References

[1] J. SHANG, Z. LUO, D. FAN and F. Zhao, “A six wheeled robot with active self-adaptive suspension for LUNAR exploration,” in International Technology and Innovation Conference (ITIC), 2006, 2006.

[2] K. L. Moore and N. S. Flann, “A Six-Wheeled Omnidirectional Autonomous Mobile Robot,” Control System Magazine, vol. 20, no. 6, pp. 53-66, December 2000.

[3] D. Sanders, “Analysis of the effects of time delays on the teleoperation of a mobile robot in various modes of operation,” Emerald, Industrial Robot: An International Journal, vol. 36, no. 6, pp. 570-584, 2009.

[4] N. Keiji, A. Yamasaki, Y. Kazuya and T. Adachi, “Developement and control method of six-wheel robot with rocker structure,” in International Workshop on Safety, Security and Rescue Robotics (SSRR), 2007, 2007

[5] Dung Duong Quoc, Jinwei Sun, Van Nhu Le and Lei Luo, “Complementary Filter Performance Enhancement through Filter Gain”, International Journal of Signal Processing, Image Processing and Pattern Recognition,2015

[6] S.A.Quadri and Othman Sidek, “Error and Noise Analysis in an IMU using Kalman Filter”, International Journal of Hybrid Information Technology,2014.

[7] Pengfei Gui, Liqiong Tang, Subhas Mukhopadhyay, “ MEMS based IMU for tilting measurement: Comparison of complementary and kalman filter based data fusion”, IEEE 10th conference paper on industrial electronics and appliction,2015

[8] Sebastian O.H. Madgwick, “An efficient orientation filter for inertial and inertial/magnetic sensor arrays”, report 2010

[9] Sławomir ROMANIUK* , Zdzisław GOSIEWSKI,” KALMAN FILTER REALIZATION FOR ORIENTATION AND POSITION ESTIMATION ON DEDICATED PROCESSOR”

[10] Ray-Shine Runa , Guo-Gang Sunb,” The cost-effective GPS Guided Autonomous Vehicle – A Feasibility Study based on Self-balancing Scooter”, Proceedings of the 2017 IEEE International Conference on Applied System Innovation

[11]Amit Yadav1 , Ajeet Gaur1 , and D K Chaturvedi2,” Navigation, Guidance & Control Program for GPS based Autonomous Ground Vehicle”, 2017 IJEDR | Volume 5, Issue 4

[12]N. Nath1 , C.Saunders1 and S.H. Lee," IMPLEMENTING A HYBRID BUG ALGORITHM USING LOW COST ULTRASONIC SENSORS FOR NAVIGATION ON FREE RANGE CHICKEN FARMS", 2013 Society for Engineering in Agriculture (SEAg) Conference

[13]https://gunjanpatel.wordpress.com/2016/07/07/complementary-filter-design/

[14]http://scottlobdell.me/2017/01/gps-accelerometer-sensor-fusion-kalman-filter-practical-walkthrough/

[15] https://github.com/TKJElectronics/KalmanFilter

[16] https://github.com/mkconer/GPSRobot

[17] https://github.com/LamaNIkesh/MazeSolver-NoIntelligence

[18] https://github.com/bolderflight/uNavINS

[19] https://ieeexplore.ieee.org/document/5946136

[20] https://nptel.ac.in/courses/108106098/46

[21] http://www.pieter-jan.com/node/11

[22] http://arduiniana.org/libraries/tinygps/

[23]https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/

[24] http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

[25] https://www.movable-type.co.uk/scripts/latlong.html

[26] http://ardupilot.org/rover/docs/common-gps-how-it-works.html

[27] http://x-io.co.uk/open-source-imu-and-ahrs-algorithms/

Appendix

1. NEO GPS 6M NMEA Parsing [22]

NMEA format:

- $GPGGA: Global Positioning System Fix Data
- $GPGSV: GPS satellites in view
- $GPGSA: GPS DOP and active satellites
- $GPRMC: Recommended minimum specific GPS/Transit data

These strings contain many GPS parameters like: Time, Date, Longitude, Latitude, speed, no. of satellites in used, altitude and many other things. For any location coordinates and time, we can use $GPGGA and $GPRMC. For Date and time we can use $GPRMC string.

Fig A.1 GPS Data when receiver is not in range

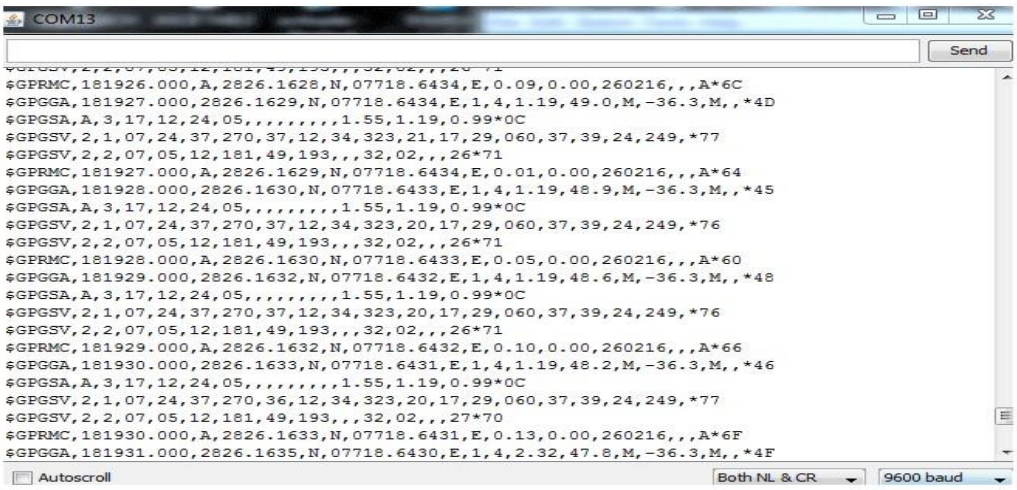


Fig A.2 GPS Data when receiver is in range

When we use GPS module **for tracking any location, we only need coordinates and we can find this in $GPGGA string**. Only $GPGGA (Global Positioning System Fix Data) String is mostly used in programs and other strings are ignored. This string consists fix data as below:

$GPGGA,104534.000,7791.0381,N,06727.4434,E,1,08,0.9,510.4,M,43.9,M,,*47

$GPGGA,HHMMSS.SSS,latitude,N,longitude,E,FQ,NOS,HDP,altitude,M,height,M,,checks um data

Identifier	Description
$GPGGA	Global Positioning system fix data
HHMMSS.SSS	Time in hour minute seconds and milliseconds format.
Latitude	Latitude (Coordinate)
N	Direction N=North, S=South
Longitude	Longitude(Coordinate)
E	Direction E= East, W=West
FQ	Fix Quality Data
NOS	No. of Satellites being Used
HDP	Horizontal Dilution of Precision
Altitude	Altitude (meters above from sea level)
M	Meter
Height	Height
Checksum	Checksum Data

Table A.1 Identifiers and Description

And **$GPRMC string** mainly contains velocity, time, date and position

$GPRMC,123519.000,A, 7791.0381,N, 06727.4434,E,022.4,084.4,230394,003.1,W*6A

$GPRMC,HHMMSS.SSS,A,latitude,N,longitude,E,speed,angle,date,MV,W,CMD

Identifier	Description
RMC	Recommended Minimum sentence C
HHMMSS.SSS	Time in hour minute seconds and milliseconds format.
A	Status // A=active and V= void
Latitude	Latitude 49 deg. 16.45 min. North
N	Direction N=North, S=South
Longitude	Longitude(Coordinate)
E	Direction E= East, W=West
Speed	speed in knots
Angle	Tracking angle in degrees
Date	Time stamp (Date in UTC)
MV	Magnetic Variation
W	Direction of variation E/W
CMD (*6A)	Checksum Data

Table A.2 Identifiers and Descriptions

2. PID controller [24]

2.1 Working:

A PID algorithm in a system has a set-value called "set point" of a particular physical quantity needs to be controlled when changes due to some external noise, the system controls the disturbance or more simply the error, to get back to the original set point in the least time possible. PID control a physical quantity's value and to make it equal to a specified value.

P-controller: Proportional control gives output which is directly proportional to current error of the plant. It compares set value with actual value and the resulting difference is multiplied with proportional constant, Kp, to get the result. If the error value is zero, then this controller response will be zero.

I-Controller: By use of only P-controller it gives an offset between the process variable and set value, then I-controller is required, which eliminate the steady state error. It integrates the error over the whole period of time until error becomes zero. Ki proportional constant is multiplied in I controller.

D-Controller: D-controller predict the upcoming behavior of the error, which lack in Icontroller. It is the rate of change of error with respect to time, multiplied by derivative constant, Kd. This stimulate the result thereby increasing system response.

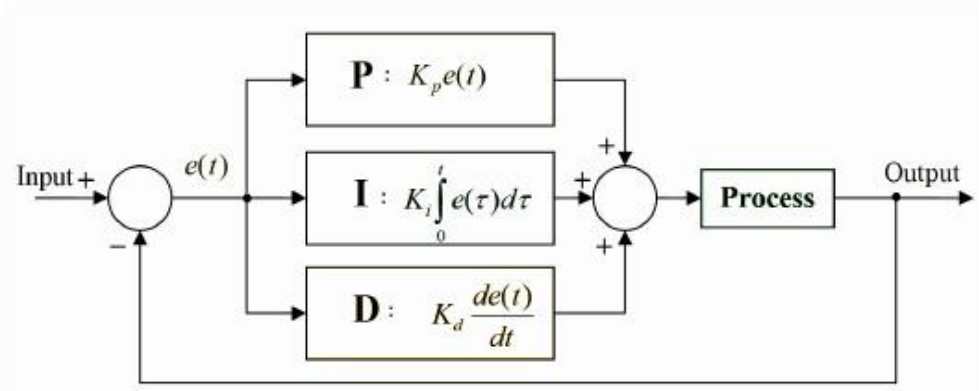


Fig A.3 PID controller [24]

2.2 Calculate PID

Proportional termed as P and it will directly store the weighted value assigned for array configuration.

- P = error

Integral termed as I and will sum all previous error.

- I = error + previous_I_value

Derivative termed as D and it is the difference between the instantaneous error from the set point, and the error from the previous instant.

- D = e – last_error_value

Equation: PID_value = (Kp*P) + (Ki*I) + (Kd*D)

The PID control scheme is implanted and helps bot to be stable and to turn smoothly. **Kp** used to vary the magnitude of the change required to get the set point. **Ki** used to vary the rate at which the change should be brought to achieve the set value. **Kd** used to vary the stability.

3. Mathematical formula used: [25]

3.1 To calculate distance to way point using GPS

```
float currentLong = choose_currentLong;
float currentLat=choose_currentLat;
float targetLong = choose_targetLong;
float targetLat=choose_targetLat;

float delta = radians(currentLong - targetLong);
float sdlong = sin(delta);
float cdlong = cos(delta);
float lat1 = radians(currentLat);
float lat2 = radians(targetLat);
float slat1 = sin(lat1);
float clat1 = cos(lat1);
float slat2 = sin(lat2);
float clat2 = cos(lat2);
delta = (clat1 * slat2) - (slat1 * clat2 * cdlong);
delta = sq(delta);
delta += sq(clat2 * sdlong);
delta = sqrt(delta);
float denom = (slat1 * slat2) + (clat1 * clat2 * cdlong);
delta = atan2(delta, denom);
int distanceToTarget =  delta * 6372795;
return distanceToTarget;
```

3.2 To calculate target heading using GPS

```
float currentLong = choose_currentLong;
float currentLat=choose_currentLat;
float targetLong = choose_targetLong;
float targetLat=choose_targetLat;

float dlon = radians(targetLong-currentLong);
float cLat = radians(currentLat);
float tLat = radians(targetLat);
float a1 = sin(dlon) * cos(tLat);
float a2 = sin(cLat) * cos(tLat) * cos(dlon);
a2 = cos(cLat) * sin(tLat) - a2;
a2 = atan2(a1, a2);
if (a2 < 0.0)
{
  a2 += TWO_PI;
}
int targetHeading = degrees(a2);
return targetHeading;
```

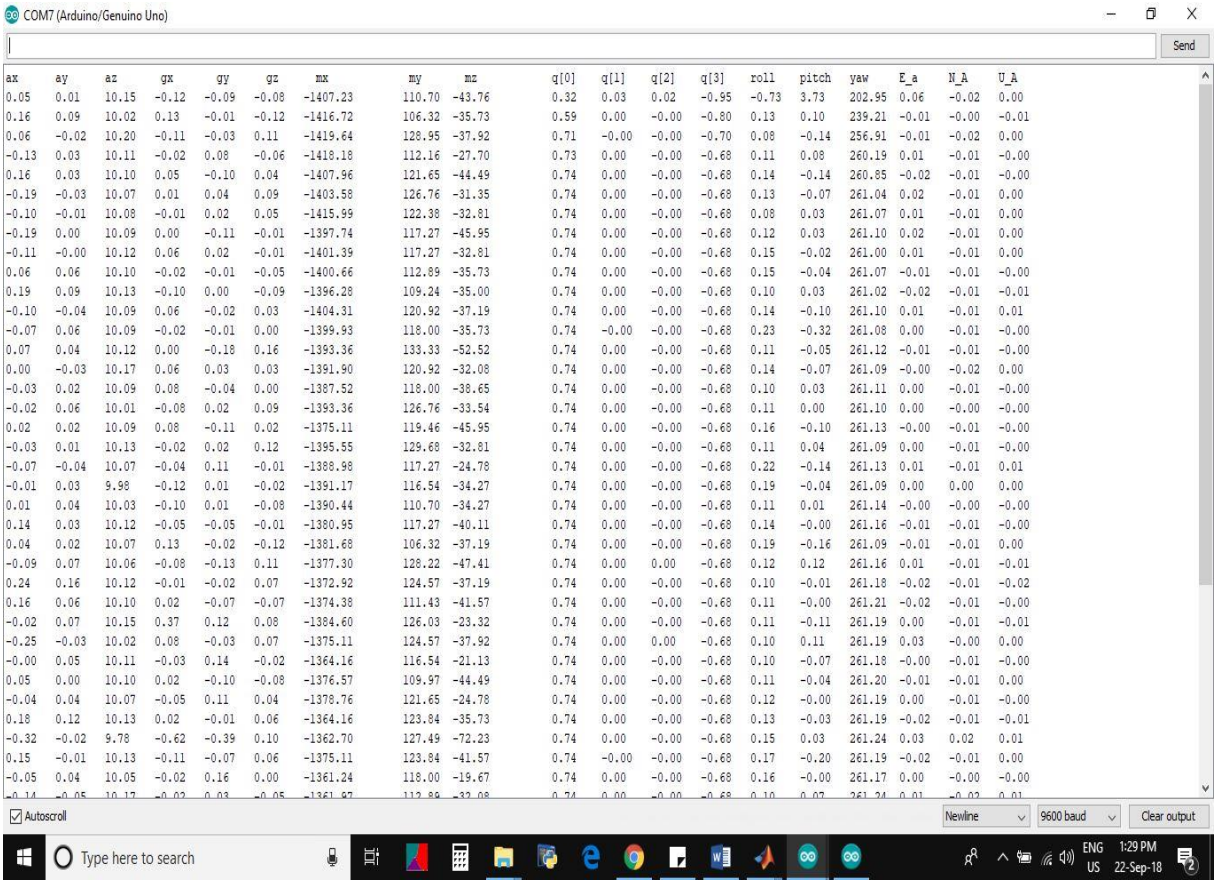
3.2 To calculate current orientation using compass

```
Wire.beginTransmission(hmc5883Address);
Wire.write(hmcDataOutputXMSBAddress);  //Select register 3, X MSB register
Wire.endTransmission();
//Read data from each axis of the Digital Compass
Wire.requestFrom(hmc5883Address,6);
if(6<=Wire.available())
{
  x = Wire.read()<<8; //X msb
  x |= Wire.read();   //X lsb
  z = Wire.read()<<8; //Z msb
  z |= Wire.read();   //Z lsb
  y = Wire.read()<<8; //Y msb
  y |= Wire.read();   //Y lsb
}
int heading = atan2(y,x)/M_PI*180;
// #define DEC_ANGLE (3.84/1000)/M_PI*180;
// heading += DEC_ANGLE;
heading=heading+15+25;
if(heading < 0)
  heading += 360;
if(heading > 360)
  heading -= 360;
return((int)heading);
```

3.4 To calculate Heading error by comparing HMC5833l & GPS

```
float targetHeading=choose_targetHeading;
float heading=choose_heading;
int headingError = targetHeading - heading;
headingError=headingError;
  if (headingError < -180)
    headingError += 360;
  if (headingError > 180)
    headingError -= 360;
delay(100);
return(headingError);
```

4. Arduino IDE data:



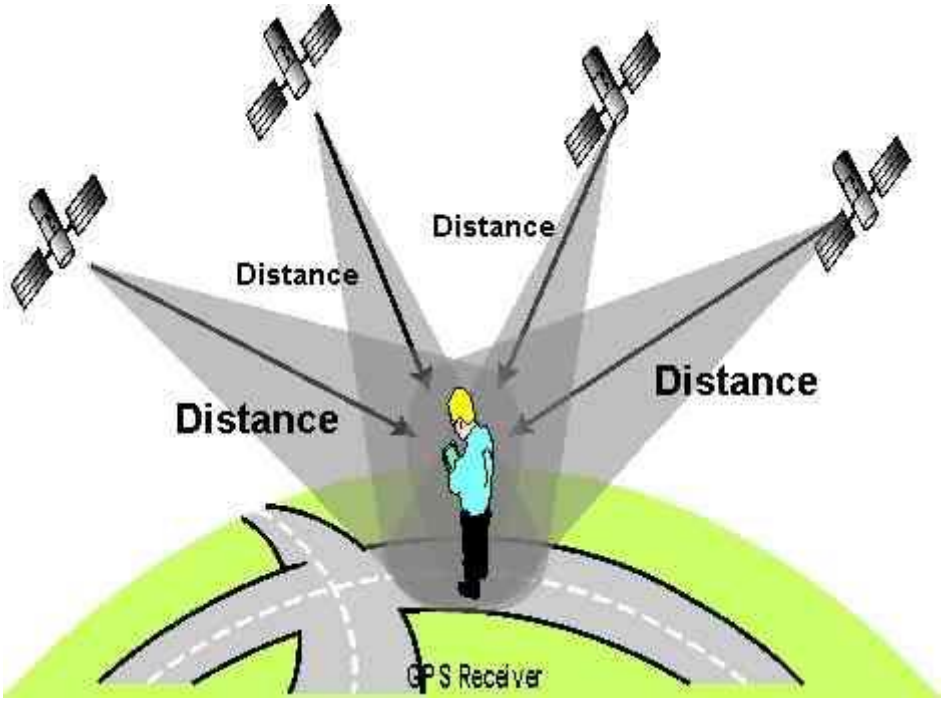
ax	ay	az	gx	gy	gz	mx	my	mz	q[0]	q[1]	q[2]	q[3]	roll	pitch	yaw	E_a	N_A	U_A
0.05	0.01	10.15	-0.12	-0.09	-0.08	-1407.23	110.70	-43.76	0.32	0.03	0.02	-0.95	-0.73	3.73	202.95	0.06	-0.02	0.00
0.16	0.09	10.02	0.13	-0.01	-0.12	-1416.72	106.32	-35.73	0.59	0.00	-0.00	-0.80	0.13	0.10	239.21	-0.01	-0.00	-0.01
0.06	-0.02	10.20	-0.11	-0.03	0.11	-1419.64	128.95	-37.92	0.71	-0.00	-0.00	-0.70	0.08	-0.14	256.91	-0.01	-0.02	0.00
-0.13	0.03	10.11	-0.02	0.08	-0.06	-1418.18	112.16	-27.70	0.73	0.00	-0.00	-0.68	0.11	0.08	260.19	0.01	-0.01	-0.00
0.16	0.03	10.10	0.05	-0.10	0.04	-1407.96	121.65	-44.49	0.74	0.00	-0.00	-0.68	0.14	-0.14	260.85	-0.02	-0.01	-0.00
-0.19	-0.03	10.07	0.01	0.04	0.09	-1403.58	126.76	-31.35	0.74	0.00	-0.00	-0.68	0.13	-0.07	261.04	0.02	-0.01	0.00
-0.10	-0.01	10.08	-0.01	0.02	0.05	-1415.99	122.38	-32.81	0.74	0.00	-0.00	-0.68	0.08	0.03	261.07	0.01	-0.01	0.00
-0.19	0.00	10.09	0.00	-0.11	-0.01	-1397.74	117.27	-45.95	0.74	0.00	-0.00	-0.68	0.12	0.03	261.10	0.02	-0.01	0.00
-0.11	-0.00	10.12	0.06	0.02	-0.01	-1401.39	117.27	-32.81	0.74	0.00	-0.00	-0.68	0.15	-0.02	261.00	0.01	-0.01	0.00
0.06	0.06	10.10	-0.02	-0.01	-0.05	-1400.66	112.89	-35.73	0.74	0.00	-0.00	-0.68	0.15	-0.04	261.07	-0.01	-0.01	-0.00
0.19	0.09	10.13	-0.10	0.00	-0.09	-1396.28	109.24	-35.00	0.74	0.00	-0.00	-0.68	0.10	0.03	261.02	-0.02	-0.01	-0.01
-0.10	-0.04	10.09	0.06	-0.02	0.03	-1404.31	120.92	-37.19	0.74	0.00	-0.00	-0.68	0.14	-0.10	261.10	0.01	-0.01	0.01
-0.07	0.06	10.09	-0.02	-0.01	0.00	-1399.93	118.00	-35.73	0.74	-0.00	-0.00	-0.68	0.23	-0.32	261.08	0.00	-0.01	-0.00
0.07	0.04	10.12	0.00	-0.18	0.16	-1393.36	133.33	-52.52	0.74	0.00	-0.00	-0.68	0.11	-0.05	261.12	-0.01	-0.01	-0.00
0.00	-0.03	10.17	0.06	0.03	0.03	-1391.90	120.92	-32.08	0.74	0.00	-0.00	-0.68	0.14	-0.07	261.09	-0.00	-0.02	0.00
-0.03	0.02	10.09	0.08	-0.04	0.00	-1387.52	118.00	-38.65	0.74	0.00	-0.00	-0.68	0.10	0.03	261.11	0.00	-0.01	-0.00
-0.02	0.06	10.01	-0.08	0.02	0.09	-1393.36	126.76	-33.54	0.74	0.00	-0.00	-0.68	0.11	0.00	261.10	0.00	-0.00	-0.00
0.02	0.02	10.09	0.08	-0.11	0.02	-1375.11	119.46	-45.95	0.74	0.00	-0.00	-0.68	0.16	-0.10	261.13	-0.00	-0.01	-0.00
-0.03	0.01	10.13	-0.02	0.02	0.12	-1395.55	129.68	-32.81	0.74	0.00	-0.00	-0.68	0.11	0.04	261.09	0.00	-0.01	-0.00
-0.07	-0.04	10.07	-0.04	0.11	-0.01	-1388.98	117.27	-24.78	0.74	0.00	-0.00	-0.68	0.22	-0.14	261.13	0.01	-0.01	0.01
-0.01	0.03	9.98	-0.12	0.01	-0.02	-1391.17	116.54	-34.27	0.74	0.00	-0.00	-0.68	0.19	-0.04	261.09	0.00	0.00	0.00
0.01	0.04	10.03	-0.10	0.01	-0.08	-1390.44	110.70	-34.27	0.74	0.00	-0.00	-0.68	0.11	0.01	261.14	-0.00	-0.00	-0.00
0.14	0.03	10.12	-0.05	-0.05	-0.01	-1380.95	117.27	-40.11	0.74	0.00	-0.00	-0.68	0.14	-0.00	261.16	-0.01	-0.01	-0.00
0.04	0.02	10.07	0.13	-0.02	-0.12	-1381.68	106.32	-37.19	0.74	0.00	-0.00	-0.68	0.19	-0.16	261.09	-0.01	-0.01	0.00
-0.09	0.07	10.06	-0.08	-0.13	0.11	-1377.30	128.22	-47.41	0.74	0.00	0.00	-0.68	0.12	0.12	261.16	0.01	-0.01	-0.01
0.24	0.16	10.12	-0.01	-0.02	0.07	-1372.92	124.57	-37.19	0.74	0.00	-0.00	-0.68	0.10	-0.01	261.18	-0.02	-0.01	-0.02
0.16	0.06	10.10	0.02	-0.07	-0.07	-1374.38	111.43	-41.57	0.74	0.00	-0.00	-0.68	0.11	-0.00	261.21	-0.02	-0.01	-0.00
-0.02	0.07	10.15	0.37	0.12	0.08	-1384.60	126.03	-23.32	0.74	0.00	-0.00	-0.68	0.11	-0.11	261.19	0.00	-0.01	-0.01
-0.25	-0.03	10.02	0.08	-0.03	0.07	-1375.11	124.57	-37.92	0.74	0.00	0.00	-0.68	0.10	0.11	261.19	0.03	-0.00	0.00
-0.00	0.05	10.11	-0.03	0.14	-0.02	-1364.16	116.54	-21.13	0.74	0.00	-0.00	-0.68	0.10	-0.07	261.18	-0.00	-0.01	-0.00
0.05	0.00	10.10	0.02	-0.10	-0.08	-1376.57	109.97	-44.49	0.74	0.00	-0.00	-0.68	0.11	-0.04	261.20	-0.01	-0.01	0.00
-0.04	0.04	10.07	-0.05	0.11	0.04	-1378.76	121.65	-24.78	0.74	0.00	-0.00	-0.68	0.12	-0.00	261.19	0.00	-0.01	-0.00
0.18	0.12	10.13	0.02	-0.01	0.06	-1364.16	123.84	-35.73	0.74	0.00	-0.00	-0.68	0.13	-0.03	261.19	-0.02	-0.01	-0.01
-0.32	-0.02	9.78	-0.62	-0.39	0.10	-1362.70	127.49	-72.23	0.74	0.00	-0.00	-0.68	0.15	0.03	261.24	0.03	0.02	0.01
0.15	-0.01	10.13	-0.11	-0.07	0.06	-1375.11	123.84	-41.57	0.74	-0.00	-0.00	-0.68	0.17	-0.20	261.19	-0.02	-0.01	0.00
-0.05	0.04	10.05	-0.02	0.16	0.00	-1361.24	118.00	-19.67	0.74	0.00	-0.00	-0.68	0.16	-0.00	261.17	0.00	-0.00	-0.00

Fig A.4 Arduino IDE Result of a Quaternion based filter

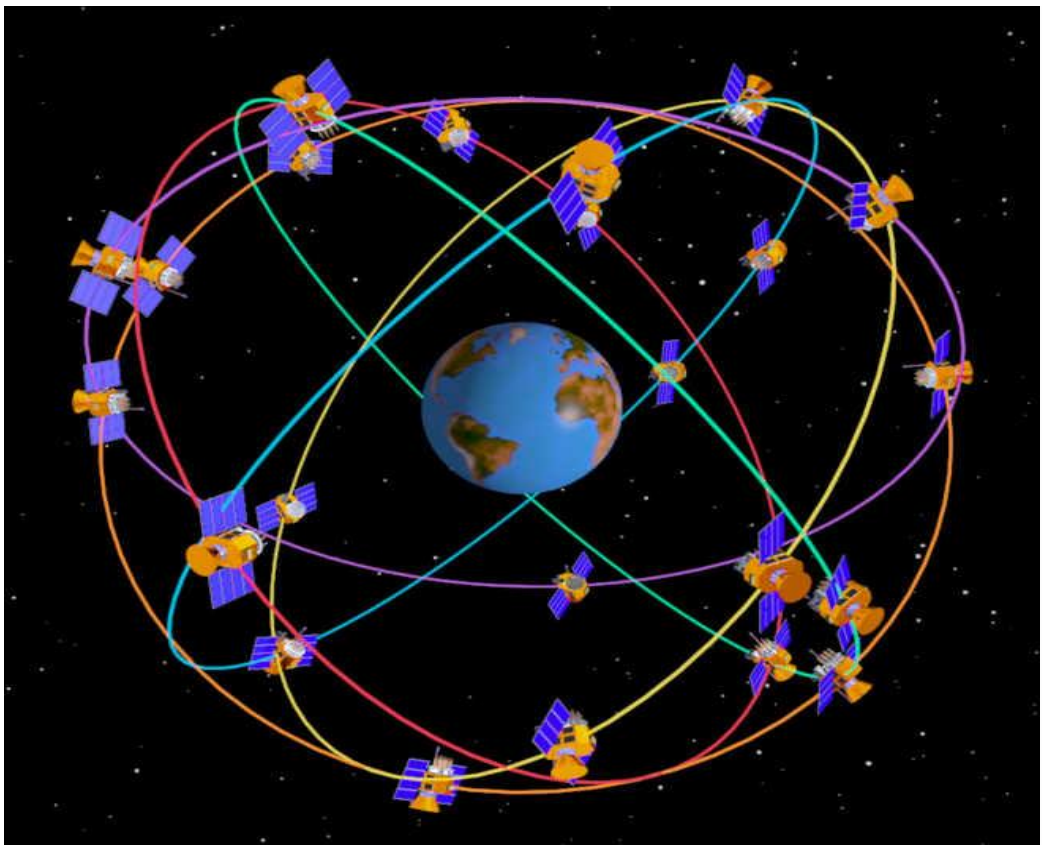
Fig 3.4 shows the all king of value obtained and calculated using Madgwick Quaternion update AHRS system. Data obtained are follow, ax,ay,az,gx,gy,gz,mx,my,mz by accelerometer, gyroscope and magnetometer. Then q[0],q[1],q[2],q[3] quaternion values are obtained by passing ax,ay,az,gx,gy,gz,mx,my,mz through Madgwick Quaternion Filter. Using q[0],q[1],q[2],q[3] values roll, pitch,yaw as well as Lnear acceleration in East, North and Down direction cn be calculated. These results are sensor data when senso is placed in static condition.

5. GPS - How it Works [26]

GPS system is a Global Navigation Satellite System (GNSS).



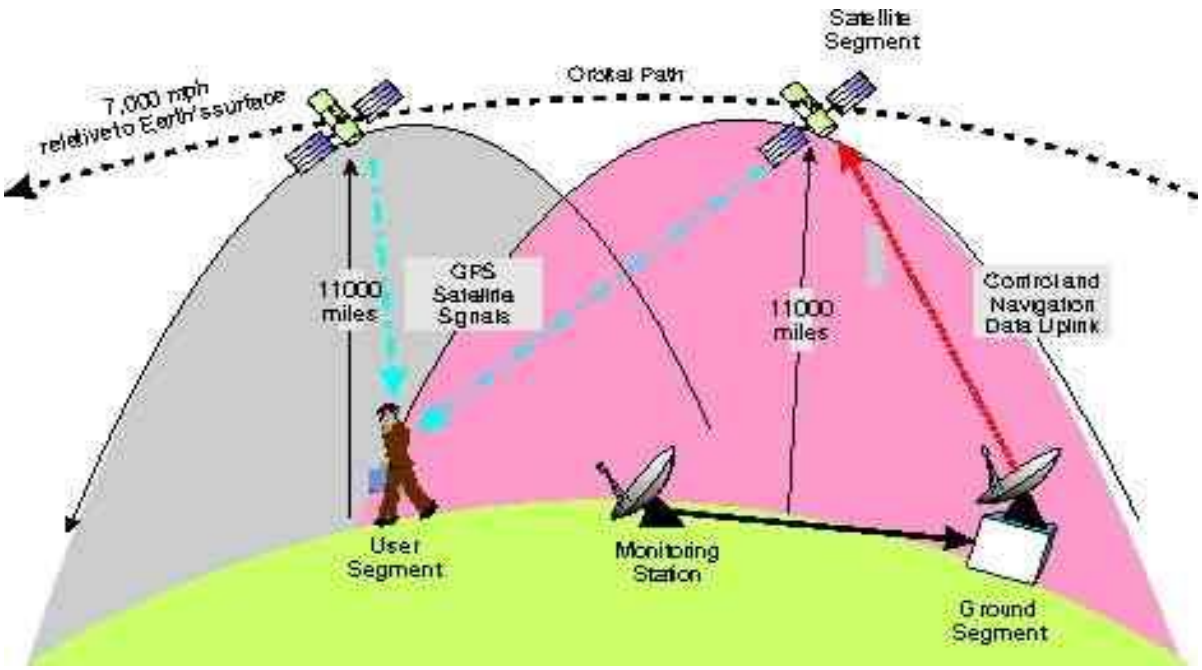
Most civil GPS receivers are using pseudorange data (C/A code) available on GPS L1 channel (1575.42 MHz) and optionally they can receive SBAS DGPS corrections giving meter precision.



Advanced and expensive civil GPS receivers can use the previous method plus carrier phase smoothing, carrier phase corrections (RTK) as well as L2 channel P(Y) code semi-codeless tracking (1227.60 MHz) for real-time local ionospheric corrections giving centimeter or even millimeter precision.

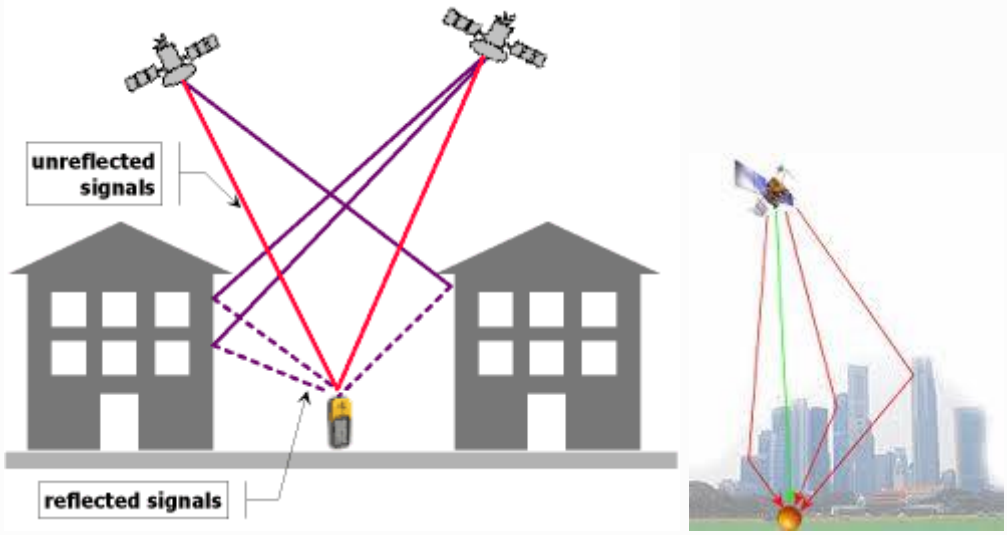
Some even more advanced GNSS receivers can combine DGPS and RTK corrections and can receive other GNSS satellite constellations (GLONASS and GALILEO) and other channels (L2C, L5...) at the same time to enhance precision and fix reliability.

Military GPS receivers are able to decode the P(Y) code available on L1 and L2 channels giving ten times more precision in real-time compared to a pseudo range basic solution.



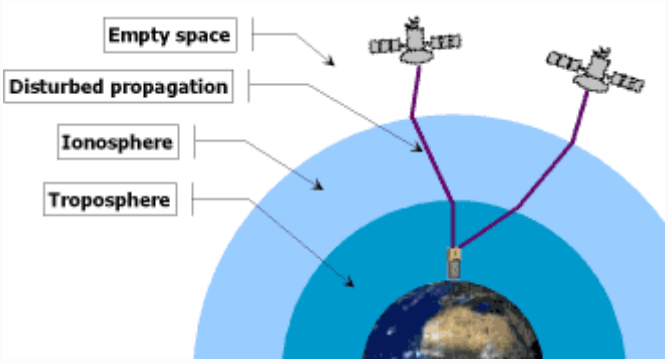
5.1 GPS multipathing

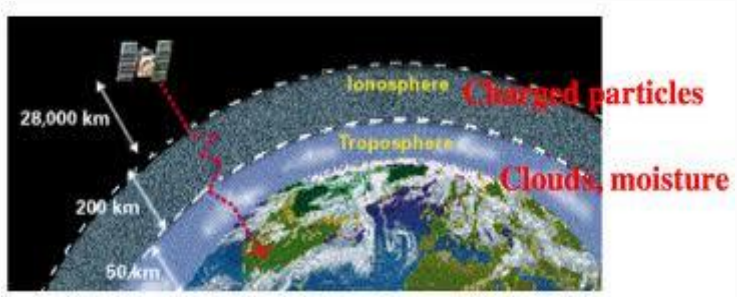
Multipathing is causing GPS position errors that are difficult to detect and compensate for. Multiple GNSS receivers are better suited to filter out multipathing thanks to a higher number of visible satellites at the same time.



5.2 GPS disturbed signal propagation

Ionospheric perturbations and magnetic storms (solar activity) can cause signal delays. This can be partially compensated using SBAS and L1 / L2 decoding.





Decoding L1 and L2 channels at the same time do allow real-time ionospheric corrections through a differential technique. Unlike SBAS broadcasted ionospheric corrections, this method gives a better match to local ionospheric conditions.

New channels (L2C, L5) for civil use will enhance signal acquisition and position precision as soon as new satellites constellations will be complete (2015 - 2020).

GPS performance data for different scenarios. Those values can be thought as a basis to understand GPS limitations. As a general rule only, clear sky view is safe enough for Auto missions!

GPS 3D precision:

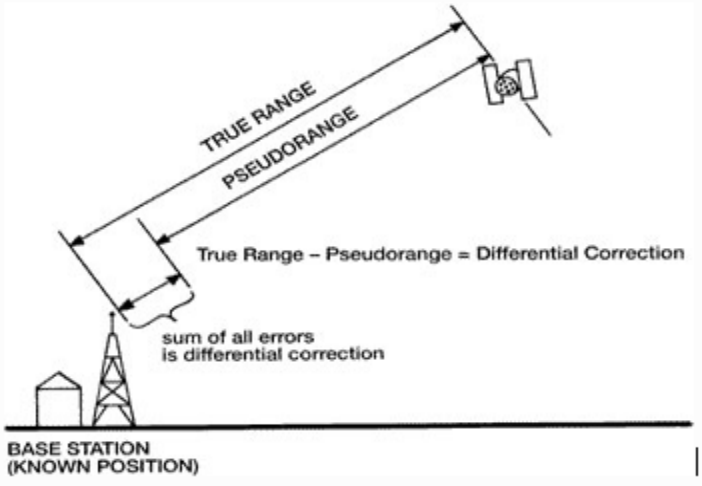
- Free Horizon (clear sky view): 2.5 m
- Extreme Multipath (Backyards or between buildings flights): 26 m
- Indoor: 55 m

GPS Availability:

- Free Horizon (clear sky view) : 99%
- Extreme Multipath (Backyards or between buildings flights) : 83 %
- Indoor: 14 %

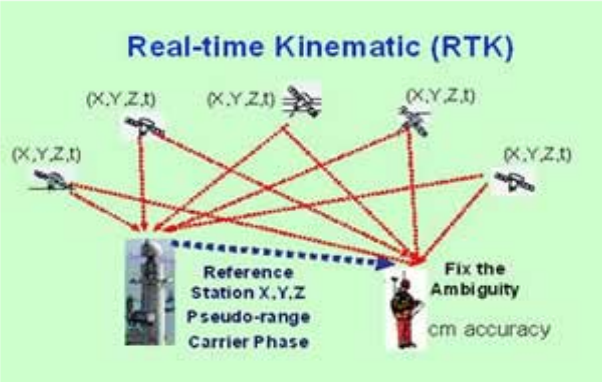
5.3 DGPS corrections

DGPS corrections are using pseudo range data only. SBAS is a form of DGPS correction freely available from one or more geostationary satellites. Each region of the world has a localized SBAS system: WAAS for America, EGNOS for Europe, MSAS for Japan.DGPS corrections give meter precision (free services) or decimeter precision (paying services)



5.4 RTK corrections

RTK corrections are using carrier phase information and needs costly advanced GPS as well as a RTK base station or a Network of RTK base stations broadcasting corrections over a modem or Internet through the RTCM protocol.



RTK is easier to use with slow motion or static applications (like surveying or farming) and give centimeter or millimeter precision.

6. Rotation matrix

Rotation Matrix ↔ Quaternion Conversion

$$\mathbf{q} = (w, x, y, z) \mapsto \mathbf{R} = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2xy - 2wz & 2xz + 2wy \\ 2xy + 2wz & w^2 - x^2 + y^2 - z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & w^2 - x^2 - y^2 + z^2 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \mapsto \mathbf{q} = [w, (x, y, z)] \text{ where } \begin{cases} w = \sqrt{\dfrac{1 + m_{11}^2 + m_{22}^2 + m_{33}^2}{4}} \\ x = \dfrac{m_{32} - m_{23}}{4w} \\ y = \dfrac{m_{13} - m_{31}}{4w} \\ z = \dfrac{m_{21} - m_{12}}{4w} \end{cases}$$

7. Pictures of Robots after Steering Modification