Autonomous Navigation of Wheeled Robot in Outdoor Environment

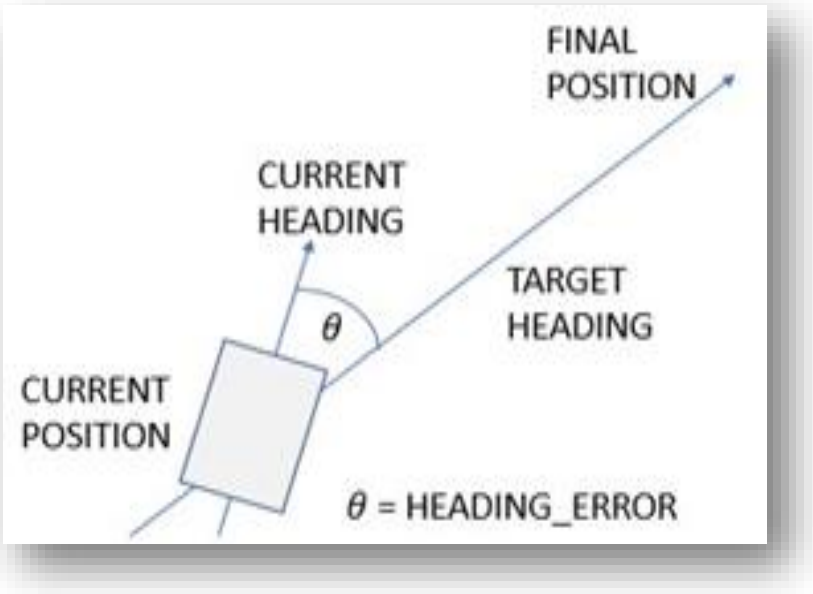
Components Used:

1. A differential drive robot
2. GPS module
3. Inertial Measurement Unit (IMU)
4. Ultrasonic Sensor

Since GPS has position inaccuracy of 3-4m so we can only reach in periphery of goal, not at an exact goal. Also, Filters like Kalman Filter and Complementary Filter has been used to get stable Orientation (Roll, Pitch and Yaw of robot).

PART A: Reaching Around Goal Position:

1. By GPS we will calculate **Target Heading** and **Target Distance** for the robot and robot will move till the Target Distance will be zero.
2. **Current heading** will be calculated using Magnetometer.
3. The difference between Target Heading and Current Heading will be called **Heading Error.** For a positive Heading error robot will take a right turn, for negative left turn and for zero, it will go straight.



1. Main Loop

```
void loop() {
  while(serial_connection.available())  // waiting for GPS to get stable data
  { gps.encode(serial_connection.read()); }

  if(gps.location.isUpdated()){
 // Calculating the Target Diatstance using GPS data
  TargetDistance=distanceToWaypoint(gps.location.lng(), gps.location.lat(), 80.026000,23.178585);
  if (TargetDistance<=2) {
     stoprobot();
     delay(10000);}

 //Calculating the Target Heading using GPS data
  TargetHeading=courseToWaypoint(gps.location.lng(), gps.location.lat(), 80.026000,23.178585);

 //Calculating the Current Heading using Magnetometer data
  CurrentHeading=readCompass();

//Claculating Heading Error of Robot
   HeadingDifference=calcDesiredTurn(TargetHeading,CurrentHeading);

//Choosing Left or Right Turn on basis of positive or negative Heading Error
  motorSpeed(HeadingDifference);  delay(500);
  }
```

2. Calculating Target Distance

```
int distanceToWaypoint(float choose_currentLong,float choose_currentLat, float choose_targetLong, float choose_targetLat)
{
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. Calculate Target Heading:

```
int courseToWaypoint(float choose_currentLong,float choose_currentLat, float choose_targetLong, float choose_targetLat){
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;
}
```

4. Calculate Current Heading

```
 int readCompass(){
  Wire.beginTransmission(hmc5883Address);
  Wire.write(hmcDataOutputXMSBAddress);
  Wire.endTransmission();
  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;

if(heading < 0)
  heading += 360;
if(heading > 360)
  heading -= 360;

return((int)heading);
 }
```

5. Calculating Heading Error (between -180 to 180)

```
int  calcDesiredTurn(int choose_targetHeading, int choose_heading){

 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);
}
```

6. Assigning motor speed via PWM for left and right turn

Case 1: HeadingError<5 && HeadingError > -5

Go forward

Case 2: HeadingError >=5 && HeadingError <=10

Left Motor RPM > Right Motor RPM

Case 3: HeadingError >= 11 && HeadingError <=60

Left Motor RPM >>> Right Motor RPM

Case 4: HeadingError > 60

Rotate clockwise

Case 5: HeadingError >= -10 && HeadingError <=-5

Right Motor RPM > Left Motor RPM

Case 6: HeadingError >= -60 && HeadingError <=-11

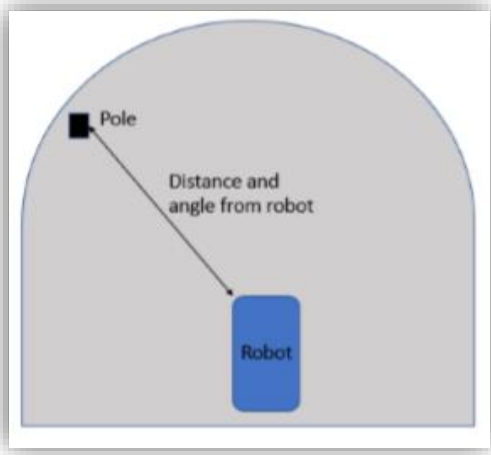
Right Motor RPM >> > Left Motor RPM

Case 7: HeadingError < -60

Rotate Anticlockwise

PART B: Pole Detection and Moving Toward It

1. Place a ultrasonic sensor on a servo motor which can rotate 180 deg.
2. Rotate servo from 0 to 180 by stepping 10 deg.
3. When the ultrasonic sensor will detect the pole, note the Servo angle reading and Distance from Pole.
4. Rotate the robot in Angle of pole by feedback of magnetometer.
5. Move till robot will reach to the pole.



Step 1: Rotate Servo to find Pole Angle and Distance

```
//Rotate and calculate Distance and Angle of pole
 int pos = 0;
 for (pos = 0; pos <= 180; pos += 10)
 {
   myservo.write(pos);
   Serial.print(pos);
   delay(50);
   unsigned int uS = sonar.ping();
   Serial.print("   "); Serial.println(" cm");
   Serial.print(uS / US_ROUNDTRIP_CM);
   if ( (uS / US_ROUNDTRIP_CM)<=150) {
    Angle=pos;
    Distance= uS / US_ROUNDTRIP_CM;
    }
    delay(800);
 }

  angleorient=(90-Angle);
  stoprobot();  delay(300);
  initialorient= readCompass();
```

2. If Pole is between 0 to 89 deg Rotate AntiClockwise

```
if( angleorient>0){
      currentorient=readCompass();
      if(initialorient >= (360-angleorient)){
          currentorient=readCompass();
          while(abs(initialorient-currentorient)<abs(365-initialorient))   {
             rotateanticlock();
             alreadyroatated=abs(initialorient-currentorient);
             delay(20);
             currentorient=readCompass();
           }
           stoprobot();     delay(300);
           currentorient=readCompass();
           while(abs(currentorient)<abs(angleorient-alreadyroatated-5)){
              rotateanticlock();
              currentorient=readCompass();
           }
      }
      else {
           while(abs(initialorient-currentorient)<angleorient){
           rotateanticlock(); delay(20);
           currentorient=readCompass();
           }
      }
  }
```

3. If Pole is between 91 to 180 deg Rotate Clockwise

```
if( angleorient<0){
    currentorient=readCompass();
    if(initialorient > abs(angleorient)){
          while(abs(initialorient-currentorient)<abs(angleorient)){
              rotateclock(); delay(20);
              currentorient=readCompass();
           }
     }
     else {
         currentorient=readCompass(); initialorient=readCompass();
         while(abs(initialorient-currentorient)<abs(initialorient+5)) {
              rotateclock();       delay(20);
              alreadyroatated=abs(initialorient-currentorient);
              currentorient=readCompass();
        }
         stoprobot();     delay(100);
         currentorient=readCompass();
         initialorient=readCompass();
         while(abs(initialorient-currentorient)-5<abs(abs(angleorient)-alreadyroatated)){
               rotateclock(); delay(20);
               currentorient=readCompass();
         }
     }
}
```

4. Move till Robot Reached the Pole

```
  while(Distance>5)
  {
  moveforward();
  delay(50);
  }

}
```