SPECTULATION DOCUMENT

ANTENNA TRACKER

Description

What a life without GPS😊

Our project can make your life a lot easier😉

In this project basically we try to connect the flying device (or object) such as a drone to an antenna that rotates pointing towards that object using both software (programming and using Arduino) and hardware(motor controllers).

Prerequisites

- ✓ GPS coordinates
- ✓ Their representations (two types)
- ✓ GPS protocols (NMEA sentences and their types)
- ✓ Haversine Formula (to calculate the distance between two GPS coordinates)
- ✓ Bearing angle (horizontal angle between the object and the true north)
- ✓ Arduino software and Arduino IDE
- ✓ How to code in Arduino (similar to C language) and it's simulation.
- ✓ How to code in processing language.
- ✓ PID_v1 library for Arduino.

Setting Up Arduino & Processing IDE >>

Here are the details about how we got set up the environment:

Arduino IDE Setup:

1. Go to this link https://www.arduino.cc/en/Main/Software and download Arduino IDE according to your system.
2. Extract the zip file wherever you want.
3. Open the extracted folder and run Arduino.exe.

4. Your Arduino IDE is ready to use.
5. Now to download PID_v1 library go to PID - Arduino Libraries and download the latest version of PID library
6. Now open your Arduino IDE go to Sketch > Include Library > Add .zip library.
7. Now select the downloaded zip file and you are all set.

Processing IDE Setup:

1. Go to this link Download \ Processing.org and download the setup according to your system.
2. Extract the zip file wherever you want.
3. Open the extracted folder and run processing.exe.
4. Your processing IDE is ready to use.

Working >>

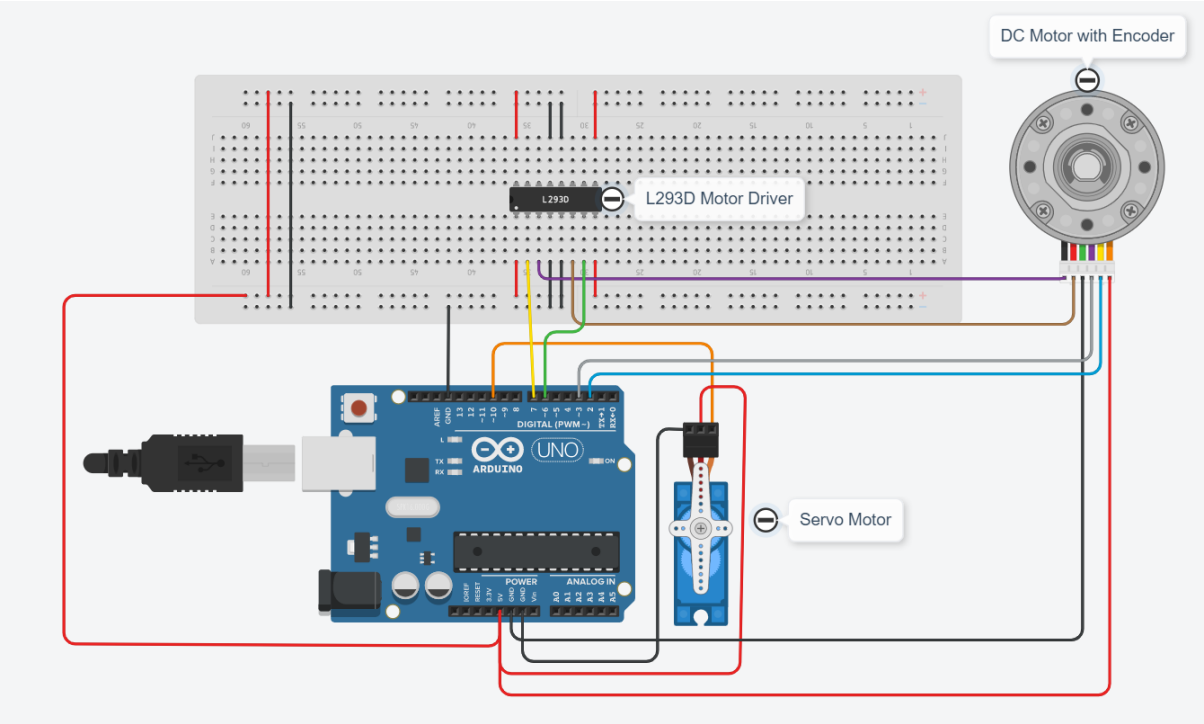
In the project we assume that there is gps transmitter over the flying object which is sending gps string in form of GPGGA format and gps module receiving that string and storing it in a file over computer.

In project we already have a file containing lots of GPGGA NEMA string. Program developed in processing language read that string and extract the information about latitude(), longitude() and altitude. This program also transmit the this information as a input via serial communication for Arduino program which calculate the distance between flying object (drone) and the antenna (having fixed co-ordinate) using 'Haversine Formula' , angle of rotation in plane using 'Bearing Angle Formula' and the third angle which is between the line joining drone and ground using 'Trigonometry'. These all values comprise as the input for motor controllers which point the antenna over flying object.

The two angles i.e., horizontal angle and vertical angle are sent to motors. Vertical angle is sent to a servo motor which rotates according to its reference state (0 deg). Horizontal angle is sent to a motor controller (L293D) which is connected to a dc motor with encoder and uses PID algorithm for precise rotation of motor.

For more information on dc motor visit-
https://youtu.be/dTGITLnYAY0
https://www.youtube.com/watch?v=K7FQSS_iAw0

Here is the Circuit diagram of our project-



GPGGA STRINGS

Structure ->

$GPGGA,[UTC],[Latitude],[direction],[Longitude],[Direction],[Quality],[No. of Satellites],[HDOP],[Altitude],[Unit],[geoidal separation],[age of correction],[station id],[check sum]

GP represents that it is a GPS position (Global Positioning), GGA is a type of NMEA message

UTC- Universal coordinated time in HHMMSS format

Latitude in the format of DDMM.MMMM

Direction of Latitude (N/S)

Longitude in the format of DDMM.MMMM

Direction of Longitude (W/E)

Indicates the quality of signal

- 1 = uncorrected coordinate
- 2 = differentially correct coordinate

4 = RTK fix coordinate (centimetre precision)

5 = RTK float (decimetre precision)

Number of satellites used to determine coordinates

Horizontal Dilution of Precision

Altitude Unit of altitude (Metre/ Feet)

Geoidal separation for more accurate height

Unit of geoidal separation (Metre/Feet)

Age of correction (0/1)

Correction station ID

Checksum (terminate the string)

Calculations >>

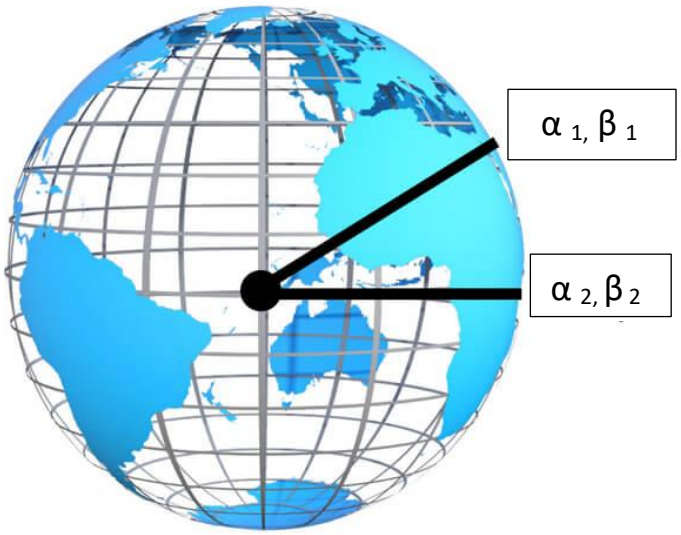
Haversine Formula:

Haversine(θ) = $Sin^2(\theta/2)$

(d/R) = Haversine (α_2-α_1) + Cos(α_1) Cos(α_2) Haversine ($\beta_1 - \beta_2$)

where R is the radius of earth (6371 km), d is the distance between two points ie flying object and antenna, α_1, α_2 are latitude of the two points and β_1 , β_2 are longitude of the two points respectively.

$d = 2R\,Sin^{-1}(sin^2((\alpha_2 - \alpha_1)/2) + Cos(\alpha_1)\,Cos(\alpha_2)\,Sin^2((\beta_2 - \beta_1)/2))^{1/2}$



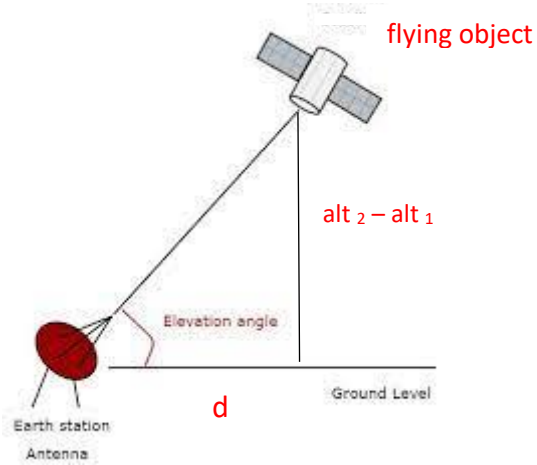
Third Angle Formula:

$\theta = \tan^{-1}((alt_2 - alt_1)/d)$

where, alt_1 is the height of the antenna (fixed)

alt_2 is the height of flying object

d is the distance between antenna and flying object (Haversine)



Bearing Angle Formula:

Bearing angle is a direction measured from north and it tracks angle in clockwise direction with north line.

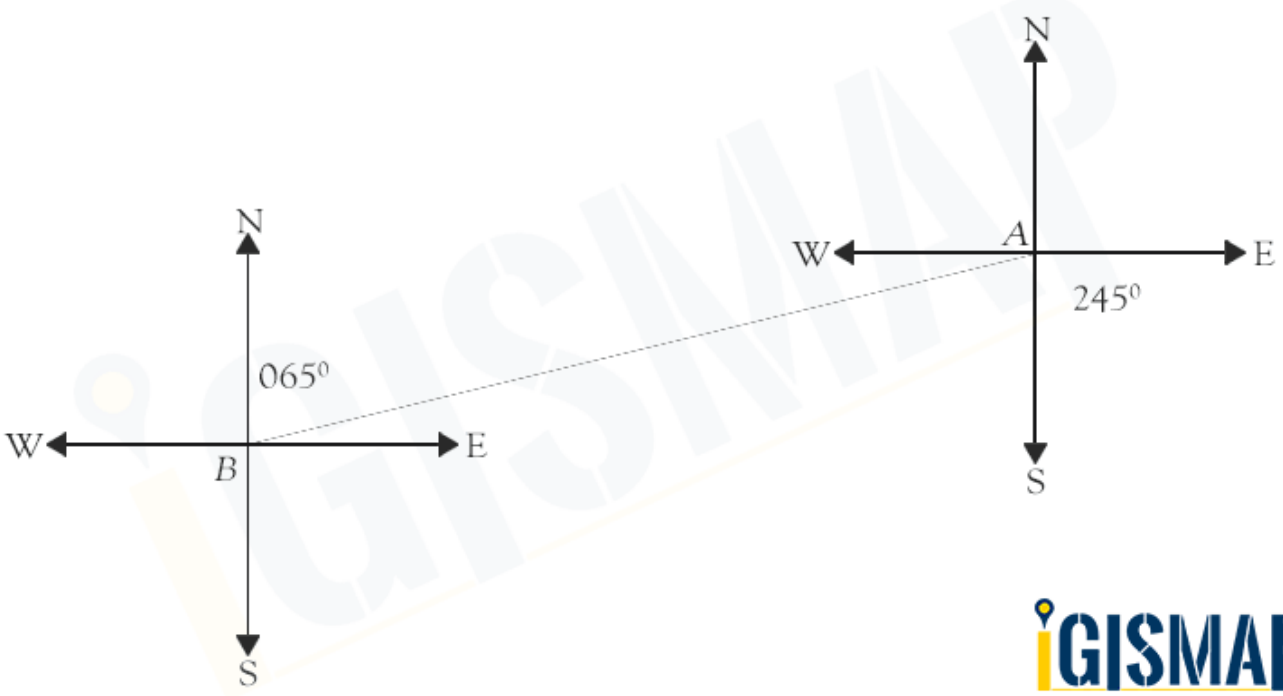
$\theta = \tan^{-1}2(X, Y)$

where, $X = \text{Cos}(\text{lat}_1) * \text{Sin}(\text{lat}_2) - [\text{Sin}(\text{lat}_1) * \text{Cos}(\text{lat}_2)] * \text{Cos}(\text{lon}_2\text{-lon}_1)$

$Y = \text{Sin}(\text{lon}_2\text{-lon}_1) * \text{Cos}(\text{lat}_2)$

lat_1, lat_2, lon_1 and lon_2 are in radians.

For example, bearing from A to B is 245°.



PID:

First let us tell you a control loop system. A control loop feedback system is a system that runs in a close loop, with a set-point to reach. The command given to the actuator to reach the set-point depends on the feedback. This feedback consists in the actuator's value measured by the sensor and compared to the set-point value. The resulting error is computed and re-injected into the initial order as a command that automatically corrects and adjusts the value of the actuator, in order to reach the set-point.

The term PID stands for **proportional**, **integral**, **derivative**. They are the three main components of the code. These three components will be calculated in function of the error given by the sensor. To simplify things, let's call these components functions: e.g. *F_i(e)* is the integral function in function of the error *e*. The original target is called set-point.

$$Command_{PID} = f_P(e) + f_I(e) + f_D(e)$$

Here are few links -
[An introduction to PID control with DC motor | by Simon BDY | luos | Medium]
https://youtu.be/t7ImNDOQIzM

ARDUINO CODE >>

Although below is a detailed explanation of the code running on Arduino, you can also find separate files for codes with comments in the GitHub.

```
#include <PID_v1.h>
#include <Servo.h>

#define MotEnable 5        //Motor Enamble pin Runs on PWM signal
#define MotFwd  6          // Motor Forward pin
#define MotRev  7          // Motor Reverse pin

long count = 1;                //a counter
float alt1 = 1.45;            //altitude 1 (antenna)
float alt2, longitude2, latitude2, finalAngle;
float oldAngle = 0.0;

const float LATITUDE1 = 39.099912;
const float LONGITUDE1 = -94.581213;

int servoPin = 10;
float servoPos = 0;

Servo myServo;

int User_Input = 0;     // This while convert input string into
integer
```

```
int encoderPin1 = 2;     //Encoder Output 'A' must connected with
intreput pin of arduino.
int encoderPin2 = 3;       //Encoder Otput 'B' must connected with
intreput pin of arduino.

volatile int lastEncoded = 0;       // Here updated value of encoder
store.
volatile long encoderValue = 0;     // Raw encoder value

int PPR = 1600;            // Encoder Pulse per revolution.
int angle = 360;           // Maximum degree of motion.
int REV = 0;              // Set point REQUIRED ENCODER VALUE
int lastMSB = 0;
int lastLSB = 0;

double kp = 5 , ki = 1 , kd = 0.01;             // modify for
optimal performance
double input = 0, output = 0, setpoint = 0;

PID myPID(&input, &output, &setpoint, kp, ki, kd, DIRECT);

void setup() {
  pinMode(MotEnable, OUTPUT);
  pinMode(MotFwd, OUTPUT);
  pinMode(MotRev, OUTPUT);
  Serial.begin(115200); //initialize serial comunication

  pinMode(encoderPin1, INPUT_PULLUP);
  pinMode(encoderPin2, INPUT_PULLUP);

  digitalWrite(encoderPin1, HIGH);            //turn pullup resistor
on
  digitalWrite(encoderPin2, HIGH);           //turn pullup resistor
on

  //call updateEncoder() when any high/low changed seen
  //on interrupt 0 (pin 2), or interrupt 1 (pin 3)
  attachInterrupt(0, updateEncoder, CHANGE);
  attachInterrupt(1, updateEncoder, CHANGE);

  myServo.attach(servoPin);

  TCCR1B = TCCR1B & 0b11111000 | 1;          // set 31KHz PWM to
prevent motor noise
  myPID.SetMode(AUTOMATIC);                 //set PID in Auto mode
  myPID.SetSampleTime(1);                   // refresh rate of PID
controller
  myPID.SetOutputLimits(-125, 125);         // this is the MAX PWM
value to move motor, here change in value reflect change in speed of
motor.
}

void loop() {
  while (Serial.available() == 0)         //program waits for serial
input using an infinite loop
  {
```

```
  }
  String coord1 = Serial.readString();    //puts the received string
in the latitude variable (drone latitude)
  latitude2 = coord1.toFloat();           //changes latitude from
string to float

  while (Serial.available() == 0)          //program waits for
serial input using an infinite loop
  {
  }
  String coord2 = Serial.readString();      //puts the received
string in longitude variable (drone longitude)
  longitude2 = coord2.toFloat();            //changes longitude from
string to float

  while (Serial.available() == 0)          //program waits for
serial input using an infinite loop
  {
  }
  String coord3 = Serial.readString();      //puts the received
string in the altitude variable (drone altitude)
  alt2 = coord3.toFloat();                  //changes altitude from
string to float

  float dAlt = alt2 - alt1;                          //differnece in
altitude
  Serial.print(count);                               //prints counter
  Serial.print(" Distance = ");

  float x = HaverSine(LATITUDE1, LONGITUDE1, latitude2,
longitude2);  //calculates distance by calling haversine function
  Serial.print(x, 3);                                //prints distance
with 3 decimal places

  count++;                                           //increases
counter
  Serial.print(", altitude = ");
  Serial.print(dAlt);                                //prints
difference in altitude

  float z = dAlt / x;                                //calculates
ratio of altitude and distance
  float angle = atan(z) * 180 / PI;                  //finds angle in
radians and is converted to degrees
  Serial.print(" and the angle = ");
  Serial.print(angle, 3);                            //prints angle
with 3 decimal places

  float angle2 = bearing(LATITUDE1, LONGITUDE1, latitude2,
longitude2);     //calculates bearing angle by calling bearing
function
  Serial.print(", and 2 angle = ");
  Serial.println(angle2, 6);                        //prints angle

  finalAngle = angle2 - oldAngle;
  oldAngle = angle2;
```

```
  servoPos = (int) angle;
  myServo.write(servoPos);

  User_Input = (int) finalAngle;

  REV = map (User_Input, 0, 360, 0, 1600); // mapping degree into
pulse
  //Serial.print("this is REV - ");
  //Serial.println(REV);               // printing REV value

  setpoint = REV;                     //PID while work to achive this
value consider as SET value
  input = encoderValue ;            // data from encoder consider as
a Process value
  //Serial.print("encoderValue - ");
  //Serial.println(encoderValue);
  myPID.Compute();                  // calculate new output
  pwmOut(output);

  delay(1000);
}

float HaverSine(float lat1, float lon1, float lat2, float lon2)
{
  float ToRad = PI / 180.0;
  float R = 6371;

  float dLat = (lat2 - lat1) * ToRad;
  float dLon = (lon2 - lon1) * ToRad;
  float a = sin(dLat/2) * sin(dLat/2) + cos(lat1 * ToRad) * cos(lat2
* ToRad) * sin(dLon/2) * sin(dLon/2);

  float c = 2 * atan2(sqrt(a), sqrt(1-a));

  float d = R * c;
  return d;
}

float bearing(float latit1,float longit1,float latit2,float longit2)
{
  float toRad = PI / 180.0;

  float latitud1 = latit1 * toRad;
  float latitud2 = latit2 * toRad;

  float dLat = (latit2 - latit1) * toRad;
  float dLon = (longit2 - longit1) * toRad;

  float y = sin(dLon) * cos(latitude2);
  float x = cos(latitud1)*sin(latitud2) -
sin(latitud1)*cos(latitud2)*cos(dLon);
  float brng = atan2(y,x);

  brng = brng / toRad;// radians to degrees
  return brng;
```

```
}

void pwmOut(int out) {
  if (out > 0) {                          // if REV > encoderValue
motor move in forward direction.
    analogWrite(MotEnable, out);          // Enabling motor enable
pin to reach the desire angle
    forward();                            // calling motor to move
forward
  }
  else {
    analogWrite(MotEnable, abs(out));          // if REV <
encoderValue motor move in forward direction.
    reverse();                             // calling motor to move
reverse
  }
}

void updateEncoder(){
  int MSB = digitalRead(encoderPin1); //MSB = most significant bit
  int LSB = digitalRead(encoderPin2); //LSB = least significant bit

  int encoded = (MSB << 1) |LSB; //converting the 2 pin value to
single number
  int sum  = (lastEncoded << 2) | encoded; //adding it to the
previous encoded value

  if(sum == 0b1101 || sum == 0b0100 || sum == 0b0010 || sum ==
0b1011) encoderValue ++;
  if(sum == 0b1110 || sum == 0b0111 || sum == 0b0001 || sum ==
0b1000) encoderValue --;

  lastEncoded = encoded; //store this value for next time

}

void forward () {
  digitalWrite(MotFwd, HIGH);
 digitalWrite(MotRev, LOW);

}

void reverse () {
  digitalWrite(MotFwd, LOW);
 digitalWrite(MotRev, HIGH);

}

void finish () {
  digitalWrite(MotFwd, LOW);
 digitalWrite(MotRev, LOW);

}
```

Processing Program Code >>

Although below is a detailed explanation of the code running in processing IDE, you can also find separate files for codes with comments in the GitHub

```
import processing.serial.*;                          //imports serial
library
Serial port = new Serial (this, "COM3", 115200);
//opens COM port for communication with name port

void setup () {
  size (300, 300);
}

void draw () {
  String[] lines = loadStrings("gps.txt");
//open file named "gps.txt" and reads its lines and stores as an
array

  float la, lo, lat, lon;
  String lad, lod, alt;

  for (int i = 0; i < lines.length; i++)    //loop for going through
lines
  {
    println(lines[i]);                        //prints line i
    String[] list = split(lines[i], ',');
//splits line i with commas and put it in a array

    la = float(list[2]);         //extracts latitude and changes it
to float
    lad = list[3];               //extracts latitude direction
    lo = float(list[4]);        //extracts longitude and changes it
to float
    lod = list[5];               //extracts longitude direction
    alt = list[9];                 //extracts altitude as a string
    lat = converter(la);
//converts latitude to degrees by calling converter function
    lon = converter(lo);
//converts longitude to degrees by calling converter function

    if (lad.equals("S") == true)
//checks if latitude direction is equal to S
    {
      lat *= -1;                                    //if true
multiplies latitude by -1
    }

    if (lod.equals("W") == true)                    //checks if
longitude direction is equal to W
```

```
    {
      lon *= -1;                                        //if true
multiplies longitude by -1
    }

    println(lat + ", " + lon + ", " + alt);
//prints latitude, longitude, altitude

    String latit = String.valueOf(lat);   //changes latitude to
string
    String longit = String.valueOf(lon);  //changes longitude to
string

    port.write(latit);                  //Serially writes latitude
    port.write(longit);                   //Serially writes longitude
    port.write(alt);                      //Serially writes altitude

  }
  println(" ");
}

float converter(float l)                                //converts
DDMM.MMMM coordinates to degrees
{
    float a = l % 100;
    float b = a / 60;
    float c = (l - a) / 100;
    float d = c + b;
    return d;
}
```

Helpful links

- ✓ https://learn.sparkfun.com/tutorials/what-is-an-arduino
- ✓ https://learn.sparkfun.com/tutorials/installing-arduino-ide
- ✓ https://learn.sparkfun.com/tutorials/data-types-in-arduino
- ✓ https://learn.sparkfun.com/tutorials/serial-communication
- ✓ https://learn.sparkfun.com/tutorials/pulse-width-modulation
- ✓ YouTube links>>
- ✓ https://youtube.com/playlist?list=PLA567CE235D39FA84
- ✓ https://youtube.com/playlist?list=PLGs0VKk2DiYw-L-RibttcvK-WBZm8WLEP

Challenges >>

The challenges we faced are:

- ✓ How to take input from the file having GPGGA string as a input for our Arduino ide program for calculating distance and bearing angle, as Arduino is not able to read the string from the file over computer and extract the latitude, longitude and altitude of flying object (drone).
- ✓ Since the whole project is done in online mode, there was lack of hardware and actual sensible data. Though we try to do the maximum work using simulators but there were many times when we just can't do anything.
- ✓ Using PID control algorithm with dc motor was also one of the challenge that we faced.

OUR MENTORS

Baldeep sir (2 yr)

Vishesh sir (2 yr)

OUR TEAM

Shrey Agrawal (1 yr)

Lakshit Sharma (1 yr)

Pranjal Gautam (1 yr)

Kashish Agarwal (1 yr)

Aditya Sen (1 yr)

Special Thanks to

Priyanshu Somvanshi sir (3 yr)