Due:05/3/2023

Rover Mission #4 – Advanced GPS Navigation

a. Navigate to specified GPS waypoint that will be provided to you. You may hardcode the GPS waypoint coordinates for this mission. The rover should stop within 15 feet of the marked GPS waypoint.
b. (Extra Credit) The rover should read the GPS coordinates from an RF transmitter beacon placed at the starting location, place a physical marker within 15 feet of the specified waypoint, and return to original start location. The distance between the physical marker and the marked waypoint will be measured along with the distance between start and end positions of the rover. The cumulative error should not exceed 25 feet.
    - The GPS coordinates of the waypoint will be transmitted as a 12-character message. The latitude coordinate will be 5 digits preceded by an 'N', the longitude will be 5 digits preceded by a 'W'. The 5 digits of the latitude represent the decimals that would be appended to 39; the five digits following W represent the decimals that would be appended to -108 to form the longitude. For example, the message for the rover to travel to 39.09495N by -108.58784W would be "N09495W58784"
    - Coordinates will be continuously transmitted by the RF beacon at 5 second intervals.

Results

PASSED

Rover Version

Rover V.4 was used to attempt this mission which included a DC motor, Arduino uno, L298P shield R3 motor driver module, Futaba S3003 standard servo, 9.6V 2000mAH NiMH battery pack, MG90S micro servo,433 MHz RF receiver and a Adafruit Ultimate GPS breakout module.

Arduino Code

Mission 4A

```
//This sketch will allow the rover, from a random position and heading, to eventually reach a waypoint set in the sketch within 3 meters
#include <TinyGPS++.h> // loads Tiny gps libary
TinyGPSPlus gps;    //identifies the gps
#include <Servo.h>  //include Servo library
#include <gSoftSerial.h> //include gsoftserial library as the traditional serialsoftware creates timing issues with the servo library
//https://github.com/night-ghost/gsm-modem/tree/master/libraries/gSoftSerial

//variables for the steering servo
Servo STEER;                  //create Steer servo
#define SERVO_PIN_STEER A3    //attach steer servo to pin A3
#define FRONT 92            //Initial position for Steer servo
int RIGHT=FRONT+40;
int LEFT=FRONT-40;
```

```
//DC motor settings
int POWER=120;                //Power rating
int directionPin = 12;        //Diriction pin on 12
int pwmPin = 3;               //power pin on 3
int brakePin = 9;             //brake pin on 9

//payload servo settings
Servo DUMP;                   //create Dump servo for payload
#define SERVO_PIN_DUMP A2     //attach Dump servo to A2
int Payload=140;              //initial position for dump servo
int Dump=Payload-125;         //position for servo to drop the payload

//GPS settings
#define GPSBaud 9600 //sets baud rate for the gps module to arduino
#define RXPin 4    //the recieve pin on the arduino will go to the 'TX' pin on the GPS module
#define TXPin 7   //the transmistion pin(Pin3) on the arduino 'RX' pin on the GPS module
gSoftSerial gpsSerial(RXPin, TXPin); //sets pinsets for the serial port for the GPS module

//constants for calculating
float tConstant = 13;        // angle times constant = turn time in ms for each degree heading off
float distanceToGo;          //distance to waypoint from current position
float driveconstant = 260;   //constant to drive forward in ms. Left low to allow multiple turns to account for drift
float longitude;             //sets longitude as a float
float latitude;             // sets latitude as a float
float latitude2;             // sets latitude 2 as a float
float longitude2;            // sets longitude 2 as a float
float latFinal;             //sets waypoint latitude as a float
float longFinal;             //sets waypoint longitude as a float



void setup() {
Serial.begin (9600); //baud rate for serial monitor
gpsSerial.begin(GPSBaud); //baud rate for GPS module

//pin output layouts for DC motor
pinMode(directionPin, OUTPUT);
pinMode(pwmPin, OUTPUT);
pinMode(brakePin, OUTPUT);

//Attach servos
STEER.attach(SERVO_PIN_STEER);
DUMP.attach(SERVO_PIN_DUMP);
//Servo start locations
STEER.write(FRONT);
DUMP.write(Payload);
}

//Payload dropping instructions
void dump_payload()
{
  delay(500);
  DUMP.write(Dump);          //turns servo for payload to drop it
  delay(2000);
  DUMP.write(Payload);             //return payload mechanism to origional spot
}

//Braking instructions
void Brake()
{
  analogWrite(pwmPin, 0);
  digitalWrite(brakePin, HIGH);
  delay(2000);
}

//drive forward instructions
void drive_forward(int steps, int POWER)
{
  digitalWrite(directionPin, HIGH);
  digitalWrite(brakePin, LOW);
  analogWrite(pwmPin, POWER);
  delay(steps);
}
```

```
//reverse instructions
void Reverse(int steps, int POWER)
{
  digitalWrite(directionPin, LOW); //DC motor HIGH->forward
  digitalWrite(brakePin, LOW);      //release breaks
  analogWrite(pwmPin, POWER);       //turn on DC motor
  delay(steps);
  Brake();
  delay(2000);
}

//drive left instructions
void drive_left(float tangle, int speed)  //takes tangle(bearing difference) and speed inputs in void loop
{
  float turnTime =tangle*tConstant;  // turn time equals the bearind difference multiplyed by the time constant
  STEER.write(LEFT);                 //turn steering servo to left position0
  delay(1000);
   digitalWrite(directionPin, HIGH); //drive forward while servo is turned
  digitalWrite(brakePin, LOW);
  analogWrite(pwmPin, POWER);
  delay(turnTime);                  //have DC motor stay on for turn time
  Brake();                          //stop DC motor
  STEER.write(FRONT);               //reset steering servo to straight position
  delay(500);
}
//drive right instructions
void drive_right(float tangle, int speed)  //takes tangle(bearing difference) and speed inputs in void loop
{
  float turnTime =tangle*tConstant;         // turn time equals the bearind difference multiplyed by the time constant
   STEER.write(RIGHT);                      //turn steering servo to left position
  delay(1000);
   digitalWrite(directionPin, HIGH);        //drive forward while servo is turned
  digitalWrite(brakePin, LOW);
  analogWrite(pwmPin, speed);
  delay(turnTime);                        //have DC motor stay on for turn time
  Brake();//stop DC motor
  STEER.write(FRONT);                     //reset steering servo to straight position
  delay(500);
}

 // instructions to display GPS information
void displayInfo(){
  latitude = gps.location.lat() ;      // floats line for GPS latitude
  longitude = gps.location.lng();       // floats line for Gps longitude
 if (gps.location.isValid())               // checks if data is still valid.
  {
   Serial.print("latitude");                  // prints "latitude" in serial monitor
   Serial.println(latitude,6);                 // prints latitude data from gps
   Serial.print("longitude");              // prints "longitude" in serial moniter
   Serial.println(longitude,6);                // prints "longitude data from gps
  latitude = gps.location.lat() ;      // floats line for GPS latitude
  longitude = gps.location.lng();       // floats line for Gps longitude
  }
  else
  {
    Serial.println("no connection");
  }
  }

// instructions to update the GPS
void updateGPSObject()
{
  // This sketch displays information every time a new sentence is correctly encoded.
  while (gpsSerial.available() > 0)   // while there is data coming in from serial port.
    if (gps.encode(gpsSerial.read()))  // read data coming in from sereall port .
      {
        displayInfo();
       }
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 void loop() {
////////////////////////////////////////////////////////////////////////////////
  latFinal = 39.080430;                       //waypoint lat and long
longFinal = -108.560791;

    // run through enough times to make sure that you get a full sentence
    for(double i=0; i<40000; i++) {
      updateGPSObject();
    }
  delay(10);

  Serial.println("starting to drive to establish heading");         // Debuging serail print
 drive_forward(2000,POWER);   // drives forward for 4 seconds at motor speed.
 Brake();
 delay(4000);                          // delay car for 4 seconds
 Serial.println("finished drive to establish heading");      // Debuging serial print


 latitude2 = latitude;              // sets starting latitude to latitude 2
 longitude2 = longitude;            // sets starting longitude to longitude 2

    // run through enough times to make sure that you get a full sentence
    for(double i=0; i<40000; i++) {
      updateGPSObject();
    }

  delay(10);

 double bearing = gps.courseTo(latitude2, longitude2, latitude, longitude); // calculates bearing using lat/long from previous lat/long
 double wpHeading = gps.courseTo(latitude, longitude, latFinal, longFinal);  // calculates bearing to waypoint from current lat/long
distanceToGo = gps.distanceBetween(latitude, longitude, latFinal, longFinal);  // calculates distance to waypoint from current lat/long
 Serial.print("my heading is: ");
 Serial.println(bearing);
 Serial.print("distance to go: ");
 Serial.println(distanceToGo);

if (distanceToGo <= 3)                                  //if distance to final position is less than 3 meters
  {
   Brake();                                              //ensure the rover is stopped
   Serial.println("I think I'm at the waypoint");         // debugging serial print
  dump_payload();                                         //drop the payload
  while(1);                                               //stop void loop
  }

////Turning command//////////////////////////////////////////////
if (bearing > wpHeading)                               // if statement for bearing greater then waypoint heading
  {
    if ((bearing - wpHeading) < 180)                    // if bearing-heading is less then 180 turn left
    {
      Serial.print("Turn left by: ");
      Serial.println(bearing- wpHeading);
      Serial.print("Drive a distance of: ");
      Serial.println(distanceToGo);
      drive_left((bearing - wpHeading),POWER);
      drive_forward(distanceToGo*driveconstant,POWER);
      Brake();

    }
      else                                                 // if bearing-heading is greater then 180 turn right
      {
        Serial.print("Turn right by: ");
      Serial.println(bearing- wpHeading);
      Serial.print("Drive a distance of: ");
      Serial.println(distanceToGo);
        drive_right((360-(bearing-wpHeading)),POWER);
         drive_forward(distanceToGo*driveconstant,POWER);
        Brake();
      }
  }
```

```
else                                                    //if waypoint heading is greater than bearing
  {
   if((wpHeading - bearing) < 180)                     // if heading-bearing is less then 180 turn right
   {
     Serial.print("Turn right by: ");
     Serial.println(bearing- wpHeading);
     Serial.print("Drive a distance of: ");
     Serial.println(distanceToGo);
     drive_right((wpHeading - bearing),POWER);
      drive_forward(distanceToGo*driveconstant,POWER);
     Brake();
   }
     else                                                    // if heading-bearing is greater then 180 turn left
     {
       Serial.print("Turn left by: ");
     Serial.println(bearing- wpHeading);
     Serial.print("Drive a distance of: ");
     Serial.println(distanceToGo);
       drive_left(360-(wpHeading - bearing),POWER);
        drive_forward(distanceToGo*driveconstant,POWER);
        Brake();
     }
  }

 //after completing a turn reset lat and long
    latitude2 = latitude;              // sets starting latitude to latitude 2
    longitude2 = longitude;            // sets starting longitude to longitude 2

 delay(10);
}
```

Mission 4B (Extra Credit)

N/A

*All mission are now complete, no more following modifications will be evaluated.