Due: 04/19/2023

Rover Mission #3 – Basic GPS Navigation

a. Point due North. From an arbitrary start location and heading, navigate so that the rover is pointing due North. Point north within +/- 20° and complete the mission within 180 seconds.

*Requires two consecutive successful runs

Results

PASSED

Rover Version

Rover V.3 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 3A

```
//This sketch will allow the rover, from a random heading,
to eventually point north and drop the payload when complete

#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 85             //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;
```

```
//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

float tConstant = 20; // angle times constant = turn time in ms 16.25
float forwarddistance;    // sets forward distance as a float
float distanceArr;            // sets distance arr as a float
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

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 POWER)
{
  float turnTime =tangle*tConstant;  // turn time equals the angle multiplyed by the time constant
  STEER.write(LEFT);
  delay(1000);
   digitalWrite(directionPin, HIGH);
  digitalWrite(brakePin, LOW);
  analogWrite(pwmPin, POWER);
  delay(turnTime);
  Brake();
  STEER.write(FRONT);
  delay(500);
}
//drive right instructions
void drive_right(float tangle, int POWER)
{
  float turnTime =tangle*tConstant;
   STEER.write(RIGHT);
  delay(1000);
   digitalWrite(directionPin, HIGH);
  digitalWrite(brakePin, LOW);
  analogWrite(pwmPin, POWER);
  delay(turnTime);
  Brake();
  STEER.write(FRONT);
  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() {

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

 delay(10);                              // Delays program for 10 milseconds needed for updateGPSObject function or program wont run
 Serial.println("update GPS");          // Debuging serail print
drive_forward(4500,POWER);    //go forward 4500ms at regular power(120)
Brake();
delay(10000);                           // delay car for 10 seconds to settle out GPS
Serial.println("function test 3");      // 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);
Serial.println("2nd update GPS ");    // debugging serial print
 double courseTo = gps.courseTo(latitude2, longitude2, latitude, longitude); // calculates bering using starting lat and long also using the second lat and long

   Serial.print("Current heading: ");   // serial prints for debugging
   Serial.println(courseTo);            // serial prints bearing for debugging

   if (courseTo >= 20 && courseTo < 180)      // if the bearing othe the rover is greater then 20 degrees north and less then 180 degrees south
     {
      drive_left(courseTo,POWER);           // turns rover left the amount of the angle needed for north, at 250 motor speed and time
      Serial.println("turn left");           // debugging serial print
      Serial.println(courseTo);             // debugging serial print
     Reverse(3000, POWER);                  //reverse after turning to account for limited space
     }
   else if (courseTo >= 180 && courseTo <= 340)   // if the berring is greater then or equal 180 degrees south to and berring is less then and equal to 340 degrees north
     {
       drive_right(360-courseTo,POWER);       // turns rover right the amount of angle needed to point north.
       Serial.println("turn right");          // debuging serial print
       Serial.println(360-courseTo);         // debuging serial print
       Reverse(3000, POWER);                  //reverse after turning to account for limited space
     }
    else                                        // else if the rover is faceing north between 20 degrees and 360 degrees
      {
       Serial.println("brake");                // debugging serial print
       Serial.println(courseTo);               // debugging serial print
        Brake();
       delay(2000);                             // wait for 2 seconds
         Serial.println("I think I'm pointing North");     // debugging serial print
        dump_payload();                                 //drop payload
       while(1);                                // stops program from running again
      }


  Serial.println("I turned and I'm going to try again");      // debugging serial print
  delay(3000);                                               // delays for 3 seconds

}
```

Following Modifications

Moving forward to Mission #4 we are considering adding in an Arduino Mega or Arduino slave since the sensor, Arduino and RF receiver libraries are acting weird for Mission 4B.