Due: 03/29/2023

Rover Mission #2 – Obstacle Avoidance

a. Drive in an enclosed area (at least 10’x10’) without contacting walls or obstacles. The mission will last 60 seconds.
b. Drive forward to waypoint 30’ directly ahead of robot. Demonstrate avoidance algorithm by continuing to original waypoint after navigating around a single obstacle. The rover should stop within 5ft of the marked waypoint and complete the mission in under 120 seconds.

*Without GPS Navigation

*Requires two consecutive successful runs

Results

PASSED

Rover Version

Rover V.2 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 HC-SRO4 ultrasonic sensor.

Arduino Code

Mission 2A

```
#include <Servo.h>  //include Servo library

//create variables for servos
Servo STEER;                  //create Steer servo

#define SERVO_PIN_STEER A3    //attach steer servo to pin A3
#define FRONT 100             //Initial position for Steer servo
int RIGHT=FRONT+20;
int HARD_RIGHT=FRONT+40;
int LEFT=FRONT-20;
int HARD_LEFT=FRONT-40;

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

Servo servoLook;
#define SERVO_PIN_SENSOR_FRONT 6
#define SENSOR_FRONT 90
int SENSOR_LEFT=SENSOR_FRONT+60;
int SENSOR_RIGHT=SENSOR_FRONT-60;
```

```
//DC motor settings
int POWER=100;                //Power rating
int directionPin = 12;        //Diriction pin on 12
int pwmPin = 3;               //power pin on 3
int brakePin = 9;             //brake pin on 9

//Range Sensor
#define echo    A0 // Ultrasonic Echo pin connect to D11
#define trig    A1  // Ultrasonic Trig pin connect to D12

byte maxDist = 200;                                 //Maximum sensing distance
byte stopDist = 80;                                 //Minimum distance from an object to stop in cm
float timeOut = 2*(maxDist+10)/100/340*1000000;   //Maximum time to wait for a return signal

void setup() {

  Serial.begin (9600); //baud rate

  //pin layouts for Range Sensor
  pinMode(trig, OUTPUT); //pin set for ping send
  pinMode(echo, INPUT); //pin set for echo recieve

//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);
servoLook.attach(SERVO_PIN_SENSOR_FRONT);

Brake();
STEER.write(FRONT);
delay(1000);

}

void Reverse()
{
  digitalWrite(directionPin, LOW); //DC motor HIGH->forward
  digitalWrite(brakePin, LOW);      //release breaks
  analogWrite(pwmPin, POWER);       //turn on DC motor
}

void Forward()
{
  digitalWrite(directionPin, HIGH);
  digitalWrite(brakePin, LOW);
  analogWrite(pwmPin, POWER);
}

void Brake()
{
  analogWrite(pwmPin, 0);
  digitalWrite(brakePin, HIGH);
}
```

```
void turnRight(int duration)
{
  STEER.write(RIGHT);
  delay(duration);
}

void turnLeft(int duration)
{
  STEER.write(LEFT);
  delay(duration);
}

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
}

int getDistance()                                   //Measure the distance to an object
{
unsigned long pulseTime;                           //Create a variable to store the pulse travel time
int distance;                                      //Create a variable to store the calculated distance
digitalWrite(trig, HIGH);                          //Generate a 10 microsecond pulse
delayMicroseconds(10);
digitalWrite(trig, LOW);
pulseTime = pulseIn(echo, HIGH, timeOut);          //Measure the time for the pulse to return
distance = (float)pulseTime * 340 / 2 / 10000;     //Calculate the object distance based on the pulse time
return distance;
}
int checkDirection()                              //Check the left and right directions and decide which way to turn
{
int distances [2] = {0,0};                                  //Left and right distances
int turnDir = 1;                                               //Direction to turn, 0 left, 1 reverse, 2 right
servoLook.write(180);                                          //Turn servo to look left
delay(500);
distances [0] = getDistance();                                 //Get the left object distance
servoLook.write(0);                                            //Turn servo to look right
delay(1000);
distances [1] = getDistance();                                 //Get the right object distance
if (distances[0]>=200 && distances[1]>=200)                    //If both directions are clear, turn left
turnDir = 0;
else if (distances[0]<=stopDist && distances[1]<=stopDist)     //If both directions are blocked, turn around
turnDir = 1;
else if (distances[0]>=distances[1])                           //If left has more space, turn left
turnDir = 0;
else if (distances[0]<distances[1])                            //If right has more space, turn right
turnDir = 2;
return turnDir;
}

void loop() {
{
servoLook.write(90);                             //Set the servo to look straight ahead
delay(750);
int distance = getDistance();                    //Check that there are no objects ahead
if(distance >= stopDist)                         //If there are no objects within the stopping distance, move forward
{
Forward();
}
while(distance >= stopDist)                  //Keep checking the object distance until it is within the minimum stopping distance
{
distance = getDistance();
delay(250);
}
```

```
Brake();                                    //Stop the motors
int turnDir = checkDirection();               //Check the left and right object distances and get the turning instruction
Serial.print(turnDir);
switch (turnDir)                              //Turn left, turn around or turn right depending on the instruction
{
case 0:                                      //Turn left
turnLeft (400);
break;
case 1:                                      //Turn around
turnLeft (700);
break;
case 2:                                      //Turn right
turnRight (400);
break;
}
}
}
```

Mission 2B

```
//This sketch allows the rover to go forward to a point 30ft in front of it
//and to avoid a 6foot length obstacle that is placed perpendicular in front of it

#include <Servo.h>  //include Servo library
#include <NewPing.h> //for the Ultrasonic sensor function library.
//https://github.com/eliteio/Arduino_New_Ping

//create variables for servos
Servo STEER;                    //create Steer servo

#define SERVO_PIN_STEER A3    //attach steer servo to pin A3
int FRONT=80;                   //Initial position for Steer servo
int RIGHT=FRONT+30;
int LEFT=FRONT-30;

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;

Servo servoLook;
#define SERVO_PIN_SENSOR_FRONT 6

//DC motor settings
int POWER=100;                  //Power rating
int directionPin = 12;          //Diriction pin on 12
int pwmPin = 3;                 //power pin on 3
int brakePin = 9;               //brake pin on 9

//Range Sensor
#define echo_pin    A0 // Ultrasonic Echo pin connect to D11
#define trig_pin    A1  // Ultrasonic Trig pin connect to D12
#define maximum_distance 200
NewPing sonar(trig_pin, echo_pin, maximum_distance); //sensor function
int distance = 100;
```

```
long timeto;

void setup() {

  Serial.begin (9600); //baud rate

//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);
servoLook.attach(SERVO_PIN_SENSOR_FRONT);

//initial range distance gathering
    delay(2000);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);

//servo initial points
servoLook.write(90);
STEER.write(FRONT);
DUMP.write(Payload);

}

void Reverse()
{
  digitalWrite(directionPin, LOW); //DC motor HIGH->forward
  digitalWrite(brakePin, LOW);      //release breaks
  analogWrite(pwmPin, 100);       //turn on DC motor
}

void Forward()
{
  digitalWrite(directionPin, HIGH);
  digitalWrite(brakePin, LOW);
  analogWrite(pwmPin, POWER);
}

void Brake()
{
  analogWrite(pwmPin, 0);
  digitalWrite(brakePin, HIGH);
```

```
void turnRight()
{
  STEER.write(RIGHT);
  delay(1000);
}
void turnStraight()
{
  STEER.write(FRONT);
  delay(1000);
}

void turnLeft()
{
  STEER.write(LEFT);
  delay(1000);
}
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
}

void loop() {
//for debugging
  Serial.print("Distance: ");//for debugging range finder
  Serial.println(distance);

  //if obstacle is less than 90cm run avoidance protocol
  if (distance <= 90){

    Brake();
    delay(500);
    Reverse();
    delay(2000);
    Brake();
    delay(1500);
    Forward();
   STEER.write(40);
   delay(1600);
   STEER.write(90);
   delay(1100);
   STEER.write(160);
   delay(2800);
   STEER.write(90);
   delay(500);
   STEER.write(50);
   delay(650);
   STEER.write(90);
   delay(500);
   Brake();
```

```
   delay(500);
   Reverse();
   delay(1000);
   Forward();
   //uses later timeto variable for remaining forward drive distance after obstacle
   delay(7000-timeto);
   Brake();
   dump_payload();
   while(1);

  }
  //drive forward
  else{
    STEER.write(90);
   Forward();
   //variable for how long the rover went before obstacle
  timeto=(millis()-3000);
  }
    distance = readPing();
}


//command to get distance for rangefinder
int readPing(){
  delay(60);
  int cm = sonar.ping_cm();
  if (cm<=852){
    cm=250;
  }
  return cm;
}
```

Following Modifications

Moving forward to Mission #3, we plan to add in a GPS module to help navigate itself using GPS coordinates which is our overall most important customer requirement.