// Include the Servo library
#include <Servo.h>

// Declare the Servo pin
int servoPin = 11;

// Create a servo object
Servo Servo1;

int mot1a=5 , mot1b=2 , mot1c=6 , mot2a=3 , mot2b=4 , mot2c=10 ;
#define outputA 7 // pins for encoder
#define outputB 8 // pins for encoder

int conter = 0; // value given by the encoder
int aState; // initial state for encoder
int aLastState; // measured (final) state for encoder
const int trigPin = 9; // for ultrasonic emitter
const int echoPin = 12; // for ultrasonic sensor

long duration;

int distance; // distance provided by the ultrasonic reading (obstacle distance)
int td=0; // total distance to be planted (input from bluetooth)
int od=-1; // offset distance to be planted (input from bluetooth)
int tx=1; // connected to RXD bluetooth pin
int rx=0; // connected to TVD bluetooth pin
int l=1; // for initial conditions
int b=1; // for initial conditions
int z=0;

int md=0; // measured distance
int i; // counter for loop

void setup() {
  // We need to attach the servo to the used pin number
Servo1.attach(servoPin);
pinMode (outputA,INPUT);//output A of the encoder
pinMode (outputB,INPUT);//output B of the encoder
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
pinMode(mot1a,OUTPUT);//sets the pin (mot1a) as an output pin
pinMode(mot1b,OUTPUT);//sets the pin (mot1b) as an output pin
pinMode(mot1c,OUTPUT);//sets the pin (mot1c) as an output pin
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(6,OUTPUT);
pinMode(tx, OUTPUT); //defines the mode of work for tx pin
pinMode(rx, INPUT);//defines the mode of work for rx pin
Serial.begin(9600);

aLastState = digitalRead(outputA);//initialize the value of the variable "alaststate" to the value of the reading of the encoder at A
digitalWrite(mot1a,HIGH);
digitalWrite(mot1b,HIGH);
}

void loop() {
digitalWrite(trigPin, LOW);//turn off the trigpin
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);// Sets the trigPin on HIGH state for 10 micro seconds
delayMicroseconds(10);
digitalWrite(trigPin, LOW);// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);// Calculating the distance
distance= duration*0.034/2;
Serial.print("Distance: ");
Serial.println(distance); // Prints the distance on the Serial Monitor
if (Serial.available() > 0 && l==1) // condition that a value is received from bluetooth
{ td= Serial.read(); // assign the value received by bluetooth to td
l=0;
//td = td*10;
Serial.print(td);
}

if (Serial.available() > 0 && Serial.available()!=td && b==1) // condition another value is received
from bluetooth
{ od= Serial.read(); // assign the value received from bluetooth to od
//od=od*10;
z=od;
Serial.print(od);
b=0;
} // end of bluetooth
if ( td !=0 && od ==0 ) // case of continous seeding
{if (distance >10) // no obstacle
(if (md <td) // moved distance < total distance given by the user
{digitalWrite(mot1a,HIGH);
digitalWrite(mot1b,LOW); // turns motor on
analogWrite(mot1c,160); // control voltage supplied to the motor, thus controlling motor speed
for (i=1;i<300;i++)
{
Servo1.write(180); // rotate the servo 180 degrees
aState = digitalRead(outputA); // Reads the "current" state of the outputA
// If the previous and the current state of the outputA are different, that means a Pulse has occurred

if (aState != aLastState) {
    // If the outputB state is different to the outputA state, that means the encoder is rotating clockwise
    if (digitalRead(outputB) != aState) {
        conter ++;
    }
    Serial.print("Position: ");
    Serial.println(conter);
}
aLastState = aState; // Updates the previous state of the outputA with the current state

for (i=1;i<300;i++) {
    Servo1.write(0); // rotate the servo back
    aState = digitalRead(outputA); // Reads the "current" state of the outputA
    // If the previous and the current state of the outputA are different, that means a Pulse has occurred
    if (aState != aLastState) {
        // If the outputB state is different to the outputA state, that means the encoder is rotating clockwise
        if (digitalRead(outputB) != aState) {
            conter ++;
        }
        Serial.print("Position: ");
        Serial.println(conter);
    }
    aLastState = aState; // Updates the previous state of the outputA with the current state
md=conter/2 ; // measured distance
}
else // moved distance = total distance
{
digitalWrite(mot1a,HIGH);
digitalWrite(mot1b,HIGH);//blocks the movement of the motor
analogWrite(mot1c,0);
}
}
else // presence of obstacle
{
digitalWrite(mot1a,HIGH);
digitalWrite(mot1b,HIGH);
analogWrite(mot1c,0);
}
}

if ( td !=0 && od !=0 )//case of offset distance seeding
{
if ( (td-od)>=0 ) // normal case
{
if (distance >10) // no obstacle
{
if (md <od)
{
digitalWrite(mot1a,HIGH);
digitalWrite(mot1b,LOW);
analogWrite(mot1c,250);
md=conter/2 ;
}
if (md >=od && b!=1) // reached the offset distance (time for seeding)
{
// initialize moved distance
digitalWrite(mot1a,HIGH);
digitalWrite(mot1b,HIGH);
analogWrite(mot1c,0);
Servo1.write(180); // rotate the servo 180 degrees
delay(1000);
Servo1.write(0); // rotate the servo back
delay(1000); // dispense
od = od+z; // total distance left to be planted
}
}
else // presence of obstacle
{
digitalWrite(mot1a,HIGH);
digitalWrite(mot1b,HIGH);
analogWrite(mot1c,0); // stop the motor
}
}
else // total distance to be planted < offset distance (no need to move robot)
{
digitalWrite(mot1a,HIGH);
digitalWrite(mot1b,HIGH);
analogWrite(mot1c,0); // stop
}

if (td<=od)
{
digitalWrite(mot1a,HIGH);
digitalWrite(mot1b,HIGH);
analogWrite(mot1c,0);
l=1;
b=1; //to allow the user to input values again using blutooth
aState = digitalRead(outputA); // Reads the "current" state of the outputA

    // If the previous and the current state of the outputA are different, that means a Pulse has occurred
    if (aState != aLastState){
        // If the outputB state is different to the outputA state, that means the encoder is rotating clockwise
        if (digitalRead(outputB) != aState) {
            counter ++;
        }
    }
    Serial.print("Position: ");
    Serial.println(counter);
}
aLastState = aState; // Updates the previous state of the outputA with the current state