```
#include <util/atomic.h> // For the ATOMIC_BLOCK macro


#define encoder11 0  // This pin takes input from encoder 1 of
 motor 1

#define encoder12 1  // This pin takes input from encoder 2 of
 motor 1

#define encoder 21 2 // This pin takes input from encoder 1 of
 motor 2

#define encoder 22 3 // This pin takes input from encoder 2 of
 motor 2


#define pwm1 5 // These pins control the pulse width modulated
 signals for both the motors

#define pwm2 6 // These pins are connected to the motor driver
 L293D


#define output11 3 // Controls the output 1 of motor 1

#define output12 4 // These pins are connected to the motor
 driver L293D which gives output to the motor as per the pwm

#define output21 7 // signal obtained from pwm pins

#define output22 8


#define trigger 9 // These pins are for the ultrasonic
 distance sensor

#define echo 10
```

```
#define camera 11


int target = 720; // Two wheel rotations needed for steering

int prevT = 0;

int pos = 0;


void setup()

{

    Serial.begin(9600);

    pinMode(encoder11, INPUT);

    pinMode(encoder12, INPUT);

    pinMode(encoder21, INPUT);

    pinMode(encoder22, INPUT);


    // Attaching interrupts to the pins to ensure measuring
 the position of the motor shaft

    attachInterrupt(digitalPinToInterrupt(encoder11),
 readEncoder1, RISING);

    attachInterrupt(digitalPinToInterrupt(encoder21),
 readEncoder2, RISING);


    pinMode(pwm1, OUTPUT);

    pinMode(pwm2, OUTPUT);
```

```
  pinMode(output11, OUTPUT);
  pinMode(output12, OUTPUT);
  pinMode(output21, OUTPUT);
  pinMode(output22, OUTPUT);

  // Pins for ultrasonic distance sensor
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);

  // Pin for camera, camera takes a picture whenever the pin camera is HIGH
  // For this we are going to create an interrupt every 4 seconds, which means the camera will
  //  take a picture every 4 seconds
  pinMode(camera, OUTPUT);

  // Code to generate interrupt every 4 seconds

  noInterrupts(); // disable all interrupts
  // Will generate a interrupt every 4 seconds

  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1 = 0;
```

```
    OCR1A = 62500;                          // compare match
 register 16MHz/256/2Hz
    TCCR1B |= (1 << WGM12);                 // CTC mode
    TCCR1B |= (1 << CS12) | (1 << CS10); // 1024 prescaler
    TIMSK1 |= (1 << OCIE1A);                // enable timer
 compare interrupt

    interrupts(); // enable all interrupts
}

ISR(TIMER1_COMPA_vect) // timer compare interrupt service
 routine, this code will run every 4 seconds
{
    digitalWrite(camera, HIGH);
    delay(50);
    digitalWrite(camera, LOW);
}

void loop()
{
    delay(5);
    digitalWrite(trigger, HIGH);
    delay(50);
```

```
  digitalWrite(trigger, LOW);


  int duration = pulseIn(echo, HIGH);
  int distance = duration * 1.5 / 70; // Measures distance
in centimeters


  if (distance > 50)
  {
      // if distance is greater than 50, move the mover
forward with full power
      digitalWrite(output11, HIGH);
      digitalWrite(output12, LOW);
      digitalWrite(output21, HIGH);
      digitalWrite(output22, LOW);


      analogWrite(pwm1, 255);
      analogWrite(pwm2, 255);
  }


  // Obstacle detected, steer the mover


  else
  {
      pos = 0;
```

```
        // This loop will run till the pos
        while (1)
        {

            if (pos > 0)
            {
                if (target - pos < 30)
                {
                    break;
                }
            }
            else
            {
                if (target + pos < 30)
                {
                    break;
                }
            }

            // PID constants
            float kp = 5;
            float kd = 1.4;
            float ki = 3;
```

```
        // time difference
        long currT = micros();
        float deltaT = ((float)(currT - prevT)) / (1.0e6);
        prevT = currT;

        // error
        int e = pos - target;

        // derivative
        float dedt = (e - eprev) / (deltaT);

        // integral
        eintegral = eintegral + e * deltaT;

        // control signal
        float u = kp * e + kd * dedt + ki * eintegral;

        // motor power
        float pwr = fabs(u);
        if (pwr > 255)
        {
            pwr = 255;
```

```
            }


            // motor direction
            int dir = 1;
            if (u < 0)
            {
                dir = -1;
            }


            // signal the motors such that they rotate in the
 opposite direction
            // enabling the mover to steer by an angle of 90
 degrees


            setMotor(dir, pwr, pwm1, output11, output12);
            setMotor(-dir, pwr, pwm2, output21, output22);


            // store previous error
            eprev = e;
        }
    }
}


void readEncoder1()
```

```
{
    // It is sufficient to only note the position of one motor
  shaft
    int b = digitalRead(encoder12);
    if (b > 0)
    {
        pos++;
    }
    else
    {
        pos--;
    }
}

void setMotor(int dir, int pwmVal, int pwm, int in1, int in2)
{
    analogWrite(pwm, pwmVal);

    if (dir == 1)
    {
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
    }
    else if (dir == -1)
```

```
    {
        digitalWrite(in1, LOW);
        digitalWrite(in2, HIGH);
    }
    else
    {
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
    }
}
```