Journal de Ciencia e Ingeniería

Journal de Ciencia e Ingeniería, vol. 9, no. 1, pp. 18-23, agosto de 2017
Investigación - Robótica Educativa
ISSN 2145-2628, e-ISSN 2539-066X

Descripción de elementos de control y guía de robots Rover vía GPS

Description of Rover robot control and guidance elements with GPS

Eduardo Berra Villaseñor ⋆

Universidad del Valle de Puebla, 3 sur 5759 colonia el Cerrito, Puebla, México

Recibido: 10/10/2016; revisado: 20/12/2016; aceptado: 09/02/2017

E. Berra Villaseñor: Descripción de elementos de control y guía de robots Rover vía GPS. *Jou.Cie.Ing*. **9** (1): 18-23, 2017. ISSN 2145-2628, e-ISSN 2539-066X.

Resumen

Este articulo describe los elementos necesarios para la construcción de un robot tipo Rover con un sistema de control y guía mediante el posicionamiento GPS, en los últimos años el auge de robots autónomos y auto dirigidos para uso general ha tenido un incremento considerable, por lo que se muestran los esquemas de control necesarios así como las interacciones que el sistema deberá realizar para el control y guía de robot en una ruta preestablecida, en su memoria mediante mapas de Google.

***Palabras Clave*:** Autónomo, guía GPS, sistema de control, Rover, posicionamiento.

Abstract

This article describes the elements necessary for the construction of a robot type Rover, with a control and guidance system through GPS positioning, in recent years the boom of autonomous robots and self directed for general use has had a considerable increase, so Which shows the necessary control schemes as well as the interactions that the system must perform for the control and guidance of a robot in a pre-established route in its memory using Google maps.

***Keywords*:** Autonomous, GPS guide, control system, Rover, positioning.

1. Introducción

A partir de la década de los 90 el término no tripulado se ha utilizado continuamente en el mundo de la robótica. En el año 2000 gracias a la aparición de los micro controladores de lenguaje programable de alto nivel como lo es atmel, con los avances tecnológicos en los campos de los circuitos integrados y los polímeros de batería es ahora posible producir Robots autónomos que pueden proporcionar beneficios [1] y los recursos asignados al desarrollo de sistemas que permitan su a autonomía va en aumento. Las actividades que un robot autónomo puede hacer son diversas y dentro de las más requeridas se encuentran: el recorrido de rutas, detección de obstáculos, vigilancia de objetivos o entrega de paquetes a una ubicación determinada en mapas.

En este trabajo se considera el desarrollo de un auto tipo Rover(diseño robótico versión auto) capaz de tomar decisiones mediante un sistema de guía en base a una brújula magnética y la señal del sistema de posicionamiento global GPS.

La principal motivación para llevar acabo la construcción de este prototipo, es generar los algoritmos

⋆ **eduardo.berra@uvp.edu.mx**

y desarrollo de librerías dll del sistema necesario para control de robots mediante posicionamiento en mapas con la SDK de Google.

Para el desarrollo y puesta a punto del prototipo se ha dividido en las siguientes secciones:

1. Actuadores y hardware.
2. Algoritmos de detección y ubicación
3. Diseño de interface de rutas.

2. Desarrollo y puesta a punto

2.1. *Actuadores y hardware*

Los elementos que conforman a un robot autónomo Rover consisten en tres tipos:

1. Sensores
2. Actuadores
3. Micro controlador

Los sensores utilizados para la detección de obstáculos son comúnmente sensores ultrasónicos de corto alcance hc-sr04, también se toma como sensores la brújula magnética digital y el sensor GPS. El sistema de sensores debe utilizar la mínima cantidad de energía posible mientras opera sobre un amplio rango de escenarios [2].

Los actuadores son los motores que permiten el movimiento de robot en las diversas direcciones y se contemplan dos tipos de motores. Motores de DC y servo motores para el control de dirección.

Figura 1. Elementos de Hardware.

Los microcontroladores los podemos dividir en dos categorías drivers y micro controladores programables. Los micro controladores drivers son aquellos que nos darán la potencia controlada para los motores de DC, los programables controlan la acción de los drivers y también controlan al servo motor y las dirección del robot. Este conjunto de elementos se muestra en la figura 1.

2.2. *Algoritmos de detección y ubicación*

El objetivo principal de los algoritmos de control son proporcionar autonomía a las operaciones del robot, esto es mediante la toma de mediciones basadas en la brújula magnética y las lecturas de las coordenadas Gps [3].

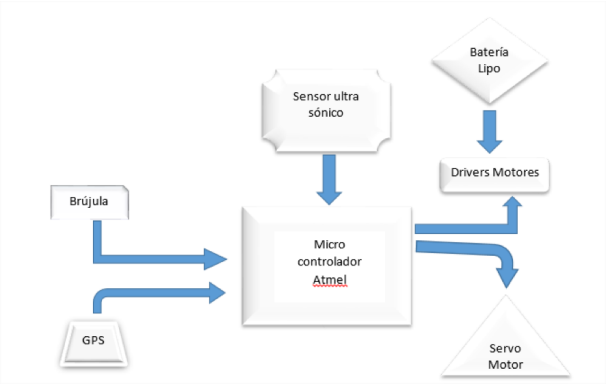

Figura 2. Diagrama de bloque para hardware.

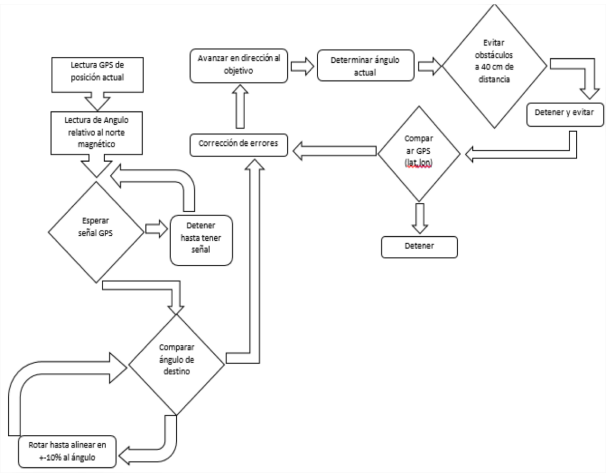

Figura 3. Diagrama de bloque para algoritmo de control.

Las principales operaciones de estos algoritmos son:

1. Robot Rover capaz de viajar del punto A al punto B sin necesidad de intervención de comandos controlados por humanos.
2. Ajustes de dirección realizados a través de planificación de trayectoria y detección de obstáculos.
3. Ubicación en dirección al punto especificado controlando los actuadores mediante corrección de errores PID.

El inicio del algoritmo embebido de control, tiene como objetivo esperar la señal GPS, [4] verificando la posición. El micro controlador tiene la labor de ubicar al robot en un ángulo alineado al destino activando los motores y comparando la posición en latitud y longitud de la señal actual del GPS con las del destino, en este punto el control de errores tiene su principal función ya que se corrige la dirección y velocidad del robot, después de igualar la latitud y la longitud del destino se procede a detener la actividad de los actuadores concluyendo que se ha alcanzado la posición deseada. Durante el recorrido el algoritmo es capaz de detectar obstáculos y tratar de maniobrar para evitarlos y retomar la ruta hacia el destino programado [5].

El origen y el destino son grabados en el micro controlador en un área reservada de memoria. La planificación de la trayectoria y la detección de obstáculos proveen al robot de la autonomía necesaria para seguir una ruta entre dos puntos de manera precisa. EL algoritmo para la detección de obstáculos mediante el sensor ultrasónico basa su funcionamiento en la siguiente tabla:

Distancia en cm	Ancho de pulso	Ancho de pulso bajo	Acciones
0	0	0	Acción des. 1
10	580	580	Acción des. 2
20	1160	1160	Acción des. 3
30	1740	1740	Acción des. 4

Tabla 1: Tabla para la detección de obstaculos.

Las acciones para evitar obstáculos constan de 4 pasos:

1. Consiste en accionar un solo motor para guiar al robot en un movimiento en forma de U hacia la izquierda.
2. Consiste en mover en diagonal al robot para tratar de encontrar una brecha que seguir para después ejecutar una acción en forma de U.
3. Consiste en mover al robot hacia la izquierda en dos acciones para calcular si la distancia se incrementa o decrementa, si la distancia decrementa se mueve a la derecha en caso contrario hacia adelante para evadir al obstáculo.
4. Consiste en mover al robot hacia la derecha en dos acciones para calcular si la distancia se incrementa o decrementa, si la distancia decrementa se mueve a la izquierda en caso contrario hacia adelante para evadir al obstáculo.

La figura 4 muestra la codificación de la tabla 1.

```
void checkDistancia ()
{
  Distancia = ultrasonic.Ranging(CM);
  if(LOG) Serial.println(Distancia);
}

void moove()
{
  if(Distancia > Dist_Obstaculo )
  {
    if(LOG) Serial.println("Adelante ");

    goAdelante();
    delay(TURN_DELAY);
  }
  else
  {
    stop();

    checkObstaculo();
  }
}

void checkObstaculo()
{
  int obsIzquier  = 0;
  int obsDerecha = 0;

  // Cuenta los Obstaculos desde Izquierda a Derecha
  for(AngServo = 0; AngServo <= 180; AngServo += 30)
  {
    headServo.write(AngServo);
    delay(TURN_DELAY);

    checkDistancia();
    if(Distancia < Dist_Obstaculo && AngServo < 90) obsIzquier++;
    else if(Distancia < Dist_Obstaculo) obsDerecha++;
  }

  if(LOG) Serial.print("TURN");

  if(obsIzquier && obsDerecha)
  {
    goAtras ();

    delay(TURN_DELAY * 2);

    if(obsIzquier < obsDerecha) goIzquier ();
    else goDerecha ();

    delay(TURN_DELAY);
```

Figura 4. Código para evasión de obstáculos.

2.3. *Diseño de interface de rutas*

Para grabar la información que el robot debe cubrir se maneja una interface basada en la SDK de google maps, la cual permite colocar pins en las ubicaciones por las cuales deberá de pasar el robot.

La SDK consiste en lo siguiente:

1. Mapa de selección
2. Obtener latitud y longitud del pin seleccionado
3. Grabar el pin inicial y final al robot

2.3.1. *Mapa de selección*

La interface se desarrolló en C por lo que el mapa se presenta en una ventana sobre la cual aparecerá un pin en la posición actual del robot, la cual es obtenida cuando se conecta al puerto USB de la pc y se encuentra en ejecución la interface. Como se puede mostrar en la figura 5.

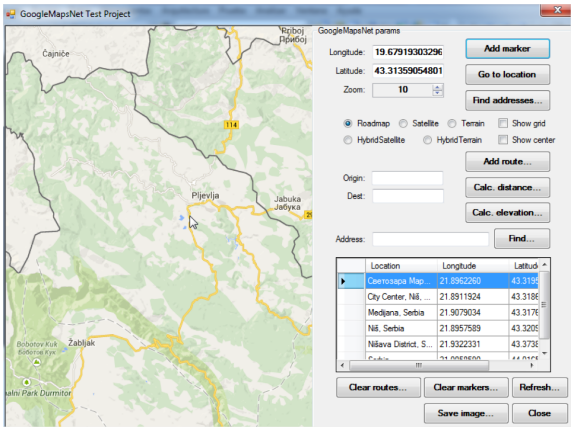

Figura 5. Interface de selección.

2.3.2. *Obtener latitud y longitud del pin seleccionado*

Al seleccionar una ubicación en el mapa podremos obtener la latitud y longitud de ese punto los cuales serán transferidos vía puerto USB seria al robot. El par de coordenadas determinadas para el origen y para el destino permite calcular la distancia entre los dos puntos así como el ángulo relativo entre ambos puntos, estos son calculados solo en la interface para referencia del usuario en relación a la distancia que el robot recorrerá, la programación necesaria para estos cálculos está relacionada en forma directa con las siguientes formulas:

$$rad = 3{,}1416/180; \quad (1)$$

$$R = 6378{,}137; //Radiodelatierraenkm \quad (2)$$

$$dLat = rad(lat2 - lat1); \quad (3)$$

$$dLong = rad(lon2 - lon1); \quad (4)$$

$$a = Math.sin(dLat/2) * Math.sin(dLat/2) + Math.cos(rad(lat1)) * Math.cos(rad(lat2)) * Math.sin(dLong/2) * Math.sin(dLong/2); \quad (5)$$

$$c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a)); \quad (6)$$

$$d = R * c; \quad (7)$$

$$bearing = Math.Atan2(y, x); \quad (8)$$

Para poder representar la latitud y la longitud obtenidas en el mapa mediante la interface con el pin seleccionado se utiliza las siguientes funciones derivadas de la SDK de Google.

```
googleMapsNet1.AddMarker
(Convert.ToDouble(textBoxLong.
Text.Trim()), Convert.ToDouble
(textBoxLat.Text.Trim()), Convert.
ToDouble(numericUpDownZoom.Value));
```

En el entorno de la librería se de vuelve la latitud y longitud en forma de texto en las casillas correspondientes.

2.3.3. *Grabar el pin inicial y final al robo*

Ambas ubicaciones aparecerán en el mapa y las variables de inicio y final de ruta tienen asignados los valores necesarios para ser grabadas en el micro controlador por los que las operaciones a realizar son las siguientes:

1. `Abrir el puerto serie`
2. `Enviar las siguientes secuencias:`
   f.Latitude = 19.02163;
   f.Longtitude = -98.21222;
   g.Latitude = 19.00355;
   g.Longtitude = -98.20562;
3. `Cerrar el puerto`

Como se puede observar en las instrucciones anteriores f son las coordenadas finales mientras que g son las iniciales, la figura 6 muestra el mapa y los pines asignados para la captura de las coordenadas.

Figura 6. Pin inicial y pin final de ruta.

La ruta es grabada en la memoria del micro procesador con lo cual el sistema de control puede comenzar los cálculos para el movimiento de robot [6]. La figura 7 muestra el Robot rover construido y programado.

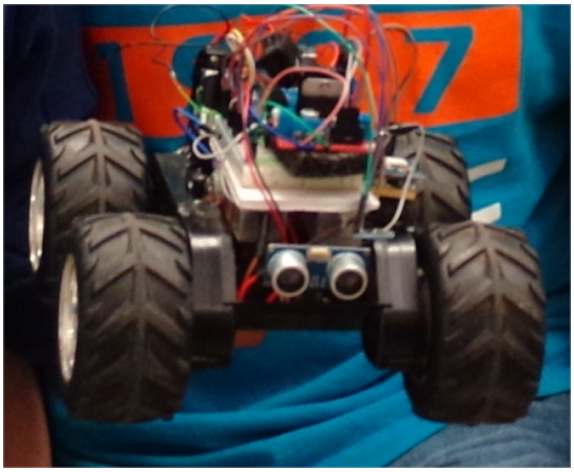

Figura 7. Robot Rover.

3. Problemas en la detección de obstáculos y precisión de la ubicación

Un censor ultrasónico es capaz de detectar un objeto en un rango de 3.5 metros con ángulos de -45° a 45° como lo muestra la Figura 9.

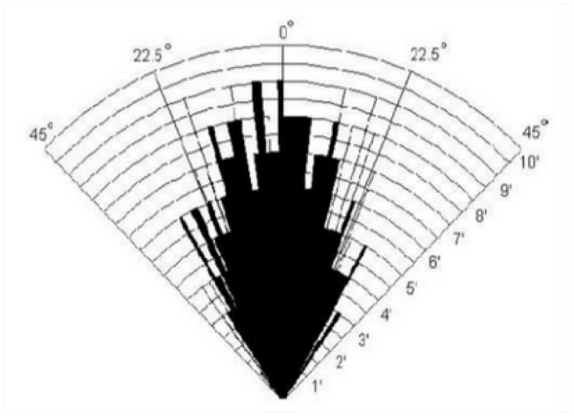

Figura 8. Campo de visión del sensor.

Un solo sensor ultrasónico plantea varios inconvenientes en el diseño de autonomía del robot [7], debido a que la detección solo puede ser frontal por lo que se pierde el control de cualquier otro obstáculo cercano. La forma más adecuada de poder generar un campo visual correcto será tener sensores alrededor del robot, solo que este arreglo tiene como inconveniente el censado para evitar la interferencia entre sensores. Un método para el censado típico basa su función en tiempos de 100 a 500 ms [8] y técnicas como la probabilística de obstáculos en forma de maya pueden mejorar el algoritmo para evitar obstáculos.

El modelo de Gps utilizado en la construcción del Robot rover dio como herror un aproximado de 3 a 4 metros, tomando en cuenta que el sensor gps es de gama media se encuentra dentro de los rangos apropiados de error, por lo que la forma más sencilla de corregirlo es utilizando un sensor de gama alta un poco más costoso pero que permita tener un rango de precisión mayor como son los Gps RTK (del inglés Real Time Kinematic) la marca Piksi tiene un rango de 1 a 0.5 metros de diferencia una premiación puntual de rango mejorada.

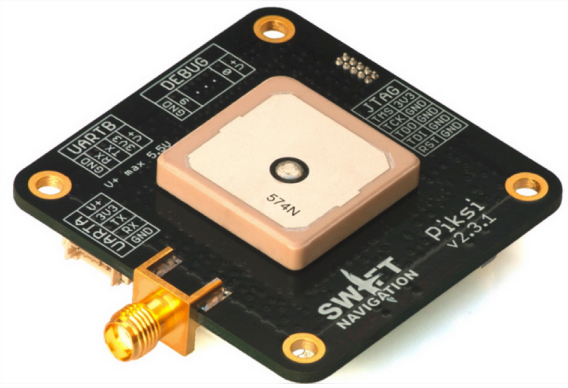

Figura 9. GPS RTK Piksi.

4. Resultados

Se realizó la construcción de una interface para fijar la ruta entre dos puntos geo referenciados, de la misma manera se diseñó un robot tipo rover capas de ser controlado por una brújula magnética, un censor ultrasónico para evitar obstáculos y un sensor GPS, para detectar su ubicación este robot fue probado en las canchas de la Universidad del Valle de Puebla pasando las pruebas de obstáculos y de seguimiento de rutas establecidas por los usuarios.

5. Conclusiones

De acuerdo con las descripciones de los elementos necesarios, para la construcción de un robot autónomo rover en el presente artículo, podemos concluir que la precisión de la ruta durante las pruebas realizadas, fue aceptable y en trabajos futuros se pretende mejorar la precisión de la detección del punto de ubicación con el GPS. Para la evasión de los obstáculos se pudo

concluir que un solo censor ultrasónico complica la implementación de un algoritmo eficiente para esquivar objetos en movimiento, por lo que se implementara una mejora mediante el incremento en 360^0 con sensores, que permitan hacer un censado panorámico de los posibles obstáculos. La elaboración del sistema cumplió con la guía autónoma mediante una brújula magnética y un sistema básico de GPS, por lo que el rover es capaz de seguir una ruta de varios puntos, realizar recorridos evadiendo obstáculos simples, mientras que las librerías dll desarrollados, servirán de base para la generación de nuevos desarrollos de software para robótica autónoma guiada por GPS.

Referencias

[1] Mithileysh Sathiyanarayanan, Syed Azharuddin, Santhosh Kumar, and Gibran Khan. Self controlled robot for military purpose. *International Journal For Technological Research In Engineering*, 1(10):1075–1077, 2014.

[2] F. A. Urbano-Molano. Redes de sensores inal ambricos aplicadas a optimizaci on en agricultura de precision para cultivos de café en colombia. *Journal de Ciencia e Ingenieria*, 5(1):46–52, 2013.

[3] Saurav Kumar. Binocular stereo vision based obstacle avoidance algorithm for autonomous mobile robots. In *Advance Computing Conference, 2009. IACC 2009. IEEE International*, pages 254–259. IEEE, 2009.

[4] Peter Sykacek, Iead Rezek, and SJ Roberts. Markov chain monte carlo methods for bayesian sensor fusion. *Dept. Eng. Sci., Univ. Oxford, Oxford, UK, Tech. Rep. PARG-00-10.[Online]. Available: http://www. robots. ox. ac. uk/˜ parg*, 2000.

[5] Roman Kuc and Victor Brian Viard. A physically based navigation strategy for sonar-guided vehicles. *The international journal of robotics research*, 10(2):75–87, 1991.

[6] POLAROID Corporation. Ultrasonic components group. Technical report, 119 Windsor Street, Cambridge, Massachusetts 02139, 1989.

[7] Johann Borenstein and Yoram Koren. Error eliminating rapid ultrasonic firing for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and automation*, 11(1):132–138, 1995.

[8] Anita M Flynn. Combining sonar and infrared sensors for mobile robot navigation. *The International Journal of Robotics Research*, 7(6):5–14, 1988.