Programación Matemática y Software (2015) 7 (2): 43-51. ISSN: 2007-3283

Reconstrucción de mapas para el recorrido de un vehículo sobre ruedas mediante un dispositivo de cómputo móvil

Map reconstruction for the path of a wheel mobile robot using a mobile computing device

Arturo Juárez-Ríos,[1]* Juan Carlos Herrera-Lozada,[1] Hind Taud,
Jesús Antonio Álvarez-Cedillo,[1] Jacobo Sandoval-Gutiérrez

[1] Centro de Innovación y Desarrollo Tecnológico en Cómputo, Unidad Profesional Zacatenco, Instituto Politécnico Nacional
Miguel Othón de Mendizábal s/n, edificio del CIDETEC, col. Nueva Industrial Vallejo.
Del. Gustavo A. Madero. CP 07700. Ciudad de México, México
* Correo-e: arturo.juarez@hotmail.com

PALABRAS CLAVE:

navegación, sistemas embebidos, mapa de reconstrucción, vehículo con ruedas, dispositivo móvil, interfaz inalámbrica

RESUMEN

Este trabajo describe el uso de la tecnología de cómputo móvil para la generación de mapas (reconstrucción de entornos). Se desarrolló un sistema completo que captura datos a través del uso de sensores ultrasónicos de distancia y una brújula digital que forma parte de un sistema embebido con base en un microcontrolador, en este caso, una tarjeta Arduino, montado en un vehículo con ruedas de tracción diferencial. El sistema de control de movimiento y procesamiento de datos es una aplicación que se ejecuta en un sistema de cómputo móvil con sistema operativo Android para obtener de manera gráfica el entorno que ha recorrido el vehículo. Por último, se construyó un sistema de comunicación inalámbrica wifi con arquitectura cliente-servidor entre las dos plataformas.

KEYWORDS:

navigation, embedded systems, reconstruction map, wheel mobile robot, mobile computing, wireless interface

ABSTRACT

This paper describes the use of mobile computing technology to generate maps (reconstruction of environments). A complete system has developed and it uses ultrasonic distance sensors and a digital compass to capturing data as part of an embedded system based on a microcontroller, in this case an Arduino board system, mounted on a mobile wheeled vehicle with differential traction. The motion control and data processing system is an application running on a mobile computing system with Android Operating System to graph the environment that the mobile wheeled vehicle has navigated. Finally, a Wi-Fi wireless communication system, with Client-Server architecture is built between the platforms.

Recibido: 15 de enero de 2015 • **Aceptado:** 28 de abril de 2015 • **Publicado en línea:** 30 de junio de 2015

1 INTRODUCCIÓN

Una de las principales tareas relacionadas con robots consiste en la posibilidad de monitorearlos y controlarlos de manera remota para que puedan desarrollar el trabajo o la tarea para la cual fueron diseñados. Un objetivo del diseño de robots móviles es dotarlos de autonomía. Un robot que desconoce su entorno, incluso su posición inicial, pero que tiene la habilidad de explorar y construir un mapa, podría crear trayectorias óptimas y realizar algunas tareas específicas sin la intervención de algún operador [1].

En este trabajo se propone el uso de dispositivos de cómputo móvil como lazo primario de control y supervisión remota, debido a que estos han incrementado sus capacidades y velocidades de procesamiento, lo que ha aumentado en gran medida su viabilidad para este propósito. Se priorizó una comunicación wifi entre los módulos y se garantizó el acceso a internet para que la aplicación sea tele- operada desde cualquier parte del mundo. Se considera también, para efectos del prototipo desarrollado, un escenario controlado con entornos de paredes horizontales y verticales, lo cual simplifica el trabajo efectuado para la captura y elaboración de los mapas; ya que la lectura o toma de la información durante la trayectoria a seguir sólo se hará cuando el vehículo móvil se encuentre en alguna de cuatro posiciones: vertical u horizontal con respecto a su posición inicial. Esto sin salir del objetivo de generar mapas en el dispositivo de cómputo móvil, con base en la información transmitida por wifi, que fue capturada por sensores remotos colocados en un vehículo móvil controlado a través de la misma comunicación inalámbrica.

2 DESARROLLO

2.1 Descripción general del sistema

En la figura 1 se muestran los elementos del sistema, que está dividido en tres subsistemas: a) el sistema de captura de datos usa un vehículo sobre ruedas y un sistema embebido, y es el encargado de recorrer toda la zona a explorar y registrar durante el recorrido las medidas de las distancias hacia las paredes frontal y laterales, así como obtener la dirección y sentido del vehículo móvil; b) el sistema de comunicación inalámbrica es el encargado de realizar la conexión

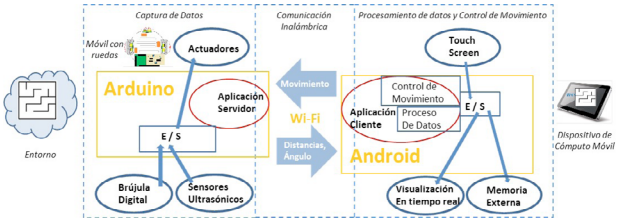


Figura 1. Diagrama del sistema

entre una tarjeta Arduino y el sistema operativo Android, además, mediante un protocolo de comunicación común, transfiere la información, desde y hacia el dispositivo de cómputo móvil, referente a las órdenes de movimiento hacia el vehículo móvil con ruedas e información obtenida por el sistema de captura de datos, y por último, c) el sistema de procesamiento de datos y control de movimiento es el encargado de generar las órdenes necesarias para que el vehículo móvil pueda realizar los movimientos a través del recorrido de la zona a explorar, además, con los datos recibidos desde el sistema de captura, representa gráficamente el mapa del entorno explorado cuyos datos son almacenados en un archivo para que puedan ser usados en otro momento y recuperar gráficamente el mismo mapa.

2.2 Captura de datos

Este sistema consta esencialmente de un vehículo móvil circular, que tiene en sus laterales dos motorreductores de corriente continua trabajando de manera aislada. En la parte baja cuenta con dos ruedas locas colocadas una al frente y otra atrás. Como etapa de potencia que alimenta a los dos motores, se utiliza el circuito integrado L293D [2, 3], que incluye cuatro circuitos para manejar cargas de potencia media, en especial pequeños motores con la capacidad de controlar una corriente hasta 600 mA en cada circuito y un voltaje entre 4.5 V y 36 V. Con un circuito integrado se forman dos puentes H completos para manejar los dos motores.

Se adaptaron también en el vehículo tres sensores de distancia ultrasónicos HC-SR04, uno en la parte frontal y los otros en los laterales. Los sensores están diseñados de acuerdo con especificaciones técnicas

para medir distancias entre 2 y 400 cm [4]. Este sensor ultrasónico utiliza 4 pines: uno para alimentación de 5 V, otro para tierra, además de un pin denominado *trigger*, que emite una señal sonora de 10 µs, espera a detectar el eco mediante la detección del fin de la señal HIGH recibida por el cuarto pin *echo* [2]. Con el tiempo medido de ida y vuelta de la señal sonora, conociendo la velocidad del sonido (340 m/s), se calcula la distancia hacia el objeto sobre el que rebotó la señal.

Sobre el vehículo se coloca un escudo (*shield*) Arduino wifi, con base en una tarjeta Arduino Mega 2560 [5]. En el escudo se utilizan los pines 3, 5, 7 y 9 como salidas con modulación por ancho de pulso (PWM, por sus siglas en inglés) para el control de movimiento de los motores a través del circuito de potencia. Seis de sus entradas analógicas (pines 41 al 46) [2, 5] se utilizan para conectar los tres sensores ultrasónicos de distancia.

Se conecta también un magnetómetro HMC5883L [6] a los pines de comunicación de datos 20 (SDA) y de reloj 21 (SCL), para que funcionen como brújula digital. El protocolo de comunicación empleado (I^2C) en este magnetómetro es compatible con la tarjeta Arduino y es empleado para la comunicación entre ellas. Su principal característica es que utiliza dos líneas para transmitir la información: SDA transfiere los datos y SCL envía la señal de reloj [2]. La plataforma Arduino cuenta con bibliotecas preestablecidas; por ejemplo, la función pulseIn() nos da el valor numérico del tiempo transcurrido en la ida y vuelta del ultrasonido generado en una variable de tipo unsigned long. La función getHeading() obtiene el ángulo respecto del Norte geográfico, que puede ser capturado en una variable de tipo float. Internamente y de manera transparente para los usuarios, realiza las conversiones analógico-digital.

Finalmente, el sistema Arduino con la aplicación desarrollada, que incluye funciones preestablecidas con el uso de bibliotecas desarrolladas por los fabricantes de las diferentes tarjetas, entrega en valores numéricos las mediciones hechas por los distintos sensores, y posteriormente, mediante algunos cálculos matemáticos, se obtienen los resultados requeridos, por ejemplo, la distancia *(d=v*t)* a partir de la medida del tiempo. La figura 2 muestra los elementos utilizados.

La aplicación Servidor desarrollada se instala en la tarjeta Arduino. En esta aplicación se programan las tareas necesarias para alimentar las entradas

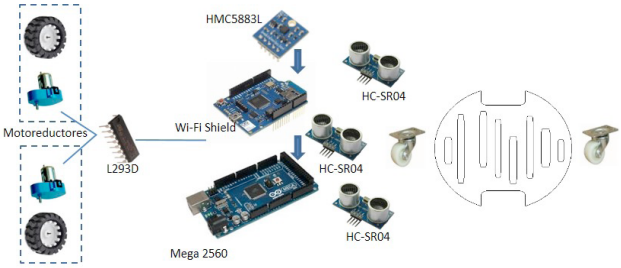


Figura 2. Elementos del sistema de captura de datos

del puente H (L293D) con un valor PWM entre 0 y 255 para controlar y hacer girar los motores en uno u otro sentido, dependiendo del tipo de movimiento requerido. A su vez, define los tiempos en que funcionarán el magnetómetro y los sensores de distancia, para posteriormente almacenar los valores registrados en diferentes variables. Los valores de tipo númerico de las medidas obtenidas son transformadas a tipo *string* y entonces concatenadas para crear una cadena con la siguiente información:

medidaFte+","+medidaIzq+","+medidaDer+","+Vizq+","+Vder+","+ángulo

Esta cadena de caracteres es enviada a la aplicación Cliente, cada vez que es solicitada, para su posterior procesamiento y almacenamiento. Los valores registrados por los sensores ultrasónicos son: *medidaFte*, *medidaIzq* y *medidaDer*. Los valores PWM son: *Vizq* y *Vder* para los motores, y ángulo representa la dirección del vehículo móvil en un momento dado con respecto al Norte geográfico.

2.3. Comunicación inalámbrica

El sistema de comunicación se basa en una estructura cliente/servidor. El servidor centraliza el servicio y lo ofrece a través de una dirección conocida (dirección IP y número de puerto). El cliente controla la interacción del servidor con el usuario [7]. La figura 3 muestra el funcionamiento de esta estructura.

El escudo Arduino se programa para conectarse a una red wifi a través de un punto de acceso o módem, donde se almacena la aplicación que trabaja como servidor, como alimentador de información para los

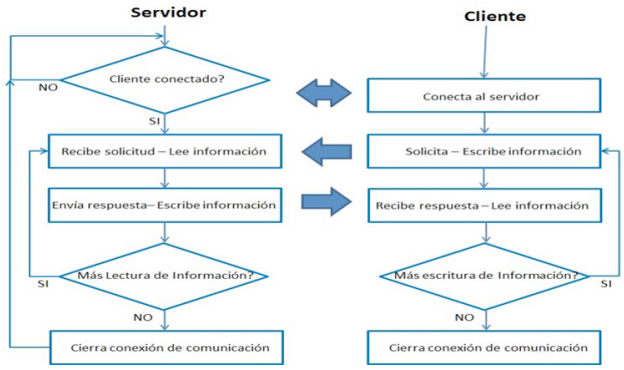


Figura 3. Estructura cliente/servidor

motores y distribuidor de las medidas otorgadas por los sensores ultrasónicos y el magnetómetro.

En este proyecto se prueban tres de las posibles configuraciones: la primera aprovecha el uso de un teléfono inteligente con la funcionalidad de punto de acceso, a través de la cual se genera una red local wifi conectada a internet; una segunda configuración utiliza un teléfono inteligente o una tableta digital con conexión wifi y sobre el vehículo un módem que construye una red local, por último, se puede usar un módem externo wifi que puede tener o no una conexión a internet, pero que finalmente construye la red inalámbrica entre el dispositivo de cómputo

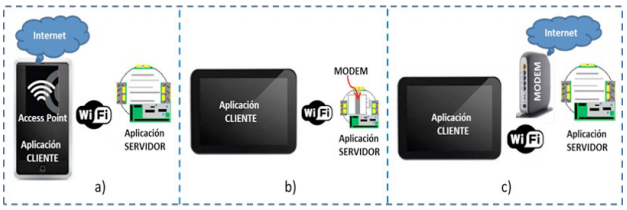


Figura 4. Configuraciones de conexión de red inalámbrica

móvil y el sistema embebido con el vehículo. Las configuraciones pueden apreciarse en la figura 4.

En cualquier caso, la aplicación Cliente está almacenada en el dispositivo de cómputo móvil, la aplicación Servidor en el vehículo, y ésta puede tener asociada una dirección IP estática (fija) acorde a la red construida o una dirección IP dinámica proporcionada de manera automática por el módem o el punto de acceso (*Access Point* en el teléfono); el puerto elegido para la programación del *socket* de conexión es el 2000. En la aplicación Servidor se define el puerto con *WiFiServer server(2000)* y, en caso de configurar la dirección IP fija, por ejemplo, *WiFi.config(192.168.1.99)* en la aplicación Cliente para configurar la dirección IP fija y el puerto asociado al Servidor que se pretende conectar, se define una instancia, por ejemplo *sk=new Socket(192.168.1.99, 2000)*.

Un *socket* define las reglas que van a seguir los programas que utilizan los servicios de nivel de transporte en una red TCP/IP para permitir el intercambio de información entre estos programas [7]. En este proyecto se da entre dos programas instalados en diferentes plataformas, Android y Arduino, pero que ocupan el mismo protocolo de comunicación, TCP/IP. Cada *socket* está identificado por una dirección diferente (IP) más un número de puerto (de 16 bits).

La aplicación Servidor permanece en un ciclo de espera hasta que un cliente solicita conexión. En el sistema de comunicación programado, el que decide la desconexión es el Cliente, en ese momento se desconecta de la red y el Servidor al no detectar la conexión se desconecta también y se produce un nuevo ciclo de espera.

Figura 5. Elementos de la pantalla de inicio

2.4 Procesamiento de datos y control de movimiento

Esta es la aplicación Cliente. Se define una pantalla de inicio dónde se muestra el directorio de los mapas almacenados, del cual se puede elegir uno para visualización en la misma pantalla, o la opción de crear un nuevo mapa. La figura 5 muestra los elementos de esta pantalla.

Los nombres de los mapas almacenados en memoria empiezan con una *m* y continúan con un número consecutivo de dos dígitos. Se definió la

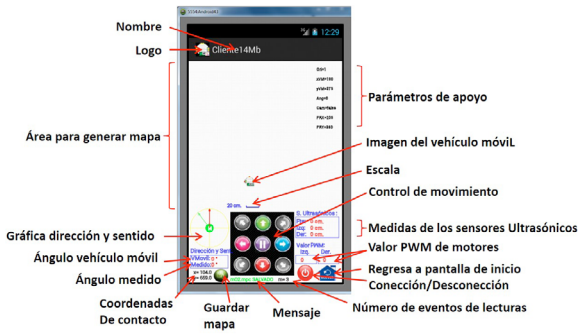


Figura 6. Elementos de la interfaz de la aplicación para proceso de datos y control de movimiento

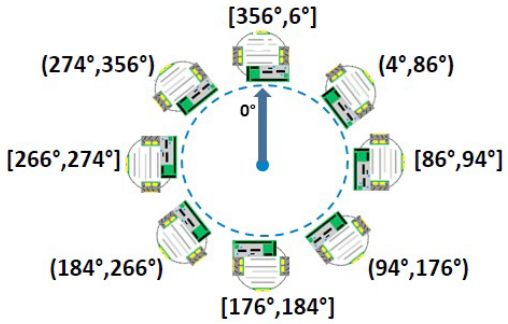


Figura 7. Representación gráfica de la dirección y sentido del vehículo móvil con ruedas

extensión ".mpc" para indicar que es un archivo con formato del tipo "mapa proyecto CIDETEC", y que es de esta manera como se almacena.

Al seleccionar la opción 'Nuevo', se define el área donde se muestra la interfaz gráfica para controlar al vehículo motorizado y los datos obtenidos desde el servidor, donde se coloca la gráfica en tiempo real. La figura 6 muestra los elementos de esta pantalla.

En el caso de ambas pantallas, se define el método *onTouchEvent* que monitorea cuándo se toca la pantalla del dispositivo móvil para obtener la posición (*x,y*) con la cual se puede saber cuál fue el "botón" presionado [8]. Al identificar con un valor numérico los botones de control de movimiento de izquierda a derecha y de arriba abajo, se definen de la siguiente manera: 2 adelante; 4 giro a la izquierda; 5 detener y tomar datos; 6 giro hacia la derecha; 8 retroceder; 1 y 3 aumentar valor PWM para motor izquierdo y derecho, respectivamente; 7 y 9 para disminuir valor PWM para el motor izquierdo y derecho, respectivamente. Se cuenta con una gráfica que representa la dirección y sentido del vehículo y la posición del Norte geográfico. Se muestra un segmento de recta que representa 20 cm del plano, con lo que podemos saber la escala de la gráfica. Se visualiza también mediante un icono la dirección y sentido del vehículo en cada movimiento con respecto de la posición inicial (cero grados y mirando hacia enfrente), tomando las mediciones en sentido al movimiento de las manecillas del reloj, en la figura 7 se muestra cada una de las ocho posibles posiciones del vehículo con ruedas, dependiendo del ángulo medido desde el sensor.

Elegir una tolerancia de 4° de desviación hacia un sentido u otro permite saber si el vehículo se encuentra en la posición correcta, ya sea hacia enfrente, 0° (356°- 4°); hacia la izquierda, 270° (266°-274°); hacia la derecha, 90° (86°-94°); hacia atrás, 180° (176°-184°). De acuerdo con el icono mostrado se puede saber si el vehículo se encuentra en la ruta adecuada, por ejemplo, al desplazarlo hacia enfrente, girarlo hacia la derecha y detectar el momento en el que alcanza los 90° requeridos para proseguir el camino. Para este proyecto se consideran necesarias sólo estas ocho posibles posiciones.

Se cuenta con un botón en la parte inferior derecha de la pantalla para realizar la conexión o desconexión con el sistema de captura. Al finalizar la conexión, genera el archivo del mapa correspondiente. Mientras está conectado, los datos obtenidos se almacenan en un arreglo *dist* de cuatro campos, con: *dist[i][0]*=medida sensor frontal, *dist[i][1]*=medida sensor izquierdo, *dist[i][2]*=medida sensor derecho, y *dist[i][3]*=ángulo del vehículo a partir de una posición inicial de 0 grados. La *dist[i][3]* se calcula restando la medida tomada por el magnetómetro a la primer medida tomada por el mismo, esto es:

$$dist[i][3] = dist[i][3] - dist[1][3]$$

Si este resultado da un valor negativo, se aplica la siguiente sentencia para conversión del ángulo en el rango de 0 a 360 grados:

Si *dist[i][3] es menor a 0 entonces dist[i][3] = dist[i][3] + 360;*

La variable *i* representa la posición del conjunto de datos obtenidos con valores entre *1* y *m*, siendo *m* la última posición. A partir de este arreglo se grafican tres puntos por cada línea de datos, con lo que al final se tienen *m* conjuntos de tres puntos.

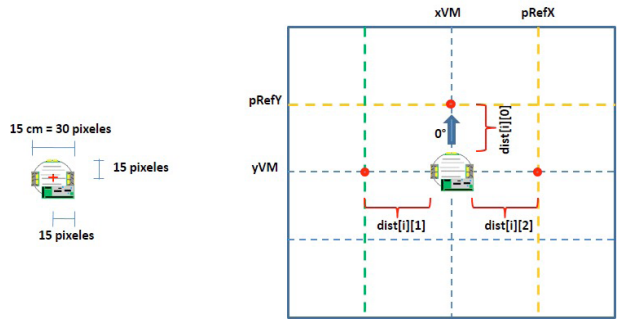


Figura 8. Posición frontal del vehículo con ruedas y graficación del mapa

Para graficar los puntos correspondientes al mapa generado, se considera una posición inicial (*xVM*,*yVM*) del vehículo en la pantalla del dispositivo de cómputo móvil, y se toma el primer conjunto de datos del arreglo: *dist[1][0]*, *dist[1][1]*, *dist[1][2]*, *dist[1][3]*. El último, *Dist[1][3]*, es el valor de desplazamiento angular con respecto del Norte geográfico, que será utilizado para indicar que a partir de ahí será medido el giro del vehículo, y se registra esa posición en una variable con valor inicial *Angulo=0*. El tamaño real del vehículo con ruedas es de 15 cm de ancho por 15 cm de largo. Se considera una relación 1:2 (1 cm = 2 pixeles), como se muestra en la figura 8.

En esta figura se aprecian tres valores de referencia, uno es el ángulo inicial de 0°, otro es el punto de referencia en el eje de las *y*, *pRefY* (la coordenada en *y* de la pared frontal), y el tercero es el punto de referencia en el eje de las *x*, *pRefX* (la coordenada en *x* de la pared derecha). Se considera una variable llamada orientación, cuyo valor entero entre 1 y 4 indica la posición del vehículo (delante, izquierda, atrás, derecha), y otra variable *cambio* (*cambio = true* o *cambio = false*) que nos informa si hubo un cambio entre una posición y otra. Si es la primera medida (*i=1*) o hubo un cambio (*cambio=true*) en la posición frontal (*orientacion=1*), los puntos de referencia en *x* y *y* son calculados de la siguiente manera:

$$pRefX = xVM + dist[i][2] + 15$$
$$pRefY = yVM - dist[i][0] - 15$$

Si durante el recorrido longitudinal a su posición inicial en su sensor derecho midiera una distancia mayor o menor a las lecturas anteriores, tomando un valor de 5 cm como límite hacia ambos lados, significaría que hay una pared más lejana o cercana respectivamente, la cual se tomaría como nuevo punto de referencia en *x (pRefX)*. Igual pasaría si durante su recorrido transversal a su posición inicial, el sensor detectara algún cambio sensible en su medición, entonces se tomaría éste como nuevo punto de referencia en *y (pRefY)*.

Si se sigue una ruta frontal, la posición del vehículo y los tres puntos a graficar se calculan de la siguiente manera:

Vehículo móvil: *(xVM, yVM)*, siendo

$$xVM = pRefX - dist[i][2] - 15$$
$$yVM = pRefY + dist[i][0] + 15$$

punto frontal: *(xVM, pRefY)*
punto izquierda: *(xVM - dist[i][1] – 15, yVM)*
punto derecha: *(pRefX, yVM)*

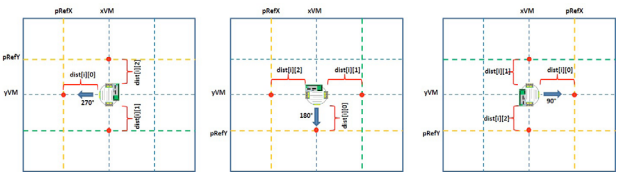


Figura 9. Posición izquierda, atrás y derecha, respectivamente, del vehículo móvil

En la figura 9 se muestran los casos cuando hay un cambio de posición del vehículo hacia la izquierda, atrás o derecha, y la forma de graficar los puntos obtenidos en esas posiciones.

Si hubo un cambio (*cambio=true*) a la posición izquierda (*orientacion=2*), los puntos de referencia en *x* y *y* se calculan de la siguiente manera:

$$pRefX = xVM - dist[i][0] - 15$$
$$pRefY = yVM - dist[i][2] - 15$$

La posición del vehículo y los tres puntos a graficar se calculan de la siguiente manera:

Vehículo: *(xVM, yVM)*, siendo

$$xVM = pRefX + dist[i][0] + 15$$
$$yVM = pRefY + dist[i][2] + 15$$

punto frontal: *(pRefX, yVM)*
punto izquierda: *(xVM, yVM + dist[i][1] + 15)*
punto derecha: *(xVM, pRefY)*

Si hubo un cambio (*cambio=true*) a la posición atrás (*orientacion=3*), los puntos de referencia en *x* y *y* se calculan de la siguiente manera:

$$pRefX = xVM - dist[i][2] - 15$$
$$pRefY = yVM + dist[i][0] + 15$$

La posición del vehículo móvil y los tres puntos a graficar se calculan de la siguiente manera:

Vehículo: *(xVM, yVM)*, siendo
$xVM = pRefX + dist[i][2] + 15$
$yVM = pRefY - dist[i][0] - 15$
punto frontal: *(xVM, pRefY)*
punto izquierda: *(xVM + dist[i][1] + 15, yVM)*
punto derecha: *(pRefX, yVM)*

Si hubo un cambio (*cambio=true*) a la posición derecha (*orientacion=4*), los puntos de referencia en *x* y *y* se calculan de la siguiente manera:

$$pRefX = xVM + dist[i][0] + 15$$
$$pRefY = yVM + dist[i][2] + 15$$

La posición del vehículo móvil y los tres puntos a graficar se calculan de la siguiente manera:

Vehículo móvil: *(xVM, yVM)*, siendo
$xVM = pRefX - dist[i][0] - 15$
$yVM = pRefY - dist[i][2] - 15$
punto frontal: *(pRefX, yVM)*
punto izquierda: *(xVM, yVM - dist[i][1] - 15)*
punto derecha: *(xVM, yVM + dist[i][2] + 15)*

3 RESULTADOS

Durante el desarrollo se probaron diferentes plataformas directamente en el emulador de Eclipse con: a) un teléfono inteligente con sistema operativo Android 4.1.2, utilizándolo como punto de acceso y cliente; b) un teléfono inteligente con sistema Android 2.3.6, con un módem externo, y c) una tableta de 10 pulgadas con sistema Android 4.0.4, también con un módem externo. La conexión se realizó exitosamente, las diferentes órdenes de movimiento enviadas mediante la pantalla táctil se recibieron y ejecutaron en el sistema Arduino. El teléfono inteligente recibió y visualizó las medidas de las distancias, valores PWM y ángulo enviados por el sistema Arduino. La figura

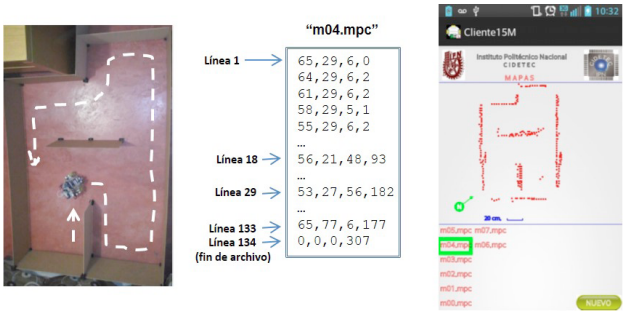


Figura 10. Resultados obtenidos

10 muestra el resultado de uno de los escenarios explorados.

Se construyeron escenarios con tablas de 23 cm de altura y varias medidas de longitud. En esta figura se muestra el recorrido realizado por el vehículo por uno de los escenarios. Durante el recorrido por este escenario se obtuvo un conjunto de 134 líneas de datos (o conjunto de mediciones), que a su vez se envió durante la comunicación inalámbrica, desde el servidor hacia el cliente, y posteriormente se almacenó dentro del dispositivo de cómputo móvil en un archivo llamado *m04.mpc*. La línea 18 muestra los datos obtenidos en el primer giro de 90 grados del recorrido; la línea 29 muestra el segundo giro hacia abajo de la imagen; la línea 133 obtiene el último registro, el momento que se muestra en la pantalla del teléfono inteligente en el lado derecho de la imagen. En la última línea del archivo se almacena el primer valor del ángulo medido, que es el ángulo que existe entre el Norte geográfico y la posición inicial del vehículo, medido en el sentido de giro de las manecillas del reloj, lo que permite graficar en pantalla la dirección y sentido del Norte, que se muestra como una flecha de color verde.

De acuerdo con los valores obtenidos por el sensor ultrasónico frontal, podemos apreciar que las mediciones se realizaron cada 3 cm de avance del vehículo, aproximadamente, lo que nos proporciona de manera gráfica un mapa con buena resolución.

Por la manera de obtener datos, solo en posiciones longitudinales y transversales a la posición inicial podemos apreciar huecos en las esquinas internas por donde el vehículo circula, porque son lugares donde no se toma información, a diferencia de las esquinas del lado externo, donde no existe algún problema para registrar la información pertinente.

4 CONCLUSIONES

Los sistemas de cómputo móvil, básicamente teléfonos inteligentes y tabletas, pueden emplearse como dispositivos primarios de control, en este caso, para el monitoreo y control de un sistema embebido para la reconstrucción de mapas de entorno a través de un sistema de comunicación inalámbrica. Se logró la comunicación entre las dos entidades: el dispositivo de cómputo móvil con Android y el sistema embebido Arduino a través de elementos comunes, como el manejo de sockets y el mismo protocolo de comunicación. La propuesta original de utilizar sensores de distancia ultrasónicos redujo el costo final del prototipo; no obstante, la calibración y adecuación de éstos es una actividad sensible que puede modificar los resultados del diseño. Este proyecto podrá servir de base para otros trabajos que requieran monitorear o controlar un sistema remoto de manera inalámbrica a través del acceso a una red local o internet, sin ser exclusivo para robots sino que puede extenderse a aplicaciones domóticas y similares.

5 TRABAJO A FUTURO

Por el momento, el control del vehículo sobre ruedas se realiza de manera manual (teleoperado), por lo que se trabajará para programar la función automática. Por el tipo de entornos o escenarios probados se podría seguir una ruta siguiendo las paredes, siempre del lado derecho y hacia enfrente del vehículo, lo cual permitiría realizar todo el recorrido hasta llegar al punto inicial. Si se conocen las dimensiones externas del área explorada se podría programar la aplicación Cliente para identificar puntos de salida. Existen sensores de distancia como el láser, con el cuál se obtendría una mejor resolución del mapa, además que la captura de información podría ser en cualquier dirección y sentido, no sólo en las cuatro utilizadas, con lo que se podrían mapear objetos o paredes de distintas formas; aunque, cabe señalar, esta solución es más costosa no sólo en el aspecto económico sino también en el computacional.

AGRADECIMIENTOS

Los autores agradecen el apoyo brindado por la Secretaría de Investigación y Posgrado del Instituto Politécnico Nacional a través del proyecto con registro 20144053 – Cómputo Móvil para la Supervisión de Sistemas Embebidos de Bajo Costo. De igual manera agradecen al Consejo Nacional de Ciencia y Tecnología (Conacyt) a través del Proyecto Cátedras Conacyt 1507 – Arquitecturas de Sistemas Embebidos para la investigación científica y la integración de tecnologías.

REFERENCIAS

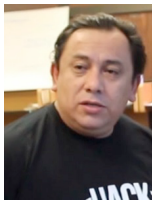
1. Zanatta Juárez, A. G. Generación automática de mapas de entorno usando un robot móvil con visión por computadora. *Tesis de Maestría*. México, DF: Instituto Politécnico Nacional, Centro de Investigación en Computación, 2010.
2. Torrente Artero, O. *Arduino: curso práctico de formación*. México, DF: Alfaomega Grupo Editor, México, pp. 61-127, 355-357, 446.
3. Texas Instrument. L293, L293D, *Cuadruple Halh-H Drivers*, Reference Code SLRS008C September 1986-Revised November 2004.
4. Cytron Technologies. *HC-SR04 Ultrasonic Sensor. User's Manual* V1.0, 2013.
5. Arduino. Embebidos [en línea]. Recuperado el 10 de octubre de 2014. http://www.arduino.cc/es/ , 2014.
6. Honeywell International Inc. *3-Axis Digital Compass IC HMC5883L, Form #900405 Rev B*, October 2010.
7. Tomás Gironés, J. *El gran libro de Android*. México, DF: Alfaomega Grupo Editor, 2013, pp. 349-359.
8. Amaro Soriano, J. E. *Android: programación de dispositivos móviles a través de ejemplos*. Madrid: Alfaomega Grupo Editor, 2012, pp. 70-120.

Acerca de los autores

Arturo Juárez Ríos es Ingeniero en Comunicaciones y Electrónica, especialidad en Comunicaciones, por la Escuela Superior de Ingeniería Mecánica y Eléctrica del Instituto Politécnico Nacional, desde 1992. Actualmente estudia la Maestría en Tecnología de Cómputo en el Centro de Innovación y Desarrollo Tecnológico en Cómputo del Instituto Politécnico Nacional; ha sido autor de artículos relacionados con el control y el sondeo mediante comunicación inalámbrica y ponente en un congreso internacional. Sus áreas de interés son las aplicaciones en dispositivos de cómputo móviles y los sistemas embebidos de bajo costo.

Juan Carlos Herrera Lozada es Ingeniero en Comunicaciones y Electrónica por la Escuela Superior de Ingeniería Mecánica y Eléctrica del Instituto Politécnico Nacional, desde 1996. En 2002 obtuvo el grado de Maestro en Ingeniería de Cómputo con especialidad en Sistemas Digitales y en 2011 obtuvo el grado de Doctor en Ciencias de la Computación, ambos en el Centro de Investigación en Computación del Instituto Politécnico Nacional. Desde 1998 está adscrito al Centro de Innovación y Desarrollo Tecnológico en Cómputo del Instituto Politécnico Nacional; ha sido autor de diversos artículos y ponente en congresos nacionales e internacionales; ha dirigido diversos proyectos de investigación y desarrollo tecnológico, ha registrado una patente y es miembro del Sistema Nacional de Investigadores. Sus áreas de interés son los sistemas embebidos y el diseño con lógica reconfigurable.

Hind Taud es física de formación, obtuvo el grado de Maestra en Electrónica en 1987 y el doctorado "*3ème cycle*" en Ciencias Físicas en 1989 por la Facultad de Ciencias de Rabat, en Marruecos. Realizó una estancia posdoctoral y de investigación en la Universidad París VI en Francia, donde obtuvo el grado de Doctora en Ciencias de la Tierra en 1993. Actualmente labora en el Centro de Innovación y Desarrollo Tecnológico en Cómputo del IPN. Su trayectoria de investigación está relacionada con el procesamiento, análisis y reconocimiento de patrones en imágenes digitales con diversas aplicaciones y en diferentes campos.



Jesús Antonio Álvarez Cedillo es Ingeniero en Comunicaciones y Electrónica por la Escuela Superior de Ingeniería Mecánica y Eléctrica del Instituto Politécnico Nacional, desde 1997. Obtuvo el grado de Maestro en Ciencias de la Informática por la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas del Instituto Politécnico Nacional en 2002. Desde 2007 es candidato a Doctor en Tecnologías Avanzadas en el Centro de Investigación e Innovación Tecnológica del Instituto Politécnico Nacional. Ha ejercido como profesor e investigador de tiempo completo del Centro de Innovación y Desarrollo Tecnológico en Cómputo desde 2000 y ha publicado diversos artículos relacionados con arquitectura de computadoras, procesamiento paralelo y algoritmos de alto desempeño.

Jacobo Sandoval Gutiérrez es Ingeniero en Robótica Industrial por la Escuela Superior de Ingeniería Mecánica y Eléctrica del Instituto Politécnico Nacional, desde 2004. Obtuvo los grados de Maestro y Doctor en Tecnología Avanzada en el Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada del Instituto Politécnico Nacional en 2006 y 2010, respectivamente. Actualmente participa como profesor invitado en el Centro de Innovación y Desarrollo Tecnológico en Cómputo del Instituto Politécnico Nacional. Ha publicado diversos artículos relacionados con el campo de la robótica, asimismo, ha registrado patentes.