Robot móvil tipo oruga para detección de minas terrestres metálicas.

Barrios Santiago Miguel A.
Venegas Monroy Miguel E.

Instituto Tecnológico de Ensenada.
Boulevard tecnológico #150, colonia ex–ejido Chapultepec.
Ensenada, B.C., C.P. 22780. Teléfono 646 1775680.
Robótica móvil.
E-mail: miguelvmonroy@gmail.com, ingsardico1@gmail.com

Resumen

En la actualidad el desarrollo de robots móviles ha sufrido un crecimiento exponencial debido a las soluciones que proporciona en la industria principalmente. Dentro de esta área de la robótica los robots basados en cintas de deslizamiento (oruga) representa una elección adecuada para entornos no preparados, teniendo de esta manera un campo de investigación mayor en áreas enfocadas en seguridad y exploración.

Aprovechando las ventajas para este tipo de entornos, aplicaciones de alto riesgo como lo son la detección de minas terrestres son consideradas ideales para estos sistemas ya sea como sistemas autónomos o guiados.

1. Introducción

El uso de robots se ha incrementado con el paso de los últimos años; los robots móviles han surgido de la necesidad de hacer de estos dispositivos electromecánicos autónomos, es decir, mediante la programación y el control por software sean capaces de realizar tareas específicas. Este tipo de robots se clasifican según su entorno de trabajo (aéreo, terrestre, marítimo, anfibio) sin embargo los robots terrestres siguen liderando debido a su alta demanda y capacidades de trabajo continuo en industrias.

Hoy en día aun con el uso de software de alto nivel, así como controladores complejos resulta un reto el realizar un sistema autónomo, esto debido a la cantidad inmensa de variables que existen en el ambiente y sucesos para los cuales estos sistemas no puedan reaccionar. (Aguilera Hernandez, Bautista, & Iruegas, 2007) El problema básico de un robot móvil es el de navegación, es decir, moverse de un lugar a otro auxiliado de sensores, planeación y control. El problema de navegación es simplemente encontrar una trayectoria desde un inicio a un objetivo.

El uso de robots móviles no solo se limita a la industria, si bien es el área de mayor demanda e inversión, otras áreas como la investigación, exploración o seguridad comienzan a apostar por sistemas autónomos o guiados los cuales permitan explotar las ventajas de tener un sistema con capacidades de trabajo continúo reduciendo los riesgos posibles al mínimo. La automatización completa representa una solución idónea en términos de seguridad y eficiencia, pero tiende a producir sistemas complejos, a veces ineficientes cuando se trata de cubrir todo el espectro de casos posibles. Una solución intermedia consiste en utilizar al operario en el lazo de control utilizando técnicas de tele operación y control "colaborativo" que ayude a resolver aquellos problemas para los que el subsistema automático no está preparado (Ponticelli Lima, 2011).

2. Objetivos

2.1 Objetivo general.

- Desarrollar un sistema robótico móvil basado en locomoción por bandas deslizantes (oruga).

2.2 Objetivos específicos.

- Crear un sistema capaz de responder a comandos programados para trazar trayectorias específicas.
- Permitir que el sistema creado pueda responder ante un obstáculo imprevisto en una trayectoria definida y posteriormente lograr su meta.

- Realizar la detección de elementos metálicos haciendo alusión a minas terrestres y que a su vez el robot responda ante estos elementos.

3. Marco teórico

3.1 Robots móviles y locomoción.

Un robot móvil puede ser definido como un sistema o dispositivo de transporte automático, es decir, una plataforma mecánica dotada de un sistema de locomoción capaz de navegar a través de un determinado ambiente de trabajo, dotado de cierto nivel de autonomía (programado) para su desplazamiento portando cargas.

Los robots móviles se clasifican en guiados y no guiados. El vehículo guiado está restringido a un conjunto de trayectorias predefinidas en su área de trabajo. Estas trayectorias están indicadas por líneas ópticas, magnéticas o una secuencia de movimientos guardados en la memoria. El robot en ningún momento puede abandonar la trayectoria. A estos robots se les llama también vehículos automáticos guiados. Los vehículos no-guiados no están restringidos a una trayectoria predefinida. A este tipo de robots no-guiados pertenecen los robots submarinos (bajo el agua), los del espacio aéreo y/o espacial y los terrestres. Los robots terrestres han sido clasificados de acuerdo a su locomoción (por su sistema de movimiento) en robot con ruedas, con patas o con rieles. Así cada robot desarrolla diferentes tareas de acuerdo a su capacidad [1].

La locomoción mediante ruedas permite el diseño de vehículos con estructura mecánica relativamente simple, donde cada rueda consiste básicamente en un disco acoplado a una articulación rotatoria. La configuración de las ruedas (número de ruedas, sus diámetros, anchos y patrones de la superficie de rodadura, ruedas en configuración “oruga”, etc.) influye directamente en la maniobrabilidad que ofrece el vehículo y que afecta su capacidad de evadir obstáculos durante el desplazamiento [2].

3.2 Diseño y modelado de robots móviles tipo oruga.

El uso de un sistema de locomoción distinto a las ruedas representa una elección interesante; este implica el uso de pistas o cintas las cuales realizan un deslizamiento en el terreno permitiendo un mayor contacto entre el robot y su área de trabajo. El diseño de robots oruga se presenta como un sistema intermedio teniendo menor complejidad que el de patas y menos simplicidad que el de ruedas.

Las principales ventajas de una locomoción basada en pistas de deslizamiento u orugas son consideradas a continuación (Gonzalez , Rodriguez, & Guzman, 2015):

- Es una solución muy adecuada para una plataforma versátil que opere sobre diversos tipos de terrenos e incluso en diferentes condiciones meteorológicas (barro, nieve, etc.). Las aplicaciones de búsqueda y rescate son un buen ejemplo para sistemas basados en orugas.
- Las orugas generan una baja presión en el suelo, incluso ante una elevada carga, lo que lleva a conservar mejor el terreno y el entorno en el que opera el vehículo. Esto motiva la elevada aplicación en agricultura y en actividades mineras
- Se previene la posibilidad de que el vehículo quede estancado en arena o nieve; por ello, los vehículos con orugas son muy utilizados en terrenos arenosos como en operaciones militares (transporte de artillería o personal).
- El centro de gravedad del vehículo se mantiene bajo, permitiendo conseguir una buena estabilidad y movilidad.

A pesar de la flexibilidad que otorga este tipo de robot móvil a su adaptabilidad al entorno, tiene grandes dificultades, algunas relacionadas con características propias del suelo (cohesión, densidad, módulo de deformación, etc.) y otros con fenómenos como el deslizamiento (slip) y el hundimiento (sinkage). De esta manera lo que resulta en su mayor ventaja (adaptabilidad a diferentes terrenos) también es uno de sus mayores retos (slip y sinkage), por ello se simplifica en modelos más simples para su posterior estudio matemático. Basados en el modelo

de (Gonzalez , Rodriguez, & Guzman, 2015) en donde se utiliza un robot móvil con orugas desplazándolo a velocidades bajas, se asume que su deslizamiento lateral (problema muy común debido al área de contacto con el terreno) es muy pequeño; de esta manera se aproxima a un modelo basado en un mecanismo diferencial pero con un efecto de deslizamiento longitudinal. Las expresiones matemáticas se muestran a continuación:

$$\dot{x}^{sl}(t) = \frac{v_r(t)(1 - i_r(t)) + v_l(t)(1 - i_l(t))}{2}\cos\ ^{sl}(t)$$

$$\dot{y}^{sl}(t) = \frac{v_r(t)(1 - i_r(t)) + v_l(t)(1 - i_l(t))}{2}\sin\ ^{sl}(t)$$

$$\dot{}^{sl}(t) = \frac{v_r(t)(1 - i_r(t)) - v_l(t)(1 - i_l(t))}{b}$$

En donde $[x^{sl} \quad y^{sl} \quad {}^{sl}]^T \quad \mathbb{R}^3$ representa la posición del robot móvil, ***Vr, Vl*** son las velocidades lineales de las orugas derecha e izquierda respectivamente, ***b*** es el ancho del robot e ***ir, il*** es el deslizamiento de las orugas derecha e izquierda, respectivamente.

$$\begin{bmatrix} e_x(t) \\ e_y(t) \\ e\ (t) \end{bmatrix} = \begin{bmatrix} \cos({}^{sl}(t)) & \sin({}^{sl}(t)) & 0 \\ \sin({}^{sl}(t)) & \cos({}^{sl}(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^{rf}(t) - x^{sl}(t) \\ y^{rf}(t) - y^{sl}(t) \\ {}^{rf}(t) - {}^{sl}(t) \end{bmatrix}$$

Reformulando se puede obtener que $e_x \ e_y \ e$ son los errores longitudinal, lateral y en orientación, respectivamente, y $[x^{rf} \quad y^{rf} \quad {}^{rf}]^T$ es la posición del robot de referencia.

3.3 Planificación de trayectorias.

Se reconoce a la planeación de trayectorias como la búsqueda de una sucesión de posiciones de un robot que permiten llevarlo de un estado inicial a uno final, entendiéndose como estado a la descripción de la ubicación del robot referenciada a un marco absoluto generalmente expresada por la combinación de las coordenadas cartesianas del centro del robot y la posición angular del eje principal de éste. La configuración que adquiere la trayectoria se define por la distribución de los obstáculos a lo largo de todo el ambiente de trabajo y por supuesto de la

geometría del robot, así como de sus capacidades del movimiento (Mariscal Garcia, 2005).

Dentro de la literatura existen gran variedad de algoritmos para realizar la planificación de rutas, la mayor parte pueden ser agrupados en 3 tipos que se presentan a continuación (Gonzalez Sieira, 2011):

- Técnicas geométricas: Esta aproximación está basada en la diferenciación de las zonas libres del entorno y las ocupadas, manteniendo una representación del entorno que permita el cálculo de rutas en las regiones etiquetadas como libres. Estos algoritmos son completos, en el sentido que son capaces de conectar dos puntos cualesquiera del espacio libre, siempre que se cumplan dos condiciones:
    - Accesibilidad, dos puntos cualesquiera del espacio libre están conectados entre sí por una ruta libre de obstáculos.
    - Conservación de la conectividad, es posible representar el espacio libre de obstáculos mediante un grafo donde todas las rutas entre puntos del espacio libre estén presentes como conexiones entre nodos del grafo.

  Uno de los problemas más relevantes de esta aproximación es que cuando el entorno tiene un gran número de obstáculos su representación requiere una gran capacidad computacional, y puede llegar a ser imposible encontrar una solución al problema en un tiempo razonable.

- Campos de potencial: Los campos de potencial se utilizan para decidir cuál es la dirección más prometedora en cada instante para alcanzar la meta. Para conseguirlo, utilizan la teoría de los campos eléctricos y el robot, la meta y los obstáculos del entorno tienen asociado un potencial. Lo que se pretende conseguir es que el robot sea atraído por la meta, a la vez que es repelido por los obstáculos.

  Una de las principales ventajas de estos métodos es que consiguen resolver problemas en entornos simples utilizando muy pocos recursos computacionales. Por contrapartida, no son métodos muy recomendables para

abordar la resolución de problemas complejos, puesto que siguen una estrategia greedy (estrategia de búsqueda consistente en elegir la opción optima general): en cada momento se estudia la influencia de los potenciales emitidos por los diferentes objetos del entorno, para decidir cuál es la dirección de avance más prometedora. Esto hace que el método no sea robusto ante posibles mínimos locales, siendo necesaria una estrategia de más alto nivel para evitarlos

- Algoritmos basados en muestreo: Estos tratan de resolver el problema de la planificación de movimientos evitando la descripción explícita de las regiones ocupadas por obstáculos, tratando de representar el espacio a partir de un esquema de discretización. Esta familia de algoritmos es completamente independiente del problema de detección de colisiones y, por tanto, también del modelo geométrico utilizado para describir el entorno del robot.
  Este tipo de aproximación se puede considerar como la más exitosa de todas las que se encuentran en el estado del arte de la planificación de movimientos, y es ampliamente utilizada en robótica móvil, problemas de manufactura, y aplicaciones biológicas con muy alta dimensionalidad, que serían intratables desde las perspectivas anteriores y que requieren una representación completa del espacio de obstáculos (…). Una ventaja importante de esta aproximación respecto a las otras es que es posible introducir las limitaciones del modelo cinemático del robot en el proceso de planificación, definiendo además de un espacio de estados, un espacio de acciones realizables.

3.3.1 Diagramas de Voronoi.

Los diagramas de Voronoi son definidos por (Janich, 1984) como una retracción con preservación de la continuidad. Si el conjunto ***Ct*** define las posiciones libres de obstáculos de un entono, la función de retracción ***RT*** construye un subconjunto ***Cv*** continuo de ***Ct***:

$$RT(q): C_l \rightarrow \frac{C_v}{C_v} \subset C_l$$

Por lo tanto, se asume que existe un camino desde una posición inicial ***Qa*** hasta una posición final ***Qf***, libre de obstáculos, si y solo si existe una curva continua desde *RT(Qa)* hasta *RT(Qf)*.

Por ello, el diagrama de Voronoi resulta el lugar geométrico de las configuraciones que se encuentran a igual distancia de los 2 obstáculos más próximos del entorno. El diagrama está formado por elementos rectilíneos y parabólicos (…). De esta forma, el lugar geométrico de las configuraciones que se hallan a igual distancia de dos aristas de dos obstáculos diferentes en una línea recta, mientras que en el caso de tratarse de un vértice y una arista resulta una parábola (Batorune, 1995).

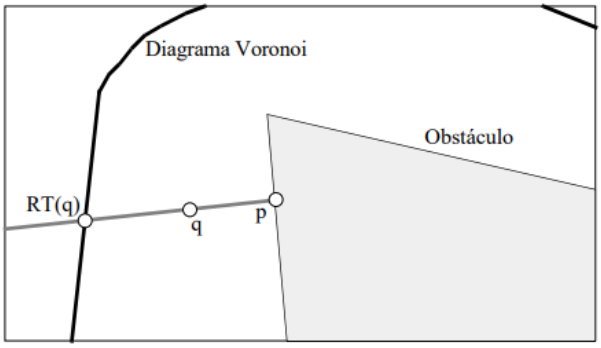

Figura 1. Configuración Q en el diagrama de Voronoi

3.4 Herramientas de procesamiento y control.

El procesar datos, consiste en someter estos a un conjunto de operaciones tales como, ordenación, selección, ejecución, cálculos, etc.; de forma que nos permita extraer conclusiones de los datos que manipulamos. De acuerdo con esta definición podemos deducir cierto tipo de operaciones que un procesador debe realizar como lo son (Bishop, 1992):

- Entrada de datos: Suministrar información desde su entorno exterior
- Salida de datos: Obtener la información de un computador
- Almacenamiento: Hacer copia permanente de la información
- Recuperación: Leer de nuevo la información almacenada
- Transmisión: Transferir información a otro computador

- Recepción: Recibir información desde otro computador
- Tratamiento: Operaciones de datos como selección, combinación, ejecución, cálculos, etc.

3.4.1 Arduino mega

El Arduino Mega 2560 es una placa de microcontrolador basado en el ATmega2560. Tiene 54 pines digitales de entrada/salida (de los cuales 14 se pueden usar como salidas PWM), 16 entradas analógicas, 4 UART (puertos serie de hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP, y un botón de reinicio.

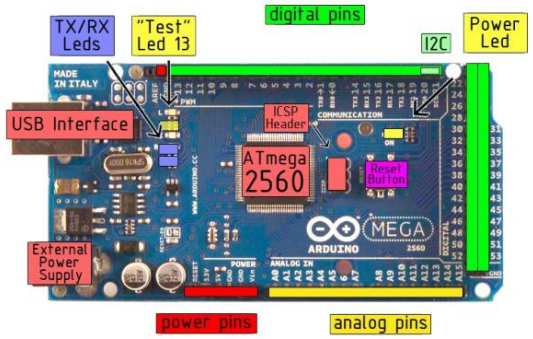

Figura 2. Placa Arduino Mega2560

Cada uno de los 54 pines digitales del Mega se puede usar como entrada o salida, usando la programación adecuada establecida por su IDE y funciones de lectura digital. Estas operan a 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia interna de pull-up (desconectada por defecto) de 20-50 kOhms. Además, algunos pines tienen funciones especializadas como comunicación RX y TX, entre otras.

El Mega2560 tiene 16 entradas analógicas, cada una de las cuales proporciona 10 bits de resolución (es decir, 1024 valores diferentes). Por defecto miden desde tierra hasta 5 voltios, aunque es posible cambiar el extremo superior de su rango usando el pin AREF y programación respectiva.

3.5 Sistemas de detección de minas terrestres

Las herramientas y procedimientos disponibles actualmente para las tareas de detección de minas antipersonas se asemejan a aquellas empleadas en la Segunda Guerra Mundial: un equipo de operarios provistos con un detector de metales y una herramienta para "pinchar" el terreno (varilla, destornillador, etc.). Primero se limpia el terreno de vegetación densa y después se divide en celdas de 1 × 1 m aproximadamente. Un operario procede entonces avanzando lentamente por cada celda, moviendo el detector de metal cerca de la superficie. Cuando se detecta una alarma, un segundo operario procede a determinar con la varilla, pinchando el terreno, si la alarma corresponde a una mina enterrada (…). Actualmente, ninguna tecnología de detección puede operar efectivamente contra todo tipo de minas o en todo tipo de condiciones ambientales. Por ejemplo, sensores por resonancia de cuadrupolo nuclear (Nuclear Quadrupole Resonance) pueden encontrar rápidamente minas que contienen explosivo de tipo cyclotrimethylenenitramine (conocido como Royal Demolition Explosive RDX), pero resultan lentos confirmando la presencia de explosivo de tipo trinitrotolueno (TNT). Los sistemas de detección acústica demuestran tener una relación baja de falsos positivos, pero no pueden detectar minas enterradas a mayor profundidad que unos pocos centímetros. Los sensores de vapores químicos son capaces de detectar minas plásticas en terrenos húmedos, pero tienen dificultades en ambientes secos, y en general cada tipo de sensor, como el Radar Penetrante en Tierra (GPR), sensores químicos, narices artificiales, presentan cada uno ventajas muy específicas. Dadas las limitaciones individuales de cada sistema sensor y cada tecnología actual, se hace necesario desarrollar sistemas multi-sensoriales que combinen las propiedades de distintas tecnologías (Ponticelli Lima, 2011).

4. Desarrollo y resultados

4.1 Estructura

La base del robot móvil oruga, se encuentra basado en una plataforma de aluminio anodizado resistente a la corrosión diseñado por OSEPP.

La utilización de este tipo de material permite trabajar en ambientes con condiciones más exigentes sin afectar en gran medida a la estructura. En la figura 3 se muestra el ensamble parcial para cada uno de los ejes.

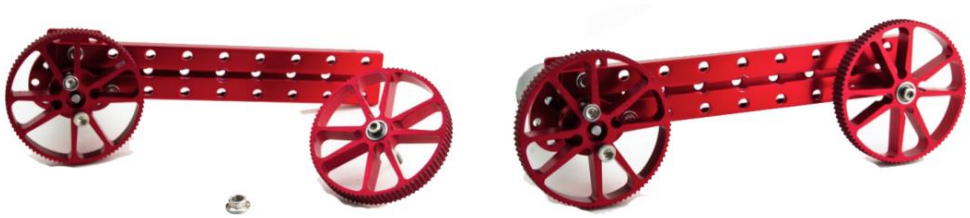

Figura 3. Ensamble de la estructura para cada eje lateral

4.2 Motores y etapa de potencia.

En la figura 4 se presenta un esquema donde se describen las mediciones de los motores utilizados pertenecientes a Dongshun Motors. Estos motores cuentan con una transmisión reductora 1:45 la cual otorga una velocidad (sin carga) de aproximadamente 188 RPM y un torque de 1.5 Kg*cm.; esto para una alimentación de 9 V.

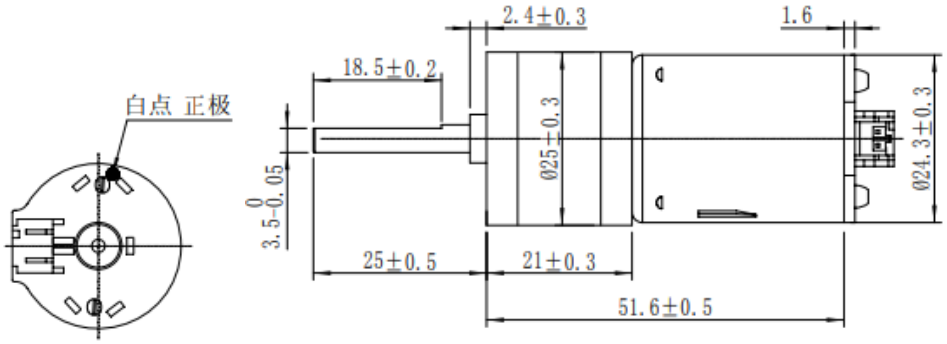

Figura 4. Mediciones para la vista superior y lateral de los motores

La corriente de consumo para estos motores considerando ya el acoplamiento con su respectiva transmisión es de 220 mili Amperes, teniendo el motor sin carga. En la figura 5 se muestran los motores acoplados a los ejes laterales que conforman la estructura. En base a la carga total del robot se estimó un consumo de corriente por motor de 450 mili Amperes máximo.

Figura 5. Motores ensamblados a los ejes laterales

Con la estimación en corriente para el motor con carga se procedió con la etapa de potencia, para esto se escogió el módulo L298N que se muestra en la figura 6. Este módulo permite un consumo de corriente de hasta 2 Amperes por canal y una alimentación de 35V máx., para los motores, así como 12V máx., para el modulo en si, por lo que su alimentación por medio de la placa de control Arduino es posible, sin embargo, su respectiva alimentación se obtiene de una etapa de regulación desde la batería, su placa de conexión se puede observar en la figura 6.

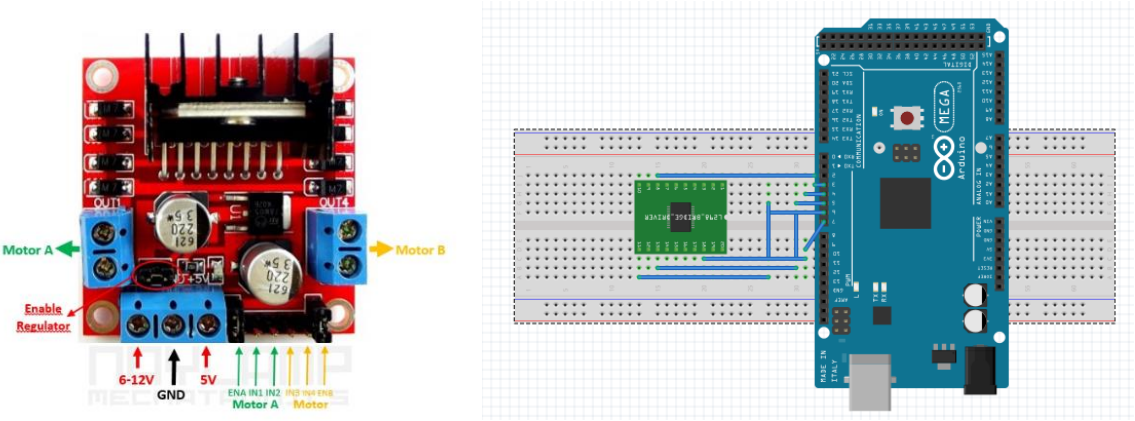

Figura 6. a) Esquema del módulo L298N. b) Conexiones de control Arduino Mega a módulo L298N usando Fritzing.

La etapa de alimentación para los distintos sub sistemas puede ser dividida en 3 secciones como se muestra en el siguiente diagrama.

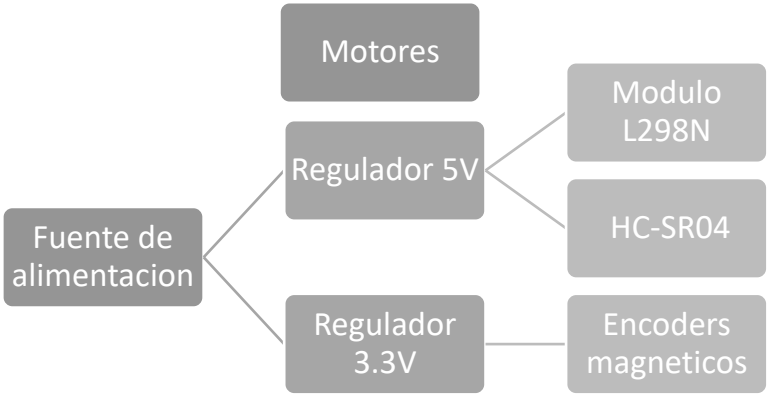

Diagrama 1. Etapas de regulación en el robot

- Fuente de alimentación.

La alimentación para el robot móvil tipo oruga fue seleccionada considerando una las desventajas de este tipo de sistemas, su consumo elevado de energía. El uso de una batería tipo Li.-Po fue la opción más adecuada dada sus características para el suministro de energía constante y descarga lenta.

Se utilizó una batería Li-Po Gforce de 2200 mili amperes a 11.1 V, así como una descarga de 30C.

Figura 7. Batería Li-Po Gforce.

- Regulador 5V.

Esta etapa consta de un circuito integrado 7805, regulador de voltaje; ofrece un voltaje constante a 5v con una alimentación de hasta 25v y un consumo de corriente de salida de hasta 1 A. El circuito utilizado se muestre en la figura 8.

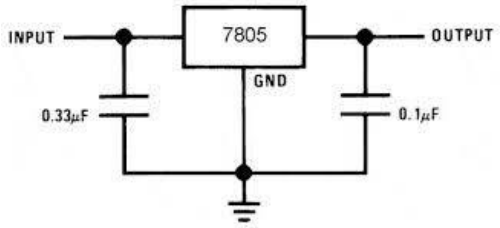

Figura 8. Circuito regulador a 5v con C.I 7805.

- Regulador 3.3V.

Esta etapa se ve regulada por una placa comercial basado en el C.I regulador de voltaje AMS 1117; este regulador permite un consumo de corriente de hasta 1.5 A, sin embargo, su alimentación se limita a una entrada de 12V máx. Su alimentación se realizó desde la etapa regulador de 5 V, dado que el consumo contemplado de corriente no es elevado para ninguna de ambas etapas. El esquema utilizado en la placa, así como la misma se puede apreciar en la figura 9.

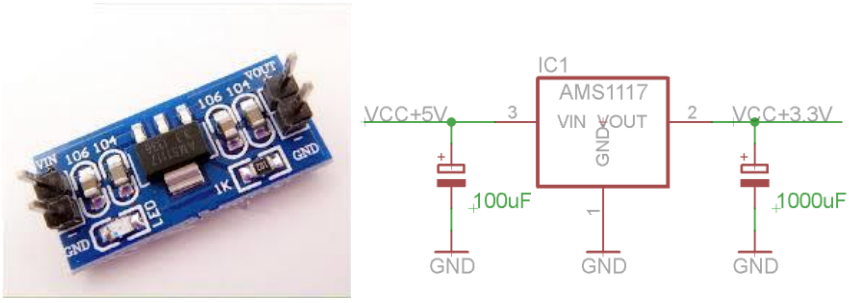

Figura 9. a) Placa reguladora 3.3 V. b) Diagrama de electrónico de la placa basada en el C.I. AMS1117

4.3 Procesamiento y control del robot.

La unidad de procesamiento utilizada para el robot móvil se basó en la plataforma Arduino, en específico el modelo Mega 2560, esto debido a su cantidad considerable de pines analógicos, así como digitales los cuales fueron necesarios para el control de los motores, así como lectura de los encoders y la detección de obstáculos mediante el modulo ultrasónico.

- Encoders.

  El robot oruga cuenta con un encoder magnético genérico para cada motor (dos), estos se basan en el modelo de Pololu de 12 pulsos por revolución, ofreciendo 2 canales en cuadratura, así como 11 pulsos por revolución.

  Los pulsos en cuadratura que ofrecen estos encoders representan valores lógicos por lo que su uso directo a la placa de control fue posible, omitiendo de esta forma módulos conversores o programación adicional al código final.

  El control realizado en el robot se vio en su mayor parte influenciado por la lectura de los encoders, estos permitieron conocer la distancia que recorría el robot al activar los motores; mediante un algoritmo programado en el Arduino se lograron desarrollar direcciones, tomando las cuentas por revolución de los encoders. En la figura 10 se muestran los encoders utilizados.

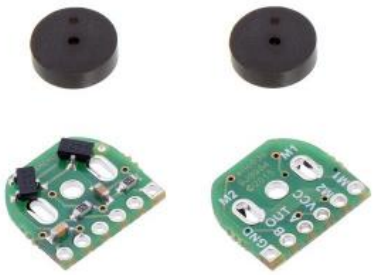

Figura 10. Imán y modulo magnético del encoder.

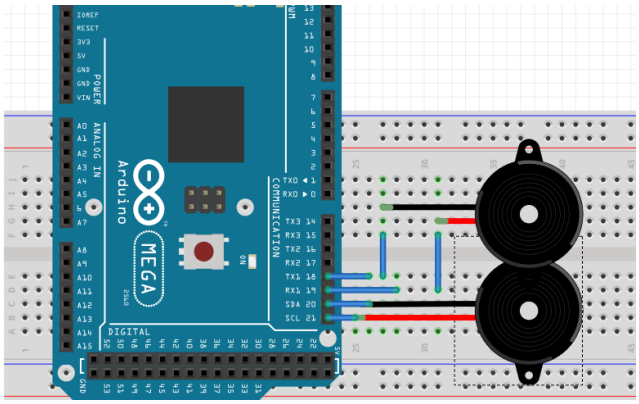

Figura 11. Conexiones digitales de los encoders (sin alimentación).

La conexión para la instalación de los encoders se aprecia en la figura 11. Los encoders requieren de 2 pines digitales cada uno, esto debido a que se requieren activar una serie de interrupciones en el Arduino para realizar el conteo según avance o retroceda el robot.

- Modulo ultrasónico.

  El sensor HC-SR04 es comercialmente uno de los sensores más utilizados dada su facilidad de implementación, así como sus características de detección de objetos a distancias considerables. El uso de este dispositivo en el robot oruga permite la detección a una distancia deseada no mayor a 4 metros en un tiempo de respuesta relativamente bueno. La respuesta de este sensor depende de la distancia a la cual se desee detectar el objeto ya que se basa en el tiempo de vuelo de las ondas para arrojar un valor y posteriormente lo pueda procesar el Arduino, en la figura 12 se puede apreciar el modulo ultrasónico.

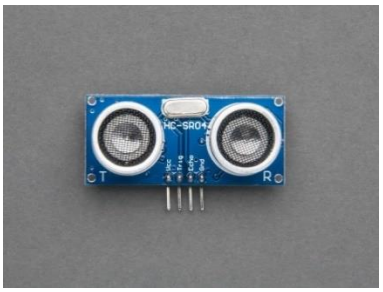

Figura 12. Módulo HC-SR04

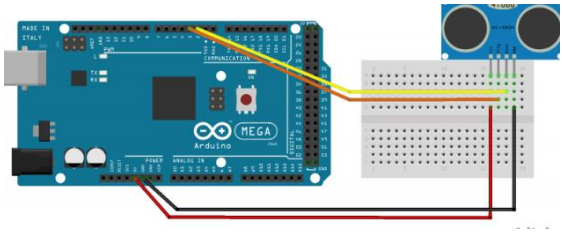

Figura 13. Conexiones a pines digitales del módulo ultrasónico.

En la figura 14 se ilustra la adquisición de los datos en tiempo real. El monitor serie de Arduino nos permite observar los valores arrojados por los módulos encoder, así como la distancia medida por el sensor ultrasónico.

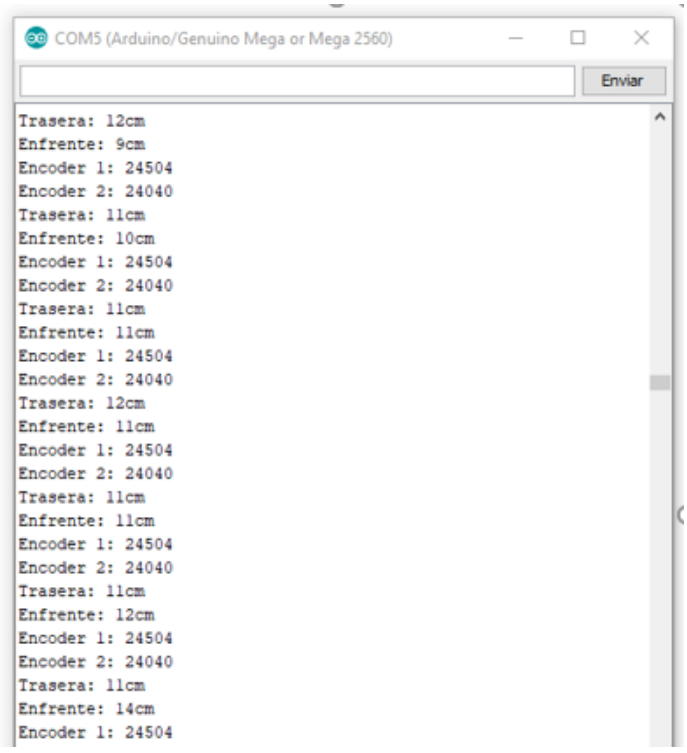

Figura 14. Datos adquiridos por los sensores.

4.4 Trayectorias y evasión de obstáculos.

El desarrollo de trayectorias en el robot oruga se realizó en su totalidad mediante el uso de los datos adquiridos por los encoders. Las consideraciones se realizaron respecto a las cuentas por revolución, así como el uso del PWM en el Arduino; al realizar una relación para ambas es posible determinar una conversión de cuentas a centímetros (considerando una caracterización previa de los encoders), así como una determinada velocidad (en términos del PWM).

La estructura general a considerar para el desarrollo de las trayectorias se presenta en el siguiente diagrama.

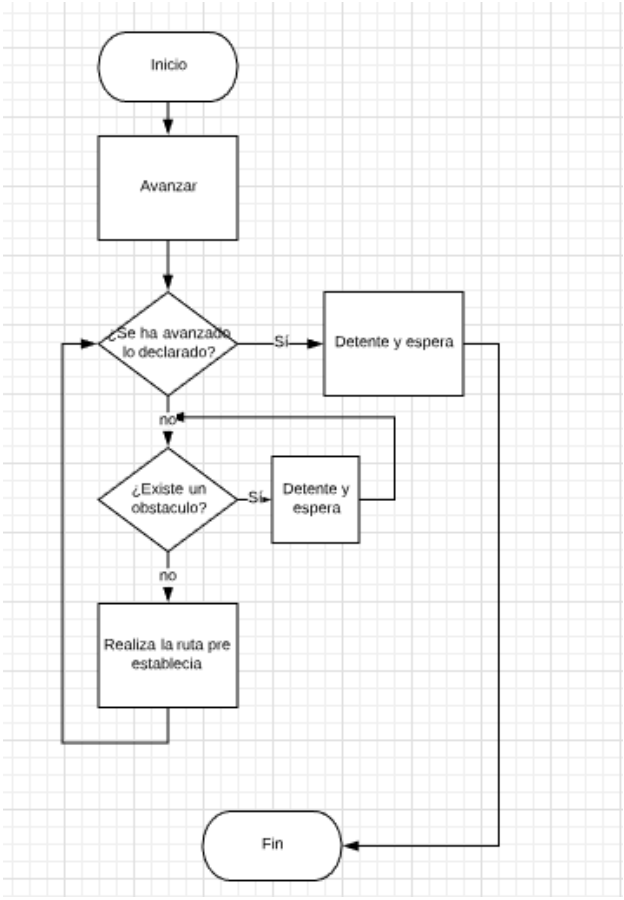

Diagrama 2. Desplazamiento general del robot

- Trayectorias.

El mapa, así como sus respectivos objetivos se puede apreciar en la figura 15. En ella se muestra un punto de inicio, así como su meta para 3 puntos distintos.

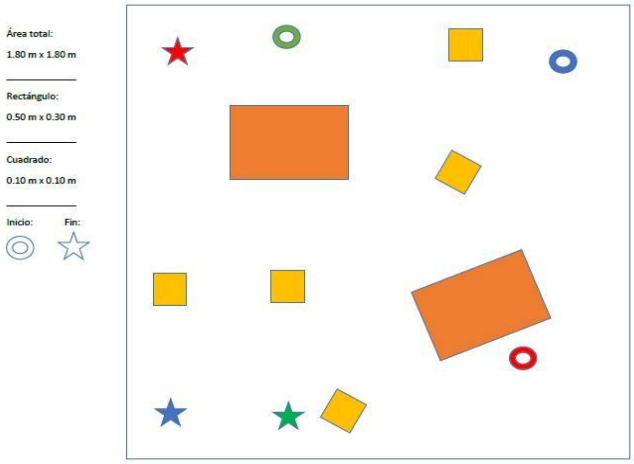

Figura 15. Mapa para pruebas del robot móvil.

El cálculo de estas trayectorias distintas se realizó mediante diagramas de Voronoi, obteniendo las resultantes más optimas y que se muestran en las figuras 16, 17 y 18 respectivamente.

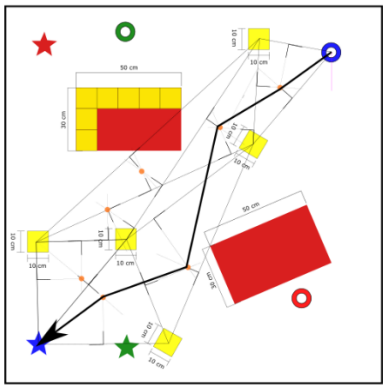

Figura 16. Trayectoria azul por diagramas de Voronoi

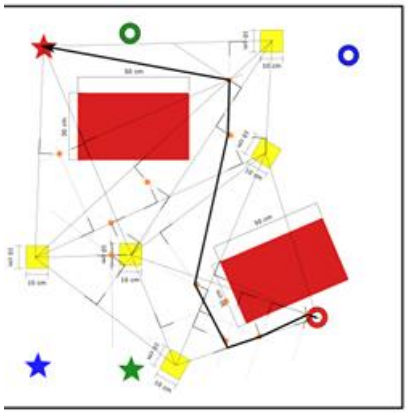

Figura 17. Trayectoria roja por diagramas de Voronoi

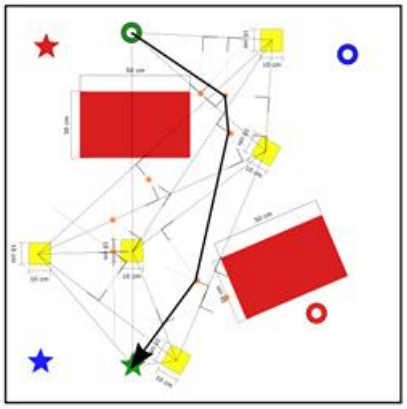

Figura 18. Trayectoria verde por diagramas de Voronoi

Dentro de la programación de Arduino, cada sub-trayectoria (recta en los diagramas de Voronoi) se basó en una sola subrutina que descrita de forma general se presenta como la siguiente

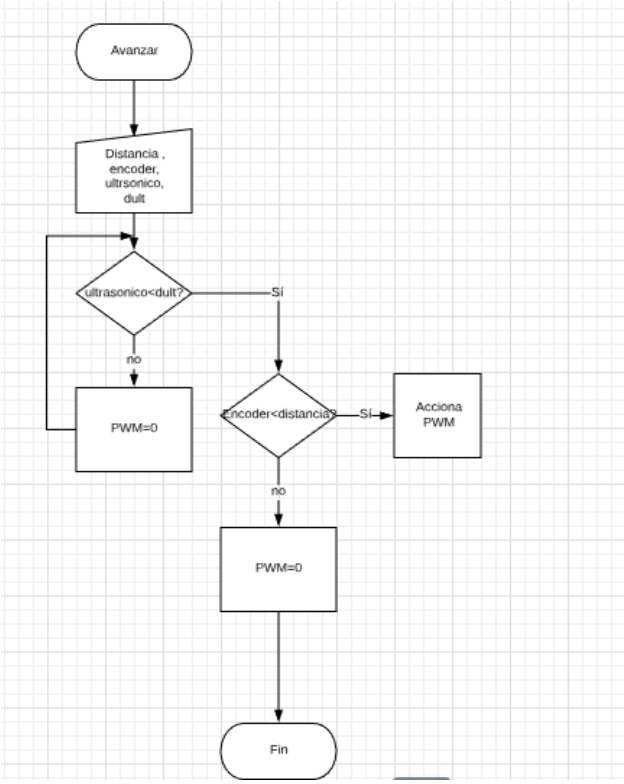

Diagrama 3. Función avanzar con lectura de encoders y ultrasónico.

Al haberse logrado la relación de distancia con velocidad de los motores el código expresado en Arduino se describe de la siguiente manera:

```
void doEncodeA()
{
      if (digitalRead(channelPinA) == digitalRead(channelPinB))
      {
         IsCW = true;
         if (ISRCounter + 1 <= maxSteps) ISRCounter++;
      }
      else
      {
         IsCW = false;
         if (ISRCounter - 1 > minSteps) ISRCounter--;
      }
}

// funcion que hace que el carro avance
void avanzar(int vel, float dist)
```

```
while (abs(ISRCounter2)<abs(distl))
{
 obs = sensorfrontal.Ranging(CM);
 if (obs>dl)
 {
    mi_pwml(motorl_PWM,pwm_ideal_1);
    mi_pwm2(motor2_PWM,pwm_ideal_2);
  }
 else
 {
    mi_pwml(motorl_PWM,0); // 0 - 255
    mi_pwm2(motor2_PWM,0);
```

Como se observa en el código, mediante la creación de subrutinas se leen los valores obtenidos de los encoders; al ser encoders en cuadratura se contempla el conteo en subida y bajada, esto con la finalidad de obtener un conteo positivo o negativo según giren los motores. Los valores son guardados en una variable contadora al cual posteriormente se utiliza para asignar las distancias que deben ser recorridas por cada motor.

- Evasión de obstáculos.

Al ser un sistema en lazo abierto, el robot oruga desconoce su posición en el plano, por esto una evasión de obstáculos como tal resulta altamente compleja. Para lograr una evasión parcial se consideraron las trayectorias dentro de una decisión anidada en la cual se introducen los valores medidos del sensor ultrasónico y que a su vez son comparados con una distancia dada como se alcanza a observar en el diagrama 3. Esto permite resolver de manera limitada el choque con obstáculos imprevistos, logrando que el robot reaccione y al retirar dicho obstáculo continúe con su ruta hacia su objetivo según la trayectoria.

Una resolución optima al problema de la evasión de obstáculos, seria cerrar el lazo mediante un método de posicionamiento para el robot; con eso el robot conocería en cada momento su ubicación exacta en el

plano y le permitiría desplazarse con mayor precisión y facilidad. La triangulación mediante sonidos a distintas frecuencias se propone como una medida interesante y controlada en la cual el robot mediante un micrófono tiene la capacidad de lograr determinar su distancia de cada objeto por la variación en frecuencias y con esto lograr una evasión completa de obstáculos imprevistos aun tomando en cuenta los obstáculos fijos.

4.5 Aplicación.

El uso de un sensor capaz de detectar a distancia objetos metálicos fue requerido, para esto se procedió a implementar un sensor magnético de uso simple como lo es el reed switch; esto con la finalidad de simula la detección de minas terrestres.

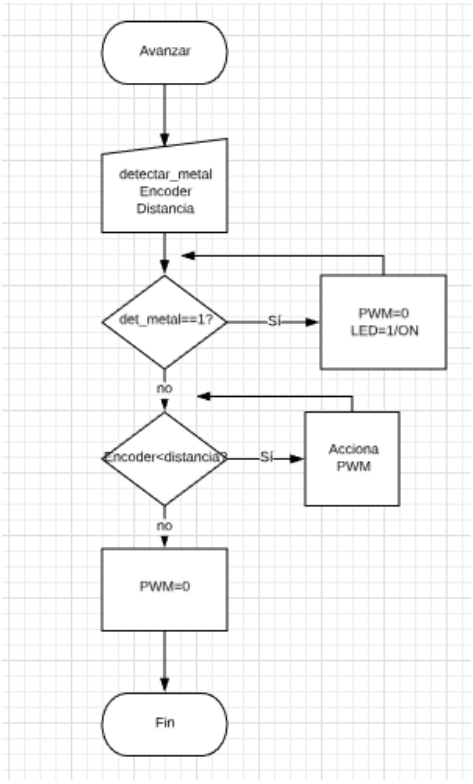

Diagrama 4. Algoritmo para la detección de objetos metálicos.

El principio que sigue es el mismo que el del sensor ultrasónico, es decir, al detectar un objeto metálico detiene su movimiento accionando una luz intermitente. El objeto en si se ha declarado como un obstáculo, siguiendo la misma lógica que con el ultrasónico, esto permite facilitar su implementación y adaptabilidad para futuras mejoras que se deseen incluir en el prototipo. Las conexiones básicas se observan en la figura 19, cabe señalar que al ser un sensor del tipo switch, su conexión se puede realizar a los pines digitales sin la necesidad de la conversión de valores mediante código y facilitando la activación de interrupciones en la placa Arduino para el posterior llamado de sub rutinas.

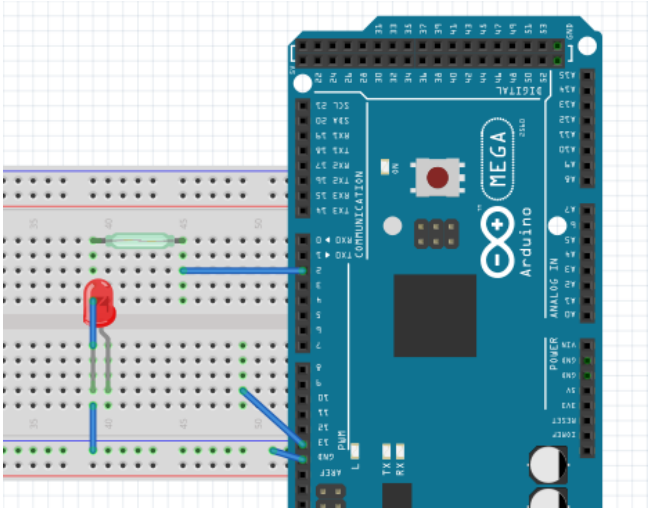

Figura 19. Conexiones a los pines del Arduino para el sensor magnético.

El sensor magnético presenta un campo de acción demasiado reducido; esto sumado a la limitada alimentación del robot supone un gran reto ya que gran parte de los sensores utilizados en la detección de minas terrestres en la actualidad, se encuentran basados en arreglos de sensores (como lo son los radares), los cuales hacen uso de sus características individuales para generar una mayor potencia y directividad o en su defecto cubrir desventajas individuales de los mismos.

4.6 Costos

Elaboración de costos		
Articulo	*Costo*	*Cantidad*
Estructura tanque OSEPP	$89.95	1
Batería Li-Po Gforce	Reutilizado	1
Regulador 5V	Reutilizado	1
Regulador 3.3V	Reutilizado	1
Arduino Mega2560	Reutilizado	1
Sensor ultrasónico HC-SR04	Reutilizado	1
Encoder magnético	$9	1 (2 módulos por paquete)
Puente H	$7	1
Sensor magnético reed	Reutilizado	1

5. Conclusiones

El impulso que han ofrecido los robots móviles a distintos sectores en los últimos años ha permitido que formen actualmente parte de una necesidad; esto ha resultado ampliamente favorable para las ramas de la investigación en robótica ya que se ven apoyados por las industrias en busca de solucionar los retos que surgen día a día con estos sistemas.

Los robots móviles de ruedas siguen siendo hoy en día de los más utilizados por su facilidad de uso en ambientes controlados; a diferencia de estos, los robots oruga o de patas representan la minoría, pero no por esto de menor importancia ya que actualmente se siguen trabajando con ellos en áreas de alta exigencia como lo son exploración, rescate, cargas pesadas, etc.

Los robots oruga resultan interesantes de desarrollar no por su complejidad estructural o de programación, sino por los factores externos que le afectan en mayor medida a comparación de los robots con patas o de ruedas. Al tener una mayor superficie en contacto con su área de trabajo requiere de mayor consumo de energía, pero también de mejores controladores, más complejos que permitan la estabilización de estos sistemas en diferentes entornos.

Dada su alta capacidad de adaptabilidad y fuerza los robots oruga representan la mejor opción al momento de seleccionar misiones de exploración; debido a esto es posible utilizar estos robots como detectores de minar terrestres, aplicaciones de alto riesgo como esta requieren sea realizada por una máquina que sea capaz de moverse libremente por el terreno a distancias considerables de su operador, por ello mientras exista la exploración de entornos poco confiables, existirá un área de investigación y desarrollo para estos robots.

Anexos

- **Código Arduino**

```
const int REED_PIN = 2;
const int LED_PIN  = 13;

#include <Ultrasonic.h>

Ultrasonic sensorfrontal(53,52);   // (Trig PIN,Echo PIN)
// variables para canales del sensor
// de los dos encoder: ini
const int channelPinA = 18;
const int channelPinB = 19;
const int channelPinC = 20;
const int channelPinD = 21;
// variables para canales del sensor
// de los dos encoder: fin

// variables para motores: ini
#define motor1A 8 // In1
#define motor1B 7 // In2
#define motor1_PWM 9////////// //PWM_1
#define motor2A 5 //In3
#define motor2B 6 //In4
#define motor2_PWM 4 ///////////PWM_2
// variables para motores: fin


// variables de entrada: ini
int pwm_ideal_1;
int pwm_ideal_2;
// variables de entrada: fin
// variables del encoder: ini
const int timeThreshold = 0;
const int timeThreshold2 = 0;
long timeCounter = 0;
long timeCounter2 = 0;
long const int maxSteps = 1000000;
long const int maxSteps2 = 1000000;
long const int minSteps = -1000000;
long const int minSteps2 = -1000000;
volatile long ISRCounter = 0;
volatile long ISRCounter2 = 0;
int counter = 0;
int counter2 = 0;
bool IsCW = true;
```

```
bool IsCW2 = true;
 // variables del encoder: fin
void setup()
{
  delay(3000);//tiempo de inicio
   Serial.begin(115200);
   pinMode(REED_PIN,INPUT_PULLUP);
pinMode(LED_PIN,OUTPUT);
   pinMode(channelPinA, INPUT_PULLUP);
   pinMode(channelPinC, INPUT_PULLUP);
   attachInterrupt(digitalPinToInterrupt(channelPinA), doEncodeA, CHANGE);//change: ocurre cuando el pin
cambia de estado
   attachInterrupt(digitalPinToInterrupt(channelPinB), doEncodeB, CHANGE);// rising + falling
   attachInterrupt(digitalPinToInterrupt(channelPinC), doEncodeD, CHANGE); //rising:ocurre un flanco de
subida de low a high
   attachInterrupt(digitalPinToInterrupt(channelPinD), doEncodeC, CHANGE);// falling, ocurre en el flanco
de bajada de high a low
  // motor: ini
   pinMode(motor1A, OUTPUT); //In1  Activa el motor su salida
   pinMode(motor1B, OUTPUT); //In2  Activa el motor su salida
   pinMode(motor2A, OUTPUT); //In3  Activa el motor su salida
   pinMode(motor2B, OUTPUT); //In4  Activa el motor su salida
  // motor: fin
}

void loop()
{
  //atras
//  avanzar(230,.75);
//  giro(220,45);
//  avanzar(230,.51);
//  giro(220,46);
//  avanzar(230,.38);
//  giro(-220,115);
//  avanzar(230,.88);
//  giro(-220,95);
//  avanzar(230,.35);


//avanzar
//giro(-255,9.2);
//avanzar(230,.35);
//giro(255,48);
//avanzar(230,.78);
//giro(255,90);
//avanzar(230,.27);
//giro(-255,99.5);
//avanzar(230,.57);
//giro(-255,126);
```

```
//avanzar(230,.84);
//giro(255,130);

//nuevo mapa de la otra bomba
//giro(-255,90);
//avanzar(230,1.1);
//sensor();
//giro(255,50);
//avanzar(230,1);
//giro(255,70);
//avanzar(255,.85);
//giro(-255,68);
//avanzar(255,1.7);
//giro(-255,97);
//avanzar(255,1.3);
//giro(-255,103);
//delay(10000);
//sensor();
//avanzar(255,1.56);
//giro(-255,80);
//avanzar(255,1);
//
//giro(255,83);
//avanzar(255,1.7);
//giro(255,90);

//sensor deteccion
//avanzar(255,.5);
////giro(255,90);
//sensor();
//avanzar(255,.5);
//sensor();
////giro(255,90);

//trabajo en equipo
avanzar(255,1);
giro(-255,114);
avanzar(255,1);
giro(-255,105);
avanzar(255,.3);
delay(7000);
giro(255,10);
avanzar(255,1);
giro(-255,90);
avanzar(255,.7);
delay(5000);
giro(-255,90);
avanzar(255,1);
```

```
  exit(0); // Salir del éxito
}

void doEncodeA()
{
    if (digitalRead(channelPinA) == digitalRead(channelPinB))
    {
      IsCW = true;
      if (ISRCounter + 1 <= maxSteps) ISRCounter++;
    }
    else
    {
      IsCW = false;
      if (ISRCounter - 1 > minSteps) ISRCounter--;
    }
}

void doEncodeB()
{
    if (digitalRead(channelPinA) != digitalRead(channelPinB))
    {
      IsCW = true;
      if (ISRCounter + 1 <= maxSteps) ISRCounter++;
    }
    else
    {
      IsCW = false;
      if (ISRCounter - 1 > minSteps) ISRCounter--;
    }
}

void doEncodeC()
{
    if (digitalRead(channelPinC) == digitalRead(channelPinD))
    {
      IsCW2 = true;
      if (ISRCounter2 + 1 <= maxSteps2) ISRCounter2++;
    }
    else
    {
      IsCW2 = false;
      if (ISRCounter2 - 1 > minSteps2) ISRCounter2--;
    }
}


void doEncodeD()
{
    if (digitalRead(channelPinC) != digitalRead(channelPinD))
```

```
    {
      IsCW2 = true;
      if (ISRCounter2 + 1 <= maxSteps2) ISRCounter2++;
    }
    else
    {
      IsCW2 = false;
      if (ISRCounter2 - 1 > minSteps2) ISRCounter2--;
    }
}

// funciones para motores: ini
int mi_sat(int pwm)
{
  if (pwm>255){
    return(255);
  }
  else if (pwm<0){
    return(0);
  }
  else {
    return(pwm);
  }
}

void mi_pwm1(int mot_pwm,int x)
{
  if (x > 0){
    digitalWrite(motor1A,HIGH);
    digitalWrite(motor1B,LOW);
  }
  else{
    digitalWrite(motor1A,LOW);
    digitalWrite(motor1B,HIGH);
  }
  analogWrite(mot_pwm,mi_sat(abs(x)));
}

void mi_pwm2(int mot_pwm,int y)
{
  if (y > 0){
    digitalWrite(motor2A,HIGH);
    digitalWrite(motor2B,LOW);
  }
  else{
    digitalWrite(motor2A,LOW);
    digitalWrite(motor2B,HIGH);
  }
  analogWrite(mot_pwm,mi_sat(abs(y)));
```

```
}
// funciones para motores: fin

// funcion que hace que el carro avance
void avanzar(int vel, float dist)
{
  //variable dada por el usuario para que no choque
  int d1=15;

  ///variable para distancia: fin
  //variable distancia que mide sensor ultrasonico maximo
  int obs;
  ///variable distancia que mide sensor ultrasonico maximo: fin

  int pwm_ideal_1;
  int pwm_ideal_2;
  float dist1;
  pwm_ideal_1 = vel; // velocidad ideal 0-255
  pwm_ideal_2 = vel;
  dist1 = dist*9541;
  ISRCounter = 0;
  ISRCounter2 = 0;
  while (abs(ISRCounter2)<abs(dist1))
  {
   obs = sensorfrontal.Ranging(CM);
   if (obs>d1)
   {
     mi_pwm1(motor1_PWM,pwm_ideal_1); // 0 - 255 // proporcional y regula la velocidad de la rueda 1 para
//30                                          // ajustarse a la rueda 2
     mi_pwm2(motor2_PWM,pwm_ideal_2);
    }
   else
   {
     mi_pwm1(motor1_PWM,0); // 0 - 255
     mi_pwm2(motor2_PWM,0);
   }
  }
  mi_pwm1(motor1_PWM,0); // 0 - 255
  mi_pwm2(motor2_PWM,0);
}

// funcion que hace que el carro gire
void giro(int vel, float ang)
{
  int pwm_ideal_1;
  int pwm_ideal_2;
  float ang1;
  pwm_ideal_1 = vel; // velocidad ideal 0-255
  pwm_ideal_2 = vel;
```

```
  ang1 = 5300*ang/360; // 5200 funciona bien hacia derecha, pero no hacia izquierda, por eso 5300
  ISRCounter = 0;
  ISRCounter2 = 0;
  while (abs(ISRCounter2)<abs(ang1)){
  mi_pwm1(motor1_PWM,pwm_ideal_1); // 0 - 255 // proporcional y regula la velocidad de la rueda 1 para
                                                // ajustarse a la rueda 2
   mi_pwm2(motor2_PWM,-pwm_ideal_2);
  }
  mi_pwm1(motor1_PWM,0); // 0 - 255
  mi_pwm2(motor2_PWM,0);
}

void curva(int vel, float r, int lados)
{
  int i=1;
  for (i=1;i<=lados;i++)
  {
    avanzar(vel,r*sin(180/lados*PI/180)/sin((90-180/lados)*PI/180));
    giro(vel,360/lados);
    avanzar(vel,r*sin(180/lados*PI/180)/sin((90-180/lados)*PI/180));
  }
}

int sign(int x)
{
  if (x<0) return (-1);
  else{
    if (x>0) return (1);
    else return(0);
  }
}

void sensor()
{
int proximity = digitalRead(REED_PIN);
  if(proximity ==LOW)

  {
    Serial.println("!!!!Deteccion de explosivo!!!!");
    digitalWrite(LED_PIN,HIGH);
  }

  else
  {
    digitalWrite(LED_PIN,LOW);
  }
  //exit(0); // Salir del éxito
}
```

Referencias

[1] Aguilera Hernandez, M. I., Bautista, M. A., & Iruegas, J. (2007). *Diseño y control de robots moviles*. Obtenido de Mecamex: http://www.mecamex.net/anterior/cong02/papers/art24.pdf

[2] Ponticelli Lima, R. C. (2010). *Sistema de exploracion de terrenos con robots moviles: aplicacion en tareas de deteccion y localizacion de minas antipersonas.* Recuperado el 28 de 05 de 2018, de Eprints: http://eprints.ucm.es/12318/1/T32658.pdf

[3] Gonzalez , R., Rodriguez, F., & Guzman, J. L. (2015). *Robots moviles con orugas: Historia, modelado, localizacion y control*. Revista Iberoamericana de Automática e Informática industrial, 3-12.

[4] Mariscal-García, E. (2005). *Planeación y seguimiento de trayectorias de robots móviles en una simulación de un ambiente real*. Ra Ximhai, *1* (1), 177-200.

[5] Gonzalez Sieira, A. (2011). *Planificacion de movimiento en robotica movil utilizando reticulas de estados.* Obtenido de Citius: http://persoal.citius.usc.es/adrian.gonzalez/pdf/Gonzalez-Sieira11_thesis.pdf

[6] Janich, K. (1984). *Topology.* Alemania: Universitat Regensburg.

[7] Batorune, A. O. (5 de Julio de 1995). *Planificacion de trayectoria para robots moviles.* Obtenido de Webpersonal: http://webpersonal.uma.es/~VFMM/PDF/cap2.pdf

[8] Bishop, R. P. (1992). *Fundamentos de informatica.* Anaya.