VOLUMEN 28
Verano de la Ciencia XXIX
ISSN 2395-9797
www.jóvenesenlaciencia.ugto.mx

Diseño y construcción de un robot de combate con rutinas programables

Juan Ulises Calderón Huerta[1], Jesús Eduardo González Trigueros[1], Emilio Muñoz Rivera[1], Fabian Ramírez Berduzco[1], Paola del Roció Reyes Gonzalez[1], Mariana Jaqueline Peñaran Prieto[1], Dora Luz Almanza Ojeda[2].

[1]División de Ingenierías del Campus Irapuato-Salamanca, Universidad de Guanajuato.
[2]Departamento de Ingeniería Electrónica, División de Ingenierías del Campus Irapuato-Salamanca, Universidad de Guanajuato.
{ju.calderonhuerta, je.gonzalezztrigueros, e.munozrivera, pdr.reyesgonzalez, f.ramirezberduzco, dora.almanza}@ugto.mx

Resumen

Este artículo se enfoca en el diseño y construcción de un robot de combate con rutinas programables, utilizando dos opciones de comunicación: Arduino uno con control de radiofrecuencia y ESP32 IoT. Se trabaja un modelo de robot de combate, para el cual se describen las etapas de diseño, desde la selección de componentes hasta la programación en leguaje C++ (adaptado para la plataforma Arduino) de rutinas específicas de combate, como defensa, ataque y movilidad. Se programan 3 rutinas en el robot para probar diferentes niveles de velocidad y destreza en los movimientos. Los resultados muestran las capacidades y limitaciones de la configuración del robot en términos de ergonomía, consumo energético y conexión. Además, se proponen mejoras futuras como la integración de sensores adicionales, cámaras para procesamiento de imágenes y herramientas de visualización de datos en tiempo real.

Palabras clave: Robot de combate, implementación IoT, programación, eficiencia energética.

Introducción:

En 1989, la empresa japonesa FUJISOFT Inc. realiza una competencia de robots sumos llamada “All Japan Robot Sumo Tournament”, en la cual obtiene la victoria el robot que derriba o saca al oponente del área de pelea. A este evento asistieron 33 participantes, sin embargo, su impacto fue importante, introduciendo un torneo similar en Estados Unidos al año siguiente. Desde entonces la cantidad participantes en Japón se fue incrementando, y para el año 2004 su número excedía los 4000, haciéndolo de las competencias de robots más populares. Lo anterior, dio inicio a la normativa de competencia en las actividades organizadas por IEEE. (Choto Chariguaman, Pozo Safla, Aquino Arroba, & Morillo Robles, 2012)

La robótica de combate es una disciplina especializada dentro de la robótica, cuyo enfoque principal es el desarrollo de robots diseñados para participar en enfrentamientos directos. Estos robots pueden ser controlados de manera remota o actuar de forma autónoma, con el objetivo final de inmovilizar a sus oponentes en una arena de combate con armas para dificultar el enfrentamiento. La capacidad de un robot para inmovilizar a otro no solo depende de su diseño físico, sino también de su programación y estrategias de combate. La mecánica de un robot de combate incluye la resistencia a impactos, la disposición de continuar operando pese a recibir daños significativos, y la utilización de materiales robustos que le permitan mantenerse funcional. (dgrant7, 2017)

Además de la solidez estructural, las rutinas programables de estos robots son fundamentales para su desempeño en el campo de batalla. Estas rutinas incluyen instrucciones y estrategias preprogramadas que el robot puede ejecutar de manera autónoma o a través de un control para adaptarse a diferentes situaciones de combate. La resistencia mecánica y la programación define la eficacia de un robot de combate. Mientras la estructura física del robot le permite soportar y resistir los daños, su programación le otorga la inteligencia necesaria para anticiparse a los movimientos del oponente, esquivar ataques y ejecutar maniobras ofensivas de manera eficiente. En este trabajo presentamos el diseño del robot de combate llamado “vespa” y que fue el prototipo ganador del 3er lugar en el torneo “Batalla de robots Liga GTO” organizada por la Secretaria de Educación de Guanajuato y el Gobierno del estado. Se presentan los diagramas de diseño en solid works, las caracteristicas y conexiones de los componentes electrónicos y la descripcion de las rutinas programadas para dotar al robot de rutinas inteligentes de escape, ataque y navegación en el escenario.

Marco Teórico y Materiales:

Los robots de combate deportivo son máquinas especialmente diseñadas para competir en eventos como el popular torneo estadounidense "BattleBots" (BattleBots, n.d.). En esta competencia, los robots se enfrentan en una arena con el objetivo de deshabilitar o destruir a sus oponentes. Los robots están equipados con diversas armas y sistemas de defensa que les permiten ejecutar estrategias ofensivas y defensivas durante los combates. La construcción de los robots de combate requiere una combinación de habilidades en ingeniería mecánica, electrónica, programación y un mínimo de componentes que se describen a continuación.

Tabla 1 Descripción de componentes y elementos básicos de un robot de combate

Componente	Funcionalidad
Chasis	Es la estructura base del robot, proporcionando soporte y protección a todos los demás elementos.
Motor	Proporcionan la potencia necesaria para el movimiento del robot.
Drivers de Motor	Estos dispositivos gestionan la velocidad y dirección de los motores.
Microcontroladores	Actúan como unidades de procesamiento central, encargadas de coordinar las operaciones y ejecutar los comandos que guían el comportamiento del robot.
Controlador Electrónico de Velocidad (ESC)	Especialmente diseñado para motores sin escobillas trifásicos utilizados en drones, aeromodelos y robots.
Armamento	Varía según el diseño y propósito del robot. En competiciones deportivas, las armas comunes incluyen sierras, martillos y lanzallamas, entre otras.
Batería	Esencial para proporcionar la energía necesaria para el funcionamiento del robot durante los combates.

Materiales:

Se enlistan los materiales utilizados para el diseño y construcción del robot:

1. **Motores:**

   1.1 Motores de Tracción (AndyMark 775 RedLine Motor)
   1.2 Motor de Arma (Flipsky BLDC Belt Motor Battle Hardened 6384)

2. **Control de Radiofrecuencia:** Radiolink T8S 8 Channels 2.4GHz RC Transmitter and Receiver R8EF RX
3. **Llantas**: Colson Gris TPE Performa Soft Rubber Wheel
4. **Batería:** Batería Lipo de 14.8V 6500mAh
5. **Escudo de Controlador de Motor:** Pololu Dual G2 High Power Motor Shield
6. **Controlador Electrónico de Velocidad (ESC):** SkyRC Hobbyking 100A ESC
7. Arduino Uno REV3 [A000066]
8. Interruptor Anti-Chispas Inteligente Flipsky 280ª
9. ESP32-WROOM-32D
10. Regulador de corriente 12/24 V a 5V

Metodología:

Diseño:

Para llevar a cabo el diseño del prototipo "Vespa", se ha optado por utilizar un chasis fabricado en aluminio industrial de alta resistencia. Este material fue seleccionado específicamente por su habilidad para resistir impactos severos y altas temperaturas. La estructura del chasis ha sido diseñada para albergar y proteger los componentes internos del robot, y además para integrar características que mejoran su desempeño en combate. El diseño fue realizado por software SolidWorks.

a) Vista esquina superior izquierda
b) Vista esquina superior derecha
c) Vista lateral izquierda
d) Vista lateral derecha
e) Vista esquina inferior izquierda
f) Vista esquina inferior derecha
g) Vista frontal

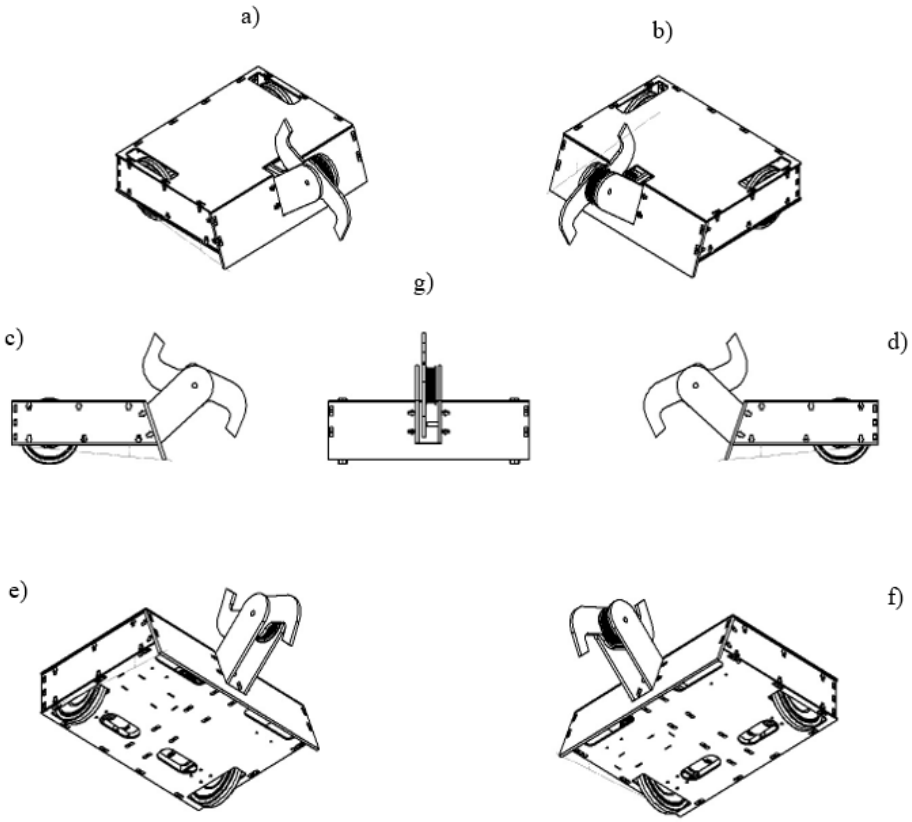

Figura 1 Vistas en perspectiva isométrica y vistas ortográficas del prototipo "Vespa"

La geometría del chasis se ha optimizado para maximizar el espacio disponible, asegurando una disposición compacta de todos los elementos del robot. Esta disposición no solo facilita la disposición y mantenimiento de los componentes, sino que también garantiza su protección durante los combates del robot. Se detallan los planos de los laterales y las placas superior e inferior del robot de combate. La parte lateral y trasera, con dimensiones de 298.05 mm y 393.70 mm de largo por 80 mm de alto (Figura 2), incluyen ranuras y orificios para el ensamblaje estructural, y la integración de componentes como motores, ejes, baterías y armamento. Las placas superior e inferior, con dimensiones de 412.75 mm de largo por 274.54 mm y 304.80 mm de ancho (Figura 3), presentan múltiples orificios y ranuras para la fijación de componentes electrónicos, garantizando un ensamblaje preciso y robusto. Estos diseños aseguran un diseño más combativo y fuerte para fines de las batallas y escenarios a los que se enfrenta un robot de este tipo.

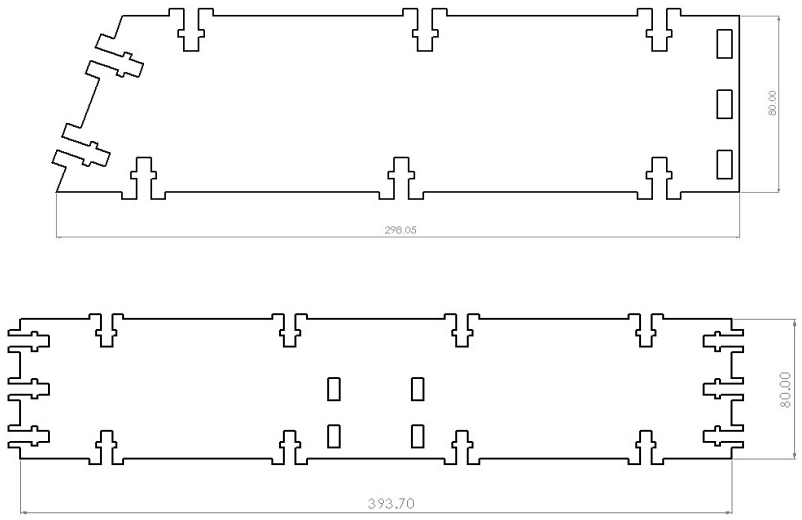

Figura 2 Partes laterales y traseras del prototipo "Vespa"

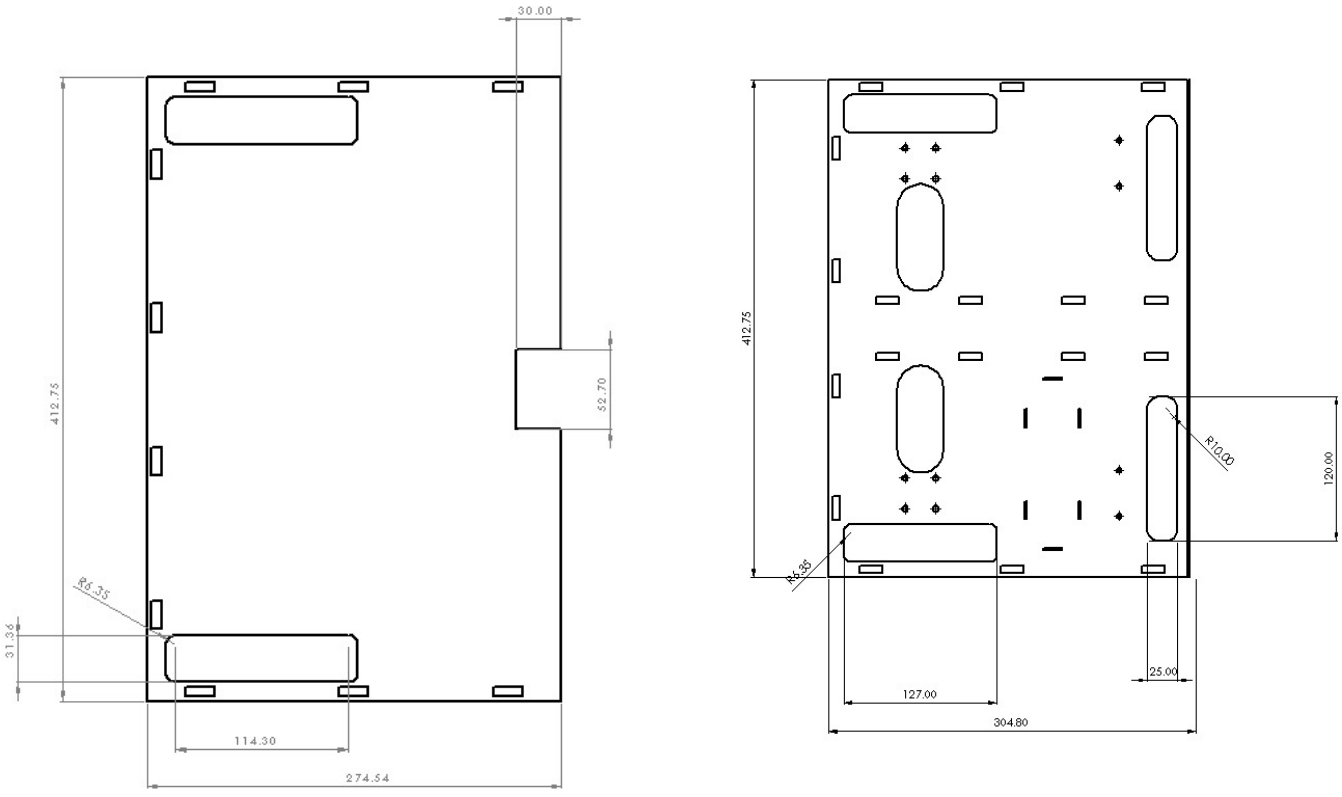

Figura 3 Placa superior e inferior del prototipo "Vespa"

Uno de los aspectos destacados del diseño es la inclusión de una rampa ubicada al frente en el chasis. Con medidas de 412 mm por 120 mm (Figura 4) esta rampa cumple una doble función durante los combates. Primero, actúa como un escudo activo contra los ataques giratorios y otros impactos directos de los oponentes, desviando y absorbiendo parte del daño para proteger los componentes críticos del robot. Segundo, la rampa puede ser utilizada ofensivamente levantando y desequilibrando a los oponentes, aprovechando su estructura robusta y la fuerza del motor para causar daño con el arma del robot.

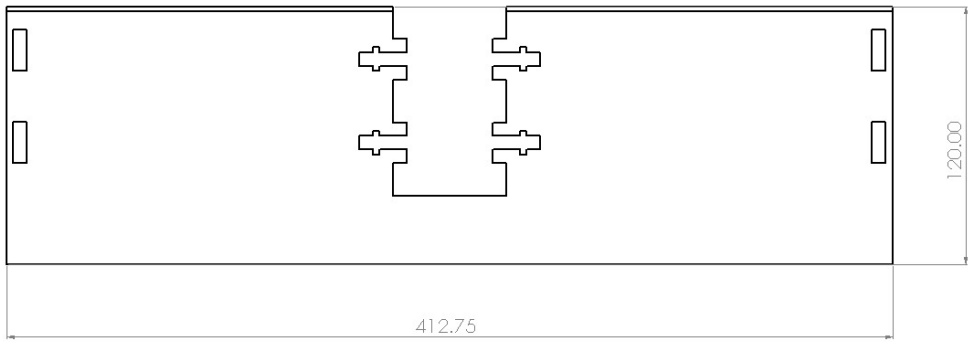

Figura 4 Rampa del prototipo "Vespa"

El soporte del arma, como se muestra en el plano (Figura 5), es una pieza diseñada para proporcionar una base estable para el la pieza del arma. Con una longitud total de 115.00 mm y una altura de 55.88 mm, el soporte incluye orificios estratégicos, como uno con un diámetro de 15.00 mm, que facilitan la fijación del arma y el ensamblaje con otras partes del chasis. Las ranuras de 35.00 mm y 75.00 mm y la curvatura con un radio de 38.10 mm están diseñadas para alojar componentes adicionales y permitir movimientos precisos del arma. Los recortes en la base aseguran un ensamblaje preciso con el chasis del robot, optimizando la estabilidad y funcionalidad del armamento.

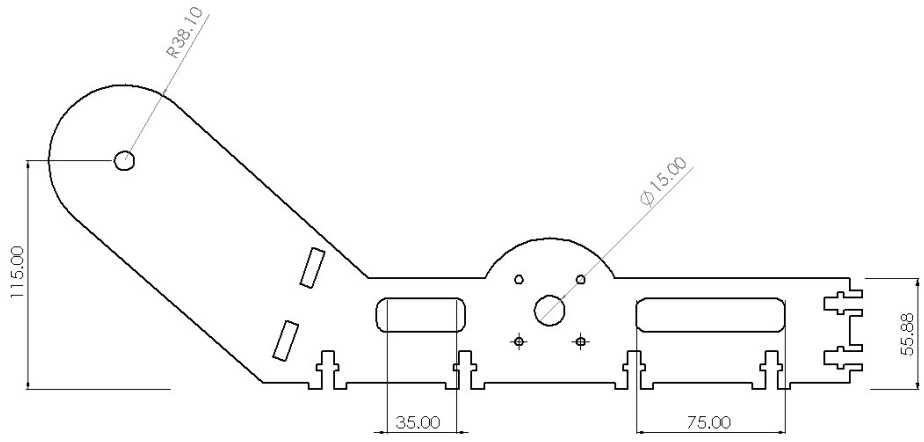

Figura 5 Soporte del arma del prototipo "Vespa"

Los planos proporcionados en la Figura 6 detallan las dimensiones generales del robot, con 412.75 de largo por 495.97mm de largo por 269.20 mm de altura, donde se observa un ensamble total del robot. Con este enfoque integral en el diseño del chasis, el prototipo "Vespa" busca mejorar su habilidad para llevar y utilizar armamento mecánico que causa severos daños a los robots contrincantes.

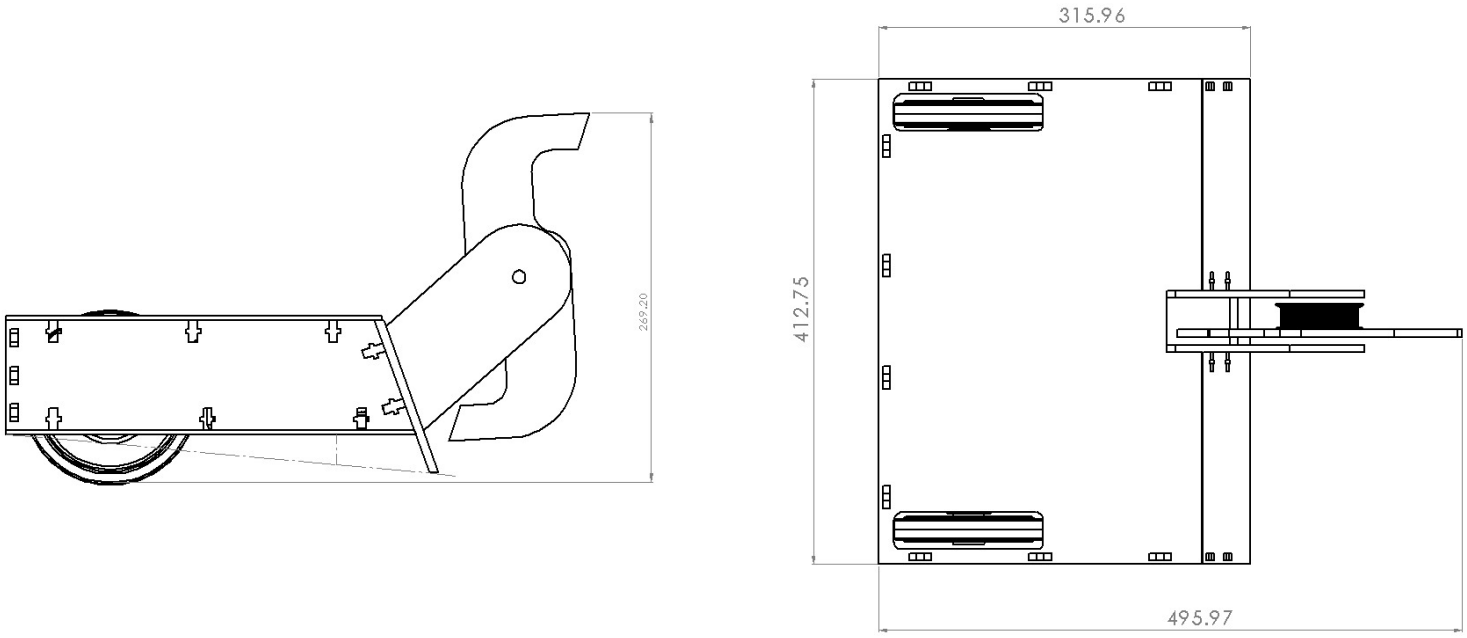

Figura 6 Vista superior y vista lateral del prototipo "Vespa"

El armamento del prototipo 'Vespa' está compuesto por una cuchilla de acero industrial de alto impacto, con 233.55 mm por 114.30, y un diámetro para la polea de 28.58 (Figura 7), diseñada con una geometría que maximiza su eficacia para causar daño durante los combates. Esta cuchilla ha sido especialmente configurada con puntas afiladas que aumentan su efectividad de destrucción de los oponentes.

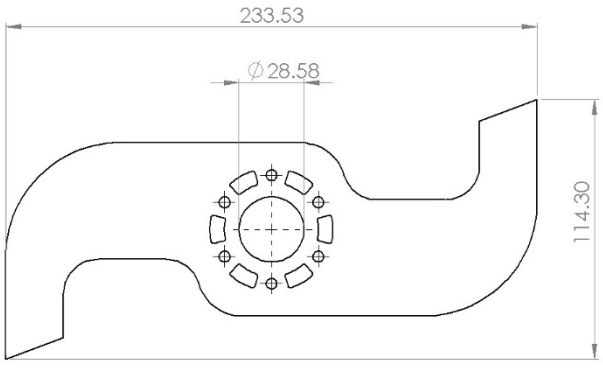

Figura 7 Arma del prototipo "Vespa"

Conexiones

Las conexiones realizadas en el robot de combate se detallan en la Figura 9, en donde el interruptor se conecta a través de la batería principal para administrar la corriente suministrada a los diversos componentes del sistema. Este interruptor actúa como un controlador central del flujo de energía, permitiendo la activación y desactivación de la corriente según sea necesario. A través de esta conexión, el interruptor asegura que los motores, sensores y sistemas de control del robot reciban un suministro eléctrico estable y regulado. Se utiliza una batería LIPO de 14.8V como fuente principal de energía. Esta batería se conecta tanto al controlador Pololu Dual G2 como al controlador del motor del arma mediante conectores XT60. El terminal positivo de la batería se conecta al terminal ViN del controlador Pololu y al driver del motor del arma a través de los conectores XT60, proporcionando la energía necesaria para ambos sistemas. El terminal negativo de la batería se conecta al terminal GND del controlador y al driver del motor del arma, completando el circuito de alimentación y estableciendo una referencia de tierra común.

Para alimentar y controlar los motores mediante el controlador Pololu Dual G2 y una placa Arduino (Figura 8), se establecen las siguientes conexiones:

1. Conexiones de Alimentación:
   El terminal positivo de la batería se conecta al terminal ViN del controlador Pololu para suministrar energía, mientras que el terminal negativo de la batería se conecta al terminal GND para completar el circuito y establecer una referencia de tierra.
2. Conexiones de los Motores:
   - Motor 1 (B1): El cable positivo se conecta al terminal M1A y el cable negativo al terminal M1B del controlador Pololu.
   - Motor 2 (B2): El cable positivo se conecta al terminal M2A y el cable negativo al terminal M2B del controlador Pololu.
3. Conexiones del Receptor RC:
   El cable de señal del receptor RC se conecta al pin analógico A0 de la placa Arduino para leer las entradas de control, y el cable de tierra se conecta al terminal GND del controlador Pololu para una referencia común.
4. Conexiones entre Pololu Dual G2 y Arduino:
   La Pololu Dual G2 se conecta a la placa Arduino mediante pines de encabezado, permitiendo que la placa Arduino se monte en la parte superior del escudo, proporcionando control y alimentación al sistema.
5. Conexiones Específicas del Encabezado de 6 Pines:
   - Pin 1 (ViN): Conecta a la salida de voltaje de 5V de la placa Arduino.
   - Pin 2 (GND): Conecta a la tierra de la placa Arduino.
   - Pin 3 (A0): Conecta a un pin analógico de la placa Arduino para lecturas adicionales.
   - Pin 4 (M1A): Conecta al pin de control de dirección del motor 1.
   - Pin 5 (M1B): Conecta al pin de control de velocidad del motor 1.
   - Pin 6 (M2A): Conecta al pin de control de dirección del motor 2.
   - Pin 7 (M2B): Conecta al pin de control de velocidad del motor 2.

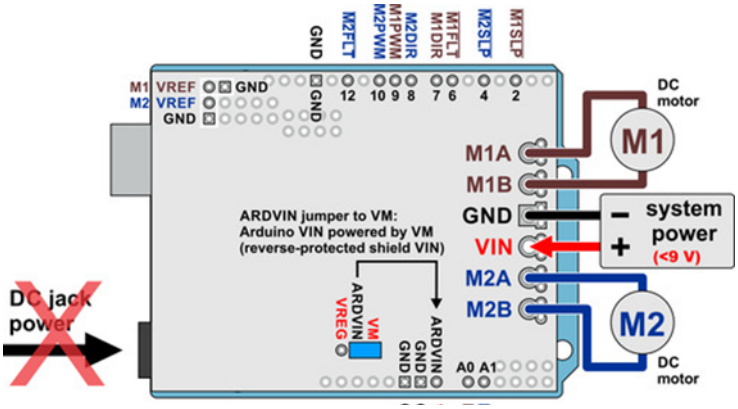

Figura 8 Diagrama de conexiones del Pololu Dual G2

El controlador ESC Flipsky FSESC6.9 gestiona al motor del arma Flipsky BLDC Belt Motor Battle Hardened 6384 mediante las siguientes conexiones: el cable azul del motor se conecta a la terminal A del ESC, el cable negro a la terminal B y el cable amarillo a la terminal C. Para las conexiones de la batería, el cable rojo de la batería se conecta al terminal BAT+ del ESC y el cable negro al terminal BAT- del ESC. Por último, el receptor RadioLink se conecta a la salida de 5V de la placa Arduino a través del canal 1, utilizando el polo negativo para la conexión. Las rutinas programadas se gestionan a través del canal 6 del receptor, que se conecta al pin 5 del Arduino. El canal 5 del receptor tiene la función de cambiar las rutinas asignadas en el canal 6, permitiendo la selección entre tres rutinas distintas, y también ajusta las velocidades de los motores. El canal 5 permite la selección de las rutinas, las cuales se activan a través del canal 6. Todas las rutinas tienen

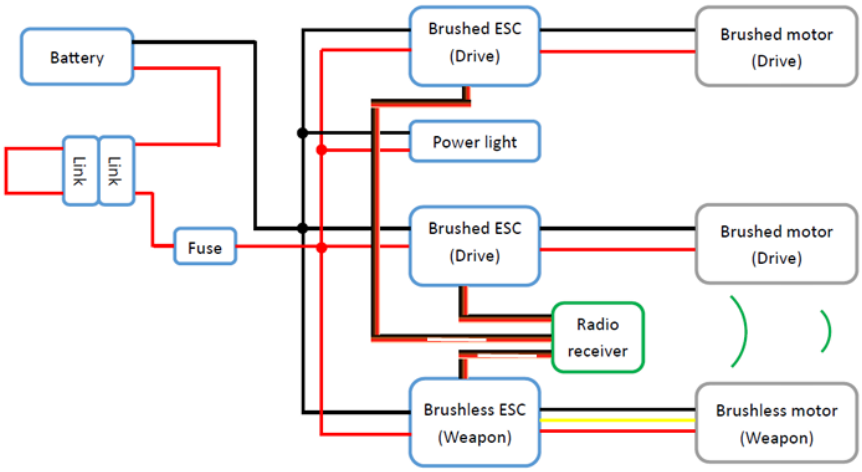

Figura 9 Diagrama de conexiones del prototipo "Vespa"

velocidades predeterminadas, lo que significa que los cambios de velocidad realizados mediante el canal 5 no afectan la velocidad de las otras rutinas. Algunas rutinas tienen una velocidad fija, debido a la naturaleza de la rutina misma, y esta velocidad no puede ser modificada. El control del movimiento de los motores hacia adelante y hacia atrás se realiza a través del canal 1, mientras que el movimiento hacia la izquierda y la derecha se gestiona mediante el canal 2. El motor del arma se controla utilizando el canal 3. Esta configuración permite un control preciso y flexible del robot, asegurando que cada componente funcione de acuerdo con las necesidades del combate y las estrategias programadas.

Diagrama de conexiones del ESP32 (Alternativa de conexión)

El circuito presentado en la Figura 10 es una solución para integrar un ESP32, que opera a 3.3V, con componentes que requieren una lógica de 5V. El ESP32-WROOM actúa como el cerebro del sistema, procesando señales y ejecutando las rutinas programables del robot, operando a un voltaje de 3.3V. Se utilizan transistores como el 2N2222A y el BC547, que funcionan como interruptores electrónicos permitiendo la conversión de señales de 3.3V a 5V, asegurando así que las señales de bajo voltaje del ESP32 puedan controlar componentes que requieren señales de 5V. El Pololu Dual G2 High-Power Motor Driver Shield controla los motores del robot, permitiendo movimientos en diferentes direcciones y velocidades, recibiendo señales de dirección y velocidad (DIR, PWM) desde el ESP32 a través de los transistores.

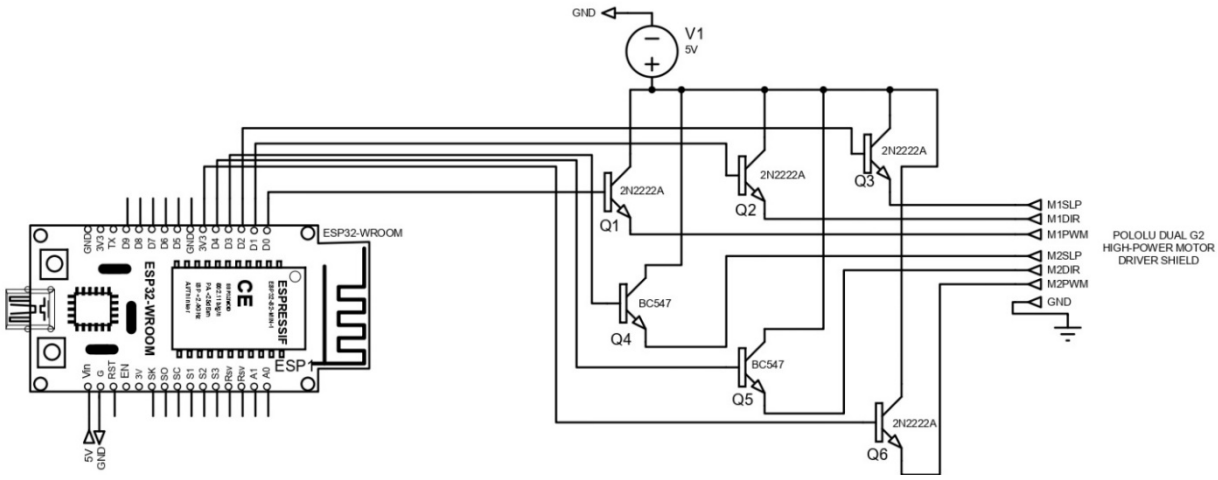

Figura 10 Diagrama de conexiones del ESP32

Los diodos en el circuito protegen contra posibles picos de voltaje, asegurando que la corriente fluya en la dirección correcta, mientras que las resistencias limitan la corriente que pasa a través de los transistores, protegiéndolos de daños. El ESP32 se conecta y se alimenta con 3.3V, comunicándose con el resto del circuito a través de pines específicos. Los transistores amplifican las señales de control del ESP32, permitiendo que las señales de bajo voltaje controlen componentes de mayor voltaje. El Motor Driver Shield recibe estas señales amplificadas y controla dos motores conectados a sus salidas, permitiendo un control preciso del robot. Este circuito permite al ESP32 controlar el robot de combate utilizando un protocolo de comunicación moderno como el IoT.

Rutinas programables

Las rutinas implementadas se realizaron en el lenguaje C++ (adaptado para la plataforma Arduino), enfocándose en tres movimientos principales: Defensa, Ataque y movilidad, adicionalmente se aplico un regulador de velocidad para manipular las 3 velocidades de las rutinas que son 100%, 62.5% y 25%

Rutina de movilidad: La rutina de movilidad implementada en el código demuestra cómo el robot puede moverse de manera precisa y controlada, ejecutando varios movimientos que mejoran su capacidad de maniobra. La secuencia de comandos comienza con el retroceso del robot a una velocidad de -50 unidades durante 1 segundo. Este movimiento le permite al robot retroceder aproximadamente 20 centímetros, creando una distancia de seguridad frente a un oponente. Luego, el robot se detiene brevemente durante 0.5 segundos. Posteriormente, el robot avanza a una velocidad de 50 unidades durante 1 segundo, cubriendo aproximadamente 20 centímetros hacia adelante. El motor M2 se mueve hacia atrás a -50 unidades mientras el motor M1 avanza a 50 unidades, ambos durante 1 segundo. Este giro le permite al robot cambiar de dirección hacia la izquierda, cubriendo un arco de aproximadamente 90 grados. Después del giro, el robot realiza un avance controlado nuevamente a una velocidad de 50 unidades durante 1 segundo. Finalmente, ambos motores se detienen. Esta secuencia de movimientos controlados—retroceso, avance, giro hacia la izquierda y detención—demuestra que la rutina está orientada a maximizar la movilidad del robot. Permite al robot moverse de manera precisa y rápida en respuesta a diversas situaciones en la arena de combate. La capacidad de retroceder y avanzar rápidamente, combinado con giros precisos, asegura tanto la evasión de ataques como el posicionamiento estratégico para ofensivas o defensas futuras, haciendo de esta rutina una excelente estrategia de movilidad.

```
Function loop()
    Set M2 speed to -50
    Set M1 speed to -50
    Wait 1000 milliseconds
    Set M2 speed to 0
    Set M1 speed to 0
    Wait 500 milliseconds
    Set M2 speed to 50
    Set M1 speed to 50
    Wait 1000 milliseconds
    Set M2 speed to -50
    Set M1 speed to 50
    Wait 1000 milliseconds
    Set M2 speed to 50
    Set M1 speed to 50
    Wait 1000 milliseconds
    Set M2 speed to 0
    Set M1 speed to 0
End function
```

Rutina de defensa: La rutina de defensa implementada en el código demuestra cómo el robot maneja situaciones defensivas. La secuencia comienza con el retroceso del robot a una velocidad de -50 unidades durante 1.5 segundos, lo que permite al robot alejarse aproximadamente 50 cm de una amenaza. A continuación, el robot realiza un giro en el lugar durante 0.65 segundos, con el motor M2 avanzando a 50 unidades y el motor M1 retrocediendo a -50 unidades. Este giro permite al robot cambiar su orientación hacia la izquierda en un arco de 60°. Después del giro, el robot avanza a una velocidad de 50 unidades durante 1.5 segundos, cubriendo una distancia de aproximadamente 50 cm hacia adelante. Finalmente, ambos motores se detienen. Esta secuencia de movimientos —retroceso, giro hacia la izquierda, avance y detención— muestra que la rutina está orientada a mejorar la capacidad del robot para defenderse y reposicionarse. El retroceso inicial crea distancia, el giro cambia la orientación, y el avance controla el reposicionamiento, lo que asegura que el robot pueda adaptarse a las amenazas y mantenerse en una posición defensiva estratégica.

```
Function loop()
        Set M2 speed to -50
        Set M1 speed to -50
        Wait 1500 milliseconds
        Set M2 speed to 50
        Set M1 speed to -50
        Wait 650 milliseconds
        Set M2 speed to 50
        Set M1 speed to 50
        Wait 1500 milliseconds
        Set M2 speed to 0
        Set M1 speed to 0
End function
```

Rutina de ataque: La rutina de ataque implementada en el código demuestra cómo el robot puede ejecutar una serie de maniobras ofensivas. La secuencia comienza con el avance frontal del robot a una velocidad de 100 unidades durante 450 ms. A continuación, el robot realiza un giro hacia la izquierda con el motor M2 moviéndose hacia atrás a -80 unidades mientras el motor M1 avanza a 80 unidades durante 300 ms. Este giro cambia la dirección del robot en un arco de 30°. Después de la maniobra, el robot se detiene brevemente durante 400 ms. Luego, el robot avanza a la misma velocidad de 100 unidades durante 500 ms, cubriendo aproximadamente 40 cm hacia la izquierda. Posteriormente, el robot realiza un giro frontal con el motor M2 avanzando a 80 unidades mientras el motor M1 retrocede a -80 unidades durante 350 ms. Este giro permite al robot orientarse nuevamente hacia adelante, ajustando su dirección en un arco de 30°. El robot se detiene nuevamente durante 400 ms para estabilizarse. Luego, avanza a 100 unidades durante 450 ms, cubriendo 30 cm adicionales hacia adelante. A continuación, el robot gira hacia la derecha con el motor M2 avanzando a 80 unidades y el motor M1 retrocediendo a -80 unidades durante 300 ms. Este giro hacia la derecha cambia la dirección en un arco de 30°. El robot se detiene durante 200 ms antes de avanzar a 100 unidades durante 700 ms, cubriendo una distancia de aproximadamente 60 cm hacia la derecha. Finalmente, el robot realiza un giro hacia atrás con el motor M2 avanzando a 80 unidades y el motor M1 retrocediendo a -80 unidades durante 400 ms, seguido por un avance hacia atrás a 100 unidades durante 800 ms. Estos movimientos finales ajustan la posición del robot y lo reposicionan, cubriendo una distancia de aproximadamente 80 cm hacia atrás para prepararse para futuras maniobras. Esta rutina de ataque está diseñada para proporcionar al robot una combinación efectiva de avance, giro y reposicionamiento, optimizando su capacidad de ataque mientras mantiene una maniobrabilidad adaptativa. La secuencia de movimientos permite al robot atacar y ajustar su posición de manera precisa, asegurando una ofensiva eficiente y una buena capacidad de respuesta a las acciones del oponente.

```
Function loop()
    Set M2 speed to 100
    Set M1 speed to 100
    Wait 450 milliseconds
    Set M2 speed to -80
    Set M1 speed to 80
    Wait 300 milliseconds
    Set M2 speed to 0
    Set M1 speed to 0
    Wait 400 milliseconds
    Set M2 speed to 100
    Set M1 speed to 100
    Wait 500 milliseconds
    Set M2 speed to 80
    Set M1 speed to -80
    Wait 350 milliseconds
    Set M2 speed to 0
    Set M1 speed to 0
    Wait 400 milliseconds
    Set M2 speed to 100
    Set M1 speed to 100
    Wait 450 milliseconds
    Set M2 speed to 80
    Set M1 speed to -80
    Wait 300 milliseconds
    Set M2 speed to 0
    Set M1 speed to 0
    Wait 200 milliseconds
    Set M2 speed to 100
    Set M1 speed to 100
    Wait 700 milliseconds
    Set M2 speed to 80
    Set M1 speed to -80
    Wait 400 milliseconds
    Set M2 speed to 0
    Set M1 speed to 0
    Wait 200 milliseconds
    Set M2 speed to 100
    Set M1 speed to 100
    Wait 800 milliseconds
    Set M2 speed to 80
    Set M1 speed to -80
    Wait 400 milliseconds
End Function
```

En las diferentes rutinas programadas se manejan tres niveles de velocidad para ajustar el comportamiento del robot

- Velocidad del 25%: Cuando el valor del canal CH3 se encuentra en un rango específico, el robot opera a una velocidad máxima del 25% de su capacidad. Esto se refleja en el código como una velocidad de 100 unidades en un rango que va de -400 a 400.
- Velocidad del 62.5%: Si el valor de CH3 está en un rango intermedio, el robot opera a una velocidad máxima del 62.5% de su capacidad total. Esto se traduce en velocidades de hasta 250 unidades en un rango de -400 a 400.
- Velocidad del 100%: Cuando el valor de CH3 supera un umbral más alto, el robot opera a su velocidad máxima del 100%, equivalente a 400 unidades en el rango mencionado.

La rutina ajusta la velocidad de los motores según el valor del canal CH3 según las condiciones. La función map() convierte los valores de los canales CH1 y CH2 en rangos de velocidad específicos para los motores.

```
Function loop()
    ch[0] = pulseIn(CH1, HIGH)
    ch[1] = pulseIn(CH2, HIGH)
    ch[2] = pulseIn(CH6, HIGH)
    ch[3] = pulseIn(CH5, HIGH)
    If (ch[3] < 1600)
        Set M1 speed to map(ch[0], 980, 1980, -100, 100)
        Set M2 speed to map(ch[1], 980, 1980, -100, 100)
    Else If (ch[3] > 1800) // Velocidad del 100%
        Set M1 speed to map(ch[0], 980, 1980, -400, 400)
        Set M2 speed to map(ch[1], 980, 1980, -400, 400)
    Else // Velocidad del 62.5%
        Set M1 speed to map(ch[0], 980, 1980, -250, 250)
        Set M2 speed to map(ch[1], 980, 1980, -250, 250)
End function
```

Resultados

Esta sección presenta los resultados obtenidos a través del proyecto, donde se describe el diseño y comunicaciones del robot de combate. La Figura 11 se observa el prototipo “Vespa” completamente ensamblado, con todas sus piezas manufacturadas y arma operativa.

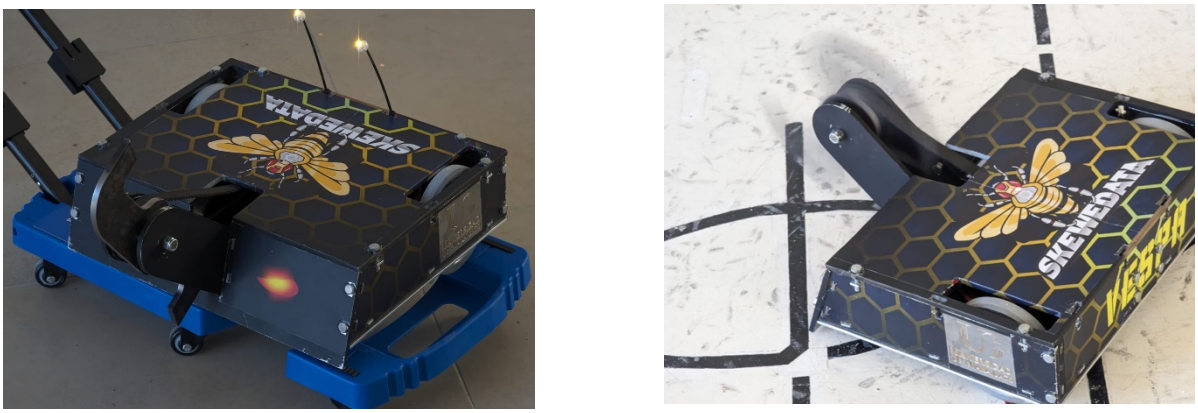

Figura 11 Vistas generales del prototipo "Vespa" ensamblado

En la Figura 12 se ilustran las conexiones realizadas en el robot para su funcionamiento básico, incluyendo las interconexiones entre la batería, los drivers, el Arduino y los motores.

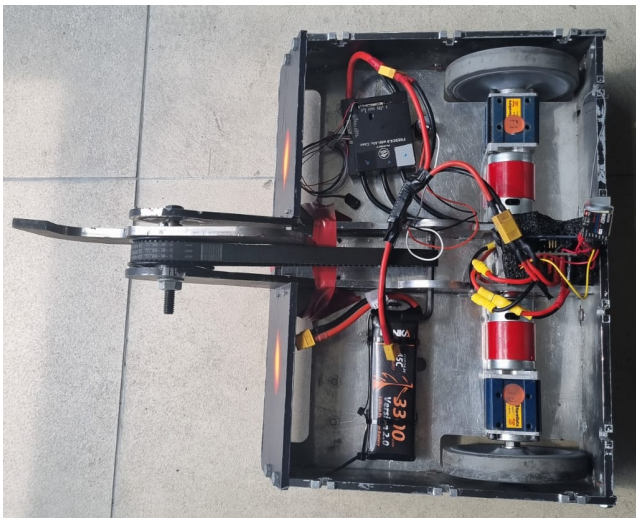

Figura 12 Conexiones internas del prototipo "Vespa"

El funcionamiento general y las rutinas programables del prototipo "Vespa" se presentan en la siguiente carpeta, donde se puede visualizar el desempeño básico del robot, sus acciones en batalla y sus rutinas programables. Esta documentación proporciona una visión completa de cómo el robot opera y responde a diferentes escenarios, demostrando tanto sus capacidades básicas como avanzadas en combate: XXIX Verano de la Ciencia_Robot de Combate Vespa

Discusión:

El robot de combate vespa desarrollado en este proyecto se caracteriza por tener tres rutinas específicas, de ataque, de defensa y de movilidad, estas rutinas se gestionan mediante un Arduino uno, que además regula

las velocidades y movimiento de los motores del robot, cada rutina tiene movimientos predefinidos que permiten al robot reaccionar y adaptarse en la arena del combate. Una alternativa de conectividad para controlar el robot es el desarrollo e implementación de la IoT (Internet de las Cosas) mediante el módulo ESP32. Por este motivo, se incorporó en el prototipo, ofreciendo opciones más modernas y prácticas para su control y operación. El robot que implementa tecnología IoT, basado en el módulo ESP32, se caracteriza por su capacidad de conectividad y comunicación en tiempo real. Este robot utiliza sensores y comunicación inalámbrica para enviar y recibir datos, lo que permite ajustar sus estrategias y movimientos en tiempo real.

El uso de Arduino UNO con control de radiofrecuencia ofrece ventajas como un diseño ergonómico y cómodo para el usuario, y conexiones simples que facilitan la configuración y manejo del robot. Aunque consume más energía que el ESP32, limitando el tiempo de operación, la plataforma Arduino UNO se destaca por su facilidad de programación y amplia documentación, simplificando la creación y depuración de rutinas de combate. En contraste, el ESP32 es notable por su eficiencia energética y bajo consumo en reposo, lo que prolonga el tiempo de operación del robot en combates. Además, el ESP32 soporta Transport Layer Security (TLS), ofreciendo protección adicional para la comunicación de datos, crucial para mantener la integridad de dispositivos y datos sensibles. Su capacidad para manejar y visualizar grandes volúmenes de datos permite un monitoreo en tiempo real y análisis de tendencias, facilitando la toma de decisiones. Por último, el ESP32 soporta una variedad de protocolos de comunicación, proporcionando flexibilidad en la implementación de tecnologías IoT según las necesidades del proyecto

Tabla 2 Resumen de la comparativa entre RF y ESP32

Características	**RF**	**ESP32 IoT**
Comodidad de Uso	Control ergonómico, conexiones simples	-
Consumo de Energía	Mayor consumo	Bajo consumo en reposo
Facilidad de Programación	Rutinas más sencillas	-
Seguridad	-	TLS: seguridad en la capa de transporte
Análisis y Visualización de Datos	-	Capacidades avanzadas de análisis y visualización
Protocolo Soportados	-	Amplia variedad de protocolos

El análisis comparativo está basado en el video adjuntado a la carpeta "Vespa vs Zangano" donde se observan las dos alternativas de conexión. Para avanzar en el desarrollo del robot de combate. En futuras investigaciones, se propone integrar sensores adicionales como los de proximidad, o cámaras con algoritmos de procesamiento de imágenes para mejorar la detección de obstáculos y enemigos, así como el reconocimiento de patrones y análisis en tiempo real. Además, la visualización por computadora mediante herramientas como Grafana permitirá monitorear y analizar el rendimiento del robot en tiempo real, optimizando las estrategias de combate y proporcionando una perspectiva completa del sistema. Estas mejoras aumentarán la funcionalidad y abrirán nuevas posibilidades para aplicaciones en escenarios más complejos y dinámicos.

Conclusiones:

La construcción del robot de combate Vespa ha demostrado un rendimiento satisfactorio en general. Durante la fase de diseño y construcción, se lograron los objetivos planteados, evidenciando la efectividad y precisión de las rutinas programables implementadas. La implementación del sistema basado en Arduino UNO con control de radiofrecuencia permitió un manejo ergonómico, brindando a los operadores una interfaz cómoda

y fácil de usar durante las competencias. Este sistema también destacó por su simplicidad en las conexiones y facilidad de programación, lo cual permitió a los usuarios concentrarse más en la estrategia de combate que en aspectos técnicos complicados. La capacidad de programar rutinas de combate facilitó la adaptación del robot a diferentes escenarios y estrategias, mejorando su desempeño en la arena de batalla. Por otro lado, la incorporación del módulo ESP32 como alternativa de conectividad introdujo ventajas significativas en términos de eficiencia energética y practicidad en la conexión. Esta opción reutilizó algunas de las conexiones ya establecidas con el Arduino, mostrando así una integración efectiva de las tecnologías. El ESP32 no solo ofreció una notable reducción en el consumo de energía en estado de reposo, lo cual prolongó el tiempo operativo del robot, sino que también mejoró la practicidad al permitir el control del robot mediante un dispositivo móvil, algo con lo que todos estamos familiarizados, en lugar de un control de videojuegos. Esta capa adicional de usabilidad es crucial para facilitar el manejo del robot y asegurar una operación confiable y sencilla durante los combates. Las perspectivas del proyecto son incorporar armamento de mayor sofisticación y capacidad de daño. Estas mejoras no solo aumentarían la efectividad del robot en combates, sino que también abrirían nuevas posibilidades en términos de tácticas y estrategias de ataque. La integración de sensores avanzados y tecnologías de conexión podría transformar al prototipo “Vespa” en un contendiente aún más poderoso en la arena de combate. Estas innovaciones, junto con el continuo desarrollo de las capacidades de conectividad, mantendrán al proyecto en la vanguardia de la tecnología robótica y de IoT, estableciendo nuevos estándares en el diseño y construcción de robots de combate.

Bibliografía/Referencias

https://www.researchgate.net/publication/365768115_Preparation_and_Characterization_of_QPVAPDDA_Electrospun_Nanofiber_Anion_Exchange_Membranes_for_Alkaline_Fuel_Cells (2012). Desarrollo de un Sistema de Control para un Robot Sumo Autónomo. En *Memorias del Congreso Internacional de Robótica y Automatización 2012* (pp. 1-6). Quito, Ecuador.

dgrant7, "How to build a combat robot," V1.2, Oct. 16, 2017. [Online]. Available: http://www.fightingrobots.co.uk/documents/Build-Rules.pdf. [Accessed: Jul. 1, 2024].

Pololu Dual G2 High-Power Motor Driver Shields for Arduino User’s Guide," Pololu, [Online]. Available: https://www.pololu.com/docs/0J72/all. [Accessed: Jul. 1, 2024].

"FLIPSKY FSESC6.9 100A base on VESC6.6 With Aluminum Anodized Heat Sink," Flipsky, [Online]. Available: https://flipsky.net/products/flipsky-fsesc6-9-100a-base-on-vesc6-6-with-aluminum-anodized-heat-sink. [Accessed: Jul. 1, 2024].

Build," BattleBots, [Online]. Available: https://es.battlebots.com/build/. [Accessed: Jul. 1, 2024].

L. S. Choto Chariguaman, E. R. Pozo Safla, S. M. Aquino Arroba, y C. E. Morillo Robles, "Análisis de impacto de un robot de combate por el método de elemento finitos," [Online]. Available: https://dialnet.unirioja.es/descarga/articulo/9042968.pdf. [Accessed: Jul. 1, 2024].

J. García Macavilca, "Estudio comparativo de plataformas," M.S. thesis, Pontificia Universidad Católica del Perú, 2023. [Online]. Available: https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/24394/GARCIA_MACAVILCA_JOSE_ESTUDIO_COMPARATIVO_PLATAFORMAS.pdf?sequence=1&isAllowed=y. [Accessed: Jul. 5, 2024].