Puck Collect Robot Challenge: aplicación de control difuso y cinemática en robot móvil diferencial y omnidireccional

Guillermo M. Madariaga Sierra y Rafael E. Calderón Álvarez***

Resumen

Puck Collect Robot Challenge es un concurso que consiste en que dos robots autónomos recolecten discos de un color (rojo para uno, azul para el otro) y los trasladen a una base ubicada en una esquina del campo, que es de 250 x 250 cm del mismo color asignado. El que traiga más discos a la base ganará; o en el caso de llevar el mismo número, el que lo haga más rápido. En este artículo se presenta la implementación de un robot autónomo en la que se aplica control difuso y conceptos de cinemática, para lograr enrutar el robot hacia el disco y posteriormente hacia la base.

Cómo citar este artículo (APA): Madariaga, G., Calderón, R. (2019). Puck Collect Robot Challenge Aplicación de Control Difuso y Cinemática en Robot Móvil Diferencial y Omnidireccional. *Hashtag, 14,* 11-30.

Abstract

Puck Collect Robot Challenge is a contest in which two autonomous robots have the task of collecting pucks of one color (red for one, blue for the other) and transfer them to a base located in a corner of the field that is 250 x 250 cm of the same color assigned. The one that brings more pucks to the base, or in the case of carrying out the same number, the one that makes faster, wins. This article presents the implementation of an autonomous robot where diffuse control and kinematics concepts are applied to route the robot to the disk and then to the base.

> Palabras clave: cinemática, control difuso, lógica difusa, procesamiento de imágenes, reconocimiento de colores, robot autónomo, visión artificial

> Keywords: Autonomous Robot, Artificial Vision, Image Processing, Color Recognition, Fuzzy Logic, Fuzzy Control, Kinematics.

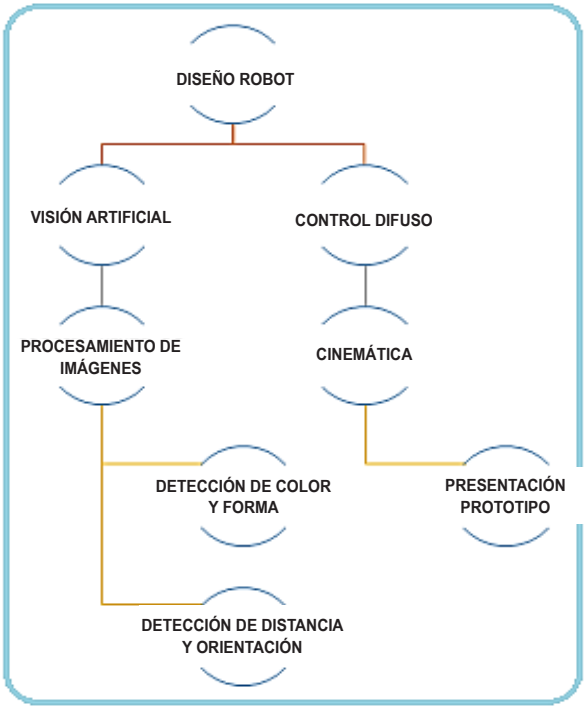
* Estudiante de Ingeniería Electrónica, CUN. Contacto: guillermo.madariaga@cun.edu.co

** Estudiante de Ingeniería Electrónica, CUN. Contacto: rafael.calderonalv@cun.edu.co

Introducción

En este artículo se presenta la implementación de una plataforma autónoma con uso de visión artificial para el reconocimiento de los colores, que trabaja con una cámara como sensor y con Python (Open CV), Matlab o Arduino (C++), como software de apoyo para este tipo de implementación. En el proceso se averiguó sobre el tamaño y forma del objeto, lo cual ayudó a calcular posición y distancia. Se explica el control difuso proporcional derivativo (PD) y su razonamiento difuso: *fuzzificación*, inferencia difusa y *deffuzzificación*, así como el modelo cinemático de dos plataformas: diferencial y omnidireccional, con el fin de exponer el desarrollo del robot con el cual se logró el enrutamiento necesario para el *Puck Collect Robot Challenge*.

Figura 1. Ruta de diseño



Fuente: elaboración propia.

Visión artificial

Reconocimiento del color

Los colores a reconocer son dos: rojo y azul, que se definieron en las reglas del concurso; sus códigos son RAL 3024 "rojo luminoso" y RAL 5013 "azul cobalto" (RobotChallenge, 2019). Para reconocer el color existen métodos que involucran un software y una cámara. En este caso se usó la cámara web, a partir de la que se toma un video en un tiempo determinado, se selecciona y procesa una imagen, para determinar el color. En general estos algoritmos pueden reconocer varios tonos del color. Se lee la imagen y se obtiene su

componente *red* del RGB. Se procesa aplicando filtros, se binariza y se aísla la imagen correspondiente. La imagen de prueba es la apreciada en la figura 2. Luego de emplear el reconocimiento de color y su aislamiento tenemos la imagen de la figura 3, así:

Figura 2. Imagen de prueba

Fuente: elaboración propia.

Figura 3. Imagen binarizada y aislada.

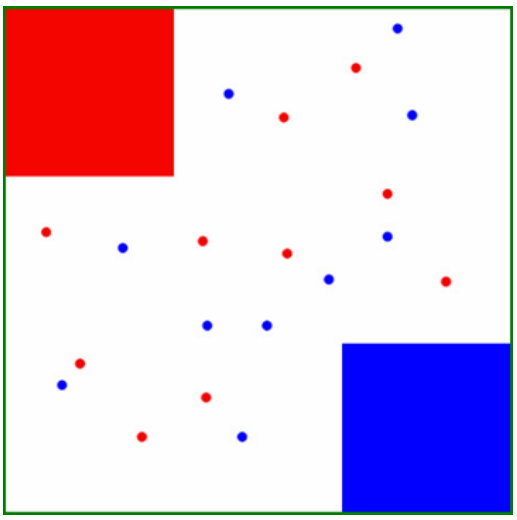
Fuente: elaboración propia.

Al usar cualquier imagen que, además de tener los colores involucrados (rojo y azul) tenga un disco de otro color, probará que el programa solo reconocerá y aislará el color que se pida. Nótese que los tonos de los colores pueden variar y aun así el reconocimiento del color es efectivo.

Conteo de objetos de un mismo color

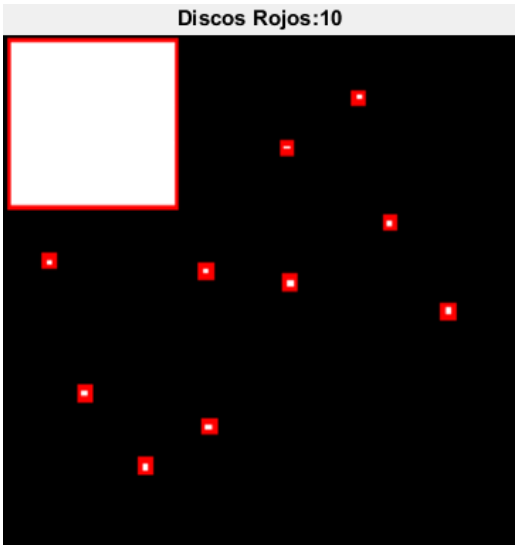
El algoritmo aplicado permite hacer un conteo de los objetos de los colores rojos detectados en una imagen. Es útil tener este dato puesto que los discos del concurso tienen un número límite de 10, los cuales se deben ir contando en el proceso de recolección, así:

Figura 4. Imagen ejemplo para contar objetos de un mismo color.



Fuente: RobotChallenge (2019, p. 3)

Figura 5. Resultado del algoritmo aplicado.



Fuente: elaboración propia.

La figura 5, con base en la figura 4, detalla los discos rojos encontrados, clasificados y contados. Lo anterior le permite saber al robot de forma autónoma, en un momento determinado, cuántos discos le falta por recoger.

Clasificación del tamaño del objeto

Calcular el tamaño del objeto que capta el robot a través de la cámara ayuda a calcular qué tan cerca está el robot del disco y, por supuesto, si se está cerca o lejos de la base que se le haya asignado. También distinguirá, de acuerdo a la forma que detecte, si se trata de la base o uno de los discos a recolectar. Todo esto con ayuda del siguiente paso a detallar: medir el área de la figura y determinar su tamaño, lo que da un estimado de cercanía al objeto o la base.

Identificación de la forma geométrica del objeto

Usando la circularidad que se mide con la propiedad de perímetro y el área del objeto detectado, se permite determinar qué forma geométrica tienen los objetos de una imagen; es otro de los usos del algoritmo implementado. Este método de visión artificial logra los objetivos iniciales cuando entrega dicha información, como parte de la solución al problema planteado.

Implementación del controlador difuso

La lógica difusa es ampliamente utilizada en el control de máquinas. El término *difuso* (en inglés, *fuzzy*) se refiere al hecho de que la lógica involucrada puede tratar conceptos que no pueden expresarse como verdaderos o falsos, sino más bien como parcialmente verdaderos. Si bien los enfoques alternativos, como los algoritmos genéticos y las redes neuronales pueden funcionar tan bien como la lógica difusa en muchos casos, la lógica difusa tiene la ventaja de que la solución al problema puede plantearse en términos que los operadores humanos puedan entender,

de modo que su experiencia pueda ser utilizada en el diseño del controlador. Esto facilita la mecanización de tareas que ya son realizadas con éxito por los seres humanos (Del Brío y Molina, 2002).

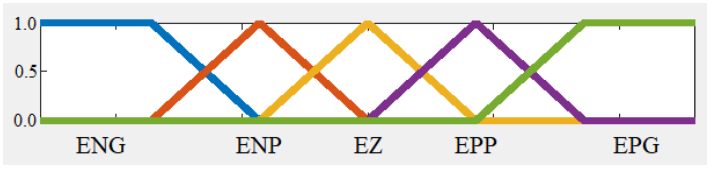
El control implementado para la velocidad del robot, que se mueve de forma autónoma por la pista, es a través de un algoritmo de control difuso. Se debe hacer el análisis necesario para generar las variables lingüísticas y sus términos. Así se infiere que cuanto más cerca se esté del objeto (disco), más lento debe ser el movimiento del robot para garantizar la recolección. Además, este tipo de control puede generar el recorrido hacia la base dispuesta al comenzar la partida. Esto también puede lograrse gracias a la medición de la distancia que se detecte desde la posición actual hasta la base, a través de todos los datos que se han analizado en este artículo.

Control difuso proporcional derivativo (PD)

El control proporcional derivativo, en la lógica difusa, es un tipo de control en el que las variables lingüísticas de entrada son el error y la velocidad de cambio de ese error, que se den en la planta, según las acciones que se tomen para modificar su estado actual. El estado de equilibrio de un control proporcional derivativo es que el error y su derivada sean cero. En una dinámica de lazo cerrado siempre va a tender a dejarlo en error y tasa de cambio cero. La derivada del error es interpretada como una velocidad: qué tan rápido se está modificando la variable de control. En este caso específico de controlar la velocidad lineal y angular del robot autónomo se usa un PD difuso.

Las variables lingüísticas de entrada {X1, X2} se definen de la siguiente forma: *Error*, para X1, y *Derivada del Error* para X2. Ambas variables se refieren a la distancia detectada por la cámara.

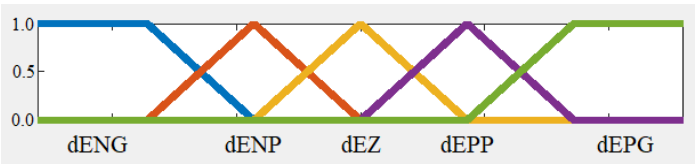
Figura 6. Términos del *Error* X1



Fuente: elaboración propia.

En la figura 6, las etiquetas corresponden a los términos así: ENG, Error Negativo Grande; ENP, Error Negativo Pequeño; EZ, Error Cero; EPP, Error Positivo Pequeño; y EPG, Error Positivo Grande.

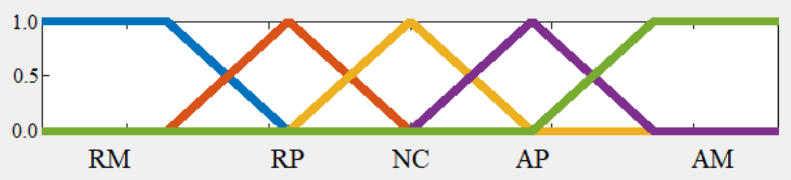
Figura 7. Términos de la *Derivada del Error* X2



Fuente: elaboración propia.

En la figura 7, las etiquetas corresponden a: dENG, derivada del Error Negativa Grande; dENP, derivada del Error Negativa Pequeña; dEZ, derivada del Error Cero; dEPP, derivada del Error Positiva Pequeña; y dEPG, derivada del Error Positiva Grande.

Figura 8. Variable lingüística de Salida (Velocidad Lineal)



Fuente: elaboración propia.

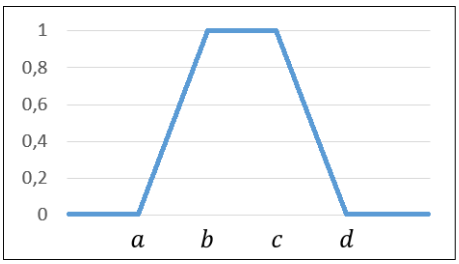
En la figura 8, las etiquetas corresponden a: RM, Retroceder Mucho; RP, Retroceder Poco; NC, No Cambio; AP, Avanzar Poco; AM, Avanzar Mucho.

Este mismo proceso se hace para el controlador de posición, cuyo universo de discurso se define por la posición del objeto en la imagen, para generar así la salida de la Velocidad Angular.

Fuzzificación

El método de fuzzificación emplea conjuntos difusos trapezoidales, cuya función matemática se detalla en la figura 9. Los valores difusos resultantes para los términos de X1 y X2 se relacionan en cada controlador, operando con una AND difusa, para obtener los valores de activación de las reglas difusas en la memoria asociativa difusa.

Figura 9. Conjunto difuso de tipo trapezoidal



$$\mu_A(u) = \begin{cases} 0.0 & ;u \le a \\ \dfrac{u-a}{b-a} & ;a < u < b \\ 1.0 & ;b \le u \le c \\ \dfrac{d-u}{d-c} & ;c < u < d \\ 0.0 & ;u \ge d \end{cases}$$

Fuente: elaboración propia.

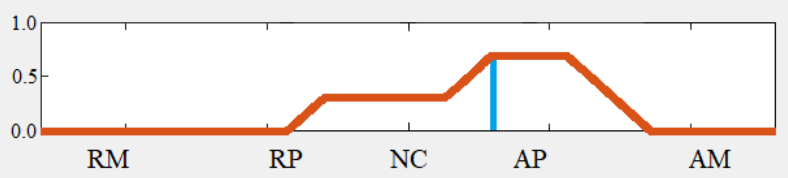
Defuzzificación

Para *defuzzificar*, se usa el método ya conocido del centroide, expresado de la siguiente forma:

$$y\ \text{centroide} = \frac{\sum_{x \in X} x\, \mu A(x)}{\sum_{x \in X} \mu A(x)}$$

(Ecuación 1)

Figura 10. Gráfica de salida VL (ejemplo para Velocidad Lineal)



Fuente: elaboración propia.

Ecuaciones del modelo cinemático

Considerando el uso de términos como Velocidad Lineal y Velocidad Angular, que son propios de la cinemática de un robot móvil, esta se detalla a continuación. El modelo cinemático de un robot diferencial considera, por ejemplo, su posición en un punto de coordenadas (x; y), y dado un punto de origen y un punto de llegada, se calcula por medio de la derivada de esas posiciones una velocidad lineal y angular (Ollero, 2001; Tzafestas, 2014).

La velocidad lineal es aquella cuya dirección siempre es tangencial a la trayectoria de una partícula en rotación, y su vector es perpendicular al eje de rotación (Ecured, 2019a). Se denomina velocidad angular a "la magnitud que caracteriza la rapidez con que varía el ángulo barrido por la línea que une la partícula que gira con el centro de rotación" (Ecured, 2019b). En este caso, la velocidad angular es proporcionada por el controlador de posición horizontal, mientras que la velocidad lineal es proporcionada por el controlador de distancia.

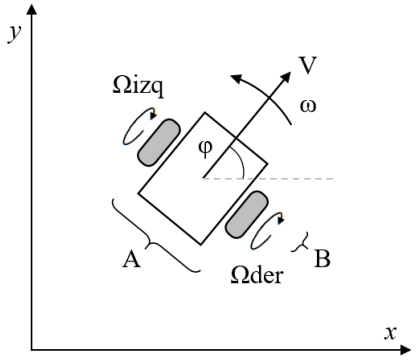
Plataforma diferencial con 2 ruedas

Considerando las dimensiones del vehículo en la figura 11, las ecuaciones de velocidad lineal (V) y angular (ω) para la plataforma diferencial están dadas por las siguientes expresiones:

$$V = \frac{\Omega_{der}+\Omega_{izq}}{2} B \qquad \omega = \frac{\Omega_{der}-\Omega_{izq}}{A} B$$

(Ecuación 2)

Figura 11. Representación del vehículo diferencial



Fuente: elaboración propia.

Con la matriz de la ecuación 3 se modela el movimiento rotacional de un cuerpo en el espacio, y con la ecuación 4 se expresa el modelo cinemático del vehículo diferencial.

$$\begin{bmatrix} cos(\varphi) & 0 \\ sen(\varphi) & 0 \\ 0 & 1 \end{bmatrix}$$

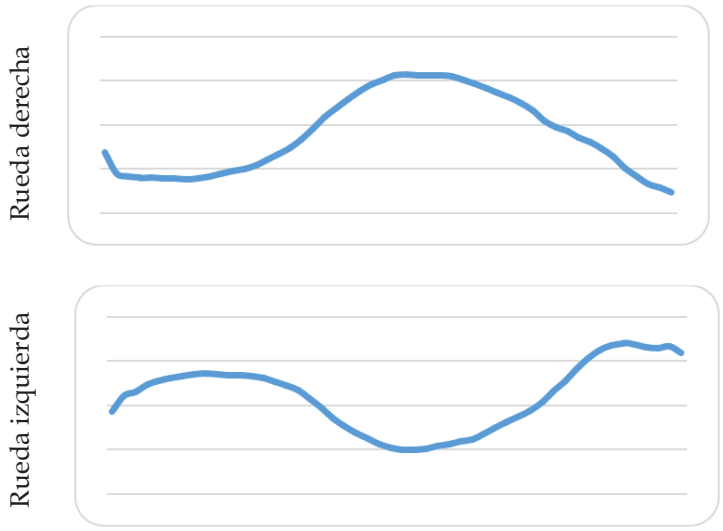
(Ecuación 3)

$$\begin{bmatrix} \Omega_{der} \\ \Omega_{izq} \end{bmatrix} = \frac{1}{B} \begin{bmatrix} cos(\varphi) & sen(\varphi) & +A/2 \\ cos(\varphi) & sen(\varphi) & -A/2 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix}$$

(Ecuación 4)

En la ecuación 4, v es la velocidad lineal, ω es la velocidad angular, ***B*** es el radio de las ruedas, ***A*** es el ancho entre las ruedas, y {**Ωder**, **Ωizq**} son las velocidades angulares de las ruedas, es decir, las velocidades con las que el móvil se mueve hacia el objeto. Usando las medidas del robot, las velocidades de las ruedas pueden ser graficadas en Excel y así, el comportamiento de las ruedas del vehículo diferencial se puede observar en la figura 12.

Figura 12. Velocidades angulares de las 2 ruedas

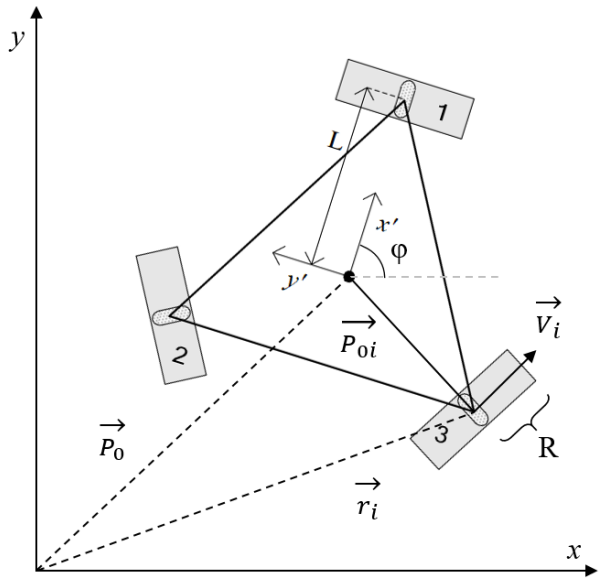


Fuente: elaboración propia.

Plataforma omnidireccional de tres ruedas

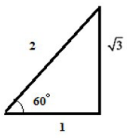
Para una plataforma de 3 ruedas las ecuaciones son las siguientes:

Figura 13. Análisis vectorial de una plataforma de 3 ruedas suecas.



Fuente: adaptado de Gracia (2006, p. 119)

Como resultado la expresión matricial que satisface el comportamiento del robot de tres ruedas tipo omnidireccional, siguiendo el análisis matemático siguiente:



$$cos(60°) = \frac{1}{2} \; ; \; sen(60°) = \frac{\sqrt{3}}{2}$$

$$\text{como } |P_{0i}| = L$$

$$\vec{P}_{01} = L \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\vec{P}_{02} = L \begin{bmatrix} -1/2 \\ \sqrt{3}/2 \end{bmatrix} = \frac{L}{2} \begin{bmatrix} -1 \\ \sqrt{3} \end{bmatrix}$$

$$\vec{P}_{03} = L \begin{bmatrix} -1/2 \\ -\sqrt{3}/2 \end{bmatrix} = \frac{L}{2} \begin{bmatrix} -1 \\ \sqrt{3} \end{bmatrix}$$

(Ecuación 5)

Las direcciones tangenciales de las ruedas se analizan de la siguiente manera:

$$\vec{D}_i = \frac{1}{L} R_Z[90°] * \vec{P}_{0i}$$

$$\text{donde } R_Z = \begin{bmatrix} cos(\varphi) & -sen(\varphi) \\ sen(\varphi) & cos(\varphi) \end{bmatrix}$$

$$\text{y dado que } R_Z(90°) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$\vec{D}_1 = \frac{1}{L} * \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} * L * \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\vec{D}_2 = \frac{1}{L} * \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} * \frac{L}{2} \begin{bmatrix} -1 \\ \sqrt{3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -\sqrt{3} \\ -1 \end{bmatrix}$$

$$\vec{D}_3 = \frac{1}{L} * \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} * \frac{L}{2} \begin{bmatrix} -1 \\ \sqrt{3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \sqrt{3} \\ -1 \end{bmatrix}$$

(Ecuación 6)

El movimiento de cada motor se analiza de la siguiente manera:

$$\vec{r}_i = \vec{P}_0 + R_Z(\varphi) * \vec{P}_{0i}$$

derivando: $\dot{\vec{r}}_i = \dot{\vec{P}}_0 + R_Z(\dot{\varphi}) * \vec{P}_{0i}$ se tiene $\vec{V}_i$

$$\vec{V}_i = \left[\dot{\vec{P}}_0 + R_Z(\dot{\varphi}) * \vec{P}_{0i}\right] * \left[R_Z(\varphi) * \vec{D}_i\right]$$

$$\vec{V}_i = \left[\dot{\vec{P}}_0 * R_Z(\dot{\varphi}) * \vec{D}_i\right] + \left[R_Z(\dot{\varphi}) * \vec{P}_{0i} * R_Z(\varphi) * \vec{D}_i\right]$$

simplificando: $\vec{V}_i = \left[\dot{\vec{P}}_0 * R_Z(\dot{\varphi}) * \vec{D}_i\right] + L\dot{\varphi}$

(Ecuación 7)

De esa manera, las ecuaciones de velocidades tangenciales se calculan así:

$$\vec{V}_1 = \left[\dot{\vec{P}}_0 * R_Z(\varphi) * \vec{D}_1\right] + L\dot{\varphi}$$

$$\vec{V}_2 = \left[\dot{\vec{P}}_0 * R_Z(\varphi) * \vec{D}_2\right] + L\dot{\varphi}$$

$$\vec{V}_3 = \left[\dot{\vec{P}}_0 * R_Z(\varphi) * \vec{D}_3\right] + L\dot{\varphi}$$

(Ecuación 8)

Considerando el radio R de las ruedas, se obtiene la siguiente expresión (Martínez y Sisto, 2009):

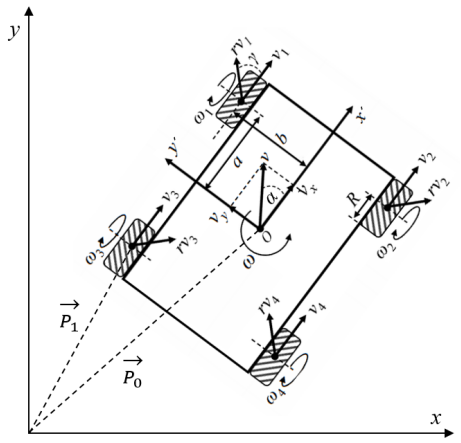
$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \frac{1}{R}\begin{bmatrix} -sen(\varphi) & cos(\varphi) & L\\ -sen\left(\frac{\pi}{3}-\varphi\right) & -cos\left(\frac{\pi}{3}-\varphi\right) & L\\ sen\left(\varphi+\frac{\pi}{3}\right) & -cos\left(\varphi+\frac{\pi}{3}\right) & L\end{bmatrix}\begin{bmatrix}V_x\\ V_y\\ \omega\end{bmatrix}$$

(Ecuación 9)

La ecuación 9 representa las velocidades angulares que debe tener cada rueda cuando se planee el recorrido hacia el disco, en caso de emplear este tipo de plataforma.

Plataforma omnidireccional de cuatro ruedas

Figura 14. Análisis vectorial de una plataforma de 4 ruedas *mecanum*.



Fuente: adaptado de Maulana, Muslim y Hendrayawan (2015, p. 51)

Planteando el análisis vectorial, se debe llegar a la expresión matricial que permita darle una velocidad a cada rueda, para planear el recorrido hacia el disco en el caso del *Puck Collect Robot Challenge*.

Las ecuaciones que caracterizan a la plataforma en cuestión son:

$$\overrightarrow{P'}_1 = [\vec{a} \quad \vec{b}]$$
$$\overrightarrow{P'}_2 = [\vec{a} \quad -\vec{b}]$$
$$\overrightarrow{P'}_3 = [-\vec{a} \quad -\vec{b}]$$
$$\overrightarrow{P'}_4 = [-\vec{a} \quad \vec{b}]$$
$$\overrightarrow{P'}_1 = \dot{\vec{P}}_0 + R_Z(\dot{\alpha})\overrightarrow{P'}_1$$
$$\vec{V}_1 = R_Z(\alpha)\,\vec{D}_1$$

(Ecuación 10)

Las direcciones tangenciales de las ruedas se tienen ahora de la siguiente manera:

$$\overrightarrow{D_1} = \begin{bmatrix}1\\0\end{bmatrix}$$
$$\overrightarrow{D_2} = \begin{bmatrix}1\\0\end{bmatrix}$$
$$\overrightarrow{D_3} = \begin{bmatrix}1\\0\end{bmatrix}$$
$$\overrightarrow{D_4} = \begin{bmatrix}1\\0\end{bmatrix}$$

(Ecuación 11)

Considerando la matriz de rotación $R_z(a)$ se tiene entonces:

$$R_Z = \begin{bmatrix}cos(\alpha) & -sen(\alpha)\\ sen(\alpha) & cos(\alpha)\end{bmatrix}$$
$$\overrightarrow{D_1}[R_Z(\alpha)] = \begin{bmatrix}cos(\alpha) & -sen(\alpha)\\ sen(\alpha) & cos(\alpha)\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}$$
$$\overrightarrow{D_i}[R_Z(\alpha)] = \begin{bmatrix}cos(\alpha)\\ sen(\alpha)\end{bmatrix}$$

(Ecuación 12)

Considerando V_x , V_y en términos de V_F, V_L (Velocidad frontal y lateral) entonces,

$$R_Z(\alpha)\begin{bmatrix}V_F\\V_L\end{bmatrix} = \begin{bmatrix}cos(\alpha) & -sen(\alpha)\\ sen(\alpha) & cos(\alpha)\end{bmatrix}\begin{bmatrix}V_F\\V_L\end{bmatrix} = \begin{bmatrix}V_F * cos(\alpha) - V_L * sen(\alpha)\\ V_F * sen(\alpha) + V_L * cos(\alpha)\end{bmatrix}$$

(Ecuación 13)

La velocidad angular de cada rueda puede expresarse de la siguiente manera:

$$\overrightarrow{P'}_1 = [V_F - b\omega]$$
$$\overrightarrow{P'}_2 = [V_F + b\omega]$$
$$\overrightarrow{P'}_3 = [V_F - a\omega]$$
$$\overrightarrow{P'}_4 = [V_F + a\omega]$$

(Ecuación 14)

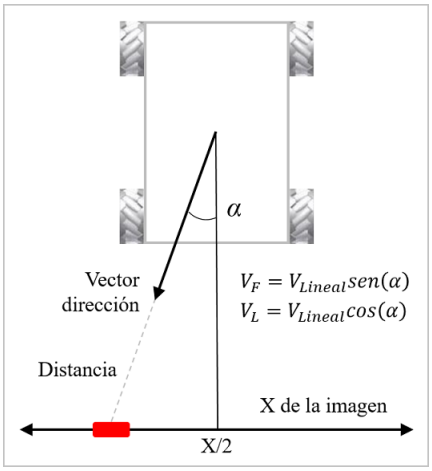
Finalmente el modelo cinemático que se encontró para esta plataforma es la siguiente, aunque, por el alcance de este trabajo no se validó, ni en simulación ni con implementación real.

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix} = \frac{1}{R}\begin{bmatrix}1 & 0 & -b\\ 1 & 0 & b\\ 1 & 0 & -a\\ 1 & 0 & a\end{bmatrix}\begin{bmatrix}V_F\\V_L\\\omega\end{bmatrix}$$

(Ecuación 15)

Teniendo el resultado del análisis vectorial, vemos que la velocidad lineal es el resultado de sus componentes V_F y V_L, velocidad frontal y lateral respectivamente. Por tanto, para poder obtenerlas del controlador difuso propuesto anteriormente, hay que conocer el vector dirección de la velocidad lineal. Esto requiere un análisis del procesamiento de imagen para llegar a la situación representada en la figura 15.

Figura 15. Análisis gráfico del posicionamiento del robot.



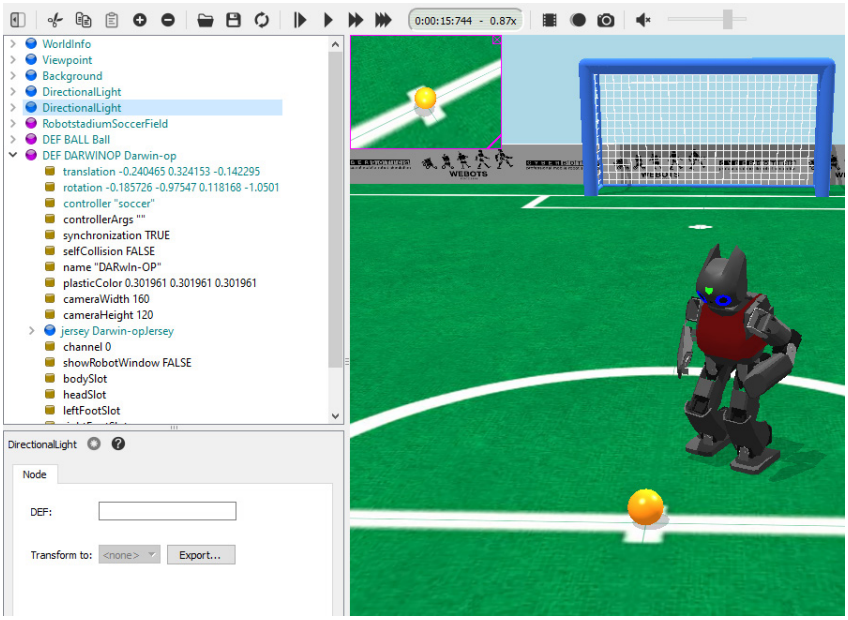
Fuente: elaboración propia.

La figura 15 corresponde con la ubicación del objeto en pantalla y la estimación de la posición del móvil y el objeto con respecto del centro de la pantalla en el eje X (la medida entre ese centro y la ubicación del objeto) y la distancia medida entre el objeto y el móvil: la distancia corresponde con el vector dirección. Se forma un triángulo rectángulo y con esas dos componentes se hallan el ángulo α del móvil y aplicando sen(α) y cos(α) se obtendrían la V_F y V_L.

Simulación en Webots (Cyberbotics)

Webots es un software de simulación que, a través de varios lenguajes de programación como Java, Matlab, C++, entre otros, permite ver el comportamiento de un robot y cómo interactúa dentro de un ambiente, que también puede ser creado en Webots (Cyberbotics, 2016a, 2016b). La interfaz cuenta, entre otras funciones, con una ventana de visualización del ambiente, en el que se insertan elementos principales como el robot y sus componentes electrónicos y mecánicos. Además, permite simular la dinámica del robot: peso, fricción, tipo de material, desgastes, etc.

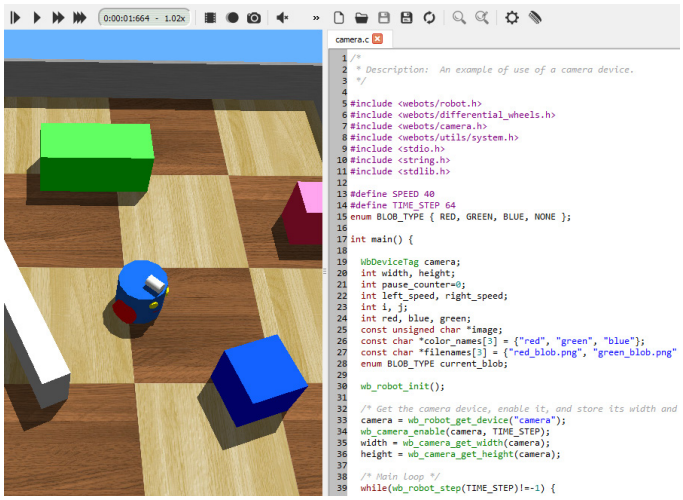
Figura 16. Interfaz gráfica de Webots.



Fuente: (Cyberbotics, 2016c)

En otra ventana del mismo software se encuentra el espacio de programación que es en donde insertamos el código fuente que será el controlador del robot. En ese espacio se puede crear, abrir, guardar o compilar el código fuente dentro del propio ambiente de trabajo de Webots, o enlazar a través de otros IDE para usarlo como controlador.

Figura 17. Espacio para el código fuente en Webots.



Fuente: (Cyberbotics, 2016c)

Webots cuenta con sus propias librerías y sus funciones están descritas en sus manuales, en los lenguajes de programación que ya han sido mencionados. Se puede acceder a estos en el panel de ayuda, o presionando directamente la tecla F3 para ver el manual de usuario (Cyberbotics, 2016a) o la tecla F4 para ver el manual de referencia (Cyberbotics, 2016b). En la figura 18 se detalla el uso de librerías para la percepción con cámara y para mover un robot diferencial.

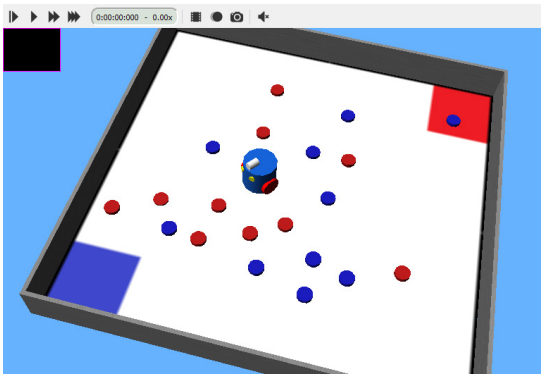
Figura 18. Algunas librerías disponibles en Webots.

```
 5 #include <webots/robot.h>
 6 #include <webots/differential_wheels.h>
 7 #include <webots/camera.h>

33   camera = wb_robot_get_device("camera");
34   wb_camera_enable(camera, TIME_STEP);
35   width = wb_camera_get_width(camera);
36   height = wb_camera_get_height(camera);

42     image = wb_camera_get_image(camera);

87       for (i = width/3; i < 2*width/3; i++) {
88         for (j = height/2; j < 3*height/4; j++) {
89           red += wb_camera_image_get_red(image, width, i, j);
90           blue += wb_camera_image_get_blue(image, width, i, j);
91           green += wb_camera_image_get_green(image, width, i, j);
92         }
93       }

114        left_speed = -40;
115        right_speed = 40;

147    wb_differential_wheels_set_speed(left_speed, right_speed);
```

Fuente: (Cyberbotics, 2016c)

En este proyecto se usó Webots como un simulador para mostrar la funcionalidad del robot. Se mueve en el ambiente del *Puck Collect Robot Challenge*, simulando incluso el escenario en donde están distribuidos los discos.

Figura 19. Diseño preliminar del robot tipo diferencial.



Fuente: elaboración propia en Webots.

En la figura 19 se simuló el campo del *Puck Collect Robot Challenge*, los discos rojos y azules distribuidos y las bases. El robot cuenta con un diseño tipo diferencial, tiene elementos electrónicos y sensores tales como cámara, motores DC y baterías. El controlador o código fuente está programado en un lenguaje soportado por el simulador, adaptando las librerías incluidas dentro del software o enlazado con el propio IDE para comprobar su funcionamiento.

Implementación del prototipo diferencial

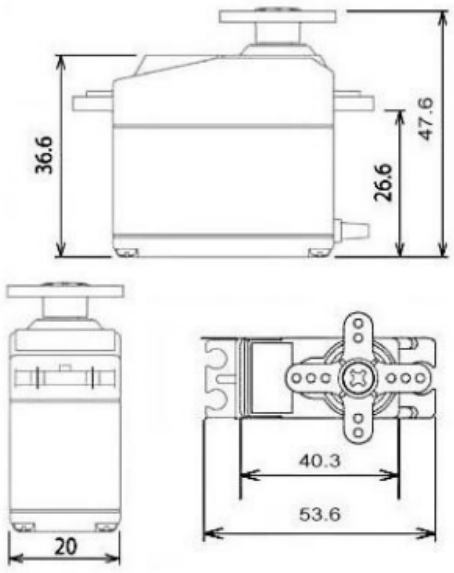
La implementación para el enrutamiento del móvil es en un vehículo tipo diferencial. Con la cámara web (con conexión IP) se captan las imágenes a las que se le aplican los algoritmos de visión artificial. Los análisis de las imágenes se combinan con cada controlador difuso PD de Posición y Velocidad, que dan como resultado la Velocidad angular y lineal del móvil, tal como se expuso anteriormente. Empleando la matriz de transformación calculada en los análisis cinemáticos, se obtienen las velocidades angulares de ambas ruedas, junto con su sentido de giro. Los elementos empleados en esta implementación se describen a continuación.

Elementos electrónicos y mecánicos (especificaciones técnicas)

Servo motor de giro continuo

Los motores que se usaron en la práctica son servomotores MG996R Tower-Pro. Este cuenta con engranajes de metal que lo proveen con una gran torsión.

Figura 20. Motor servo de giro continuo.



Fuente: Tower-Pro (2017, p. 1)

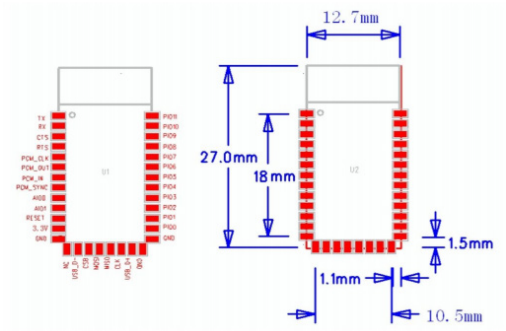
Arduino uno

Arduino se utiliza para construir diferentes tipos de circuitos electrónicos. Es utilizado en físico o como placa de circuito programable, generalmente microcontrolador, empleando códigos que se ejecutan en el computador mediante conexión USB entre este y el Arduino. El lenguaje de programación utilizado es una versión simplificada de C ++.

Módulo serial TTL y Bluetooth HC-05

El módulo HC-05 es un módulo Bluetooth SPP (protocolo de puerto serie) fácil de usar, diseñado para configuración de conexión en serie inalámbrica. Está diseñado para conexiones Bluetooth V2.0 + EDR (velocidad de datos mejorada), 3 Mbps, modulación con transceptor de radio y banda base completos de 2.4 GHz. Utiliza CSR Bluecore04-Sistema Bluetooth de un solo chip externo, con tecnología CMOS y con AFH. Se complementa la conexión al computador con un convertidor de USB a serial TTL.

Figura 21. HC-05 Módulo Bluetooth.



Fuente: ITed Studio (2010, p.2)

Baterías

Para alimentación del circuito, que costa de los elementos anteriores, se usa una batería de litio de 7.4 v. Son recargables y tienen una autonomía de 2000 mA horas.

Características:

- 7.4 V paquete de 2 celdas
- 2000 mAh de carga
- 20 C velocidad de descarga continua

Dimensiones:

- 55mm x 30mm x 14mm.

Peso:

- 36g

Figura 22. Batería recargable.

Fuente: SanDoRobotics (2018, párr. 1)

Es excelente para alimentar cualquier proyecto R/C, robótico o portátil, para cualquier móvil que requiera una batería pequeña con mucha fuerza.

Cámara web

La cámara utilizada en la implementación es tipo web que alcanza una resolución de 8 Mpx, 3264x2448 píxeles, flash LED (cámara celular configurada como web), aunque en Código de programa es reducida a la resolución MJP-G_1280x720 lo cual agiliza el procesamiento de imágenes y videos.

Computador

Debe estar equipado con Puertos de Comunicación disponibles para protocolos RS 232, serial TTL, conexión WiFi y requerimientos mínimos de hardware que soporte el software utilizado.

Plataforma móvil tipo diferencial con rueda castor

En el prototipo se usó una plataforma móvil como la mostrada en la figura 23. Es útil para diferentes tipos de proyectos que requieran los movimientos típicos de un móvil tipo diferencial: hacia delante, en retroceso, y giros con tendencia a la derecha o a la izquierda.

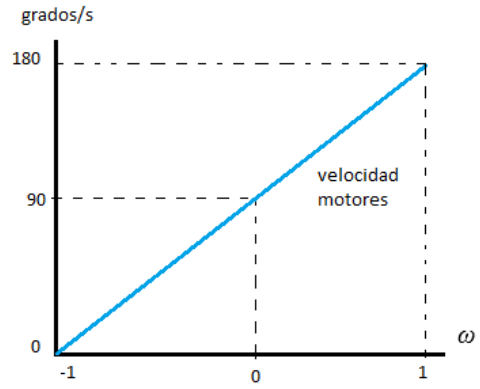
Figura 23. Plataforma móvil diferencial con rueda castor

Fuente: Verma (2017, párr. 3)

Conversión de unidades

Una vez relacionados los resultados de velocidades angulares de las ruedas con los parámetros eléctricos necesarios para lograr estas velocidades en los motores usados, se realizaron pruebas para evaluar el enrutamiento del robot móvil diferencial. Dicha relación se detalla en la figura 24.

Figura 24. Velocidad angular de las ruedas en grados/s.



Fuente: elaboración propia.

Como el rango de velocidades de cada rueda se estableció entre {-1; 1}, en representación de {0; 180} grados/s, se debe hallar la correspondencia con respecto a las velocidades que se ejecutan en los servo motores de giro continuo. Estas velocidades son: 1) Velocidad mínima en cierto sentido de giro, desde 1 grado hasta 90 grados disminuyendo velocidad cada vez que se acerca a 90 grados. 2) Si el motor está perfectamente calibrado logra detenerse en 90 grados, y desde 91 grados el motor va aumentando velocidad en el sentido contrario hasta llegar a 180 grados, donde se detiene. Todo esto en cuanto a la forma de programar las velocidades de los motores, y un ejemplo del código en Arduino se detalla en la figura 25.

Figura 25. Velocidades

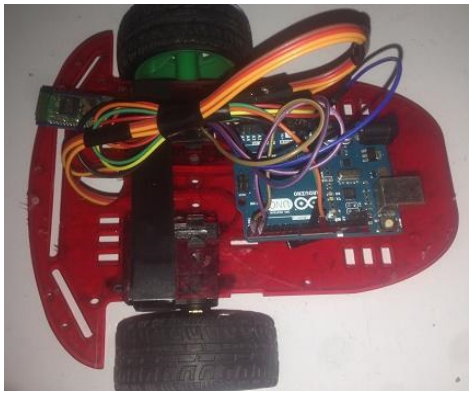
VELOCIDADES_

```
1  Servo VAderecha  ;
2  Servo VAizquierda;
3
4  void setup(){
5       VAderecha.attach(9)   ;
6       VAizquierda.attach(11);
7  }
8
9  void loop(){
10     //sentido de Giro 1
11     for(int i=0; i <= 90 ; i++){
12         VAderecha.write(i)  ;
13         VAizquierda.write(i);
14         delay(100);
15     }
16     for(int i=90; i <= 180 ; i++){
17         VAderecha.write(i)  ;
18         VAizquierda.write(i);
19         delay(100);
20     }
21 }
22 }
```

Fuente: elaboración propia.

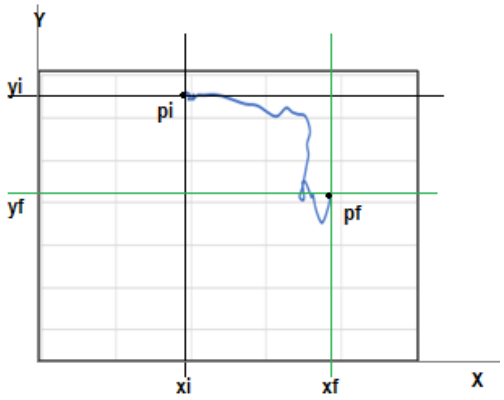
Una vez halladas las velocidades equivalentes se realizaron varias pruebas reales para evaluar el enrutamiento, empleando la plataforma móvil de la figura 26. En la figura 27 se muestra un ejemplo de la ruta entre la posición inicial (pi) y final (pf), en una de las pruebas realizadas, vista desde arriba.

Figura 26. Primera plataforma diferencial para las pruebas.



Fuente: elaboración propia.

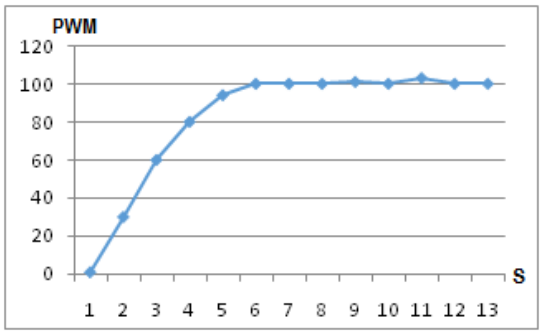
Figura 27. Ejemplo de una ruta del robot al detectar la ficha.



Fuente: elaboración propia.

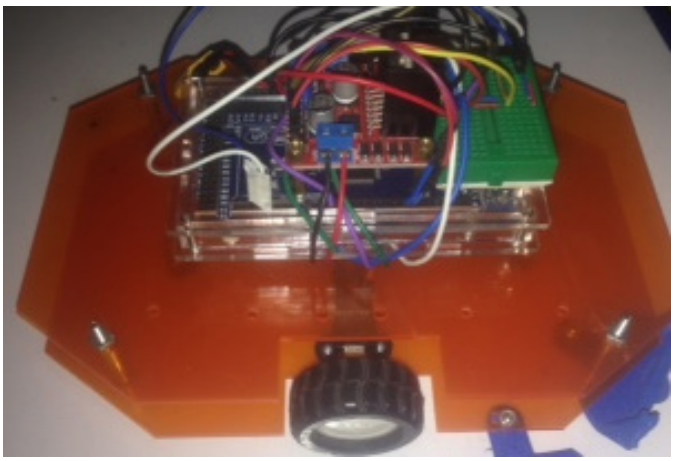
En la figura 28 se detalla el comportamiento del controlador PD difuso, en cuanto al manejo del PWM, y en la figura 29 se muestra la segunda plataforma diferencial con la que finalmente se logró el objetivo de enrutar el móvil hacia el disco.

Figura 28. Grafica del comportamiento del PD difuso



Fuente: elaboración propia.

Figura 29. Segunda plataforma diferencial con motores DC



Fuente: elaboración propia.

Conclusiones

El procesamiento de imágenes es la base para empezar con la correcta utilización e implementación de la visión artificial, lo cual se requirió en el móvil para el *Puck Collect Robot Challenge*. Esto es aplicable al procesamiento del video en tiempo real, puesto que en ciertos intervalos de tiempo se capturan fotogramas y se procesan para determinar la figura geométrica que está frente a la cámara, que área tiene, de qué color es, cuántos objetos hay, qué tan cerca están del robot, etc.

Una vez procesado el video, se usa un diseño de control difuso para lograr el movimiento del robot tanto con velocidad lineal como con velocidad angular. El prototipo usa las técnicas detalladas en este artículo para lograr todos los objetivos del proyecto. Con esto se espera que pueda obtenerse el resultado deseado independientemente de la estructura del robot. Como trabajo futuro se desea implementar un robot autónomo omnidireccional con ruedas *mecanum*, para validar su modelo cinemático y para evaluar la aplicabilidad de los controladores difusos acá reportados.

Referencias

Cyberbotics Ltd. (2016a). Webots User Guide: release 8.4.0, Lausanne, Switzerland.

Cyberbotics Ltd. (2016b). Webots Reference Manual: release 8.4.0, Lausanne, Switzerland.

Cyberbotics Ltd. (2016c). Webots (release 8.4.0): 3D mobile robot simulation software, PRO license. Lausanne, Switzerland.

Del Brío, B. y Molina, A. (2002). Redes neuronales y sistemas difusos. Bogotá: Alfaomega.

Ecured (2019a) Velocidad angular. Recuperado de https://www.ecured.cu/Velocidad_angular

Ecured (2019b). Velocidad lineal. Recuperado de: https://www.ecured.cu/Velocidad_lineal

Gracia, L. I. (2006). Modelado cinemático y control de robots móviles con ruedas. Tesis Doctoral, Universidad Politécnica de Valencia, Valencia, España. Recuperado de https://pdfs.semanticscholar.org/4e2c/9529ca8ab327fa369e016c4a2cb0b932d1ce.pdf

ITed Studio. (2010). HC-05 -Bluetooth to Serial Port Module. Recuperado de https://www.estudioelectronica.com/wp-content/uploads/2018/09/istd016A.pdf

Martínez, S. y Sisto, R. (2009). Control y comportamiento de robots omnidireccionales. Proyecto de grado, Estado del arte. Universidad de la República. Montevideo, Paraguay. Recuperado de https://www.fing.edu.uy/inco/grupos/mina/pGrado/easyrobots/doc/SOA.pdf

Maulana, E., Muslim, M., Hendrayawan, V. (2015) Inverse kinematic implementation of four-wheels mecanum drive mobile robot using stepper motors. International Seminar on Intelligent Technology and Its Applications (ISITIA) (pp. 51-56). IEEE.

Ollero, A. (2001). Robótica: manipuladores y robots móviles. Marcombo.

RobotChallenge (2019). Puck Collect Rules. Recuperado de http://www.robotchallenge.org.cn/doc/en/RC-PuckCollect.pdf

SanDoRobotics. (2018). Batería LiPo Turnigy 500mAh 2S 20C. Recuperado de https://sandorobotics.com/producto/t500-2s/

Tower-Pro. (2017). MG996R High Torque Metal Gear Dual Ball Bearing Servo. Recuperado de https://www.electronicoscaldas.com/datasheet/MG996R_Tower-Pro.pdf

Tzafestas, S. G. (2013). Introduction to mobile robot control. Elsevier.

Verma, P. (2017). Smartphone Controlled Robot. Recuperado de https://thestempedia.com/project/basic-differential-drive-robot/