Pontificia Universidad Javeriana

Facultad de Ingeniería
Carrera de Ingeniería Electrónica

Trabajo de Grado

VEHÍCULOS ELÉCTRICOS A ESCALA CONTROLADOS Y SIMULADOS DENTRO DE UN ENTORNO CONTROLADO.

Director:
Ing. Diego A. Patiño G.

Estudiantes:
José David Rengifo Fuentes
Jaime Felipe Reyes Müller

Abril – Noviembre 2013

Pontificia Universidad Javeriana

Facultad de Ingeniería
Carrera de Ingeniería Electrónica

Trabajo de Grado

RECTOR UNIVERSIDAD:	Padre Joaquín Sánchez García S.J
DECANO ACADÉMICO:	Ing. Jorge Luis Sánchez Téllez M.ScM.Ed
DECANO DEL MEDIO UNIVERSITADIO:	P. Antonio José Sarmiento Novoa, S.J.
DIRECTOR DE CARRERA:	Ing. Jairo Alberto Hurtado Londoño Ph.D
DIRECTOR DEL PROYECTO:	Ing. Diego Alejandro Patiño Guevara Ph.D

AGRADECIMIENTOS

De ante mano agradecemos a nuestras familias por estar siempre a nuestro lado, por darnos fuerza y compañía durante la elaboración de este trabajo de grado y durante todo el transcurso de nuestra carrera. También agradecemos a todos nuestros compañeros de trabajo que nos ayudaron e hicieron parte activa en este proyecto siendo nuestros pares académicos para la construcción de conocimiento conjunto, sin olvidar a todos nuestros amigos como acompañantes en este proceso. A nuestro director y nuestros asesores que nos guiaron hacia la consecución de nuestras metas planteándonos nuevos retos de los cuales obtuvimos nuevos conocimientos. Por último, a la Pontificia Universidad Javeriana por brindarnos una formación integral basada en valores y principios para ser ciudadanos activos de nuestra sociedad y del mundo.

CONTENIDO

LISTAS ESPECIALES

Lista de Figuras:

Lista de Tablas:

Lista de Gráficas:

INTRODUCCIÓN

Hoy en día, uno de los avances electrónicos que se encuentran en desarrollo son los vehículos eléctricos, pues tienen como meta reemplazar los vehículos tradicionales a gasolina para disminuir la emisión de gases de invernadero y emplear tecnologías verdes. Estos vehículos eléctricos son fabricados con el ánimo de emplear nuevas fuentes de energía para evitar la quema de combustibles fósiles, pero al ser eléctricos necesitan recargarse cada vez que su batería se agote. Si se considera el supuesto donde la mayoría de personas vuelven de sus rutinas diarias a sus casas y ponen a recargar su automóvil posiblemente a la misma hora sobrecargarían el sistema eléctrico, pues si se toma por ejemplo la ciudad de Bogotá D.C., la cual cuenta con alrededor de un millón trescientos mil carros particulares sin contar los vehículos públicos[1] y con 12 taxis eléctricos, todo este volumen de automotores eventualmente sobrecargarían el sistema eléctrico haciéndolo ineficiente y poco probable para su implementación.

Lo anterior permite pensar que el vehículo eléctrico es una gran posibilidad para un futuro cercano. Sin embargo, para darle solución a los posibles problemas que traerán se debe analizar las siguientes preguntas: ¿Qué pasaría con los sistemas eléctricos si los vehículos eléctricos se coordinan entre sí para recargarse en diferentes momentos y diferentes estaciones de recarga? Y además, ¿Si estos llegarán no solo a consumir energía sino a aportar a la red mediante fuentes de energías alternas?, ¿Podría ser esto una solución al problema planteado?

Un panorama que permite identificar estos interrogantes y plantearlos de forma adecuada, se fundamenta en el proyecto de recursos de energía móvil en redes de electricidad (MERGE) detallado en el artículo [1]: "*Introducing Electric Vehicles in the Microgrids Concept*".

Estos cuestionamientos pueden llegar a resolverse si analizamos y empleamos redes eléctricas basadas en el modelo de una red SMART GRID, como se muestra en la Figura 1, detallando los diferentes agentes que la conforman. Según el estudio realizado en Holanda y publicado en el artículo [2]: "*Network Impacts and CostSavings of Controlled EV Charging*", estas redes son capaces de identificar los principales agentes y coordinar de manera óptima los vehículos para que no haya posibilidad de sobrecargar la red eléctrica.

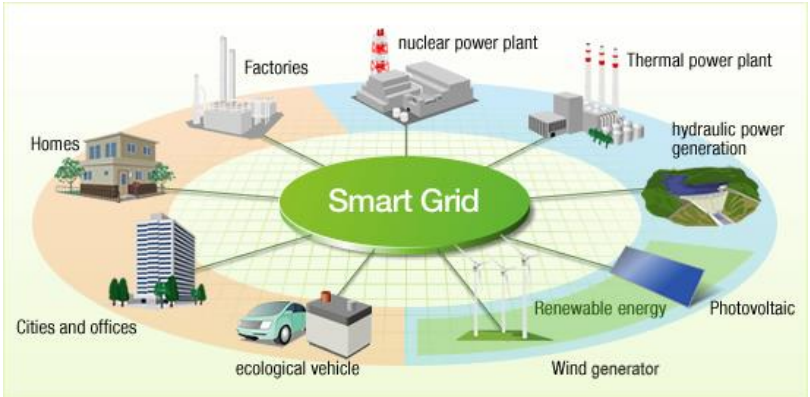


Figura 1. Diagrama de una red SMART GRID con vehículos eléctricos. Tomada de [3] para propósitos académicos.

1 Observatorio Ambiental de Bogotá. Tomado de la página web: http://oab.ambientebogota.gov.co/index.shtml?s=l&id=272. Fecha: 07 de abril de 2013

1. OBJETIVO GENERAL

Implementar una arquitectura de control para la navegación de vehículos móviles eléctricos dentro de un entorno controlado.

2. OBJETIVOS ESPECÍFICOS

- Adaptar o construir un vehículo eléctrico a escala utilizando diferentes partes (chasis, llantas, motores eléctricos de baja potencia, engranajes) que se encuentren en el mercado.
- Establecer una estrategia de control clásico (P, PD, PI o PID) para la velocidad y dirección del vehículo a escala.
- Implementar un sistema de visión por segmentación por color que permita determinar la posición de los vehículos dentro del entorno controlado.

3. MARCO CONCEPTUAL

Los vehículos eléctricos son autos que en su funcionamiento emplean un motor eléctrico para su propulsión, teniendo como fuente de alimentación baterías recargables (comúnmente llamadas celdas), las cuales gracias a la evolución en su composición ofrecen varias ventajas como: aumentar la eficiencia de la energía de un 21% con combustibles fósiles a un 62% con energía eléctrica; ser amigable con el medio ambiente a través de la implementación de fuentes alternas como energía solar, eólica, nuclear o energía suministrada por hidroeléctricas; y reducir la dependencia de los combustibles fósiles. Por otro lado, es necesario evaluar ciertos retos como: disminuir el tiempo de recarga, reducir los costos de las baterías, reducir el peso de los vehículos eléctricos y aumentar su rango de recorrido; ya que la solución de los mismos permite aprovechar de manera más eficiente los medios energéticos.

Como se mencionó anteriormente, dentro del proyecto de grado se tendrán en cuenta varios factores como la adaptación de los vehículos eléctricos, la estructura móvil, sus motores, un sistema sensitivo que será el control por segmentación de color, una batería como fuente de poder y un computador que controla cada uno de los componentes. En la Figura 2 se ilustra un diagrama de bloques con los elementos que hacen parte del proyecto de grado dentro de los cuales se encuentra la estación central que será un computador en el cual se ejecuta el programa Matlab, donde estarán programadas las funciones que permitan realizar el control del vehículo como la planeación de trayectoria a través de la información que el bloque de control por segmentación de color le brinda sobre la posición de los vehículos, un bloque de comunicación el cual se encargará de enviar los comandos de la estación central al prototipo del EV que lo integran sus respectivos componentes como la batería, el conversor y el controlador que se encargará de suministrar los comandos a los motores de velocidad (eje trasero) y dirección (llanta delantera).

Cabe aclarar que el proyecto se realizará bajo un entorno controlado que represente una Smart Grid, la cual será un plano de dos dimensiones con fondo en color negro y diferentes caminos en color blanco, esta dispondrá de un punto A que será el punto de partida y un punto B que será el punto de llegada para cada una de las trayectorias a seguir.

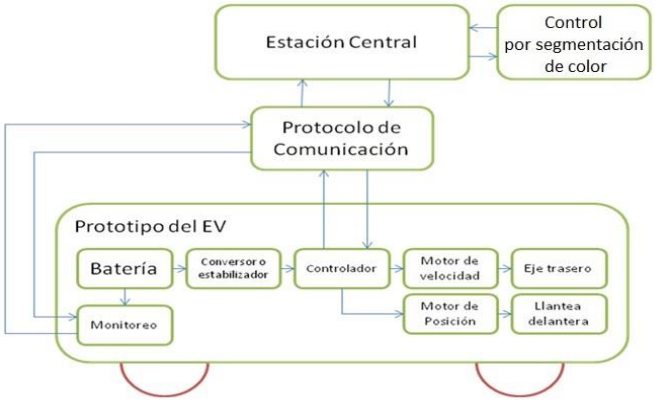


Figura 2. Diagrama de bloques del proyecto de grado.

3.1 MODELO DEL VEHÍCULO ELÉCTRICO A ESCALA

El modelo a escala a implementar será de tipo triciclo, el cual se compone de dos motores DC independientes (uno para cada una de las ruedas traseras), que proporcionarán la velocidad al vehículo y lo ayudarán en la dirección de manera diferencial; además de un servo motor con una rueda delantera giratoria que permitirá realizar cambios de dirección entre 0° y ± 45º, el cual no proporcionará tracción, sino que únicamente otorgará la dirección al vehículo.

Al ser un modelo a escala, se debe tener presente el modelo cinemático, el cual se obtiene realizando la extracción del modelo de un vehículo tipo triciclo y complementándolo con el modelo de un vehículo tipo Ackermann, el cual se diferencia con el anterior en la utilización de un eje con dos ruedas delanteras para su dirección.

Además se tendrán las siguientes consideraciones: el vehículo eléctrico a escala no realiza movimiento que implique deslizamiento o movimientos tipo holonómico, su tracción será trasera y diferencial, la dirección que proporciona la llanta delantera no afecta la velocidad en el modelo cinemático y el ángulo de giro máximo es de 45°. A partir de estas consideraciones y del modelo del vehículo triciclo que se muestra en la Figura 3, se obtiene el siguiente modelo cinemático:

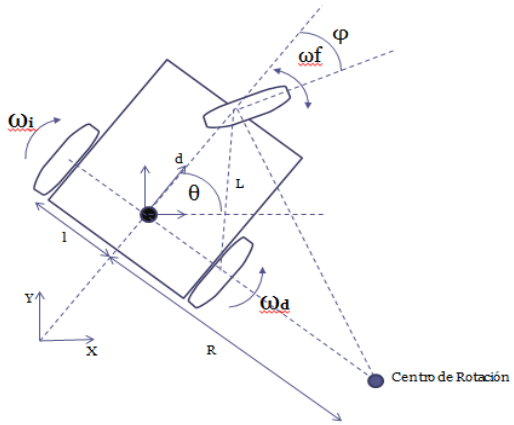


Figura 3. Análisis del modelo cinemático del vehículo eléctrico a escala.

En el cual ω_i es la velocidad angular de la rueda izquierda trasera; ω_d la velocidad angular de la rueda derecha trasera; ω_f la velocidad angular de la rueda delantera; R el radio del punto de masa al centro de rotación; l la distancia entre el punto de masa y el centro de cada llanta trasera; L la distancia entre las llantas traseras y la llanta delantera; θ el ángulo de la velocidad con respecto al eje x del plano de referencia; d es la distancia entre el punto de masa y la rueda delantera; φ el ángulo de giro de la llanta delantera y v, el vector de velocidad del vehículo eléctrico a escala. De este modelo podemos deducir las ecuaciones de la distancia entre el punto de masa y el centro de rotación (ecuación 1), la velocidad angular de la llanta delantera según el vector de velocidad y las distancias R y L (ecuación 2) y la distancia L (ecuación 3):

$$R = L * \tan\left(\frac{\pi}{2} - \varphi\right) \qquad (1)$$

$$\omega f = \frac{v}{(L^{2+}R^2)} \quad (2)$$

$$L = \sqrt{l^2 + d^2} \quad (3)$$

Partiendo del comportamiento del vehículo, las entradas del sistema serán: la velocidad del robot (v), y la velocidad angular (ω). El correspondiente modelo cinemático se caracteriza por las siguientes ecuaciones, donde sus estados son la posición en (x,y), el ángulo de v (θ), y el ángulo de giro (φ):

$$\dot{X} = v * \cos(\theta) \quad (4)$$

$$\dot{Y} = v * \text{sen}(\theta) \quad (5)$$

$$\dot{\theta} = \frac{v}{L} * \tan(\varphi) \quad (6)$$

$$\dot{\varphi} = \omega f \quad (7)$$

En su forma matricial:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \text{sen}(\theta) & 0 \\ \frac{\tan(\varphi)}{L} & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} v(t) \\ w(t) \end{bmatrix} \quad (8)$$

Adicionalmente, como la velocidad será de tipo diferencial, se deberá incluir el modelo dinámico del vehículo en las ecuaciones anteriores, mediante el análisis de las fuerzas ($\sum F = m * a$) que se ejercen en dicha velocidad y que será la sumatoria de la fuerza derecha y la izquierda (F_d, F_i) respectivamente: $\sum F = F_d + F_i$. Donde cada una de las fuerzas está dada por:

$$Fd = m * a = m * \dot{v_d} * (\cos(\theta)\,\hat{x} + sen(\theta)\hat{y}) \quad (9)$$

$$Fi = m * a = m * \dot{v_i} * (\cos(\theta)\,\hat{x} + sen(\theta)\hat{y}) \quad (10)$$

Conociendo que la velocidad será la suma de cada una de las velocidades: $v = v_i + v_d$, donde $v_i = r * \omega_i \;\; y \;\; v_d = r * \omega_d$, con r: radio de la llanta y l: distancia del punto de masa al centro de la llanta trasera. Al reemplazar en la (ecuación 8), se obtiene:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \frac{(\omega_d + \omega_i) * r}{2} \cos(\theta) & 0 \\ \frac{(\omega_d + \omega_i) * r}{2} \text{sen}(\theta) & 0 \\ \frac{(\omega_d - \omega_i) * r}{2 * l} & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} v(t) \\ w(t) \end{bmatrix} \quad (11)$$

Y,

$$v = \sqrt{\dot{x}^2 + \dot{y}^2} = \frac{(\omega_d + \omega_i) * r}{2} \; ; \; \omega = \frac{(\omega_d - \omega_i) * r}{2 * l} \qquad (12)$$

3.2 SUMINISTRO DE ENERGÍA

El Litio es un metal con gran potencial electroquímico, al ser uno de los metales más reactivos permite alcanzar una alta densidad de energía para aplicaciones que requieran un almacenamiento considerable. En la actualidad, este tipo de celdas no usan solo Litio, sino una combinación con otros elementos, que generan un compuesto no reactivo con el agua, brindando mayor seguridad [4].

La denominación de Li-Po se debe a que su composición es "Lithium-ion polymer" (Polímero de Litio), en donde se utiliza una matriz de iones de polímeros conductivos, para realizar la reacción con iones de Litio líquido [5]. Estas baterías se manejan en paquetes que contiene una o más celdas, dependiendo de este arreglo se puede determinar características como la Tensión (conexión en serie / paralelo) y su Capacidad (número de bloques en paralelo). Lo anterior se puede ver reflejado en las figuras 4 y 5.

Figura 4. Celda Li-Po, tomado de [6] con propósitos académicos.

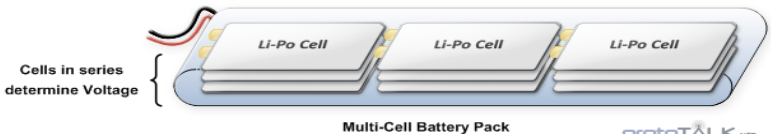


Figura 5. Grupo de celdas Li-Po, tomado de [6] con propósitos académicos.

3.2.1 Tensión de la batería

La tensión se determina según el número de celdas conectadas en serie, este valor se establece mediante un indicador o un valor numérico impreso. Según sea el caso se tiene una equivalencia para este valor según la Tabla 1.

Indicador	Tensión Nominal
1S	3.7V
2S	7.4V
3S	11.1V
4S	14.8V

Tabla 1. Indicador numérico batería Polímero de Litio, tomado de [6] con propósitos académicos.

Generalmente cada celda tiene una tensión nominal que abarca el rango de 3.7 V a 4.2 V, y una tensión promedio de 3.7 V. Dependiendo de su uso, estas empiezan a suministrar una tensión máxima de 4.2 V hasta una tensión de 3.4 V, valor en el cual la batería se agota. Si se sigue utilizando, la batería puede

llegar hasta una tensión de 3.0 V en donde el circuito de protección la desconecta [7]. Este comportamiento es visible en el perfil de descarga de la Figura 6.

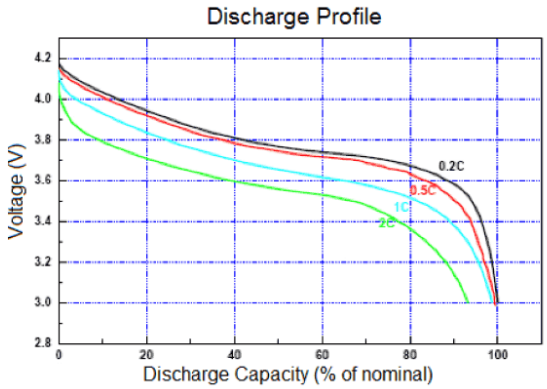


Figura 6. Perfil de descarga batería Polímero de Litio, tomado de [7] con propósitos académicos.

3.2.2 Capacidad de la Batería

La capacidad de la batería se basa en dos aspectos, el primero es la capacidad de una sola celda y el segundo es el número de bloques en paralelo. Para poder aumentar la capacidad de la batería, se debe por lo menos duplicar la cantidad de celdas, ya que al ir conectadas en paralelo se obtendría la suma de las corrientes de cada celda. Por tanto, sí se ponen dos celdas en paralelo se tendrá una corriente de 2 A que es igual a la suma de 1 A que entrega cada una de las dos celdas.

3.2.3 Máxima descarga continua de corriente

La máxima tasa de descarga continua que se puede obtener de una batería depende de la clasificación de estas, y a partir de esto se puede obtener la relación de cuántos amperios puede entregar al sistema sin sobrepasar el límite de temperatura. Para seleccionar la batería a emplear es importante saber cuánto es el consumo del sistema y su tasa de descarga ya que al emplear una batería de menor corriente a la calculada aumentaría considerablemente la temperatura y podría llegar a producir la combustión de la misma [7]. Para calcular esta tasa de descarga se emplea la siguiente ecuación:

$$\frac{Capacidad\ [mAh]}{1000} * C\ rating = \#\ Amperios \qquad (13)$$

3.3 CARACTERIZACIÓN MOTOR DC

Para plantear la función de transferencia que relacione el modelo eléctrico del motor con su modelo mecánico, es necesario conocer qué tipo de componentes integran un motor DC, debido a que esta relación permite sintonizar el control de velocidad. En la Figura 7, se ilustra el modelo eléctrico y mecánico del motor.

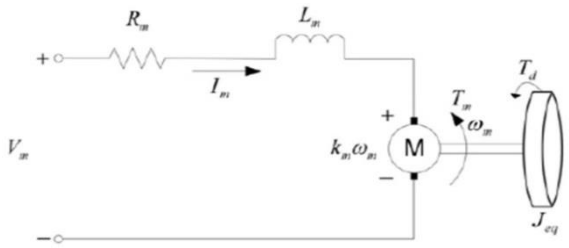


Figura 7. Modelo eléctrico y mecánico del motor**,** tomado de [8] con propósitos académicos.

Calculando la ecuación de malla del sistema, se obtiene que:

$$V_m - R_m I_m - L_m \dot{I}_m - K_m \omega_m = 0 \quad (14)$$

Con lo cual es posible estructurar el modelo que describe el motor DC, el cual está dado por:

$$G(s) = \frac{w(s)}{V_a(s)} = \frac{K}{\tau s + 1} * e^{-L*s} \qquad (15)$$

Donde K es la ganancia estática del motor, τ es la constante de tiempo electromecánica y L representa el retardo del motor. Aplicando el esquema de la Figura 8, se obtiene la relación entre la frecuencia y velocidad (RPM) según un determinado valor de tensión a la entrada.

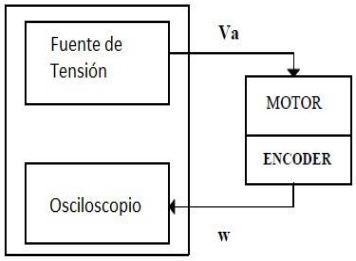


Figura 8. Esquema de conexión para la obtención de parámetros.

Realizando el análisis en estado estacionario, se obtienen las mediciones necesarias para la caracterización del motor DC (parámetros K_m y τ_m). Es necesario efectuar dicho análisis en dos estados: el primero requiere obtener la relación tensión Vs corriente cuando la velocidad angular es cero. La ecuación 16 permite obtener la resistencia de armadura del motor R_m, con la medición de corriente y el valor de la tensión de alimentación.

$$V_m - R_m I_m = 0 \qquad (16)$$

El segundo consiste en encontrar la constante de torque del motor, por lo tanto se debe realizar el análisis cuando la velocidad angular es diferente de cero en estado estacionario, ya que esto permite relacionar tanto la tensión, corriente y frecuencia obteniendo la siguiente ecuación:

$$K_m = \frac{V_m - R_m I_m}{w_m} = \frac{V_m - R_m I_m}{2 * \pi * f} \qquad (17)$$

Realizando el análisis de entrada paso al motor DC, se obtiene la curva característica de la respuesta de un sistema de primer orden, en el cual la constante de tiempo en lazo abierto τ_m será el tiempo que se demora la señal en conseguir el 63% del valor final.

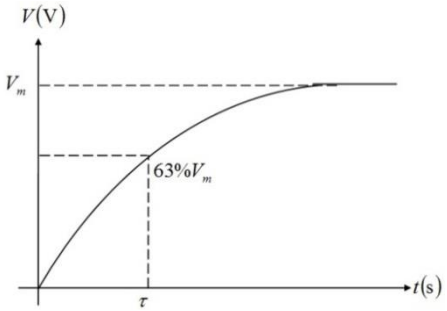


Figura 9. Constante de tiempo en lazo abierto.

Dada las características del motor a emplear, es necesario considerar la fricción que se produce en los ejes de la reducción, porque al realizar un control a bajas velocidades va a ser el parámetro que afecte la dinámica de control. Este parámetro se obtiene de las hojas de especificaciones, pero debido al momento de inercial total de todo el vehículo, hace que su influencia sea despreciable. Para evitar estos efectos en la dinámica del motor se debe escoger un punto de operación alejado de la zona inestable. Debido a lo anterior el modelo del motor DC se modifica obteniendo:

$$\mathrm{G(s)} = \frac{\mathrm{w(s)}}{\mathrm{V_a(s)}} = \frac{K}{\tau_m \mathrm{s} + 1} \qquad (18)$$

Donde,

$$K = \frac{1}{K_m} \qquad (19)$$

3.4 PROTOCOLO DE COMUNICACIÓN

3.4.1 Definición

La comunicación de dos o más dispositivos electrónicos se puede realizar a través del envío y recepción de información por medio de un canal no guiado regido bajo reglas y normas que entregan diferentes protocolos de comunicación, donde se tienen preestablecidos aspectos como la sincronización y la comunicación entre sí. Los protocolos más comunes que se pueden encontrar son Wifi, Bluetooth, Zigbee y 4G los cuales tienen sus propias características, ventajas y limitaciones. En este proyecto se empleará el protocolo de comunicación Bluetooth ya que es económico, tiene un alcance de 100 m y es el de mayor uso en términos comerciales lo que permite que el sistema se pueda conectar desde diversos dispositivos electrónicos.

3.4.2 Protocolo de Comunicación Bluetooth

La tecnología Bluetooth es una herramienta que permite la comunicación entre sí de diversos dispositivos para la transferencia bidireccional de datos y demás funciones según las necesidades del usuario. Al ser una tecnología inalámbrica permite su integración en dispositivos móviles, generando portabilidad sin tener la necesidad de conectarse mediante un medio cableado para la comunicación.

Bluetooth maneja la banda de 2.4 GHz ISM (Industrial, Scientific and Medical), con ancho de banda de 1 MHz [9]. Para la transmisión bidireccional de datos el canal se divide en intervalos de tiempo de 625 µs, los cuales dependiendo de la cantidad de paquetes, pueden hacer uso de un intervalo de tiempo como de 5 intervalos de tiempo consecutivos para la transmisión. En cuanto a la velocidad de transferencia, esta va ligada al número de usuarios de la red como también a su actividad (Estado). En la Figura 10, se ilustra el comportamiento y la velocidad de transferencia según la actividad y el número de usuarios:

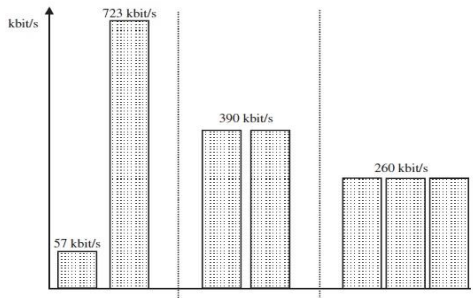


Figura 10. Velocidades de transferencia según la cantidad de usuarios. Tomado de [9] con propósitos académicos.

Este protocolo cuenta con 3 estándares para la cantidad de potencia requerida para la transmisión de datos, estos estándares se denominan Clases con las cuales se tiene una relación entre el alcance y el consumo, relacionados en la Tabla 1.

Clase	Potencia Máxima	Alcance
Clase 3	1 mW	1 m
Clase 2	2.5 mW	10 m
Clase 1	100 mW	100 m

Tabla 2. Indicador de potencia y alcance modulo Bluetooth, tomado de [9] con propósitos académicos

Todos los dispositivos Bluetooth se pueden comunicar entre sí sin importar la potencia de transmisión que cada uno tenga, sin embargo, en las conexiones bidireccionales el dispositivo con la más baja potencia de transmisión será el que limite el alcance de la red.

3.4.3 Tipo de paquetes

Para la transmisión de datos, este protocolo utiliza paquetes ACL (Asynchronous Connection-Less), los cuales consisten de 68 a 72 bits para acceso de código, 18 bits de cabecera y una carga útil entre 0 y 2744 bits. Antes de la transmisión de la cabecera se debe cifrar el paquete de datos, para tal fin se utiliza un algoritmo de corrección de error de 54 bits denominado FEC (Forward Error Correction). La Figura 7, permite ilustrar el paquete ACL.

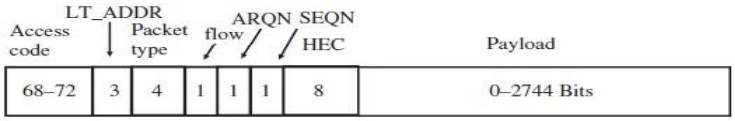


Figura 11. Paquete ACL, tomado de [9] con propósitos académicos.

En muchos tipos de aplicaciones no se utiliza el paquete ACL, debido a que no se garantiza el ancho de banda para la conexión, por lo tanto, este tipo de paquete de datos no serviría para la transmisión bidireccional en tiempo real. Para la transmisión en tiempo real se utilizan paquetes SCO (Synchronous Connection-Oriented), los cuales son intercambiados en intervalos fijos entre el Master y el Slave con un ancho de banda de 64 kbit/s. Dependiendo de la aplicación, se utiliza otro tipo de paquete denominado eSCO que mejora el mecanismo del SCO cuando se utiliza para aplicaciones de streaming. La velocidad de transferencia de datos se puede escoger durante el establecimiento de la conexión, la cual puede ser constante hasta de 288 kbit/s en modo full-dúplex.

3.4.4 Concepto de Piconet

Se denomina Piconet a una red Bluetooth en la cual varios dispositivos se comunican entre sí, cada una de estas redes utiliza un número de secuencia (Hopping sequence), con la cual se diferencian de las demás redes y permite tener varias de ellas en una misma área [9]. En una Piconet, se utiliza el concepto de Master / Slave, utilizando un dispositivo central "Master", el cual establece conexión hasta con 7 dispositivos esclavos para crear la red. Cada dispositivo puede ser Master o Slave, dependiendo si este inicia o no una red. La Figura 12, ilustra gráficamente el concepto anteriormente descrito:

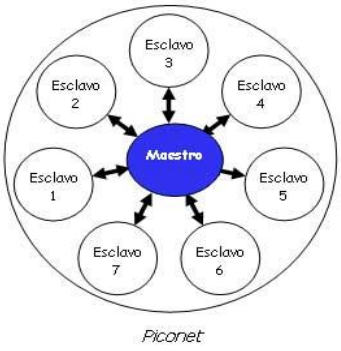


Figura 12. Concepto de red Bluetooth, tomado de [10] con propósitos académicos.

En este tipo de redes, el dispositivo central "Master" es el que controla el orden y la duración de la transferencia de datos de los dispositivos esclavos en el canal asignado. Para garantizar un canal a un dispositivo esclavo, el Master envía un paquete con el cual el Slave es identificado mediante una dirección de 3 bits ubicados en la cabecera del paquete, que se le asigna cuando se solicita la conexión. Si después de enviar dicho paquete el Master no recibe ningún dato de su contraparte, la transmisión se puede pausar hasta 800 intervalos de tiempo equivalente a 0.5 segundos con el fin de conservar energía. Dependiendo de la aplicación y la situación entre los dispositivos de una Piconet, estos pueden cambiar de roles utilizando el método de "Master – Slave role switch" [9].

3.5 TÉCNICAS DE CONTROL CLÁSICAS

Para el diseño del controlador que mantenga una velocidad deseada y constante sobre el vehículo, se deben tener en cuenta varios criterios. El primer criterio se basa en el análisis del comportamiento de los motores utilizados, si estos motores se caracterizan por una función de transferencia rápida (ganancia alta) y su tiempo muerto es despreciable, la implementación del controlador solo requerirá que se elimine el error en estado estacionario, por tanto se debe optar por un control que contenga una acción integral como un controlador PI o PID para una mayor precisión.

El segundo criterio es el orden de la dinámica de la planta a controlar, si esta es de segundo orden también será adecuado realizar un control proporcional integral o un PID.

El último criterio será fijar un tiempo de establecimiento, en el cual se debe considerar el máximo sobre pico que tendrá la respuesta del sistema, el tiempo de subida y el tiempo de duración del sobre pico. Las características de estos dos tipos de controladores mencionados anteriormente son:

El controlador proporcional integral (PI) hace que el tiempo integral se ajuste a la acción dependiendo de su función de transferencia: $Cpi(s) = Kp * \left(1 + \frac{1}{Ti*s}\right)$. Para su funcionamiento es indispensable que la señal de error sea diferente a la señal del estado deseado, y por lo tanto, esta acción anulará el error de offset, eliminando también el error en estado estacionario.

El controlador Proporcional Integral Derivativo (PID) reúne las ventajas de las tres acciones: proporcional, integral y derivativa, por lo tanto, este estabiliza de mejor manera el sistema pero lo hace más complejo. Su función de transferencia es: $Cpid(S) = Kp * \left(1 + \frac{1}{Ti*S} + Td * s\right)$.

3.6 SISTEMA DE VISIÓN POR SEGMENTACIÓN POR COLOR

Las técnicas de segmentación son muy utilizadas en el tratamiento de imágenes digitales, debido a que se basan en procedimientos de etiquetado para la determinación de contornos o regiones de la imagen a través de la información de la intensidad y/o la información espacial mediante procesos determinísticos o estocásticos [11]. Previamente, es necesario realizar un proceso de Binarización en la imagen digital, ya que esta convierte una imagen de niveles de grises o a color, en una imagen a blanco y negro, donde los pixeles negros corresponden al fondo y los pixeles blancos corresponden al objeto [12]. A través de una segmentación simple, se puede determinar la clasificación de los pixeles en las regiones homogéneas.

3.6.1 Espacio RGB

Es el espacio de color extendido y el utilizado por la gran mayoría de cámaras de video y fotográficas para construir una imagen de color. Este espacio RGB se representa mediante un cubo, donde un color viene definido por una mezcla de valores de intensidad de tres colores primarios: Rojo, Verde y Azul (Red, Green, Blue como su nombre lo indica). En una imagen convencional, el color viene descrito a través de 3 matrices que representan los tres colores primarios, en donde el color negro se obtiene de (R=0, G=0, B=0), y el color blanco se representa (R=255, G=255, B=255). Por lo tanto estas combinaciones de colores van desde un valor de 0 hasta 255. La gama acromática de escala de grises está representada por la diagonal del cubo [13]. La Figura 13 representa el concepto anterior:

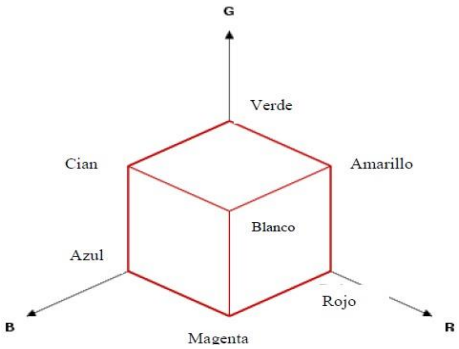


Figura 13. Espacio RGB, tomado de [13] con propósitos académicos.

3.6.2 Planificador de trayectorias

Para la construcción de la ruta a seguir por el vehículo, es necesario tener un método de planificación que permita el reconocimiento de objetos con sus características espaciales, además de definir aspectos como el costo derivado de cada tramo de la trayectoria con el fin de determinar la trayectoria con menor costo. Para poder obtener lo anterior es necesario emplear un planificador global.

- Planificación global: Construye o planifica la ruta que lleve al robot a cada una de los puntos determinados por el control de misión según las especificaciones del problema que debe resolverse. Esta planificación es una aproximación al camino final que se va a seguir, ya que en la realización de esta acción no se consideran los detalles del entorno local al vehículo [14].

- C-space: Es el espacio de configuraciones (costos) el cual se define antes de empezar la planificación global o local [14].

3.6.3 Técnicas de planificación

Para determinar la ruta desde un punto A hacia un punto B, es necesario emplear un algoritmo que permita la identificación de los puntos necesarios para cumplir esta trayectoria, por lo tanto, para el análisis de estos es necesario emplear lo siguiente:

- Mapas probabilísticos (Probabilistic Road maps): El Algoritmo A* es uno de los más empleados debido a que crea una estructura para la representación del mapa, en el cual cada nodo del mapa contiene una coordenada X y una coordenada Y para su ubicación dentro de la estructura matricial, además de un peso asociado [14].
  A través del mapa de pesos, se le otorga a cada celda que compone la imagen un valor que oscila entre 0 e ∞, con los cuales se restringen las áreas libres por las cuales se desea que el algoritmo evalué la ruta más corta y con menor costos hasta su meta. En la Figura 14 muestra con mayor claridad el mapa de pesos de un entorno:

Figura 14. Mapa de pesos de un entorno con punto de inicio y fin. Tomado de [14] con propósitos académicos.

3.6.4 Perfil de velocidad

Para la generación de un perfil de velocidad se utiliza el concepto de segmento de recta en el espacio cartesiano, ya que este permite expresar cada una de las coordenadas espaciales en función del tiempo, partiendo de la ecuación de la recta en el espacio utilizando ecuaciones paramétricas [14]. Esta relación se puede ver en la siguiente ecuación:

$$f(x, y, z) = \lambda(t)(p1 - p0) + p0 \qquad (20)$$

Donde p0 y p1, son los puntos de inicio y fin respectivamente, $\lambda(t)$ representa el tiempo de muestreo, el cual va ligado al control de la velocidad de la recta. Este resultado permite determinar el modelo del perfil de velocidad, el cual va estar compuesto por un tramo de aceleración, velocidad constante y desaceleración como se ve en la Figura 15:

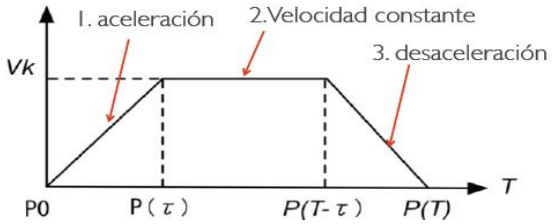


Figura 15. Perfil de velocidad trapezoidal. Tomado de [14] con propósitos académicos.

Utilizando las ecuaciones paramétricas se realiza la extracción matemática de la distancia entre P0 y P1, la cual equivale al área bajo la curva del perfil de velocidad, donde:

$$|P1 - P0| = \frac{1}{2} a\tau^2 + V_k(T - \tau) \qquad (21)$$

La aceleración se puede expresar como:

$$a = \frac{V_k}{\tau} \qquad (22)$$

Utilizando la representación a tramos de la función del perfil de velocidad, se puede obtener las funciones para cada intervalo de la trayectoria como se relaciona en las siguientes ecuaciónes:

$$f(t) = \begin{cases} \frac{1}{\|P(\tau) - P(0)\|}\left(\frac{1}{2}a\tau^2\right)\left(P(\tau) - P(0)\right) + P(0) & 0 \le t \le \tau \quad (23) \\ \frac{V_k \,.\, t}{\|P(T-\tau) - P(\tau)\|}\left(P(T-\tau) - P(\tau)\right) + P(\tau) & \tau \le t \le T - \tau \quad (24) \\ \frac{1}{\|P(T) - P(T-\tau)\|}\left(\frac{1}{2}a\tau^2 + V_k \,.\, t\right)\left(P(T) - P(T-\tau)\right) + P(T-\tau) & T - \tau \le t \le T \quad (25) \end{cases}$$

Donde la ecuación 23 representa el tramo acelerado, la ecuación 24 el tramo de velocidad constante y la ecuación 25 el tramo desacelerado. Para completar el análisis del perfil de velocidad se debe tener en cuenta los puntos intermedios $P(\tau)$ y $P(T-\tau)$, los cuales se pueden hallar mediante las siguientes ecuaciones:

$$P(\tau)_{x,y,z} = \left(\frac{1}{2}a\tau^2\right)\frac{P(\tau)_{x,y,z} - P(0)_{x,y,z}}{\|P(T) - P(0)\|} + P(0)_{x,y,z} \qquad (26)$$

$$P(T-\tau) = \frac{V_k(T-2\tau)}{\|P(T) - P(0)\|}P(T)_{x,y,z} - P(0)_{x,y,z} + P(\tau)_{x,y,z} \qquad (27)$$

Cabe aclarar que el V_k es la velocidad final a la cual se quiere que el vehículo se mueva, por lo tanto, a partir de la misma se puede determinar la duración total 'T' del recorrido agregando lo siguiente:

$$\tau = T.pt \qquad (28)$$

$$T = \frac{\|P(T) - P(0)\|}{V_k(1 - pt)} \qquad (29)$$

$$n = \frac{T}{tiempo\_muestreo} \qquad (30)$$

Donde 'pt' se define como el porcentaje de aceleración del perfil de velocidad, el cual lo define cada usuario; 'n' es el número de sub-puntos que se desea en la trayectoria los cuales equivalen a diferentes posiciones en el segmento del recorrido desde el punto de inicio hasta el final.

4. ESPECIFICACIONES

4.1 Diseño del vehículo eléctrico a escala

El vehículo eléctrico a escala tendrá unas dimensiones de 8cm de ancho x 8cm de largo, con la aclaración de que no se tiene un límite en altura y su peso máximo será de 600 g. Por otro lado los demás componentes deben cumplir con lo siguiente:

Dos motores DC (Micro Metal Gearmotor) que le otorgarán velocidad al vehículo con las siguientes especificaciones:

- Tensión de alimentación: 1 V a 6 V.
- Velocidad sin carga: 60 rpm.
- Corriente sin carga: 70 mA.
- Corriente máxima: 400 mA.
- Torque máximo: 0.19 kg*m.

Un servo motor (SG 90) que es el encargado de la dirección del robot el cual tendrá:

- Tensión de alimentación: 2 V a 6 V.
- Torque sin carga: mayor a 1 kg*cm.
- Velocidad de operación: mayor a 0.12 s cada 60°.

Un microcontrolador Arduino Micro caracterizado por:

- Tensión de operación: 5 V.
- Tensión de entrada (Recomendado): 7 V a 12 V.
- Tensión de entrada (Límite): 6 V a 20 V.
- Pines I/O Digitales: 20.
- Canales PWM: 7.
- Canales para entradas análogas: 12.
- Corriente DC por pin I/O: 40 mA.
- Corriente DC por pin 3.3V: 50 mA.
- Memoria Flash: 32 kB.
- SRAM: 2.5 kB.
- Frecuencia del reloj: 16 MHz.

Una batería de Li-Po que suministra energía a todo el vehículo eléctrico a escala con una tensión nominal de 3.7 V y una corriente de 1100 mAh.

Contará además con un módulo Bluetooth (BlueSMiRF Silver) que tendrá la función de Slave para comunicarse con la estación central especificado por:

- Radio Frecuencia: 2.4GHz Clase 2
- Versión: Bluetooth v 2.0 en adelante
- Rango de alcance: Superior a 10 m
- Velocidad de transferencia: Superior a 1 Mbps
- Comunicación UART y USB

También se dispondrá de un elevador de tensión BOOST que suministrará una tensión constante de 5 V con referencia PNT04050C y una etapa de potencia realizada con un puente H de referencia L293D, el cual aumentará la corriente de salida de los pines del microcontrolador, para la alimentación de los motores DC.

Además se emplearán llantas de POLOLU CORP las cuales contienen en su diseño el dispositivo giratorio del Encoder, además de las adaptaciones para los ejes de los motores a emplear. Estas llantas están elaboradas en plástico y caucho, y tienen un ancho de 1.8 cm y radio de 2.1 cm.

4.2 Maqueta

El entorno controlado se caracteriza por ser una maqueta de dos dimensiones: 2.43 m de largo y 1.01 m de ancho. Las vías están representadas en color blanco y cada vía tiene un ancho de 14 cm, con fondo de color negro. Esta maqueta se expone en la figura 16.

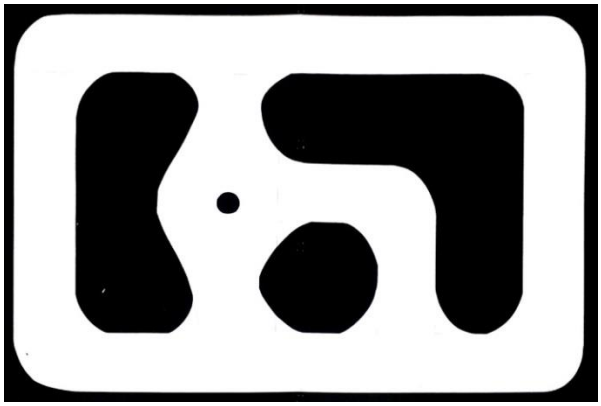


Figura 16. Maqueta.

Esta maqueta representará una de las etapas donde se implementará la MICRO SMART GRID que se encuentra en desarrollo por estudiantes de la Pontificia Universidad Javeriana dentro del proyecto SILICE III. Esta etapa, la cual se expone en la figura 17, estará ubicada entre la Av. El Dorado, la Av. La Esperanza, la Carrera. 68 y la Av. Boyacá y además se exponen los principales sitios turísticos e importantes dentro de esta zona.

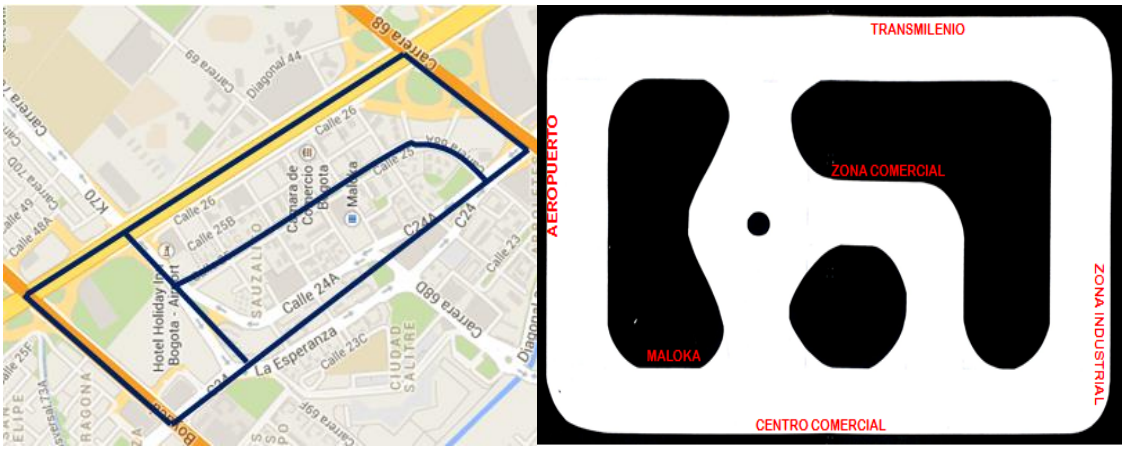


Figura 17. Zona de referencia de la SMART GRID.

4.3 Control por Segmentación de Color

Para realizar el seguimiento de los vehículos y su detección por color se requiere que el sistema de detección cumpla con los siguientes parámetros:

- Detección colores primarios y uno adicional: Rojo, Verde, Azul y Amarillo
- Contorno mínimo del objeto (en la imagen): mayor a 60 pixeles
- Detección de áreas y centroides
- Grabación en intervalos definido por el usuario

Además de una cámara capaz de conectarse con el equipo central que se va a tener y que tenga una buena calidad de imagen, es por esto que se empleará una cámara Logitech con las siguientes especificaciones:

- Referencia: Logitech HD Webcam C525
- Captura de vídeo HD: Hasta 1280 x 720 píxeles.
- Enfoque automático.
- Fotos: Hasta 8 megapíxeles.
- Certificación USB 2.0 de alta velocidad.

Además se contará con un kit de luces (Kino Flo interview kit y dos luces Kino Flo Celeb 200) que iluminen la maqueta para evitar algún tipo de problema en la detección de los colores, estas luces realzan el contraste de los colores dentro del entorno controlado, garantizando que el color empleado en lo vehículos a escala si estén de acorde con los colores del código RGB.

4.4 Controlador PI de velocidad.

Este controlador se encargará de regular la velocidad del vehículo eléctrico a escala dependiendo de los parámetros que se tomen para su diseño y sintonización. Para realizar este control se toma la decisión de realizar un controlador PI para simplificar la arquitectura y eliminar el error en estado estacionario que se encontró a partir del análisis del comportamiento del motor ya que este cuenta con una ganancia alta y no necesita un control predictivo. Por esta razón, se toma la respuesta de un sistema de segundo orden con entrada paso de tipo sobre amortiguado, definiendo ε=0.4 para que la respuesta no tenga un sobre pico elevado; un tiempo de subida de 0.25 s; una constante de tiempo en lazo abierto (τ_m) de 25.4 μs y una ganancia del motor de 15.129. Estos dos últimos parámetros se calculan en el enciso: Desarrollo del proyecto.

Con los parámetros anteriores, se encuentra también el máximo sobre pico de 22,3% y una frecuencia de oscilación de 40 Hz.

5. DESARROLLO DEL PROYECTO

Para el desarrollo del proyecto de grado es necesario comprender el funcionamiento de cada uno de los módulos, ya que mediante la familiarización de términos y funciones se puede tener una amplia perspectiva de los propósitos que se quieren cumplir.

5.1 Construcción del vehículo.

Luego de analizar los componentes que integrarán al prototipo a escala, se determinó la construcción de una estructura para el robot con la cual se pueda unir todas sus partes. Es por esta razón que se realizó la construcción del chasis, 2 planos intermedios, el eje para el servomotor y una parte superior la cual contiene dos círculos del mismo radio y un diseño a libre elección. Para realizar la elaboración de estas piezas, se emplea una impresora 3D la cual permite fabricar los componentes en plástico, siguiendo dos etapas de construcción:

La primera consiste en el modelado 3D de las piezas que compone el vehículo, esta se efectúa utilizando el software SolidWorks, el cual permite realizar todo tipo de representación 3D. Al elaborar los elementos se determinan características como el grosor, las dimensiones y el tipo de material, estableciendo su ubicación en la estructura final. Una representación de lo mencionado anteriormente se realiza en la Figura 18, donde se disponen los elementos y su organización:

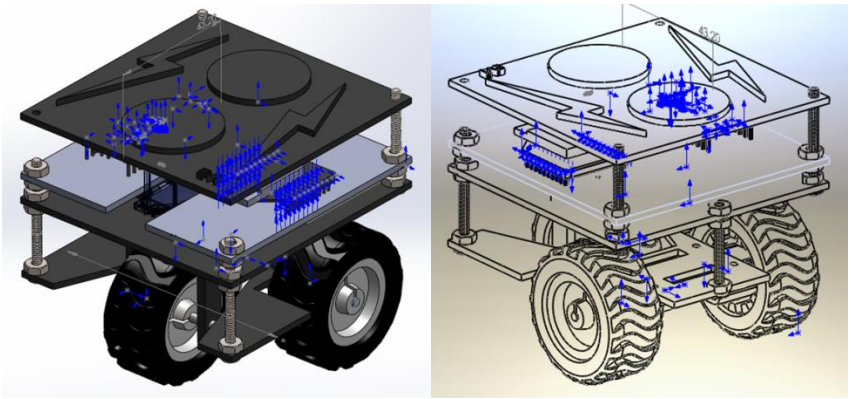


Figura 18. CAD del prototipo del Vehículo.

Por último, para comenzar el proceso de impresión es necesario exportar los archivos finales del modelo 3D al software de la impresora 3D, en donde se ubican las piezas según su tamaño y se almacenan en la memoria SD para su impresión.

La impresora 3D empleada es de referencia: Maker-Bot: Replicator 2; la cual fue adquirida por el departamento de electrónica durante el presente año. La Figura 19 es el resultado de la construcción de las piezas del prototipo y su montaje.

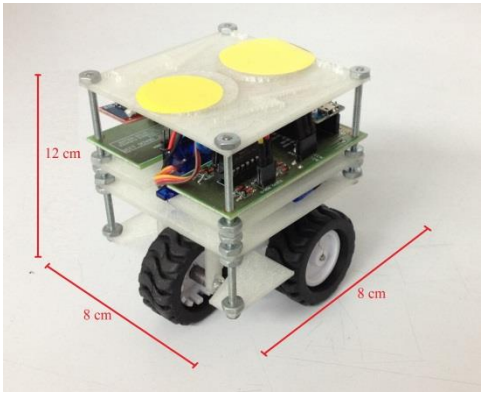


Figura 19. Resultado final del prototipo del Vehículo.

5.2 Elección de la Batería.

El consumo que genera todo el sistema en términos de corriente es uno de los grandes criterios que se deben tener en cuenta, debido a que las baterías de Polímero de Litio vienen en capacidades desde 180 mAh a 5.4 Ah, abarcando una amplia gama de opciones para lo cual la elección debe ser precisa, pues se busca un buen rendimiento además de tener un tamaño compacto.

Al hacer un balance de todos los componentes del proyecto se encontró que su consumo individual de corriente es de la siguiente manera:

Descripción	Motor DC	Servo Motor	Bluetooth	Encoder	Microcontrolador
$I_{con\ carga}$	600 mA	300 mA	-	-	-
$I_{sin\ carga}$	50 mA	-	-	-	-
$I_{promedio}$	-	-	16.5 mA a 35 mA	14 mA	120 mA a 300 mA

Tabla 3. Consumo de corriente de los componentes del proyecto.

A partir de los datos anteriores se realiza el cálculo de la corriente requerida teniendo en cuenta los dos posibles estados del vehículo:

- Consumo de corriente de arranque: 1.849 A
- Consumo de corriente en movimiento: 749 mA

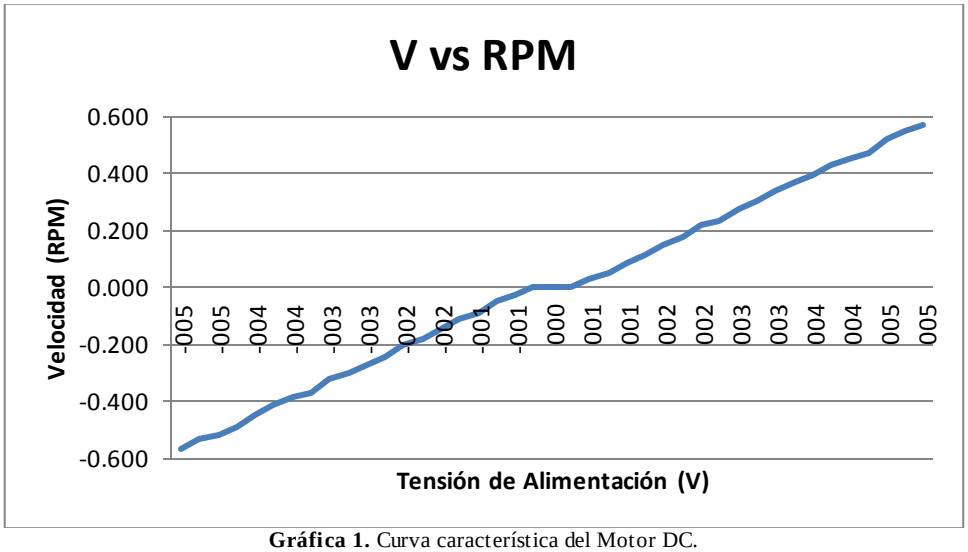
Por lo tanto la capacidad de la batería debe ser mayor a 800 mAh. Como aclaración, no siempre se tendrá un consumo elevado de corriente ni el pico de corriente de 1.85 A tendrá una duración extensa, por lo que este pico elevado de corriente solo se da cuando hay la transición a movimiento. Además, la máxima corriente que se le puede entregar al sistema sin sobrecargar la batería es de: 2.2 A.

Otro dato que se requiere para analizar el comportamiento de la batería es la duración de esta. Para este caso es de 1.5 horas a máxima carga en movimiento. Este cálculo se realizó según la siguiente ecuación:

$$Vida\ de\ la\ Bateria = \frac{capacidad\ de\ la\ batería}{consumo\ de\ corriente\ del\ sistema} = \frac{1100\ mAh}{749\ mA} = 1.5\ h \qquad (31)$$

5.3 Caracterización del motor DC.

La caracterización del motor DC se centra en hallar la curva característica que relaciona la velocidad con la tensión aplicada además de su función de transferencia. Realizando las pruebas correspondientes donde se mide la tensión de alimentación al motor y su velocidad otorgada por los respectivos Encoders, se obtiene la siguiente curva característica:



Gráfica 1. Curva característica del Motor DC.

Dados los resultados de la Grafica 1, se puede determinar la función de transferencia donde la entrada es el voltaje y la salida la velocidad angular de la siguiente manera:

$$\frac{\omega(s)}{v(s)} = \frac{K_m}{J_{eq} * R_m * s + {K_m}^2} \qquad (32)$$

Donde K = 1/Km = 15.129. Este resultado depende de la resistencia del motor y la frecuencia de los motores en cada una de las tensiones suministradas con base en el procedimiento que se detalló en el marco teórico. La resistencia de armadura del motor R_m es igual a 11.03 Ω, se obtiene a partir de la tensión de alimentación y la corriente que circula por el motor. Otro parámetro esencial es el cálculo del momento de inercia Jeq = 0.000259 kg*m^2, el cual equivale a la suma de los momentos de inercia de cada componente del sistema.

5.4 Diseño del controlador PI para la velocidad del vehículo.

Para el diseño del controlador PI se necesita conocer la función de transferencia de lazo cerrado con entrada paso, por lo que la Figura 20 permite visualizar el diagrama en bloques del proceso a realizar. Hay que tener en cuenta que el bloque F y el bloque H tienen valor igual a 1 para ser una retroalimentación unitaria, y no se tiene ninguna señal de disturbio (du=dy=0). Por otro lado, C es el bloque del controlador PI y G es la función de transferencia del motor.

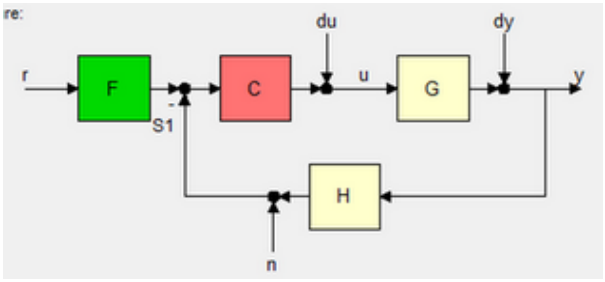


Figura 20. Controlador PI en lazo cerrado.

Con la ecuación (33) de lazo cerrado, la función de transferencia del controlador PI y el motor DC se puede despejar el denominador de la representación de la Figura 20, obteniendo de esta manera una ecuación característica de segundo orden, con la cual se pueden obtener los parámetros característicos del controlador PI:

$$Mc = \frac{PI*Motor}{1+PI*Motor} = \frac{kp*\left(\frac{1}{Ti*s}+1\right)*\frac{K}{\tau_m+1}}{1+kp*\left(\frac{1}{Ti*s}+1\right)*\frac{K}{\tau_m+1}} \qquad (33)$$

$$1 + kp * \left(\frac{1}{Ti * s} + 1\right) * \frac{K}{\tau_m + 1} = s^2 + 2 * \varepsilon * \omega_n * s + {\omega_n}^2 \qquad (34)$$

A partir del diagrama de polos y ceros de la Figura 21, se pueden interpretar los resultados obtenidos, teniendo en cuenta que el comportamiento del motor sin controlador cuenta con un polo bastante lejano al eje imaginario, por lo tanto, fue necesario emplear un cero en la parte positiva del eje real para disminuir su efecto.

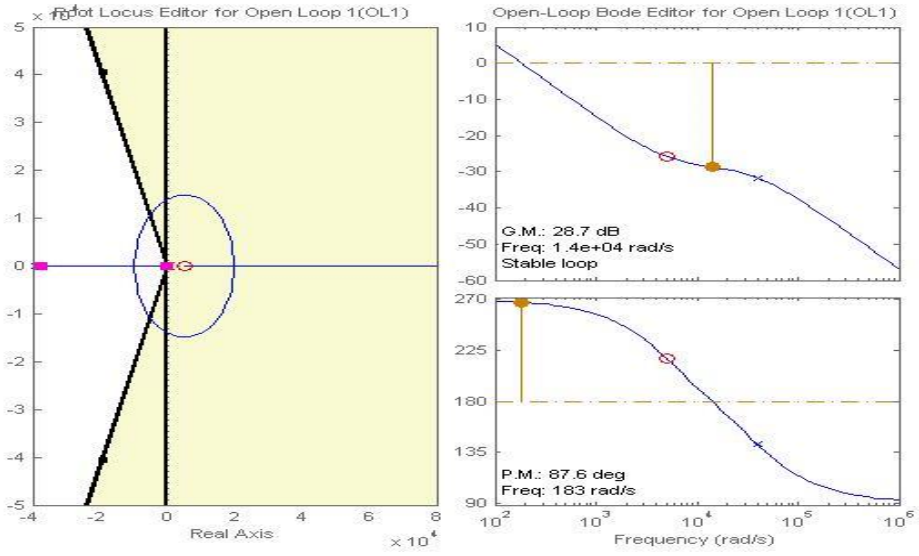


Figura 21. Diagrama de polos y ceros de todo el sistema.

Realizando el ajuste respectivo y la sintonización del control PI según los parámetros especificados anteriormente se obtiene un bloque de control (Ecuación 35) con una constante proporcional de un valor menor con respecto a la parte integral, por lo que la parte integral será la que elimine el error de estado estacionario:

$$c = -0{,}002416 + \frac{12.08}{s} \qquad (35)$$

Como resultado de lo anterior, se realiza la verificación del comportamiento del controlador PI, observando la salida del sistema y la señal de control que entra al motor DC, donde se comprueba los criterios de sintonización verificando los tiempos y que el máximo sobre pico no sea mayor a 22,3%. Donde se tiene la siguiente respuesta:

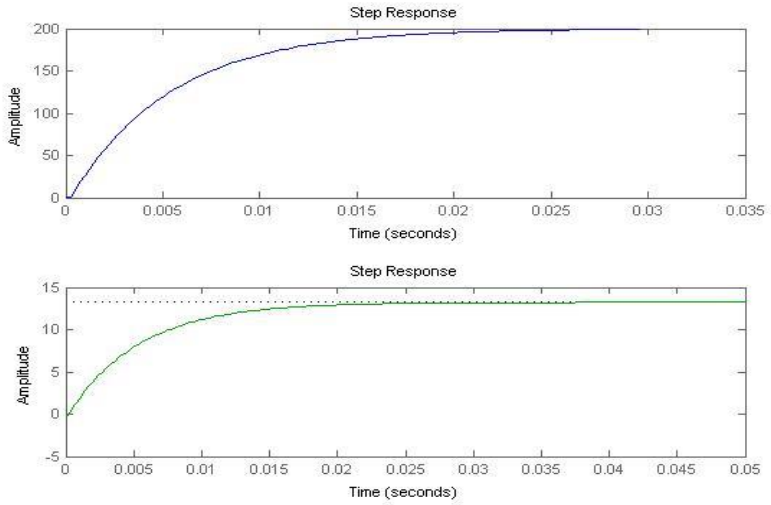


Figura 22. Respuesta a entrada paso. Señal de salida y control, respectivamente.

5.5 Control por visión / Segmentación por Color.

Para controlar los vehículos eléctricos a escala posicionados dentro de la maqueta es necesario tener los componentes cruciales del entorno, debido a que sobre estos se definen varias especificaciones como la velocidad, el ángulo de la trayectoria y el tiempo de recorrido que se involucran directamente en la tarea de navegación, partiendo del hecho que estas tareas se pueden realizar de forma separada, pero deben tener una misma estructura que permita el control básico de la navegación del vehículo. Una representación de dicha estructura se puede observar en la Figura 23.

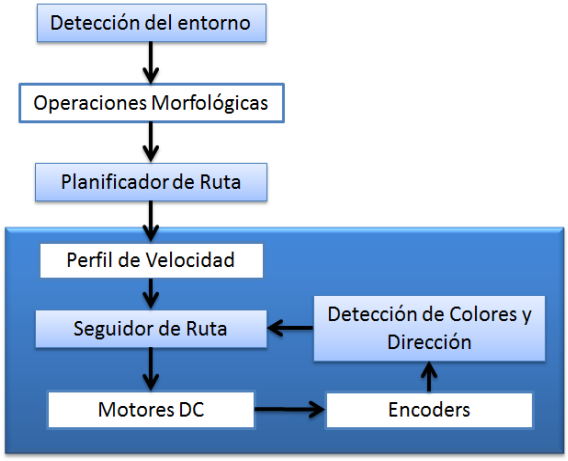


Figura 23. Estructura de control de navegación.

La detección del entorno se realiza mediante una cámara de video que permite la representación digital de la imagen en el espacio RGB. A partir de un proceso de Binarización, esta imagen se representa de forma matricial, lo cual permite tener una abstracción más adecuada de las características que se quieren referenciar. Para nuestro caso, estas características se basan en la detección de contornos y líneas las cuales se pueden obtener con el uso de operaciones morfológicas en la imagen binaria, en donde se descompone las líneas adyacentes a las vías de la maqueta.

Siguiendo en la línea de este proceso, uno de los algoritmos de planificación de rutas más común y utilizado para la navegación de robots móviles se basa en métodos de mapas probabilísticos. Empleando un algoritmo como el A* se pueden definir la matriz de costos y pesos de cada celda de la imagen. Una representación de este concepto se muestra en la Figura 24.

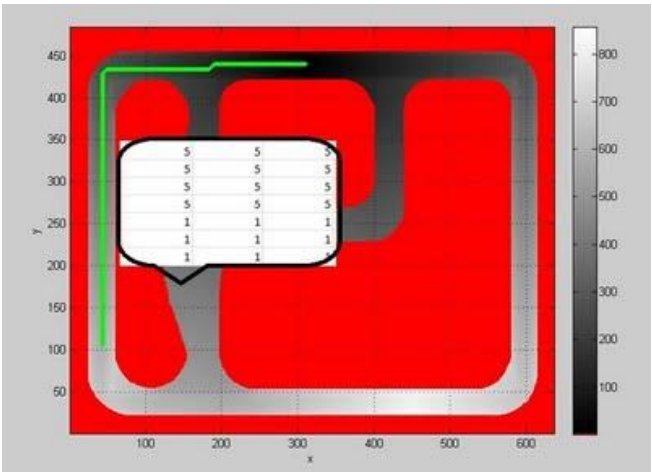


Figura 24. Matriz de costos dentro de la maqueta.

Con la representación matricial del costo y peso asociado a cada celda, se define que las áreas contiguas a las vías en la maqueta deben tener un peso elevado debido a que serán zonas restringidas en el recorrido del vehículo. Por lo tanto, se puntualizó que su peso y costo tenga un valor infinito, lo cual se puede ver con claridad en la Figura 24, donde estas zonas son de color rojo.

Una vez representado el límite por donde se quiere que el algoritmo calcule la ruta de menor costo, se debe tener en cuenta que las vías asociadas a la trayectoria van a tener un costo igual a '1', por lo que otra recomendación que se tuvo en cuenta fue la delimitación de las vías internas de la maqueta, esto se realiza con el ánimo de que el vehículo siempre circule por la zona central de la vía dejando un margen considerable entre el límite del chasis y la terminación de la vía. Haciendo referencia a lo anterior. en la Figura 24 se puede ver un recuadro con una tabla de '1's y '5's, donde el peso y costo asociado a la zona central es de '1', mientras que las zonas que delimitan la vía van a tener el peso y costo de '5', lo que el algoritmo interpreta como posible ruta pero siempre tendrá a calcular el camino que tenga el costo y peso de '1'.

Otro elemento que integra la estructura del control del vehículo a escala es el perfil de velocidades, en el cual se definen parámetros como: la posición inicial y final del tramo, la velocidad crucero, la duración de cada tramo, el porcentaje de ocupación en el perfil y el ángulo asociado a cada trayectoria. Una vez definidos estos requerimientos se obtiene como resultado un perfil de velocidad trapezoidal como el que se expone en la Figura 25.

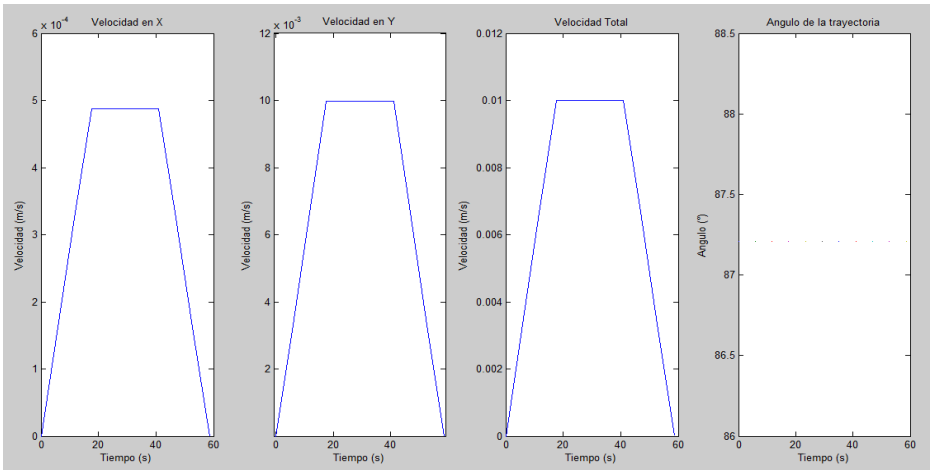


Figura 25. Perfil de velocidad trapezoidal con componentes en *X* y en *Y*, velocidad total y ángulo de trayectoria.

Cabe resaltar que cada ruta definida en la trayectoria del vehículo a escala se compone de diferentes tramos, por lo que en una ruta desde un punto A hacia un punto B pueden haber desde 1 hasta 8 diferentes tramos, los cuales tendrán un perfil de velocidad asociado diferente y una duración especificada por el largo del tramo. En la Figura 26 se ilustra el concepto anteriormente descrito.

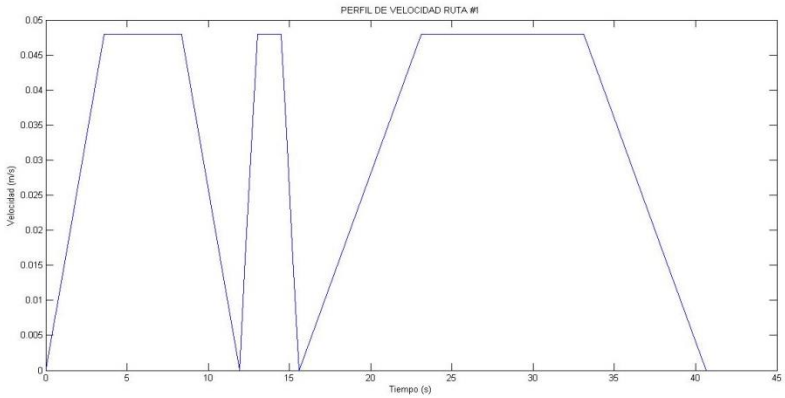


Figura 26. Perfil de velocidad trapezoidal para la ruta 1

Finalmente, los últimos componentes del control del vehículo, se basan en la detección por color, la cual extrae de la imagen en tiempo real los diferentes pigmentos asociados a los colores primarios (Azul, Verde, Rojo) con su respectiva posición en la matriz de la imagen, además del área del color del objeto. Esta posición se envía al algoritmo de control de la posición junto con la información de la velocidad al microcontrolador, el cual según la programación definida pondrá en marcha al robot. La detección de color en la maqueta de cada uno de los vehículos se muestra en la siguiente figura:

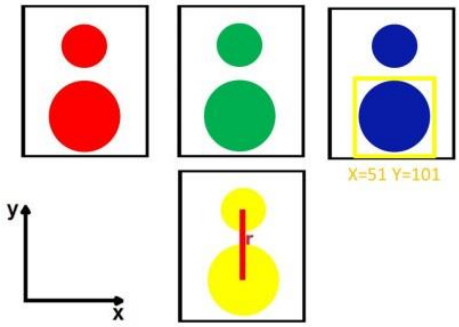


Figura 27. Detección de los vehículos por Color.

5.6 Calibración de la cámara:

La calibración de la cámara es importante en el desarrollo del proyecto, ya que es mediante este proceso con el cual se obtiene la matriz de traslación, la cual permite hacer la conversión de pixeles a unidades de longitud, en este caso a metros para la correcta identificación de los puntos obtenidos con la planificación de la ruta y el control por segmentación de color. Este proceso se realiza con el software *"Camera Calibration Toolbox for Matlab"*, el cual se muestra en la referencia [15].

Para este caso, se utiliza una cámara de alta definición de 720p con una calidad de imagen de 8 megapíxeles, mejorada mediante software. Para iniciar el proceso de calibración, se necesita que el foco y la calidad de la imagen estén en su punto óptimo, una vez obtenido este patrón se comienza por:

Primero, se toman una serie de imágenes de prueba con un tablero de ajedrez, el cual debe cumplir con que el ancho y alto de todos los cuadrados sean iguales además de tener la medida en centímetros de los mismos, ya que ayuda a que el software enfoque la cámara en cada uno de los cuadrados del tablero. Un ejemplo de este proceso se ve en la Figura 28.

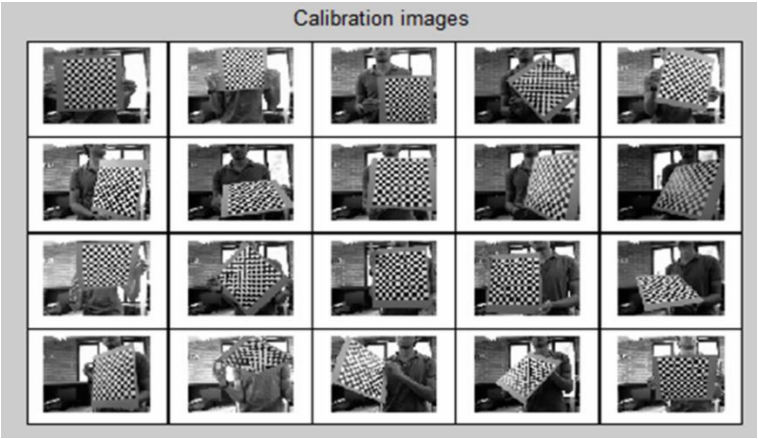


Figura 28. Imágenes de calibración de la cámara.

Luego de tener varias imágenes, se comienza por un proceso de selección e identificación de las esquinas del tablero, donde se asigna el número de cuadros y el ancho de cada uno para que el mismo programa de Matlab calibre todos los puntos de cruce, haciéndolo evidente en la Figura 29.

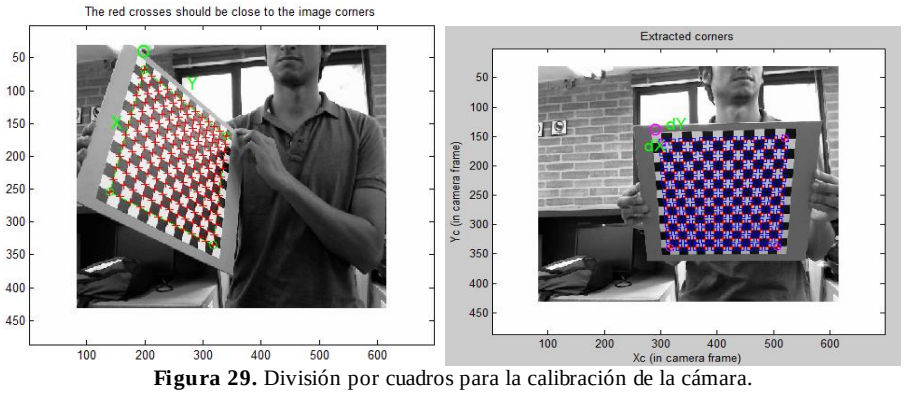


Figura 29. División por cuadros para la calibración de la cámara.

Después de realizar el proceso de identificación con cada una de las imágenes, el programa reconoce la posición de la cámara y el lugar de donde fueron tomadas las fotos, con lo cual se calcula un informe respecto al error asociado. Luego de realizar esta calibración, el foco será más exacto y preciso (Error: 0.18, Figura 30):

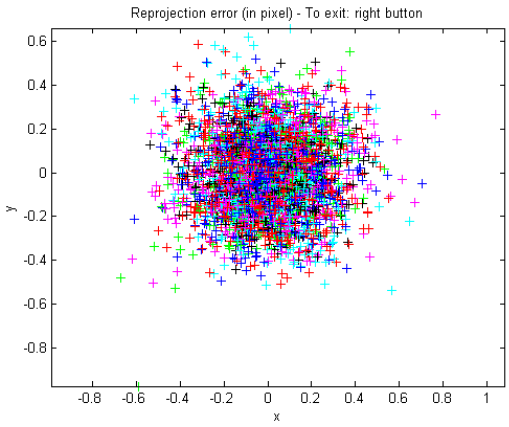


Figura 30. Error de calibración.

5.7 Control de Dirección

El control de dirección permite mantener la dirección del vehículo en la trayectoria determinada por el planificador de ruta, el cual envía un vector de comparación para cada tramo. Para entender en profundidad la función de este control, se debe tener en cuenta dos elementos que lo componen: El primero es un control por magnitudes donde cada tramo de la trayectoria contendrá un valor de magnitud y sentido asociado al vector de dirección, el cual se utiliza como valor de referencia denotado como $\theta_{referencia}$, que corresponde a:

$$\theta_{referencia} = \tan^{-1}\left(\frac{PosY}{PosX}\right) \qquad (36)$$

Donde PosX y PosY serán el inicio y fin de respectivo tramo. El segundo consiste en la abstracción de la posición del vehículo en la maqueta, el cual está determinado por dos puntos de color ubicados en la parte superior del mismo, con los cuales se obtiene dos coordenadas X y Y, que serán los puntos de referencia del vector de dirección θ_{medido}, el cual corresponde a:

$$Pos1 = [x_1 y_1] \quad , \quad Pos2 = [x_2\ y_2] \qquad (37)$$

$$Pos_{vehículo} = [x_2 - x_1 y_2 - y_1] \qquad (38)$$

$$\theta_{medido} = \tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \qquad (39)$$

Una importante aclaración es que la dirección del vehículo (parte delantera) se determina según el área de cada uno de los círculos en su parte superior, en donde el círculo de menor área será el que nos indique la parte delantera del vehículo y consecuentemente el círculo de mayor área determinará la parte trasera. Esta disposición es exhibe en la Figura 31.

Una vez obtenidos $\theta_{medido}\, y\, \theta_{referencia}$, se puede calcular la diferencia entre los ángulos de incidencia, para los cuales se quiere que el resultado obtenido sea cercano a cero. Este cálculo se muestra en la siguiente ecuación:

$$\theta_{total} = \theta_{referencia} - \theta_{medido} \qquad (40)$$

Adicionalmente, este control permite que el vehículo siempre este direccionado en la misma orientación de la trayectoria a seguir, ya que la detección del mismo se realiza en tiempo real comparando el ángulo de la trayectoria con el del vehículo para verificar que el error de estos dos sea de cero, de lo contrario, cualquier cambio en la dirección del vehículo, dependiendo del resultado de θ_{total} y el signo '+' o '-', se activa la acción de corrección en donde la configuración diferencial de los motores DC ayuden a disminuir esta diferencia. La Figura 31 ilustra lo anteriormente planteado.

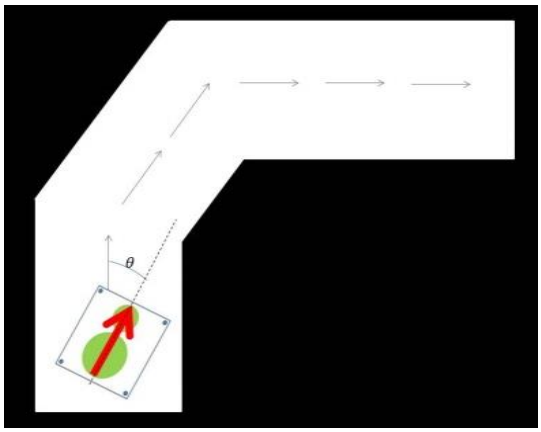


Figura 31. Vector de dirección de los vehículos eléctricos a escala.

Dado el caso en que el resultado sea positivo, esta acción de respuesta se verá refleja en los motores DC, donde el motor derecho actuará con velocidad bajas, mientras que el motor izquierdo quedará con velocidad cero; en caso contrario, si el resultado es negativo, el motor derecho quedará con velocidad cero, mientras que el motor izquierdo actuará con velocidad baja.

Esta corrección se realiza en un tiempo corto de aproximadamente tres segundos, con el fin de tener un tiempo prudente entre la corrección y la nueva detección de la posición de los círculos del vehículo. Si la diferencia es cercana a cero, la comprobación de la dirección pasará a un segundo plano hasta que se vuelva a utilizar en los puntos intermedios de cada tramo.

5.8 Codificación del microcontrolador Arduino Micro

La programación del Arduino Micro se realiza en un entorno de código abierto "Arduino IDE" dispuesto por la empresa Arduino, lo cual hace fácil escribir el respectivo código y cargarlo. Este tipo de software permite la integración de funciones preestablecidas con otros lenguajes de programación y al ser un entorno escrito en Java y basado en Processing, avr-gcc y otros programas de código abierto, permite que sus resultados sea de tipo Open Software facilitando el flujo de información.

Al realizar una analogía con el cuerpo humano, el vehículo eléctrico a escala tendrá un microcontrolador como cerebro, el cual recibirá las acciones a realizar. Estas acciones están previamente programadas en el microcontrolador, las cuales cumplen las funciones de comunicación, velocidad, medición de parámetros y direccionamiento del vehículo, por lo que el primer paso que el microcontrolador debe realizar es conectarse con la estación para recibir las respectivas acciones.

La comunicación siempre se hará de tipo inalámbrica, por lo cual el módulo Bluetooth será uno de los componentes principales a emplear además de programar la correspondiente conexión del sistema embebido a través del módulo I^2C. Luego de que la comunicación sea establecida, lo siguiente es habilitar los puertos de recepción y transmisión del microcontrolador para que al recibir los datos de la estación central, se activen los controles mencionados a lo largo del desarrollo del proyecto (velocidad y dirección). Cabe hacer la aclaración que la comunicación del Arduino Micro con el dispositivo central debe emplear la función Serial1 para habilitar los pines Tx y Rx conectados al módulo Bluetooth.

Por otro lado, el microcontrolador medirá el voltaje que tiene la batería, con el fin de realizar un posible monitoreo de la misma y dejar abierta la posibilidad de modificar el algoritmo de control para que los vehículos puedan realizar paradas o recargas según el nivel de carga. El proceso en general del microcontrolador se describe de manera gráfica en el diagrama de flujo de la Figura 32.

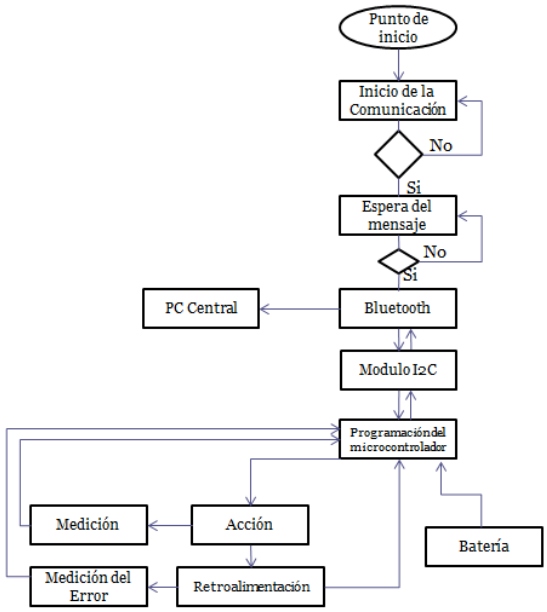


Figura 32. Diagrama de flujo del microcontrolador Arduino Micro.

Otra de las funciones que se ejecutan en el movimiento del vehículo y que se programan en el microcontrolador son: el control PID de los motores, la rotación del servo motor para realizar el giro dependiendo del ángulo de la trayectoria, las señales PWM para la habilitación de motores DC y servo motor, además de integrar la comunicación bidireccional entre el vehículo a escala y la estación central.

5.9 Comunicación Bluetooth

La comunicación bluetooth además de ser una comunicación inalámbrica permite la modificación de su estructura (protocolo) para integrar diversas funcionalidades dependiendo de la aplicación en la que se utilice. En nuestro caso, el protocolo bluetooth es provechoso si se requiere realizar una comunicación transparente entre dos dispositivos, teniendo en cuenta que para una correcta transmisión bidireccional, uno de ellos debe configurarse en modo Slave para que la comunicación se dé solo entre esos dos dispositivos. Las ventajas que admite una comunicación transparente, se reflejan en que no se debe modificar el contenido con banderas e indicadores especiales los cuales se utilizan en otro tipo de configuración. Por lo tanto, se puede mandar cadenas de caracteres sin la necesidad de configurar un bit de inicio o final para que el dispositivo que recibe entienda lo que se le envía. En este modo solo se deben configurar ciertos campos para que la comunicación sea bidireccional como:

- Tasa de transmisión (Baud Rate): 115200 baud
- Modo: Slave
- Paridad: ninguna
- Pincod: 1234
- SvrName: SPP

Para cambiar los parámetros de la configuración por defecto que trae el dispositivo bluetooth, es necesario disponer de un programa que habilite la comunicación serial entre el computador y el módulo, en nuestro caso se puede utilizar tanto el "Monitor Serial" del software de Arduino o el terminal de Windows "Hyperterminal", en el cual se debe configurar la tasa de transmisión en 115200 baudios, para que tanto el computador como el módulo se comuniquen a la misma velocidad. Después solo es necesario mandar la cadena de caracteres "$$$" con la cual se accede al modo de comando del módulo y se configura lo anterior. La Figura 33 ilustra el ejemplo que debe verse al configurar el módulo. Los demás comandos se pueden encontrar en la referencia [16].

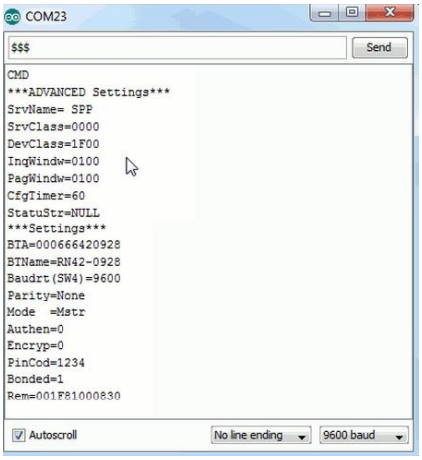


Figura 33. Ejemplo del Modo Comando del módulo Bluetooth.

5.10Puente H

La alimentación de los motores DC se hace utilizando una señal PWM para cada uno de estos y es así como se puede variar su velocidad de forma independiente. Los motores empleados requieren de una corriente mayor a la que entrega cada uno de los pines del microcontrolador. Por lo tanto, se requiere el uso de un puente H para estabilizar la señal del PWM y que no tenga efectos de carga a la salida del microcontrolador. Además este puente H permite que al vehículo a escala se le pueda incluir la opción de reversa. Para este fin, se utilizó un puente H común con referencia L293D de Texas Instrument en la siguiente configuración:

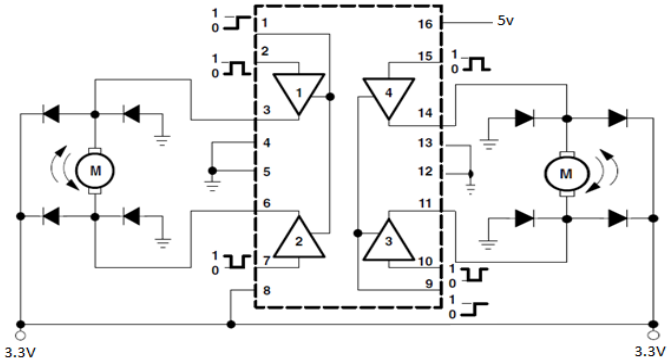


Figura 34. Configuración Puente H empleada con los motores DC.

6. RESULTADOS

En esta sección se presentan los resultados obtenidos con base en los criterios y especificaciones definidas anteriormente y luego de realizar las pruebas implementando el algoritmo de control en Matlab con los respectivos controles de velocidad, dirección y posicionamiento.

6.1 Consumo del sistema

Finalizada la construcción del vehículo eléctrico a escala, el primer resultado que se obtuvo fue el consumo en corriente que requería el vehículo. Por tanto, se midió la corriente que entregaba la batería para cada una de las acciones que realiza el robot y se obtuvo la siguiente información:

Consumo del sistema (A)	
En reposo sin conexión Bluetooth	170 mA
En reposo con conexión Bluetooth	210 mA
En velocidad alta	234 mA
En velocidad media	240 mA
En velocidad baja	254 mA
En giro a la derecha	222 mA
En giro a la izquierda	236 mA
En reversa	307 mA

Tabla 4. Consumo en corriente del sistema

Al observar los resultados de la tabla 4, podemos comprobar que el estado que solicitará mayor corriente a la batería será cuando el vehículo este en reversa y que mientras la velocidad sea mayor el consumo de corriente será menor. Estos datos están sujetos al peso del vehículo (260.16 g) y la fricción que tiene con la superficie, ya que el consumo de corriente es proporcional a estos dos factores.

6.2 Parámetros motores DC

Las primeras pruebas se realizaron para determinar el comportamiento de los motores DC y estudiar la caracterización de estos. Lo que se observó en estas pruebas y se obtuvo como resultado fue una región crítica donde el motor no tiene un comportamiento de tipo lineal, por esta razón se tomó la decisión de elegir un punto de operación cercano a 2.5 V, donde su comportamiento es más lineal y por lo tanto no se tiene el problema de llegar a operar dentro de esta zona crítica. Además, al evitar operar en la región crítica, el voltaje establecido también disminuye la posibilidad de que los motores se detengan por la fuerza de fricción.

La región crítica señalada en rojo y el punto de operación señalado en verde, se puede observar en las figuras 35 y 36.

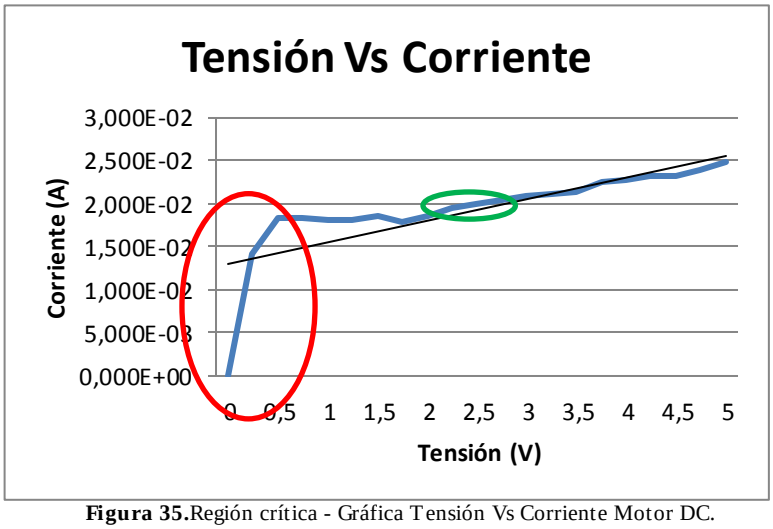


Figura 35. Región crítica - Gráfica Tensión Vs Corriente Motor DC.

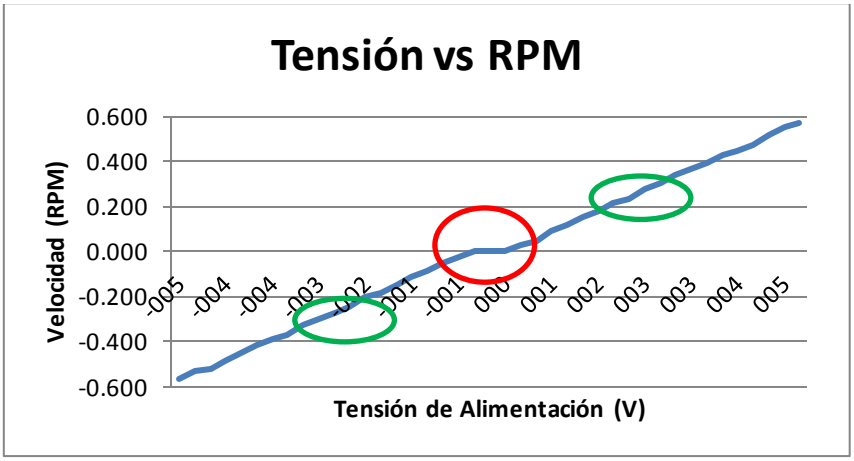


Figura 36. Región crítica – Curva característica Motor DC.

6.3 Configuración del microcontrolador

Con el fin de ejecutar los comandos de velocidad y dirección incluyendo el Control PI respectivo, se realizó la programación del microcontrolador para que los motores operen dentro del punto de operación. Al realizar la prueba de velocidad y dirección se obtuvo que para que la alineación de la llanta delantera estuviera correcta, era necesario iniciar el servo motor en una posición fija, la cual fue de 112°, posición en la cual el eje de la llanta es perpendicular al vector de dirección del robot, lo que hace posible girar en ángulos entre -45° y 45°. El código que se implementó para la programación del microcontrolador Arduino Micro se encuentra en el Anexo A.

Luego de que el microcontrolador estuviera programado, se estableció la conexión por medio del módulo bluetooth. Durante esta configuración, el envío de datos se configuró de manera serial, ya que en Matlab la comunicación entre el computador y el adaptador bluetooth se realiza de esta forma.

El proceso asociado al envió de los comandos pasa por un emparejamiento entre el computador y el modulo del vehículo eléctrico a escala, en donde Windows 7 (Sistema operativo pre-instalado) al detectar el dispositivo asigna un puerto COM ##, con el cual se identifica cada conexión asociada. Para evitar posibles confusiones en el envió de los comando respectivos, se modificó el nombre del módulo con el respectivo color del vehículo (Ejemplos: EVAZUL y EVAMARILLO).

Una vez establecida la comunicación entre el vehículo y la estación central, se prosiguió a calcular la velocidad del robot, conociendo que el código asociado al perfil de velocidad de cada tramo en Matlab se compone de tres niveles de velocidad en sus perfiles trapezoidales. Por lo tanto, para cumplir el requerimiento de las tres velocidades, se realizó el cálculo de una velocidad baja, una media y una alta para asignarle un valor respectivo en el código del microcontrolador y se decidió que al momento del giro, la velocidad diferencial también tuviera una velocidad entre media y baja para el motor correspondiente. Siendo así, se utilizaron los datos que se encuentran en la tabla 5:

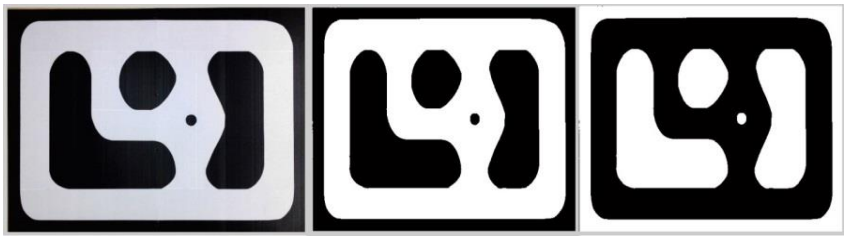
Acción	Velocidad Matlab (m/s)	Velocidad Arduino Micro	Velocidad real del EV (m/s)	Tensión de entrada al motor (V)
Velocidad Alta	0.038	210	0.038	2.53
Velocidad Media	0.025	190	0.025	2.43
Velocidad Baja	0.012	170	0.012	2.2
Giro Derecha	0.012	180	0.012	2.3
Giro Izquierda	0.012	180	0.012	2.3

Tabla 5. Velocidades del vehículo eléctrico a escala.

6.4 Segmentación por detección de color

Una parte importante en la ubicación del vehículo y la planificación de la trayectoria es el análisis de las imágenes que conforman el entorno bajo prueba. Para entender los resultados asociados, es necesario comenzar desde un nivel alto compuesto por la adquisición de imágenes, hasta un nivel bajo, el cual se encarga de enviar los parámetros necesarios para que el vehículo se mueva en la trayectoria deseada.

Antes de utilizar el algoritmo de planificación A*, es necesario aplicar algunas operaciones morfológicas que permitan cambiar la imagen de formato RGB a tipo Double en Matlab además de realizar el proceso de umbralización y el intercambio de tonalidades (Negro por Blanco y viceversa), debido a que la función del algoritmo del planificador A* que detecta los obstáculos del entorno así lo predefine. Este resultado se puede ver en la Figura 37 donde se muestra en la figura 37a) la imagen real de la maqueta, luego en la figura 37b) la imagen binarizada y por último en la figura 37c) el intercambio de los colores.



a) **b)** **c)**

Figura 37. Proceso desde la toma de imagen hasta la preparación para el planificador de ruta

Una vez obtenida la imagen que define el algoritmo de planificación, es necesario definir un punto de inicio y fin que permita el cálculo de la ruta deseada, de ser necesario, una vez realizado el análisis de la ruta, se puede obtener el espació de configuraciones (costos) asociado (C-space) para modificarlo según sea el requerimiento.

Cada ruta (denominada Path en el código de Matlab), se compone de varios tramos, que se encuentran en una matriz de 2XN, pero se tiene el inconveniente que el algoritmo siguiente, por sí solo, no reconoce cada tramo. Por lo tanto, se implementó una función que permite la identificación de cada tramo con su respectivo punto de inicio y final almacenados en los vectores correspondientes. Este resultado se representa de la siguiente forma:

$$inicioCarro\,\# = [inicio1(x,y)\ \ inicio2(x,y)\ \ inicio3(x,y)\ \ldots\ inicio\,n(x,y)] \qquad (41)$$

$$finCarro\,\# = [fin1(x,y)\ \ fin2(x,y)\ \ fin3(x,y)\ \ldots\ fin\,n(x,y)] \qquad (42)$$

Donde inicio1 y fin1 son las componentes del primer tramo con sus puntos respectivos en el eje coordenado XY. Para analizar la eficiencia de este algoritmo en la detección de las rutas, se puede calcular utilizando la tabla 6:

RUTA	Celdas asociadas	Puntos Conocidos	Puntos Detectados	Puntos no detectados	Celdas no detectadas	Error
Ruta 1	578	3	3	0	2	0.34 %
Ruta 2	370	5	5	1	60	16.21 %

Tabla 6. Detección de puntos de la trayectoria.

Realizados los pasos anteriores, se prosiguió a probar la detección de colores sobre la maqueta y fue ahí donde se encontró que la segmentación por color tenía algunos problemas con la detección de este ya que no diferenciaba el rojo del amarillo y además no detectaba el color verde, por eso se optó por la utilización del color amarillo y azul los cuales no presentaban inconveniente alguno para su detección. También, para solucionar este problema, se empleó un kit de luces Kino Flo con luz blanca para ayudar a diferenciar y a detectar los círculos correspondientes de cada uno de los vehículos.

Al superar estos inconvenientes, se programó en Matlab un punto de partida, un punto de llegada y un punto de control para que el vehículo siempre que estuviera en su punto de partida y su vector de velocidad no estuviera direccionado hacia la trayectoria a seguir lo ubique paralelo a este. También se programó para que el vehículo se detuviera en un punto medio de la trayectoria para sincronizar nuevamente la dirección del robot con respecto a la dirección del vector de la trayectoria, y para finalizar, también se programó que en cada momento que llegará a girar se detuviera para hacer el giro pausadamente hasta que el vector se alinee con el vector de la trayectoria. Por tanto, se hizo nuevamente una prueba para emplear la planeación de trayectorias y su seguimiento. La Figura 38 muestra las rutas definidas para el recorrido de los vehículos y la Figura 39 ubica las trayectorias definidas en el mapa para un mejor entendimiento. Siendo la ruta en color azul, la primera ruta y la segunda definida en color rojo.

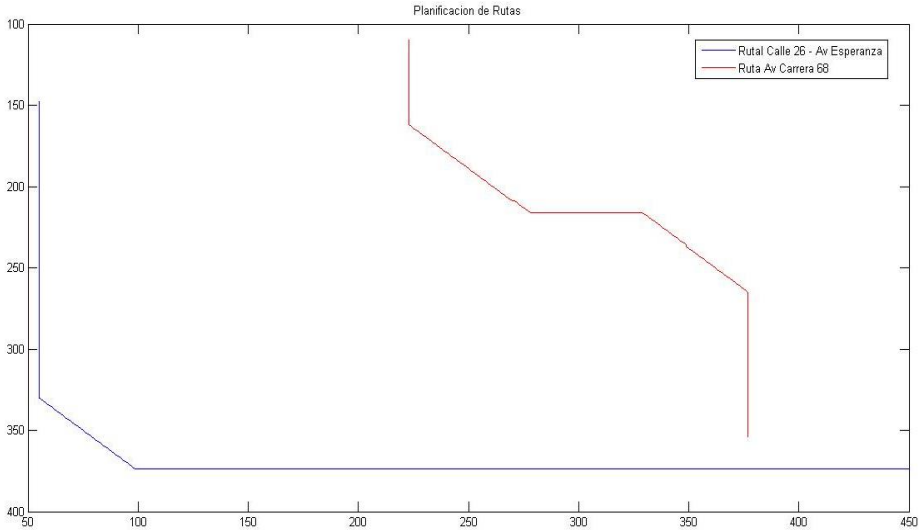


Figura 38. Resultado del planificador de rutas con los puntos deseados.

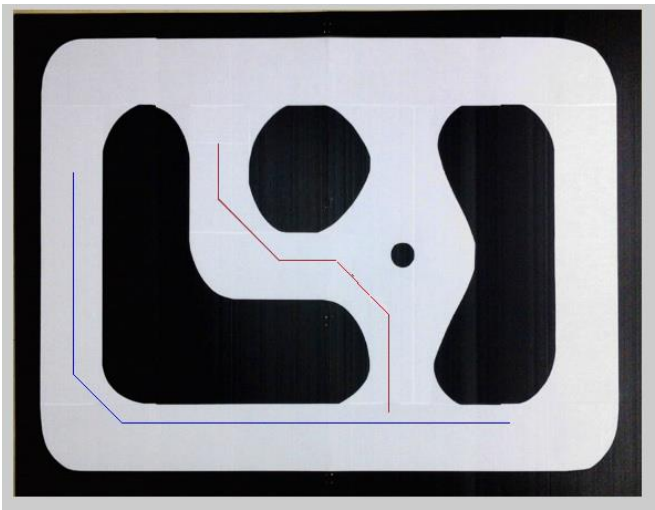


Figura 39. Ubicación de trayectorias en el entorno controlado.

Como se había explicado anteriormente, para realizar este control de dirección, el vehículo cuenta con dos círculos con diferentes radios donde el círculo con radio menor representa la punta del vector y el círculo con mayor radio representa el inicio del vector. Este vector se define en Matlab como un vector de dos valores P = [inicio fin], en el cual, para una correcta determinación del sentido, el inicio es el centro del círculo pequeño y el fin es el centro del círculo con mayor radio.

El código de la segmentación por color, la planeación de trayectorias y el envío de comandos al vehículo según los perfiles trapezoidales de velocidad se encuentra adjunto en el Anexo B y sus funciones en los Anexos C, D, E, F y G.

6.5 Controlador de velocidad PI

Luego de implementar el controlador PI con la función preestablecida en el microcontrolador Arduino Micro se obtuvieron los siguientes resultados en el caso de la velocidad alta ya que esta velocidad es la que más tiempo se emplea, las demás velocidades siempre están dentro de un rango menor a 5 segundos.

Para observar el comportamiento del controlador PI se observó la señal de salida de motor la cual es la misma señal de realimentación y se muestra en la Figura 40. La señal de control se muestra en la Figura 41. Y por último, la señal de control luego del puente H que suministra la alimentación a los motores se muestra en la Figura 42. Por otro lado, la señal de referencia es una señal PWM que genera el microcontrolador internamente a una velocidad de 210 (según código de Arduino) y esta es la que compara con la señal de la figura 38. Luego realiza el control PI y a la salida del microcontrolador se entrega la señal de control.

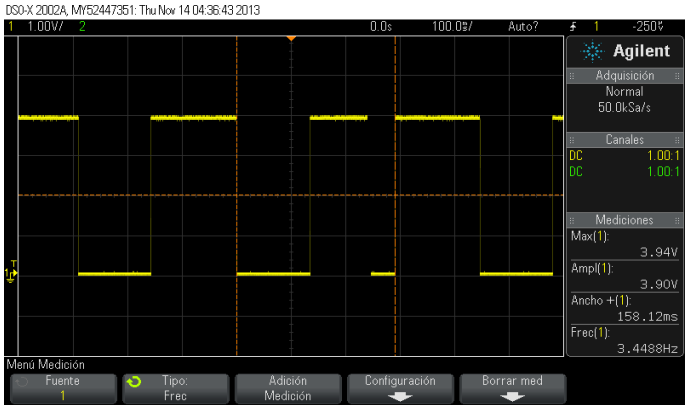


Figura 40. Señal de salida del motor suministrada por los encoders.

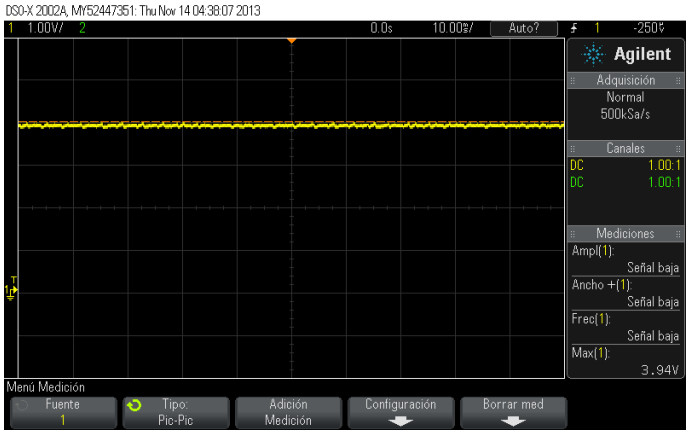


Figura 41. Señal de control al motor como salida del microcontrolador.

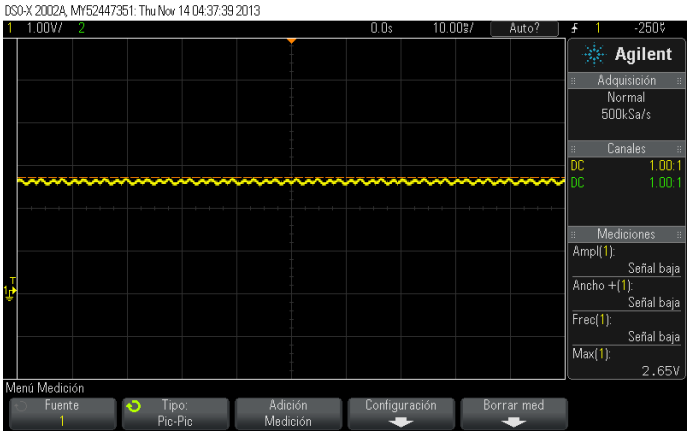


Figura 42. Señal de control como entrada al motor después del puente H.

Con estas imágenes se puede observar que efectivamente el voltaje constante que va a estar suministrando los motores DC es correspondiente a 2.65 V. Pero se puede observar que el puente H también ocasiona una caída de tensión considerable de 3.94 V a 2.65 V es aquí donde también se puede ver que los motores DC piden mucha corriente y es por esto que se vuelve necesario el empleo del puente H.

6.6 Duración de las trayectorias

Al unir todos los componentes de la arquitectura de control desarrollada para el proyecto (segmentación por color, control de dirección y velocidad) se obtuvo que el vehículo eléctrico a escala, durante la simulación de las trayectorias individualmente, duraba 141 segundos en los tramos correspondientes a la trayectoria 1 y en los tramos correspondientes a la trayectoria 2 tenía una duración de 149,4 s. Los respectivos tiempos de la trayectoria 1 y 2 se pueden observar en la tabla 7 y 8, respectivamente.

	Trayectoria 1	
Tipo Dirección	Matlab(s)	Real(s)
Derecha	186,482	184,2
Izquierda	169,587	150,0
En línea	155,431	141,0

Tabla 7. Tiempos de duración para la trayectoria 1

	Trayectoria 2	
Tipo Dirección	Matlab(s)	Real(s)
Derecha	178,265	154,8
Izquierda	188,208	184,56
En línea	169,083	149,4

Tabla 8. Tiempos de duración para la trayectoria 2

Es importante resaltar que la diferencia entre el tiempo real y el calculado se debe a los retardos que genera Matlab en el desarrollo del código, debido a que cada comprobación, desde que se ejecuta, tiene una duración entre 7 a 12 segundos, por lo tanto este tiempo va afectar la duración presupuestada.

Al realizar las dos trayectorias al tiempo, se obtuvo que la duración total hasta ambos puntos de llegada es de 4,57 minutos que equivale a sumar los tiempos individuales de cada trayectoria.

6.7 Resultado de las trayectorias

Las trayectorias relacionadas en el punto de segmentación por color solo son válidas si el vehículo a escala es capaz de mantenerla en toda la duración del trayecto, por lo tanto para comprobar el correcto funcionamiento del planificador de ruta y el control de dirección, es necesario ubicar el vehículo en 3 tipos de posiciones: La primera consiste en ubicarlo de tal modo que este quede en sentido derecho, el segundo consiste en ubicarlo de tal modo que este quede en sentido izquierdo, y por último, se deja en la misma orientación que la posición de referencia. La Figura 43 ilustra el sentido de la ubicación del vehículo a escala con su respectivo ajuste.

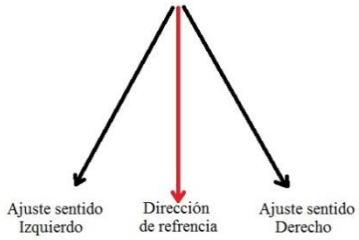


Figura 43. Ubicación del vehículo a escala y el ajuste respectivo.

A partir de estas ubicaciones se puede ejecutar en cada uno de los casos el código general, donde la detección y corrección de la posición del vehículo a escala se realiza, estos resultados son evidentes en las Figuras 44 y 45, las cuales representan cada una de los trayectos (Ruta 1, Ruta 2), en los cuales los puntos de color verde hacen referencia al ajuste en el sentido derecho, los de color morado al ajuste en el sentido izquierdo y los puntos negros es la ruta en el sentido que determina el planificador de ruta.

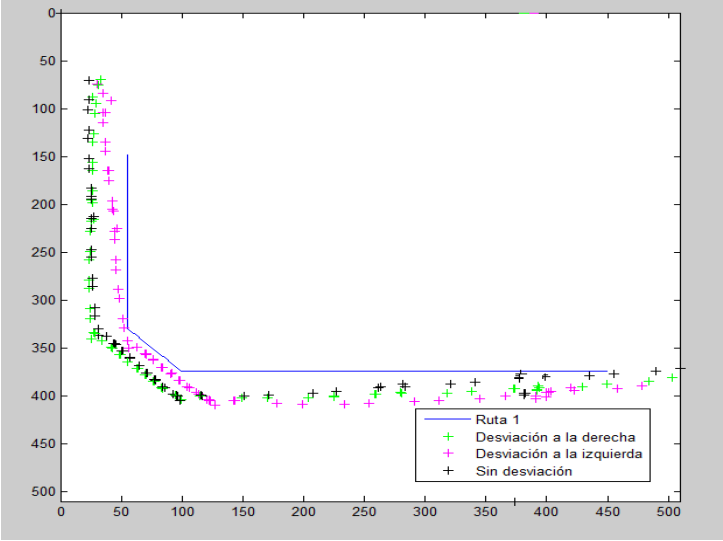


Figura 44. Resultado de la Ruta 1

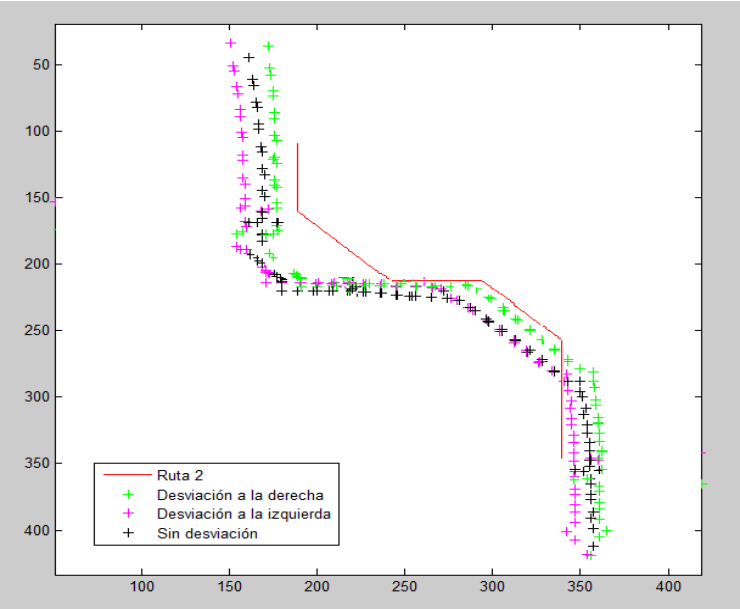


Figura 45. Resultado de la Ruta 2

Es necesario aclarar que la corrección que se realiza para el control de dirección no obedece estrictamente la posición XY del planificador de ruta, debido a que se utiliza una comprobación por orientación la cual suma una distancia reducida entre la ruta que va en la dirección y sentido del planificador con la orientación medida por la detección por color, distancia que en centímetros equivale a 0.002 cm, lo que la hace despreciable si se compara con el tamaño de la maqueta y el área del vehículo. Por lo tanto, el resultado general será correcto ya que sin importar la posición en la que se deje el vehículo o que este quede dentro de la maqueta, la corrección se realizará respecto a la orientación del resultado del planificador como es evidente en las Figuras 44 y 45, donde la orientación es la correcta para todos los tramos que componen las rutas.

6.8 Diseño del circuito impreso

Para efectos de un trabajo final es necesario realizar el diseño de un circuito impreso, el cual debe cumplir con los parámetros de medidas y formas que se tienen del esquema del vehículo a escala. En este caso, se realizó con dimensiones de 8 cm largo por 8 cm de ancho, asegurando un acople fijo en el interior del vehículo. Dada la ubicación del circuito impreso, fue necesario modificar la forma cuadrada por una de tipo en U, dejando espacio para la parte trasera del servo motor que se encuentra ubicado al mismo nivel.
En este circuito impreso, se diseñó a partir del diagrama esquemático de todos los componentes con las conexiones correspondientes, el cual puede ser consultado en el Anexo H.

En el circuito impreso se ubican estratégicamente y funcionalmente los componentes referentes al vehículo a escala, entre estos se encuentran: conectores, elevador de voltaje (PTN04050C), switch, diodos, puente H (L293D), Modulo bluetooth, y condensadores. El resultado de lo anteriormente mencionado se puede observar en la Figuras 46 y 47.

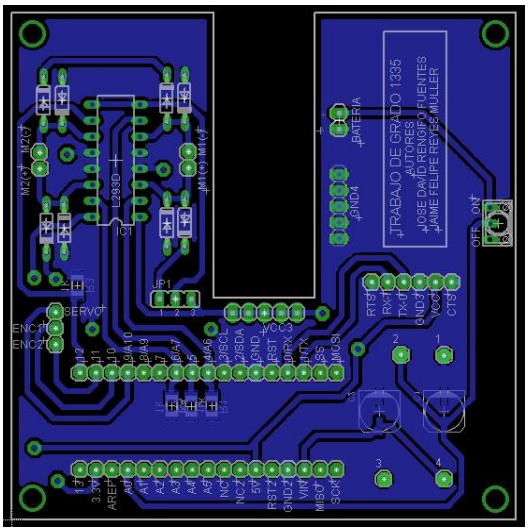


Figura 46. Diseño del circuito impreso vista inferior.

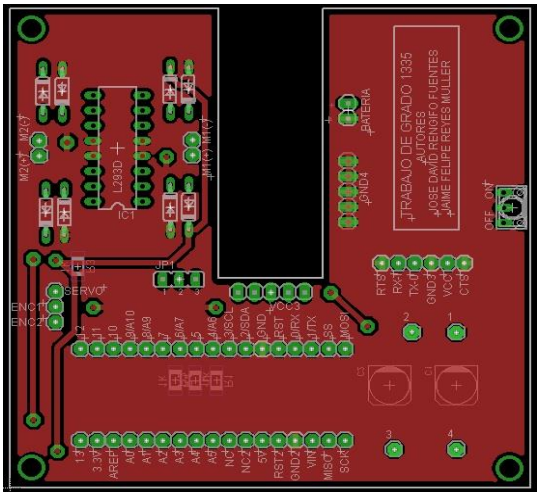


Figura 47. Diseño del circuito impreso vista superior.

Por último al ajustar las medidas con los requerimientos, se determinó que el tamaño final obedece las siguientes dimensiones:

- Largo: 8 cm
- Ancho: 8 cm
- Tamaño del rectángulo interno: 4.5 cm de largo x 1.6 cm de ancho
- Círculos de apoyo: Diámetro 3 mm
- Ubicación de los círculos: 1.68 mm de la parte superior e inferior y 1.66 mm de las partes laterales

7. CONCLUSIONES

La navegación de los vehículos eléctricos a escala dentro de un entorno controlado se desarrolló completamente cumpliendo con el objetivo general del proyecto. A partir de este objetivo se construyó un prototipo a escala con control de velocidad y dirección, lo que permitió implementar un sistema de control por segmentación por color que cumple con las especificaciones dadas.

Luego de analizar los resultados, se pudo concluir que uno de los factores que afectan el comportamiento del seguimiento de la trayectoria del vehículo es la posición inicial del mismo, ya que si se encuentra fuera del punto programado, la desviación será apreciable. Además, tanto en el punto de inicio como en el punto final se debe tener en cuenta el área que ocupa el vehículo, porque si no se tiene en cuenta, el vehículo terminará fuera del entorno controlado y no cumplirá con el objetivo de la ruta a seguir.

Por otra parte, es necesario calcular y considerar los retardos que genera Matlab en la ejecución del código, ya que este parámetro permite obtener una mayor coordinación entre el movimiento del robot y los comandos asociados, además de delimitar varias secciones del código.

Por último, este proyecto es la base practica para desarrollos futuros en donde se pueden implementar características adicionales como el control de la batería, estaciones intermedias (con tiempo de espera en cada una) y de recarga, y la posibilidad de acoplar más vehículos al entorno utilizando un algoritmo de coordinación para el movimiento. Esto permite tener una mejor idea sobre la función de los vehículos eléctricos como cargas manejables dentro de una Smart Grid.

BIBLIOGRAFÍA

[1] E. Karfopoulos, P. Papadopoulos, S. Skarvelis-Kazakos, I. Grau, L. Cipcigan, N. Hatziargyriou y N. Jenkins, «Introducing electric vehicles in the microgrids concept,» *Intelligent System Application to Power Systems (ISAP), 16th International Conference* , p. 6, 2011.

[2] R. Verzijlbergh, M. Grond, Z. Lukszo, J. Slootweg y M. Ilic, «Network Impacts and Cost Savings of Controlled EV Charging,» *IEEE TRANSACTIONS ON SMART GRID,* vol. 3, nº 3, pp. 1203-1212, 2012.

[3] «Hitachi - Smart Grid,» [En línea]. Available: http://www.hitachi.com/environment/showcase/solution/energy/smartgrid.html. [Último acceso: Noviembre 2013].

[4] «Electropaedia - Rechargeable Lithium Batteries,» [En línea]. Available: http://www.mpoweruk.com/lithiumS.htm#polymer. [Último acceso: Agosto 2013].

[5] «Battery University - Li-polymer Battery: Substance or Hype?,» [En línea]. Available: http://batteryuniversity.com/learn/article/the_li_polymer_battery_substance_or_hype. [Último acceso: Agosto 2013].

[6] A. Moore, «Revolution Robotics,» 14 Noviembre 2008. [En línea]. Available: http://revolution-robotics.com/articles/lithium_polymer_lipo_battery_guide. [Último acceso: Agosto 2013].

[7] «Adafruit Learning System - Li-Ion & LiPoly Batteries,» [En línea]. Available: http://learn.adafruit.com/li-ion-and-lipoly-batteries/voltages. [Último acceso: Agosto 2013].

[8] I. Husain, Electric and hybrid vehicles – Design fundamentals, USA, 2005.

[9] M. Sauter, From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband, Alemania, 2011.

[10] «Hardside,» [En línea]. Available: http://www.hardside.com.ar/utils/bluetooth1/imagenes/72-4.jpg. [Último acceso: Agosto 2013].

[11] J. S. J. Angulo, Segmentación de imágenes en color utilizando histogramas bi-variables en espacios color polares luminancia/saturación/matiz, Francia: Centre de Morphologie Mathematique, 2005.

[12] J. Quiroga, Procesamiento de imágenes: Imágenes Binarias, Bogotá D.C., Colombia: Pontificia Universidad Javeriana, 2011.

[13] F. O. F. Gil P. Torres, Detección de objetos por segmentación multinivel combinada de espacios de color, Alicante, España: Universidad de Alicante, 2004.

[14] J. Colorado, «Robótica móvil,» Pontificia Universidad Javeriana, Bogotá D.C., Colombia, 2013.

[15] J.-Y. Bouguet, «Camera Calibration Toolbox for Matlab,» 4 Noviembre 2013. [En línea]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/. [Último acceso: 12 Octubre 2013].

[16] Sparkfun, «learn.sparkfun.com,» [En línea]. Available: https://learn.sparkfun.com/tutorials/using-the-bluesmirf/example-code-using-command-mode. [Último acceso: Noviembre 2013].

ANEXO A - Código Programación Microcontrolador Arduino Micro

```
#include<Servo.h> //Incluir biblioteca de servo
#include<PID_v1.h> //Incluir función de PID

Servo servo; //Creación del PWM para dirección del servo
int pos=112; //Posición inicial del servo
intpL=142; //Dirección a la izquierda del servo
intpR=82; // Dirección a la derecha del servo
intdelaytime = 70; //Tiempo duración en giro de la llanta delantera
intDCmotorR = 6, DCmotorL = 5, DCmotorLB = 4, DCmotorRB= 12; //Asignación de pines para las señales de cada
motor
intVelR=180; //velocidad del motor derecho (0.5V = 100 y 5V = 255)
intVelL=180; //velocidad del motor izquierdo (0.5V = 100 y 5V = 255)
float carga, realcarga; //variables para medir el voltaje
doubleSetpoint, InputR, OutputR, InputL, OutputL; //variables para el control PI
PID motorPIDR(&InputR, &OutputR, &Setpoint,0.002416,12.08,0, DIRECT); //Control PI para motor derecho
PID motorPIDL(&InputL, &OutputL, &Setpoint,0.002416,12.08,0, DIRECT); //Control PI para motor izquierdo

voidsetup()
{
servo.attach(9); //Asignación del servo en el pin 9
pinMode(4,OUTPUT); //Pin 4 salida al puente H señal de activación
pinMode(5,OUTPUT); //Pin 5 salida al puente H señal del motor
pinMode(6,OUTPUT); //Pin 6 salida al puente H señal del motor
pinMode(10,INPUT); //Pin 10 entrada del encoder izquierdo
pinMode(11,INPUT); //Pin 10 entrada del encoder derecho
pinMode(12,OUTPUT); //Pin 11 salida al puente H señal de activación
pinMode(36,INPUT); //Pin 36 medición del voltaje de la bateria

Serial1.begin(115200); //Comunicación por bluetooth serial a 115200 baud
servo.write(pos); //Servo en la posición inicial
InputR = analogRead(11); //Lectura de la señal del encoder derecho
InputL = analogRead(10); //Lectura de la señal del encoder izquierdo
}

void loop()
{
while (Serial1.available() > 0){
char velocidad = Serial1.read(); //Se guarda en la variable velocidad el dato que se recibe del computador central
switch(velocidad){
case '0.012': //Velocidad Baja
    {
Setpoint=170; //Establecimiento de la velocidad de referencia
motorPIDR.SetMode(AUTOMATIC); //Configuración del PI derecho
motorPIDL.SetMode(AUTOMATIC); //Configuración del PI izquierdo
motorPIDR.Compute(); //Activación del PI derecho
motorPIDL.Compute(); //Activación del PI izquierdo
analogWrite(DCmotorRB,0); //Señal de activación al puente H en reversa 0
analogWrite(DCmotorLB,0); //Señal de activación al puente H en reversa 0
```

```
analogWrite(DCmotorR,OutputR);  //Señal del motor al puente H igual a la salida del PI
analogWrite(DCmotorL,OutputL);  //Señal del motor al puente H igual a la salida del PI
servo.write(pos); //Configuración de la posición del servo en la posición inicial
break;
     }
case '0.025':  //velocidad Media
     {
Setpoint=190; //Establecimiento de la velocidad de referencia
motorPIDR.SetMode(AUTOMATIC);  //Configuración del PI derecho
motorPIDL.SetMode(AUTOMATIC);  //Configuración del PI izquierdo
motorPIDR.Compute();  //Activación del PI derecho
motorPIDL.Compute();  //Activación del PI izquierdo
analogWrite(DCmotorRB,0);  //Señal de activación al puente H en reversa 0
analogWrite(DCmotorLB,0);  //Señal de activación al puente H en reversa 0
analogWrite(DCmotorR,OutputR);  //Señal del motor al puente H igual a la salida del PI
analogWrite(DCmotorL,OutputL);  //Señal del motor al puente H igual a la salida del PI
servo.write(pos); //Configuración de la posición del servo en la posición inicial
break;
     }
case '0.038':
     {
Setpoint=210; //Establecimiento de la velocidad de referencia
motorPIDR.SetMode(AUTOMATIC);  //Configuración del PI derecho
motorPIDL.SetMode(AUTOMATIC);  //Configuración del PI izquierdo
motorPIDR.Compute();  //Activación del PI derecho
motorPIDL.Compute();  //Activación del PI izquierdo
analogWrite(DCmotorRB,0);  //Señal de activación al puente H en reversa 0
analogWrite(DCmotorLB,0);  //Señal de activación al puente H en reversa 0
analogWrite(DCmotorR,OutputR);  //Señal del motor al puente H igual a la salida del PI
analogWrite(DCmotorL,OutputL);  //Señal del motor al puente H igual a la salida del PI
servo.write(pos); //Configuración de la posición del servo en la posición inicial
break;
     }
case '0.000':
     {
analogWrite(DCmotorR,0);  //Señal del motor al puente H en 0
analogWrite(DCmotorL,0);  //Señal del motor al puente H en 0
analogWrite(DCmotorRB,0);  //Señal de activación al puente H en reversa 0
analogWrite(DCmotorLB,0);  //Señal de activación al puente H en reversa 0
servo.write(pos); //Configuración de la posición del servo en la posición inicial
break;
     }
case 'I':
     {
for(int a=0;a<10;a++){
Setpoint=180; //Establecimiento de la velocidad de referencia
motorPIDR.SetMode(AUTOMATIC);  //Configuración del PI derecho
motorPIDL.SetMode(AUTOMATIC);  //Configuración del PI izquierdo
motorPIDR.Compute();  //Activación del PI derecho
motorPIDL.Compute();  //Activación del PI izquierdo
analogWrite(DCmotorRB,0);  //Señal de activación al puente H en reversa 0
```

```
analogWrite(DCmotorLB,0);  //Señal de activación al puente H en reversa 0
analogWrite(DCmotorR,OutputR);  //Señal del motor al puente H igual a la salida del PI
analogWrite(DCmotorL,0);  //Señal del motor al puente H en 0
servo.write(pR); //Configuración de la posición del servo con giro a la izquierda
        }
break;
      }
case 'D':
      {
for(int b=0;b<10;b++){
Setpoint=180; //Establecimiento de la velocidad de referencia
motorPIDR.SetMode(AUTOMATIC);  //Configuración del PI derecho
motorPIDL.SetMode(AUTOMATIC);  //Configuración del PI izquierdo
motorPIDR.Compute();  //Activación del PI derecho
motorPIDL.Compute();  //Activación del PI izquierdo
analogWrite(DCmotorRB,0);  //Señal de activación al puente H en reversa 0
analogWrite(DCmotorLB,0);  //Señal de activación al puente H en reversa 0
analogWrite(DCmotorR,0);  //Señal del motor al puente H en 0
analogWrite(DCmotorL,OutputL);  //Señal del motor al puente H igual a la salida del PI
servo.write(pL); //Configuración de la posición del servo con giro a la derecha
        }
break;
      }
case 'V':
      {
carga=analogRead(36);  //lectura del voltaje en el pin 36
realcarga=carga*5/1024;  //carga es una variable entre 0 y 1024, se realiza conversión a 0 y 5 voltios
Serial1.println(carga);  //imprime  en el computador central el voltaje de la bateria
break;
      }
case 'R':
      {
Setpoint=180; //Establecimiento de la velocidad de referencia
motorPIDR.SetMode(AUTOMATIC);  //Configuración del PI derecho
motorPIDL.SetMode(AUTOMATIC);  //Configuración del PI izquierdo
motorPIDR.Compute();  //Activación del PI derecho
motorPIDL.Compute();  //Activación del PI izquierdo
analogWrite(DCmotorR,0);  //Señal del motor al puente H en 0
analogWrite(DCmotorL,0);  //Señal del motor al puente H en 0
analogWrite(DCmotorRB,OutputR);  //Señal de activación al puente H en reversa igual a la salida del PI
analogWrite(DCmotorLB,OutputL);  //Señal de activación al puente H en reversa igual a la salida del PI
servo.write(pos); //Configuración de la posición del servo en la posición inicial
break;
      }
   }
  }
}
```

ANEXO B – CÓDIGO MATLAB

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            VEHÍCULOS ELÉCTRICOS A ESCALA CONTROLADOS Y SIMULADOS            %
%                          DENTRO DE UNA SMART GRID                           %
%                                                                             %
% PONTIFICIA UNIVERSIDAD JAVERIANA                                            %
% FACULTAD DE INGENIERIA                                                      %
% DEPARTAMENTO DE ELECTRÓNICA                                                 %
%                                                                             %
% AUTORES:                                                                    %
% JOSE DAVID RENGIFO FUENTES                                                  %
% JAIME FELIPE REYES MULLER                                                    %
%                                                                             %
% FECHA:                                                                      %
% JUNIO-NOVIEMBRE 2013                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Cargar las rutas de los archivos y librerías
clear all
path(path,'C:\Users\pc373\Dropbox\Tesis\Codigo Matlab\Ultimo Codigo')
path(path,'C:\Users\pc373\Dropbox\Tesis\Codigo Matlab\Ultimo Codigo\rvctools')
startup_rvc
clc

% Inicalización del puerto serial
PS=serial('COM20');
set(PS,'Baudrate',115200);  % velocidad a 115200 Baudios
set(PS,'StopBits',1);  % un bit de parada
set(PS,'DataBits',8);  % dato es de 8 bits
set(PS,'Parity','none');  % sin paridad
set(PS,'OutputBufferSize',8);  % ”n” es el número de bytes a enviar
set(PS,'Timeout',6);  % 6 segundos de tiempo de espera

pause(15);

PS2=serial('COM22');
set(PS2,'Baudrate',115200);  % velocidad a 115200 Baudios
set(PS2,'StopBits',1);  % un bit de parada
set(PS2,'DataBits',8);  % dato es de 8 bits
set(PS2,'Parity','none');  % sin paridad
set(PS2,'OutputBufferSize',8);  % ”n” es el número de bytes a enviar
set(PS2,'Timeout',6);  % 6 segundos de tiempo de espera

%% DEFINICIÓN DE TRAYECTORIAS

% Lectura del mapa
foto=getsnapshot(vid);
figure(1);
imshow(foto);
title('Imagen Original');
mapa = imresize(foto, [440 568]);

% Definición de puntos de inicio y fin de cada ruta
 goal1 = [450, 374]; %Rutal Calle 26 - Av Esperanza
start1 = [56, 148];

goal2 = [223, 110]; %Ruta Av Carrera 68
start2 = [377, 355];
```

```
% Algoritmo de planificacio A*
[path1]=aestrella(mapa,goal1,start1);
[path2]=aestrella(mapa,goal2,start2);

% pp=load('camin.mat');
% path1=pp.path1;
% path2=pp.path2;

%% PERFIL DE VELOCIDAD TRAPEZOIDAL

% RUTA # 1
[inicioR1,finR1] = detsegmentos(path1);
[velocidad1,tiempo1,angulo1,habilitador1] = ptrapezoidal(inicioR1,finR1);
inicioR1 = int32(inicioR1);
finR1 = int32(finR1);

% RUTA # 2
[inicioR2,finR2] = detsegmentos(path2);

% inicioR2=[377 339 375 263 329 216 274 212 271 209]; REFERENCIA
% finR2=[376 264 330 217 274 213 272 210 223 125];

[velocidad2,tiempo2,angulo2,habilitador2] = ptrapezoidal(inicioR2,finR2);
inicioR2 = int32(inicioR2);
finR2 = int32(finR2);

%% SEGMENTACIÓN POR COLOR Y CONTROL DE UBICACIÓN

[lCar1,longCar1] = size(velocidad1);
[lang1,tang1] = size(angulo1);

[lCar2,longCar2] = size(velocidad2);
[lang2,tang2] = size(angulo2);

%Detección de puntos intermedios
newhabilitador1 = [];
newhabilitador2 = [];

newhabilitador1 = dethabilitador(habilitador1);
newhabilitador2 = dethabilitador(habilitador2);

% Detección del sentido de la ruta
%Ruta1
signXR1=[];
signYR1=[];
[rowR1,colR1] = size(inicioR1);

for(i=0:colR1/2-1)
   signXR1=[signXR1 finR1(1,2*i+1)-inicioR1(1,2*i+1) finR1(1,2*i+1)-inicioR1(1,2*i+1)];
   signYR1=[signYR1 finR1(1,2*i+2)-inicioR1(1,2*i+2) finR1(1,2*i+2)-inicioR1(1,2*i+2)];
end

signXR1=[signXR1 signXR1(1,6) signXR1(1,6)];
signYR1=[signYR1 signYR1(1,6) signYR1(1,6)];

%Ruta2
signXR2=[];
```

```
signYR2=[];
[rowR2,colR2]  = size(inicioR2);

for(m=0:colR2/2-1)
   signXR2=[signXR2  finR2(1,2*m+1)-inicioR2(1,2*m+1)  finR2(1,2*m+1)-inicioR2(1,2*m+1)];
   signYR2=[signYR2  finR2(1,2*m+2)-inicioR2(1,2*m+2)  signYR2  finR2(1,2*m+2)-inicioR2(1,2*m+2)];
end

% Ubicación del robot movil
newhabilitador1=[0  7 13 20  27  34 38  41];
angulo2=[89.3  45.6 3.1  45  90];
permiso = 0;
siguiente = 0;
pasos = 1;
next1 = 1;
continuar1 = 0;
angulor1 = 1;
angulod1 = 0;
next2 = 1;
continuar2 = 0;
angulor2 = 1;
angulod2 = 0;

pause(5);

fopen(PS);
pause(6);
fopen(PS2);

while(pasos<longCar2)

   %Inicialización de los vehiculos
   fwrite(PS,'0.000','uchar');
   fwrite(PS2,'0.000','uchar');

   if permiso==0 && pasos<longCar1 %Habilitador del Carro Azul (Ruta1)
      if continuar1 == newhabilitador1(1,next1)  % Control de posicion y dirección
         if angulor1<=2
            angulod1 = angulo1(1,1);
         elseif angulor1>2 && angulor1<=4
            angulod1 = angulo1(1,2);
         else
            angulod1 = angulo1(1,3);
         end

         listo1 = 0;

         while(listo1~=1)
            % Deteccion del robot movil
            [posCarro1,anguloCarro1,sentidoCarro1]  = detcolor('b');
            tetatotal1 = angulod1-anguloCarro1;

            % Diferencias menores a 4° no implican corrección
            if abs(tetatotal1)>4
               if signXR1(1,next1)>=0  && signYR1(1,next1)>=0
                % Caso en el que la ruta sea hacia abajo
                if signXR1(1,next1)<signYR1(1,next1)&&signXR1(1,next1)<3
```

```
            if sentidoCarro1(1,1)>0
               fwrite(PS2,'D','uchar');
               pause(0.20);
               fwrite(PS2,'0.00','uchar');
            else
               fwrite(PS2,'I','uchar');  %Giro a la derecha
               pause(0.10);
               fwrite(PS2,'0.00','uchar');
            end
         elseif signXR1(1,next1)<signYR1(1,next1)&&signXR1(1,next1)>3  % Diagonal
            if sentidoCarro1(1,1)>0
               fwrite(PS2,'I','uchar');
               pause(0.10);
               fwrite(PS2,'0.00','uchar');
            else
               fwrite(PS,'D','uchar');
               pause(0.10);
               fwrite(PS2,'0.00','uchar');
            end
         else % Caso en el que la ruta sea hacia la derecha
            if sentidoCarro1(1,2)>0  %Giro a la derecha
               fwrite(PS2,'I','uchar');
               pause(0.20);
               fwrite(PS2,'0.00','uchar');
            else
               fwrite(PS2,'D','uchar');
               pause(0.10);
               fwrite(PS2,'0.00','uchar');
            end
         end
      end

      else
         listo1 = 1;
         fwrite(PS2,'0.000','uchar')
      end
   end
   next1 = next1+1;
   angulor1 = angulor1+1;

else
   %Envio del comando de velocidad
   fwrite(PS2,num2str(velocidad1(1,continuar1)),'uchar');
   pause(tiempo1(1,continuar1+1)-tiempo1(1,continuar1));
   fwrite(PS2,'0.00','uchar');
end
fwrite(PS,'0.000','uchar');
permiso = 1;
continuar1 = continuar1+1;


else %Habilitador del Carro Amarillo (Ruta2)
   if continuar2 == newhabilitador2(1,next2)  %Seleccion del angulo a comparar
      if angulor2<=2
         angulod2 = angulo2(1,1);
      elseif angulor2>2 && angulor2<=4
         angulod2 = angulo2(1,2);
      elseif angulor2>4 && angulor2<=6
```

```
            angulod2 = angulo2(1,3);
        elseif angulor2>6 && angulor2<=8
            angulod2 = angulo2(1,4);
        else
            angulod2 = angulo2(1,5);
        end

        listo2 = 0;

        while(listo2~=1)
            % Deteccion del robot movil
            [posCarro2,anguloCarro2,sentidoCarro2]  = detcolor('a');
            tetatotal2 = angulod2-anguloCarro2;

            % Diferencias menores a 4° no implican corrección
            if abs(tetatotal2)>4
                if signXR2(1,next2)<0  && signYR2(1,next2)<0  %Corrección Inicial
                    if sentidoCarro2(1,1)>=0  && sentidoCarro2(1,2)<=0
                        fwrite(PS,'I','uchar');
                        pause(0.20);
                        fwrite(PS,'0.000','uchar');
                    end
                end
                if sentidoCarro2(1,1)<=0  && sentidoCarro2(1,2)<=0  && angulod2>72 %Corrección en ángulos
superiores a 72°
                    if abs(sentidoCarro2(1,1))>=0
                        fwrite(PS,'D','uchar');
                        pause(0.20);
                        fwrite(PS,'0.000','uchar');
                    else
                        fwrite(PS,'I','uchar');
                        pause(0.20);
                        fwrite(PS,'0.000','uchar');
                    end
                elseif abs(sentidoCarro2(1,2))>1  %Correccion para diagonales
                    fwrite(PS,'I','uchar');
                    pause(0.15);
                    fwrite(PS,'0.000','uchar');
                elseif angulod2>30 && angulod2<70 %Correccion para angulo de 45°
                    fwrite(PS,'D','uchar');
                    pause(2.2);
                    fwrite(PS,'0.000','uchar');
                end
            else
                listo2 = 1;
                fwrite(PS,'0.000','uchar');
            end
        end
        next2 = next2+1;
        angulor2 = angulor2+1;

    else
        %Envio del comando de velocidad
        fwrite(PS,num2str(velocidad2(1,continuar2)),'uchar');
        pause(tiempo2(1,continuar2+1)-tiempo2(1,continuar2));
        fwrite(PS,'0.000','uchar');
    end
    fwrite(PS,'0.000','uchar');
```

```
        continuar2 = continuar2+1;
        permiso = 0;
        siguiente = 1;
    end

    if siguiente==1 %Habilita que el cambio se realice cada 2 procesos
        pasos = pasos+1;
        siguiente = 0;
    end
end
fwrite(PS,'0.000','uchar');
fwrite(PS2,'0.000','uchar');
disp('Codigo terminado');

fclose(PS);
fclose(PS2);
delete(PS);
delete(PS2);
clear PS;
```

ANEXO C - FUNCIÓN DEL PLANIFICADOR A*

```
function [ruta]=aestrella(imagen1,goal,start)

% Variables de la función
ruta=[];
path(path,'C:\Users\Jose David\Downloads\rvctools')
path(path,'C:\Users\Jose David\Pictures\Tesis')
path(path,'C:\Users\Jose David\Downloads\Trabajo de Grado')
startup_rvc

%Recorte de la imagen
imagen = imresize(imagen1, [440 568]);

%Umbralizacion de la Imagen
u=graythresh(imagen);
BW=im2bw(via,imagen);

%Paso de tipo Logico a Double
BW2=im2double(BW);

% Cambio de color: Blanco - Negro
Trace = ~BW2;
Trace2= im2double(Trace);

% Algoritmo de navegacion D*
dstar = Dstar(Trace2, 'quiet');

C=load('Costmap.mat'); % Obtiene el C-Space(Mapa de costos) de la maqueta
C1=C.Costmap;
dstar.costmap_set(C1);

% Tramo #1
dstar.plan(goal);
tic;
ruta = dstar.path(start);
```

```
toc;
dstar.reset();
end
```

ANEXO D - FUNCIÓN DE DETECCIÓN POR COLOR

```
function [posColor,tetaColor,sentido]=detcolor(tipo)

% Inicialización de la camara
vid = videoinput('winvideo',1,'I420_800x600');
set(vid, 'FramesPerTrigger', Inf);
set(vid, 'ReturnedColorspace', 'rgb');

%Variables de la funcion
posColor = [];
tetaColor = 0;
sentido = 0;
posAzul = [];
posAmarillo = [];
tetaAzul = 0;
tetaAmarillo = 0;

%% Para conocer la posicion inicial del vehiculo
vid.FrameGrabInterval = 5;

if tipo=='b' %Color Azul
   start(vid)

   while(vid.FramesAcquired<=10)
      Azulpos=[];;
      foto = getsnapshot(vid);
      data = imresize(foto, [440 568]);

      %Deteccion de Color Azul
      azul = imsubtract(data(:,:,3), rgb2gray(data));
      azul = medfilt2(azul, [3 3]);
      azul = im2bw(azul,0.18);
      azul = bwareaopen(azul,80);
      bw03 = bwlabel(azul, 4);
      stats3 = regionprops(bw03, 'BoundingBox', 'Centroid');

      %Recorte de color Azul
      area1=getfield(stats3(1,1),'BoundingBox');
      punto1=round(getfield(stats3(1,1),'Centroid'));
      area2=getfield(stats3(2,1),'BoundingBox');
      punto2=round(getfield(stats3(2,1),'Centroid'));

      if area1(1,3)>area2(1,3)
         Azulpos=[punto1 punto2];
      else
         Azulpos=[punto2 punto1];
      end

      %Determinacion del angulo
      posAzul = abs([Azulpos(1,3)-Azulpos(1,1) Azulpos(1,4)-Azulpos(1,2)]);
      tetaAzul = rad2deg(atan(posAzul(1,2)/posAzul(1,1)));
      posColor = Azulpos;
```

```
        tetaColor = tetaAzul;
        sentido = [Azulpos(1,3)-Azulpos(1,1)   Azulpos(1,4)-Azulpos(1,2)];
    end
    % Eliminar  los datos de video en la memoria
    stop(vid);
    flushdata(vid);
end

if tipo=='a' %Color Amarillo
    start(vid)

    while(vid.FramesAcquired<=10)
        Amarillopos=[];
        foto = getsnapshot(vid);
        data = imresize(foto,  [440  568]);

        %Deteccion de Color Amarillo
        color1 = imsubtract(data(:,:,1),  rgb2gray(data));
        color2 = imsubtract(data(:,:,2),  rgb2gray(data));
        amarillo  = color1+color2;
        amarillo  = medfilt2(amarillo,[3  3]);
        amarillo  = im2bw(amarillo,0.1);
        amarillo  = bwareaopen(amarillo,60);
        bw04 = bwlabel(amarillo,  4);
        stats4 = regionprops(bw04,'BoundingBox','Centroid');

        %Recorte de color Amarillo
        area1=getfield(stats4(1,1),'BoundingBox');
        punto1=round(getfield(stats4(1,1),'Centroid'));
        area2=getfield(stats4(2,1),'BoundingBox');
        punto2=round(getfield(stats4(2,1),'Centroid'));

        if area1(1,3)>area2(1,3)
            Amarillopos=[punto1 punto2];
        else
            Amarillopos=[punto2 punto1];
        end

        %Determinacion del angulo
        posAmarillo  = abs([Amarillopos(1,3)-Amarillopos(1,1)   Amarillopos(1,4)-Amarillopos(1,2)]);
        tetaAmarillo  = rad2deg(atan(posAmarillo(1,2)/posAmarillo(1,1)));
        posColor = Amarillopos;
        tetaColor = tetaAmarillo;
        sentido = [Amarillopos(1,3)-Amarillopos(1,1)   Amarillopos(1,4)-Amarillopos(1,2)];
    end
    % Eliminar  los datos de video en la memoria
    stop(vid);
    flushdata(vid);
end
end
```

ANEXO E - FUNCIÓN CREACIÓN DEL HABILITADOR

```
function [newhabilitador]=dethabilitador(habilitador)

newhabilitador = [];
recorrer = 1;
salto = 0;
[row,col]=size(habilitador);

for hab=1:(col*2)-1
   num = round(habilitador(1,2)/2);
   if salto==0
      newhabilitador(1,hab)=habilitador(1,recorrer);
      recorrer = recorrer+1;
      salto=1;
   else
      num = num+habilitador(1,recorrer-1);
      newhabilitador(1,hab)= num;
      salto=0;
   end
end
end
```

ANEXO F - FUNCIÓN DE DETECCIÓN DE SEGMENTOS

```
function [pos0,pos1]=detsegmentos(path)
% Deteccion de tramos rectos y diagonales para obtener P0 y P1
secx = 0;
secy = 0;
pos0=[];
pos1=[];
fin=1;
[largo,ancho]=size(path);

while fin<largo

   switch fin~=largo-1
      case fin<largo
         pos0=[pos0 path(fin,1) path(fin,2)];

         if path(fin+9,1)==path(fin+11,1)
            secx=1;
         else
            secx=0;
         end
         if path(fin,2)==path(fin+9,2)
            secy=1;
         else
            secy=0;
         end

         stop=0;
         ij=fin+9;
         jj=fin+9;
         yj=fin+1;

         if secx==1 && secy==0;
            while stop<2
```

```
            if ij==largo
               stop=2;
            elseif path(ij,1)~=path(ij+1,1)
               stop=stop+1;
               ij=ij+1;
            else
               ij=ij+1;
            end
         end
         fin=ij;
         pos1=[pos1 path(ij-1,1)  path(ij-1,2)];
      end

      if secx==0 && secy==1;
         while stop<4
            if jj==largo
               stop=4;
            elseif path(jj,2)~=path(jj+1,2)
               stop=stop+1;
               jj=jj+1;
            else
               jj=jj+1;
            end
         end
         fin=jj;
         pos1=[pos1 path(jj,1)  path(jj-1,2)];
      end

      if secx==0 && secy==0;
         while stop<1
            if yj==largo
               stop=1;
            elseif path(yj,2)~=path(yj+1,2)
               yj=yj+1;
            else
               stop=stop+1;
            end
         end
         fin=yj;
         pos1=[pos1 path(yj-1,1)  path(yj-1,2)];
      end

   otherwise
      fin=largo;
  end
end
end
```

ANEXO G - PERFIL TRAPEZOIDAL

```
function [vel,tiempo,angulo,habilitador]=ptrapezoidal(x,y)

%Variables de Inicio
vel=[];
tiempo=[];
angulo=[];
habilitador=[];
condicion=0;

[col,length]=size(x);

if length/2<2
   p0sn = [x(1,1) x(1,2)];
   p1sn = [y(1,1) y(1,2)];
   continuar = 0;
else
   p0sn = x;
   p1sn = y;
   continuar = 0;
end

p0n = single(p0sn)./400;
p1n = single(p1sn)./400;
next=1;

while continuar<length/2
   p0s = [p0n(1,next) p0n(1,next+1)];
   p1s = [p1n(1,next) p1n(1,next+1)];

   resx = p0s(1,1);
   resy = p0s(1,2);

   p0 = [p0s(1,1)-resx p0s(1,2)-resy];
   p1 = [abs(p1s(1,1)-resx) abs(p1s(1,2)-resy)];

   %Requerimientos
   pt = 0.30;
   tm = 10E-3;
   n = 10;

   vk = 0.038; % Velocidad maxima del robot
   T = (norm(p1-p0))/(vk*(1-pt)); % Tiempo de duracion del recorido

   k = round(pt*n);
   kc = round(n-2*k);

   %Linea recta en el espacio
   tao = T*pt;
   Dt = p1-p0;
   a = vk/tao;

      %Para conocer el punto intermedio Ptao
      xpt = (1/(2*norm(Dt))*a*(tao^2)*Dt(1,1))+p0(1,1);
      ypt = (1/(2*norm(Dt))*a*(tao^2)*Dt(1,2))+p0(1,2);

      %Para conocer el punto intermedio PT-tao
      xpT = (vk*(T-2*tao)/norm(Dt)*Dt(1,1))+xpt;
```

```
    ypT = (vk*(T-2*tao)/norm(Dt)*Dt(1,2))+ypt;

    %Se tiene como resultado los vectores Ptao y PT-tao
    Pt = [xpt, ypt]; %Ptao
    PT = [xpT, ypT] ;%P(T-tao)

%Tramo acelerado
Dt1 = Pt-p0;
T1 = (2*norm(Dt1))/vk;
a1 = vk/T1;

    %El vector de tiempo del tramo se define como
    x1 = [];
    y1 = [];
    dx1 = [];
    dy1 = [];

    time1 = [0:T1/k:T1];
    syms tk1

    c=(1/(2*norm(Dt1))*a*(tk1^2)*Dt1(1,1))+p0(1,1);
    c2=(1/(2*norm(Dt1))*a*(tk1^2)*Dt1(1,2))+p0(1,2);

    d1=diff(c,tk1);
    d2=diff(c2,tk1);

    for tk1 = 0:T1/k:T1;
        x1 = [x1 (1/(2*norm(Dt1))*a*(tk1^2)*Dt1(1,1))+p0(1,1)];
        y1 = [y1 (1/(2*norm(Dt1))*a*(tk1^2)*Dt1(1,2))+p0(1,2)];

        dx1 = [dx1 eval(d1)];
        dy1 = [dy1 eval(d2)];
    end

%Tramo velocidad constante
Dt2 = PT-Pt;
T2 = norm(Dt2)/vk;

    %El vector de tiempo del tramo se define como
    x2 = [];
    y2 = [];
    dx2 = [];
    dy2 = [];

    time2 = [0:T2/kc:T2];
    syms tk2

    c3=(vk*tk2/norm(Dt2)*Dt2(1,1))+Pt(1,1);
    c4=(vk*tk2/norm(Dt2)*Dt2(1,2))+Pt(1,2);

    d3=diff(c3,tk2);
    d4=diff(c4,tk2);

    for tk2 = 0:T2/kc:T2;
        x2 = [x2 (vk*tk2/norm(Dt2)*Dt2(1,1))+Pt(1,1)];
        y2 = [y2 (vk*tk2/norm(Dt2)*Dt2(1,2))+Pt(1,2)];

        dx2 = [dx2 eval(d3)];
```

```
        dy2 = [dy2 eval(d4)];
    end

%Tramo desacelerado
Dt3 = p1-PT;
T3 = 2*norm(Dt3)/vk;
a3 = -vk/T3;

    %El vector de tiempo del tramo se define como
    x3 = [];
    y3 = [];
    dx3 = [];
    dy3 = [];

    time3 = [0:T3/k:T3];
    syms tk3

    c5=(1/norm(Dt3)*((1/2*a3*tk3^2)+vk*tk3)*Dt3(1,1))+PT(1,1);
    c6=(1/norm(Dt3)*((1/2*a3*tk3^2)+vk*tk3)*Dt3(1,2))+PT(1,2);

    d5=diff(c5,tk3);
    d6=diff(c6,tk3);

    for tk3 = 0:T3/k:T3;
        x3 = [x3 (1/norm(Dt3)*((1/2*a3*tk3^2)+vk*tk3)*Dt3(1,1))+PT(1,1)];
        y3 = [y3 (1/norm(Dt3)*((1/2*a3*tk3^2)+vk*tk3)*Dt3(1,2))+PT(1,2)];

        dx3 = [dx3 eval(d5)];
        dy3 = [dy3 eval(d6)];
    end

% Vector de direccion
teta = rad2deg(atan(p1(1,2)/p1(1,1)));

Velx = [dx1 dx2 dx3];
Vely = [dy1 dy2 dy3];

vel = [vel sqrt(Velx.^2+Vely.^2)];
tiempo = [tiempo time1 time2+tk1 time3+tk1+tk2];

angulo= [angulo teta];

% Aclaracion de variables
next=next+2;
continuar=continuar+1;

[a,b]=size(tiempo);

if condicion <1
    habilitador=[habilitador 0];
    condicion = 1;
end
vel = [vel 0];
tiempo = [tiempo 0];
habilitador=[habilitador b];
%    angulo = [angulo 0];
end
end
```

ANEXO H – ESQUEMÁTICO