UNIVERSIDAD TECNOLÓGICA ISRAEL

TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE:

INGENIERO EN ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES

TEMA: IMPLEMENTACIÓN DE UNA PLATAFORMA ROBÓTICA MÓVIL DIFERENCIAL PARA SEGUIMIENTO DE TRAYECTORIA, MEDIANTE UN ALGORITMO DE CONTROL NO LINEAL.

AUTOR: MARTÍN DAVID LARCO CHIRIBOGA

TUTOR: Ing. RENÉ ERNESTO CORTIJO LEYVA, Mg

QUITO- ECUADOR

AÑO: 2019

UNIVERSIDAD TECNOLÓGICA ISRAEL

DECLARACIÓN

Yo, **MARTÍN DAVID LARCO CHIRIBOGA,** con cédula de identidad Nº 1724305105 declaro que este trabajo de titulación **"IMPLEMENTACIÓN DE UNA PLATAFORMA ROBÓTICA MÓVIL DIFERENCIAL PARA SEGUIMIENTO DE TRAYECTORIA, MEDIANTE UN ALGORITMO DE CONTROL NO LINEAL"** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico de trabajo de grado en mención.

QUITO D.M. Febrero del 2019

……

LARCO CHIRIBOGA MARTÍN DAVID

C.I: 1724305105

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación certifico:

Que el trabajo de titulación **"IMPLEMENTACIÓN DE UNA PLATAFORMA ROBÓTICA MÓVIL DIFERENCIAL PARA SEGUIMIENTO DE TRAYECTORIA, MEDIANTE UN ALGORITMO DE CONTROL NO LINEAL",** presentado por el **Sr. MARTÍN DAVID LARCO CHIRIBOGA**, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

QUITO D.M. Febrero del 2019

TUTOR

……

ING. RENÉ ERNESTO CORTIJO LEYVA

AGRADECIMIENTO

A todas las personas que estuvieron a mi lado apoyándome especialmente a mis padres los cuales han sido fuente de inspiración a lo largo de toda mi vida con su empeño, esfuerzo y dedicación, me dieron ese empuje para lograr finalizar este gran peldaño en mi carrera estudiantil y profesional, forjándome un futuro y dándome todo el conocimiento necesario para desenvolverme por mi propia cuenta en este gran campo profesional.

Eterno agradecimiento a mi hermano Carlos quien me ha dado todo su apoyo incondicional y con su conocimiento una gran ayuda para lograr finalizar mis estudios, quien en los momentos más difíciles siempre me ha dado su mano y su aporte muy bien apreciado por mi persona.

A todos los grandes profesionales que conforman la Universidad Tecnológica Israel quien es el ente promotor de nuevos profesionales, quien me aportó a lo largo de todos estos años mis conocimientos y sembró en mí la semilla del conocimiento, gracias a su gran aporte formó en mí las bases necesarias para responder a la sociedad, como un gran profesional.

DEDICATORIA

Dedico este trabajo de titulación a mis padres, quienes fueron los pilares principales en mi carrera profesional, con su gran apoyo, cariño, dedicación, esfuerzo y sin número de cualidades que ellos poseen, han forjado en mi persona las herramientas necesarias para culminar con mi vida profesional.

A mi hermano Carlos quien, con su aporte valioso me encamino para alcanzar esta gran meta profesional, a todos mis amigos y personas allegadas que me han brindado todo su apoyo.

Al gran personal capacitado de docentes de la Universidad Tecnológica Israel quienes poseen un acervo de conocimiento, y me han proporcionado las herramientas necesarias para desenvolverme de una manera óptima en el campo profesional.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto de titulación se enfoca en la investigación y desarrollo de un algoritmo de seguimiento de trayectoria para el posicionamiento de un robot móvil de un punto $X_{(0)}$ $Y_{(0)}$ a una posición $X_{(d)}$ $Y_{(d)}$ respecto al sistema de referencia tierra, el problema planteado es de control no lineal con restricciones no integrables el cual se reduce de las ecuaciones de *Euler-Lagrange* a un sistema de control dinámico de posición angular de los motores y un sistema de control cinemático, cuyo algoritmo de control será aplicado a una plataforma robótica móvil de tracción diferencial basada en el microcontrolador STM32F407 de 32 bits, el cual será programado mediante el software Matlab.

La realimentación de estados se basa en las ecuaciones de odometría conjuntas de los encoders y el sensor giroscópico acopladas mediante la ecuación cinemática del móvil, que permite obtener la posición y el desplazamiento angular del robot.

El diseño del hardware de la plataforma móvil se basa en la integración del microcontrolador STM32F407 el cual será programado mediante USB a través del microcontrolador STM32F103 por medio del software *ST-Link,* el cual es una ventana del software Matlab gracias a las librerías *Waijung*, el microcontrolador STM32F407 permite controlar los drivers de los motores, así como el sistema de comunicación serial *USART* (Trasmisor – receptor asíncrono universal) que transmite los datos al módulo *Bluetooth* integrado, la comunicación *I2C* (Circuito Inter-Integrado) que permite recolectar los datos del sensor inercial, los encoders acoplados a los motores, y todos los elementos pasivos que se requieren para el buen funcionamiento del hardware, también se desarrolló y se modeló en 3D la carcasa del móvil, así como la implementación de la *PCB* (Placa de Circuito Impreso) donde se montarán los elementos DIP y SMD (Montaje superficial).

Palabras claves: Robot móvil, seguimiento de trayectoria, control no lineal, librerías waijung

ABSTRACT

The present project of titulación focuses in the investigation and development of a algorithm of pursuit of trajectory for the positioning of a mobile robot of a point X (0) And (0) to a position X (d) And (d) respect to the ground reference system, the proposed problem is a non-linear control with non-integrable constraints which is reduced from the Euler-Lagrange equations to a system of dynamic control of the angular position of the motors and a kinematic control system, whose algorithm control will be applied to a mobile robotic platform of differential traction based on the 32-bit STM32F407 microcontroller, which will be programmed by the Matlab software.

The feedback of states is based on the combined odometry equations of the encoders and the gyro sensor coupled by the kinematic equation of the mobile, which allows to obtain the position and angular displacement of the robot.

The hardware design of the mobile platform is based on the integration of the STM32F407 microcontroller which will be programmed via USB through the STM32F103 microcontroller through the ST-Link software, which is a Matlab software window thanks to the Waijung libraries, the STM32F407 microcontroller allows to control the drivers of the motors, as well as the serial communication system USART (Transmitter - universal asynchronous receiver) that transmits the data to the integrated Bluetooth module, the I2C communication (Inter-Integrated Circuit) that allows to collect the sensor data Inertial, the encoders coupled to the motors, and all the passive elements that are required for the proper functioning of the hardware, the housing of the mobile was also developed and modeled in 3D, as well as the implementation of the PCB (Printed Circuit Board) where the DIP and SMD elements will be mounted (Surface mounting).

Keywords: Mobile robot, trajectory tracking, non-linear control, waijung libraries

INTRODUCCIÓN

ANTECEDENTES

El problema del seguimiento de trayectoria es un reto a nivel mundial dado que se trata de un problema de control no lineal con restricciones holonómicas, lo que hace al sistema de ecuaciones linealmente dependientes, por lo que representa un problema de consideración en la rama de control.

El reto principal es debido a que las ecuaciones del modelo del robot son acopladas y multivariables, lo que presenta inconvenientes en el control dado que para ello se deben desacoplar los estados lo cual genera que el tiempo de convergencia sea alto y la trayectoria seguida no sea óptima. Analizada esta problemática el proyecto se centra en realizar el control del robot móvil con el modelo acoplado del mismo.

El tratamiento de este problema es imperante ya que se aplica a un sin número de casos, ya sean industriales o domésticos, tales como robots de transporte de carga de un espacio a otro mediante el seguimiento de la trayectoria planificada por el operador, los sistemas previamente asistidos como es la conducción de robots CNC, también aplicaciones masivas como sistemas de aspiración y limpieza automáticas entre un sin número más de aplicaciones. Por lo que el Ecuador enfrenta un reto de investigación si quiere competir a nivel mundial en estas aplicaciones, de tal manera que se requiera enfrentar el problema de tracking y otros relativos a la robótica móvil.

PLANTEAMIENTO DEL PROBLEMA

Los sistemas autónomos que permitan llevar a cabo trayectorias determinadas para diversas funciones en amplios ámbitos, es uno de los más grandes inconvenientes que presentan los ingenieros al momento de desarrollar robots, ya que estos sistemas son complejos por ser no lineales con restricciones holonómicas y acoplar todas las ecuaciones en un solo sistema que rija el movimiento, no es tarea fácil.

La unificación de un algoritmo para lo cual se utilizará la cinemática del robot como la odometría y sus ecuaciones del movimiento angular conlleva una alta complejidad, estos sistemas son tratados como problemas de la rama del control automático.

Dado que el objetivo fundamental es llevar al sistema de una trayectoria actual a una trayectoria deseada, el primer problema es encontrar la posición del móvil en el sistema de

referencia tierra en sistemas *indoor* o sistemas en entornos internos donde la señal del GPS no es exacta y los métodos de navegación externos no son fiables dado que en interiores la precisión de movimiento del móvil debe ser de centímetros, de los métodos diseñados para sistemas *indoor* se ocupa el método llamado odometría *hodos* (viajar, trayecto) y *metron* (medida) la cual consiste en determinar la posición relativa del móvil durante el desplazamiento en función de la estimación de los parámetros cinemáticos (Navarro Merino, 2017), esto resulta de la integral de las velocidades generalizadas del modelo, por tanto implícito en el proceso lleva el error acumulativo integral de la velocidad.

El segundo problema consiste en desarrollar las ecuaciones de control multivariable que permitan seguir una trayectoria deseada con una convergencia del error de posición de forma que el:

$$\lim_{t\to\infty}(\mathrm{Xd(t)} - \mathrm{X(t)}) = 0$$

JUSTIFICACIÓN

La investigación sobre el seguimiento de trayectorias abre campos de investigación cuyas aplicaciones son ilimitadas, y la implementación de nuevas ideas permitan mejorar el proceso encargado de manejar las aplicaciones de forma óptima y más segura, por lo que cualquier camino que se tome en la investigación que revele datos sobre el problema puede ayudar a futuras investigaciones que aprovechen estos datos para hallar mejoras o soluciones al mismo.

El tratamiento de este proyecto es imperante ya que se aplica a un sin número de casos, ya sean industriales o domésticos, entre estos los robots que transportan carga que siguen una trayectoria dentro de los galpones de las fábricas, los sistemas previamente asistidos en la conducción de robots CNC, así como aplicaciones masivas, es el caso de los sistemas de aspiración y limpieza automáticas entre un sin número más de aplicaciones.

OBJETIVOS

Objetivo general

Implementar un robot móvil de tracción diferencial con el hardware para seguimiento de posición, sobre el cual se programará el algoritmo de control que permita realizar seguimiento de trayectoria.

Objetivos específicos

- Implementar el hardware de un robot móvil diferencial para fines académicos, que permita obtener los datos de posición y orientación del mismo, además de controlar los motores DC, se usará el microcontrolador STM de 32 bits.

- Realizar el modelado matemático del robot mediante las ecuaciones de *Euler-Lagrange.*

- Desarrollar el algoritmo de control que permita realizar el seguimiento de trayectorias.

- Programar el microcontrolador con el algoritmo de control con el software *Matlab*.

- Diseñar el *GUI* en *Matlab* que permita visualizar la trayectoria deseada y seguida, así como procesar la información estadística de la trayectoria.

ALCANCE

El presente proyecto se enfoca en el desarrollo de un algoritmo de seguimiento de trayectoria de un vehículo móvil, donde se unificará las variables en un solo sistema de ecuaciones que involucra la cinemática del robot, su dinámica, su odometría, y sus restricciones tanto holonómicas cono no holonómicas aplicado las ecuaciones de *Euler Lagrange*, a su vez se involucra el análisis del seguimiento de posición que permitirá visualizar la trayectoria que sigue el robot en tiempo real.

El algoritmo se implementará en la plataforma matemática (*MATLAB*) que me permite introducir todas las ecuaciones calculadas en un entorno de programación *Simulink*, la cual es una ventana del programa *MATLAB*, que permite realizar bloques de programación donde se configurará todas las ecuaciones y a través de las librerías *waijung* se puede programar directamente el microprocesador STM32F407.

Se implementará una plataforma móvil diferencial para en ella realizar las pruebas del algoritmo de control para el seguimiento de trayectoria mediante el microcontrolador STM32F407, como elementos utilizados para la realimentación de datos de velocidad angular y orientación se utilizará el giroscopio y encoder´s acoplados a las ruedas del robot,

por ser un sistema no linealmente acoplado, con lo cual me permitirá realizar las correcciones de posición respectivas.

En el programa MATLAB se desarrollará un *GUI* que me permita visualizar en tiempo real la trayectoria que tiene el robot en sus ejes de desplazamiento, la velocidad en sus dos ruedas y su posición en sus variables X, Y, θ, para él envió de los datos se utilizará la comunicación inalámbrica vía Bluetooth.

DESCRIPCIÓN DE LOS CAPÍTULOS

El Capítulo I Fundamentación Teórica, se dará a conocer todos los principales elementos electrónicos utilizados en la fabricación del hardware de la plataforma robótica móvil diferencial, a su vez los programas utilizados para el desarrollo del software y la teoría usada para el algoritmo, conjuntamente con todas las herramientas matemáticas que permitirán simplificar el trabajo.

El Capítulo II Marco Metodológico, se enfocará en el método de investigación realizado para llevar acabo el desarrollo e implementación del seguimiento de trayectoria.

El Capítulo III, se desarrollará toda la etapa de la propuesta para el diseño del algoritmo de control e implementación de la plataforma móvil robótica diferencial, que permita encontrar una solución para el seguimiento de trayectoria detallando las etapas que conllevará, y los programas a utilizar.

El Capítulo IV, se realizará la implementación donde se introducirá las ecuaciones encontradas en el programa MATLAB, se detallará el desarrollo de la plataforma robótica móvil diferencial utilizada para en ella probar el algoritmo de control, se configurará el GUI que me permita visualizar la trayectoria del robot, y pruebas del mismo.

CAPÍTULO I

FUNDAMENTACIÓN TEÓRICA

1.1 Robótica

Es una rama de la mecatrónica, impulsada hacia el desarrollo de máquinas autómatas que permitan realizar diversas funciones en el campo industrial, artesanal, domestico, etc. (Sanz Valero, 2006). La robótica en términos generales se inició desde la antigua Grecia donde desarrollaron sistemas o equipos autómatas utilizados para la fabricación, elaboración y desarrollo de la sociedad Griega, pero se la considero una rama de la ciencia alrededor de la década de los setenta con la invención del microprocesador el cerebro de los robots en la actualidad.

1.2 Robot móvil diferencial

Un robot diferencial es aquel robot que tiene solo dos ruedas montadas sobre un eje, cada rueda independientemente manipulada por un motor que proporciona tracción y dirección, dado por el cambio de velocidad entre cada rueda lateral para generar su giro. (Cuánticos, C, 2011)

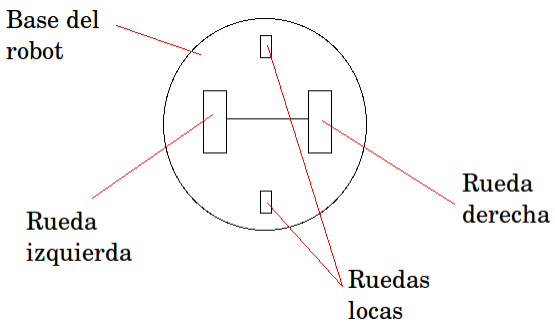


Figura. 1. 1 Tracción de un robot diferencial (Cuánticos, C, 2011)

La estabilidad del robot se logra con dos ruedas locas, sueltas sin ningún motor que produzca su tracción, el mismo será capaz de rotar con libertad con las ruedas sujetas a la base del robot.

El cambio de dirección se logra gracias a la rotación de las llantas del robot, dadas por la velocidad entre cada motor acoplado a la rueda, con una variación de velocidad angular

entre sus ruedas, con una rotación mayor en una de sus ruedas y una menor en la otra, de esta manera se desplaza en otra dirección. (Robots, 2014)

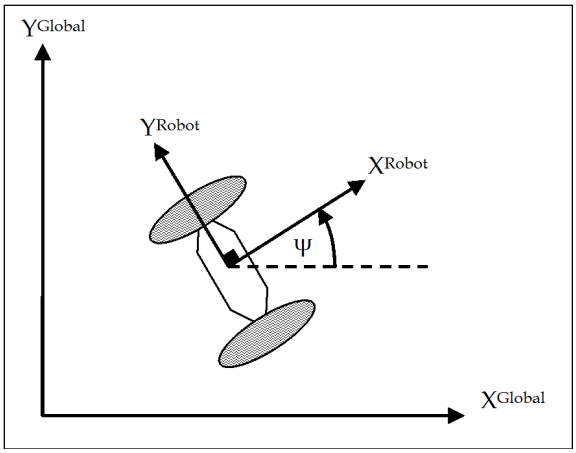


Figura. 1. 2 Rotación de un robot, cambio de dirección (Robots, 2014)

1.2.1 Movimiento del robot

El movimiento es el cambio de posición de un robot en el transcurso de un intervalo de tiempo. El robot puede tener diversos tipos de movimiento los cuales se describirán a continuación. (Hilario, J, 2016)

1.2.2 Rotatorio

Es el desplazamiento del robot en un plano de coordenadas X, Y, θ en el cual el robot gira 360°sobre su eje, para lograr este movimiento el desplazamiento de las ruedas debe ser a una misma velocidad, pero en sentidos contrarios, la una respecto a la otra. (Hilario, J, 2016)

Figura. 1. 3 Movimiento rotatorio (Hilario, J, 2016)

1.2.3 Giratorio

El robot se desplazará sobre una de sus llantas la cual se convertirá en su eje de giro, para moverse a la izquierda o derecha, rotará la rueda hacia delante o hacia atrás esto dependerá a donde vaya su movimiento si es hacia delante o hacia atrás. (Hilario, J, 2016)

Figura. 1. 4 Movimiento Giratorio (Hilario, J, 2016)

1.2.4 Doblar

El robot tendrá su eje de referencia fuera del mismo, para producir este movimiento el robot desplazará una de sus ruedas ya sea hacia delante o hacia atrás con mayor velocidad que la otra esto producirá que esta doble. (Hilario, J, 2016)

Figura. 1. 5 Doblar (Hilario, J, 2016)

1.3 Control de robots móviles

Los sistemas de control son ocupados en el análisis y diseño de componentes interactuantes en un sistema con configuración determinada que brinde un comportamiento deseado. En la teoría de sistemas de control la principal configuración se basa en el concepto de realimentación, el cual consiste en el proceso de medir las variables de interés y usar dicha información para controlar su comportamiento.

Las ventajas de un sistema de control eficiente en el campo industrial son inmensas, entre las que incluyen mejoras en la calidad de producción, reducción del consumo energético, minimización de residuos de desecho, mayores niveles de seguridad y reducción de emisiones contaminantes. El inicio en el análisis de un sistema de control es su representación del modelo matemático, generalmente como un operador entre entradas y salidas del sistema, o como un conjunto de ecuaciones diferenciales. La mayoría de los modelos matemáticos usados tradicionalmente en control son lineales. (Aguilera Hérnandez, Bautista, & Iruegas, 2007)

Los robots móviles comprenden un grupo de problemas cuyo modelo matemático es no lineal por lo que el tratamiento de las variables se realiza como una función de estados, que permita llevar al sistema de una posición X, Y, θ a una posición deseada en un tiempo dado.

Los métodos de control no lineal permiten obtener una respuesta estable en el sentido de *Lyapunov* lo que permite que el controlador pueda llevar al sistema a cualquier punto de forma estable, la generación de trayectorias entrega las coordenadas de los vectores, posición en función de un parámetro temporal, los cuales han de ser seguidos por este controlador, el desempeño del mismo se mide por la capacidad de reducir el error durante toda la trayectoria. (Ingeniería Técnica Industrial, 2008)

1.4 Seguimiento de trayectoria y seguimiento de camino

Se deben diferenciar dos conceptos los cuales permiten definir el caso sobre el cual se trabajará, ya que es común encontrar confusión entre los términos Camino y Trayectoria.

Camino: Es el conjunto de coordenadas generalizadas geométricas distribuidas en el espacio.

Trayectoria: Es un conjunto de coordenadas generalizadas parametrizadas en el tiempo.

1.5 Odometría

La odometría se utiliza para estimar la posición relativa de un robot móvil. En este caso, interesa la posición del robot en el plano, es decir, estimar el vector (x, y, θ). Como se observa hay dos palabras claves: estimación y relativa. Se hablará de estimación de posición ya que saber a ciencia cierta la posición de un robot móvil es sencillamente imposible, por

qué los métodos matemáticos utilizados para calcular una posición no tienen una precisión absoluta. La única forma de obtener información de un robot móvil y su entorno es a través de sensores. La información dada por los sensores será utilizada para aplicar modelos matemáticos y cálculos, que permitirán obtener medidas más exactas de la posición del robot, pero sin estar exentos de errores, ya que los valores enviados por los sensores varían esto se debe a la precisión de los mismos. (Navarro Merino, Aplicaciones de la construcción 3D: Odometría visual e integración con la realidad irtual, 2017).

1.6 Restricciones holonómicas

Se define como una restricción de movimiento producto de los vínculos cinemáticos del mismo, por ejemplo:

Una mesa inclinada sobre la cual los objetos se ven atados a las fuerzas de vínculo cinemático genera un vínculo expresado por la ecuación del plano de la mesa

$$ax + by + cz = cte$$

Una cuerda atada a una masa genera una restricción sobre la cual la longitud de la cuerda es invariante.

$$x\text{^}2 + y\text{^}2 = l\text{^}2$$

Una rueda que no se desliza mantiene un vínculo entre la velocidad angular y la velocidad lineal del centro de la rueda.

$$R * \dot{\theta} = V \text{ D}$$

Esta última restricción depende de las velocidades del sistema sin embargo se puede integrar dando como resultado la relación entre el desplazamiento angular y la distancia recorrida por lo que a este vínculo se lo conoce como holonómico o vinculo integrable, de forma que las restricciones que no puedan ser integradas se las denomina no-holonómicas. (Arechavaleta, 2010)

Por vinculo no se debe mal interpretar a las fuerzas de origen dinámico por ejemplo un planeta que gira alrededor del sol genera un plano de rotación y una trayectoria definida, de ninguna forma este plano o trayectoria conforman un vínculo cinemático ya que en este se deben descartar todos los vínculos de origen dinámico o producto de una fuerza fundamental.

1.7 Modelo matemático

Los robots móviles son sistemas físicos con restricciones holonómicas y no holonómicas, por tanto deben ser modelados de acuerdo a las ecuaciones físicas que describen el movimiento, es decir las leyes de newton, sin embargo estas leyes deben aplicarse a un móvil sometido a un sistema de referencia no inercial por lo que las ecuaciones que permiten el modelado se resumen en dos tipos, vectoriales y escalares, la primera de las cuales fue desarrollada por *Euler* conocida como el método de *NEWTON-EULER*, esta incluye las componentes de las fuerzas virtuales provocadas por el marco de referencia no inercial y la segunda desarrollada por *Lagrange* conocida como ecuaciones de *EULER-LAGRANGE*, la misma que permite utilizar un grupo de ecuaciones diferenciales escalares que incluyen en su formulación los efectos no inerciales. (Poznyak, 2005).

1.8 Ecuaciones de EULER-LAGRANGE

Las ecuaciones de *Euler LaGrange* son un grupo de ecuaciones diferenciales, las cuales representan la trayectoria física del sistema, la ventaja de la formulación deriva del hecho de que las ecuaciones permiten contar con un sistema analítico que ayuda a llegar conocer a las ecuaciones del movimiento Newtoniano en sistemas no inerciales mediante la expresión de energías cinéticas y potenciales, las cuales son escalares del sistema. (Rodriguez-Alfaro & Alorta-Garcia, 2014)

La ecuación de *Euler-LaGrange* se presenta a continuación:

$$\frac{d}{dx}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = Q_j + \sum_i \lambda i \frac{\partial f i}{\partial \dot{q}_j} \qquad \textbf{(Ecuación. 1.1)}$$

Donde:

$L = T - V$ Es la lagrangiana del sistema y es igual a la diferencia entre la energía cinética y la energía potencial.

q_j Son las coordenadas generalizadas del movimiento.

$\dot{q}_j$ Son las velocidades generalizadas que se obtienen al derivar temporalmente las coordenadas generalizadas.

Q_j Es la fuerza generalizada.

λi Son los multiplicadores de lagrange

fi son las restricciones no-holonómicas del sistema

El método de *Euler LaGrange* para restricciones cinemáticas no holonómicas permite encontrar las ecuaciones diferenciales no lineales que caracterizan el sistema físico sobre el cual se aplica este procedimiento. (Rodriguez-Alfaro & Alorta-Garcia, 2014). El modelo permite determinar el comportamiento del robot mediante la matriz de inercia, y coriolis, además permite añadir las ecuaciones no holonómicas de la restricción cinemática que son características de un robot cuya fricción en las ruedas impide que estas se deslicen, por lo cual el modelo obtenido es bastante preciso y puede compactarse de forma matricial para su análisis.

1.9 Variables de estado

El estado de un sistema dinámico es el conjunto más pequeño de variables (llamadas variables de estado) tal que, el conocimiento de esas variables en un determinado instante t0 junto con el conocimiento de los valores de la señal de entrada para los instantes t ≥ t0, permiten determinar el comportamiento y evolución del sistema para cualquier instante de tiempo t ≥ t0. (Perez & Escobar, 2008)

Las variables de estado son agrupadas en el llamado vector de estado y el espacio n-dimensional que determinan los posibles valores de esas variables, se denominan espacio de estados. (Perez & Escobar, 2008)

La dinámica del sistema se puede describir en función del valor del vector de estados y de la señal de entrada (se considera un sistema no autónomo) mediante la ecuación 1.2 que tendrá la forma: (Casado Fernández, 2005)

$$x(k+1) = f(x(k), u(k), k)$$
$$y(k) = g(x(k), u(k), k) \qquad \text{(Ecuación. 1.2)}$$

Donde la notación *ξ(k)* indica el valor tomado por *ξ* en el instante de tiempo *k*, y f, g pueden ser cualquier tipo de función.

Las variables de estado son un método multivariable que permite expresar un sistema con ecuaciones diferenciales de primer orden, con esto queda el camino libre para ocupar

métodos analíticos para discretizar las ecuaciones no lineales y obtener una respuesta discreta que pueda ser tratada por el microcontrolador. (Perez & Escobar, 2008)

La definición de variable de estado viene ligada a la definición de coordenada generalizada, por lo cual se puede expresar las ecuaciones de Euler-LaGrange a través de transformaciones para reducir el orden del sistema a uno.

1.10 Entorno de programación

Dado que tanto la descripción matemática como el algoritmo de control son matriciales el programa de mejor entorno de trabajo es MATLAB, a continuación, se explica brevemente que es y qué tipo de funciones maneja.

MATLAB es el nombre abreviado de *"MATRIX LABORATORY"*. Es un programa para realizar cálculos numéricos con vectores y matrices, y por tanto se puede trabajar también con números escalares (tanto reales como complejos), con cadenas de caracteres y con otras estructuras de información más complejas. *Matlab* es un lenguaje de alto rendimiento para cálculos técnicos, es al mismo tiempo un entorno y un lenguaje de programación. (Casado Fernández, 2005) Uno de sus puntos fuertes es que permite construir las propias herramientas reutilizables. Se puede crear fácilmente sus propias funciones y programas especiales (conocidos como M-archivos) en código *Matlab*, los pueden agrupar en *toolbox* (también llamadas librerías): colección especializada de M-archivos para trabajar en clases particulares de problemas.

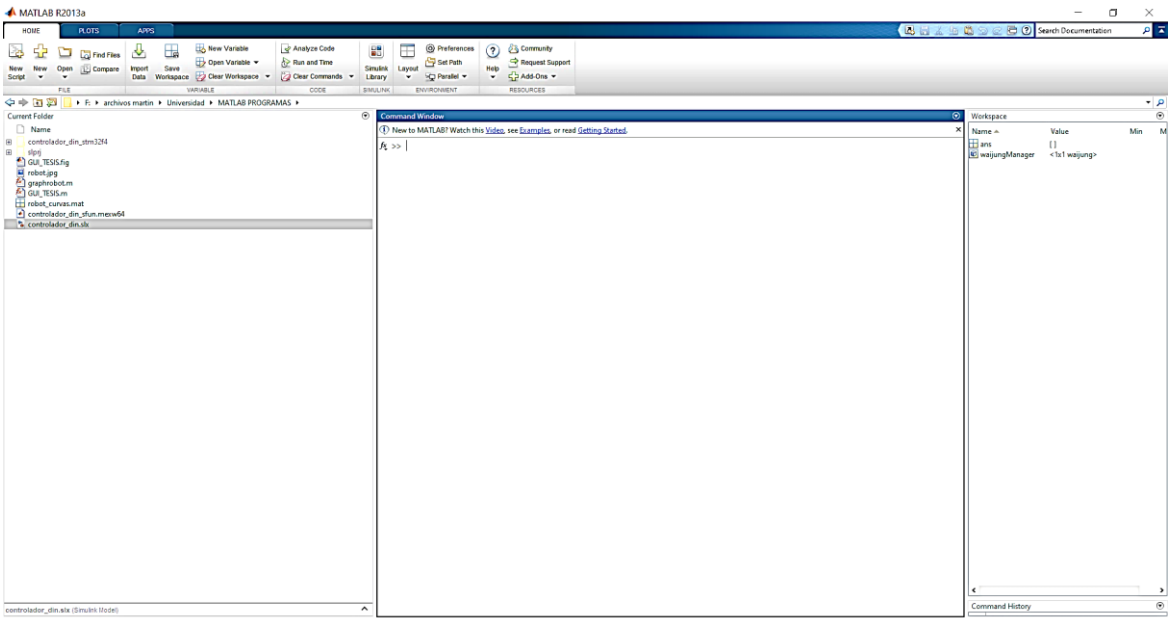


Figura. 1. 6 Ventana de programación MATLAB

Matlab permite visualizar las gráficas de las funciones y modificar su escala para observarlas de mejor manera, estas imágenes se las puede crear tanto en 2D como en 3D.

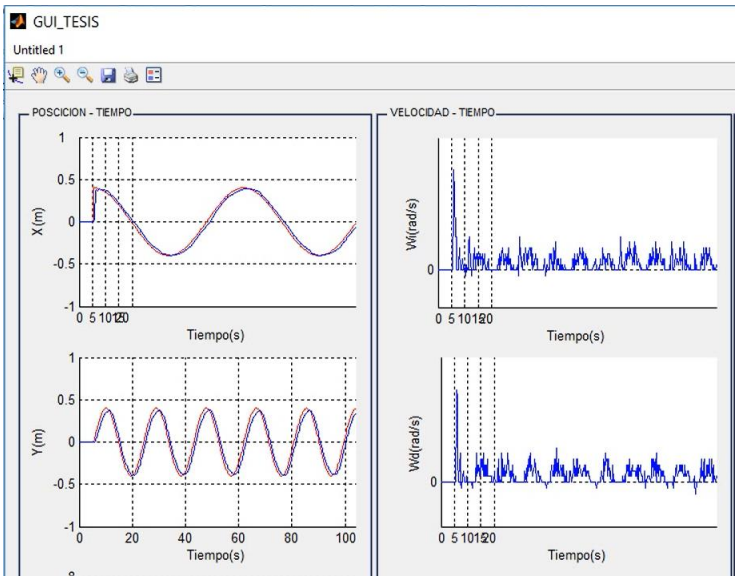


Figura. 1. 7 Diagrama de funciones 2D

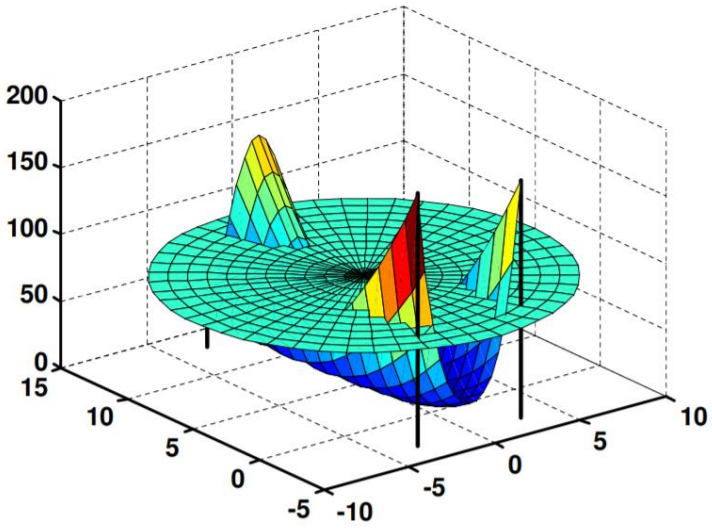


Figura. 1. 8 Diagrama de funciones 3D (Corcuera, 2015)

SIMULINK es una *toolbox* (librerías) especial de MATLAB que sirve para simular el comportamiento de los sistemas dinámicos. Puede simular sistemas lineales y no lineales, modelos en tiempo continuo y tiempo discreto y sistemas híbridos de todos los anteriores. Es un entorno gráfico en el cual el modelo a simular se construye dando click´s y desplazar los diferentes bloques que lo constituyen. (Casado Fernández, 2005).

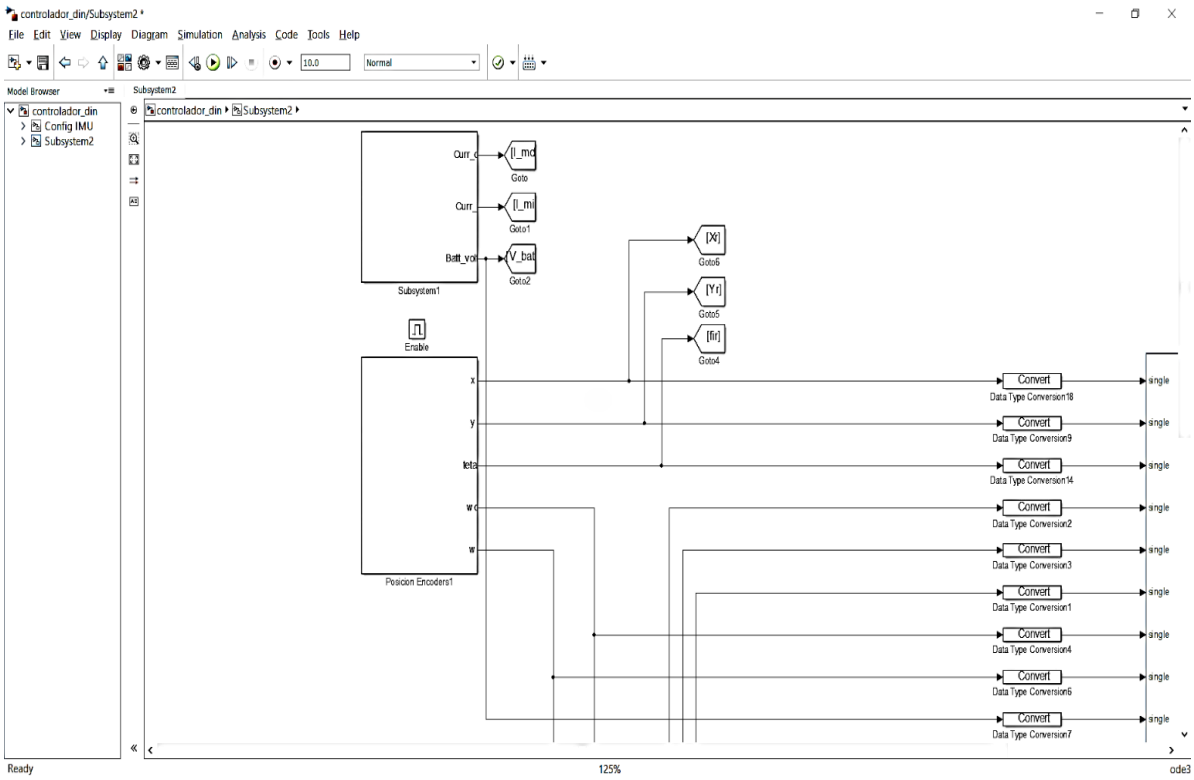


Figura. 1. 9 Simulink

1.11 Hardware

1.11.1 Microcontrolador STM32F407

Los microcontroladores (MCU) Cortex™-M4 de 32 bits STM32F4 de *STMicroelectronics* abren la puerta al mercado de los controladores de señal digital (DSC) con una familia de dispositivos cuyo software y modelo pin a pin son compatibles con la serie STM32F2 pero con mejor rendimiento, capacidad de DSP, más SRAM y mejoras de periféricos como dúplex completo I²S, RTC de menos de 1 µA, y ADC de 2,4 MSPS. Los MCU STM32F4 de *STMicroelectronics* incluyen una unidad de punto de flotación y características principales como instrucciones de ciclo único acumulado múltiple (MAC) integrado, aritmética SIMD optimizada e instrucciones aritméticas de saturación. El acelerador adaptativo de tiempo real ART *Accelerator*™ combinado con la tecnología de 90 nm de *STMicroelectronics* ofrece un rendimiento lineal de 168 MHz, que libera el rendimiento total del núcleo.

Estas prestaciones amplían la cantidad de aplicaciones accesibles en los segmentos de la industria, del consumidor y de la atención sanitaria. Los MCU de la serie STM32F4

incluyen dispositivos de memoria flash en chip con 512 KB hasta 1 MB, 192 KB de SRAM y 15 interfaces de comunicación. (Stmicroelectronics, 2012)

El STM32 es una familia de circuitos integrados de microcontroladores basados en los núcleos RISC ARM Cortex-MxF, de 32 bits. Los diseños de núcleo de ARM tienen numerosas opciones configurables, y ST elige la configuración individual para utilizar en cada diseño. ST conecta sus propios periféricos al núcleo antes de convertir el diseño en un dado de silicio. La velocidad de trabajo de los microcontroladores es mayor a los 100Mhz lo cual permite implementar un algoritmo de control multivariable sin comprometer la velocidad del mismo. (Stmicroelectronics, 2012)

Figura. 1. 10 Microcontrolador STM32F407 (PicClick, 2018)

Características

- El acelerador ART habilita el estado de espera 0 ejecutándose desde *Flash*
- Interno más de 2x USB2.0 OTG FS/HS
- SDIO
- USART, SPI, I²C
- 16-bit y 32-bit timers
- Por encima de 3x 12-bit ADC
- Por encima de 2x 12-bit DAC
- Controlador de memoria interna
- 1.7V a 3.6V bajo voltaje.

1.11.2 Encoders

El encoder es un transductor rotativo, que mediante una señal eléctrica (normalmente un pulso o una señal senoidal) indica el ángulo girado. Si este sensor rotatorio se lo conectara mecánicamente con una rueda o un husillo, también permitiría medir distancias lineales. (Ingeniería Mecafenix, 2017). Una clasificación de los encoder según el tipo de información sobre la posición que generan sería:

• Encoder incremental: La señal de salida se transmite por un hilo en el que se transmite un pulso por cada ángulo girado, de tal forma que, si se tiene un encoder de 1000 ppr, se tendrá un pulso por cada 360º/1000= 0, 360º. El inconveniente es que no se dispone de una referencia absoluta de la posición en la que se encuentra el eje. (Ingeniería Mecafenix, 2017).

• Encoder absoluto: La posición se da en valor absoluto mediante un bus paralelo. Es decir, que, si se tiene un encoder de 256 posiciones, se tendrá un bus de 8 líneas que indicará en binario cuál es su posición (normalmente estos transductores codifican la posición en código gray para evitar errores). El inconveniente de estos encoder´s es la cantidad de líneas que necesitan leer y conectar, debido a la complejidad del disco óptico que codifica las posiciones la resolución no suele ser muy elevada. (Ingeniería Mecafenix, 2017).

Un codificador rotatorio, también llamado codificador del eje (Encoder), o generador de pulsos, suele ser un dispositivo electromecánico usado para convertir la posición angular de un eje a un código digital, lo que lo convierte en una clase de transductor. Este codificador permite aplicaciones de control velocidad en los motores DC y permiten realizar odometría sobre la cual el modelo cinemático del robot puede funcionar con mucha precisión.

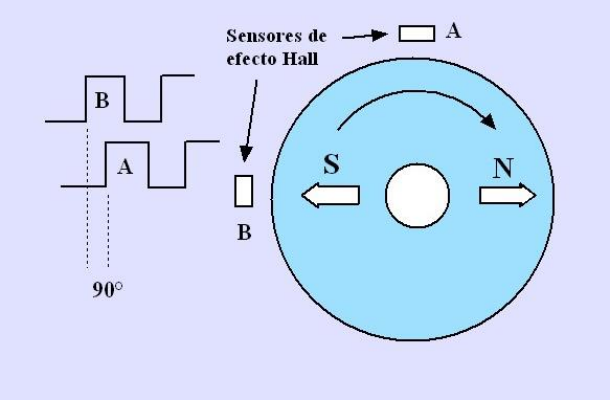


Figura. 1. 11 Encoder en cuadratura de efecto HALL (Puntoflotante.net, 2013)

1.11.3 Comunicación de RF (Radio Frecuencia)

Dado que el robot posee baterías que permiten su desplazamiento sin restricción de cables que añadan perturbaciones al sistema, este debe comunicarse vía inalámbrica, para ello se ocuparán módulos RF de transmisión para enviar y recibir los datos de la PC y ser procesados.

El sistema de comunicación inalámbrico utilizado será transmisión vía Bluetooth, para ello se utilizará el módulo HC-05 FC-114

Bluetooth es una especificación tecnológica para redes inalámbricas que permite la transmisión de voz y datos entre distintos dispositivos mediante una radiofrecuencia segura (2,4 GHz**).** Esta tecnología, por lo tanto, permite las comunicaciones sin cables ni conectores y la posibilidad de crear redes inalámbricas domésticas para sincronizar y compartir la información que se encuentra almacenada en diversos equipos. (González, 2013)

1.11.4 Distribución de pines HC-06

VCC, Voltaje positivo de alimentación, su mayoría ya vienen acondicionados para que trabajen en el rango de 3.3V a 6V.

GND, Voltaje negativo de alimentación, se tienen que conectar al GND de la placa que este use.

TX, Pin de Transmisión de datos, por este pin el HC-06 transmite los datos que le llegan desde la PC o Móvil mediante Bluetooth.

RX, pin de Recepción, a través de este pin el HC-06 recibirá los datos.

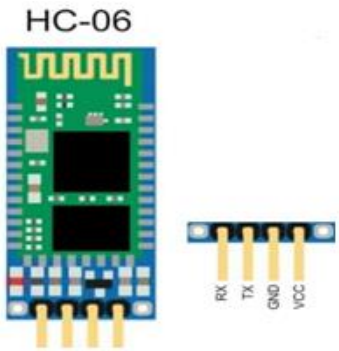


Figura. 1. 12 Módulo HC-06 pines de distribución (JSumo, 2017)

1.11.5 Motores DC

Se utilizó el Pololu 3072 "*Metal Gear motor*" un motor con eje extendido para acople de los encoder´s, viene con caja reductora acoplada 30:1, es un motor pequeño debido a las dimensiones de la práctica, pero de alta potencia, con escobillas de carbón de larga duración. (POLOLU, 2001)

Figura. 1. 13 Motor DC

Parámetros:

- Voltaje de trabajo 6V
- Rendimiento sin carga 1100 RPM con un consumo de corriente de 100mA
- La extrapolación de puesto 0.45 kg*cm con un consumo de corriente de 1.5 A

1.11.6 Driver L298P

Este driver permite controlar dos motores de corriente continua y motores paso a paso de no más de 2 amperes, existen algunos módulos disponibles para la compra que ya vienen con el driver soldado y pines para controlar las entradas y las salidas. (Stmicroelectronics, 2000)

Parámetros:

- Voltaje de entrada lógico VD: 5V
- Voltaje de entrada del variador: VIN 6.5 - 12V, PWR IN 4.8 - 24V
- Emisión actual de trabajo lógico: ≤ 36mA
- Conduzca la corriente de trabajo Io: ≤ 2A
- Disipación máxima de potencia: 25W (T = 75 ° C)

- Señal de control de entrada de nivel eléctrico:
- Nivel alto: 2.3V≤ Vin ≤ 5V
- Nivel bajo: -0.3 V ≤ Vin ≤ 15V
- Temperatura de funcionamiento: -25 ° C ~ + 130 ° C

Figura. 1. 14 Driver de los motores

Características:

- Hay un chip controlador de motor L298P en la placa para que pueda usar los dígitos IO interfaz (D9. D10. D11. D12) sin conexión de cableado engorroso.
- El zumbador incorporado (D4), puede establecer el tono de llamada de la alarma de popa.
- Interfaz de motor conveniente puede ser dos rutas de salida del motor.
- La interfaz de Bluetooth de dos vías no requiere cableado y puede enchufarlo directamente.
- Tiene siete interfaces digitales que no están ocupadas (entre las cuales están D 2, D 3, D 5, D 6, D 7, D 9).
- Cuenta con seis interfaces analógicas (A0, A1, A2, A3, A4, A5).
- Dispone de indicador para cambiar de dirección hacia adelante y hacia atrás.

1.11.7 Unidad de movimiento inercial MPU-9150

Figura. 1. 15 MPU-9150

- Chip: MPU-9150
- Alimentación: 3V / 5V
- Giroscopio de tres ejes + Acelerómetro Triaxial + Campo Magnético Triaxial
- Sensor Velocidad Angular
- Rango de escala completa programables por el usuario de ± 250, ± 500, ± 1000, y 2000°/seg
- Mejora el rendimiento de ruido de baja frecuencia
- Filtro de paso bajo programable digitalmente
- Calibrado de fábrica sensibilidad factor de escala
- Fondo de escala programable de ± 2G, ± 4G, ± 8G y ± 16G
- Rango de medición y alta resolución con menor consumo de corriente.
- Datos de salida AD 16 bits
- Rango de medición magnético de ± 1200uT

1.11.8 Giroscopio efecto giroscópico

El acelerómetro es un sensor muy susceptible al ruido pero presenta un valor absoluto respecto a la tierra que le otorga mucha precisión, mientras que el giroscopio es un sensor muy poco susceptible al ruido, sin embargo al medir velocidad angular es necesario integrar para encontrar teta, y dado que la integral acumula error, el valor del giroscopio tiene una deriva por lo que es necesaria una función sensorial. (Olmo & Nave, 2002)

CAPITULO II

MARCO METODOLÓGICO

2.1 El tipo de investigación

La investigación desarrollada en el presente trabajo es teórica práctica (Sierra, 2012), ya que por una parte se desarrollan los algoritmos de control que permiten realizar el seguimiento de trayectoria y por otro lado contempla la implementación de la plataforma robótica diferencial que servirá como plataforma de pruebas.

2.2 Diseño de la investigación

La metodología utilizada en este proyecto es el método deductivo (Sierra, 2012), el cual parte de las ecuaciones de movimiento desarrolladas con el método de *Euler-Lagrange* y de él se deducen las ecuaciones dinámicas del movimiento, las cuales por cambio de variable se pueden dividir en dos controladores, uno cinemático el cual controla todas las variables acopladas de la posición del robot y otro dinámico que controla la velocidad angular de las ruedas y arrastran toda la dinámica del sistema de forma que se puede reducir el problema a partir del análisis del modelo, además conforme avanza el análisis de las ecuaciones se puede encontrar las ecuaciones de odometría que permiten obtener la posición del móvil durante la navegación y configuran los requisitos del hardware para la implementación del mismo.

En el diseño de la plataforma móvil para las pruebas del algoritmo se utilizará el método deductivo al aplicar conocimientos sobre elementos electrónicos ya establecidos sobre la fabricación de un robot.

A su vez se aplicará el método experimental para la calibración de las constantes obtenidas durante el desarrollo del robot y que configuran el desempeño del algoritmo de control, dado que estas constantes del controlador son parte de un modelo no lineal no se puede encontrar el valor numérico exacto de las mismas, por lo que se requieren varias pruebas de experimentación hasta encontrar las magnitudes adecuadas de las constantes del controlador.

CAPITULO III

PROPUESTA

3.1 Idea de la propuesta

Se implementará una plataforma móvil de tracción diferencial con un microcontrolador de 32 bits STM32F407, la cual servirá para obtener la posición del móvil mediante odometría y procesar las ecuaciones de control no lineal que permiten seguir una trayectoria parametrizada de una curva.

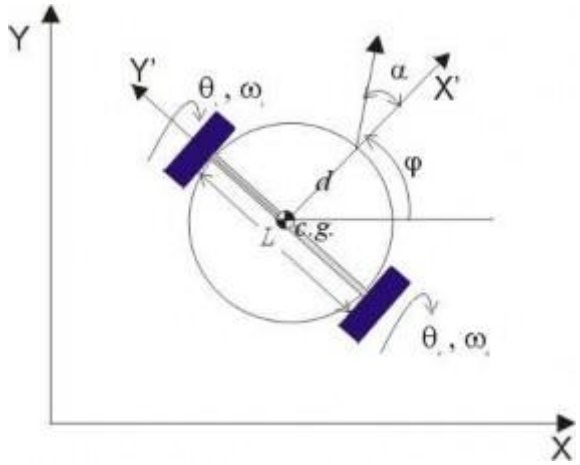


Figura. 3. 1 Variables de posición que actúan en el robot (Rodriguez, 2016)

3.2 Idea integral de la propuesta

Se desarrollará el algoritmo de control que permitirá realizar el seguimiento de trayectoria, y las ecuaciones que intervienen en el modelo matemático del robot mediante las ecuaciones de control multivariable no lineal e implementar las mismas en la ventana de *SimuLink* en *Matlab* para compilar el código a través de las librerías *Waijung* y descargar el código en el microcontrolador STM32F407, el cual será encargado de procesar todos las variables y realizar el desplazamiento a través de una trayectoria definida.

Las ecuaciones del sistema desarrolladas se implementarán en un entorno de simulación matemático *SimuLink* del programa MATLAB, y se programará a través del

mismo por medio de comunicación I2C hacia el microcontrolador del robot, el cual leerá el programa y ejecutará las sentencias.

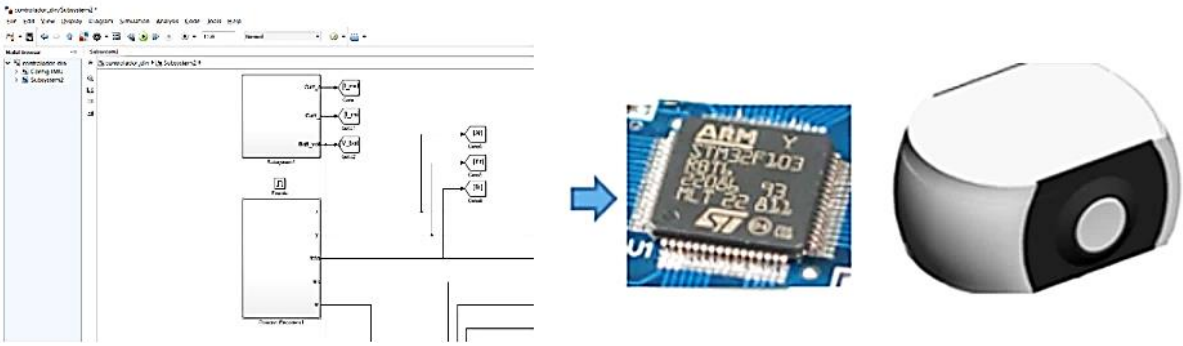


Figura. 3. 2 Esquema de programación del robot

El algoritmo de desplazamiento lo realizará el microcontrolador por lo cual para visualizar dicho procesamiento y diagramar su desplazamiento y posicionamiento en sus ejes, se desarrollará un GUI en el programa MATLAB.

El GUI es una interfaz gráfica de usuario que permite interactuar con el software sin necesidad de aprender el lenguaje de programación para hacer cambios en la aplicación, por medio de *click´s*, lo que ayuda a simplificar el uso de la herramienta, para no tener que entrar en un lenguaje de programación más profundo.

La comunicación para el envío de los datos de velocidad angular y posición angular para el gráfico de trayectoria es mediante el módulo HC-06, el cual es un módulo Bluetooth, que se comunica a otro dispositivo bluetooth conectado a la PC, mediante un puerto COM, permitiendo captar todas las señales necesarias para la graficación.

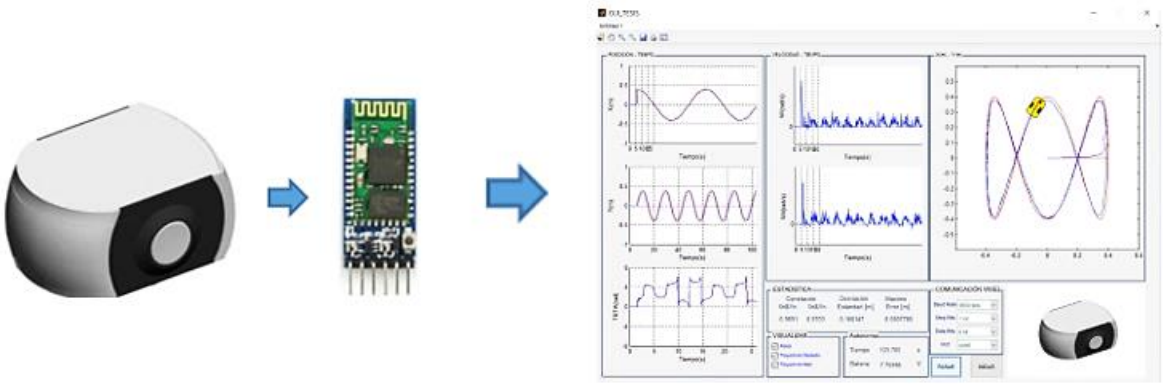


Figura. 3. 3 Esquema de envío de información

3.3 Módulos de la propuesta

Se plantea los diferentes módulos que compondrán el desarrollo tanto de la parte del hardware como de la parte del software, que permitirá cumplir con los objetivos planteados, para esto el proyecto se subdividirá en varias etapas que arán más fácil el desarrollo del mismo, se empezará por la etapa de comunicación para programar el robot y él envió de información desde la plataforma robótica hacia la PC, el módulo de alimentación para suministrar de energía a cada uno de los elementos que componen la parte física, un subsistema para el control de los motores que permitirán el desplazamiento del robot.

Entre estos módulos se destaca el módulo de control, el responsable de realizar todos los procesos para ejecutar el software del robot y poder trasladarlos a la parte del hardware del móvil que va a ser el encargado del movimiento, entre otros pequeños módulos adicionales que usara el robot, pero los más importantes serán los antes mencionados.

3.4 Modulo de comunicación

Se programará mediante un cable USB desde la PC hacia el micro STM32F103 el cual será el encargado de enviar todos los datos al microcontrolador STM32F407, esto por medio del protocolo I2C, para la comunicación del microcontrolador al dispositivo Bluetooth se realizó por medio del protocolo USART.

Como esbozo se puede visualizar en las figuras 3.2 y 3.3 el envío de datos desde la PC al robot y viceversa, de esta manera se tiene un conocimiento más claro de qué forma de comunicación se va a llevar a cabo.

3.5 Módulo de alimentación

Para esta etapa se plantea usar una batería LIPO (Litio Polímero), ya que las mismas manejan una carga de corriente mucho más alta que las baterías convencionales esto permitirá alimentar todo el robot y a su vez proveer un tiempo relativamente alto de corriente a todo el sistema, para realizar las pruebas debidas a la plataforma, otra ventaja de estas baterías es que son recargables su vida útil se extiende por largo tiempo, mientras la misma se la tenga regularmente cargada.

Figura. 3. 4 Batería LIPO

3.6 Módulo de control

EL microcontrolador STM32F407 será utilizado como corazón principal de toda la plataforma robótica móvil diferencial, encargado de procesar el algoritmo de control que se desarrollará, a su vez de enviar los datos necesarios a los módulos externos y de recibir los mismos, es el elemento encargado de ejecutar todas las líneas de comando.

Este microcontrolador permite la comunicación directa desde la PC con el programa de simulación matemática *MATLAB* hacia el mismo, gracias a la utilización de las librerías denominadas *waijung* permitiendo enviar las líneas de programación directamente hacia el STM32F407, a su vez cuenta con 100 pines de múltiple función como PWM, registros de entrada y salida, para diversos protocolos de comunicación entre estos USART e I2C.

3.7 Módulo de potencia

Encargado de enviar la señal hacia los motores que permitirá el movimiento del robot, para este módulo se utilizará el *driver* de motores L298P el cual trabaja a 5V y permite manejar una corriente de 2A por salida si se considera que los motores a utilizar serán los *POLOLU Gear Motor* con eje extendido para conexión de dos encoders, uno por cada lado, y los mismos consumen un máximo de 1.5A cuando se encuentran trabados.

El microcontrolador mediante la PWM será el encargado de enviar los pulsos al *driver* L298P el mismo que enviara esa señal a cada uno de los motores, según determiné la línea de control para selección de las respectivas salidas, que en este caso serán solo dos, este módulo contendrá una etapa de protección que permitirá proteger el micro ya que cuando se trabaja con rotores con escobillas se generará una corriente de inversa que puede quemar los elementos.

3.8 Modelado matemático

Se utilizará las ecuaciones de Euler-Lagrange en la resolución del sistema, considerando las restricciones no holonómicas y holonómicas, para encontrar las ecuaciones que determinen la dinámica, cinemática y odometría del sistema.

Todas las ecuaciones que rigen el sistema de trayectoria serán programadas en MATLAB, a través de las librerías waijung blockset, en la ventana Simulink, un espacio de programación que permite mediante bloques introducir todo el sistema de ecuaciones no lineales.

3.9 Softwares utilizados

Para el desarrollo de todo el proyecto se utilizará tres programas diferentes que permitirán realizar el diseño del software y del hardware, los cuales se describen a continuación:

3.9.1 Proteus 8.1

Es un programa de diseño y simulación de circuitos electrónicos, que contiene una amplia librería de elementos de diversos fabricantes, que permitirá desarrollar el hardware de la plataforma robótica móvil diferencial, mediante un sistema esquemático que logra unir los diversos componentes, visualizar su comportamiento y modificar sus valores antes de realizar pruebas físicas en un *protoboard* y posteriormente la placa final en PCB,

Dentro de este programa se destacan dos ventanas principales denominadas ISIS que es donde se desarrolla toda la conexión, simulación y con ello la programación de todos los elementos electrónicos de los cuales se compondrá el hardware, la otra ventana denominada ARES que permite el desarrollo de la PCB que se imprimirá luego en una baquelita, este programa traslada todos los elementos conectados en ISIS a una conexión de pistas donde se puede dar desde la forma de la placa, ubicación de elementos y diseño de la misma.

3.9.2 3D STUDIO MAX

Es un programa utilizado para la creación de objetos sólidos o figuras en 3D, con el objetivo de modelarlos, a su vez permite grabar el archivo en un formato que puedan ser impresas en 3D, toda la carcasa del robot se diseñará en este software, la cual protegerá la placa y permitirá colocar de forma ordena los diversos elementos que la componen, y también de darle un toque estético a la plataforma.

3.9.3 MATLAB

Es un entorno de simulación matemática donde se puede introducir ecuaciones para que el programa logre encontrar la solución de estas, se puede introducir todo tipo de parámetros a través de una de sus ventanas SIMULINK que permite programar mediante bloques, facilitando la programación total del algoritmo debido a que cada bloque contiene una función específica desde derivadas hasta integrales, lo cual reduce la complejidad en el uso de esta herramienta, optimizando el trabajo en la elaboración del proyecto.

Se utilizará otra de las pestañas del programa MATLAB para el desarrollo del GUI, que permitirá visualizar la trayectoria que sigue el robot vs la trayectoria planteada, a su vez visualizar las señales en el tiempo de las posiciones X, Y, θ y las velocidades angulares de cada motor en función del tiempo.

3.10 Aspectos técnicos del proyecto

Se desarrollará el software para seguimiento de trayectoria y una plataforma móvil diferencial con la que se puede hacer las pruebas de este, que permitirá realizar las pruebas de validación del algoritmo para medir el error entre la trayectoria seguida y la trayectoria establecida, este proyecto cuenta con dos partes hardware y software.

El hardware se compondrá de la etapa del PCB y soldadura de los elementos y de la carcasa.

- Tarjeta diseñada en *proteus* en impresa *solder mask*, con corte en CNC.
- Módulos de comunicación USB, bluetooth.
- 2 motores POLOLU.
- Batería LIPO.
- Microcontrolador STM32F407.

El software se desarrollará en el programa MATLAB:

- Desarrollo del algoritmo de control.
- Desarrollo del GUI para visualización de la trayectoria.

3.11 Ventajas del proyecto

Este proyecto permite abrir algunos campos de desarrollo en el ámbito de la robótica móvil, permite acoplar otros estudios como son el planeamiento de trayectoria y la evasión de obstáculos, estos sistemas acoplados dan cabida a los principios de sistemas más complejos como aspiradoras automáticas, equipos de la industria para el traslado de objetos, sistemas automatizados que se trasladen en diferentes direcciones, entre otros.

El desarrollo del proyecto logra un avance tecnológico amplio tanto en el campo industrial, como doméstico, también abre un campo de estudio para prácticas de estudiantes en el análisis del ámbito del seguimiento de trayectoria, el plan engloba un sinfín de posibilidades ya que permite determinar trayectorias dadas a un sistema móvil, que logrará realizar un sin número de tareas específicas que se desee realizar, considerando el concepto de un robot automático.

3.12 Análisis de costos

Los elementos principales que se requieren para este proyecto son los microcontroladores, el corazón de todo el robot el cual ejecutará todo el programa se compilará y el mismo ejecutará las sentencias, a su vez se requiere su programador para enviar el programa desde la PC directo hacia el micro.

Todos los elementos cotizados y comprados serán en SMD debido a la cantidad de los mismos y al peso que producirían, ya que las ecuaciones del sistema involucran el peso y el centro de masa de toda la estructura, por tales razones no se puede comprar elementos normales, si no que se debe optar por la opción de los más pequeños y livianos.

Los motores utilizados serán los *metals gear motors* por su eje extendido para el acople de los encoders que enviarán las señales de rotación de los motores, lo cual permitirá establecer su posición, a su vez se utilizarán encoders en cuadratura uno en cada uno de los motores.

La carcasa será impresa en 3D por su dureza, maleabilidad y principalmente por el peso que tendrá esta, ya que mientras más liviano toda la estructura tendrá menos peso y será más fácil poder distribuir el centro de masa justo en el medio del eje de las ruedas, donde está el eje de giro de los motores.

A continuación, se detalla los costos de la compra de elementos, de la realización de la placa, carcasa desarrollo del software, y todos los rubros que se han generado para la realización de este proyecto, abra elementos que no se consideraron como específicos en el listado, pero se colocó un rubro que los contendrá a todos.

Tabla. 3. 1 Detalle de gastos del proyecto

DETALLE GASTOS HARDWARE			
CANTIDAD	**DETALLE**	**COSTO U**	**COSTO TOTAL**
1	STM32F407 EN SMD	$50	$50,00
1	STM32F103 EN SMD	$45	$45,00
5	LED SMD	$0,50	$2,50
30	RESISTENCIAS VARIOS VALORES	$0,15	$4,50
2	MICRO GEAR MOTORS	$15	$30,00
2	ENCODER EN CUADRATURA	$9,82	$19,64
1	ULN2803	$2,30	$2,30
1	IMPRESIÓN CARCASA ROBOT	$150	$150,00
1	CAPACITOR POLARICADO SMD 20V	$0,50	$0,50
1	MPU-9150	$28	$28,00
1	BATERIA LIPO 7.5 V 230mAh	$12	$12,00
2	SOCKETS DE 2 PINES	$0,20	$0,40
2	SOCKETS DE 3 PINES	$0,30	$0,60
2	SOCKETS DE 4 PINES	$0,40	$0,80
6	INDUCTANCIA SMD	$0,15	$0,90
2	RUEDAS	$4	$8,00
6	BORNERAS	$0,35	$2,10
3	ESPADINES	$0,35	$1,05
1	ESTAÑO	$0,30	$0,30
1	POMADO	$3,00	$3,00
1	CHUPA SUELDA	$3,25	$3,25
2	BRACKET	$4,02	$8,04
1	Driver DRV8833	$8,04	$8,04
2	REGULADOR AMS1117	$4,30	$8,60
2	LLANTAS	$4,00	$8,00
1	JUEGO DE CABLES	$4,50	$4,50
2	PCB SOLDER MASK	$30,00	$60,00
1	EXTRAS	$100,00	$100,00
		TOTAL	$562,02
DETALLE GASTOS SOFTWARE			
CANTIDAD	**DETALLE**	**COSTO U**	**COSTO TOTAL**
1	MATLAB 2013	$150	$150
1	ASESORIA	$300	$300
		TOTAL	$450

COSTO TOTAL DEL PROYECTO		
TOTAL HARDWARE	$562,02	$562,02
TOTAL SOFTWARE	$325	$450
	TOTAL	$1012,02

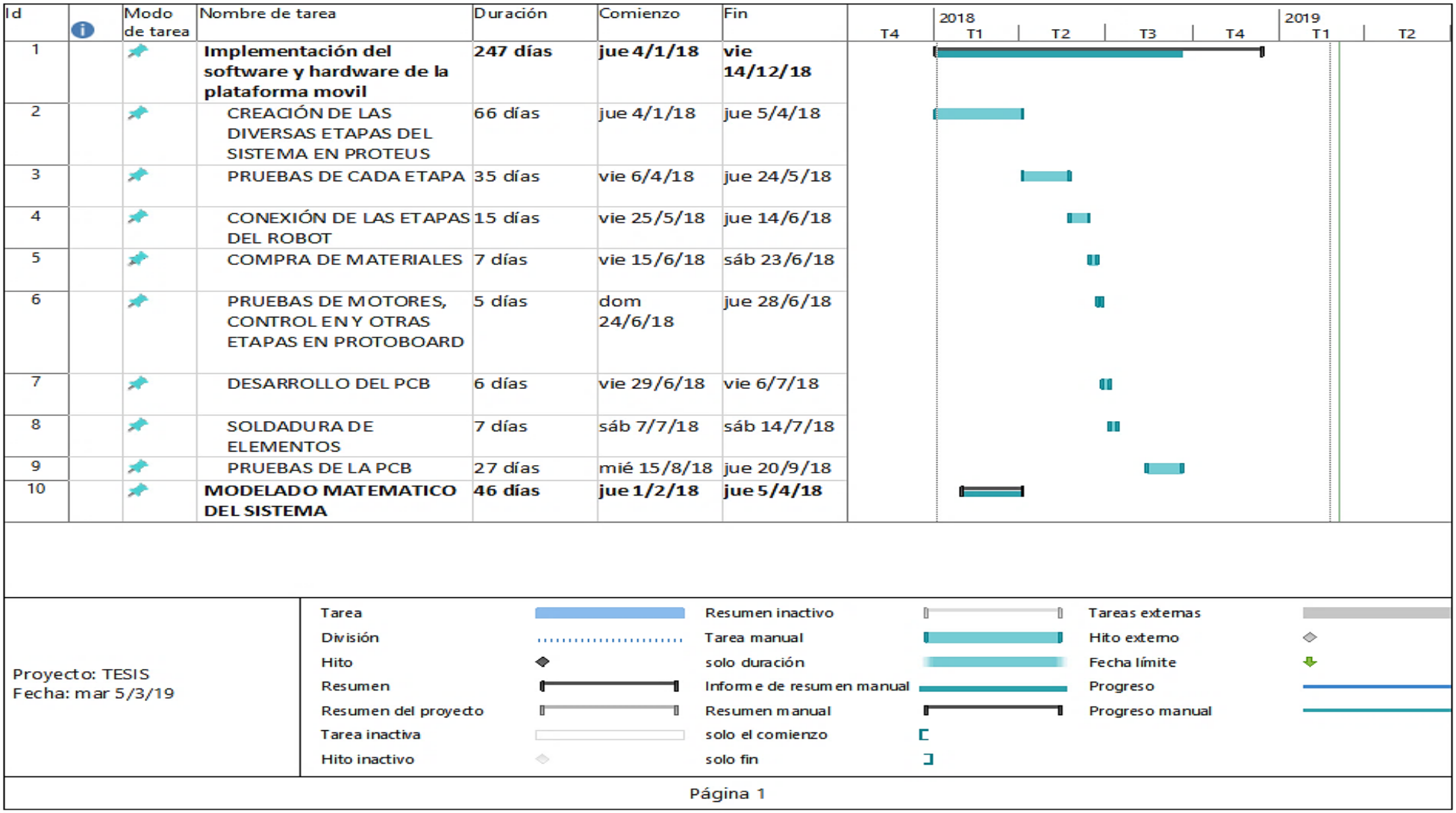
3.13 Cronograma de actividades

En el análisis de tiempo se realiza una descripción del cronograma de actividades, realizado durante las fases que comprenden el desarrollo completo del proyecto, el cual se lo realizó en un total de 247 días de donde se componen varias etapas, en la primera se define parte del hardware de la plataforma móvil diferencial, los elementos electrónicos que lo componen, tarjeta de control, para ello se buscó en el mercado cuales son los dispositivos electrónicos que cumplen con las especificaciones técnicas, para luego diseñar las diversos módulos que permitirán cumplir con los objetivos específicos, y que permitirán la elaboración del robot. Esto tomó alrededor de 196 días, considerando todo el tiempo que se tomó en realizar todo el diseño, esto empezó del 04 de enero al 20 de septiembre del 2018 como se aprecia en la Figura 3.5.

La siguiente etapa está dedicada al desarrollo matemático del algoritmo de control, la cual se ha separado en dos subtemas, el modelado matemático del sistema y el diseño del algoritmo de control. Esto debido a que se realiza un estudio del modelado con las ecuaciones de Euler-Lagrange. Su tiempo es de 104 días, desde el 01 de febrero al 22 de junio del 2018 como se observa en la Figura 3.6.

La elaboración de la carcasa del robot se lo realizó con el software 3D Studio MAX en el que plasma los diseños de la estructura del robot, para ser impresa en 3D con material PLA, la implementación de la PCB con diseño para los motores, y módulos que la componen se lo elaboró en el programa PROTEUS. Su tiempo de ejecución fue de 15 días, mostrada en la Figura 3.6.

Como etapas finales corresponden al desarrollo del software en el programa MATLAB en la ventana de desarrollo SIMULINK, desde el mismo se programó al microcontrolador, para realizar las pruebas de validación y funcionamiento con la elaboración del trabajo escrito que concluyo hasta el 20 de diciembre del 2018 según la Figura 3.7.



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		**Implementación del software y hardware de la plataforma movil**	**247 días**	**jue 4/1/18**	**vie 14/12/18**
2		CREACIÓN DE LAS DIVERSAS ETAPAS DEL SISTEMA EN PROTEUS	66 días	jue 4/1/18	jue 5/4/18
3		PRUEBAS DE CADA ETAPA	35 días	vie 6/4/18	jue 24/5/18
4		CONEXIÓN DE LAS ETAPAS DEL ROBOT	15 días	vie 25/5/18	jue 14/6/18
5		COMPRA DE MATERIALES	7 días	vie 15/6/18	sáb 23/6/18
6		PRUEBAS DE MOTORES, CONTROL EN Y OTRAS ETAPAS EN PROTOBOARD	5 días	dom 24/6/18	jue 28/6/18
7		DESARROLLO DEL PCB	6 días	vie 29/6/18	vie 6/7/18
8		SOLDADURA DE ELEMENTOS	7 días	sáb 7/7/18	sáb 14/7/18
9		PRUEBAS DE LA PCB	27 días	mié 15/8/18	jue 20/9/18
10		**MODELADO MATEMATICO DEL SISTEMA**	**46 días**	**jue 1/2/18**	**jue 5/4/18**

Figura. 3. 5 Cronograma de actividades

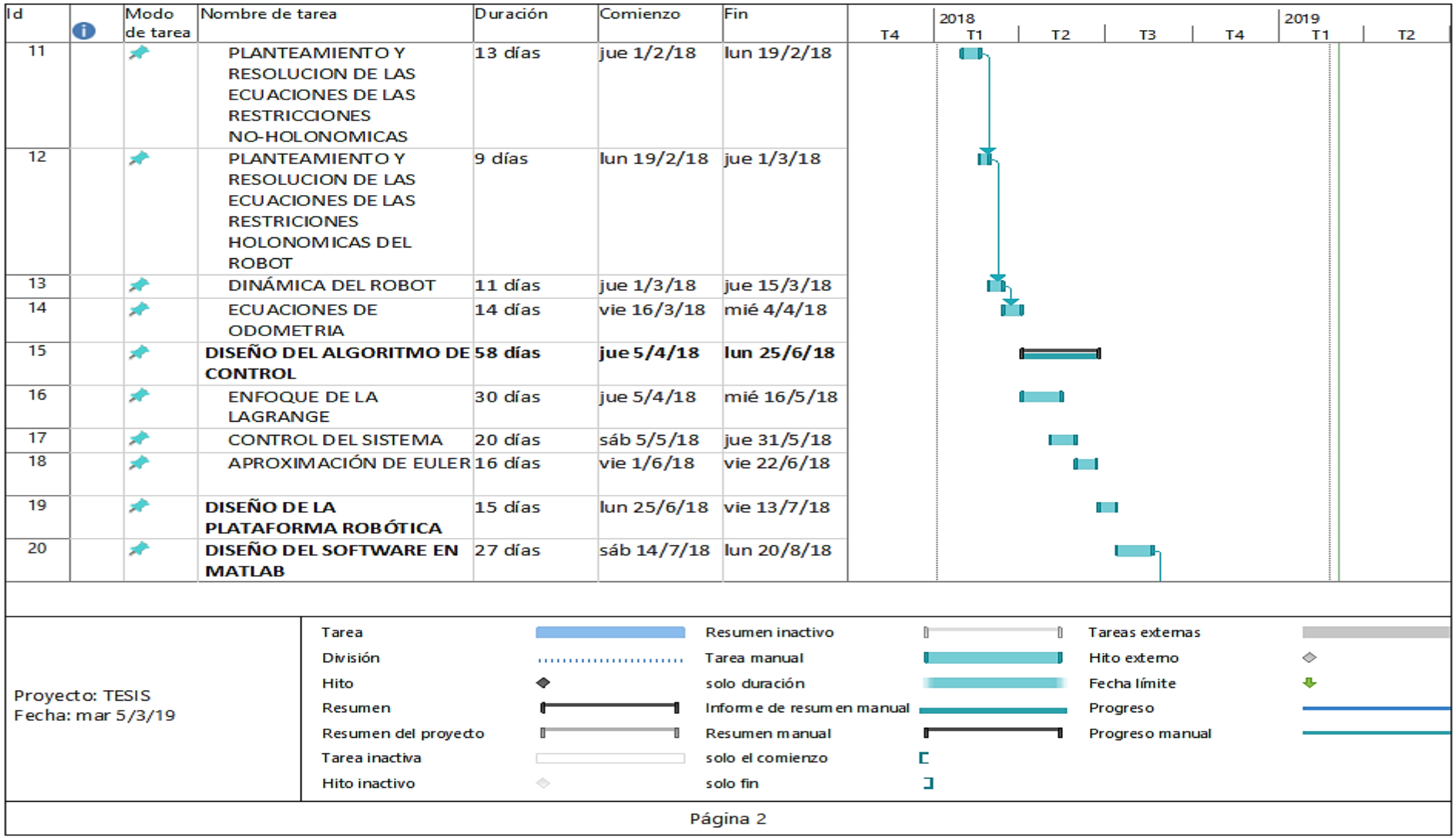
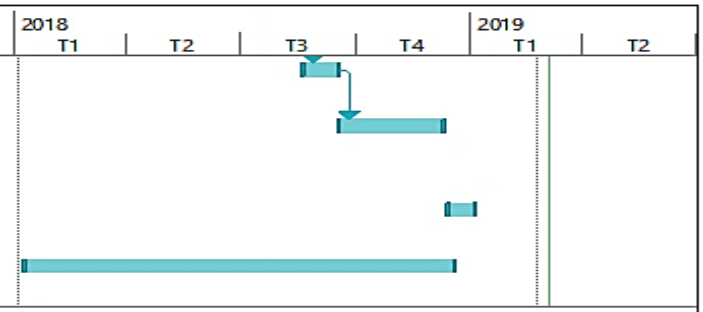


Figura. 3. 6 Cronograma de actividades

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
21		**PROGRAMACIÓN DEL MICROCONTROLADOR**	21 días	mar 21/8/18	mar 18/9/18
22		**CALIBRACIÓN DEL ALGORITMO Y PRUEBAS DE FUNCIONAMIENTO**	60 días	mié 19/9/18	mar 11/12/18
23		**DEPURACIÓN DE ERRORES**	17 días	sáb 15/12/18	sáb 5/1/19
24		**ELABORACIÓN DEL ESCRITO DEL PROYECTO**	247 días	mié 10/1/18	jue 20/12/18



Proyecto: TESIS
Fecha: mar 5/3/19

Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Progreso
Progreso manual

Página 3

Figura. 3. 7 Cronograma de actividades

CAPITULO IV

IMPLEMENTACIÓN

En este capítulo se detallará el desarrollo de la plataforma robótica móvil diferencial, su estructura, diagrama del circuito, programación del algoritmo en el programa MATLAB, desarrollo del GUI y pruebas de este.

4.1 Desarrollo de la plataforma robótica móvil

Se realizó el diseño de la placa del robot en el programa Proteus, como sistemas de control el microcontrolador STM32F407 como corazón de todo el circuito, el mismo que ejecutará todo el procesamiento de la información adjunta en el controlador.

El desarrollo de todo el hardware se lo puede dividir en 4 etapas para su analisis, las mismas que se detallan a continuación.

4.1.1 Etapa de comunicación

El robot maneja dos tipos de comunicaciones, serial e inalámbrica, se segmentó de esta manera por motivos que se detallan a continuación:

Para la programación se realizaró mediante conexión USB y para la recepción de datos se lo hace mediante la comunicación Bluetooth.

- **Programación de la plataforma robótica**

El envio de datos desde la PC con el programa MATLAB hacia el robot será mediante comunicación USB, por un puerto de comunicación USB (PC) y mini USB (Robot), la información de la programación es convertida a lineas de asembler, lenguaje del programador mediante el software ST-Link por medio del mismo se ancla a Matlab a travez de las librerias Waijung Blockset y se envian al microcontrolador STM32F103 el cual recepta los comandos del software y los transfiere a la memoria flash del microcontrolador STM32F104 mediante la comunicación SPI.

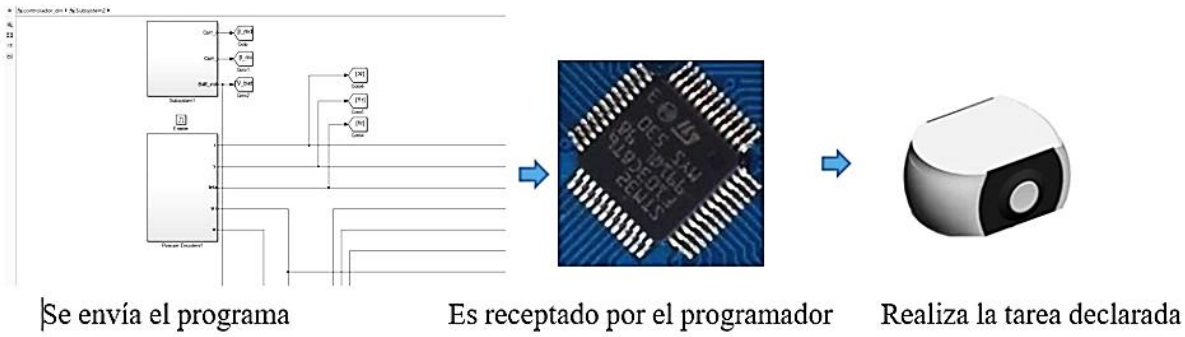


Figura. 4. 1 Esquema de programación

El envio de datos se realiza mediante el microcontrolador de la tarjeta ST-link V2 la cual recomienda el fabricante y entrega un diagrama del sistema en su tarjeta de desarrollo Discovery, el programa del microcontrolador se encuentra con un firmware no editable en la página oficial de STM electronics y este código es cerrado, sin embargo el mismo no requiere edición ya que se encuentran todas las herramientas necesarias para programar y hacer un debug del microcontrolador.

El esquemático desarollado se basa en la versión más actual proporcionada por el fabricante de la tarjeta discovery y se presenta a continuación.

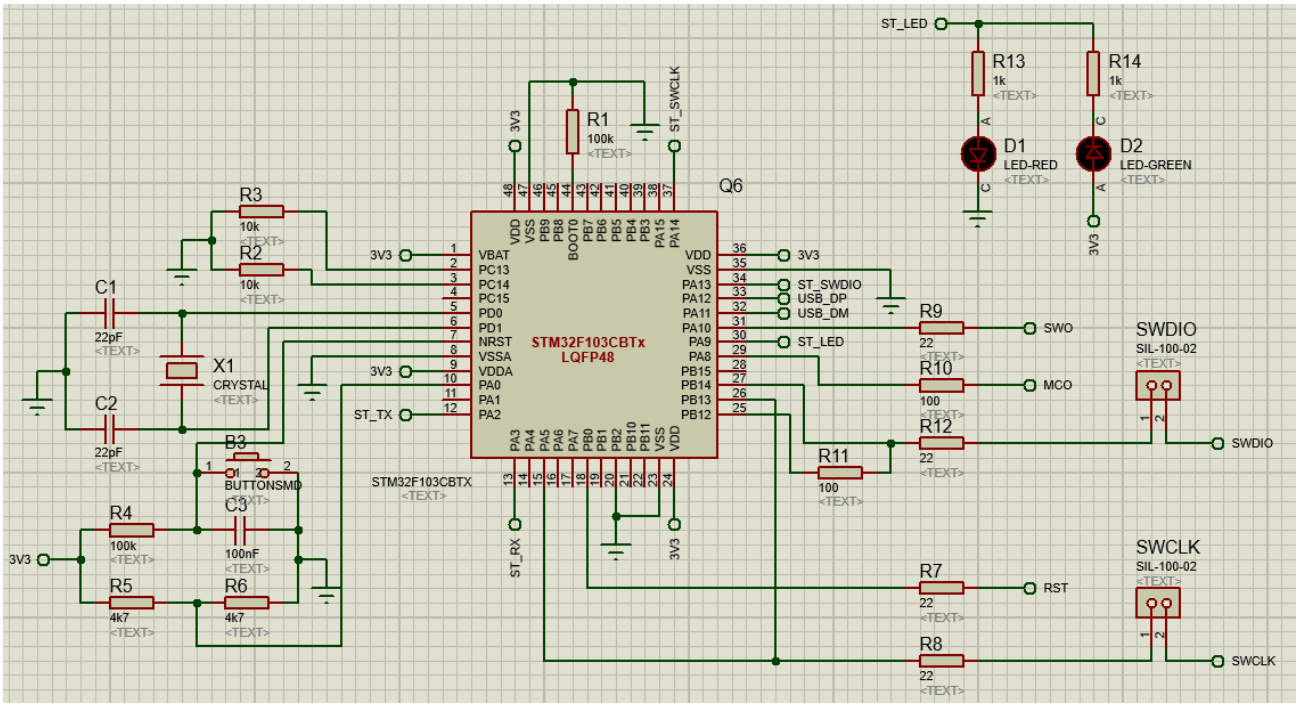


Figura. 4. 2 Circuito de la etapa de programación

Los Jumpers SWDIO y SWCLK permiten el desacoplamiento de la tarjeta de programación externa. El microcontrolador también tiene salidas de los puertos de programación que permiten descargar el firmware del programa.

- **Comunicación Bluetooth**

Se utiliza el módulo Bluetooth para el envio de información, procesada en el microcontrolador como posición actual, odométria, posición deseada en el tiempo, velocidad angular de los motores para su analisis en el GUI, esta tranmisión es en tiempo real para proyectar una gráfica posición versus tiempo.

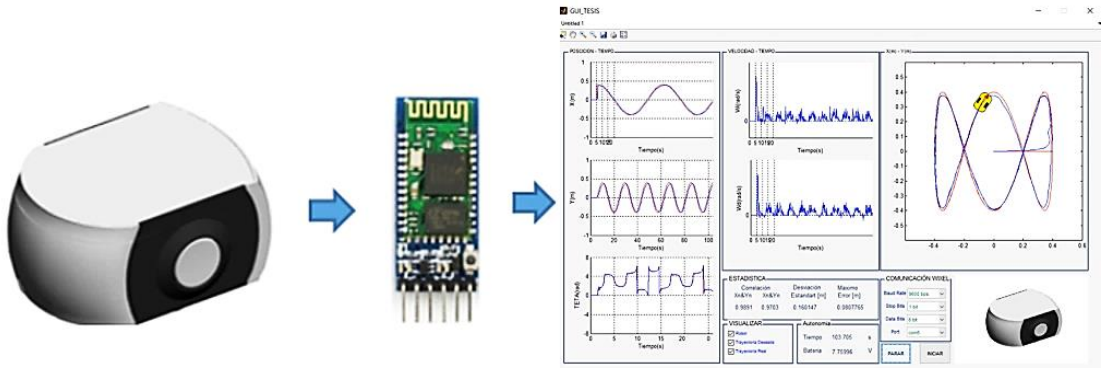


Figura. 4. 3 Esquema de envío de información del robot

La comunicación utilizada entre el microcontrolador y el módulo Bluetooth es la USART, para la comunicación con la PC se utiliza un módulo Bluetooth externo con la configuración *master/slave*, despues de receptar los datos son convertidos a comunicación USB mediante un puerto COM y son los que procesa el GUI.

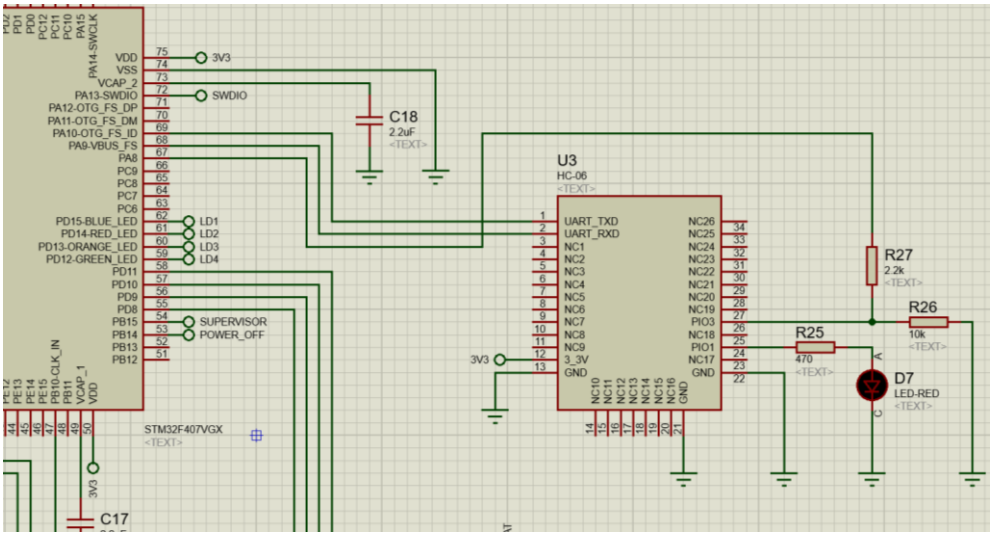


Figura. 4. 4 Diagrama de conexión en proteus

La comunicación con el Bluetooth se realiza mediante el módulo USART del microcontrolador a través de los pines PA9-PA10, y un pin digital PA8 el cual activa la entrada *enable* del módulo permite colocar la unidad en STANDBY para bajo consumo, el divisor de tensión se debe ocupar ya que la salida digital es de colector abierto.

4.1.2 Etapa de alimentación

Como fuente de energía se trabajará con una batería LIPO 2S de 7.4V que debe ser acondicionada, para ello se ocupan reguladores de voltaje los cuales proprocionan la tensión adecuado para cada módulo.

El AMS 1117 se utiliza para alimentar los circuitos de 5V, como la etapa de control del driver de los motores, el módulo Bluetooth encargado del envio de información, y la placa de los sensores inerciales, asi como para alimentar la etapa de regulación a 3.3V la cual suministra corriente a los microcontroladores.

Los capacitores C5-C9 y C11 son capacitores no polarizados para filtrar ruidos de altas frecuencias o provocados por el módulo de comunicación inalambrica y las lineas de conmutación PWM entre otros mientras que C4 y C10 son filtros de ruido de bajas frecuencia provocados por los motores al arrancar.

Los capacitores C13 y C14 cumplen la misma función sin embargo esta etapa no requiere filtrar los armonicos de la red ya que los elementos que alimenta la etapa de 5V no tiene demasiada sensibilidad al cambio de frecuencia, a diferencia de los microcontroladores debido a la frecuencia alta de trabajo que poseen.

Los valores tomados de los capacitores para la entrada al microcontrolador son datos obtenidos del Datasheet, valores que recomienda el fabricante para que proteja el circuito, de las frecuencias antes mencionadas. Adicional se ha colocado un led indicador, que permite visualizar la alimentación de corriente al microcontrolador. Con la consideración que son pocos miliamperios que circulará por el diodo emisor de luz para lo cual por experimentación se coloca una resistencia de 1kΩ.

Al sistema se le considero acoplar un pequeño touch para el encendido y apagado del robot, pero en las pruebas realizadas no funcionó correctamente, ya que al momento de contacto la plataforma debía encenderse, pero la misma después de unos segundos terminaba apagándose.

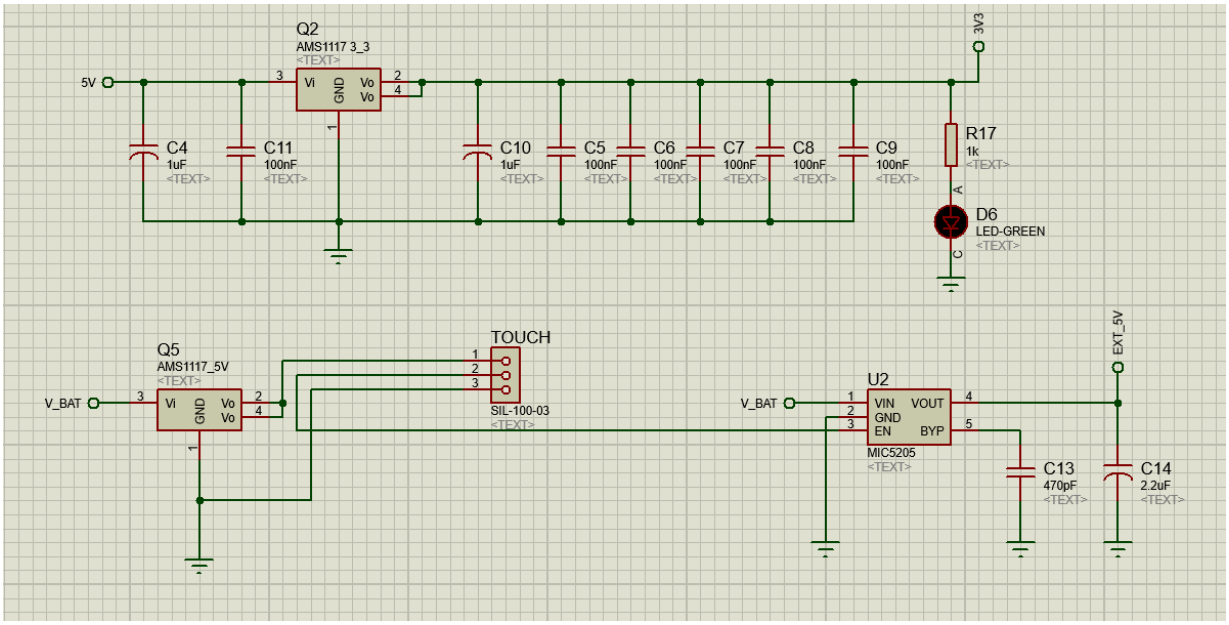


Figura. 4. 5 Diagrama de conexión eta de alimentación

4.1.3 Módulo de potencia

Se utilizará los motores *Metal Gear Motor* con eje extendido que permite conectar con mayor facilidad los encoders, tiene a su vez una caja reductora 30:1 acoplada que permite mayor potencia de tracción, a pesar de que el proyecto no se enfoca en que la plataforma pueda desplazarse por terrenos escabrosos. Maneja una velocidad de 1100 RPM con un consumo nominal de 100mA, se llegará a tener un consumo máximo de 1.5 A esto debido a que el rotor se bloquee.

Para el control de estos motores se utilizará el driver L298P el cual es un puente H doble, y es encargado de amplificar la señal de corriente de la línea de control para proporcional el accionamiento de los motores, a las líneas de entrada *enable*, se las alimenta mediante señales PWM de los pines PA5 y PB10 y las líneas de accionamiento del puente se las alimenta con las líneas digitales de los pines PA8-PA11 de esta forma solo se requieren dos señales PWM que serán enviadas por el microcontrolador.

El driver requiere los diodos antiparalelo-externos los cuales cierran el paso de las corrientes de retorno producidas en la inductancia del motor y los capacitores C31-C32 permiten eliminar los transitorios de voltaje producto de la conmutación, entre el cambio de operación de un motor a otro.

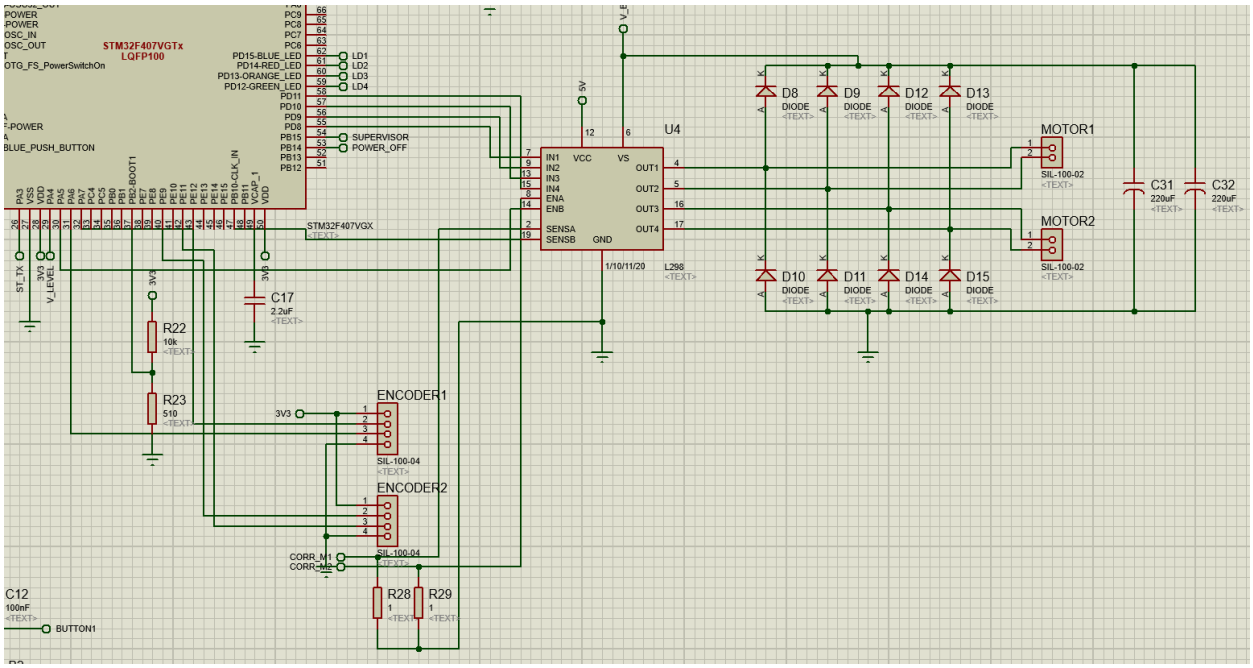


Figura. 4. 6 Diagrama de la etapa de potencia

El driver L298P fue seleccionado ya que permite manejar corrientes de 4 amperios, dado que la corriente máxima cuando el rotor está bloqueado es de 1.5 A la peor condición en la cual se envía al motor una señal de voltaje con giro en reversa provocará una corriente de 3A por lo que es necesario un driver de 3.75 A o superior por canal.

4.1.4 Módulo de control

Todo el sistema será controlado mediante el microcontrolador STM32F407 de 32 bits que será el encargado de realizar todo el control del robot, donde se ejecutará el programa del algoritmo de seguimiento de trayectoria, el microcontrolador por su gran cantidad de pines el modelo es en *SMD*, como se puede observar en la figura 4.8 no se utilizarán todos los pines del microcontrolador, solo los necesarios, que me permitan el manejo de los motores, recepción de datos desde los encoder´s como giróscopo, alimentación, programación, ect.

El pin Vbat permite la alimentación ya sea de una batería o de un supercapacitor o en su defecto como recomienda el fabricante un grupo de capacitores que sumados den una capacitancia suficiente para mantener la información del reloj de tiempo real del microcontrolador en caso de pérdida de alimentación.

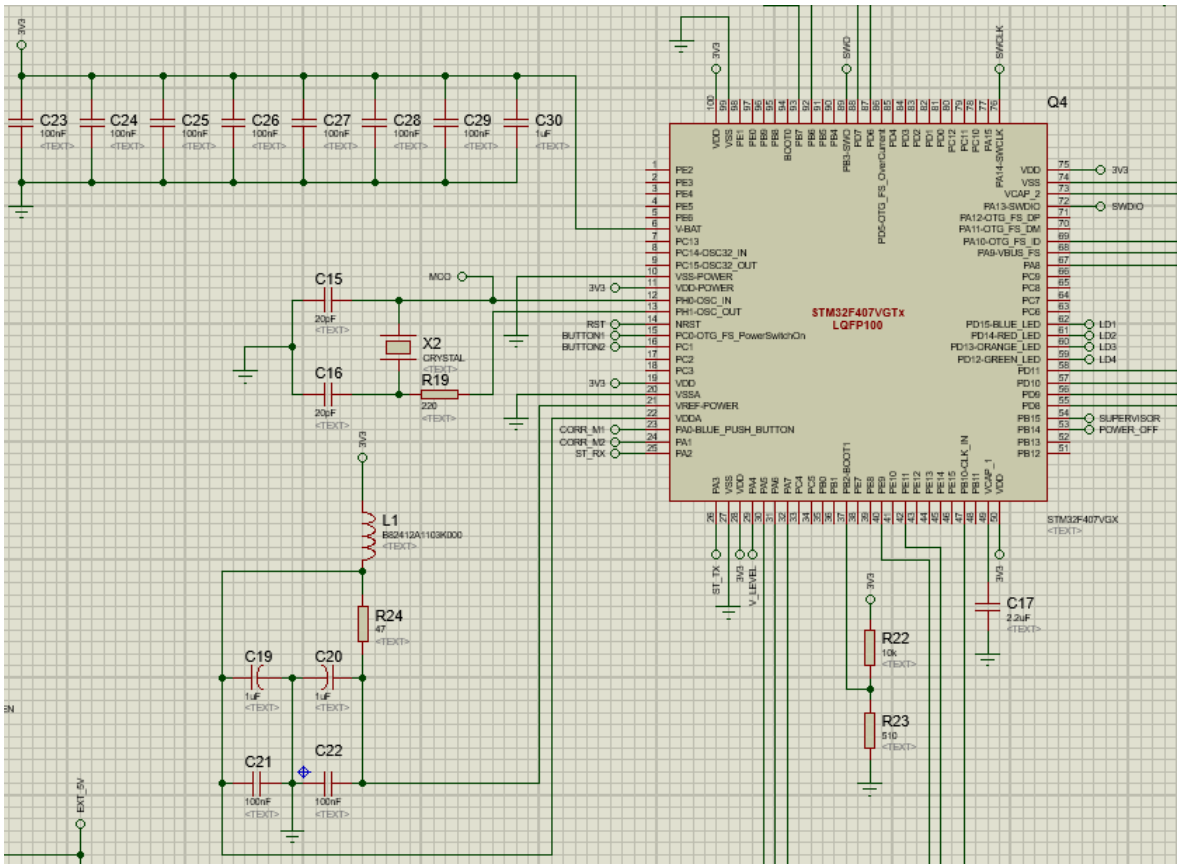


Figura. 4. 7 Diagrama de conexión del STM32F407

Los pines *VSSA y VREF-POWER* son pines de entrada de alimentación y referencia del conversor analógico para esto el fabricante recomienda un circuito de filtrado para eliminar corrientes transitorias que puedan provocar fallos en la operación de la *ALU* de alta frecuencia.

El PIN BOOT1 conectado al divisor de tensión permite la activación del modo de programación mediante la sección de código *BOOT-LOADER* espacio de memoria que puede ser editado para programar el microcontrolador desde cualquier recurso del mismo, este método de programación no se ha usado por lo que este pin debe permanecer con un voltaje menor a 0.18V por lo que el divisor de tensión dado por la siguiente ecuación mantendrá esta salida siempre en bajo.

$$3.3 * \left(\frac{510}{10k + 510}\right) = 0.16V$$

La salida VCAP1 y VCAP2 son salidas diseñadas por el fabricante para la conexión de capacitores externos para filtrado de señales que por recomendación deben ser de 2.2uF.

4.2 Comunicación IMU

La comunicación con la *IMU* (Unidad de medición inercial) o sensor inercial se realiza mediante el protocolo de comunicación *I2C* a través de los pines PB6-PB7 con las resistencias *PULL-UP* requeridas por la normativa en el valor recomendado de 4.7 kΩ.

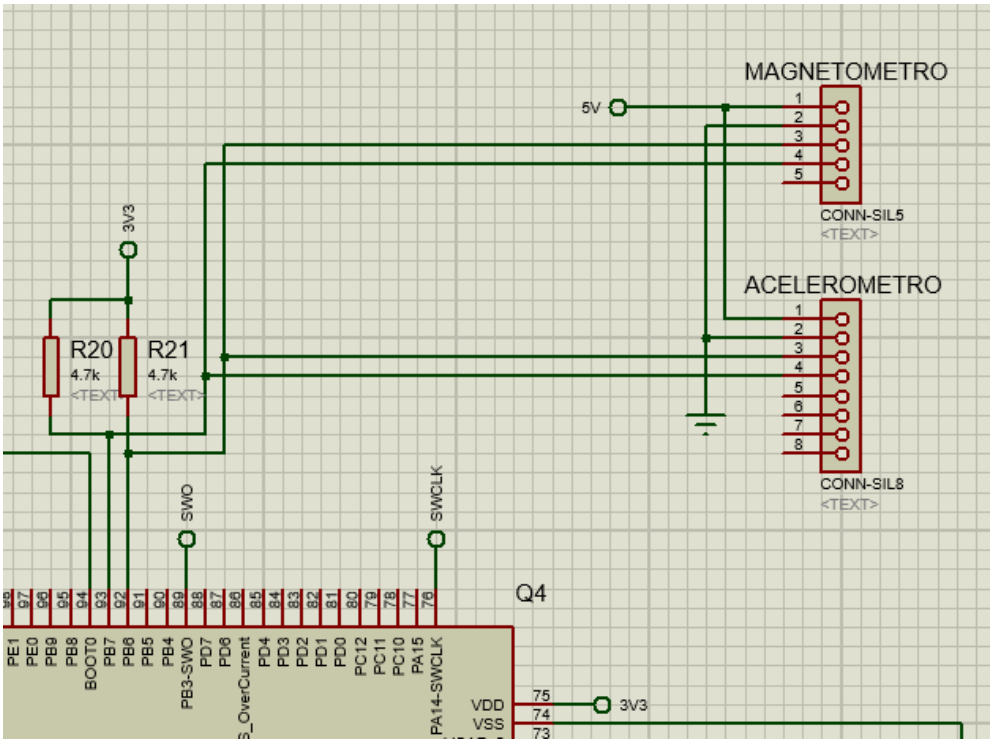


Figura. 4. 8 Diagrama de conexión de los sensores inerciales

4.3 Análisis de carga

En el análisis de carga se analizará los elementos que consumen más potencia cuando estos trabajan a su máxima carga, se representará en una tabla de valores:

ELEMENTO	VOLTAJE	CORRIENTE	CANTIDAD	POTENCIA
STM32F407	3.3 V	59 mA	1	0.19 W
STM32F103	3.3 V	20 mA	1	0,066 W
MPU9150	5 V	11.7 mA	1	0,058 W
POLOLU	6 V	427 mA	2	2.4 W
HC-06	5 V	48 mA	1	0,24 W
			TOTAL	**2.95 W**

$$Pt = 2.95\ W * 1.5$$

$$Pt = 4.425\ W$$

La plataforma robótica diseñada será exclusivamente para pruebas del algoritmo, por lo cual se analizará y validará el seguimiento de trayectoria, se promedia alrededor de 40 min. Por consiguiente, se instaló una batería LIPO de 7.4 V – 2200 mA.

$$\Delta q = mAh * V/Vm\grave{a}x$$

$$\Delta q = 2200mAh * \frac{(7.6 - 6.8)V}{7.6V}$$

$$\Delta q = 289.47\ mAh \text{ W}$$

Se considera la carga total consumida por toda la plataforma:

$$Iin = \frac{2.95\ mA}{7.1}$$

$$Iin = 415\ mA$$

Los minutos de trabajo en operación continua será de:

$$t = \frac{289.47\ mAh}{415\ mA} * 60$$

$$\boldsymbol{t \approx 40\ min}$$

El tiempo resultante es de 40 min de autonomía aproximadamente, esto considerando que el robot trabajara a una carga constante, que los rotores no variaran su velocidad muy bruscamente, será casi constante a una velocidad determinada.

4.3. Censado de la batería

Uno de las protecciones necesarias cuando se ocupa una batería de litio polímero es el muestreo del voltaje de la batería ya que si su nivel baja del límite establecido por fabrica que en el caso de la batería *TENERGY* es de 6.6V la batería no vuelve a cargar y será inservible y si se intenta realizar la carga por fuerza puede explotar y provocar daños materiales y humanos debiéndose entonces considerar una entrada analógica del voltaje de esta batería para lo cual se diseña de forma que el voltaje máximo de entrada analógica de referencia sea de 3.3V, y el voltaje de entrada de la batería por seguridad se toma un valor de 10V con lo cual el divisor de tensión se calcula a continuación.

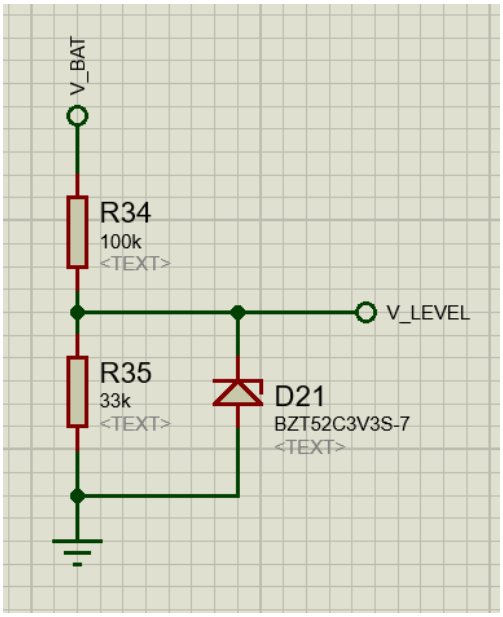


Figura. 4. 9 Diagrama de divisor de tensión para censado del voltaje de batería

$$3.3V = 10 * \left(\frac{R35}{R34 + R35}\right)$$

$$3.3V * (R34 + R35) = 10V * R35$$

La corriente de consumo del ramal no debe superar los 100 uA para que la batería no sufra un consumo por medida.

$$100uA = 10V * \left(\frac{1}{R34 + R35}\right)$$

$$100K = (R34 + R35)$$

Con lo que se tienen dos ecuaciones y dos incógnitas, se despeja los valores de R34 y R35.

$$3.3V * (R34 + R35) = 10V * R35$$

$$3.3V * 100K = 10V * R35$$

$$R35 = 33K$$

$$R34 = 100K$$

El diodo zéner se usa en caso de que ingrese un voltaje superior al pin y pueda dañarlo por lo que se debe limitar el voltaje con el zéner de 3.3V, el pin analógico del micro que recibe esta tensión es el PA4.

4.4. Entradas digitales de usuario

Se añadieron las entradas digitales de usuario para cualquier proceso que requiera la intervención humana, estas entradas se usaron en el proceso de calibración y prueba de hardware, sin embargo, para la presentación final del robot ya no son necesarias, ya que todo el algoritmo es automático y no requiere intervención alguna por parte del usuario a través de estas entradas.

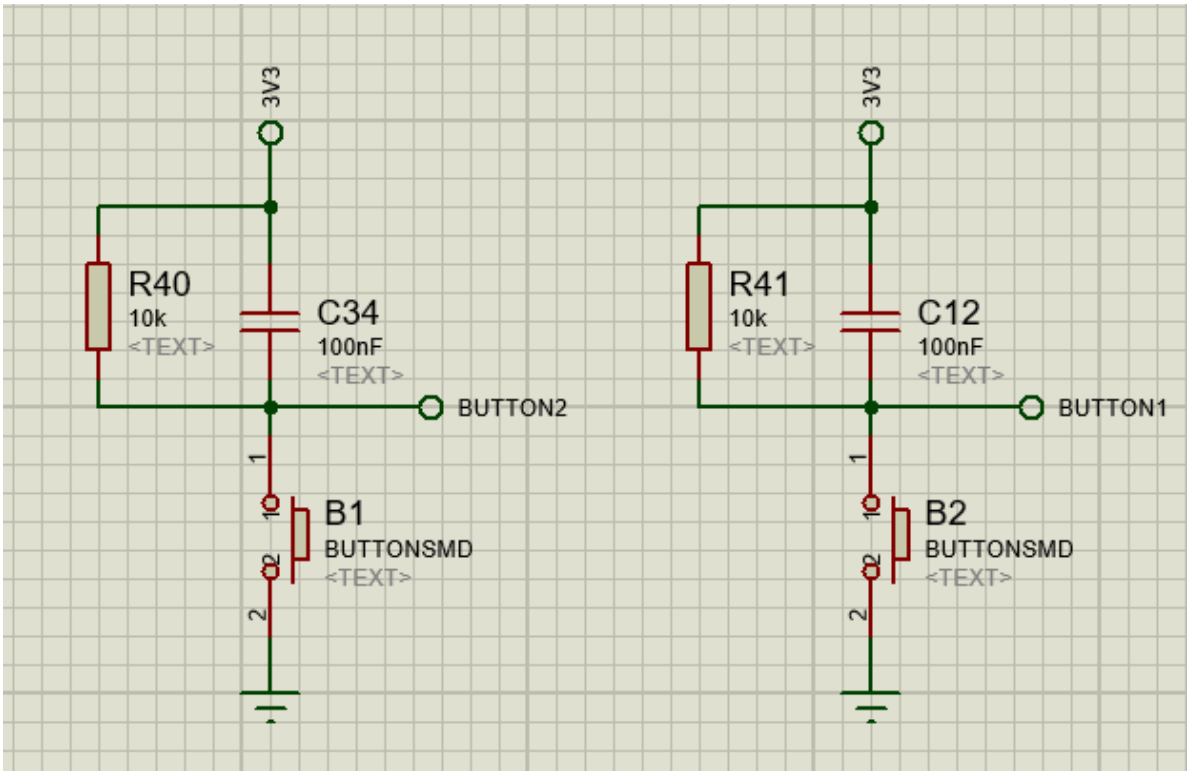


Figura. 4. 10 Diagrama de entradas digitales por pulsador

Los capacitores C34 C12 son capacitores antirebote, ya que la señal mecánica de los pulsadores tiene rebotes de voltaje de duraciones de alrededor de 100 milisegundos sin embargo el microcontrolador puede leer a una velocidad de 170 MHz y provocar lecturas de varias pulsaciones, entonces se debe eliminar esos pulsos con filtros capacitivos.

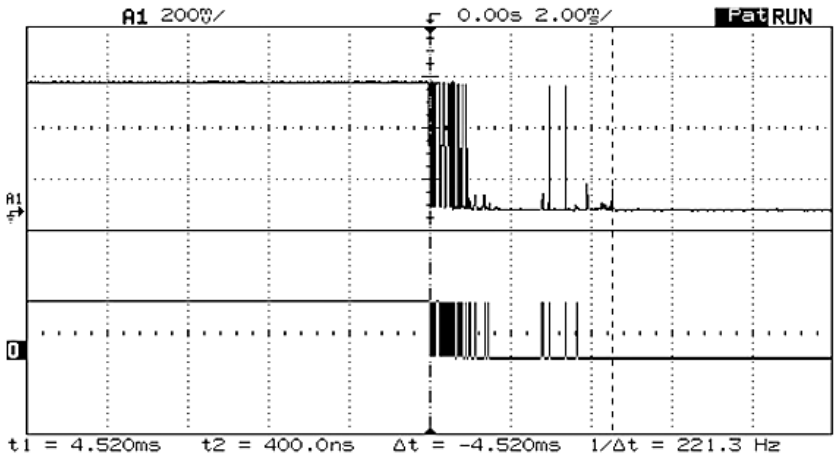


Figura. 4. 11 Señal de osciloscopio de los rebotes de voltaje del pulsador (Wordpress, 2017)

4.5 Entrada USB

La entrada USB se realiza por el puerto mini USB de 5 pines más la carcasa, la cual requiere para el puerto data más una resistencia *pull-up* ya que la salida en este pin es a colector abierto.

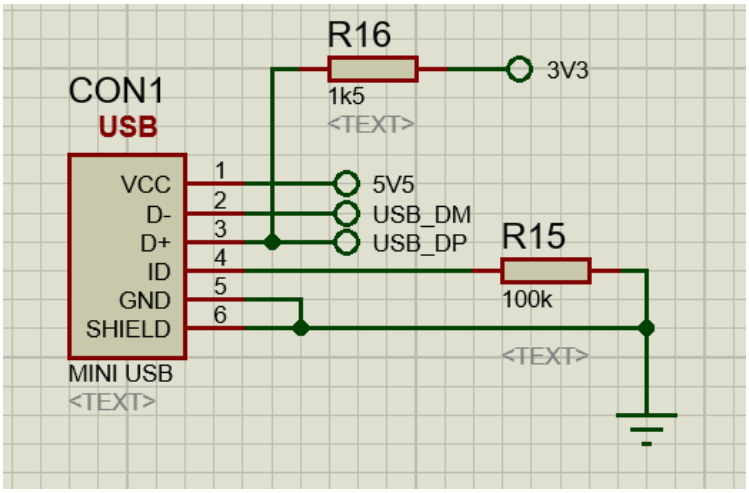


Figura. 4. 12 Diagrama de conexión del USB

4.6 Diseño del PCB

Para la realización de la placa electrónica PCB se ocupó las herramientas del software *Proteus* como se pude observar en la figura 4.12 donde están todas las conexiones de los elementos, para facilidad de conexión y desconexión de algunos elementos se colocaron sockets, la mayoría de elementos son en *SMD* para que toda la estructura sea liviana y a su

vez no ocupe mucho espacio, su diseño se obtiene del corte del plano con las tapas frontal del robot en el diseño 3D para que la misma se adapte a la superficie y pueda sobresalir el pin de conexión *USB*, toda la placa se diseñó con malla a tierra para evitar ruido capacitivo entre las pistas y evitar que las emisiones electromagnéticas del medio induzcan voltajes en las pistas del *PCB (palca de circuito impreso)*, la placa presenta dos capas en las caras *Botton* y *Top* ya que la densidad de conexión no se puede obtener en una sola capa.

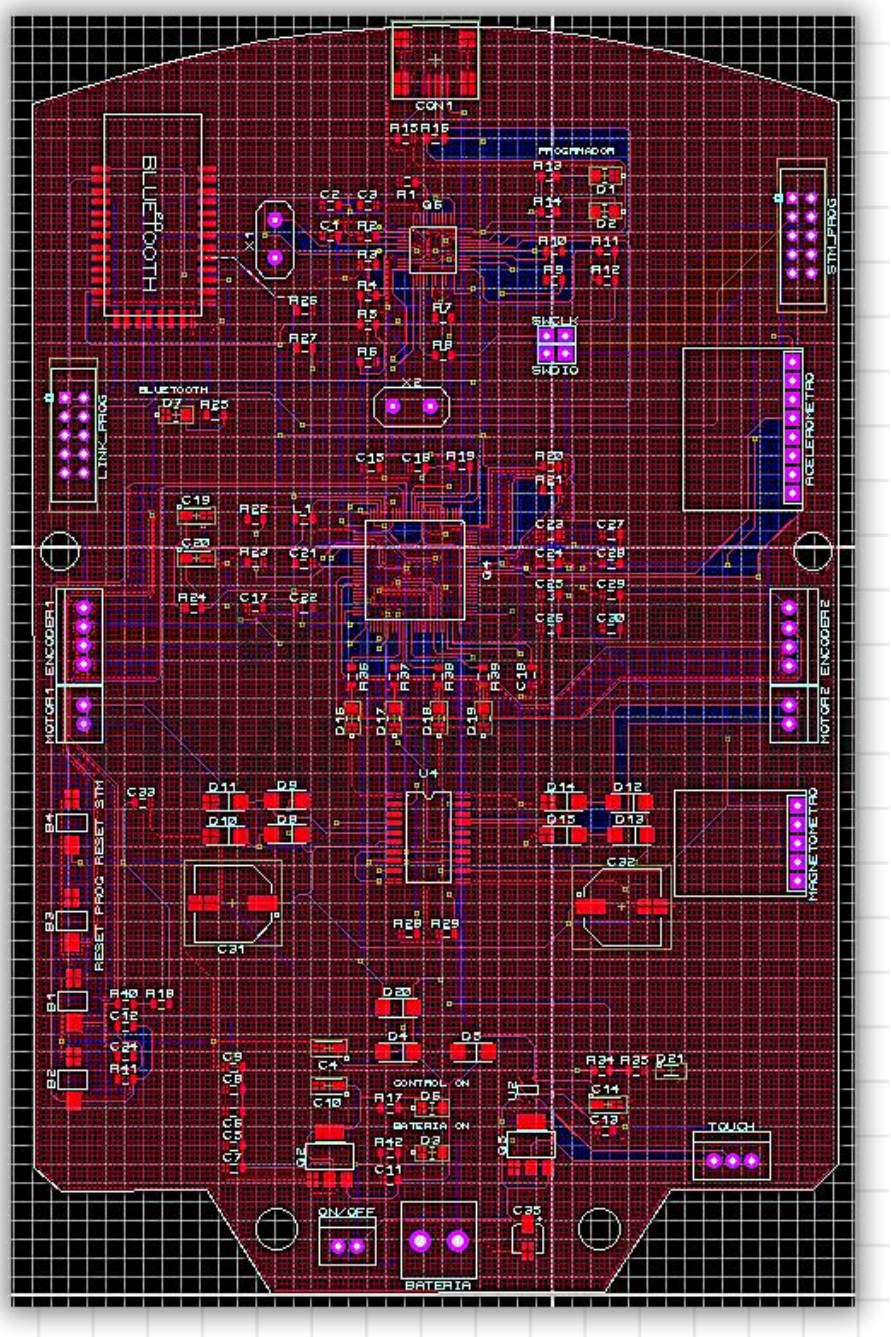


Figura. 4. 13 Diagrama PCB de la placa

4.7 Diseño de la estructura de la carcasa del robot

El diseño de la estructura del robot se realizó en el software *3D Studio MAX* perteneciente a *autodesk* y permite diseñar piezas con precisión para su renderización y animación y se escogió el software por afinidad.

La estructura del robot se la diseñó externamente basada en el modelo del robot industrial *OMROM LD* y su diseño interno es aporte del autor para mantener el centro de masa lo más cerca del centro geométrico así como una distribución práctica que permita anclar la placa a los motores y almacenar la batería, el diseño está realizado en piezas de anclaje para su montaje sencillo y facilitar la impresión en 3D la sujeción entre piezas se realiza mediante pernos y todo el robot está cerrado esto evita los daños en la estructura mecánica y electrónica al manipular el mismo.

La estructura se imprimió en 3D ya que es el único método de creación de piezas plásticas polimórficas de alta complejidad. El material en el que se imprime la carcasa es plástico termo deformable, cuyo peso es bajo en comparación a estructuras metálicas.

A continuación, se presentan una vista renderizada de las piezas diseñadas del robot con el terminado final.

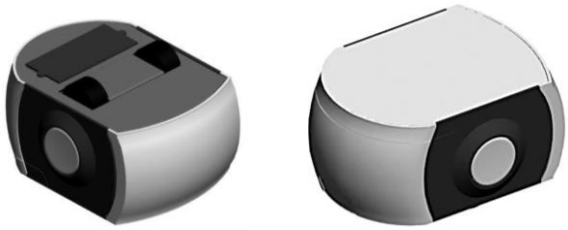


Figura. 4. 14 Armado de la estructura

Se detalla el desmontaje de las piezas que componen la carcasa del robot.

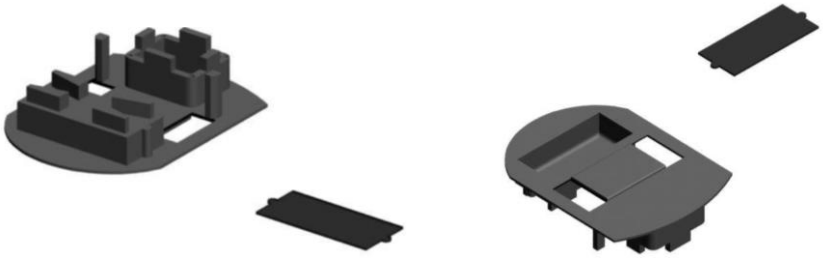


Figura. 4. 15 Base de la carcasa

En la figura 4.11 se puede observar toda la estructura base donde se colocará la placa (PCB), los motores, encoder´s y batería.

Figura. 4. 16 Escudos laterales

Las tapas laterales tanto izquierda como derecha tienen un agujero central, para temas estéticos.

Las tapas frontal y posterior tendrán una estructura curva por fines estéticos, para que el robot no tenga una forma cuadrada si no un poco redondeada.

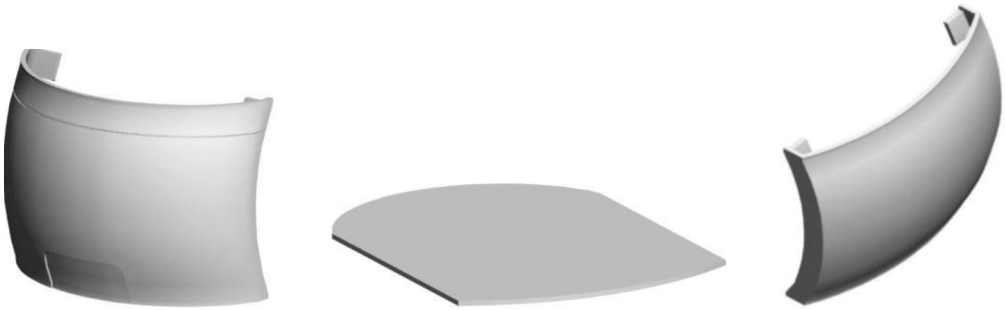


Figura. 4. 17 Tapa frontal, posterior y superior

Se diseñaron unas pequeñas tapas laterales que tendrán otro color para resaltar el diseño, y entrarán en los agujeros dejados en las tapas laterales.

Figura. 4. 18 Tapas pequeñas

Se adjunta todo el acotamiento de las piezas que componen la carcasa del robot

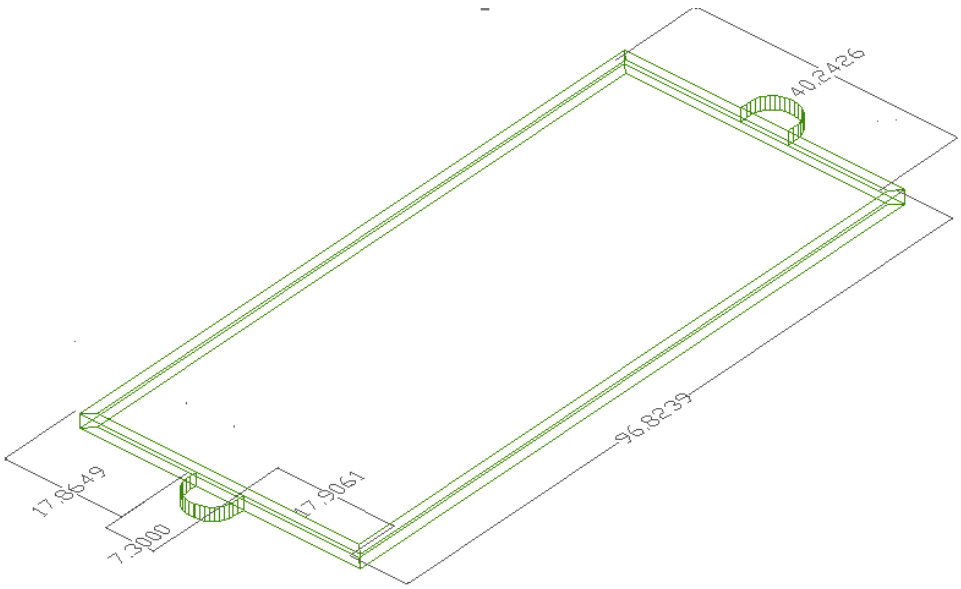


Figura. 4. 19 Cotas tapa de la ranura de la batería

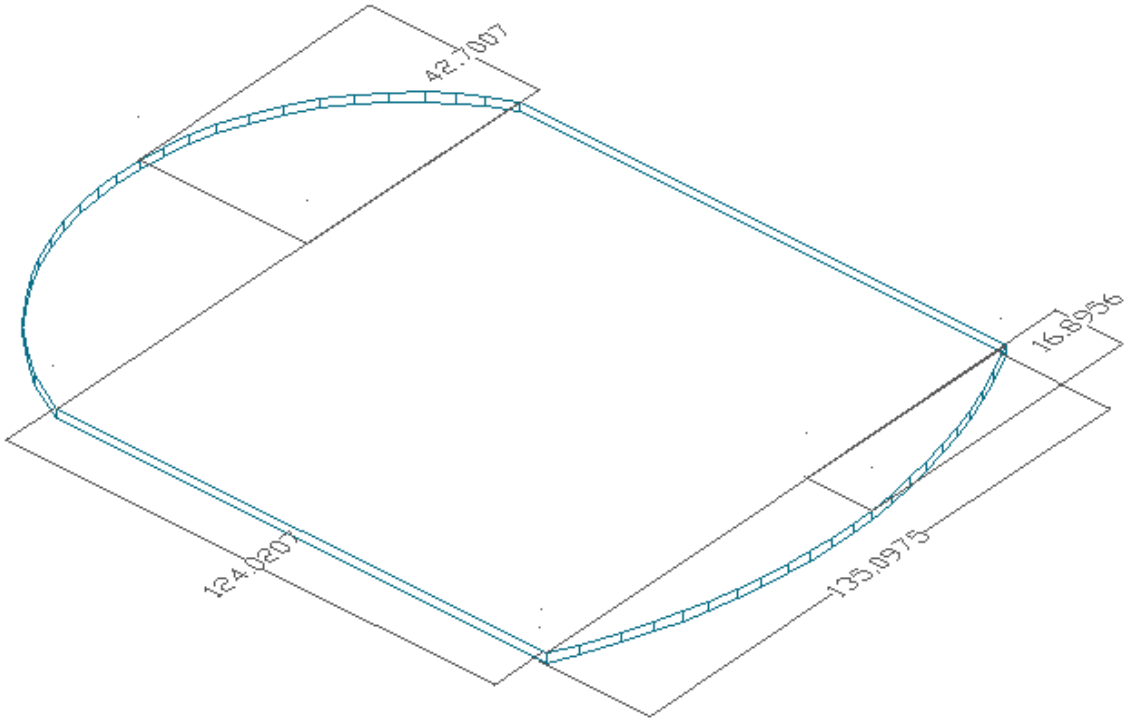


Figura. 4. 20 Cotas tapa superior

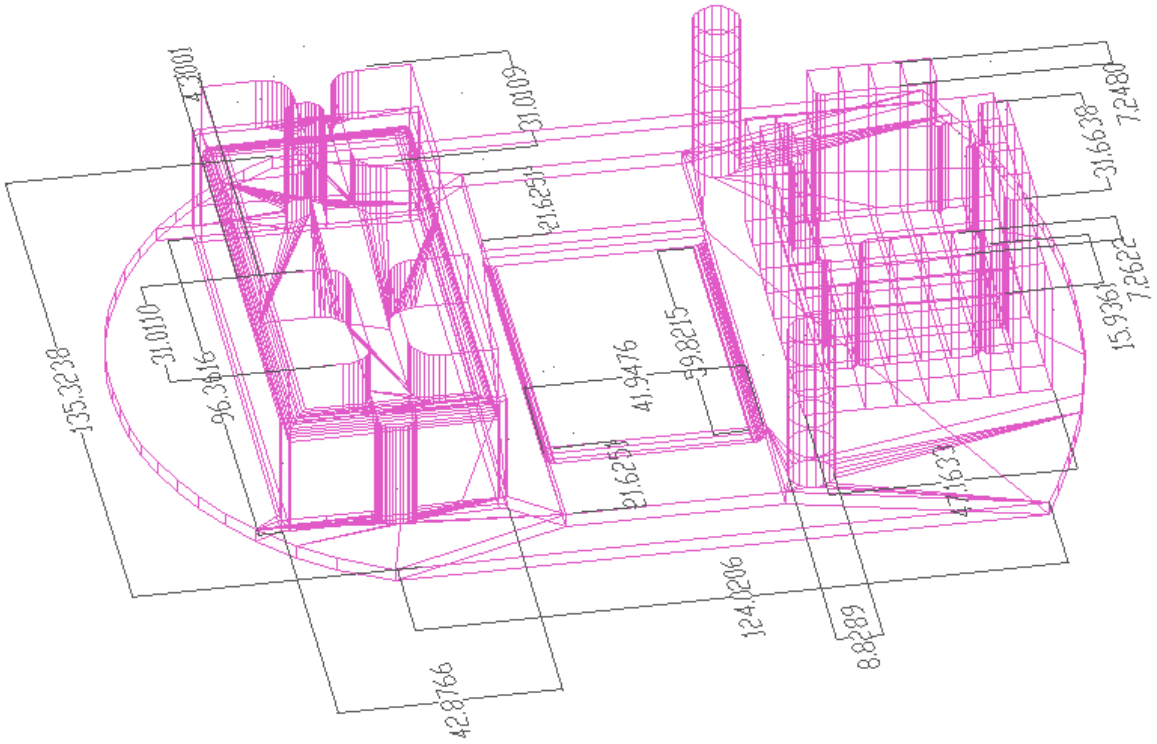


Figura. 4. 21 Dimensiones base de la estructura

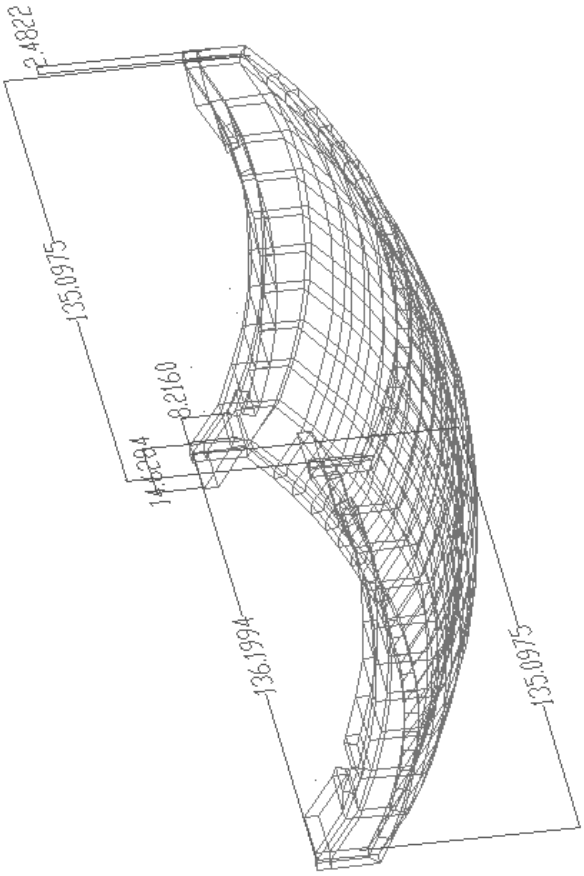


Figura. 4. 22 Dimensiones tapa posterior

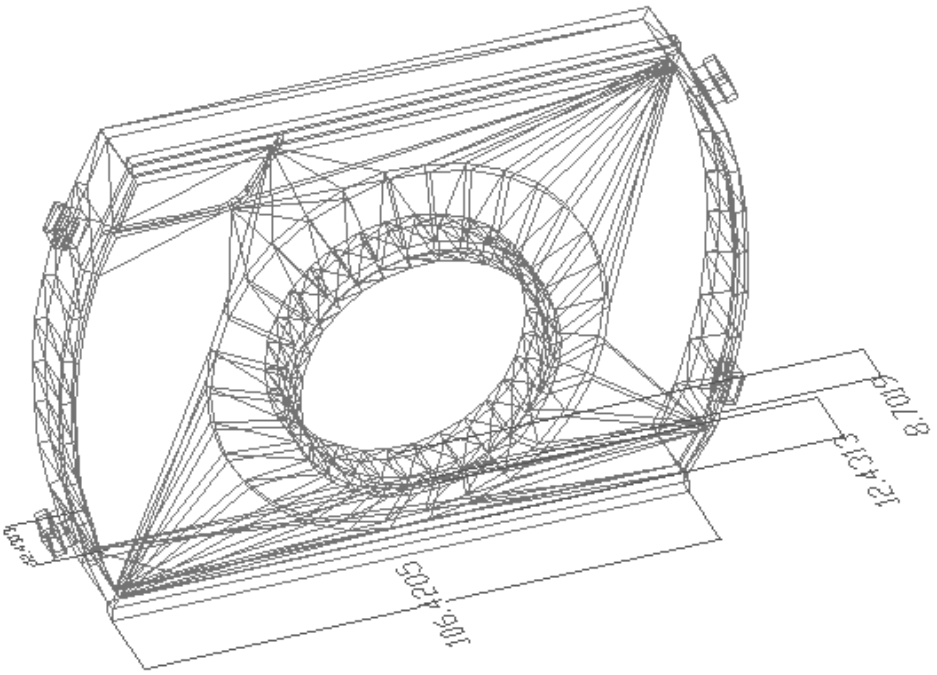


Figura. 4. 23 Cotas tapa lateral

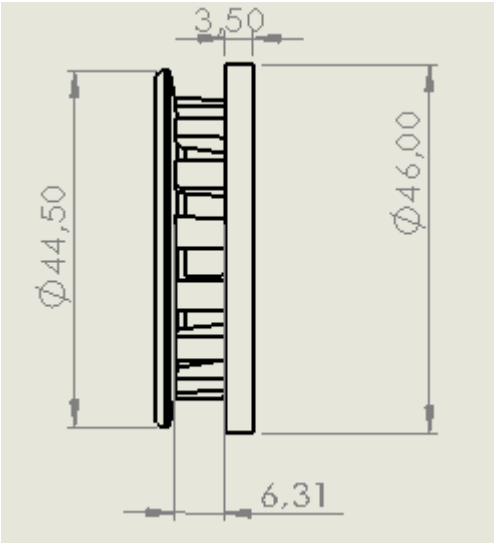


Figura. 4. 24 Cotas tapas laterales

4.8 Modelado matemático del robot

4.8.1 Eje de coordenadas

Para describir la posición del robot se maneja dos sistemas de coordenadas referenciales:

1. Sistema de coordenada inercial el cual es el entorno donde se desplaza el robot y se denomina inercial dado que para los tiempos de movimiento del robot el sistema tierra puede ser aproximado a un sistema de velocidad constante.

2. Sistema de coordenadas del robot, tiene su eje en el centro de masa del mismo por lo cual el eje se desplaza con el robot, o también denominado eje no inercial.

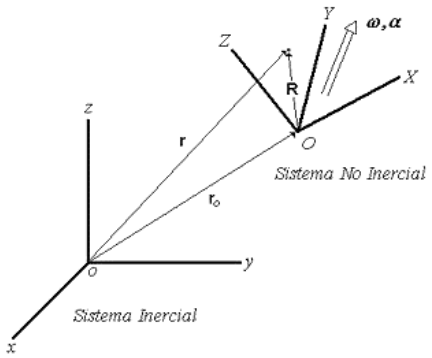


Figura. 4. 25 Ejes de coordenadas (Dhaouadi y Hatab, 2013)

El eje de coordenadas del robot se grafica en el centro de masa, que teóricamente debe estar a la distancia media del eje de las ruedas, a su vez la posición del robot en el marco de referencia inercial se puede representar por la ecuación.

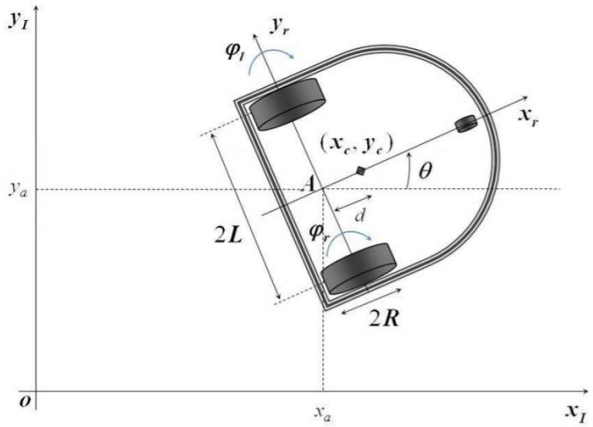


Figura. 4. 26 Robot móvil de transmisión diferencial (Dhaouadi y Hatab, 2013)

$$q = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}$$

Donde se denota como una función en el eje inercial, ya que el robot puede tener *n* cantidad de funciones inerciales.

En cada uno de los marcos de referencia se puede representar como.

$$X^r = \begin{bmatrix} x^r \\ y^r \\ \theta^r \end{bmatrix} \quad \text{y} \quad X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

En donde X^r se define como las coordenadas del robot y X son las coordenadas inerciales.

Los dos sistemas de referencia se pueden relacionar en una transformación ortogonal.

$$Y = R(\theta)X^r \qquad \text{(Ecuación. 4.1)}$$

4.8.2 Restricciones no-holonómicas

Por las restricciones de movimiento el robot desde el punto de vista del marco no inercial no puede desplazarse en el eje Y por lo tanto:

$$Y = 0$$

$$\therefore \quad X = R(\theta)X^r$$

Donde $R(\theta)$ es una matriz ortogonal definida por la siguiente matriz.

$$R(\theta) = \begin{pmatrix} cos\theta & sen\theta & 0 \\ -sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{(Ecuación. 4.2)}$$

El robot móvil se caracteriza por tener dos restricciones holonómicas y una restricción no-holonómica que se obtiene de dos supuestos.

4.8.3 Restricciones no-holonómicas del robot

Sin movimiento de deslizamiento lateral, esto significa que el robot no puede desplazarse en su eje Y de eje de coordenadas no inercial. Esto implica que la velocidad a lo largo del eje lateral es cero:

$$\dot{y}^r = 0$$

Mediante la matriz de rotación ortogonal ecuación 4.2

$$\begin{pmatrix} cos\theta & sen\theta & 0 \\ -sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{x}^r \\ \dot{y}^r \\ \dot{\theta}^r \end{pmatrix}$$

Se iguala a cero $\dot{y}^r$:

$$-\dot{x}sen\theta + \dot{y}cos\theta = 0 \qquad \text{(Ecuación. 4.3)}$$

4.8.4 Restricciones holonómicas del robot

Restricción de rodadura pura, las ruedas al ser circulares tienen un solo punto de contacto con el suelo denominado P, al presentar esta restricción las ruedas no generan deslizamiento en el eje longitudinal (X^r), y no patinar en eje ortogonal (Y^r), donde intervienen las velocidades de cada rueda.

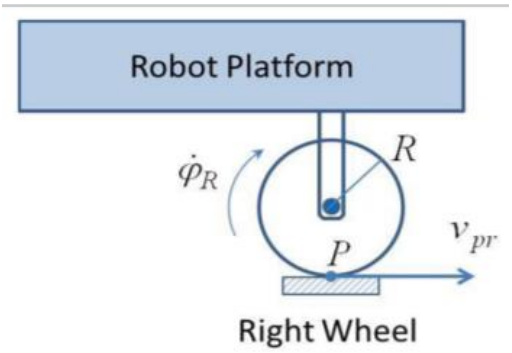


Figura. 4. 27 Restricción de movimiento de balanceo puro (Dhaouadi y Hatab, 2013)

Se puede calcular la velocidad de rodadura en el sistema, con la siguiente ecuación.

$$V_d = R\dot{\varphi}_d \qquad \text{(Ecuación. 4.4)}$$
$$V_i = R\dot{\varphi}_i \qquad \text{(Ecuación. 4.5)}$$

Donde: Vd es la velocidad lineal de la rueda derecho.

Vi es la velocidad lineal de la rueda izquierda.

R es el radio de la rueda.

$\dot{\varphi}_d$ es la velocidad angular de la rueda derecha

$\dot{\varphi}_i$ es la velocidad angular de la rueda derecha

Estas ecuaciones se las puede trasladar al sistema inercial, mediante el análisis de la función de velocidades del centro de masa del robot.

Donde: X y Y son el desplazamiento de las ruedas en los ejes

l es la distancia del centro del eje de las ruedas al centro de masa

Para la rueda derecha

$$x_d = x_r + lsin\theta \quad \text{(Ecuación. 4.6)}$$

$$y_d = y_r - lcos\theta \quad \text{(Ecuación. 4.7)}$$

Se derivan las ecuaciones 4.6 y 4.7 para obtener las velocidades.

$$\dot{x}_d = \dot{x}_r + l\dot{\theta}cos\theta$$

$$\dot{y}_d = \dot{y}_r + l\dot{\theta}sen\theta$$

Para la rueda izquierda

$$x_i = x_r - lsin\theta \quad \text{(Ecuación. 4.8)}$$

$$y_i = y_r + lcos\theta \quad \text{(Ecuación. 4.9)}$$

Se derivan las ecuaciones 4.8 y 4.9 para obtener las velocidades.

$$\dot{x}_i = \dot{x}_r - l\dot{\theta}cos\theta$$

$$\dot{y}_i = \dot{y}_r - l\dot{\theta}sen\theta$$

Se utilizará la matriz de rotación (ecuación 4.2) para obtener las siguientes ecuaciones:

$$l\dot{\theta} + \dot{x}_r cos\theta + \dot{y}_r sin\theta = R\dot{\varphi}_r \quad \text{(Ecuación. 4.10)}$$

$$-l\dot{\theta} + \dot{x}_r cos\theta + \dot{y}_r sin\theta = R\dot{\varphi}_L \quad \text{(Ecuación. 4.11)}$$

Si se suma las ecuaciones 4.10 y 4.11

$$\begin{array}{r} l\dot{\theta} + \dot{x}_r cos\theta + \dot{y}_r sin\theta = R\dot{\varphi}_r \\ -l\dot{\theta} + \dot{x}_r cos\theta + \dot{y}_r sin\theta = R\dot{\varphi}_L \\ \hline 0 + 2\dot{x}_r cos\theta + 2\dot{y}_r sin\theta = R\dot{\varphi}_r + R\dot{\varphi}_L \end{array} \quad \text{(Ecuación. 4.12)}$$

Si se resta las ecuaciones 4.10 y 4.11

$$\begin{array}{r} l\dot{\theta} + \dot{x}_r cos\theta + \dot{y}_r sin\theta = R\dot{\varphi}_r \\ -l\dot{\theta} + \dot{x}_r cos\theta + \dot{y}_r sin\theta = R\dot{\varphi}_L \\ \hline 2l\dot{\theta} + 0 + 0 = R\dot{\varphi}_r - R\dot{\varphi}_L \end{array} \quad \text{(Ecuación. 4.13)}$$

Se despeja $\dot{Y}$ de la ecuación 4.3:

$$\dot{y}_r = \frac{\dot{x}_r sen\theta}{cos\theta}$$

Se iguala en la ecuación 4.12

$$2\dot{x}_r cos\theta + 2\left(\frac{\dot{x}_r sen\theta}{cos\theta}\right) sin\theta = R\dot{\varphi}_r + R\dot{\varphi}_L$$

$$2\dot{x}_r cos^2\theta + 2\dot{x}_r sin^2\theta = (R\dot{\varphi}_r + R\dot{\varphi}_L)cos\theta$$

$$2\dot{x}_r(1) = (R\dot{\varphi}_r + R\dot{\varphi}_L)cos\theta$$

$$\dot{x}_r = \frac{R}{2}(\dot{\varphi}_r + \dot{\varphi}_L)cos\theta \qquad \text{(Ecuación. 4.14)}$$

Se despeja la ecuación 4.14 para los otros ejes el resultado da las ecuaciones cinemáticas:

$$\dot{x} = \frac{R}{2}(\dot{\varphi}_r + \dot{\varphi}_L)cos\theta \qquad \text{(Ecuación. 4.15)}$$

$$\dot{y} = \frac{R}{2}(\dot{\varphi}_r + \dot{\varphi}_L)sin\theta \qquad \text{(Ecuación. 4.16)}$$

$$\dot{\theta} = \frac{R}{2l}(\dot{\varphi}_r - \dot{\varphi}_L) \qquad \text{(Ecuación. 4.17)}$$

Se traslada las ecuaciones 4.15, 4.16 y 4.17 a matrices se obtiene:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix} = \frac{R}{2}\begin{pmatrix}cos\theta & cos\theta\\ sen\theta & sen\theta\\ \frac{1}{l} & -\frac{1}{l}\end{pmatrix}\begin{pmatrix}\varphi_d\\ \dot{\varphi}_l\end{pmatrix}$$

La cual se expresa de la siguiente forma:

$$\dot{x} = S(\theta)n \qquad \text{(Ecuación. 4.18)}$$

4.8.5 Dinámica del robot

Se encarga del estudio de las fuerzas que actúan en el sistema para conocer el movimiento del robot a lo largo de un tiempo determinado. Las variables de acción son velocidad, aceleración, torque, longitud, masa e inercia.

4.8.6 Enfoque de la Lagrange

Las ecuaciones de *Lagrange* permiten definir los lineamientos de movimiento de los sistemas mecánicos, para conseguirlo se derivan sistemáticamente las ecuaciones de movimiento donde se considerarán la cinética y el potencial energético del sistema.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_1}\right) - \frac{\partial L}{\partial q_1} = F - \Lambda^T(q)^\lambda \qquad \text{(Ecuación. 4.19)}$$

Donde: $F\ es\ el\ vector\ de\ fuerza\ generalizada$

Λ son las restricciones de matrices

λ es el vector multiplicador de la Lagrange asociado a las restricciones

Para derivar el modelo dinámico se utiliza el enfoque de *Lagrange*, primero se debe calcular las energías cinética y potencial. Dado que el sistema se encuentra en movimiento la energía potencial se considera cero.

Las coordenadas generalizadas se definirán como:

$$q = [X \;\; Y \;\; \theta \;\; \varphi_r \;\; \varphi_l]^T \qquad \text{(Ecuación. 4.20)}$$

La energía cinética del robot es la $\sum$ de las energías cinéticas del robot sin ruedas más la energía de las ruedas y actuadores.

La ecuación de la energía cinética de la plataforma robot es:

$$E_c = \frac{1}{2} m_c {V^2}_c + \frac{1}{2} I_c \dot{\theta}^2 \qquad \text{(Ecuación. 4.21)}$$

Las energías cinéticas de las ruedas:

$$E_{rd} = \frac{1}{2} m_r {V^2}_r + \frac{1}{2} I_m \dot{\theta^2} + \frac{1}{2} I_w \dot{\varphi}^2 \qquad \text{(Ecuación. 4.22)}$$

$$E_{rl} = \frac{1}{2} m_r {V^2}_r + \frac{1}{2} I_m \dot{\theta^2} + \frac{1}{2} I_w \dot{\varphi}^2 \qquad \text{(Ecuación. 4.23)}$$

Donde: m_c es la masa del robot sin ruedas ni motores

m_r es la masa de cada rueda con su motor

I_c es el momento de inercia del robot en el eje vertical del centro masa

I_w es el momento de inercia de cada rueda con un motor alrededor del eje de la rueda

I_m es el momento de inercia de cada rueda con un motor alrededor del diámetro de la rueda.

Todas las ecuaciones se expresan como una función generalizada en el marco inercial.

$${V^2}_i = {X^2}_i + {Y^2}_i \qquad \text{(Ecuación. 4.24)}$$

Las componentes X y Y son en relación al marco no inercial, mientras que el sistema de las ruedas se puede expresar en términos de la función generalizada.

$$\begin{cases} x = x_r + dcos\theta \\ y = y_r + dcsen\theta \end{cases}$$

$$\begin{cases} x_{wd} = x_r + Lcos\theta \\ y_{wd} = y_r + Lcsen\theta \end{cases}$$

$$\begin{cases} x_{wi} = x_r + Lcos\theta \\ y_{wi} = y_r + Lcsen\theta \end{cases}$$

Se relacionan las ecuaciones 4.21, 4.22, y 4.23 el total de la energía cinética del robot es:

$$E = \frac{1}{2}m\left(\dot{x}^2{}_r + \dot{y}^2{}_r\right) - m_c d\dot{\theta}(\dot{y}_r cos\theta - \dot{x}_r sen\theta) + \frac{1}{2}I_w(\dot{\varphi}^2{}_d + \dot{\varphi}^2{}_i) + \frac{1}{2}I\theta^2$$

Donde los siguientes parámetros son introducidos

$$m = m_c + 2m_w \qquad \text{(Ecuación. 4.25)}$$

De donde *m* es la masa total del robot, se trasladará al sistema inercial:

$$I = I_c + m_c d^2 + 2m_w L^2 + 2I_m \qquad \text{(Ecuación. 4.26)}$$

Se usa la ecuación de la *Lagrange* L=T de las ecuaciones de movimiento del robot se tiene:

$$m\ddot{x}_a - md\ddot{\theta}sin\theta - md\theta^2 cos\dot{\theta} = C_1$$

$$m\ddot{Y}_a - md\ddot{\theta}sin\theta - md\theta^2 cos\dot{\theta} = C_2$$

$$I\ddot{\theta} - md\ddot{x}_a sin\theta - md\ddot{y}_a cos\theta = C_3$$

$$I_w\varphi_d = \ddot{\tau}_d + C_4$$

$$I_w\varphi_\iota = \ddot{\tau}_\iota + C_5$$

Las expresiones denominadas como $(C_1, C_2, C_3, C_4, C_5)$, son coeficientes relacionados con las constantes cinemáticas, que pueden ser escritos en términos de los multiplicadores de *Lagrange*

$$\Lambda^T(q) = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{bmatrix}$$

Con las ecuaciones de movimiento obtenidas, se puede representar en forma general por la ecuación.

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} = B(q)\tau - \Lambda^T(q)^{\lambda} \qquad \text{(Ecuación. 4.27)}$$

Donde:

$$M(q) = \begin{bmatrix} m & 0 & -mdsen\theta & 0 & 0 \\ 0 & m & mdcos\theta & 0 & 0 \\ -mdsen\theta & mdcos\theta & I & 0 & 0 \\ 0 & 0 & 0 & I_w & 0 \\ 0 & 0 & 0 & 0 & I_w \end{bmatrix}$$

$$V(q,\dot{q}) = \begin{bmatrix} 0 & -m_c d\dot{\theta} cos\theta & 0 & 0 & 0 \\ 0 & -m_c d\dot{\theta} sen\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B(q) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, and\ \Lambda^T(q)\lambda = \begin{bmatrix} -sen\theta & cos\theta & cos\theta \\ cos\theta & sen\theta & sen\theta \\ 0 & L & -L \\ 0 & -R & 0 \\ 0 & 0 & -R \end{bmatrix} x \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

La ecuación 4.27 se puede expresar de otra forma más conveniente para el análisis de control y simulación. El propósito es eliminar el término de restricción $\Lambda^T(q)^{\lambda}$ dado que los multiplicadores de *Lagrange* son desconocidos, para tal propósito se calcula el vector reducido.

$$\dot{\eta} = \begin{bmatrix} \dot{\varphi}_d \\ \dot{\varphi}_i \end{bmatrix} \qquad \text{(Ecuación. 4.28)}$$

Las velocidades en las coordenadas generalizadas que utilizan el modelo cinemático directo se las puede expresar de la siguiente manera:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta} \\ \dot{\varphi}_d \\ \dot{\varphi}_i \end{bmatrix} = \frac{1}{2} \begin{bmatrix} Rcos\theta & Rcos\theta \\ Rsen\theta & Rsen\theta \\ \frac{R}{L} & -\frac{R}{L} \\ 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \dot{\varphi}_d \\ \dot{\varphi}_i \end{bmatrix} \qquad \text{(Ecuación. 4.29)}$$

Esto se puede expresar como:

$$\dot{q} = S(q)\eta \qquad \text{(Ecuación. 4.30)}$$

Se puede verificar que la matriz de la transformación S(q) están en nulo:

$$S^T(q)\Lambda^T(q) = 0 \qquad \text{(Ecuación. 4.31)}$$

Se deriva de la ecuación 4.30

$$\ddot{q} = \dot{S}(q)\eta + S(q)\dot{\eta} \qquad \text{(Ecuación. 4.32)}$$

Se sustituye las ecuaciones 4.30, 4.31 y 4.32 en la ecuación 4.27

$$M(q)\left[\dot{S}(q)\eta + S(q)\dot{\eta}\right] + V(q,\dot{q})[S(q)\eta] = B(q)\tau - \Lambda^T(q)\lambda$$

$$S^T(q)M(q)S(q)\dot{\eta} + S^T(q)\left[M(q)\dot{S}(q) + V(q,\dot{q})S(q)\right]\eta = S^T(q)B(q)\tau - S^T(q)\Lambda^T(q)\lambda(45) \qquad \text{(Ecuación. 4.33)}$$

Ahora se determina la ecuación 4.33 en matrices:

$\bar{M}(q) = S^T(q)M(q)S(q)$

$\bar{V} = S^T(q)M(q)\dot{S}(q) + S^T(q)V(q,\dot{q})S(q)$

$\bar{B} = S^T(q)B(q)$

Las ecuaciones dinámicas son reducidas a la expresión

$$\bar{M}(q)\eta + \bar{V}(\dot{q},\dot{q})\eta = \bar{B}(q)\tau \qquad \text{(Ecuación. 4.34)}$$

Donde:

$$\bar{M}(q) = \begin{bmatrix} I_w + \frac{R^2}{4L^2}(mL^2 + I) & \frac{R^2}{4L^2}(mL^2 + I) \\ \frac{R^2}{4L^2}(mL^2 + I) & I_w + \frac{R^2}{4L^2}(mL^2 + I) \end{bmatrix} \qquad \text{(Ecuación. 4.35)}$$

$$\bar{V}(q,\dot{q}) = \begin{bmatrix} 0 & \frac{R^2}{4L}m_c d\dot{\theta} \\ -\frac{R^2}{4L}m_c d\dot{\theta} & 0 \end{bmatrix}, \bar{B}(q) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{(Ecuación. 4.36)}$$

En la expresión anterior se muestra que la ecuación de la dinámica del robot se expresa en relación de las velocidades angulares de cada una de las ruedas $(\dot{\varphi}_d, \dot{\varphi}_i)$, la velocidad angular del robot $\dot{\theta}$ y el torque de los motores (τ_d, τ_i).

4.8.7 Modelado de los motores

Se utilizarán motores de corriente continua con armadura, para lo cual la tensión del inducido se utiliza como entrada de control mientras se mantiene las condiciones en el

circuito de campo. Las ecuaciones que definen el motor de corriente continua de imán permanente son:

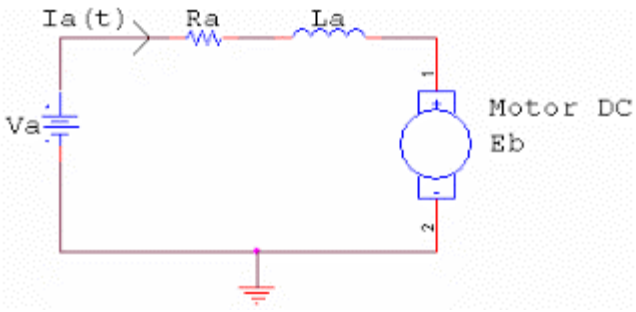


Figura. 4. 28 Diagrama eléctrico equivalente del motor DC (Manual, 2012)

$$\begin{cases} v_{a=R_a i_a + L_a \frac{di_a}{dt} + e_a} \\ e_a = K_b \omega_m \\ \tau_m = K_t i_a \\ \tau = N \tau_m \end{cases} \qquad \text{(Ecuación. 4.37)}$$

Donde: i_a es la corriente de armadura

R_a, L_a la resistencia e inductancia del devanado respectivamente

e_a es la espalda de la fuerza electromotriz

ω_m es la velocidad angular del rotor

τ_m es el torque del motor

K_t, K_b son las constantes de: torque y energía electromotriz respectivamente.

N es la relación de transmisión

τ el torque de salida aplicado a la rueda

Los motores de la plataforma robótica móvil diferencial están acoplados mecánicamente a las ruedas a través de los engranajes, por tal razón las ecuaciones mecánicas del movimiento de los motores están vinculados directamente con la dinámica mecánica del robot.

$$\begin{cases} \omega_r = N\dot{\varphi}_d \\ \omega_i = N\dot{\varphi}_i \end{cases}$$

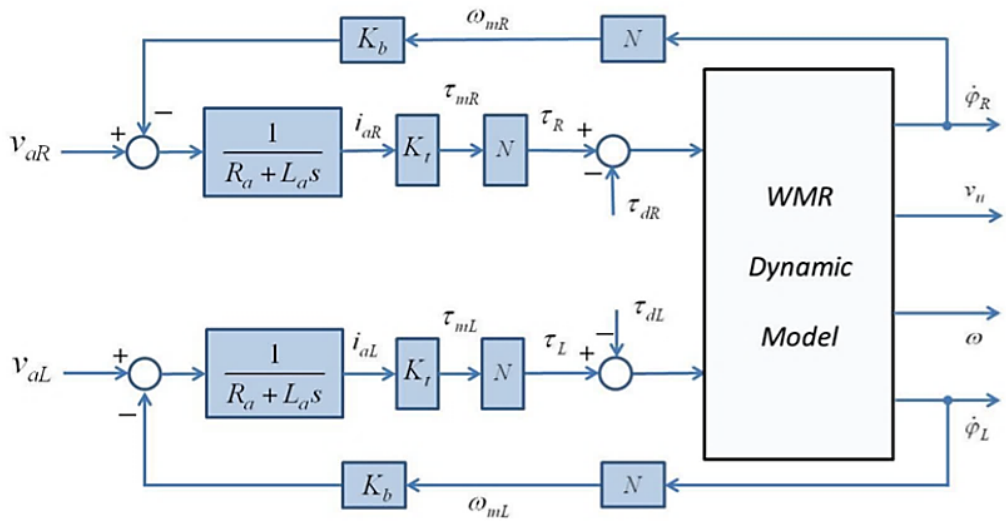


Figura. 4. 29 Diagrama del modelo dinámico con motores acoplados (Dhaouadi y Hatab, 2013)

4.9 Control del sistema

Para desarollar las ecuaciones de control se debe resumir las ecuaciones del robot que permiten describir su dinamica.

$$\bar{M}(q)\eta + \bar{V}(\dot{q}, \dot{q})\eta = \bar{B}(q)\tau \qquad \text{(Ecuación. 4.38)}$$

$$\dot{X} = S(q)\eta \qquad \text{(Ecuación. 4.39)}$$

Donde η es una matriz de las velocidades angulares

$$n = \begin{pmatrix} w_d \\ w_i \end{pmatrix}$$

y S es una matriz ortogonal representada con la siguiente expresión

$$S = \frac{R}{2}\begin{pmatrix} cos\theta & cos\theta \\ sen\theta & sen\theta \\ \frac{1}{l} & -\frac{1}{l} \end{pmatrix}$$

La propuesta de control presentada consiste en aplicar el control PID sobre la ecuacion dinámica del robot, dada la pequeña no linealidad presentada en la ecuación 4.36, la cual depende no linealmente de la distancia entre el centro de masas del robot y el eje de rotación

del mismo, se intenta minimizar el diseño del robot por lo que el control PID puede funcionar muy bien sobre el sistema dinámico y un control no lineal sobre el sistema cinemático el mismo que se trata primero, dado que es el más importante ya que trata directamente sobre las variables de control de trayectoria.

4.10 Aproximación de Euler

Consiste en encontrar la solución de una ecuación diferencial de primer orden y valores iniciales conocidos para un rango de valores.

$$Y_{k+1} = Y_k + h.f(x_k, y_k) \qquad \text{(Ecuación. 4.40)}$$

Donde: Y es la solución de la ecuación diferencial

f es la ecuación diferencial en función de las variables independientes

Se evalúa en el modelo cinemático.

$$\frac{dy}{dx} = f(x_k, y_k)$$

$$Y(k+1) = Y(k) + h\frac{dy}{dx} \qquad \text{(Ecuación. 4.41)}$$

h debe ser un número tal que se considere que $h^2 \to 0$

$$Y(k+1) = Y(k) + hf(x_k, y_k)$$

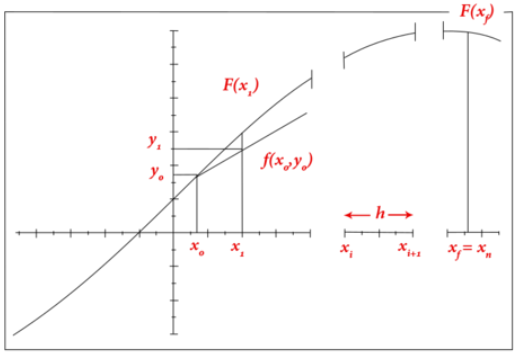


Figura. 4. 30 Diagrama de una función que evalua en los puntos K y K+1 (Cuánticos, C, 2011)

Esta aproximación de *Euler* permite discretizar las ecuaciones diferenciales de primer orden y dado que la ecuación cinemática es una ecuación diferencial de primer orden se puede realizar esta aproximación para pasar a tiempo discreto la ecuación cinemática.

$$\dot{q} = S(q)\eta$$

$$q(n+1) = q(n) + \Delta t.S\big(q(n)\big)\eta(n)$$

$$q(n+1) - q(n) = \Delta t.S\big(q(n)\big)\eta(n)$$

Se puede definir el vector $fd = q(n+1) - q(n)$ y $St = \Delta t.S\big(q(n)\big)$

$$fd = St.\eta \qquad \text{(Ecuación. 4.42)}$$

4.11 Control del vector de posiciones

Dada la ecuación discreta del modelo cinemático el problema a resolver se resume en encontrar el vector de velocidades angulares que permite obtener el vector fd para los puntos en el tiempo futuro dada la posición inicial del robot.

Se obtiene la solución de un sistema de tres ecuaciones que se plantean a continuación.

$$ax + by + cz = D_1$$

$$a'x + b'y + c'z = D_2$$

$$a''x + b''y + c''z = D_3$$

Se expresa de forma matricial las ecuaciones.

$$\begin{pmatrix} a & b & c \\ a' & b' & c' \\ a'' & b'' & c'' \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix} \qquad \text{(Ecuación. 4.43)}$$

$$A * X = Y \qquad \text{(Ecuación. 4.44)}$$

La matriz A es la matriz característica del sistema.

Para saber si el sistema tiene solución se debe contemplar tres posibilidades, la primera que tiene solución, la segunda que tiene infinitas soluciones y por último que tenga solución exacta.

Para saber en cual caso del sistema se halla la solución, se ocupa el teorema de *Rouché-Frobenius* el cual enuncia que si el rango de la matriz A es igual al número de columnas entonces tiene solución única, de otro modo existen infinitas soluciones para el sistema.

Si el sistema tiene solución está dada por la siguiente ecuación:

$$(A^{-1}A)x = A^{-1}y$$

$$x = A^{-1}y \qquad \text{(Ecuación. 4.45)}$$

Por tanto, para verificar si la ecuación discreta tiene solución se debe encontrar el rango de la matriz St para lo cual se realiza la reducción por filas para llegar a la identidad

$$\begin{pmatrix} cos\theta & cos\theta \\ sen\theta & sen\theta \\ \frac{1}{l} & -\frac{1}{l} \end{pmatrix} \xrightarrow{F1=F_1\frac{1}{cos\theta}} \begin{pmatrix} 1 & 1 \\ sen\theta & sen\theta \\ \frac{1}{l} & -\frac{1}{l} \end{pmatrix} \xrightarrow{F2=F_2\frac{1}{sen\theta}} \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ \frac{1}{l} & -\frac{1}{l} \end{pmatrix}$$

$$\xrightarrow{F3=F_3 l} \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & -1 \end{pmatrix} \xrightarrow{F1=F_1+F3} \begin{pmatrix} 2 & 0 \\ 1 & 1 \\ 1 & -1 \end{pmatrix} \xrightarrow{F2=F_2+F3} \begin{pmatrix} 2 & 0 \\ 2 & 0 \\ 1 & -1 \end{pmatrix} \xrightarrow{F3=F_3-F1/2} \begin{pmatrix} 2 & 0 \\ 2 & 0 \\ 0 & -1 \end{pmatrix} \xrightarrow{F3=F_3*(-1)} \begin{pmatrix} 2 & 0 \\ 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\xrightarrow{F1=F_1/2} \begin{pmatrix} 1 & 0 \\ 2 & 0 \\ 0 & 1 \end{pmatrix} \xrightarrow{F2=F_2/2} \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Para encontrar una solución exacta del sistema basta con encontrar la inversa de la matriz S, sin embargo, dado que la matriz presenta dos filas linealmente dependientes el rango de la matriz es dos, por lo que el sistema no presenta una solución exacta.

Si el rango de la matriz es el problema para encontrar solución exacta la única forma es ocupar dos variables y dejar que la tercera variable siga la solución del sistema, a este método se le conoce como control por desacoplamiento de estados.

Dado que $X\ y\ \theta$ son linealmente independientes, se puede escoger cualquiera de las dos variables.

$$\begin{pmatrix} X_{n+1} - X_n \\ \theta_{n+1} - \theta_n \end{pmatrix} = \Delta t \frac{R}{2} \begin{pmatrix} cos\theta & cos\theta \\ \frac{1}{l} & -\frac{1}{l} \end{pmatrix} \qquad \text{(Ecuación. 4.46)}$$

El rango (S)=2

Se determina si el sistema tiene solución exacta, se tiene 2 variables y si la variable Y sigue linealmente a X. Esto produce que el tiempo de establecimiento sea más alto ya que teta puede tomar valores libres y X también, pero Y no puede seguir la trayectoria libremente porque está atada linealmente a X, por tanto, la convergencia depende de las condiciones en las que X se aproxime a Y.

4.12 Mínimos Cuadrados

Es una técnica de optimización matemática, en la que un conjunto de puntos en el eje de coordenadas y una serie de funciones permite determinar la función continua, en términos generales es intentar hallar la línea recta que mejor se aproxima a una serie de datos en el plano de manera que se pueda predecir el comportamiento del fenómeno en estudio, en otros

términos, es encontrar los valores de los parámetros que minimicen la suma de los errores al cuadrado.

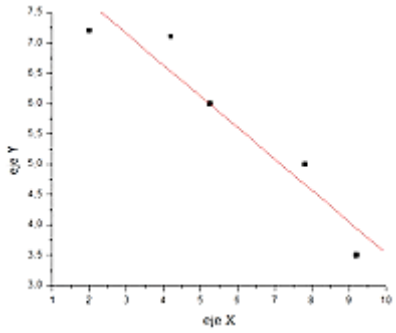


Figura. 4. 31 Diagrama de distorsión de puntos con la recta de tendencia (Miprofe, 2014)

Si por ejemplo se realizarán medidas experimentales sujetas a ruido estos puntos tienen una dispersión, pero presentan una tendencia, dado que una recta pasa por dos puntos al tener varias mediciones, no se puede encontrar una solución exacta al problema, pero se puede encontrar los parámetros de una recta que mejor se aproxime a todos los puntos de la medición.

Se desarrolló el caso de una recta en el plano X-Y, se define por la ecuación de la recta, al tomar varios puntos se puede evaluar el valor que existe entre las distancias del valor esperado y el valor medido con ruido.

$$a_1 x1 + b_1 y1 = C_1$$

$$\vdots \quad \vdots \quad \vdots$$

$$a_1 xn + b_1 yn = C_1$$

$$\begin{pmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix} \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} = \begin{pmatrix} C_1 \\ \vdots \\ C_1 \end{pmatrix}$$

Se reduce la ecuación en una expresión más sencilla.

$$Ax = y \qquad \text{(Ecuación. 4.47)}$$

El objetivo es minimizar la diferencia entre el valor de la recta y el de los puntos medidos.

$$\|Y_r - Y_{med}\|_2^2 = \|Y_r - A_x\|_2^2$$

$$\|Y_r - Y_{med}\|_2^2 = (Y_r - A_x)^t (Y_r - A_x)$$

$$\|Y_r - Y_{med}\|_2^2 = Y^tY - Y^tA_x + x^tA^tA_x$$

$$min\|Y_r - Y_{med}\|_2^2 = \frac{d}{dx}(Y^tY - 2Y^tA_x + x^tA^tA_x)$$

$$0 = -2Y^tA_x + x^tA^tA_x$$

$$2A^ty = 2A^tA_x$$

$$A^tAX = A^ty$$

$$X = (A^tA)^{-1}A^ty \qquad \text{(Ecuación. 4.48)}$$

A la ecuación que permite encontrar el valor de x de la denomina ecuación normal.

4.13 Condiciones de solución exacta

Dado que el sistema tiene una fila linealmente dependiente se puede obligar al sistema a cumplir la condición para que el sistema tenga solución exacta en todas sus variables, sin embargo, dado que en un inicio el robot puede localizarse con posición y ángulo diferentes a las necesarias para la solución exacta esta condición debe ser introducida como el valor deseado a las que el controlador debe tender. Para ello se debe encontrar el espacio columna de la matriz ampliada para encontrar la condición necesaria de nulidad que permita obtener soluciones infinitas sobre la variable de restricción, de forma que quienes determinan la solución son las dos variables restantes que son linealmente independientes.

Para encontrar el espacio columna de la matriz se debe reducir por filas la matriz ampliada.

$$\begin{pmatrix} cos\theta & cos\theta & \vdots\, \Delta x \\ sen\theta & sen\theta & \vdots\, \Delta y \\ \frac{1}{l} & -\frac{1}{l} & \vdots\, \Delta\theta \end{pmatrix} \xrightarrow{F1=F_1\frac{1}{cos\theta}} \begin{pmatrix} 1 & 1 & \vdots\, \frac{\Delta x}{cos\theta} \\ sen\theta & sen\theta & \vdots\, \Delta y \\ \frac{1}{l} & -\frac{1}{l} & \vdots\, \Delta\theta \end{pmatrix}$$

$$\xrightarrow{F2=F_2\frac{1}{sen\theta}} \begin{pmatrix} 1 & 1 & \vdots\, \frac{\Delta x}{cos\theta} \\ 1 & 1 & \vdots\, \frac{\Delta y}{sen\theta} \\ \frac{1}{l} & -\frac{1}{l} & \vdots\, \Delta\theta \end{pmatrix} \xrightarrow{F1=F_1-F2} \begin{pmatrix} 0 & 0 & \vdots\, \frac{\Delta x}{cos\theta} - \frac{\Delta y}{sen\theta} \\ 1 & 1 & \vdots\, \frac{\Delta y}{sen\theta} \\ \frac{1}{l} & -\frac{1}{l} & \vdots\, \Delta\theta \end{pmatrix}$$

Por tanto, la condición del espacio columna es la siguiente.

$$\frac{\Delta x}{cos\theta} - \frac{\Delta y}{sen\theta} = 0$$

$$\frac{\Delta y}{\Delta x} = tan\theta \qquad \text{(Ecuación. 4.49)}$$

La ecuación 4.49 señala que si el ángulo del robot es exactamente el ángulo formado entre el punto siguiente y el actual el sistema tiene solución exacta para todas sus variables y por tanto al aplicar el algoritmo de minimización la solución obtenida multiplicada por la matriz cinemática será la identidad, por lo que la ecuación cinemática se linealizará y por ende el controlador seguirá la trayectoria definida por una ecuación diferencial lineal que es conocida.

Cabe recalcar que esta condición no se cumple siempre, ya que los puntos de la curva parametrizada no siempre serán los puntos que cumplan esta condición, pero si se obliga a teta a través del controlador a corregir el ángulo para que apunte hacia el valor deseado rápidamente se puede corregir el error de posición de forma que converja lentamente y permita que el controlador linealice por realimentación el sistema.

4.14 Controlador acoplado del vector posición

Dado que el sistema de ecuaciones discretas no tiene solución exacta se puede ocupar el concepto de las ecuaciones normales para encontrar un valor que minimice la distancia entre el valor en el tiempo futuro y el valor actual, por lo que el error no va a ser exactamente cero, sin embargo a fines prácticos es mejor encontrar un vector de velocidades angulares que aproxime al robot en un tiempo de muestreo, por tanto el error converge en un tiempo menor que el caso de control por estados desacoplados ya que las tres variables se minimizan en cada tiempo de muestreo a la vez.

$$fd = St.\eta \qquad \text{(Ecuación. 4.50)}$$

Se aplica sobre esta ecuación el mismo concepto de mínimos cuadrados se obtiene.

$$Error = \|fdreal - fd\|_2 = \|fdreal - St.\eta\|_2$$

Por lo que la solución del problema está dada por la ecuación normal.

$$\eta = \left(S_t{}^t S_t\right)^{-1} S_t{}^t.fd \qquad \text{(Ecuación. 4.51)}$$

Donde $fb = X_{dn+1} - X_n$ es la relación entre el valor deseado en tiempo futuro y el valor en el tiempo actual, por lo que el error es proporcional lo cual provoca grandes acciones de control y que el sistema pueda ser inestable.

Dado que la ecuación diferencial en sus componentes x, y linealizada tiene la forma

$$X_{dn+1} - X_n = a * u$$

Por lo que u podrá tener la forma:

$$u = a^{-1}(X_{dn+1} - X_n - k(X_{dn} - X_n))$$

Dónde: K es la constante de convergencia x, y

$$X_{n+1} = X_n + X_{dn+1} - X_n - k(X_{dn} - X_n)$$

$$E_{n+1} = kE_n$$

La ecuación es estable en el tiempo para el error y es convergente a cero siempre que $0 < k < 1$, si el valor de k es cercano a cero el error converge rápidamente, pero genera acciones de control altas las cuales generan dinámicas altas en el robot, y valores cercanos a uno provocan convergencia del error en un tiempo más alto, pero con acciones de control suaves.

En el caso del ángulo teta como se mencionó en el subcapítulo anterior el valor debe converger más rápidamente que el controlador de posición y debe tener un error de posición de cero, por lo que la corrección de la acción de control ocupado anteriormente no es factible, ya que cualquier ruido que presente el sistema puede ocasionar una inestabilidad que produzca oscilaciones entonces para el caso del ángulo teta se debe colocar una corrección del error proporcional e integrativo dado por la siguiente ecuación

$$f_\theta = k_p E_\theta + k_i Se = k_p(\theta_{dn} - \theta n) + k_i Se \qquad \text{(Ecuación. 4.52)}$$

Donde Se es el valor integral discreto dado por

$$Se(k) = Se(k-1) + \Delta t * E_\theta(k)$$

Dicho esto, el controlador tenderá a la siguiente forma:

$$\eta = \left(S_t{}^t S_t\right)^{-1} S_t{}^t . fd \qquad \text{(Ecuación. 4.53)}$$

$$\eta = \frac{1}{R * tm}\begin{pmatrix}\cos\theta & \cos\theta & l \\ \sin\theta & \sin\theta & -l\end{pmatrix}\begin{pmatrix}x_{dn+1} - x_n - k(x_{dn} - x_n) \\ y_{dn+1} - y_n - k(y_{dn} - y_n) \\ k_p(\theta_{dn} - \theta n) + k_i Se\end{pmatrix}$$

$$\eta = \frac{1}{R * tm}\begin{pmatrix}Dx * \cos\theta + Dy * \sin\theta + Dteta * l \\ Dx * \cos\theta + Dy * \sin\theta - Dteta * l\end{pmatrix}$$

$$\theta_{dn} = \operatorname{atan}\left(\frac{\Delta y}{\Delta x}\right) \qquad \text{(Ecuación 4.54)}$$

4.15 Controlador de velocidad angular de las ruedas.

Para el control de la velocidad angular de las ruedas se debe tomar en cuenta que la ecuación dinámica presenta no linealidad en su componente $\bar{V}(q,\dot{q})$ dado que depende de la velocidad angular, esta no linealidad es producto de

$$\frac{R^2}{4L} m_c d\dot{\theta}$$

donde el valor de *d* que es la distancia entre el centro de masa y el eje de rotación es muy pequeña para el caso de un robot homogéneamente distribuido, por lo que el diseño debe tener una distribución que permita anular o al menos reducir este valor, con lo cual la aproximación de esta ecuación sea lineal e independiente de la velocidad angular.

$$\bar{M}(q)\dot{\eta} + \bar{V}(\dot{q},\dot{q})\eta = \bar{B}(q)\tau \qquad \text{(Ecuación. 4.55)}$$

$$\bar{M}(q) = \begin{bmatrix} I_w + \frac{R^2}{4L^2}(mL^2 + I) & \frac{R^2}{4L^2}(mL^2 + I) \\ \frac{R^2}{4L^2}(mL^2 + I) & I_w + \frac{R^2}{4L^2}(mL^2 + I) \end{bmatrix} \qquad \text{(Ecuación. 4.56)}$$

$$\bar{V}(q,\dot{q}) = \begin{bmatrix} 0 & \frac{R^2}{4L} m_c d\dot{\theta} \\ -\frac{R^2}{4L} m_c d\dot{\theta} & 0 \end{bmatrix} \qquad \text{(Ecuación. 4.57)}$$

$$\bar{B}(q) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{(Ecuación. 4.58)}$$

Se determina que la matriz coriolis transformada ecuación 4.57, se anula.

$$\bar{V}(q,\dot{q}) = \begin{bmatrix} 0 & \frac{R^2}{4L} m_c d\dot{\theta} \\ -\frac{R^2}{4L} m_c d\dot{\theta} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Por lo tanto, la ecuación dinámica 4.55 es la siguiente.

$$\bar{M}(q)\dot{\eta} = \bar{B}(q)\tau \qquad \text{(Ecuación. 4.59)}$$

Dado que la matriz de la ecuación 4.56 tiene rango dos es invertible, por lo que se puede despejar $\dot{\eta}$

$$\dot{\eta} = \bar{M}(q)^{-1}\bar{B}(q)\tau \qquad \text{(Ecuación. 4.60)}$$

Discretizar la ecuación 4.60 mediante la aproximación de Euler se obtiene.

$$n(k+1) = n(k) + \Delta t \bar{M}(q)^{-1}\tau$$

$$n(k+1) - n(k) = \Delta t \bar{M}(q)^{-1}\tau$$

Por lo que el valor de τ se puede encontrar exactamente.

$$\tau = \frac{1}{\Delta t}\bar{M}(q)\Delta n(k) \qquad \text{(Ecuación. 4.61)}$$

Por lo que se puede aplicar un controlador PID sobre los dos motores como regulador lineal de la velocidad angular de cada motor.

4.16 Implementación de la estructura

La placa se envió a fábricar como prototipo con un acabado en *solder mask* en los lados *Top y Bottom* con xerografiado del *silk screen*, los puntos de conexión con acabado en plata para evitar la oxidación, así como corte y perforación CNC.

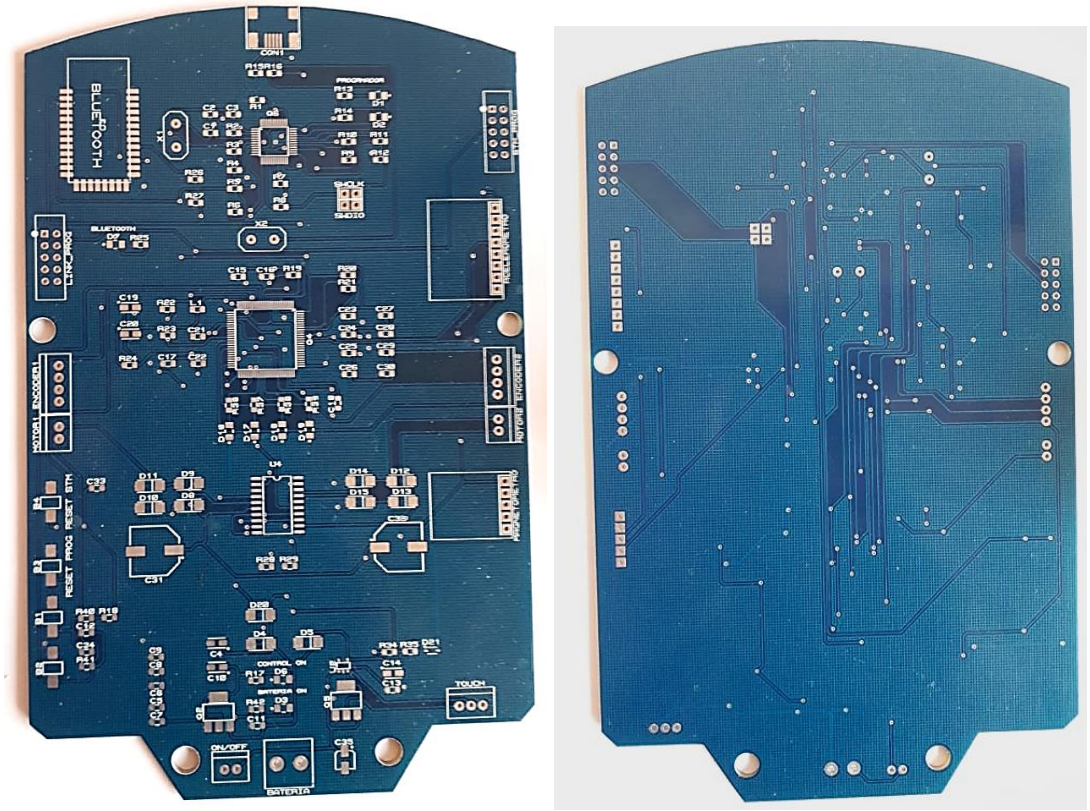


Figura. 4. 32 Placa maquilada con acabado profesional

La estructura impresa en 3D se ensambló uniéndola con pernos, se colocaron los elementos electrónicos motores y la tarjeta PCB con todos los elementos soldados para las pruebas de hardware. Se conectaron elementos adicionales mediante cables soldados a la placa o conectados a través de los jacks diseñados para este propósito.

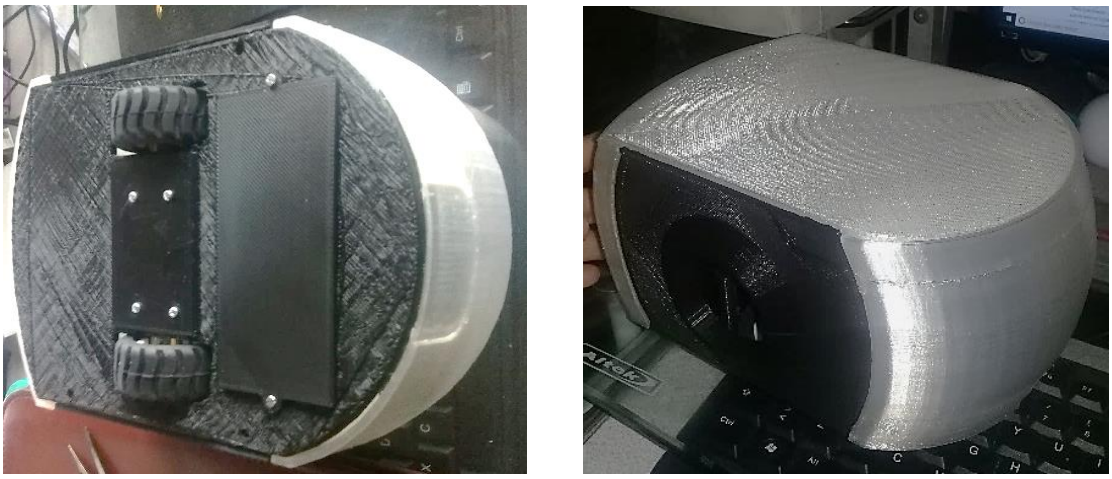


Figura. 4. 33 Armado de la carcasa del robot

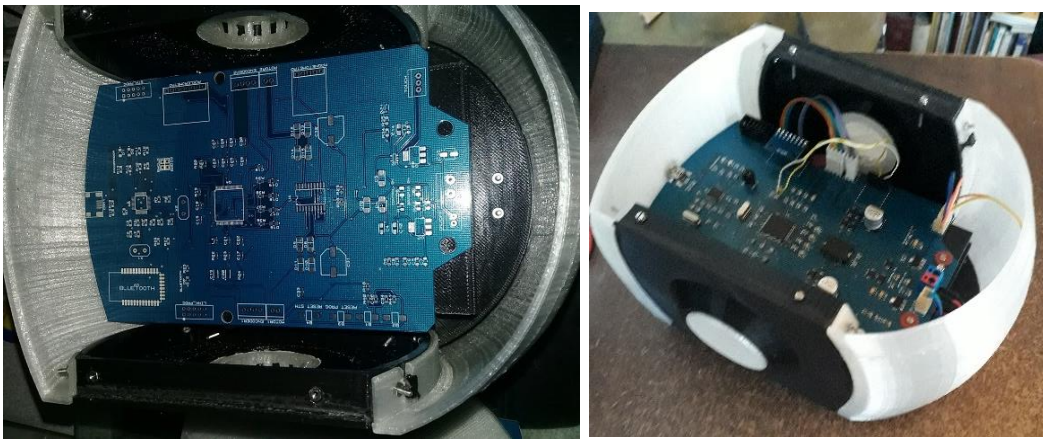


Figura. 4. 34 Visualización da la colocación de la placa

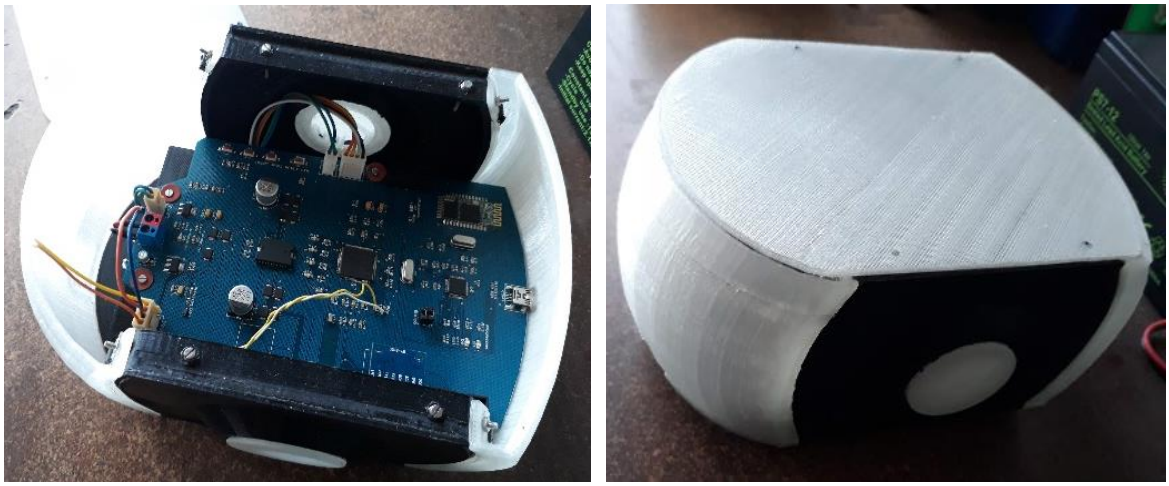


Figura. 4. 35 Vista del robot sin tapa superior y con tapa superior

Figura. 4. 36 Conexión y puesta de la batería

4.17 Desarrollo del controlador en el software de programación SIMULINK (MATLAB)

Se desarrolló el algoritmo de control en la ventana Simulink del programa MATLAB el cual permite elaborar toda la configuración mediante bloques de programación gráficos, que reducen el uso de extensas lineas de código, los comandos vienen intergrados en cada bloque, y solo se debe modificar ciertas variables y en otros casos se detallará la programación en lenguaje *"script .m"*, para que el sistema de simulink reconozca la tarjeta es necesario instalar las librerias *waijung blockset*, las cuales contienen todas las funciones utiles que permiten acoplar el hardware con el software.

En la figura 4.37 se define en escencia el bloque principal del controlador, las configuraciones que tendrán los diferentes tipos de comunicaciones utilizadas, los parámetros, registros utilizados, tiempo de reloj, *time out*, pines utilizados para la transmisión, etc.

Para iniciar la secuencia del programa se genera un pulso, conocido como escalón unitario, el cual debe pasar por una etapa de comparación de dato, ya que el siguiente bloque pueda que maneje el mismo u otro tipo de dato, los que podrian ser: double, single, binary entre otros, una vez que los datos sean compatibles se ejecuta la configuración de la IMU, para despues empezar a correr el algoritmo del controlador.

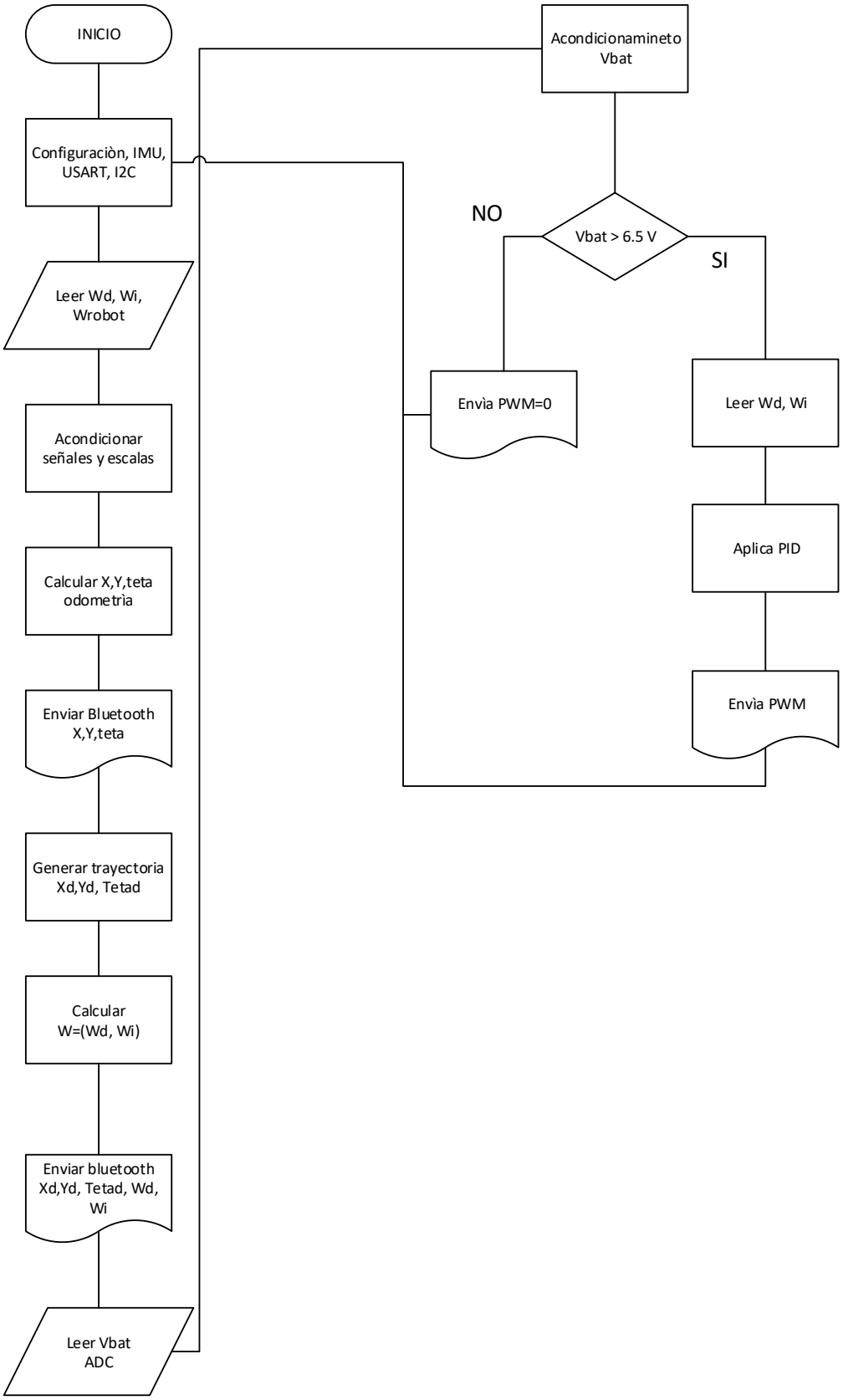


Figura. 4. 37 Flujograma del desarrollo del algoritmo en el programa MATLAB

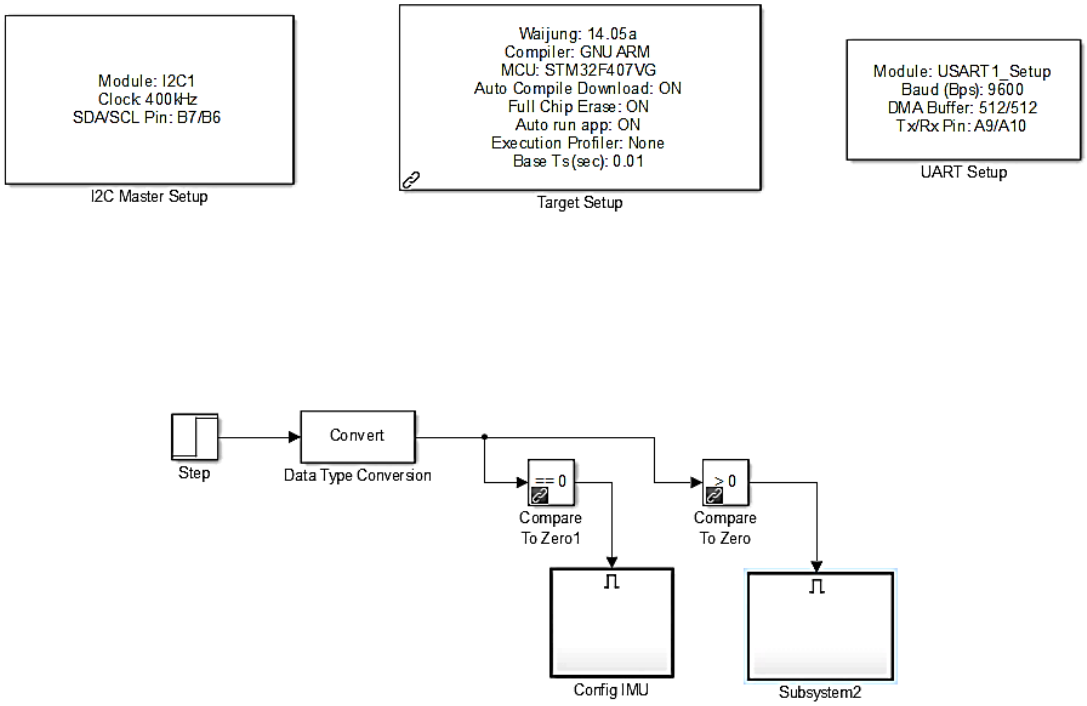


Figura. 4. 38 Programación algoritmo de control

4.17.1 Configuración IMU

En esta etapa se definen los valores de arranque de la IMU, tiempo de espera *(sleep)* y fondo de escala que permita muestrear la máxima velocidad angular de rotación de las ruedas en grados por segundo, dicha velocidad se determina por la siguiente formula:

$$W_{max} = 1100\ rpm\ x\frac{2\pi}{60} \approx 115\ rad/s$$

$$\dot{\theta}_{max} = \frac{R}{2l}\ (W_{dmax} - (W_{imin})) = \frac{R}{l}W_{max}$$

$$\dot{\theta}_{max} = \frac{4.2}{8.44} * 115 rad/s$$

$$\dot{\theta}_{max} = 57.22\frac{rad}{s} \approx 3278.8\ °/s$$

Se puede observar el registro de escalas en el datasheet del giroscopo para definir cual será la mejor, y se llegue a cumplir con este parámetro calculado.

Tabla. 4. 1 Configuraciones giróscopo (Invensense, 2013)

FS_SEL	Full Scale Range
0	± 250 °/s
1	± 500 °/s
2	± 1000 °/s
3	± 2000 °/s

Según la tabla 4.1 se puede observar que para la condición dada se debe utilizar el fondo de escala 3 para que me pueda medir el giróscopo el máximo grado de curvatura que el robot da en un segundo.

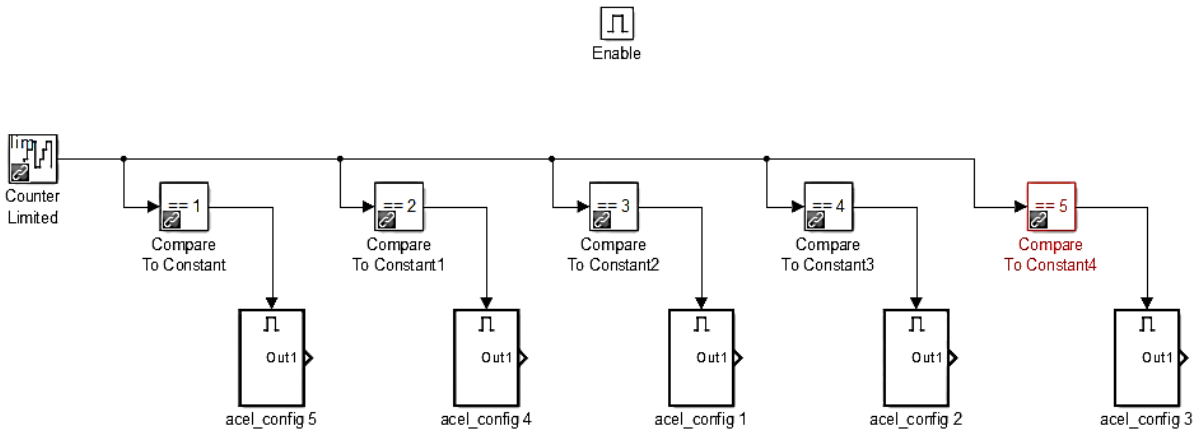


Figura. 4. 39 Diagrama de bloques configuración IMU

En la figura 4.39 se determina el valor a tomar mediante la comparación de la señal de entrada, mediante esta se determina un valor de fondo de escala por medio de una comparación, se considera los registros de lectura y escritura en hexadecimal.

Tabla. 4. 2 Registro 27 lectura y escritura (Invensense, 2013)

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

4.17.2 Controlador

Las ecuaciones resultantes desarrolladas para el algoritmo de control del seguimiento de trayectoria se configurarán en la ventana Simulink del programa Matlab mediante bloques, la estructura que se realiza es denominada diagrama de control del sistema, dentro

de los bloques se configuró subsistemas, que permitirán controlar otros elementos del hardware que están fuera de las ecuaciones del sistema.

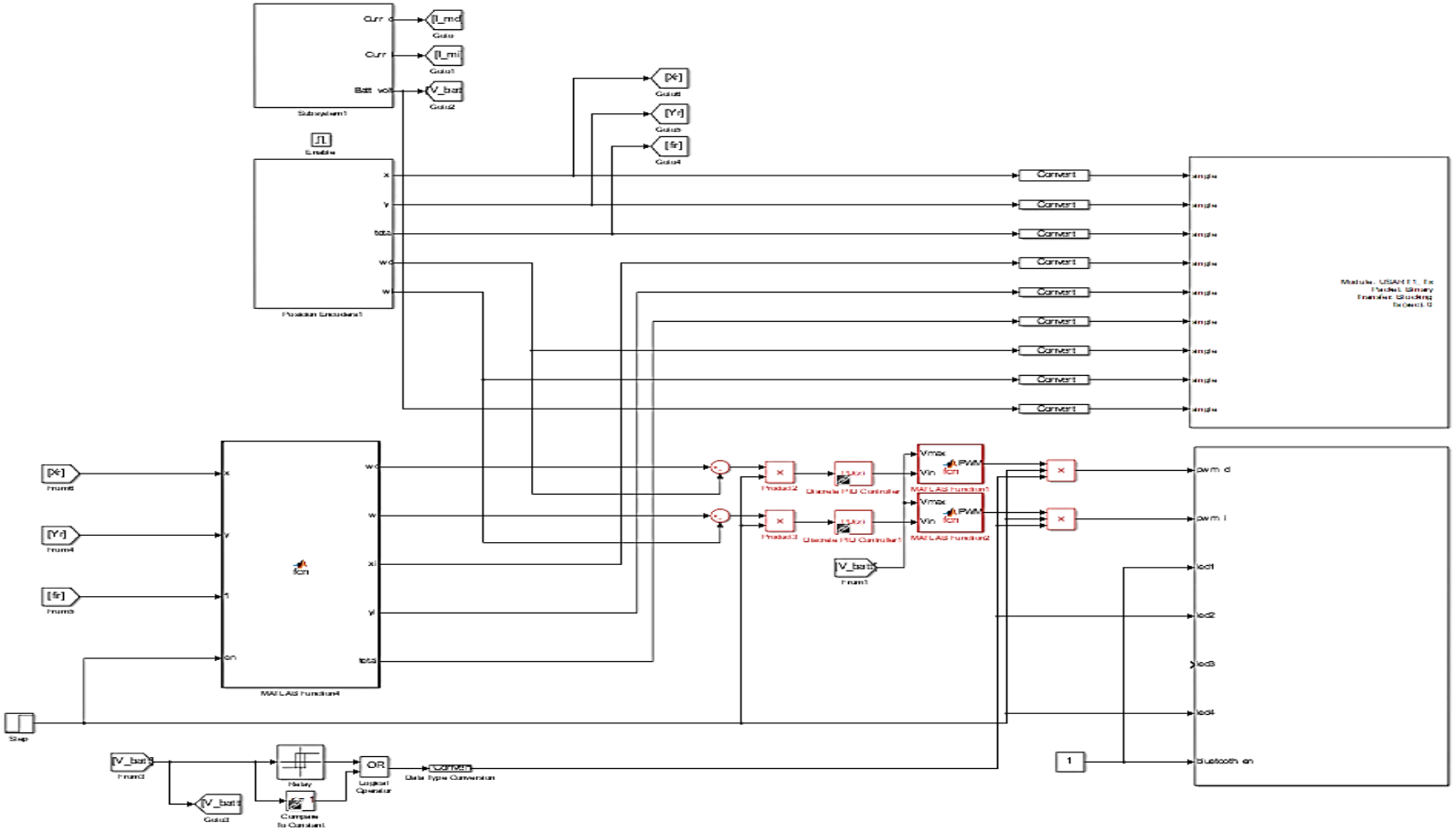
El subsistema se diseñó en base al diagrama de bloques de control automático, y cuenta con las siguientes etapas:

- Controlador no lineal de posición.
- Regulador ADC
- Odometría
- PWM
- Modulo USART
- Controlador PID
- Sensor de Voltaje
- Coordenadas de posición
- Sub-bloques de control

Todos estos bloques se describen a detalle en la siguiente sección, el funcionamiento, las conexiones, y la tarea que cumple.

Esta etapa se encarga del desplazamiento del robot que permite determinar las trayectorias a seguir y realimentar la información con la posición actual, de tal manera que se analiza la velocidad angular de cada rueda para que cumpla la trayectoria dada, en esta etapa ingresan los valores de las señales de los encoder's, y giróscopo que son los elementos electrónicos que me permiten determinar la posición del robot.

El diagrama de la figura 4.40 se puede determinar como un diagrama de bloques de todo el sistema, ya que es el controlador quien comanda el desplazamiento del robot, y en donde se puede verificar la retroalimentación del sistema, es todo el sistema principal de control de todo el proyecto.



Titulo:	CONTROLADOR PLATAFORMA ROBOTICA	
Nombre:	Martin Larco	
Fecha:	24 de Febrero del 2019	

Figura. 4. 40 Diagrama de control desarrollado en Simulink

- **Controlador no lineal de posición**

En el bloque del controlador ingresan las posiciones X, Y, θ las cuales son corregidas y a la salida se obtienen las velocidades angulares deseadas, estas serán las que controlen a los motores w_d, w_i y permitan el movimiento del robot a su vez envía las posiciones X, Y, θ hacia el Bluetooth para trasmitirlos a la PC y graficar la trayectoria del robot, con las posiciones actuales.

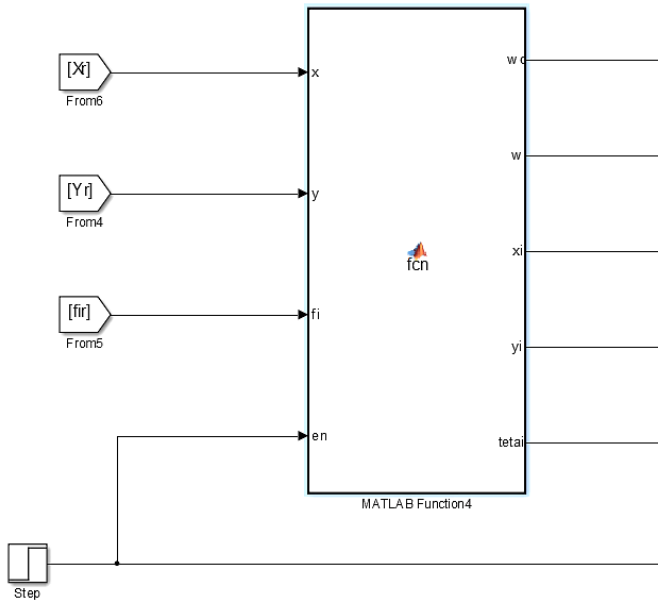


Figura. 4. 41 Controlador no lineal cinemático

Se adjuntan las líneas de programación elaboradas en la etapa del controlador, en esta etapa se define la trayectoria que seguirá el robot, se tiene pre-configuradas 5 trayectorias, un círculo, una parábola una trayectoria circular de un nodo y de dos nodos, ya que debido a que las funciones deben ser periódicas se definen como senos y cosenos, para parametrizar en un espacio o longitud dada.

Si se define una línea recta con la ecuación punto pendiente esta seguirá al infinito y no se podrá visualizar de manera correcta el funcionamiento del robot, la idea del controlador es definir una trayectoria en el espacio de trabajo.

Se deben manejar las funciones y variables persistentes para que estas se mantengan como variables globales al salir del *script*, aunque acabe de ejecutarse el bloque e inicie nuevamente el *script* estas variables mantengan su valor, a continuación, se detalla el programa de control que se configuró dentro del bloque.

```
function [wd,wi,xi,yi,tetai] = fcn(x,y,fi,en)

persistent cv Ts t Fid k Ydn1 Ydn Xdn1 Xdn Dx Dy ke r l Tf
persistent Fidn1 Dfi_c Fidn Dfi Xn Yn Fin Wd Wi S Kp Ki

if isempty (cv)
   ke=0.92; r=4.2/200; l=8.448/200;
   S=0; Kp=0.4; Ki=0.2;  %para traytectorias circular y con uno y dos nodo

   %S=0; Kp=0.15; Ki=0.02; %para linea recta

   %S=0; Kp=0.15; Ki=0.008; %para parabolica

   %Generacion de trayectorias
   cv=25.0; Ts=0.01; Tf=300; t=0;
   Xdn=0; Xdn1=0;
   Ydn=0; Ydn1=0;
   Fid=0;
   Fidn=0; Fidn1=0;
   k=0;
   Wd=0;
   Wi=0;
end


if(en==1)
   if(t<Tf)

      %Xdn1 = 0.4*cos(t/8); %trayectoria con un nodo componente x
      %Ydn1 = 0.4*sin(t/4); %trayectoria con un nodo componente y

      %Xdn1 = 0.4*cos(t/9); %trayectoria con dos nodo componente x
      %Ydn1 = 0.4*sin(t/3); %trayectoria con dos nodo componente y

      Xdn1 = 0.4*cos(t/4); %circulo componente x
      Ydn1 = 0.4*sin(t/4); %circulo componente y

      %Xdn1 = 0.4*cos(t/4); %recta componente x
      %Ydn1 = 0.4*cos(t/4); %recta componente y

      %Xdn1 = 0.4*sin(t/4); %parabola componente x
      %Ydn1 = -0.4+0.8*(sin(t/4))^2; %parabola componente y

      Xn=x;
      Yn=y;
      Fin=fi;

      Dx=Xdn1-ke*(Xdn-Xn)-Xn;

      Dy=Ydn1-ke*(Ydn-Yn)-Yn;

      Fid=atan2(Ydn1-Yn,Xdn1-Xn);


      if Fid<0
            Fid=2*pi+Fid;
```

```
        end

        Fidn1=Fid;

        Dfi_c = Fid-Fin;

        if abs(Dfi_c)>=pi
            if Fin>=pi
                Fin=-2*pi+Fin;
            end
            if Fid>=pi
                Fidn1=-2*pi+Fidn1;
            end
            if Fidn>pi
                Fidn=-2*pi+Fidn;
            end
        end

        %Dfi=Fidn1-kf*(Fidn-Fin)-Fin;

        Dfi=Kp*(Fidn1-Fin)+Ki*S;
        S=S+Ts*(Fidn1-Fin);
        Fidn=Fidn1;

        Wd=[1/(Ts*r)]*[Dx*cos(Fin)+Dy*sin(Fin)+l*Dfi];

        Wi=[1/(Ts*r)]*[Dx*cos(Fin)+Dy*sin(Fin)-l*Dfi];

        Xdn=Xdn1;
        Ydn=Ydn1;

        k=k+1;
        t=Ts*k;
    end
    if(t==Tf)
        Wd=0;
        Wi=0;
    end
    if(t>Tf)
        Wd=0;
        Wi=0;
    end
end

    wd=Wd;
    wi=Wi;
    %wd=3.1416;
    %wi=0;

    xi=Xdn1;
    yi=Ydn1;
    tetai=Fidn1;
```

- **Regulador ADC**

Este subsistema permite recolectar los valores del voltaje de la batería, y corriente de los motores, ya que al utilizar baterías LIPO estas no se pueden descargar hasta un cierto voltaje detallado en la misma, se utiliza una batería LIPO de 7.4 V 2200mAh y no puede bajar su voltaje a menos de 6.6 V para evitar daños materiales y humanos.

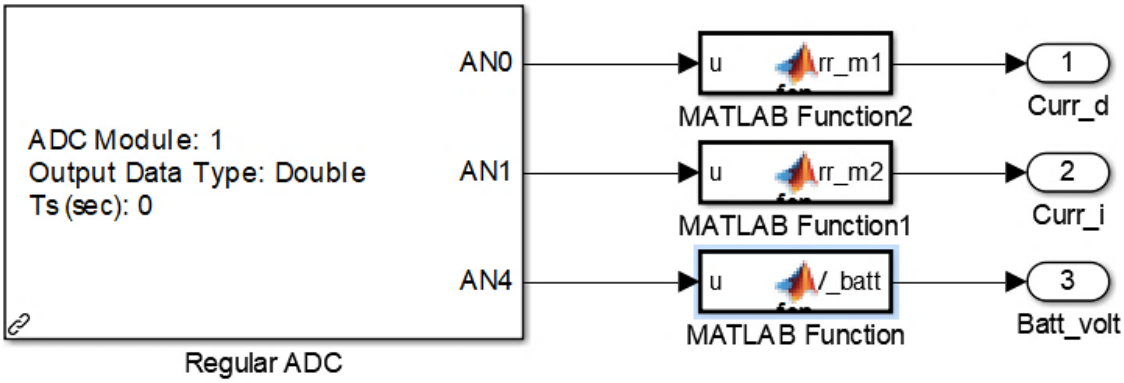


Figura. 4. 42 Diagrama del regulador ADC

Dentro de este subsistema se maneja una función que me permite considerar los parámetros para el análisis máximo del voltaje:

$$\text{function V\_batt} = \text{fcn(u)}$$

$$\text{V\_batt} = (134/33)*3.3*(\text{u}/4096);$$

La función relaciona la resistencia utilizada para el cálculo de la batería, con relación a los 12 bits que utiliza el conversor ADC, que permita inutilizar los motores y el Bluetooth en caso de que el voltaje decaiga por debajo de 6.8 V y así evitar los consumos más altos de corriente del sistema.

- **Odometría**

En este subsistema se encuentran las ecuaciones de odometría, las cuales empiezan con las lecturas de las velocidades angulares calculadas a partir de los pulsos enviados desde los encoder de los motores y de la velocidad angular del giroscopio enviada en dos bytes a través del módulo de comunicación I2C, para obtener de esta manera las posiciones relativas del sistema mediante las ecuaciones de odometría.

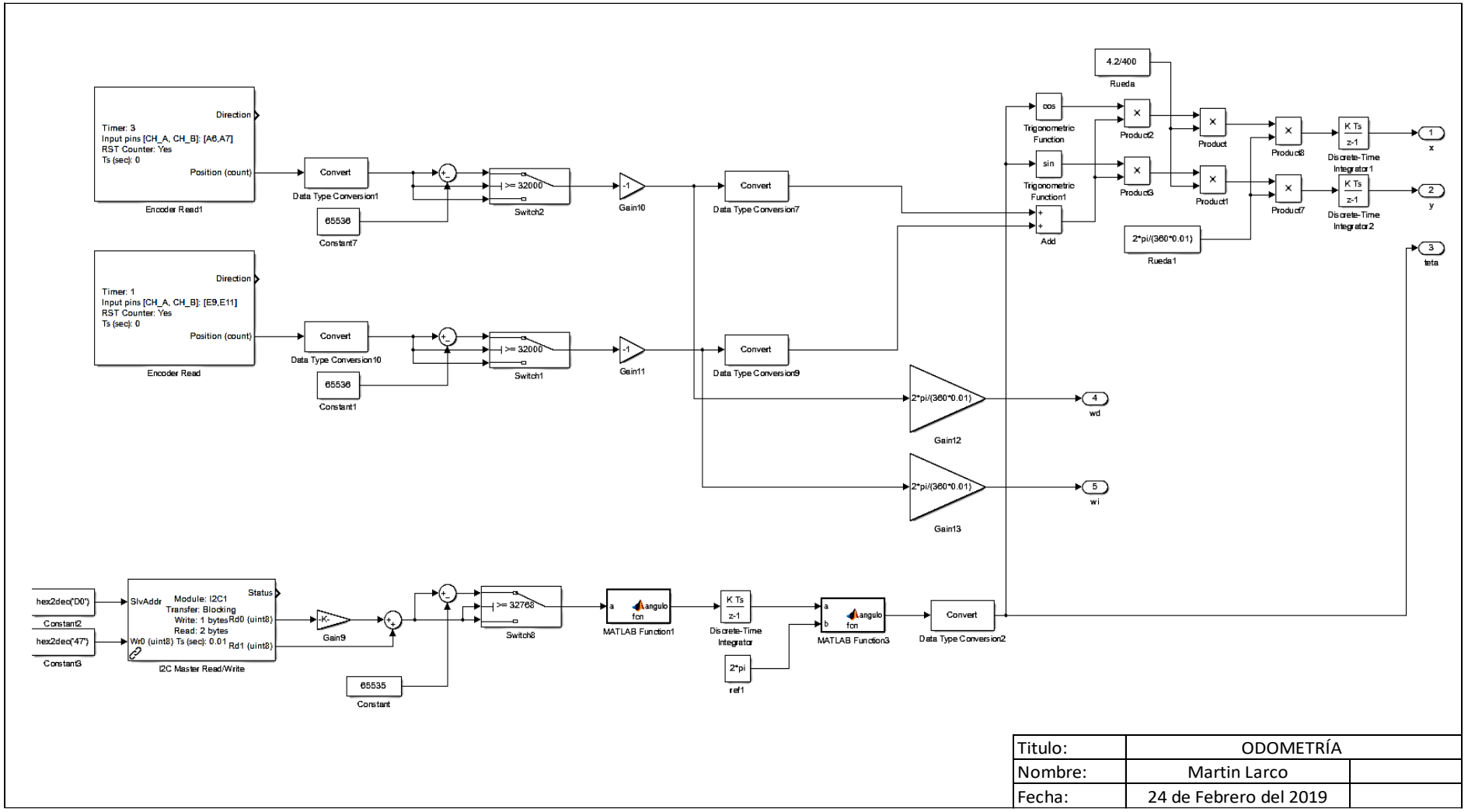


Figura. 4. 43 Subsistema del posicionamiento del robot

- **PWM**

En este subsistema se realiza el reconocimiento de la dirección de giro del motor y envía la PWM de cada motor con una magnitud absoluta y una señal digital determina la dirección física de giro, esto activa los buffers del driver del motor.

Dentro de este módulo también se encuentran las salidas digitales que van a los leds de visualización, así como la señal de entrada al *enable* del módulo Bluetooth.

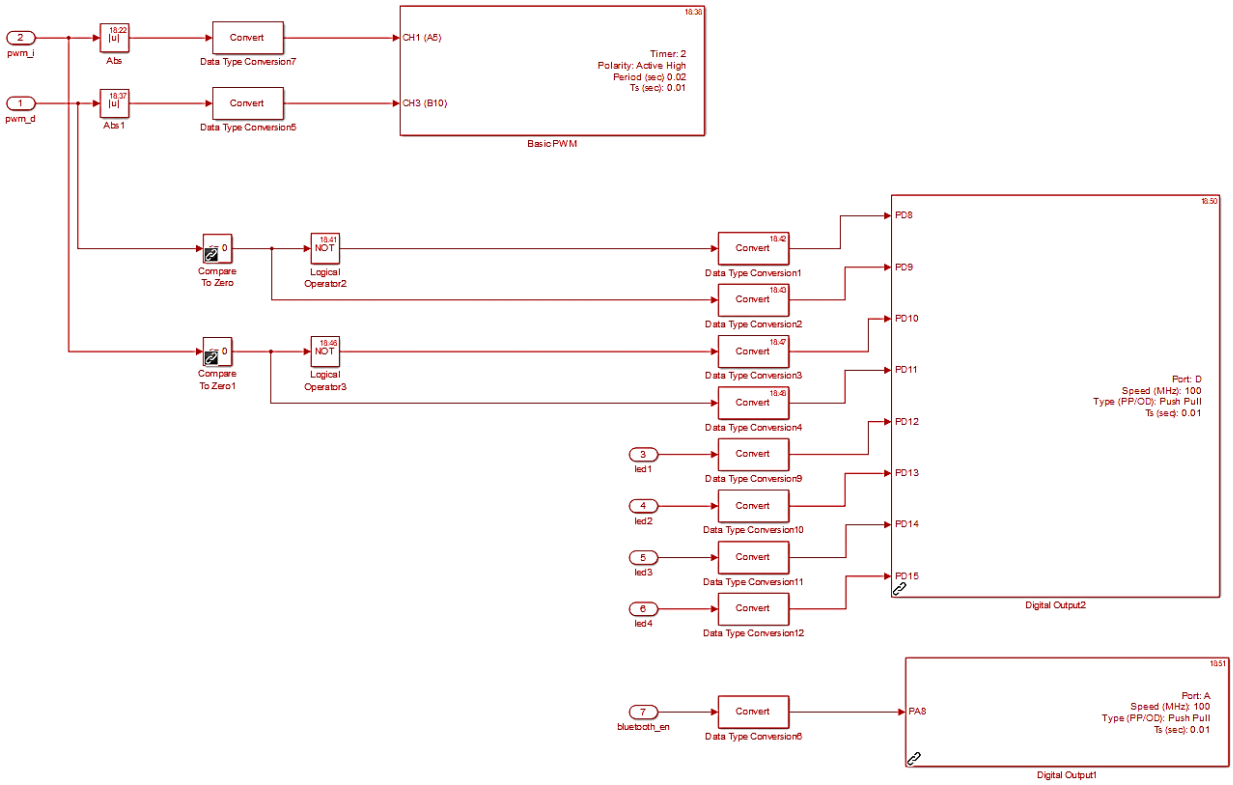


Figura. 4. 44 Estructura del algoritmo de control de motores y salidas digitales.

- **Módulo USART**

El protocolo de comunicación serial, es utilizado para transmitir los datos de las posiciones reales y deseadas, más el voltaje de la batería, desde el microcontrolador hacia el módulo Bluetooth el cual enviará todos estos datos hacia la PC, ya que esta procesa dichos datos para representarlos en gráficas de posición vs tiempo y velocidad vs tiempo, lo cual se puede visualizar en las gráficas que se desarrollaron en el GUI.

Todos los datos ingresados a este módulo son pasados a través de algunos conversores, los cuales permiten que toda la información sea legible para el módulo, esto también permite

deslindar el análisis de qué tipo de dato es el que llega, ya sea *single*, binario, ASCII, *double*, entre otros.

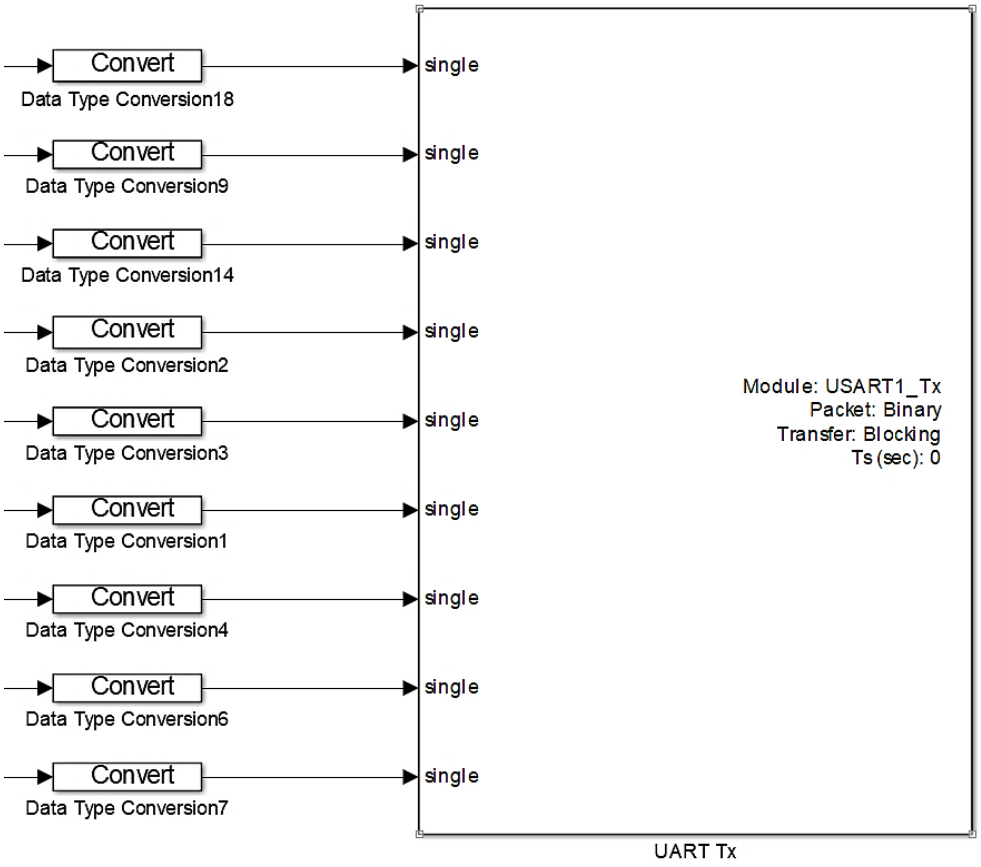


Figura. 4. 45 Módulo USART

4.18 Desarrollo del GUI

Se elaboró el GUI en el programa MATLAB el cual me permite visualizar la posición del robot en los diferentes ejes X, Y, θ, la gráfica de la trayectoria, las velocidades angulares de cada una de las ruedas, datos de la comunicación Bluetooth, tiempo y voltaje de la batería, estas son las variables utilizadas a lo largo del desarrollo del algoritmo, ya que estas son las más importantes por tal motivo deben ser las variables visibles.

También se puede obtener la estadística del robot como las correlaciones, desviación, y el error determinado. El envío de los datos es vía Bluetooth, de forma inalámbrica y en el momento de funcionar el robot, ya que la traza de gráfica de la trayectoria es en tiempo real.

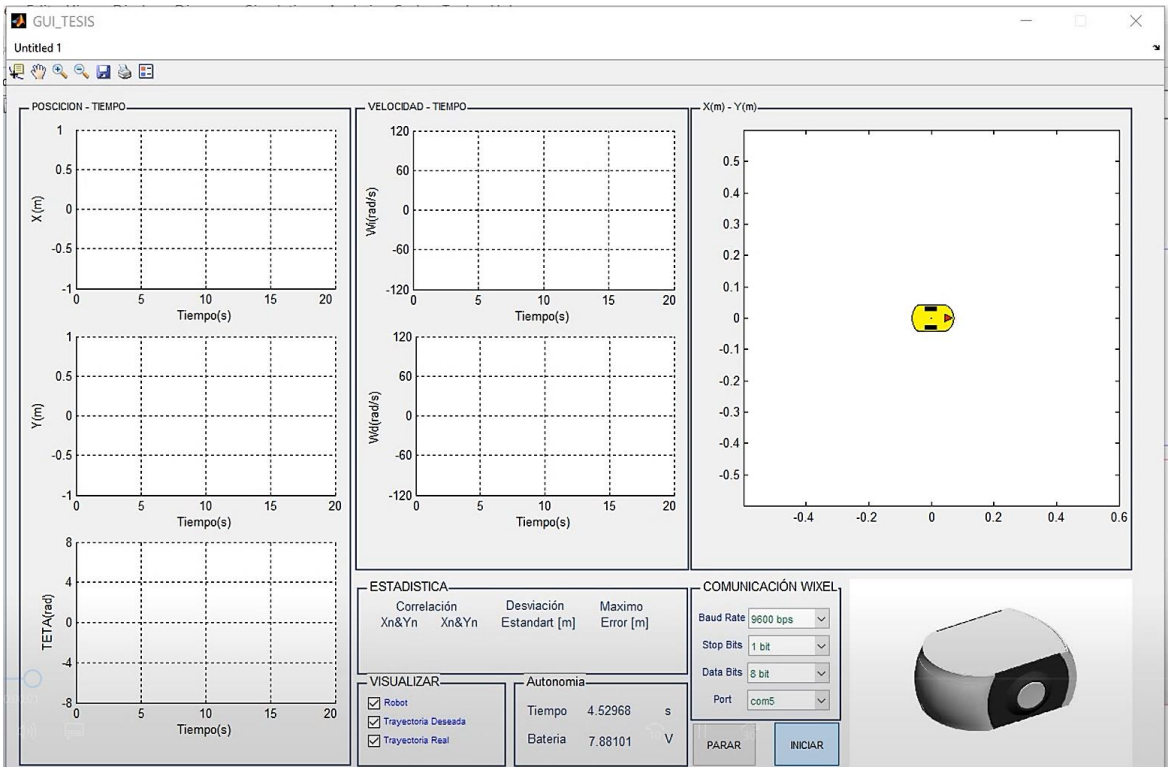


Figura. 4. 46 GUI desarrollado en MATLAB

En el anexo 2 se adjunta todo el programa de la configuración del GUI de cada uno de los recuadros, las variables utilizadas, para visualizar las gráficas del desplazamiento de la plataforma robótica. A su vez se determina la escala utilizada para cada uno de los gráficos, de tal manera que sea visible y entendible.

4.19 Pruebas y resultados

En esta sección se detalla las pruebas realizadas del seguimiento de trayectoria del robot móvil, con diversas trayectorias para ver el comportamiento del robot, si acumula error o es lo más cercano a la trayectoria determinada.

4.19.1 Pruebas de la plataforma móvil diferencial

Para las pruebas se debe tener una base totalmente lisa para que no haya perturbaciones del suelo que afecten al sistema o a la traslación del robot, por tales razones se utiliza una tabla en donde el robot se desplazará, las ecuaciones de la trayectoria deben ser acorde a las dimensiones de la tabla.

Figura. 4. 47 Pruebas de desplazamiento de la plataforma

Se observa la precisión que tiene la plataforma para seguir una trayectoria dada, que se denota en el GUI con color rojo la trayectoria deseada, y con color azul la trayectoria realizada por el robot.

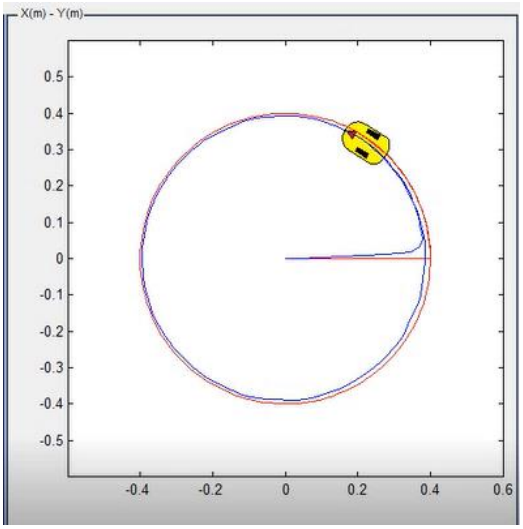


Figura. 4. 48 Gráfica de la trayectoria circular

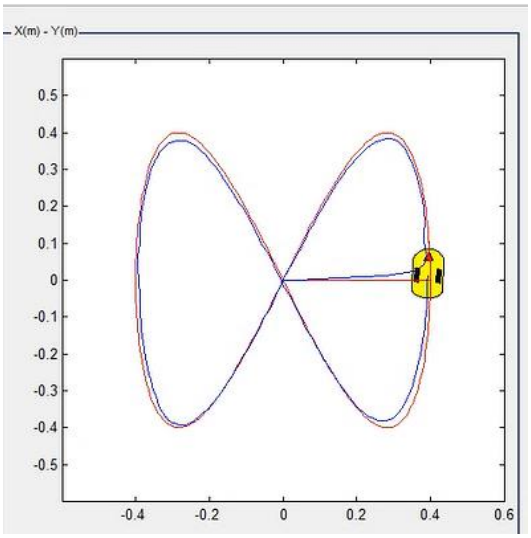


Figura. 4. 49 Gráfica de trayectoria de un lazo

Como se puede observar en las figuras 4.48 y 4.49 las líneas azul y rojo están casi solapadas, determinándose que el error de la trayectoria es pequeño, lo que define que el robot sigue la trayectoria demarcada con un valor que se mide de forma estadística respecto a todos los puntos de la trayectoria los cuales aparecen en el GUI como correlación la cual se encarga de medir la relación de linealidad de la trayectoria seguida respecto a la trayectoria deseada, este valor mientras más cercano a uno sea implica que los valores presentan una alta linealidad, mientras que los valores cercanos a cero implican que no existe relación entre los valores deseados y seguidos por el móvil.

La desviación estándar mide el valor de los promedios cuadráticos de la suma de error en las dos componentes de posición con lo cual se tiene un valor que contiene el 68% de las medidas de error dentro de toda la trayectoria.

EL error máximo indica el valor del error máximo medido entre la trayectoria deseada y la seguida por el robot en todos los puntos de la trayectoria.

Para obtener todas las gráficas configuradas en el GUI se debe detener el envío de datos, para que pueda procesar los datos obtenidos y graficar todas las variables de posición.

La trayectoria más exigente en cuanto a complejidad es la trayectoria elíptica de uno y dos nodos y se presenta a continuación

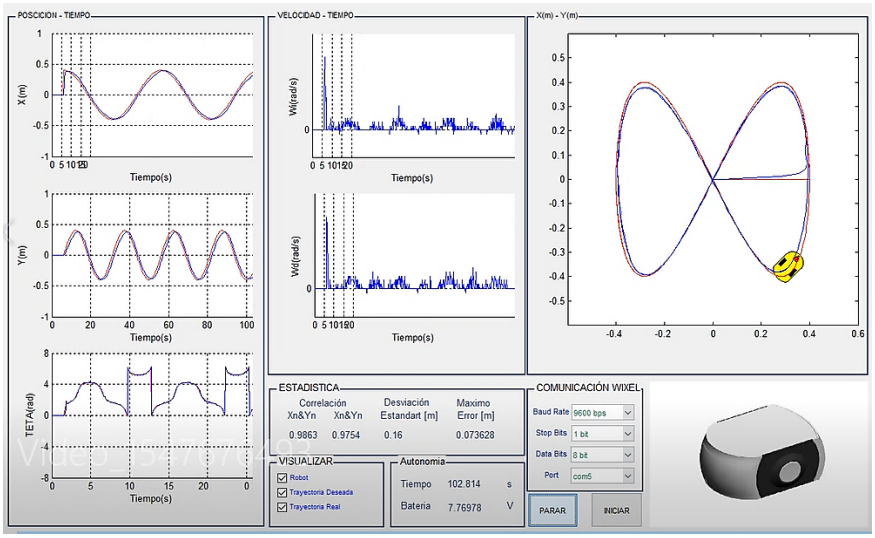


Figura. 4. 50 Resultados de la trayectoria elíptica con un nodo

Se puede observar que los valores de correlación en las dos componentes de posición son superior a 0.97 por lo que el controlador permite un seguimiento de la trayectoria de forma lineal y se ajusta muy bien a la trayectoria, en cuanto a la desviación estándar se tiene un valor de 16 cm el cual indica que el 68% de las medidas de la suma de error en posición de las variables X & Y es de 16cm, el cual es un valor bajo si se toma en cuenta que la suma de los ejes principales del robot es de 24.4 cm, por lo que la trayectoria siempre permanece dentro de los limites internos del robot, el valor máximo del error es de 7 cm el cual es bajo si se considera que la longitud del eje de rotación es 8.4cm, por lo que se puede concluir que el desempeño del controlador es satisfactorio, y se puede mejorar con un afinación de las constantes de control.

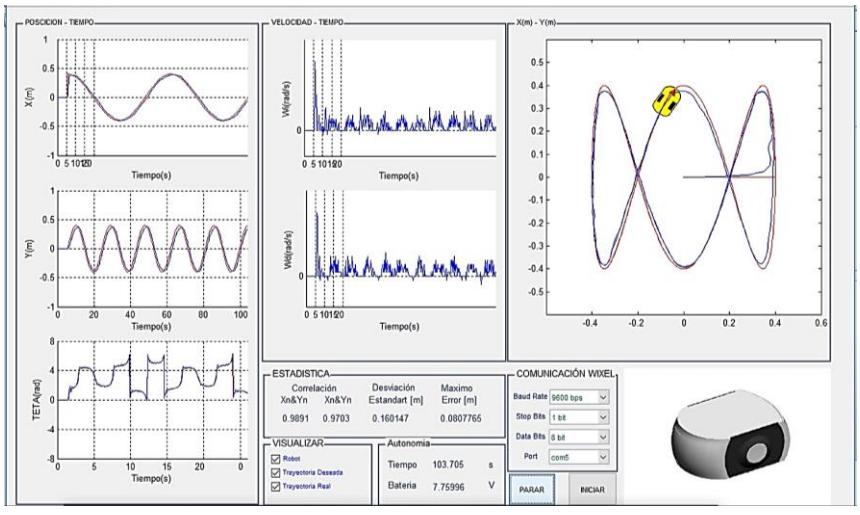


Figura. 4. 51 Resultados de la trayectoria elíptica con dos nodos

Se puede apreciar que los valores estadísticos de esta trayectoria son muy parecidos a las de la trayectoria anterior, con diferencias mínimas por lo que la conclusión respecto a estos es que el controlador tiene un desempeño satisfactorio con resultados de seguimiento sobresalientes esto si se considera la complejidad de la trayectoria que presenta aceleraciones variables.

4.19.2 Verificación de la corrección de la trayectoria

Al definir una trayectoria se lo hace en los ejes no inerciales, por lo tanto, el robot debe partir de su posición inicial 0,0,0 y debe llegar a la posición dada, desde el cual va a empezar a realizar el seguimiento de trayectoria, la posición de inicio de la trayectoria es diferente al de la posición con la que inicia el robot en el mismo eje de coordenadas.

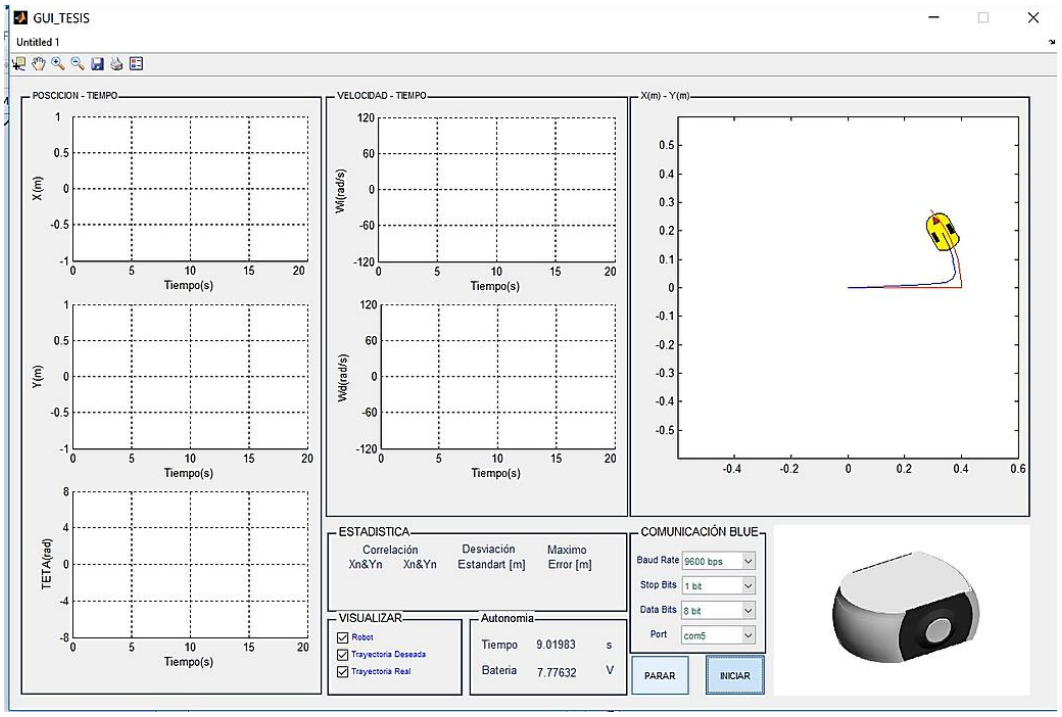


Figura. 4. 52 Corrección de la posición inicial

Como se puede observar en la figura 4.52 al iniciar el robot trata de corregir la trayectoria dada, en el menor tiempo posible, de esta manera se visualiza que el controlador configurado actúa de forma correcta.

La etapa de control PID se calibró con respecto a las pruebas generadas, para conseguir que la trayectoria seguida por el robot sea lo más aproximada posible a la realidad, que el error de convergencia sea el mínimo, en si el proyecto no trata sobre cómo resolver la etapa

de corrección de errores, pero se configuró tomando en cuenta una mínima proporcional del error adquirido, ya que debido al incremento a lo largo de un tiempo relativamente considerable se propone que el camino seguido por el robot sea correcto.

La carga de la batería es una de las variables que afecta la trayectoria del robot, debido a que se manejan variables considerando un valor relativo de la carga que es de 7.4V a 6.4V, pero debido a que al momento de cargar la LIPO después de haber sido descargada esta adquiere un voltaje más alto de carga de 8.2V esto es un error de fábrica que no se puede corregir con facilidad, ya que se debería modificar la programación en donde interactúa las variables de voltaje, esto puede llegar a ser engorroso y traer errores si se modifica de mala manera la configuración, este problema se corrige después de que la pila empieza a descargarse y llega a los valores nominales para los cuales el robot trabaja correctamente.

4.20 Análisis de resultados

Las pruebas realizadas a la plataforma móvil diferencial, programado el algoritmo de seguimiento de trayectoria, dio como resultado un error mínimo entre la trayectoria dada y la seguida por el robot, las desviaciones de la posición relativa del objeto en estudio es alrededor del 5% por lo cual la PID diseñado para este proyecto, a pesar de que no fue el objeto de estudio, corrige de forma satisfactoria el error de cambio de dirección con una corrección del 95%, después de un tiempo dado alrededor de 10 min constantes de recorrido se empieza a acumular error que produce un cambio de trayectoria, esto es debido al diseño del controlador proporcional, integral y derivativo, que no fue creado con el fin de mantener un trabajo constante si no con el de poder visualizar el funcionamiento del algoritmo.

En el capítulo 4.3 se analizó el consumo de energía que el robot genera en el trabajo normal de la plataforma dando como resultado un tiempo de autonomía de 40 min, después de realizadas las pruebas físicas del robot se pudo evidenciar que la duración de la plataforma es mucho más larga, esto debido a que en cada prueba de trayectoria la duración es de alrededor de 2min para visualizar las gráficas de las trayectorias seguidas, por tal razón el consumo aumenta, a su vez en los cálculos se consideró un consumo promedio de trabajo del robot a carga constante y en la vida real el robot al dar una curva baja su velocidad que implica que consume menos corriente y esto es menos descarga a la batería lo que aumenta el tiempo de autonomía.

CONCLUSIONES

Al finalizar el proyecto se logró completar todos los objetivos y alcances del mismo de forma satisfactoria, de tal manera que se llegó al objetivo general de diseñar e implementar una plataforma móvil diferencial para el seguimiento de trayectoria, a través de un algoritmo de control no lineal, mediante el uso de las ecuaciones de Euler-Lagrange

Se desarrolló el algoritmo de control para el seguimiento de la trayectoria del robot móvil diferencial mediante las etapas de control de posición acopladas en todas sus coordenadas generalizadas y el controlador dinámico mediante la realimentación de los estados de velocidad angular cuya función multivariable se obtuvo mediante el modelado de Euler-Lagrange.

Debido a que la matriz de acoplamiento cinemático tiene rango igual a 2, es decir que una de sus filas es linealmente dependientes de las otras razón por lo cual no es posible obtener una solución exacta al vector de velocidades angulares deseadas de los motores, debido a esto es necesario recurrir al concepto de optimización multivariable a través de las ecuaciones normales, y así se obtiene un vector de control que lleva a los estados al punto que minimiza la distancia entre el valor deseado de la posición y el valor actual, esto permite acoplar los estados del sistema.

Mediante la condición proporcionada por el espacio columna de la matriz de acoplamiento cinemático se obtiene la restricción que permite obtener el valor del ángulo deseado, al cual el valor de la posición angular debe tender a través del controlador, con esto se consigue obligar al sistema a tener una solución exacta a la cual permite linealizar los estados por realimentación a la convergencia del vector posición con la trayectoria deseada.

Debido a que las ecuaciones dinámicas del sistema contienen una matriz coriolis dependiente de la velocidad angular del sistema se dice que la ecuación es no lineal, sin embargo esta matriz es proporcional a la distancia entre el centro de masa y el eje de rotación del sistema, por lo que dada las características de construcción del móvil se puede minimizar esta distancia de forma que la contribución no lineal de esta matriz prácticamente se anule, esto permite ocupar un controlador proporcional derivativo en un lazo menor de control que permita llevar al sistema dinámico a las velocidades angulares de los motores proporcionados por el controlador de posición.

Se desarrolló la plataforma móvil diferencial de forma que se acerque a los parámetros y a las condiciones deseadas que se analizaron en el desarrollo del control, con un centro de masas cercano al centro geométrico donde está el eje de giro de las ruedas, la plataforma se desarrolló enteramente desde el software de diseño 3d STUDIO MAX y se imprimió en 3D para su aplicación práctica.

Se desarrolló el hardware necesario para las pruebas del algoritmo todo basado en el procesador del microcontrolador STM32F407 que cumple con los objetivos planteados y tomar en cuenta las recomendaciones del fabricante en cuanto al desarrollo de los filtros y protecciones necesarias para el sistema.

Se diseñó y elaboró la PCB con todo el hardware requerido para la programación USB desde el computador mediante el software Matlab a través de las librerías waijung, así como el hardware requerido para la comunicación, el hardware de adquisición de datos para la odometría y el control de los motores DC.

Se logró implementar los algoritmos requeridos para el controlador mediante el software Simulink, además se desarrollaron todos los bloques para adquisición odometría y envió de datos inalámbricos, los cuales se compilaron de forma satisfactoria y sin errores, además se desarrolló la interface de usuario en un GUI de MATLAB que permite visualizar los datos enviados por el microcontrolador, así como el análisis estadístico de la trayectoria que permite obtener el desempeño del controlador.

Se desarrollaron las pruebas para las trayectorias parametrizadas circular elíptica con uno y dos nodos, línea recta y parabólica con lo cual se obtuvo resultados satisfactorios con valores de correlación de 0.95, los cuales indican que la linealidad entre la trayectoria deseada y la seguida por el móvil es alta, lo que corrobora el buen desempeño del controlador.

RECOMENDACIONES

Debido a la relación inercia y torque de los motores, se tiene un sistema que presenta un alta dinámica, por lo que las velocidades de referencia son relativamente bajas con respecto a las velocidades que podrán generar los motores, por lo que se recomienda el uso de un controlador predictivo para la etapa dinámica del controlador, lo cual permitirá incrementar las velocidades del sistema sin que existan oscilaciones que produzcan inestabilidades en el sistema.

Debido a que el controlador de posición del sistema presenta cuatro constantes a ser calibradas se recomienda en virtud de mejorar la respuesta del sistema realizar la aproximación practica mediante métodos de validación estadística que permitan obtener los valores de la matriz de inercia y coriolis reales del sistema y de esta forma plantear un método de control basado en un método de realimentación de estados cuyas constantes se pueden obtener mediante simulación, lo cual ahorra tiempo de calibración y permite obtener una mejor respuesta.

Ya que el robot ocupa para su alimentación una batería de litio polímero para el diseño del robot se debe considerar ampliar el volumen del almacenamiento de la batería y la carcasa y añadir los drivers de control de la corriente de carga y consumo de la batería, de esta forma integrar en el robot el sistema de carga de baterías y de gestión de optimización de los recursos energéticos, que no se trataron en este proyecto por la alta densidad de temas que se trata en la misma.

Dado que la plataforma tiene como objetivo el seguimiento de trayectoria no se desarrolló el hardware que permita hacer detección de objetos con lo cual se recomienda continuar el estudio de algoritmos de evasión de obstáculos con seguimiento de trayectorias, de tal manera que se empiece por el diseño de hardware acoplado al puerto de comunicación I2C de la tarjeta y añadir los algoritmos de control necesarios.

Se recomienda en virtud de la comprensión total de este trabajo realizar la lectura de los textos de la bibliografía para total comprensión de los temas de ámbito teórico que no se pudieron tratar en este proyecto, debido a la gran cantidad de materia existente sobre los temas de control y que son necesarios para entender el proyecto a fin de editarlo y mejorar los algoritmos que aquí se han tratado.

BIBLIOGRAFÍA

Aguilera Hérnandez, M., Bautista, M. A., & Iruegas, J. (2007). *Diseño y control de robots móviles.* Nuevo Laredo.

Arechavaleta, G. (2010). Optimización de trayectorias para sistemas sujetos a restricciones no holónomicas. *Computación y Sistemas*, 1-3.

Casado Fernández, C. (2005). *MATLAB.* Servicios Informaticos U.C.M.

Corcuera, P. (2015). *Gráficos 3D en MATLAB*. Obtenido de Personales.unican.es: http://personales.unican.es/corcuerp/Matlab_Simulink/Slides/Matlab_graficos3D.pdf

Cuánticos, C. (20 de 11 de 2011). *Robótica: Estimación de posición por odometría*. Obtenido de https://cuentos-cuanticos.com/2011/12/15/robotica-estimacion-de-posicion-por-odometria/

Dhaouadi y Hatab. (2013). *Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework.* . Obtenido de https://www.omicsonline.org/open-access/dynamic-modelling-of-differentialdrive-mobile-robots-using-lagrange-and-newtoneuler-methodologies-a-unified-framework-2168-9695.1000107.pdf

González, A. J. (06 de Septiembre de 2013). *Bluetooth.* Obtenido de profesores.elo: http://profesores.elo.utfsm.cl/~agv/elo322/1s13/project/reports/Bluetooth.pdf

Hilario, J. (2016). *Tipos de movimiento de un robot*. Obtenido de prezi.com: https://prezi.com/etbqwsbtd_xp/tipos-de-movimiento-de-un-robot/

Ingeniería Mecafenix. (30 de 09 de 2017). *Encoder*. Obtenido de https://www.ingmecafenix.com/automatizacion/encoder/

Ingeniería Técnica Industrial. (2008). *ESTABILIDAD EN SISTEMAS DE ECUACIONES.* Ampliación de Matemáticas.

Invensense, I. (18 de 09 de 2013). *MPU-9150*. Obtenido de https://www.invensense.com/wp-content/uploads/2015/02/MPU-9150-Datasheet.pdf

JSumo. (2017). *Modulo de recepcion serial HC-06*. Obtenido de https://www.jsumo.com/hc-06-bluetooth-module-serial-receiver-module

Manual, A. (11 de 03 de 2012). *Modelo matemático de un motor de corriente*. Obtenido de http://www.lajpe.org/mar12/25_LAJPE_611_Manuel_Alvarez_preprint_corr_f.pdf

Miprofe. (2014). *Mínimos cuadrados*. Obtenido de MiProfe.com: https://miprofe.com/minimos-cuadrados/

monografias.com. (1 de Marzo de 2013). *Principales tipos de investigación.* Obtenido de https://www.monografias.com/trabajos58/principales-tipos-investigacion/principales-tipos-investigacion.shtml

Navarro Merino, F. (2017). *Aplicaciones de la construcción 3D: Odometría visual e integración con la realidad irtual.* Madrid: Universidad Politécnica de Madrid.

Navarro Merino, F. (2017). *Aplicaciones de la reconstrucción 3D: Odometría visual e integración con la realidad virutal.* Madrid: Universidad Politecnica de Madrid.

Olmo, M., & Nave, R. (2002). *Giroscopio.* Obtenido de http://hyperphysics.phy-astr.gsu.edu/hbasees/gyr.html

Perez, K., & Escobar, C. (2008). *Ingeniería de Sistemas I.* Obtenido de http://virtual.usalesiana.edu.bo/web/contenido/dossier/22011/700.pdf

PicClick. (2018). *1pcs STM32F407VET6 STM32F407 STM32LQFP100 32-bit ARM*. Obtenido de PicClick. (2018). 1pcs STM32F407VET6 STM32F407 STM32 LQFP100 32-bit ARM. Disponible en: https://picclick.com/1pcs-STM32F407VET6-STM32F407-STM32-LQFP100-32-bit-ARM-MCU-321433106563.html

POLOLU. (2001). *30:1 Micro Metal Gearmotor HPCB 6V*. Obtenido de https://www.pololu.com/product/3062

Poznyak, A. S. (2005). *Modelado Matemático.* New York.

Puntoflotante.net. (2013). *FUNCIONAMIENTO DEL ENCODER CUADRATURA DE EFECTO HALL PARA MEDICIÓN DE VELOCIDAD DE GIRO*. Obtenido de Puntoflotante.net. (2013). FUNCIONAMIENTO DEL ENCODER CUADRATURA DE EFECTO HALL PARA MEDICION DE VELOCIDAhttps://www.puntoflotante.net/FUNCIONAMIENTO-ENCODER-CUADRATURA-EFECTO-HALL.htm

Robots. (07 de 03 de 2014). *¿Si fueras un robotmovil, como te ubicarías espacialmente?* Obtenido de https://razek07.wordpress.com/2014/03/07/si-fueras-un-robot-movil-como-te-ubicarias-espacialmente/

Rodriguez, H. (2016). *Vista de panificación de Movimiento Mediante Campo de Potencial con Restricciones Dinámicas para Robots Móviles*. Obtenido de Revistas.utp.ac.pa: Rodriguez, H. (2016). Vista de Planificación de Movimiento Mediante Campo de Potencial con Restricciones Dinhttp://revistas.utp.ac.pa/index.php/id-tecnologico/article/view/109/html

Rodriguez-Alfaro, L., & Alorta-Garcia, E. (2014). De la representación de sistemas Euler-Lagrange. *NOVA SCIENTIA*, 4-12.

Sanz Valero, J. P. (2006). *Introducción a la robótica inteligente.* Dep. Legal CS-369-2006.

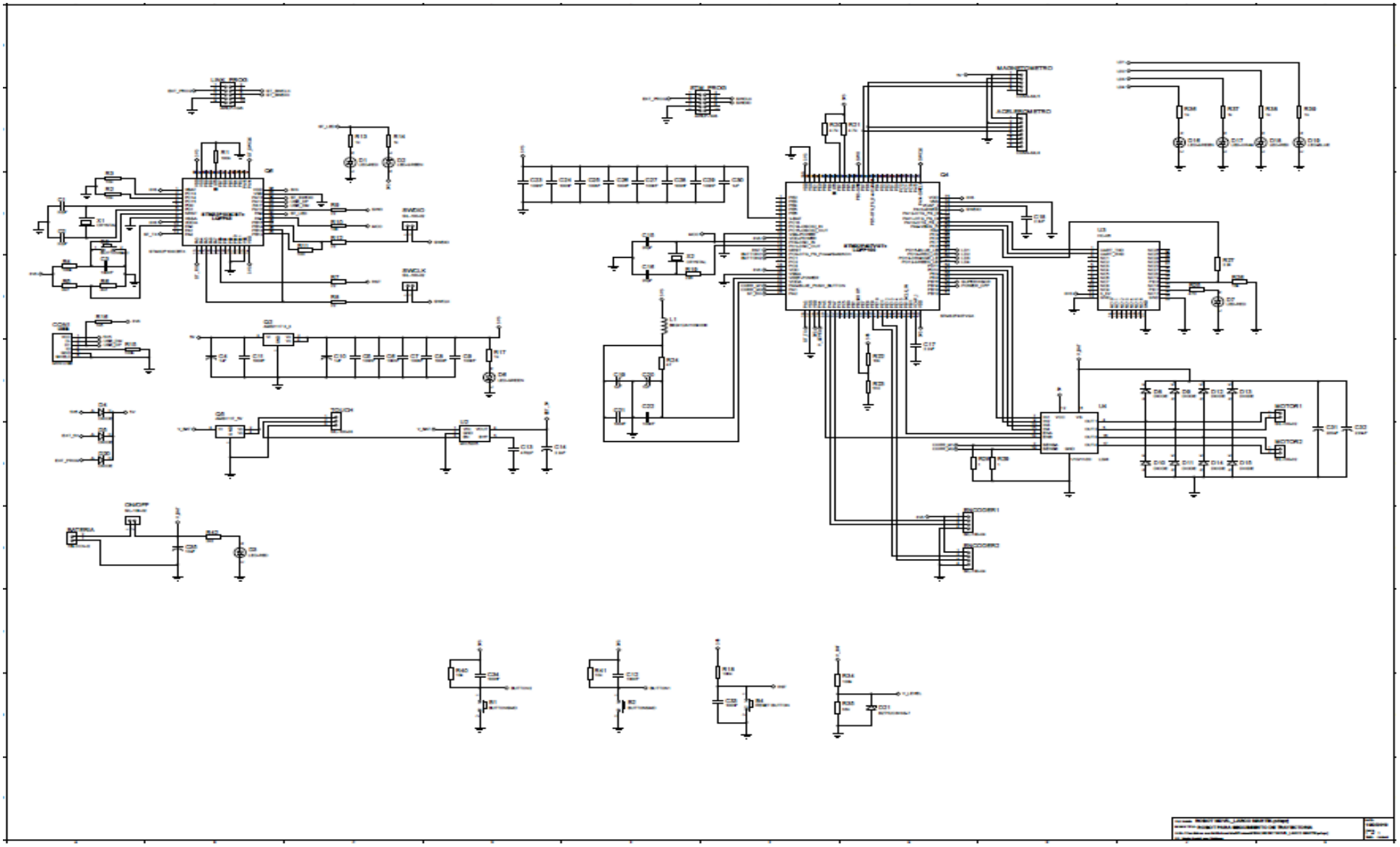
Sierra, P. (Enero de 2012). *Investigación.* Obtenido de Tipos más usuales de investigación: https://www.uaeh.edu.mx/docencia/P_Presentaciones/prepa3/tipos_investigacion.pdf

Stmicroelectronics. (2000). *L298P*. Obtenido de https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

Stmicroelectronics. (09 de 11 de 2012). *STM32F4 DISCOVERY*. Obtenido de http://www.disca.upv.es/aperles/arm_cortex_m3/curset/guia_iniciacion_STM32F4_discovery.pdf

Wordpress. (2017). *Practicas repaso arduino*. Obtenido de Aprendiendo arduino: https://aprendiendoarduino.wordpress.com/tag/debounce/

ANEXOS



ANEXO 2. PROGRAMACIÓN DEL GUI

```
function varargout = GUI_TESIS(varargin)
% GUI_TESIS MATLAB code for GUI_TESIS.fig
%      GUI_TESIS, by itself, creates a new GUI_TESIS or raises the existing
%      singleton*.
%
%      H = GUI_TESIS returns the handle to a new GUI_TESIS or the handle to
%      the existing singleton*.
%
%      GUI_TESIS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_TESIS.M with the given input arguments.
%
%      GUI_TESIS('Property','Value',...) creates a new GUI_TESIS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_TESIS_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI_TESIS_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_TESIS

% Last Modified by GUIDE v2.5 10-Jan-2019 12:35:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_TESIS_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_TESIS_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before GUI_TESIS is made visible.
function GUI_TESIS_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_TESIS (see VARARGIN)

% Choose default command line output for GUI_TESIS
axes(handles.axes1);
```

```
cla
xlabel('Tiempo(s)')
ylabel('X(m)')
axes(handles.axes2);
cla;
xlabel('Tiempo(s)')
ylabel('Y(m)')
axes(handles.axes3);
cla;
xlabel('Tiempo(s)')
ylabel('TETA(rad)')
xlim([0 20]);
ylim([-8 8]);
axes(handles.axes4);
cla;
xlabel('X(m)')
ylabel('Y(m)')
xlim([-0.6 0.6]);
ylim([-0.6 0.6]);
axes(handles.axes5);
cla
xlabel('Tiempo(s)')
ylabel('Wi(rad/s)')
xlim([0 20]);
ylim([-120 120]);
axes(handles.axes6);
cla;
xlabel('Tiempo(s)')
ylabel('Wd(rad/s)')
xlim([0 20]);
ylim([-120 120]);

axes(handles.logo)
handles.imagen=imread('robot.jpg');
imagesc(handles.imagen)
axis off

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI_TESIS wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI_TESIS_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in togglebutton1.
```

```
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ser1=get(handles.popupmenu1, 'Value');
ser2=get(handles.popupmenu2, 'Value');
ser3=get(handles.popupmenu3, 'Value');
ser4=get(handles.popupmenu4, 'Value');

switch ser1
   case 1
      Br=2400;
   case 2
      Br=9600;
   case 3
      Br=19200;
   case 4
      Br=57600;
end
switch ser2
   case 1
      Sb=1;
   case 2
      Sb=2;
end
switch ser3
   case 1
      Db=8;
   case 2
      Db=7;
   case 3
      Db=5;
end
switch ser4
   case 1
      Cp='COM1';
   case 2
      Cp='COM2';
   case 3
      Cp='COM3';
   case 4
      Cp='COM4';
   case 5
      Cp='COM5';
   case 6
      Cp='COM6';
   case 7
      Cp='COM7';
end

PS=serial(Cp);
set(PS,'Baudrate',Br); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',Sb); % se configura bit de parada a uno
set(PS,'DataBits',Db); % se configura que el dato es de 8 bits, debe estar entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
set(PS,'Timeout',0.2); % 50 mili segundos de tiempo de espera
fopen(PS);
pop = get(handles.togglebutton1, 'Value');
set(handles.togglebutton1,'Enable','inactive');
```

```
set(handles.togglebutton2,'Enable','on');

graph1=handles.axes1;
graph2=handles.axes2;
graph3=handles.axes3;
graph4=handles.axes4;
graph5=handles.axes5;
graph6=handles.axes6;

xlim(graph4, [-0.6 0.6]);
ylim(graph4, [-0.6 0.6]);

k=2;
tiempo(1)=0;
%X=zeros(9,1001);
while pop==1
   tic;
   b=fread(PS,1,'uint8');
if(b==126)
   out = fread(PS,9,'single');
   b=fread(PS,1,'uint8');
   X(:,k)=out;
   axes(handles.axes4);
   graphrobot(X(1,k),X(2,k),X(3,k),0.01206,'y');
   xlim(graph4, [-0.6 0.6]);
   ylim(graph4, [-0.6 0.6]);
   hold(graph4, 'on');
   plot2handle4 = plot(X(4,1:k),X(5,1:k),'r','linewidth',1, 'Parent', graph4);
   plot2handle4 = plot(X(1,1:k),X(2,1:k),'b','linewidth',1, 'Parent', graph4);
   hold(graph4, 'off');
   pause(0.005);
   time=toc;
   tiempo(k)=tiempo(k-1)+time;
   set(handles.text24,'String',tiempo(k));
   set(handles.text25,'String',X(9,k));
   k=k+1;
end
aux = get(handles.togglebutton2, 'Value');
  if aux==1
    pop=0;
  end
end
fclose(PS);
check1=get(handles.checkbox1,'Value');
check2=get(handles.checkbox2,'Value');
check3=get(handles.checkbox4,'Value');
xlim(graph1, [0 max(tiempo)]);
ylim(graph1, [-1 1]);
hold(graph1, 'on');
if(check2==1)
   plot2handle1 = plot(tiempo,X(4,:),'r','linewidth',1, 'Parent', graph1);
end
if(check3==1)
   plot2handle1 = plot(tiempo,X(1,:),'b','linewidth',1, 'Parent', graph1);
end
xlim(graph2, [0 max(tiempo)]);
ylim(graph2, [-1 1]);
hold(graph2, 'on');
if(check2==1)
```

```
    plot2handle2 = plot(tiempo,X(5,:),'r','linewidth',1, 'Parent', graph2);
end
if(check3==1)
    plot2handle2 = plot(tiempo,X(2,:),'b','linewidth',1, 'Parent', graph2);
end
xlim(graph3, [0 max(tiempo)]);
ylim(graph3, [-8 8]);
hold(graph3, 'on');
if(check2==1)
    plot2handle3 = plot(tiempo,X(6,:),'r','linewidth',1, 'Parent', graph3);
end
if(check3==1)
    plot2handle3 = plot(tiempo,X(3,:),'b','linewidth',1, 'Parent', graph3);
end
xlim(graph4, [-0.6 0.6]);
ylim(graph4, [-0.6 0.6]);
hold(graph4, 'on');
if(check2==1)
    plot2handle4 = plot(X(4,:),X(5,:),'r','linewidth',1, 'Parent', graph4);
end
if(check3==1)
    plot2handle4 = plot(X(1,:),X(2,:),'b','linewidth',1, 'Parent', graph4);
end
xlim(graph5, [0 max(tiempo)]);
ylim(graph5, [min(X(7,:))-10 max(X(7,:))+10]);
hold(graph5, 'on');
plot2handle5 = plot(tiempo,X(7,:),'b','linewidth',1, 'Parent', graph5);
xlim(graph6, [0 max(tiempo)]);
ylim(graph6, [min(X(8,:))-10 max(X(8,:))+10]);
hold(graph6, 'on');
plot2handle6 = plot(tiempo,X(8,:),'b','linewidth',1, 'Parent', graph6);

R1 = corrcoef(X(1,:),X(4,:));
R2 = corrcoef(X(2,:),X(5,:));
max1=max((X(4,:)-X(1,:)).^2+(X(5,:)-X(2,:)).^2);
desv=std((X(4,:)-X(1,:))+(X(5,:)-X(2,:)));

set(handles.text20,'String',R1(2));
set(handles.text21,'String',R2(2));
set(handles.text22,'String',max1);
set(handles.text23,'String',desv);

save('robot_curvas.mat','X');

set(handles.togglebutton2,'Value',0);
set(handles.togglebutton1,'Value',0);
set(handles.togglebutton1,'Enable','on');
set(handles.togglebutton2,'Enable','inactive');
% Hint: get(hObject,'Value') returns toggle state of togglebutton1


% --- Executes on button press in togglebutton2.
function togglebutton2_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton2
```

```
% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1


% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox2


% --- Executes on button press in checkbox4.
function checkbox4_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox4


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu2

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu3 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu3

% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu4.
function popupmenu4_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu4 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu4

% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

ANEXO 3. DATASHEET MPU-9150

3 Register Map for Gyroscope and Accelerometer

The register map for the MPU-9150's Gyroscope and Accelerometer section is listed below. The Magnetometer's register map can be found in section 5.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	XG_ST	YG_ST	ZG_ST	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		
23	35	FIFO_EN	R/W	TEMP_FIFO_EN	XG_FIFO_EN	YG_FIFO_EN	ZG_FIFO_EN	ACCEL_FIFO_EN	SLV2_FIFO_EN	SLV1_FIFO_EN	SLV0_FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT_MST_EN	WAIT_FOR_ES	SLV_3_FIFO_EN	I2C_MST_P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0_RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0_EN	I2C_SLV0_BYTE_SW	I2C_SLV0_REG_DIS	I2C_SLV0_GRP	I2C_SLV0_LEN[3:0]			
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1_RW	I2C_SLV1_ADDR[6:0]						
29	41	I2C_SLV1_REG	R/W	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1_EN	I2C_SLV1_BYTE_SW	I2C_SLV1_REG_DIS	I2C_SLV1_GRP	I2C_SLV1_LEN[3:0]			
2B	43	I2C_SLV2_ADDR	R/W	I2C_SLV2_RW	I2C_SLV2_ADDR[6:0]						
2C	44	I2C_SLV2_REG	R/W	I2C_SLV2_REG[7:0]							
2D	45	I2C_SLV2_CTRL	R/W	I2C_SLV2_EN	I2C_SLV2_BYTE_SW	I2C_SLV2_REG_DIS	I2C_SLV2_GRP	I2C_SLV2_LEN[3:0]			
2E	46	I2C_SLV3_ADDR	R/W	I2C_SLV3_RW	I2C_SLV3_ADDR[6:0]						
2F	47	I2C_SLV3_REG	R/W	I2C_SLV3_REG[7:0]							
30	48	I2C_SLV3_CTRL	R/W	I2C_SLV3_EN	I2C_SLV3_BYTE_SW	I2C_SLV3_REG_DIS	I2C_SLV3_GRP	I2C_SLV3_LEN[3:0]			
31	49	I2C_SLV4_ADDR	R/W	I2C_SLV4_RW	I2C_SLV4_ADDR[6:0]						
32	50	I2C_SLV4_REG	R/W	I2C_SLV4_REG[7:0]							
33	51	I2C_SLV4_DO	R/W	I2C_SLV4_DO[7:0]							
34	52	I2C_SLV4_CTRL	R/W	I2C_SLV4_EN	I2C_SLV4_INT_EN	I2C_SLV4_REG_DIS	I2C_MST_DLY[4:0]				
35	53	I2C_SLV4_DI	R	I2C_SLV4_DI[7:0]							
36	54	I2C_MST_STATUS	R	PASS_THROUGH	I2C_SLV4_DONE	I2C_LOST_ARB	I2C_SLV4_NACK	I2C_SLV3_NACK	I2C_SLV2_NACK	I2C_SLV1_NACK	I2C_SLV0_NACK
37	55	INT_PIN_CFG	R/W	INT_LEVEL	INT_OPEN	LATCH_INT_EN	INT_RD_CLEAR	FSYNC_INT_LEVEL	FSYNC_INT_EN	I2C_BYPASS_EN	-
38	56	INT_ENABLE	R/W	-	-	-	FIFO_OFLOW_EN	I2C_MST_INT_EN	-	-	DATA_RDY_EN

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3A	58	INT_STATUS	R	-	-	-	FIFO_OFLOW_INT	I2C_MST_INT	-	-	DATA_RDY_INT
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							
49	73	EXT_SENS_DATA_00	R	EXT_SENS_DATA_00[7:0]							
4A	74	EXT_SENS_DATA_01	R	EXT_SENS_DATA_01[7:0]							
4B	75	EXT_SENS_DATA_02	R	EXT_SENS_DATA_02[7:0]							
4C	76	EXT_SENS_DATA_03	R	EXT_SENS_DATA_03[7:0]							
4D	77	EXT_SENS_DATA_04	R	EXT_SENS_DATA_04[7:0]							
4E	78	EXT_SENS_DATA_05	R	EXT_SENS_DATA_05[7:0]							
4F	79	EXT_SENS_DATA_06	R	EXT_SENS_DATA_06[7:0]							
50	80	EXT_SENS_DATA_07	R	EXT_SENS_DATA_07[7:0]							
51	81	EXT_SENS_DATA_08	R	EXT_SENS_DATA_08[7:0]							
52	82	EXT_SENS_DATA_09	R	EXT_SENS_DATA_09[7:0]							
53	83	EXT_SENS_DATA_10	R	EXT_SENS_DATA_10[7:0]							
54	84	EXT_SENS_DATA_11	R	EXT_SENS_DATA_11[7:0]							
55	85	EXT_SENS_DATA_12	R	EXT_SENS_DATA_12[7:0]							
56	86	EXT_SENS_DATA_13	R	EXT_SENS_DATA_13[7:0]							
57	87	EXT_SENS_DATA_14	R	EXT_SENS_DATA_14[7:0]							
58	88	EXT_SENS_DATA_15	R	EXT_SENS_DATA_15[7:0]							
59	89	EXT_SENS_DATA_16	R	EXT_SENS_DATA_16[7:0]							
5A	90	EXT_SENS_DATA_17	R	EXT_SENS_DATA_17[7:0]							
5B	91	EXT_SENS_DATA_18	R	EXT_SENS_DATA_18[7:0]							
5C	92	EXT_SENS_DATA_19	R	EXT_SENS_DATA_19[7:0]							
5D	93	EXT_SENS_DATA_20	R	EXT_SENS_DATA_20[7:0]							
5E	94	EXT_SENS_DATA_21	R	EXT_SENS_DATA_21[7:0]							
5F	95	EXT_SENS_DATA_22	R	EXT_SENS_DATA_22[7:0]							
60	96	EXT_SENS_DATA_23	R	EXT_SENS_DATA_23[7:0]							
63	99	I2C_SLV0_DO	R/W	I2C_SLV0_DO[7:0]							
64	100	I2C_SLV1_DO	R/W	I2C_SLV1_DO[7:0]							

38	56	INT_ENABLE	R/W	-	-	-	FIFO_OFLOW_EN	I2C_MST_INT_EN	-	-	DATA_RDY_EN

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
65	101	I2C_SLV2_DO	R/W	I2C_SLV2_DO[7:0]							
66	102	I2C_SLV3_DO	R/W	I2C_SLV3_DO[7:0]							
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	-	-	-	-	-	FIFO_COUNT[10:8]		
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

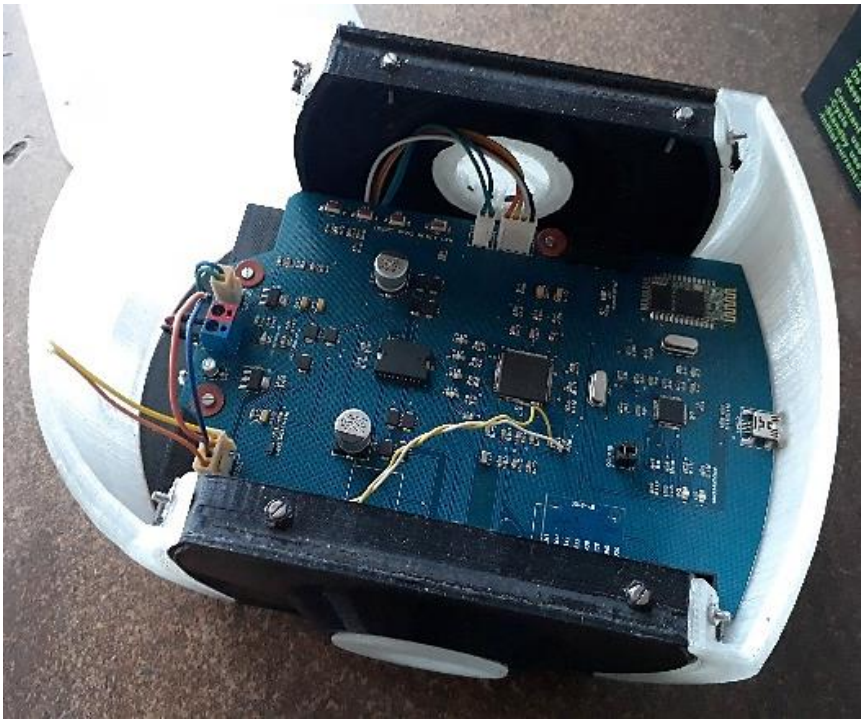
Note: Register Names ending in _H and _L contain the high and low bytes, respectively, of an internal register value.

In the detailed register tables that follow, register names are in capital letters, while register values are in capital letters and italicized. For example, the ACCEL_XOUT_H register (Register 59) contains the 8 most significant bits, *ACCEL_XOUT*[15:8], of the 16-bit X-Axis accelerometer measurement, *ACCEL_XOUT*.

The reset value is 0x00 for all registers other than the registers below.

- Register 107: 0x40.
- Register 117: 0x68.

ANEXO 4. ESTRUCTURA ROBOT DIFERENCIAL



ANEXO 5. DESPIECE CARCAZA ROBOT

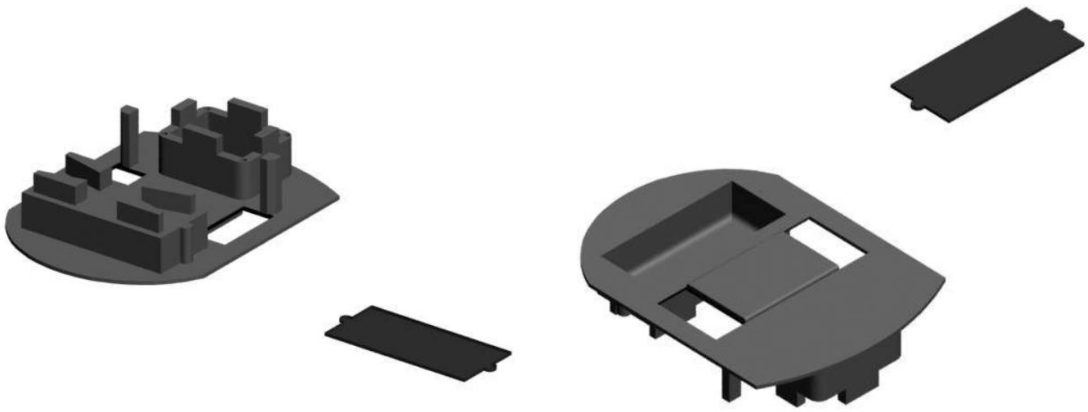
Armado de la estructura

Tapa superior

Tapa frontal y posterior

Tapas pequeñas

Escudos laterales



Base de la carcasa

ANEXO 6. ESQUEMA ELECTRICO STM32F407

7 Electrical schematics

Figure 9. STM32F407G-DISC1

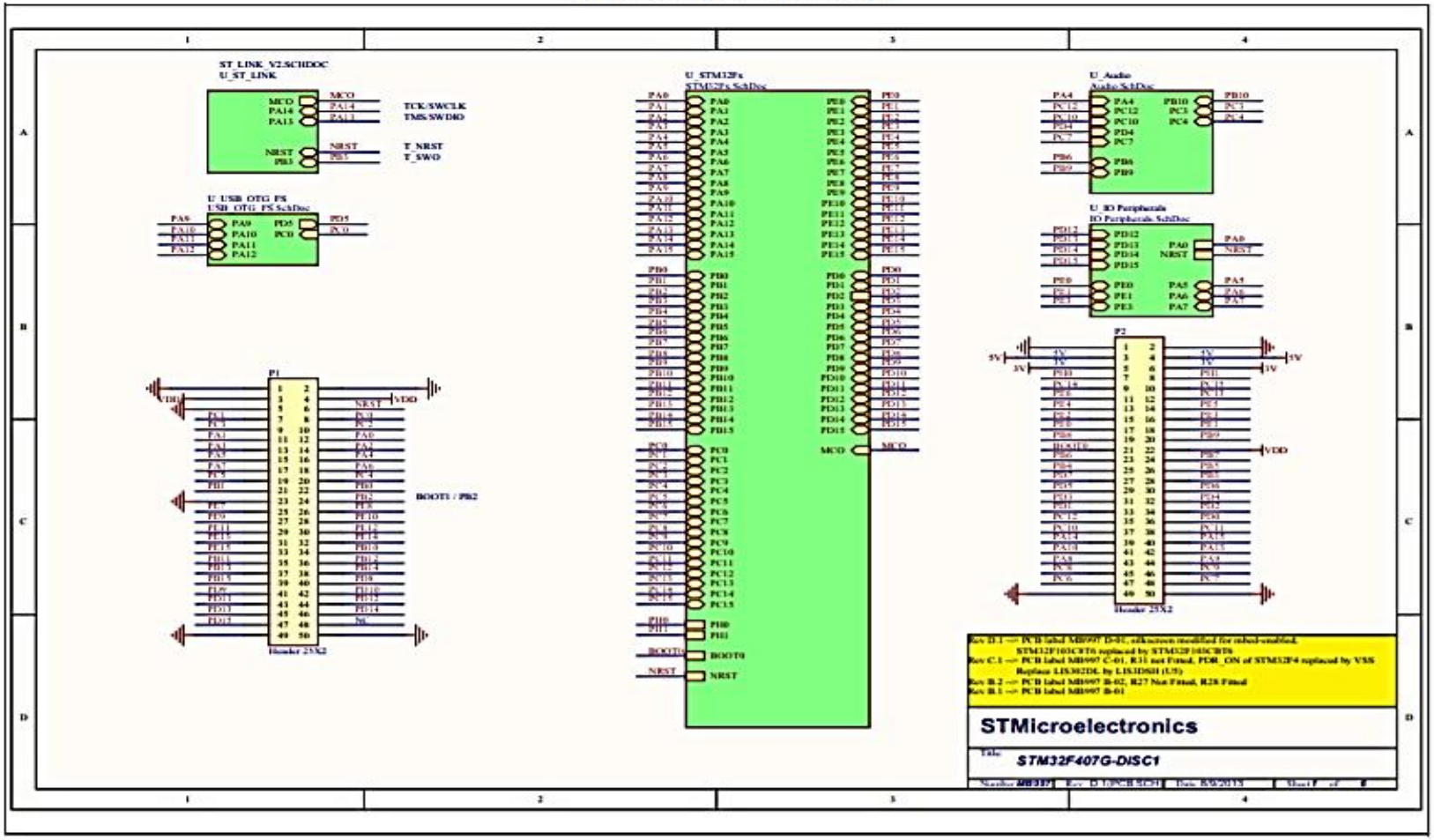


Figure 10. ST-LINK/V2 (SWD only)

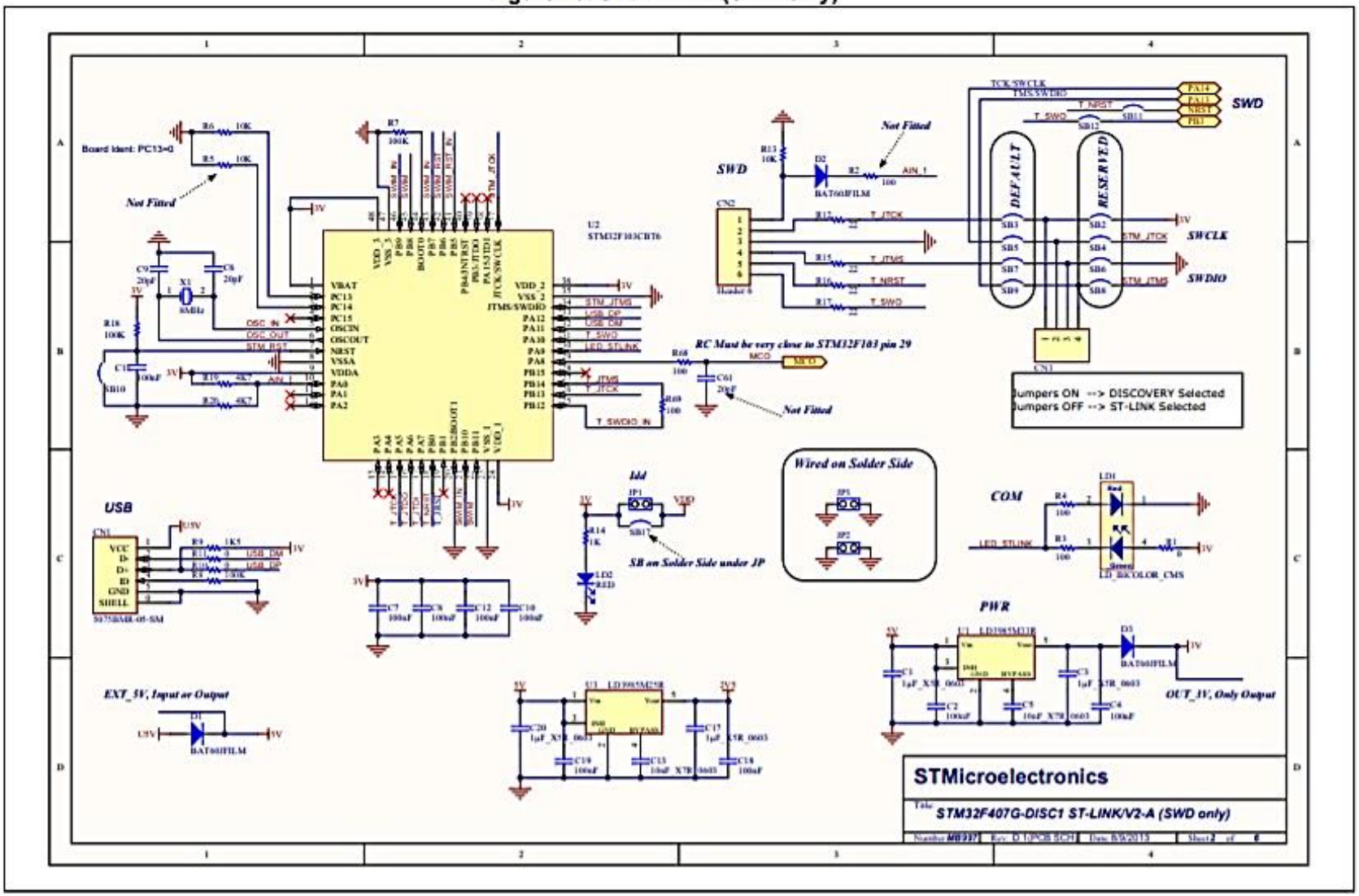
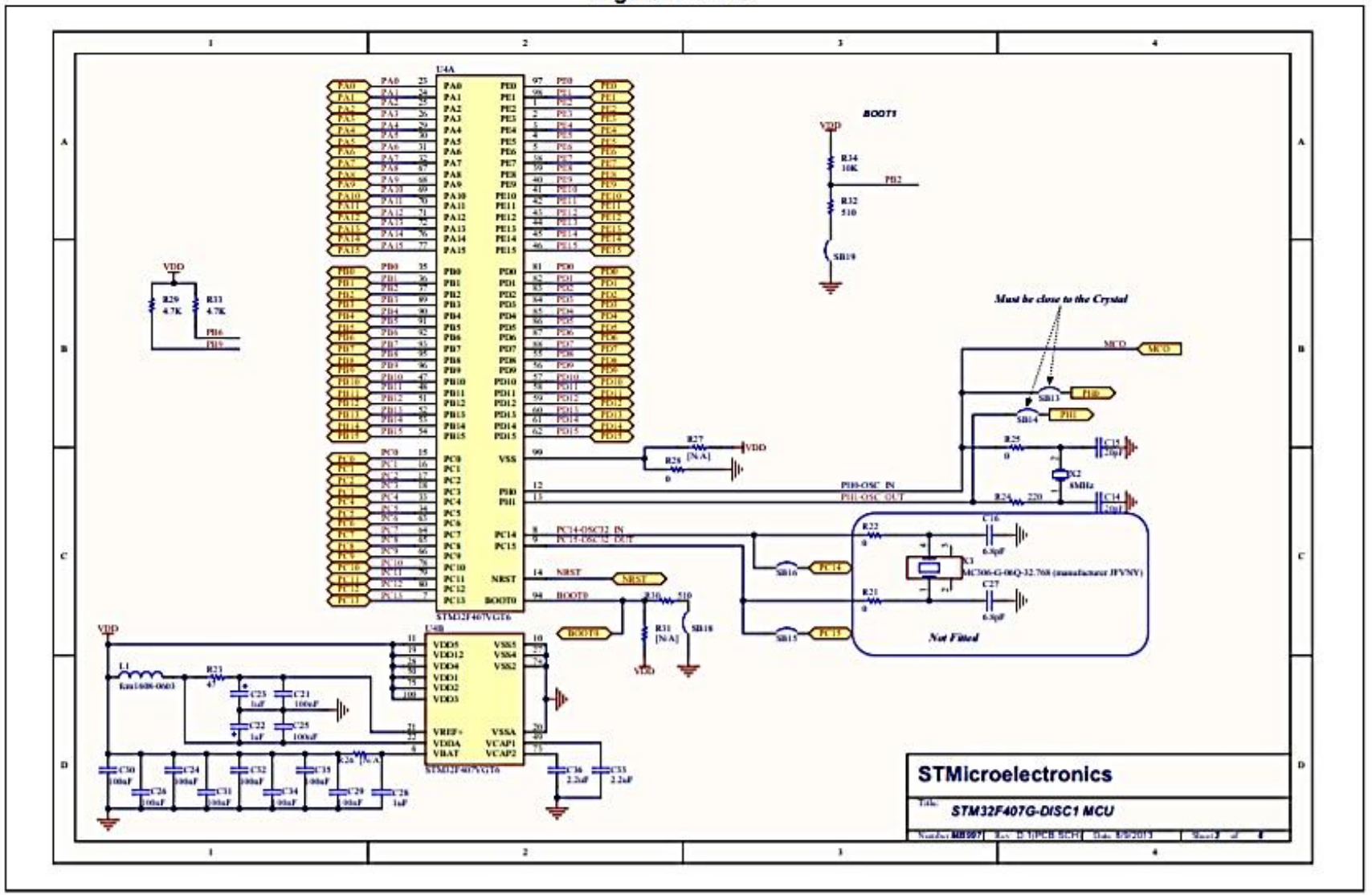


Figure 11. MCU

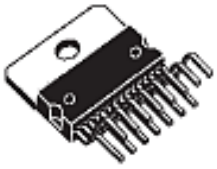


ANEXO 7. DATASHEET L298

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15

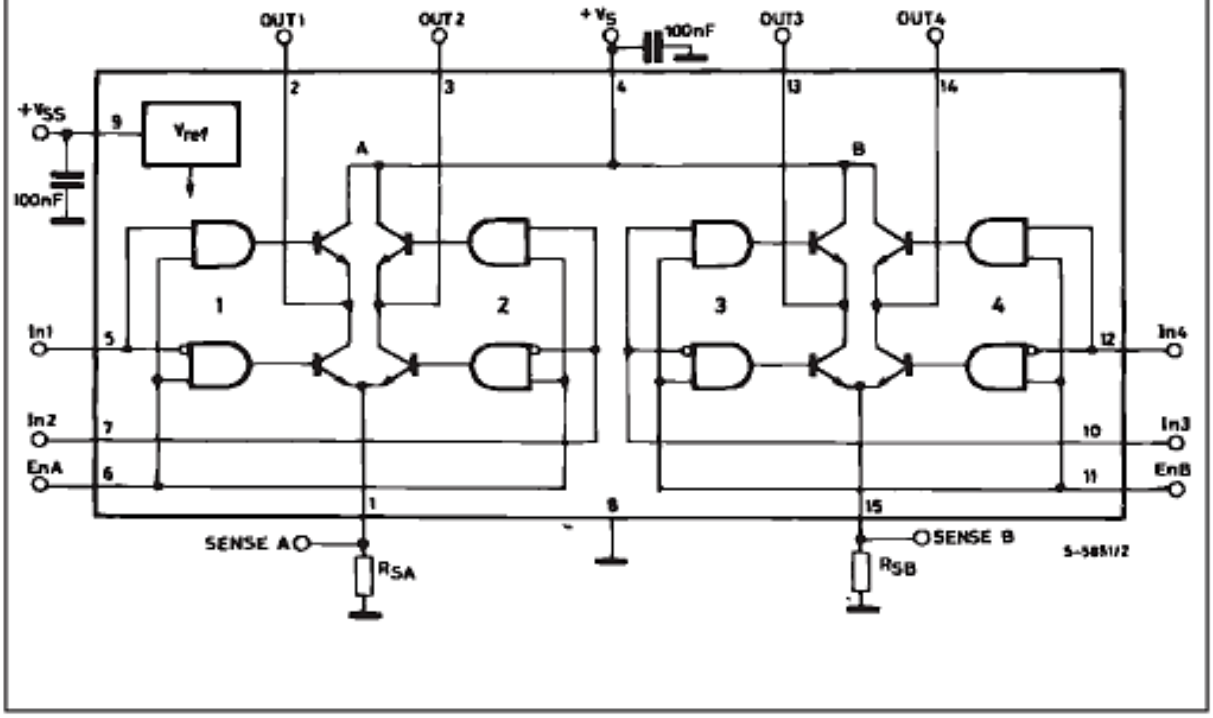
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

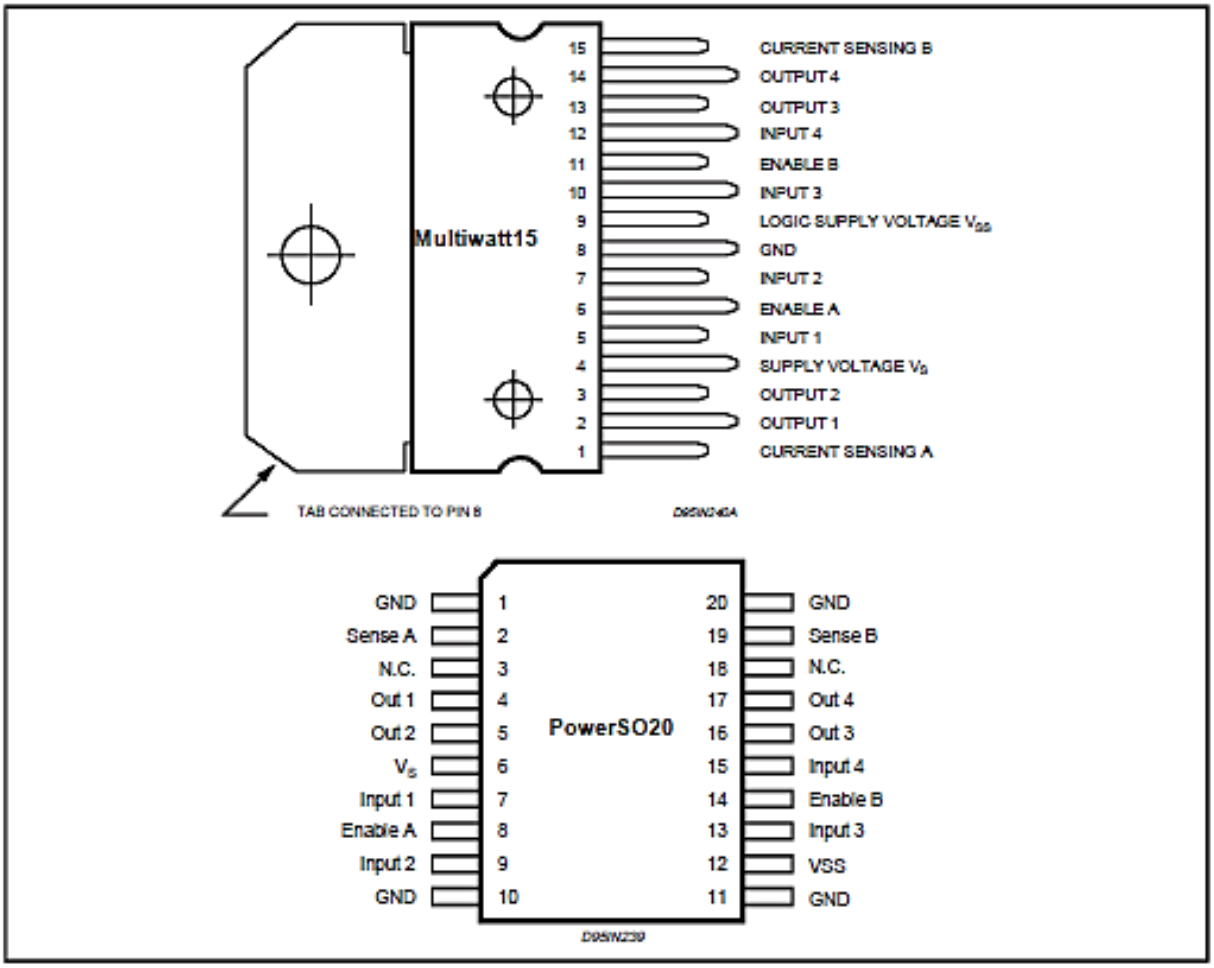
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

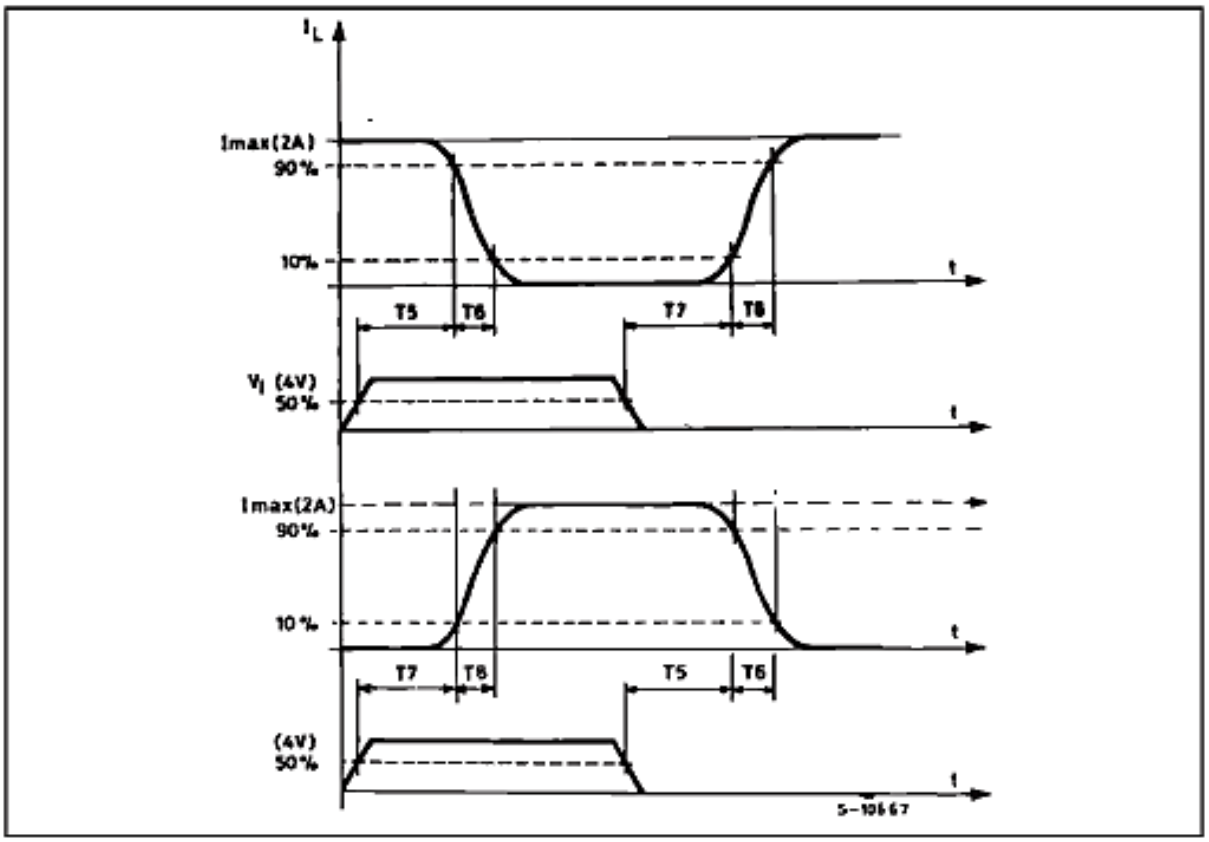
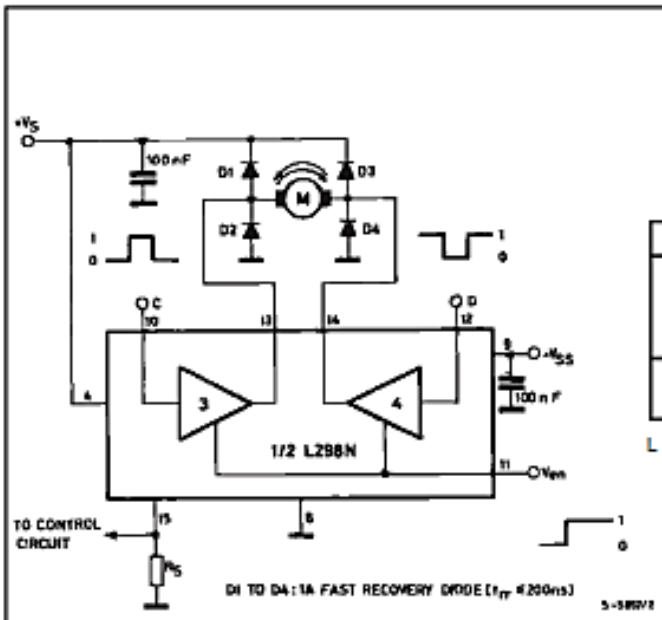


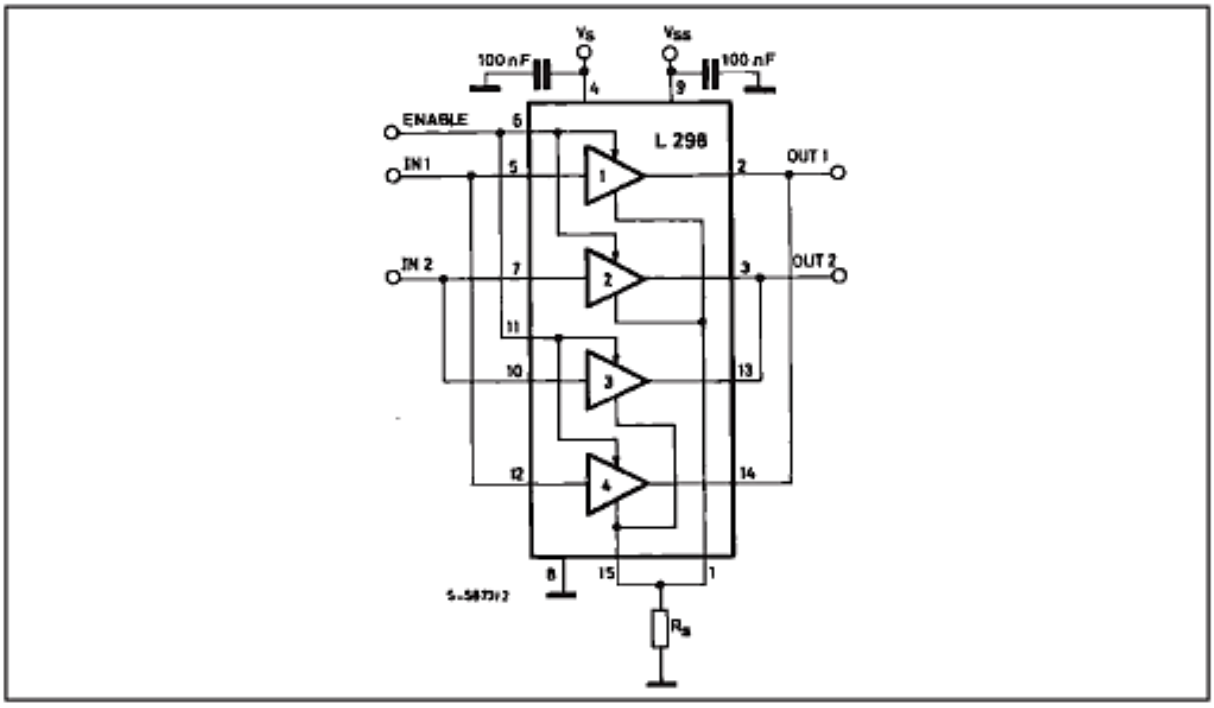
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A ; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output : an external resistor (R_{SA} ; R_{SB}.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements ($t_{rr} \leq 200$ nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped; Shottky diodes would be preferred.

UNIVERSIDAD TECNOLÓGICA ISRAEL

MANUAL DE USUARIO

PLATAFORMA ROBÓTICA MÓVIL DIFERENCIAL PARA SEGUIMIENTO DE TRAYECTORIA

MARTÍN DAVID LARCO CHIRIBOGA

QUITO – ECUADOR

AÑO: 2019

ÍNDICE

INTRODUCCIÓN

Este manual detallará la forma de uso y funcionamiento de la plataforma robótica móvil diferencial, utilizada para probar el algoritmo de seguimiento de trayectoria.

Como compilar el programa desde el programa MATLAB, modo de conexión del robot, conexión de la batería

CARACTERÍSTICAS:

- Microprocesador 32 bits, 8Mhz/168Mhz.
- Programador ST-Link V2 vía USB.
- Comunicación Esclavo Bluetooth.
- Batería LIPO 2S 2000 mA.
- Programable vía Simulink.

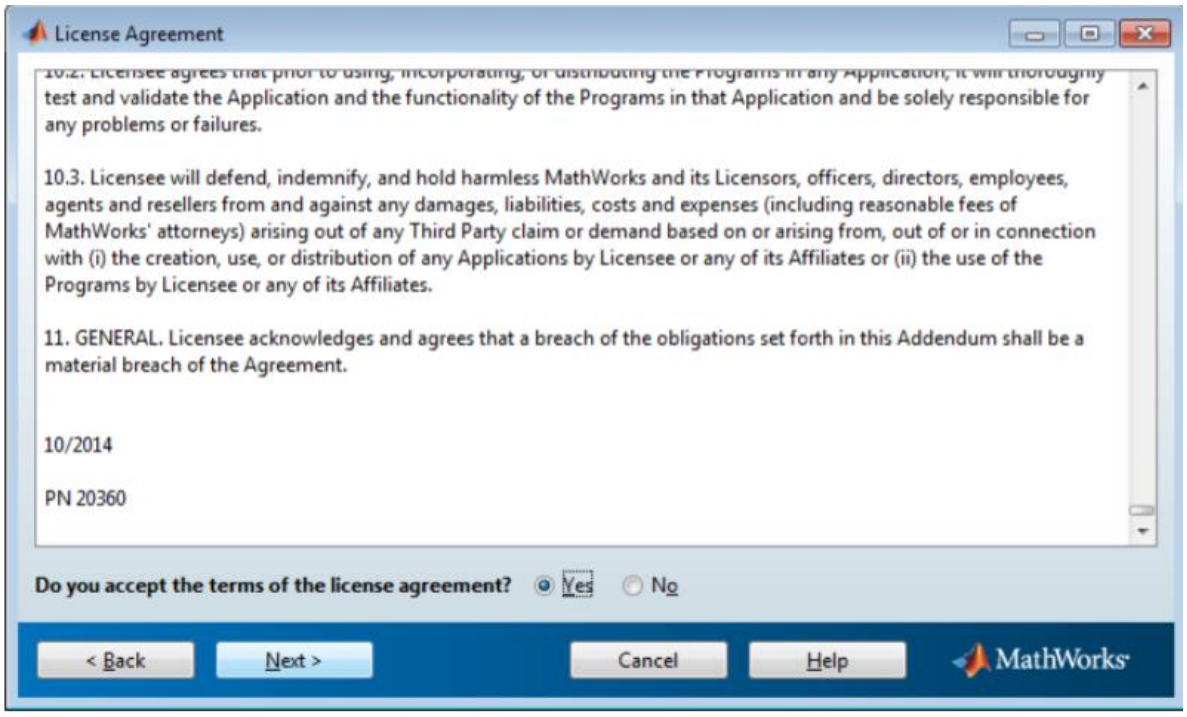
REQUISITOS:

- Matlab (https://la.mathworks.com/downloads/)
- Waijunj Blockset (https://www.aimagin.com/download/)

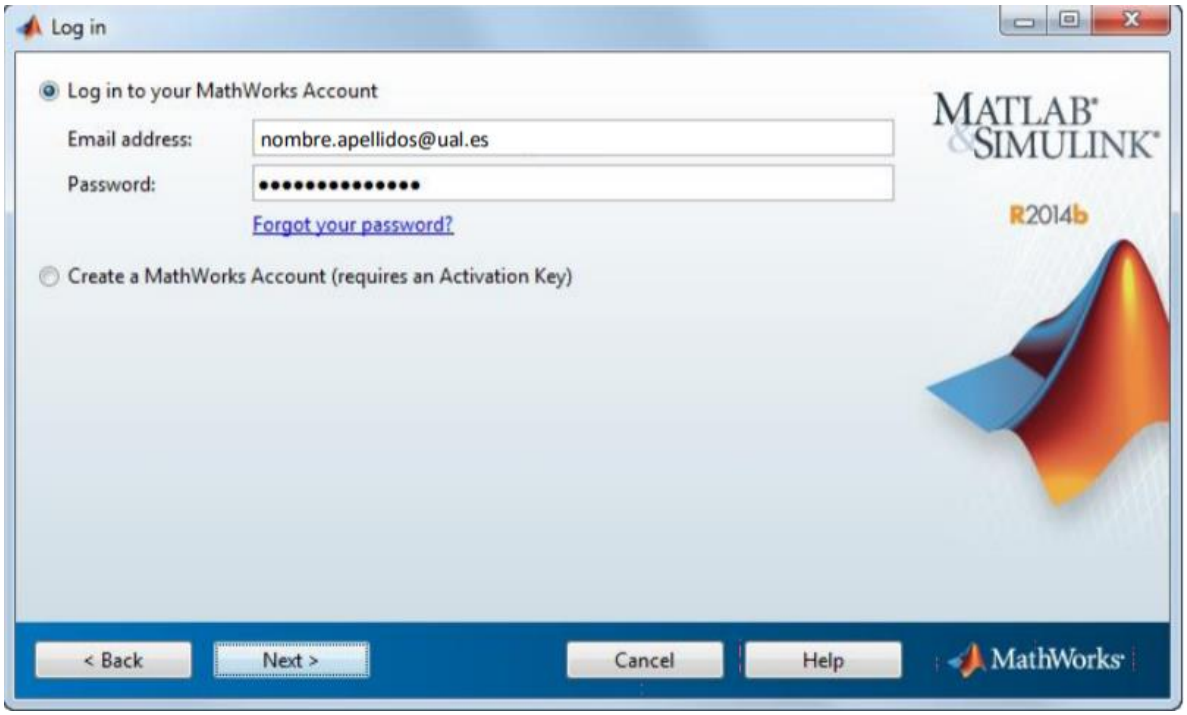
1. INSTALACIÓN MATLAB SIMULINK

- Ejecute el archivo .exe como administrador.

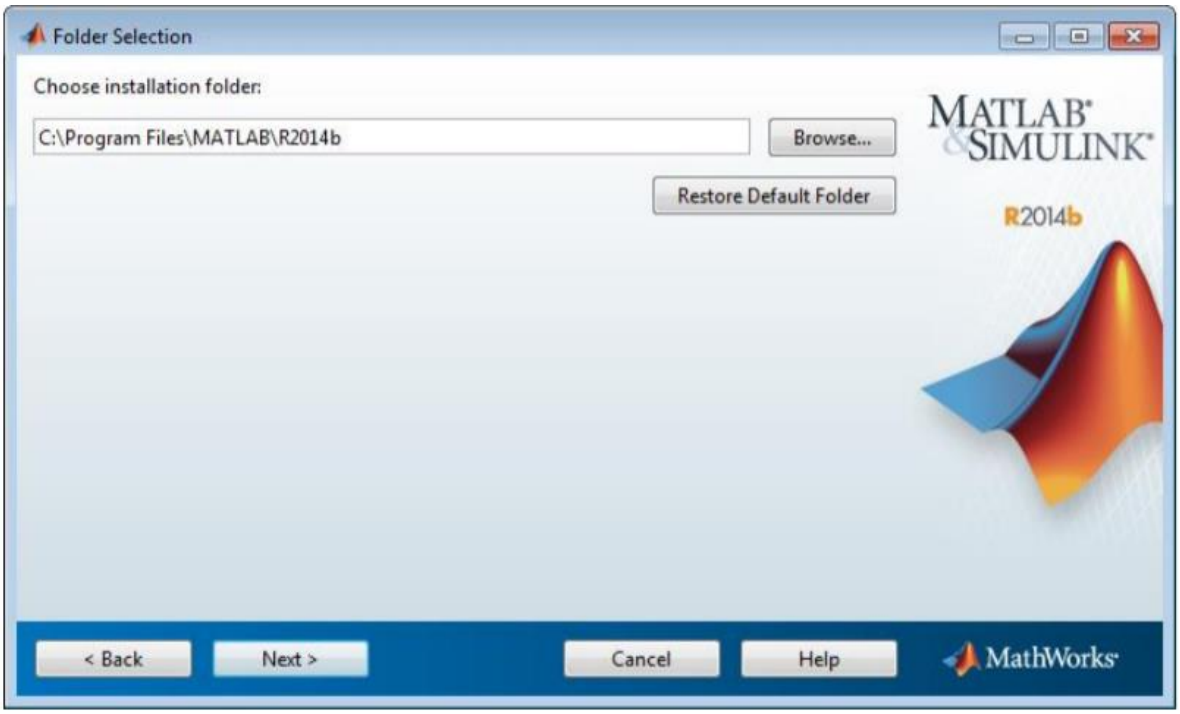
- Se acepta el contrato de licencia.



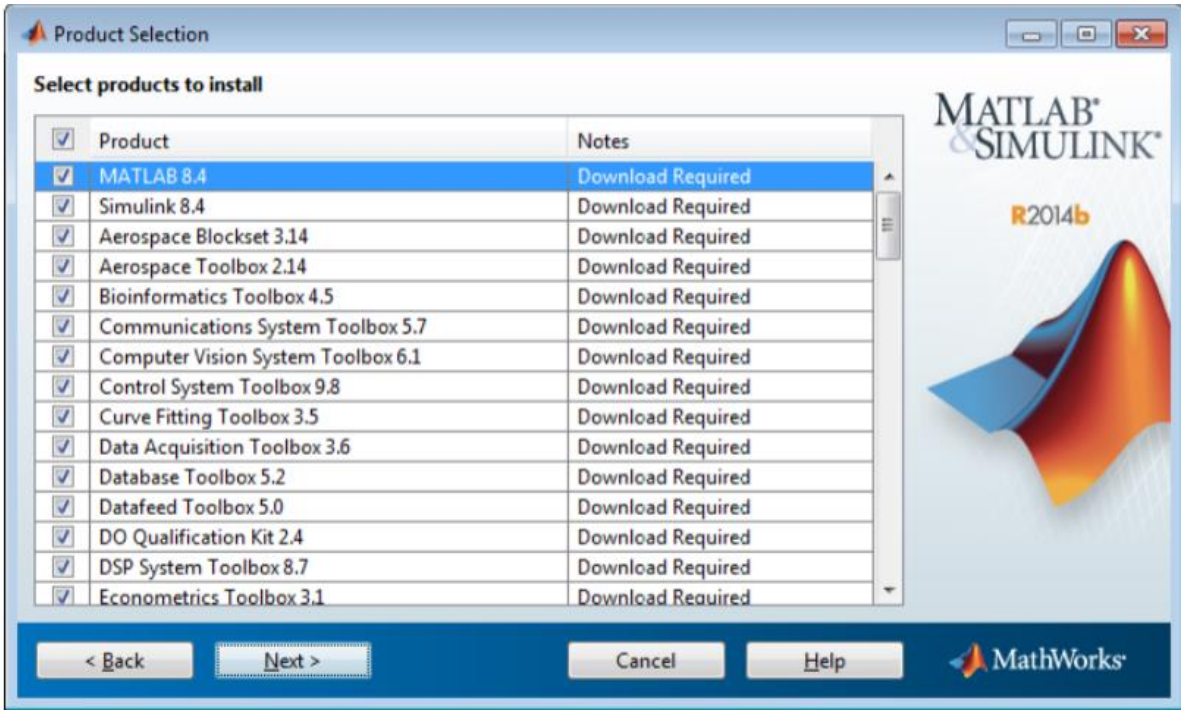
- Introducir la cuenta de *MathWorks Account*.



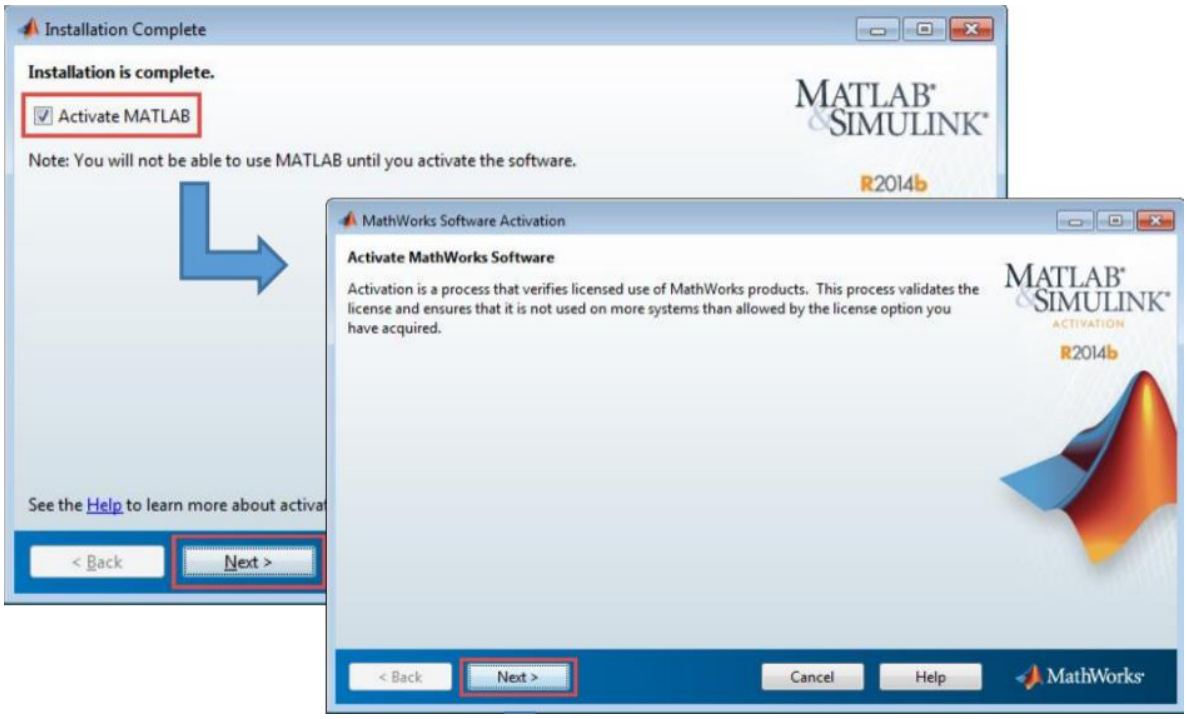
- Seleccionar la licencia asociada a la cuenta de usuario. Seleccionar la carpeta donde se almacenará el programa.



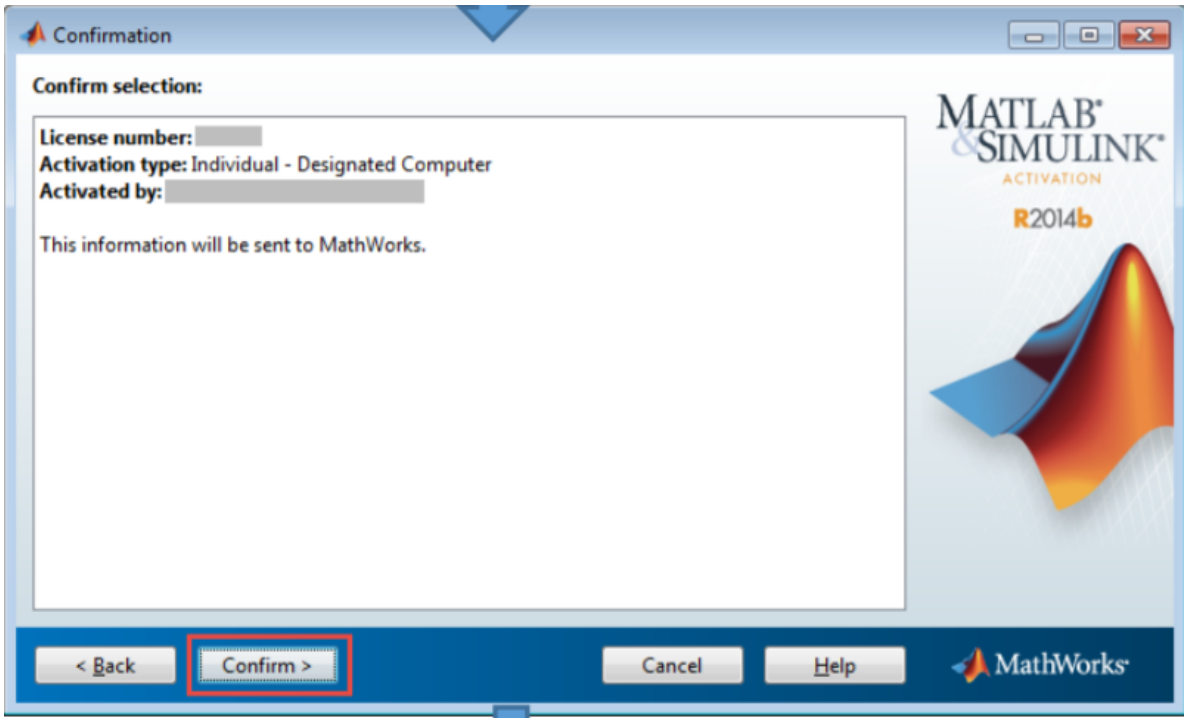
- Seleccione todas las librerías que se descargaran. Pulse siguiente e instalar.



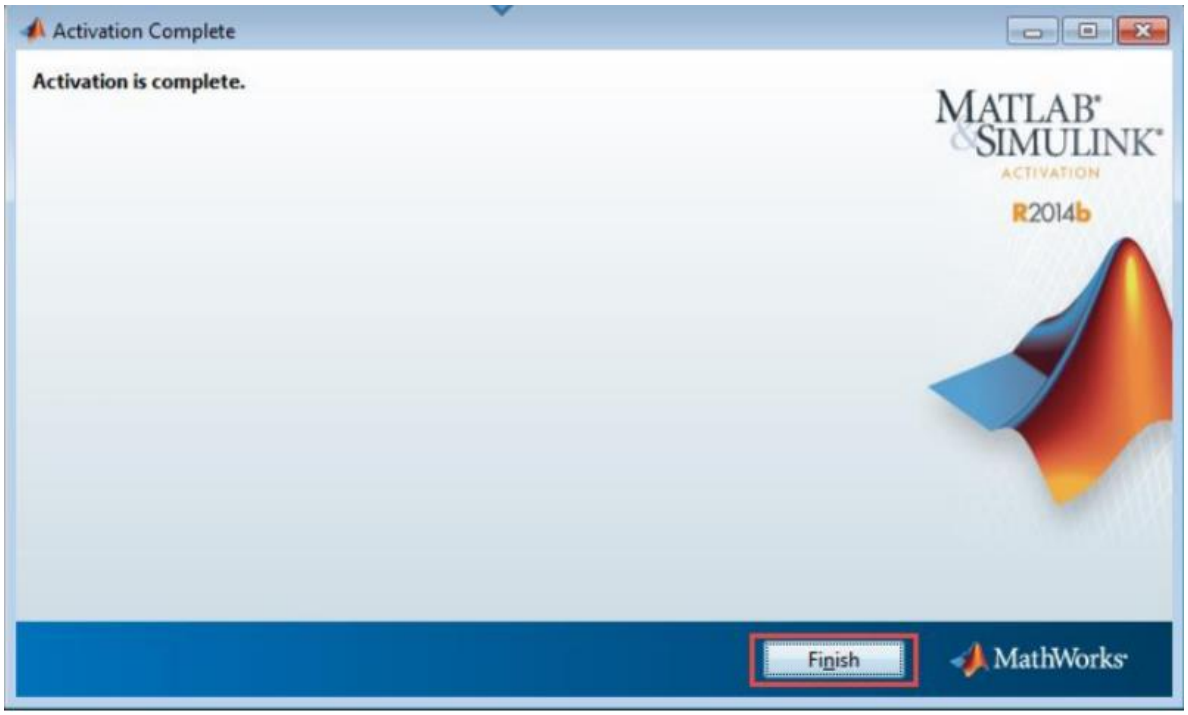
- Como último paso es la activación del software.



- Se introduce los datos de la licencia que fueron enviados a su cuenta de MathWorks.



- Se confirma los datos y se finaliza la instalación, después de lo cual ya puede empezar a utilizar el programa.



2. INSTALACION DE LAS LIBRERIAS WAIJUNG

Para instalar el software se deben seguir los siguientes pasos:

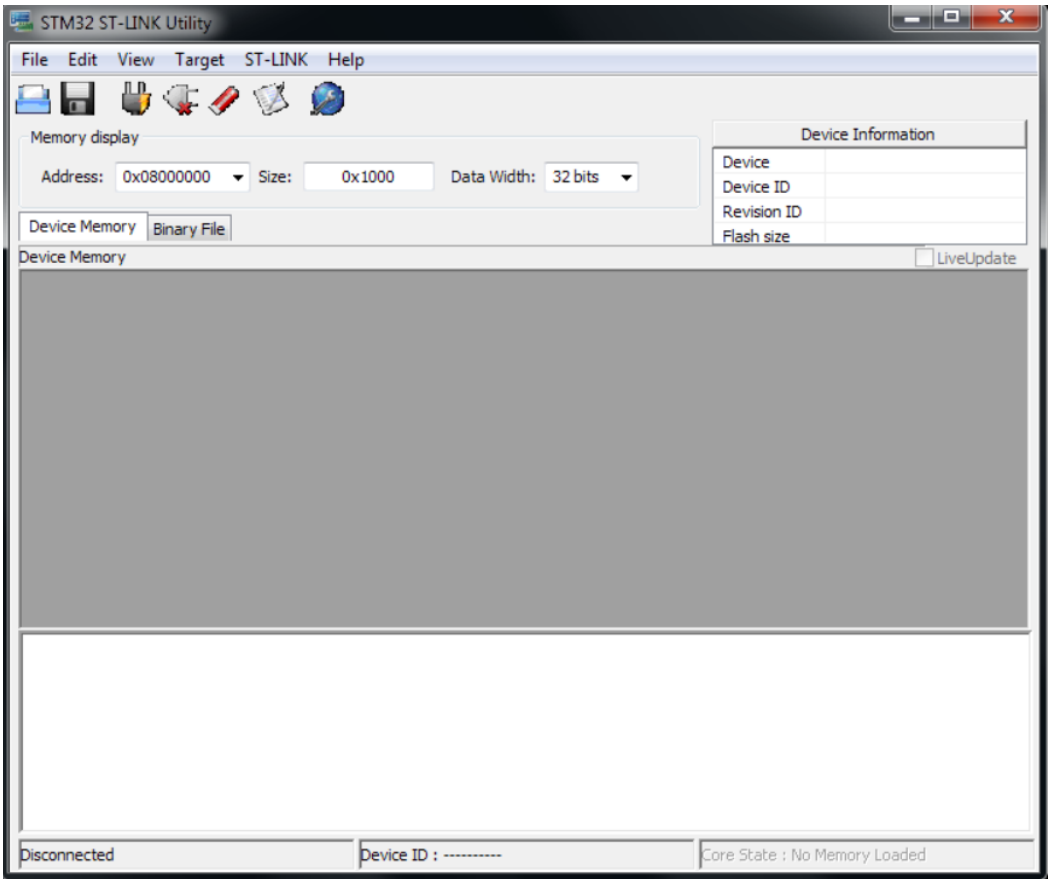
Descargue e instale STM32 ST-Link Utility and ST-Link/V2 USB driver desde el link http://www.st.com/web/catalog/tools/FM146/CL1984/SC724/SS1677/PF251168.

Para Windows 7 y anteriores ST-Link/V2 USB driver es instalado automáticamente al ejecutar STM32 ST-Link Utility.

Para Windows 8 ST-Link/V2 USB driver debe ser instalado después de ejecutar STM32 ST-Link Utility.

ST-Link Utility es utilizado para descargar archivos ejecutables (*.bin o *.hex) hacia la memoria flash de los STM32.

Después de haber descargado e instalado la utilidad ST-Link, se puede abrir la ventana de ST-Link.



Descargar e instalar FTDI USB driver for aMG USB Connect.

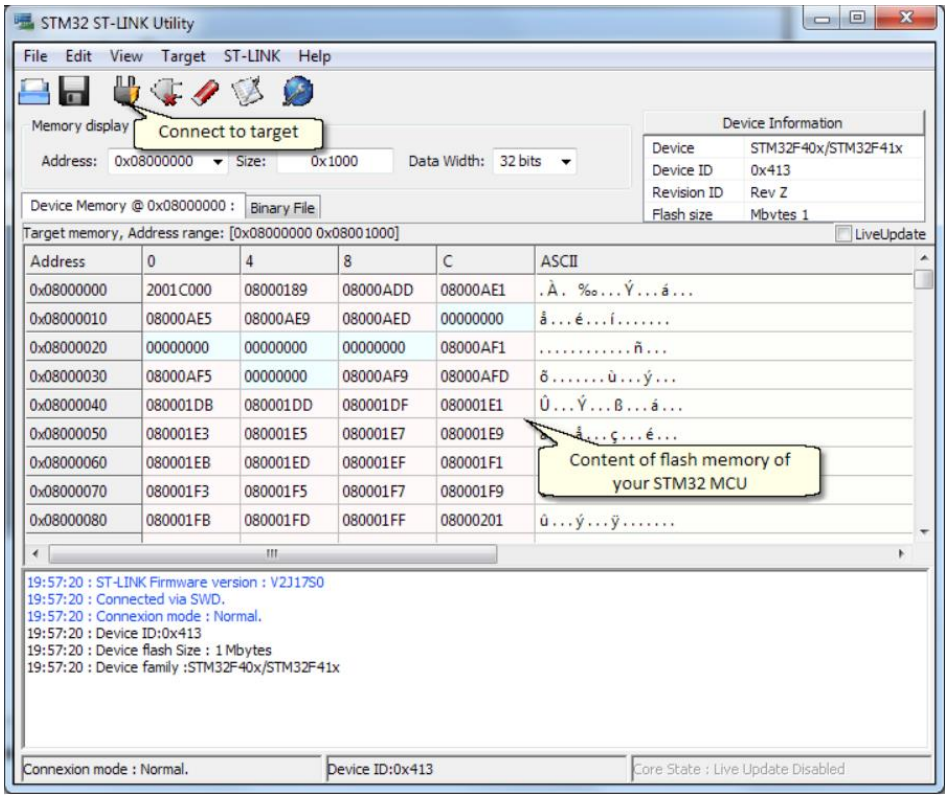
Descargue el archivo desde el link http://www.ftdichip.com/Drivers/VCP.htm.

Si desea más instrucciones para la instalación del driver consulte el siguiente link: http://www.ftdichip.com/Support/Documents/InstallGuides.htm

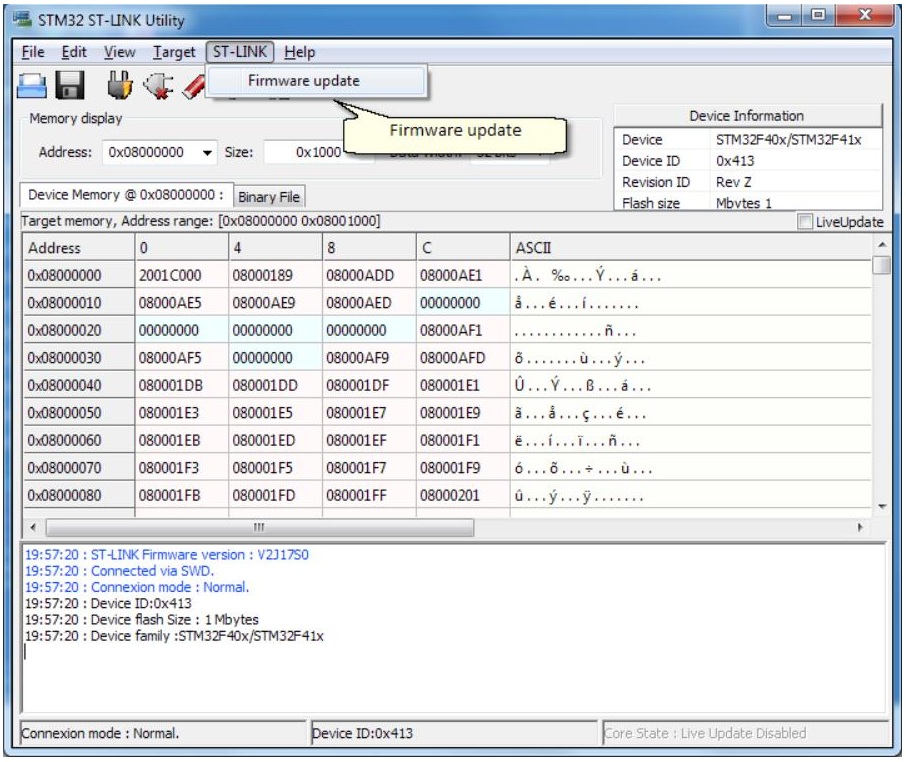
Prueba de conexión de la placa.

Conecte el robot a través del puerto mini USB con el cable mini USB y ejecute el programa STM32 ST-Link.

Para realizar la prueba de conectividad presione la tecla “Conect to target” y debería reflejar una información similar como se muestra en la imagen siguiente:



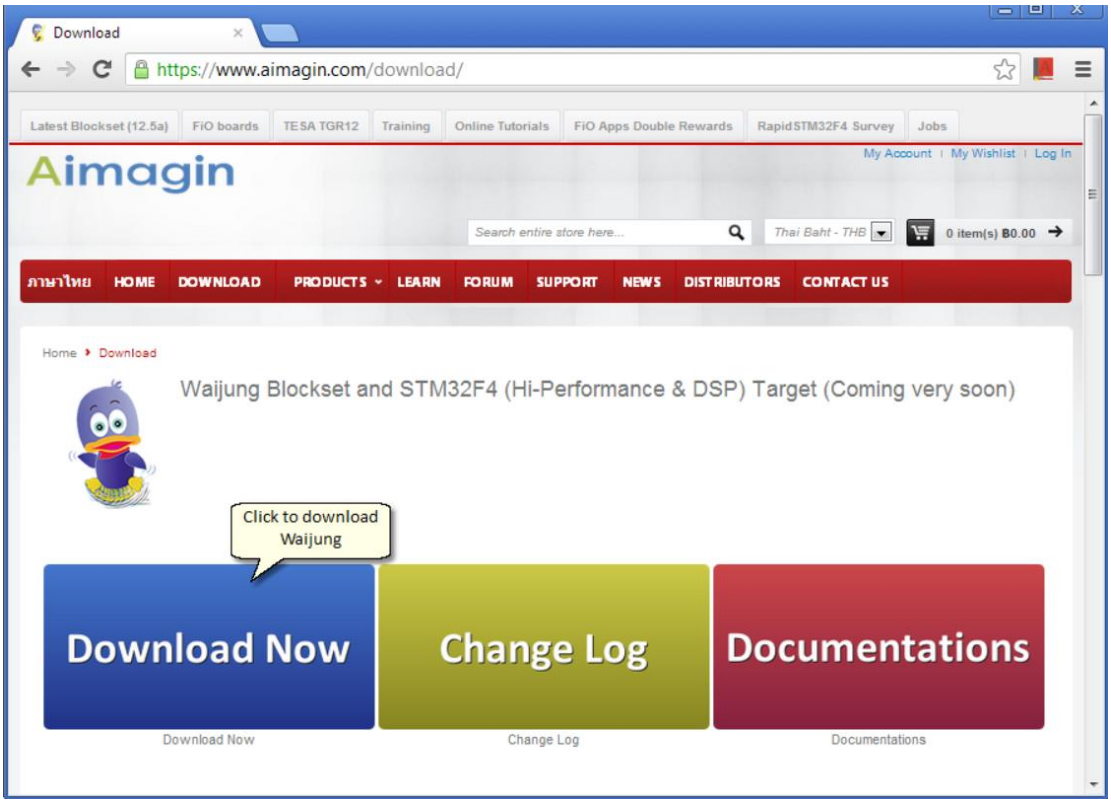
Dependiendo del hardware del ST-Link es posible que deba actualizar el firmware del hardware ST-Link, para lo cual presione en la barra de menú ST-Link-Firmware Update y siga los pasos de la pantalla.



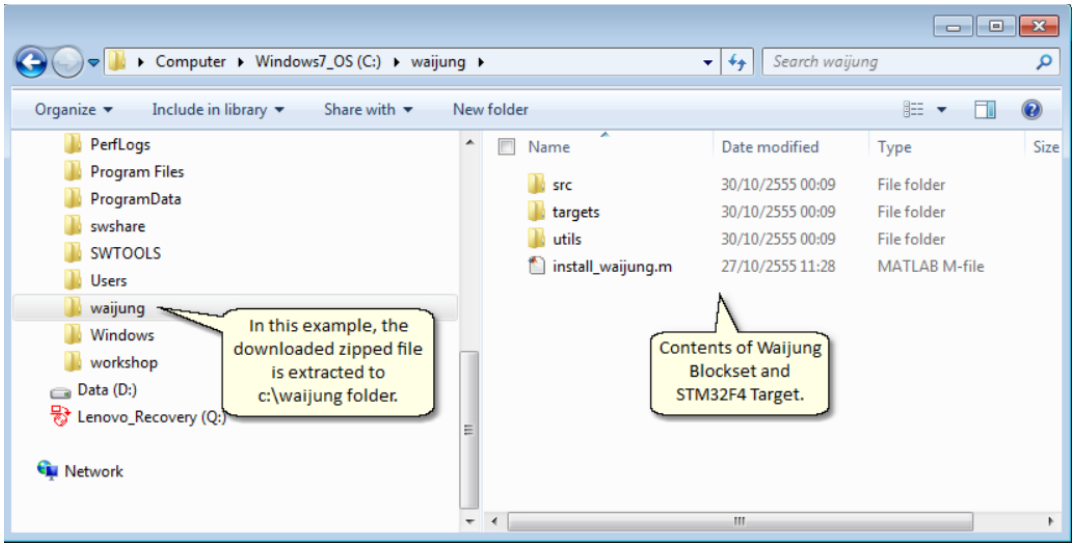
Si el programa advierte sobre "el dispositivo no está en el modo correcto", simplemente desconecte y vuelva a conectar la fuente de alimentación y ejecute nuevamente la actualización del firmware.

Instalar Waijung Blockset

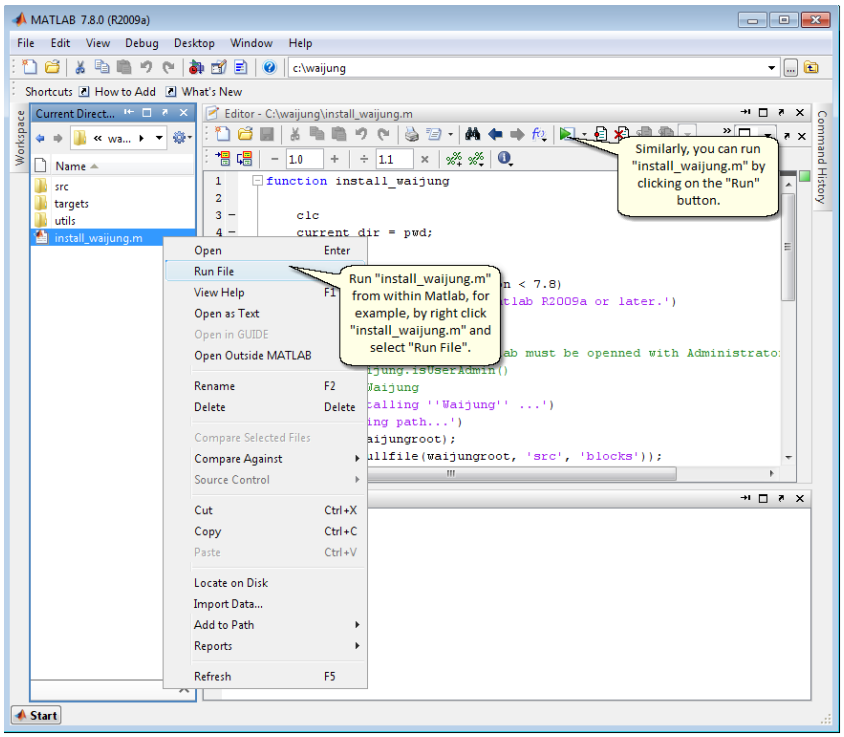
Descargue el paquete de instalación de Waijung Blockset y STM32F4 Target (waijung _ ## _ ##. 7z) desde http://www.aimagin.com/download/.



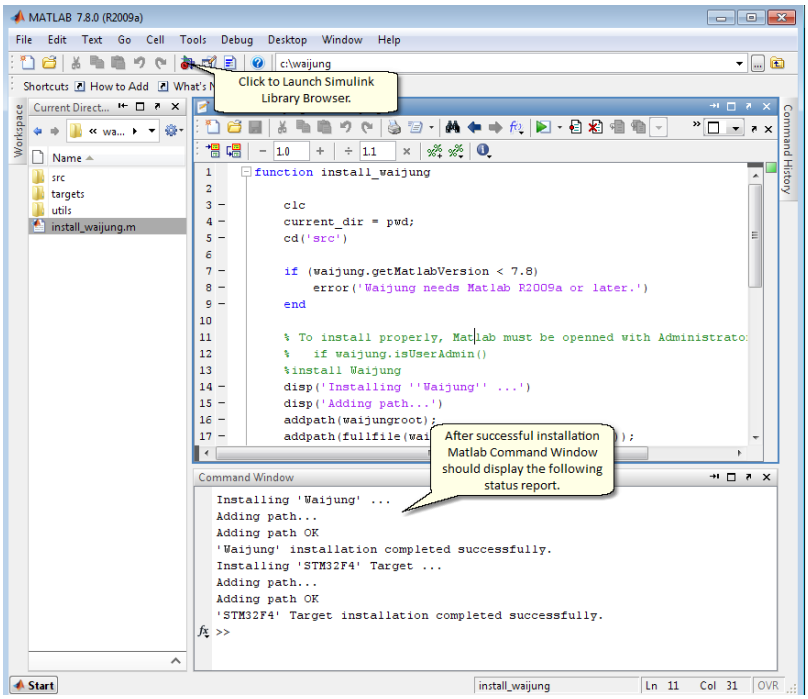
Una vez finalizada la descarga. Extraiga el paquete de instalación descargado a su directorio deseado. Este será el directorio raíz de Waijung Blockset y STM32F4 Target. Aunque puede extraer y ejecutar Waijung desde cualquier directorio en su computadora. Es recomendable extraer a un directorio donde tenga acceso completo de escritura. Si está actualizado el software, puede reemplazar el contenido del software anterior con la nueva versión y ejecutar install_waijung.m de manera normal.



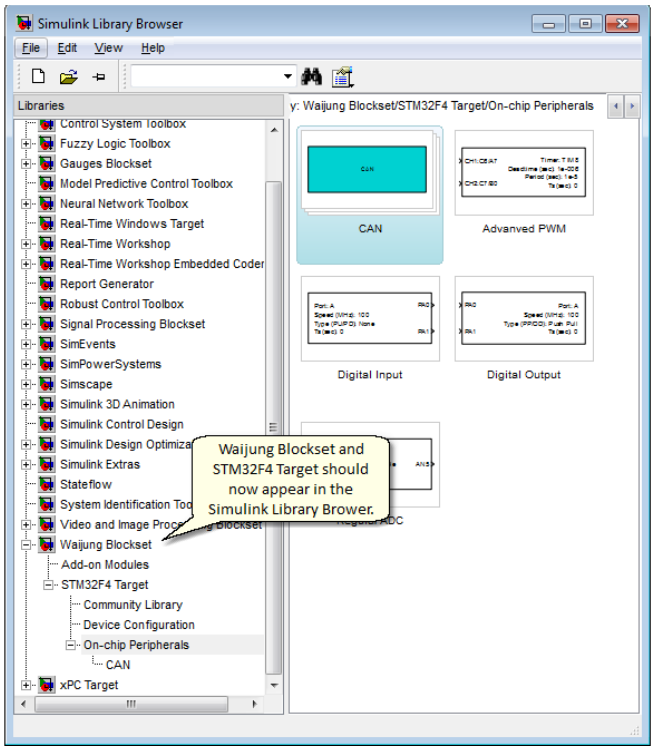
Abra Matlab como administrador y ejecute "install_waijung.m".



Después de una instalación exitosa, la ventana de comandos de Matlab debe informar el estado como se muestra a continuación.



Abra el Explorador de bibliotecas de Simulink y verá Waijung Blockset y STM32F4 Target. Esto completa el proceso de instalación.

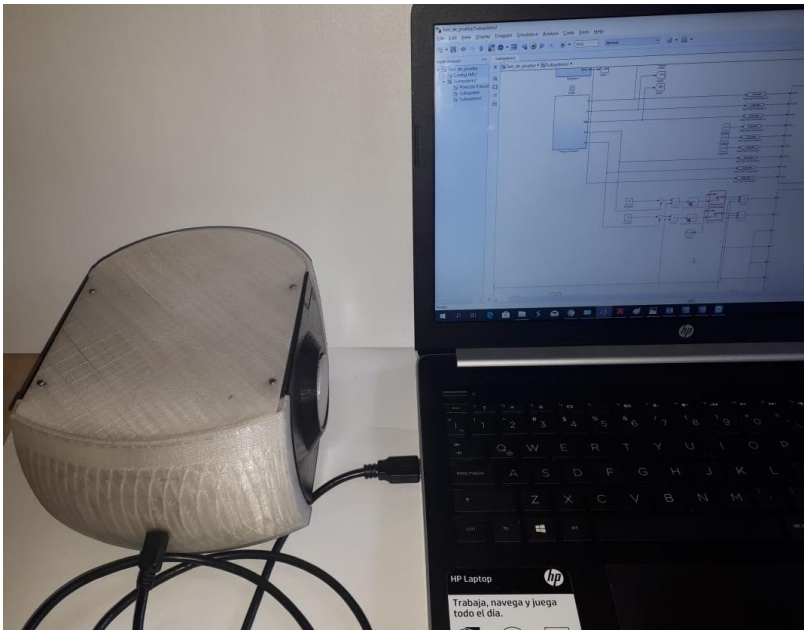


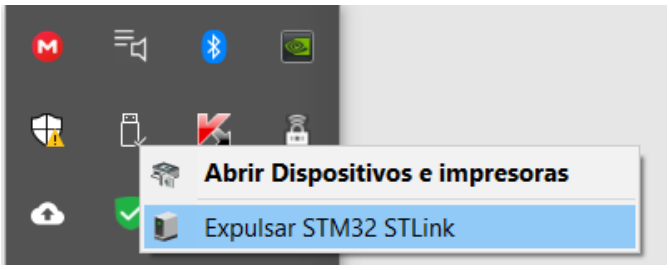
3. VINCULAR VISUAL STUDIO 2010

- Para la vinculación de click izquierdo en el siguiente enlace https://la.mathworks.com/videos/integrate-code-into-visual-studio-77402.html

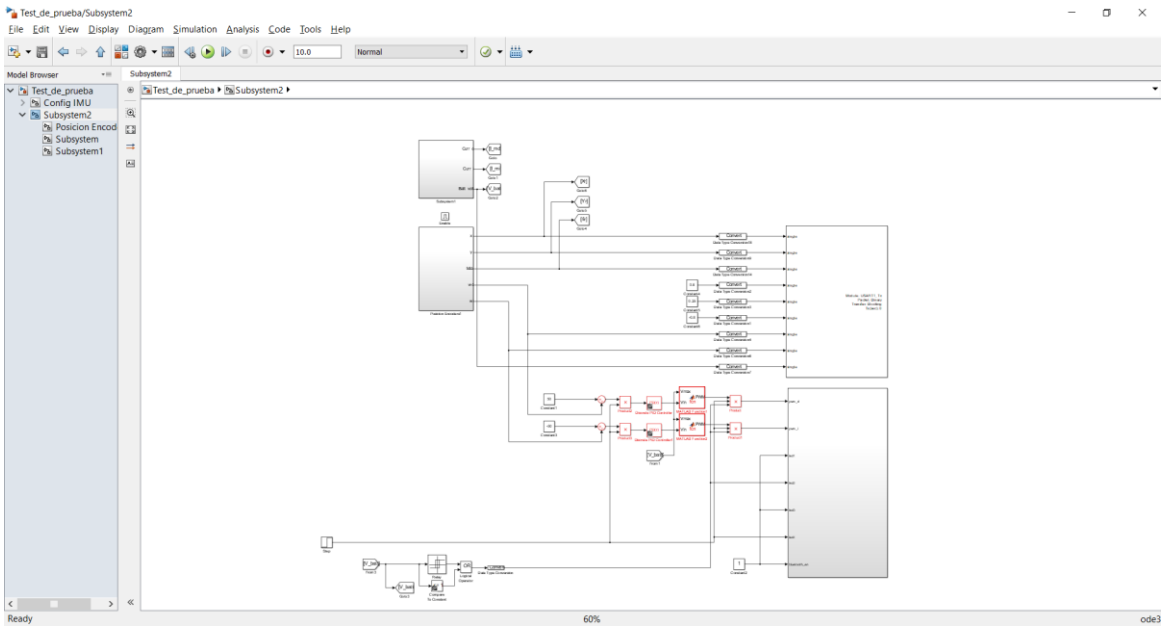
4. TEST DEL ROBOT

- Conecte el robot a la PC a través del puerto USB, y verifique que la PC reconozca el dispositivo como STM32 STLink como se muestra en la siguiente imagen:

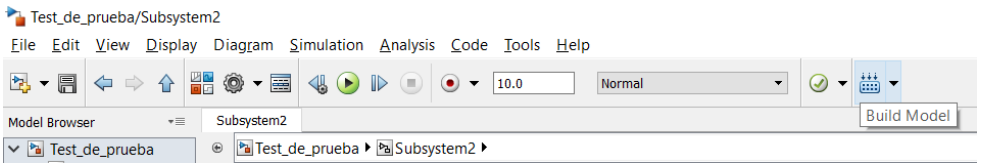




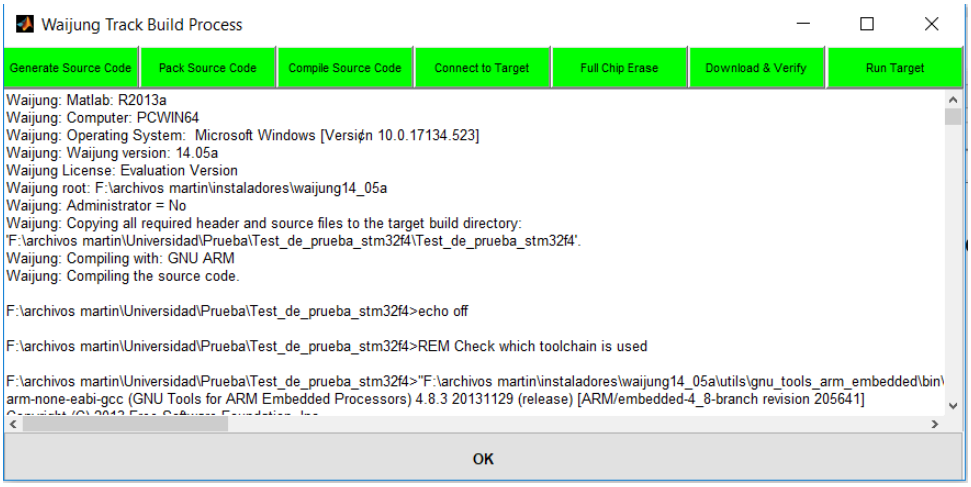
- Abra la carpeta “Test del robot” del CD entregado y abrir el archivo “Test de prueba” archivo Matlab.



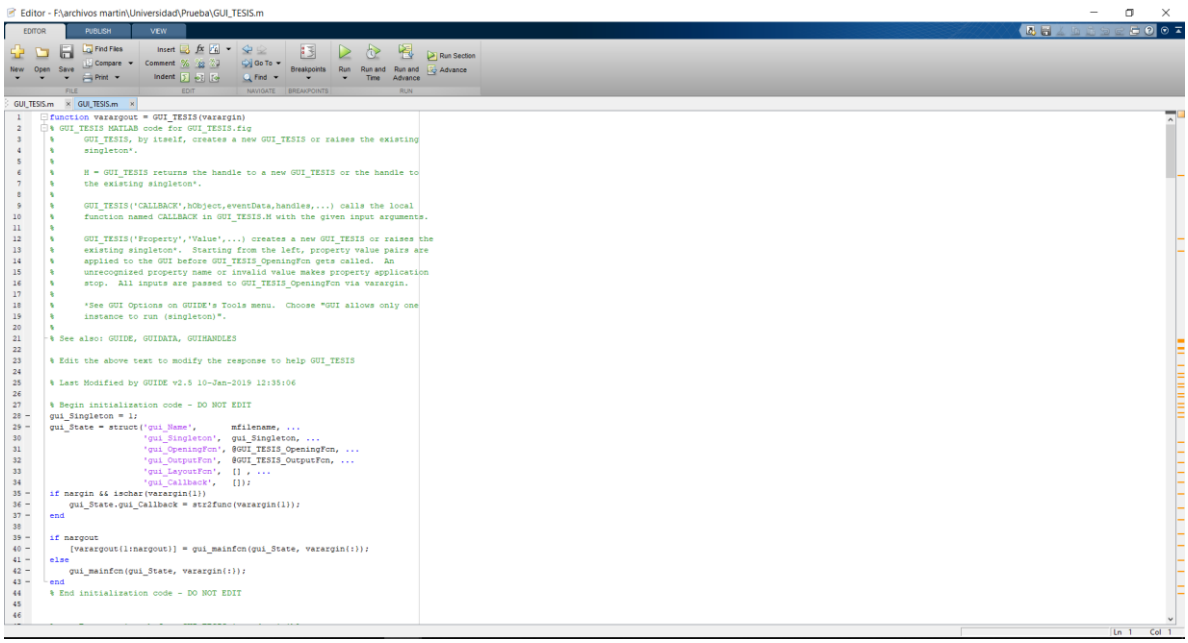
- Compilar el programa, dando click izquierdo en el ícono Build Model ubicado en la barra de tareas



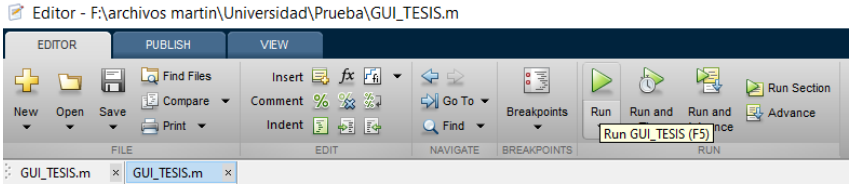
- Esperar a que termine el proceso de compilación del programa, la tarea es automática.

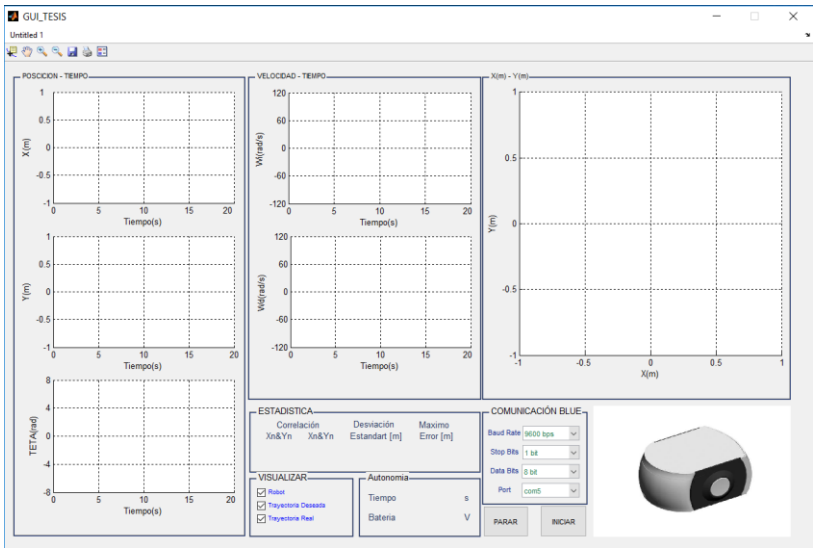


- Abrir desde Matlab el archivo “GUI_Tesis” de la carpeta “Test del robot/HMI”.

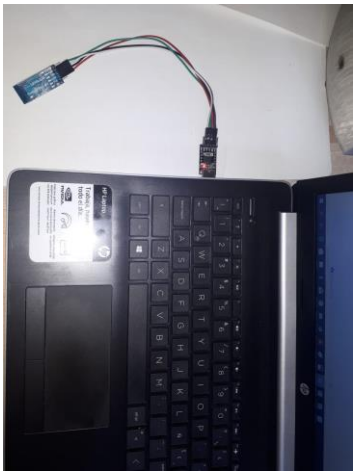


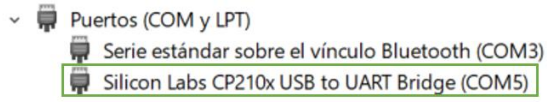
- Dar click izquierdo al ícono Run_GUI_TESIS de la barra de herramientas del MATLAB.



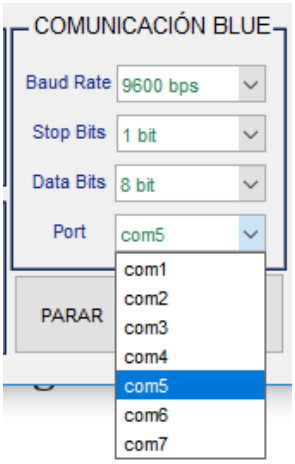


- Conecte el módulo de comunicación Bluetooth externo, verifique que se reconozca el dispositivo y en administrador de dispositivos Puertos (COM y LPT) y verifique el número de puerto COM asignado por su equipo.



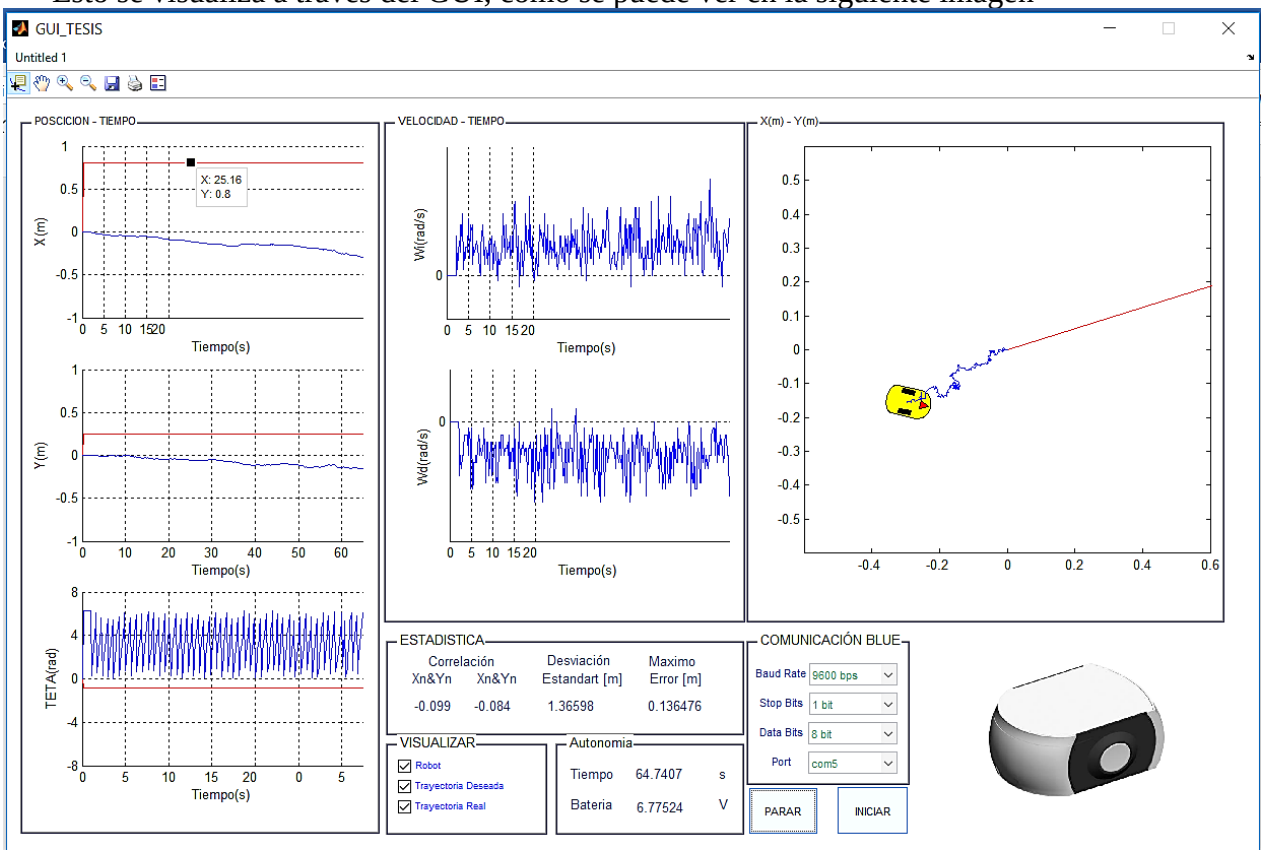


- En el puerto HMI se selecciona el puerto COM que se asignó para el módulo de comunicación.



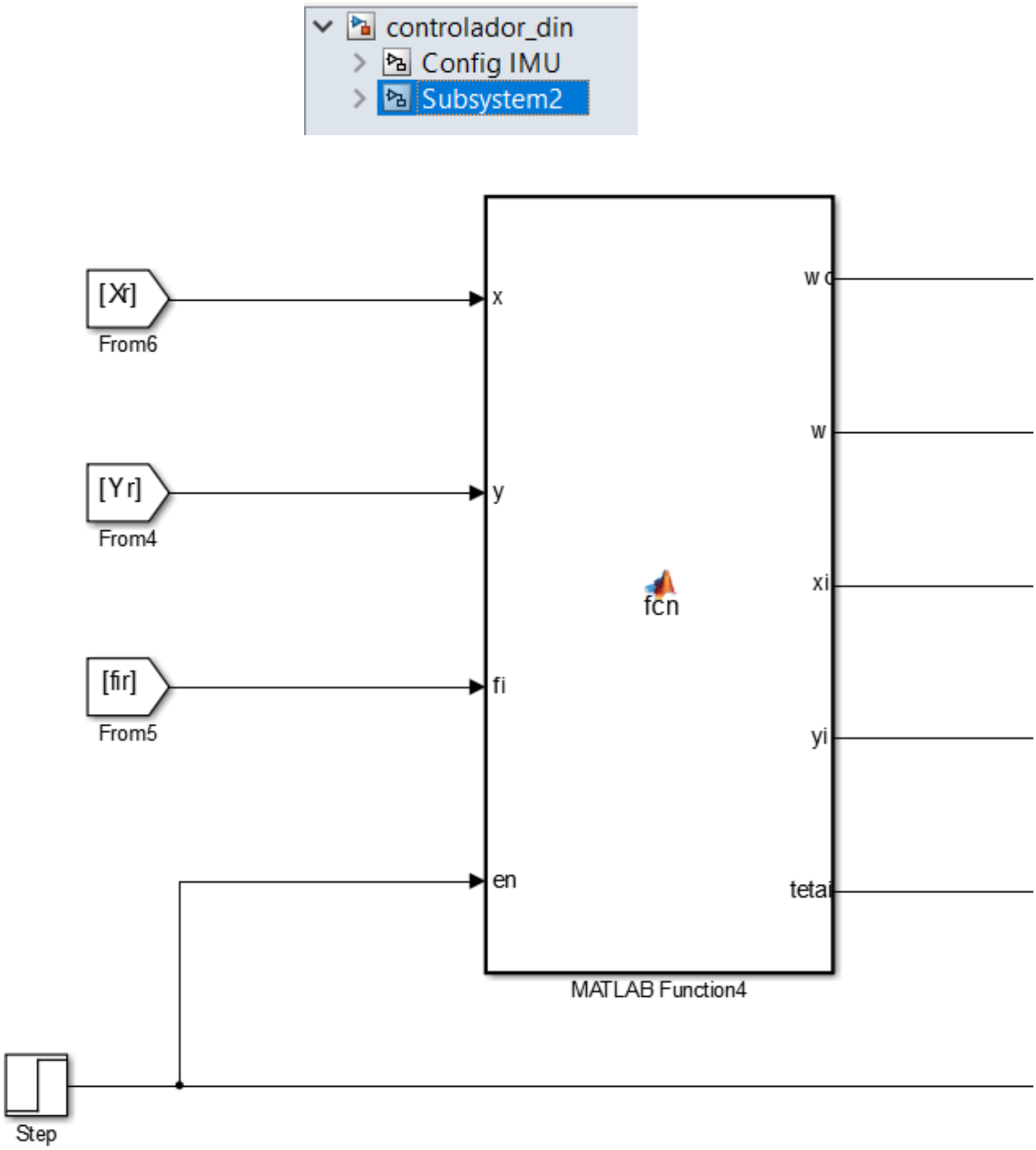
- Se coloca el robot en la superficie de pruebas, en el centro del área en las coordenadas (0,0,0).

- El robot empezará a girar en círculos debido a que las ruedas se desplazaran con velocidades angulares opuestas (giraran en sentidos contrarios).
- Esto se visualiza a través del GUI, como se puede ver en la siguiente imagen



5. PROGRAMACIÓN DE TRAYECTORIA

- Para la carga de la trayectoria seguir los pasos del ítem N°4 (Test de prueba) ya que es el mismo procedimiento, pero cargar la carpeta “Trayectorias” y “GUI_Tesis” desde el programa Matlab.
- Dar click izquierdo en Subsytem2 en el bloque MATLAB Function4 dar doble click izquierdo.



- El programa tiene preconfigurado diversas trayectorias de prueba, que pueden ser modificadas, o aumentadas según sus requerimientos. Desde la línea de comandos 8

hasta la línea 35 se visualiza las trayectorias con sus variables que permiten estabilidad al sistema.

```
 8        %S=0; Kp=0.4; Ki=0.2;  %para traytectorias circular y con uno y dos nodo
 9 -      S=0; Kp=0.15; Ki=0.02; %para linea recta
10        %S=0; Kp=0.15; Ki=0.008; %para parabolica
11        %Generacion de trayectorias
12 -      cv=25.0; Ts=0.01; Tf=300; t=0;
13 -      Xdn=0; Xdn1=0;
14 -      Ydn=0; Ydn1=0;
15 -      Fid=0;
16 -      Fidn=0; Fidn1=0;
17 -      k=0;
18 -      Wd=0;
19 -      Wi=0;
20      end
21
22
23 -    if(en==1)
24 -        if(t<Tf)
25
26              %Xdn1 = 0.4*cos(t/8); %trayectoria con un nodo componente x
27              %Ydn1 = 0.4*sin(t/4); %trayectoria con un nodo componente y
28              %Xdn1 = 0.4*cos(t/9); %trayectoria con dos nodo componente x
29              %Ydn1 = 0.4*sin(t/3); %trayectoria con dos nodo componente y
30              %Xdn1 = 0.4*cos(t/4); %circulo componente x
31              %Ydn1 = 0.4*sin(t/4); %circulo componente y
32 -            Xdn1 = 0.4*cos(t/4); %recta componente x
33 -            Ydn1 = 0.4*cos(t/4); %recta componente y
34              %Xdn1 = 0.4*sin(t/4); %parabola componente x
35              %Ydn1 = -0.4+0.8*(sin(t/4))^2; %parabola componente y
```

- Como comentario en la parte derecha se define el tipo de trayectoria a seguir, para seleccionar una trayectoria debe borrar el signo de porcentaje de las variables y funciones para que el programa reconozca la línea de comando, y en las demás trayectorias colocar al comienzo del texto los símbolos de porcentaje.

- Como ejemplo si se desea seguir una trayectoria circular:

```
%S=0; Kp=0.4; Ki=0.2;  %para trayectorias circular y con uno y dos
 nodo
S=0; Kp=0.15; Ki=0.02; %para línea recta
%S=0; Kp=0.15; Ki=0.008; %para parabólica

S=0; Kp=0.4; Ki=0.2;  %para trayectorias circular y con uno y dos
 nodo
%S=0; Kp=0.15; Ki=0.02; %para línea recta
%S=0; Kp=0.15; Ki=0.008; %para parabólica

    %Xdn1 = 0.4*cos(t/8); %trayectoria con un nodo componente x
    %Ydn1 = 0.4*sin(t/4); %trayectoria con un nodo componente y
    %Xdn1 = 0.4*cos(t/9); %trayectoria con dos nodos componente x
```

```
%Ydn1 = 0.4*sin(t/3); %trayectoria con dos nodos componente y
%Xdn1 = 0.4*cos(t/4); %circulo componente x
%Ydn1 = 0.4*sin(t/4); %circulo componente y
Xdn1 = 0.4*cos(t/4); %recta componente x
Ydn1 = 0.4*cos(t/4); %recta componente y
%Xdn1 = 0.4*sin(t/4); %parábola componente x
%Ydn1 = -0.4+0.8*(sin(t/4))^2; %parábola componente y

%Xdn1 = 0.4*cos(t/8); %trayectoria con un nodo componente x
%Ydn1 = 0.4*sin(t/4); %trayectoria con un nodo componente y
%Xdn1 = 0.4*cos(t/9); %trayectoria con dos nodos componente x
%Ydn1 = 0.4*sin(t/3); %trayectoria con dos nodos componente y
Xdn1 = 0.4*cos(t/4); %circulo componente x
Ydn1 = 0.4*sin(t/4); %circulo componente y
%Xdn1 = 0.4*cos(t/4); %recta componente x
%Ydn1 = 0.4*cos(t/4); %recta componente y
%Xdn1 = 0.4*sin(t/4); %parábola componente x
%Ydn1 = -0.4+0.8*(sin(t/4))^2; %parábola componente y
```

- Como se observa se borra el símbolo de porcentaje y se coloca el símbolo en las funciones que no se desean seguir, el robot ejecuta una trayectoria a la vez.

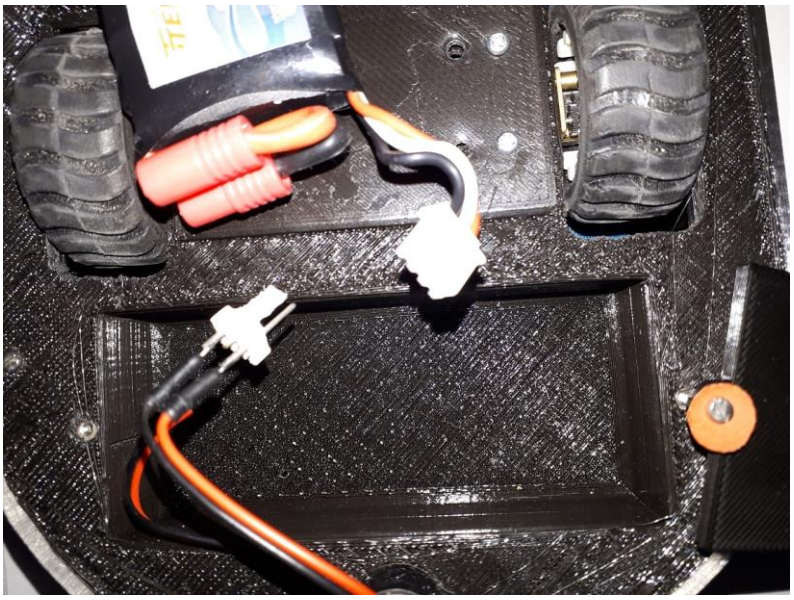
- Para la generación de trayectorias el robot realiza una suma de dos funciones Xdn1 y Ydn1, ya que el procesamiento de funciones sinusoidales es más sencillo para el microcontrolador.

- Después de este procedimiento se compila el programa, y se continua con los pasos del ítem 4.

6. CARGA DE LA BATERIA

- Se procede a desatornillar la tapa posterior del robot donde se encuentra localizada la batería.

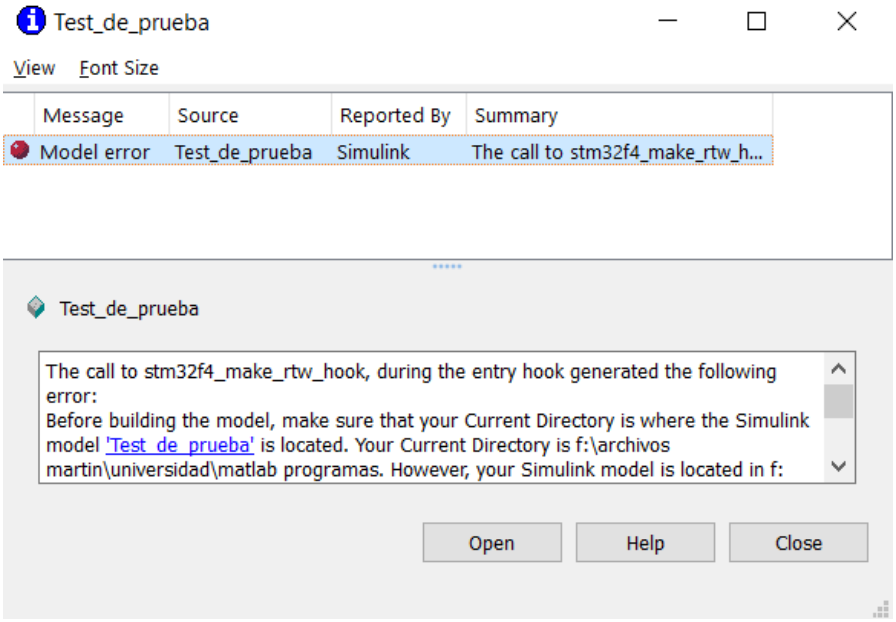
- Saque la batería con cuidado, y desconecte el molex como se indica en la siguiente figura.



- Conecte la batería al cargador en el molex apropiado, y espere hasta que las tres luces led indicadoras se pongan en verde.

POSIBLES ERRORES Y CORRECCIONES

Si al abrir el archivo del programa en MATLAB le aparece el siguiente cuadro de dialogo, es porque no cargó todos los archivos y el programa le pide ejecutar una extensión.



Si al ejecutar el archivo GUI no recibe los datos del Bluetooth cierre el programa, vuelva abrirlo, y verifiqué el puerto COM correcto del Bluetooth.