INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL

INTERDISCIPLINARIA DE INGENIERÍA

CAMPUS ZACATECAS

U P I I Z

Ingeniería Mecatrónica

Trabajo Terminal

“Sistema de Emulación de GPS en Entornos Cerrados para Monitoreo y Control de Vehículos A Escala”

Presenta:

Erik Alejandro Garay Rodríguez

Asesores:

M. en C. Flabio Dario Mirelez Delgado

M. en C. Omar Désiga Orenday

M. en P. y M. Jorge Talavera Otero

Zacatecas, Zac., junio de 2021

Folio
UPIIZ/ESA/113/2022

100 Aniversario de la Escuela Superior de Ingeniería y Arquitectura
50 Aniversario de la UPIICSA
50 Aniversario del CECyT 10 "Carlos Vallejo Márquez"
25 Aniversario del CIECAS, CIITEC y del CIIDIR, Unidad Sinaloa

Asunto
DESIGNACIÓN DE ASESORES ERIK ALEJANDRO GARAY RODRÍGUEZ
INGENIERÍA MECATRÓNICA
BOLETA: 201767045198
GENERACIÓN: 2017-2022

Zacatecas, Zac., a 25 de abril de 2022

C. ERIK ALEJANDRO GARAY RODRÍGUEZ
PRESENTE

Mediante el presente se hace de su conocimiento que este Departamento acepta que el **M. en C. Flabio Darío Mirelez Delgado, M. en C. Omar Désiga Orenday,** y el **M. en P.y M. Jorge talavera Otero** sean **Asesores,** en el tema que propone usted a desarrollar como prueba escrita de la opción Curricular, con el título y contenido siguiente:

"Sistema de Emulación de GPS en Entornos Cerrados para Monitoreo y Control de Vehículos a Escala".

Se concede un plazo de máximo de un año, a partir de esta fecha, para presentarlo a revisión por el jurado asignado.

SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO POLITÉCNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA CAMPUS ZACATECAS
DEPARTAMENTO DE EVALUACIÓN

Dra. Martha Monserrat Saldaña Noriega
Jefa del Departamento de Evaluación
y Seguimiento Académico

SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO POLITÉCNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA CAMPUS ZACATECAS
DIRECCIÓN

M. en C. Fernando Flores Mejía
Director de la UPIIZ

Folio
UPIIZ/ESA/114/2022

Asunto
AUTORIZACIÓN DE IMPRESIÓN DE TRABAJO DE TITULACIÓN
ERIK ALEJANDRO GARAY RODRÍGUEZ
INGENIERÍA MECATRÓNICA
BOLETA: 201767045198
GENERACIÓN: 2017-2022

Zacatecas, Zac., a 25 de abril de 2022

El suscrito tengo el agrado de informar a usted, que habiendo procedido a revisar el trabajo de titulación que presenta con fines de titulación denominada:

"Sistema de Emulación de GPS en Entornos Cerrados para Monitoreo y Control de Vehículos a Escala"

Encontré que el citado **Trabajo de Titulación,** reúne los requisitos para **autorizar** la impresión y proceder a la presentación del Examen Profesional debiendo tomar en consideración las indicaciones y correcciones que al respecto se hicieron.

M. EN C. FLABIO DARÍO MIRELEZ DELGADO

M. EN C. OMAR DÉSIGA ORENDAY

M. EN P. Y M. JORGE TALAVERA OTERO

Autorización de uso de obra

INSTITUTO POLITÉCNICO NACIONAL

P r e s e n t e

Bajo protesta de decir verdad **el** que suscribe **Erik Alejandro Garay Rodríguez,** estudiante del programa de **Ingeniería Mecatrónica**, con número de boleta **2017670451**, adscrito a la Unidad Profesional Interdisciplinaria de Ingeniería campus Zacatecas; manifiesto ser autor y titular de los derechos morales y patrimoniales de la obra titulada ***"Sistema de Emulación de GPS en Entornos Cerrados para Monitoreo y Control de Vehículos a Escala"*,** en adelante "El Trabajo de Titulación" y de la cual se adjunta copia, por lo que por medio del presente y con fundamento en el Artículo 27 Fracción II, inciso b) de la Ley Federal del Derecho de Autor, otorgo al Instituto Politécnico Nacional, en adelante el "IPN", autorización no exclusiva para comunicar y exhibir públicamente total o parcialmente en medios digitales "El Trabajo de Titulación" por un periodo indefinido contado a partir de la fecha de la presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso al "IPN" de su terminación.

En virtud de lo anterior, el "IPN" deberá reconocer en todo momento mi calidad de autor de "El Trabajo de Titulación".

Adicionalmente, y en mi calidad de autor y titular de los derechos morales y patrimoniales de "El Trabajo de Titulación", manifiesto que la misma esoriginal y que la presente autorización no contraviene a ninguna otra otorgada por el suscrito respecto de "El Trabajo de Titulación", por lo que deslindo de toda responsabilidad al "IPN" en caso de que el contenido de "El Trabajo de Titulación" o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumo las consecuencias legales y económicas de cualquier demanda o reclamación que puedan derivarse del caso.

Zacatecas, Zac., a 08 de abril del 2022.

Atentamente

Erik Alejandro Garay Rodríguez

Nombre y firma del alumno

INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL
INTERDISCIPLINARIA DE INGENIERÍA
CAMPUS ZACATECAS

U P I I Z

Ingeniería Mecatrónica

Trabajo Terminal

“Sistema de Emulación de GPS en Entornos Cerrados para Monitoreo y Control de Vehículos A Escala”

Diseño y construcción de dispositivos o componentes Mecatrónicos.

Presenta:

Erik Alejandro Garay Rodríguez

M. en C. Flabio Dario Mirelez Delgado

M. en C. Omar Désiga Orenday

M. en P. y M. Jorge Talavera Otero

M. en I. Adán Orenday Delgado

M. en I. Umanel Azazael Hernández González

Zacatecas, Zac., junio de 2021

ÍNDICE

Índice de Figuras

Índice de Tablas

AGRADECIMIENTOS

Agradezco al maestro Flabio Dario Mirelez Delgado por su apoyo tanto académico como personal para la realización del proyecto. Ha sido parte importante para la culminación del mismo y para mi formación académica y profesional.

Agradezco a mis asesores, al presidente del jurado evaluador y a los técnicos docentes de la institución por brindarme su apoyo y compartir su valioso conocimiento conmigo.

Agradezco a mi madre y mis hermanos, por ser mi amada familia y por estar siempre en los momentos en que más les necesito apoyándome con todos los retos que se me ponen enfrente en la vida.

A mis abuelos, por cuidar siempre de mí y permitir que mis sueños se hagan realidad.

Agradezco a Natalie, quien es mi compañera de aventuras y mi gran amor. Le agradezco por motivarme a cumplir mis metas y por el enorme amor que nos tenemos.

Agradezco a Monse, Lica, Akari, Osiris, Fátima y a Guillermo por ser mis grandes amigos y por brindarme total apoyo y compañía durante los años de universidad.

Agradezco a Anastasiia, Japjot, Meer, Vania y a Julio por ser mis amigos en la distancia y que, a pesar de ello, me acompañan y apoyan en todo momento.

DEDICATORIA

A mi madre, fuente eterna de sabiduría y amor que siempre me ha guiado, motivado y permitido perseguir mis sueños.

Resumen

En el presente documento se propone el diseño, construcción e implementación de un robot Cartesiano que en conjunto con una cámara emulará la función de un GPS actuando como observador en ambientes cerrados para el monitoreo de vehículos a escala. El prototipo resultante será capaz de posicionar una cámara RGB-D en diferentes posiciones de acuerdo con su espacio de trabajo para que la cámara abarque un campo de visión más amplio en comparación con una cámara fija en techo brindando así la posibilidad de realizar pruebas de seguimiento, evasión de obstáculos y planeación de trayectorias con vehículos autónomos para la futura implementación de sistemas de transporte inteligentes. La implementación de este tipo de tecnologías en las carreteras trae consigo un aliciente para la disminución de accidentes automovilísticos causados por errores humanos, así como la optimización en el transporte de mercancías y personal, significando un impacto positivo en la economía y el ahorro de combustible.

palabras clave

Autopistas automatizadas, GPS, monitoreo, robot cartesiano, vehículos autónomos.

Abstract

The following document advances the design, construction, and implementation of a Cartesian robot that, working together with a camera, emulates the GPS functionality when it acts as an observer in closed environments to monitor scaled vehicles. The resulting prototype will be able to position an RGB-D camera in different positions according to the manipulator's workspace to amplify the camera's visual field in comparison to fixed cameras in ceilings, providing the possibility to test autonomous-vehicles' tracking, obstacle avoidance, and path planning for further application of Intelligent Transportation Systems. Future implementation of these technologies in highways works as an incentive on the reduction of transit accidents due to human mistakes and even on optimization in wares and personnel transportation, causing a positive impact on the economy and fuel saving.

keywords

Automated highways, autonomous vehicles, Cartesian robot, GPS, monitoring.

Capítulo 1. Introducción

1.1. Objetivos

1.1.1. Objetivo general

Diseñar, construir e implementar un sistema robótico Cartesiano para emulación de GPS y monitoreo de vehículos a escala en un entorno cerrado.

1.1.2. Objetivos específicos o particulares

- ◇ Diseñar y construir un sistema de tipo Cartesiano con una cámara RGB-D integrada en el efector final que se desplace en el plano $x - y$ y que se pueda colocar en el techo de un salón o laboratorio.
- ◇ Diseñar e implementar un control cinemático para el efector final del robot Cartesiano.
- ◇ Añadir librerías de código abierto para capturar imágenes y así utilizar la cámara en conjunto con el sistema de control del robot Cartesiano.

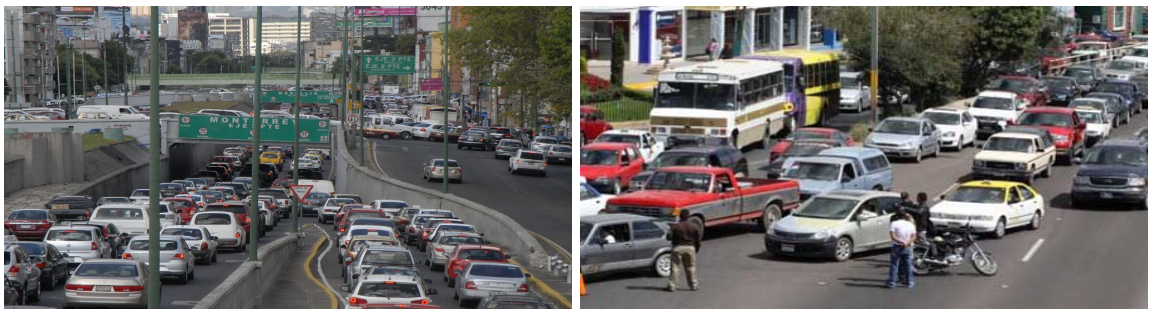
1.2. Justificación

Las nuevas tendencias tecnológicas apuntan hacia la utilización de vehículos autónomos, no sólo como medio de transporte seguro, sino también para logística. Basándose en los números dentro de México, según el INEGI, se reportaron en total 365,281 accidentes de tránsito terrestre en 2018 (véase la Tabla 1), de los cuales 4,227 casos resultaron en muerte de los pasajeros y 89,220 a heridos [1].

Tabla 1: *Accidentes de tránsito terrestre reportados en México del 2008 al 2018. Fuente: INEGI; elaboración propia.*

	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
Total	466,435	428,467	427,267	387,185	390,411	385,772	380,573	382,066	360,051	367,789	365,281
Conductor	381,293	384,942	389,026	366,133	371,718	362,118	355,759	353,640	327,706	336,018	337,266
Mala condición del camino	3,785	4,446	4,748	4,538	8,064	11,623	11,331	15,350	10,709	10,975	9,565
Falla del vehículo	3,519	4,337	4,181	3,537	2,843	2,677	2,985	2,750	3,883	3,566	3,567
Peatón o pasajero	3,975	4,773	4,573	4,735	3,612	3,069	3,324	3,516	3,244	2,958	3,089
Otra	73,863	29,969	24,739	8,242	4,174	6,285	7,174	6,810	14,509	14,272	11,794

Como puede apreciarse en la Tabla 1, en diez años se ha logrado una reducción significativa en el número de accidentes automovilísticos. Entre 2012 y 2015, el total de accidentes se redujo en aproximadamente 8 mil casos. Parte importante de ello fue el inicio de campañas de seguridad vial que incluían tecnología en las vialidades para monitorear en tiempo real a los usuarios y para elaborar sistemas de infracción por exceso de velocidad [2].



(a) CDMX, México. (b) Boulevard en Zacatecas, México.

Figura 1: Problemas del transporte carretero.

Como se puede observar en la figura 1, es muy común encontrarse en situaciones en las que las vialidades se encuentran completamente bloqueadas o que la densidad de tráfico es muy elevada. El flujo vehicular limitado y el congestionamiento vial afectan en gran medida las actividades diarias de las personas, la logística de las empresas y también pueden provocar accidentes viales.

Implementar Sistemas de Transporte Inteligentes (ITS, por sus siglas en inglés) es una fuente de oportunidad para la iniciativa privada y para la sociedad en general. Las empresas de capital privado pueden aprovechar la oportunidad de participar en proyectos que integren ITS pues la rentabilidad es elevada, sobre todo porque la mayoría de las aplicaciones pueden cobrarse a los usuarios, quienes adquieren los beneficios finales, pues la inversión que hacen en seguridad, menor tiempo de traslado y calidad de viaje hace que las ganancias se vean reflejadas en otros aspectos como el tiempo que dedican a otras actividades (productivas o no), ahorro en combustibles y reducción en los gastos del viaje [3].

Por los motivos expuestos anteriormente es que se propone el diseño y construcción de un prototipo que permita emular el monitoreo vehicular por GPS. La importancia de esto yace en que, en la actualidad, las principales empresas fabricantes de automóviles están centrando sus esfuerzos tecnológicos al desarrollo y mejoramiento de los vehículos autónomos, permitiendo así crear redes extensas de transporte inteligente tanto para mercancías como para individuos. El ahorro de combustible que esto representa impactaría positivamente en el medio ambiente y en el aspecto económico.

Capítulo 2. Marco teórico

2.1. Sistema de Posicionamiento Global (GPS)

El Sistema de Posicionamiento Global (GPS) es un servicio propiedad de los EE.UU. que proporciona a los usuarios información sobre posicionamiento, navegación y cronometría. Este sistema está constituido por tres segmentos: el segmento espacial, el segmento de control y el segmento del usuario. La Fuerza Aérea de los Estados Unidos desarrolla, mantiene y opera tanto el segmento espacial como el de control [4].

Figura 2: Imagen representativa de los datos que se pueden obtener con un GPS.

El GPS se compone de tres elementos: los satélites en órbita alrededor de la Tierra, las estaciones terrestres de seguimiento y control, y los receptores del GPS propiedad de los usuarios. Desde el espacio, los satélites del GPS transmiten señales que reciben e identifican los receptores del GPS; ellos, a su vez, proporcionan por separado sus coordenadas tridimensionales de latitud, longitud y altitud, así como la hora local precisa [4].

Hoy en día, los pequeños receptores del GPS portátiles están al alcance de todos. Con esos receptores, el usuario puede determinar con exactitud su ubicación y desplazarse fácilmente al lugar a donde desea trasladarse, ya sea andando, conduciendo, volando o navegando.

2.2. Sistemas de transporte inteligentes

Con las mejoras en cómputo y comunicaciones, las autopistas ITS fueron diseñadas para proveer una capacidad más integral de administrar el tráfico. Dichas autopistas hicieron un uso extensivo de cobertura por CCTV (Circuito Cerrado de Televisión) para monitorear el tránsito de los vehículos en tiempo real y su desempeño. En algunos casos, el servicio de monitoreo se extendió para apoyar la prioridad de vehículos de alta ocupación a través de la gestión de carriles HOV (*High-Occupancy Vehicle*)[1] y HOT (*High-Occupancy Toll*)[2] y derivaciones de medidores de rampa para vehículos de alta ocupación.

A finales de la década de 1990, se hizo mayor hincapié en la introducción de principios de ingeniería de sistemas en el diseño ITS de autopistas. Los principios de ingeniería de sistemas incluyen:

- Un proceso sistemático del ciclo de vida para establecer necesidades y objetivos, desarrollando un conjunto de alternativas de diseño, evaluando dichas opciones, diseñando el sistema e instalándolo, proporcionando los servicios logísticos necesarios para soportar el sistema, operarlo y evaluar su rendimiento con el fin de mejorar los futuros diseños y operaciones de ITS.

- Provisión de un proceso para integrar el sistema de autopistas en el sistema general de transporte, incluido el apoyo del tránsito, las operaciones de corredores, otros sistemas regionales de tráfico y gestión de transporte. El proceso enfatiza la interacción con otras partes interesadas.

- Emplear estándares aprobados para permitir la intercambiabilidad de los equipos de campo y para apoyar el intercambio de comunicación entre las partes interesadas y la comunicación con los equipos de campo [5].

[1] HOV: Carriles de alta ocupación.

[2] HOT: Carriles de peaje automático.

Desde fines de la década de los años 2000, el despliegue de las autopistas ITS es cada vez más limitado por la disponibilidad restringida de ayuda federal y fondos estatales equivalentes. Problemas como la valoración de que sea más rentable cubrir un área geográfica más amplia con tratamientos ITS o implementar dispositivos ITS con mayor intensidad en un área más pequeña, son consideraciones importantes para la planificación y el alcance del proyecto.

2.3. Robot Cartesiano

Un sistema robótico Cartesiano es un robot manipulador el cual sus tres primeras uniones son prismáticas. Para este tipo de manipuladores, las variables de las uniones están descritas por las coordenadas Cartesianas del efector final con respecto a la base. Los robots manipuladores Cartesianos son útiles para aplicaciones de ensamblaje de sobremesa y como robots de plataforma o tipo "pórtico" (*gantry robots*) para el transporte de material o carga [6].

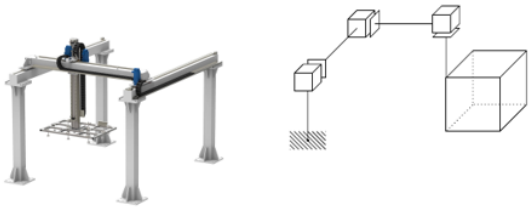


Figura 3: Ejemplo de robot Cartesiano y su espacio de trabajo.

2.4. Sensores RGB-D

Este tipo de sensores son cámaras de profundidad. Las cámaras normales capturan la luz que rebota sobre los objetos que se encuentran frente a ellas y transforman esa luz en una imagen que asemeja a lo que el ojo humano ve. Por otro lado, las cámaras de profundidad registran la distancia que existe entre los objetos que se encuentran frente a ellas (Figura 4) y éstas mismas utilizando luz infrarroja para crear una imagen que captura no sólo cómo luce el objeto en cuestión, también en dónde se encuentran en el espacio [7]. Un ejemplo de este tipo de cámaras es el sensor Kinect.

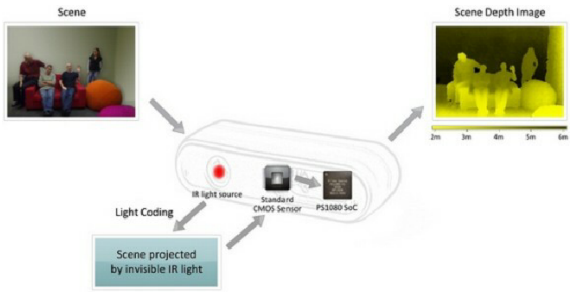


Figura 4: Estructura de un sensor RGB-D.

2.5. Sistemas de visión artificial

La visión artificial es un campo de estudio enfocado en el problema de ayudar a las computadoras a “ver”. En un nivel abstracto, la meta de los problemas que implican el uso de visión artificial es utilizar la información obtenida de la imagen observada para inferir algo sobre el mundo [8]. El objetivo de la visión por computadora es comprender el contenido de las imágenes digitales. Por lo general, esto implica desarrollar métodos que intenten reproducir la capacidad de la visión humana. Comprender el contenido de las imágenes digitales puede implicar extraer una descripción de la imagen, que puede ser un objeto, una descripción de texto, un modelo tridimensional, etc.

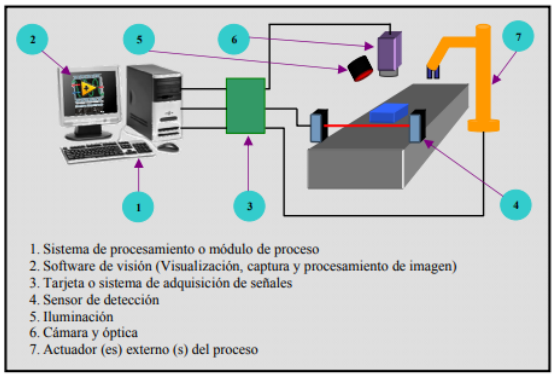


Figura 5: Sistemas de Visión Artificial y sus componentes.

Los sistemas de visión artificial se conforman tras implementar métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un ordenador. La estructura de este sistema se muestra en la Figura 5.

2.6. Retroalimentación visual (esquema “servo-visual”)

Uno de los métodos destacados en la navegación basada en la visión es el control servo-visual, que se desarrolló originalmente para manipuladores con una cámara en su efector final, pero también se ha aplicado a robots móviles no-holonómicos [9]. En algunos casos, el método se basa en la geometría del entorno y otra información métrica. En este caso, el control servo-visual basado en pose se usa para reducir el error que se estima en el espacio de pose. Otros sistemas de navegación visual no utilizan una representación explícita del entorno. En este caso, se pueden utilizar técnicas de servografía visual basadas en imágenes para reducir el error, el cual es medido directamente en la imagen [10]. La aproximación basada en la imagen elimina la necesidad de interpretar las imágenes, así como los errores debidos al modelado de la cámara y la calibración.

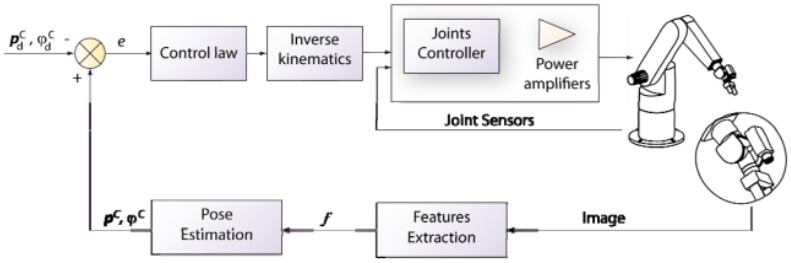


Figura 6: Esquema servo-visual basado en posición.

Capítulo 3. Antecedentes

En la Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas (UPIIZ) existe un trabajo de titulación que guarda relación con el presente proyecto. El proyecto al que se hace referencia se titula “Implementación de algoritmos de conducción autónoma en vehículos a escala 1:10”, fue realizado por Claudio Iván Esparza Castañeda y consta de un vehículo autónomo a escala que se desplaza sobre una pista de pruebas que simula un entorno urbano. La plataforma robótica está compuesta por un chasís comercial que lleva sobre sí un sistema de visión artificial basado en un sensor tipo Kinect [11].

El M. en C. Flabio Darío Mirelez Delgado presentó en 2015 el artículo “Control servovisual de un Kuka youBot para manipulación y traslado de objetos“ en el Congreso Nacional de Control Automático. La presentación fue en la ciudad de Cuernavaca, Mor., México. Y en el año 2017 redactó el capítulo “Consensus Strategy Applied to Differential Mobile Robots with Regulation Control and Trajectory Tracking“ que fue publicado en 2018 en el libro “Advanced Topics on Computer Vision, Control and Robotics in Mechatronics“ [12].

Estos antecedentes ayudarán a la realización del prototipo propuesto debido a que el proyecto realizado por Claudio Esparza puede ser utilizado en conjunto con el robot Cartesiano integrando el vehículo a escala con el sistema de visión en el efector final del robot para realizar pruebas en entornos cerrados. Así mismo, los trabajos presentados por el M. en C. Flabio Mirelez sirven como referente para tener una noción de los requerimientos funcionales del prototipo para su futura integración con sistemas servo-visuales.

Capítulo 4. Estado del arte

En la Universidad Autónoma de Zacatecas (UAZ) se realizó un proyecto a partir de la donación por parte del científico mexicano Raúl Rojas de un prototipo de vehículo autónomo que es modelo 1:10 de un auto real proveniente de Alemania. Entre los logros conseguidos, caben destacar las funcionalidades de autoestacionamiento, seguimiento autónomo de trayectorias y evasión de obstáculos.

En cuanto al monitoreo de carreteras, en el año 2014 se presentó un proyecto en Cochabamba, Bolivia. El nombre de éste es “VIRMS: Un sistema de información vehicular y monitoreo de carreteras” [2], el cual propone implementar un sistema de transporte inteligente basado en una infraestructura cliente-servidor de bajo costo y ligera que fue diseñada para mejorar la seguridad vehicular y reducir la tasa de accidentes en las carreteras.

Figura 7: Sistema embebido en tiempo real incorporado al interior del vehículo para monitoreo de su trayectoria.

En CINVESTAV Saltillo se realizó un proyecto llamado “Hierarchical Task-based Control of Multi-Robot Systems with Terminal Attractors”. En éste se utiliza una cámara para monitorear el comportamiento de múltiples robots móviles utilizando control basado en la realización de tareas [13].

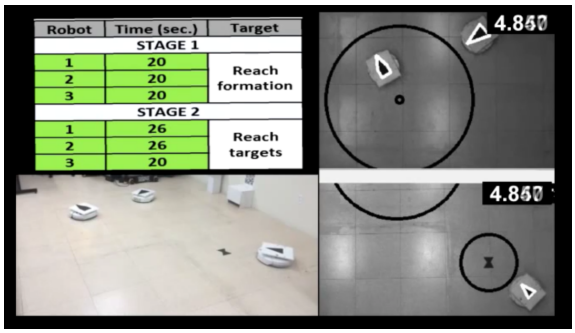


Figura 8: El movimiento de múltiples robots móviles está siendo monitoreado utilizando visión artificial.

Existe una empresa llamada Vicon que se dedica a fabricar sistemas de captura y análisis de movimiento, ya sea en la investigación, en el entretenimiento, en sistemas de realidad virtual o en “ciencias para la vida“ [14]. Este tipo de sistemas es muy preciso para realizar una gran variedad de pruebas, sin embargo, también tiene un costo muy elevado.

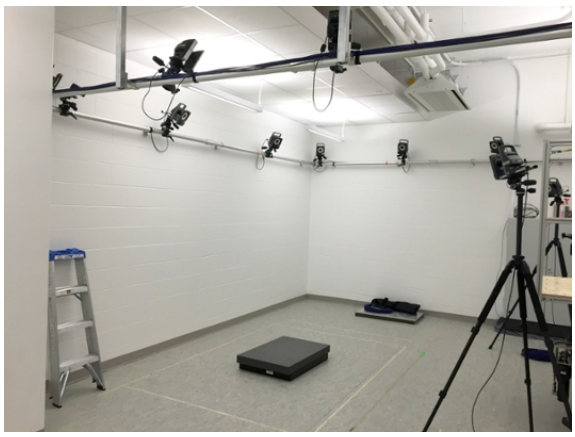


Figura 9: Configuración estándar de un sistema de captura de movimiento Vicon

De manera similar, existen también sistemas de captura de movimiento en tiempo real desarrollados por la empresa OptiTrack [15] cuyos sistemas ofrecen muchas características interesantes para investigadores y desarrolladores, pero también implica la inversión de una gran cantidad de dinero.

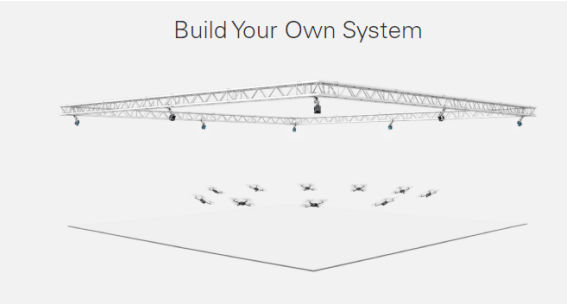


Figura 10: Sistema de captura de movimiento de OptiTrack

Capítulo 5. Planteamiento del problema

El problema a ser resuelto es la construcción e implementación de un sistema para monitorear vehículos autónomos a escala que emule la función del GPS en entornos cerrados, por lo que será necesario hacer que el sistema sea montable en el techo de un laboratorio. El entregable será un prototipo funcional de un robot Cartesiano con 80cm $\times$ 80cm de espacio de trabajo en el plano $x - y$ con un sistema de visión artificial integrado en el efector final. Este sistema será capaz de posicionar la cámara RGB-D en las coordenadas ingresadas numéricamente por el usuario desde una PC a través de la implementación de un control cinemático del manipulador. Además, una Raspberry Pi será capaz de enviar datos numéricos de posición al ordenador simulando el seguimiento de la trayectoria de un vehículo.

Capítulo 6. Desarrollo del trabajo

6.1. Diseño conceptual

Para poder profundizar en las etapas del diseño, se realizó primero la identificación de los requerimientos funcionales y técnicos del proyecto. Con base en ellos es que será más fácil determinar qué componentes son necesarios para el correcto funcionamiento del prototipo y garantizar que el diseño contemple aspectos de seguridad, la satisfacción del cliente (en este caso) y la facilidad en el mantenimiento del prototipo una vez culminado.

6.1.1. Requerimientos funcionales

A continuación se presenta un resumen de los requerimientos funcionales y su nivel de importancia. Posterior a este listado, se pueden encontrar las definiciones formales de los requerimientos funcionales con sus entradas y salidas.
Simbología: ♦ Deseable, ♦♦ Necesario, ♦♦♦ Crítico.

- El sistema debe ser capaz de operar en el techo de un salón o laboratorio. ♦♦♦
- Programar vía Bluetooth el sistema Cartesiano. ♦
- Implementar un control cinemático. ♦♦♦
- Proveer un modo manual de operación.♦♦♦
- Habilitar el modo automático de operación. ♦♦♦
- Interfaz Humano-Máquina (HMI).♦♦♦
- Acceso al dispositivo sólo por usuarios autorizados. ♦
- Mantener la cámara siempre paralela al piso.♦♦♦
- Control de inclinación de la cámara.♦

- ◇ Detectar final de carrera en el sistema Cartesiano.◆◆◆
- ◇ Proveer un paro de emergencia.◆◆◆
- ◇ Establecer comunicación entre el sistema Cartesiano y la estación de manera inalámbrica. ◆◆◆
- ◇ Cumplir con normas de seguridad en el laboratorio para aparatos/dispositivos que cuelgan del techo.◆◆
- ◇ Añadir circuitos de protección en el diseño electrónico.◆◆◆
- ◇ Realizar las tareas del sistema en tiempo real.◆
- ◇ Proveer al usuario con una manera simple de dar mantenimiento al prototipo.◆◆

6.1.2. Requerimientos técnicos

Con base en las actividades definidas en el cronograma se pueden definir los siguientes requerimientos técnicos:

- ◇ Computadora con memoria RAM recomendada de 8GB para utilizar los software de diseño y simulación.
- ◇ Software para realizar el diseño mecánico: SolidWorks (modelos 3D, simulaciones, planos), ANSYS (simulaciones), MATLAB (cálculos, gráficas y simulaciones).
- ◇ Software para diseño eléctrico y electrónico: OrCAD Capture CIS (diagramas y simulaciones), Proteus (diagramas y simulaciones); programa de dibujo para realizar diagramas de bloques.
- ◇ Software para diseño y simulación de sistemas de control: MATLAB.

6.2. Diseño mecánico

Para la parte mecánica del proyecto se requiere un mecanismo que sea capaz de trasladar a la cámara RGB-D en el plano $x - y$ implementando un control de posición realimentado. La configuración de este mecanismo se propone como

un sistema robótico Cartesiano, el cual tiene actuadores en los dos ejes de movimiento en el plano. Se muestra en la Figura 11 la configuración que se propone como solución para el aspecto mecánico.

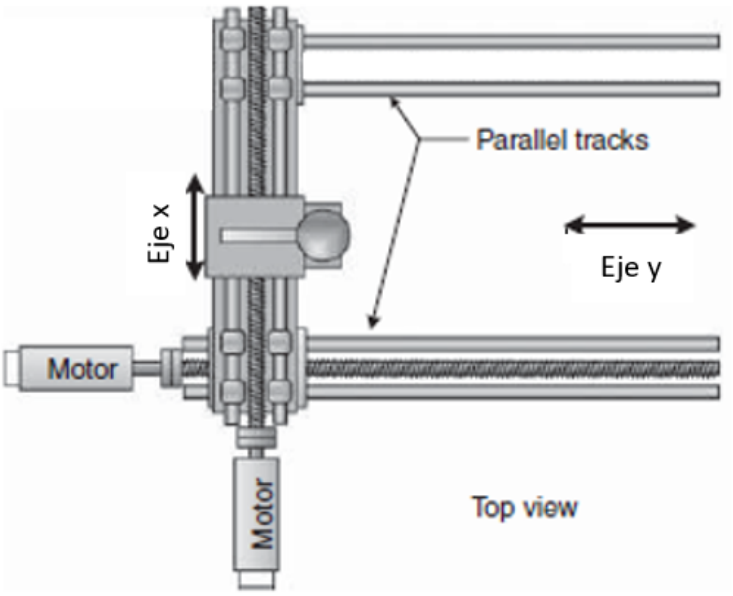


Figura 11: Sistema robótico Cartesiano visto desde arriba.

Con base en eso, se realizó un modelo 3D para el prototipo que será utilizado para operar en el techo del salón o laboratorio. Este modelo se presenta en la Figura 12, y de él se pueden destacar los siguientes componentes:

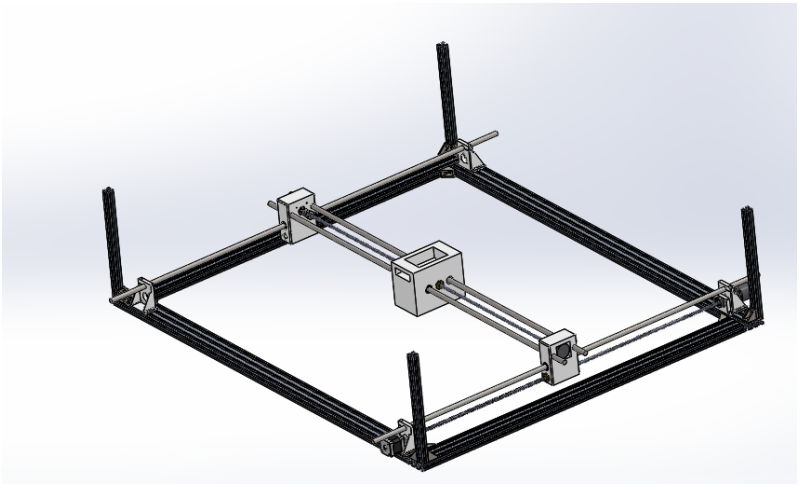


Figura 12: Modelo CAD de la propuesta de diseño.

6.2.1. Eje de movimiento en la coordenada y

Para efectuar el movimiento del efector final en el eje y, se propone utilizar un tornillo ACME de 8mm de diámetro con paso de 8mm como elemento de transmisión de potencia. Para garantizar que la trayectoria sea recta, se necesita añadir ejes como guía lineal para el mecanismo. En este caso, se plantea la posibilidad de utilizar un solo eje acerado con un diámetro mayor que el que se utilizaría con dos ejes guía.

Finalmente, se propone agregar cojinetes en cada extremo del eje, los cuales sirven para soportar las cargas axiales producidas por el movimiento del tornillo hacia cualquier dirección.

6.2.2. Elementos necesarios para el movimiento en el eje y

En la Figura 13 pueden apreciarse los elementos que se requieren utilizar en el carro para efectuar el movimiento. El tornillo requiere tener en conjunto una tuerca para desplazar la carga, en este caso, se propone utilizar dos tuercas, una hacia cada lado posible del movimiento para disminuir el retroceso y salto de pasos al cambiar la dirección de movimiento. Sin embargo, queda pendiente evaluar la utilidad de esta configuración.

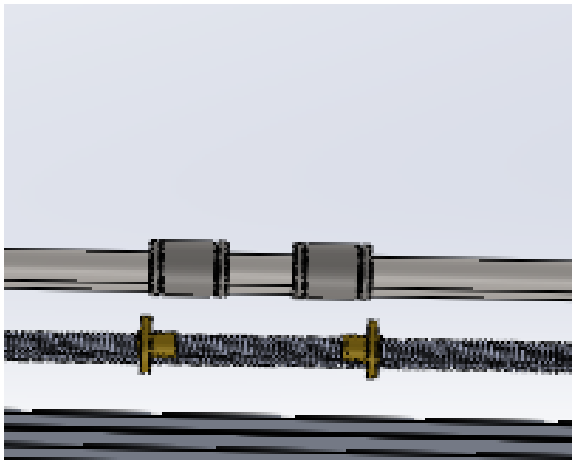


Figura 13: Rodamientos y tuercas dentro del carro del eje y.

Para reducir la fricción en la guía lineal, se propone agregar rodamientos lineales en el diseño, los cuáles ayudarán a reducir el desgaste en el eje acerado por

el movimiento, reduciendo la fricción y permitiendo que el movimiento sea más fluido.

6.2.3. Eje de movimiento x

En la Figura 14 se puede observar la configuración del eje x de movimiento, la cual es muy similar al eje y, pero se diferencia en que se utilizan dos ejes como guía lineal y se carece de rodamientos en los extremos.

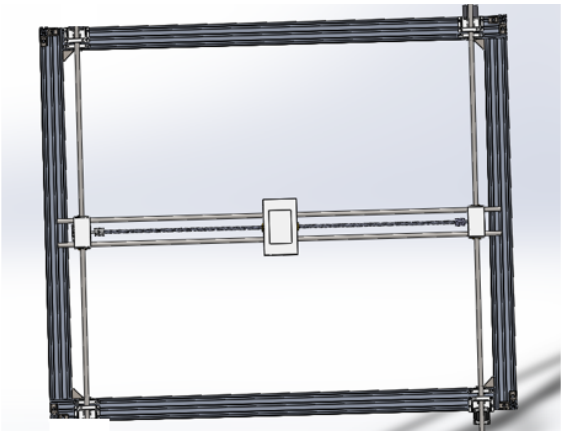


Figura 14: Vista superior del prototipo dónde se aprecia la configuración del eje x.

6.2.4. Espacio de trabajo

El espacio de trabajo del robot Cartesiano, el cual es un cuadrado de 80 x 80 centímetros. Este espacio se determinó a partir del rango de visión de la cámara a utilizar, la ASUS Xtion Pro Live, mostrado en la Figura 15.

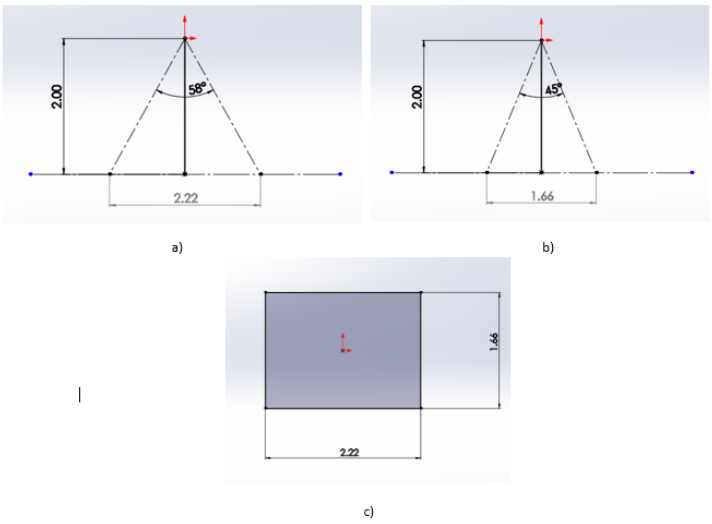


Figura 15: a) Campo de visión vertical. b) Campo de visión horizontal. c) Campo de visión total de la cámara ASUS Xtion Pro Live colocada a 2m de altura.

Al colocar la cámara a dos metros de altura, se obtiene el campo de visión total del dispositivo cuando se encuentra paralelo al piso, el cual se muestra en la Figura 15c. A partir de ahí, se suman al espacio de visión los 80cm de desplazamiento del robot Cartesiano, dando como resultado el campo de visión total del sistema de monitoreo mostrado en la Figura 16, el cual es un rectángulo de 3m x 2.45m.

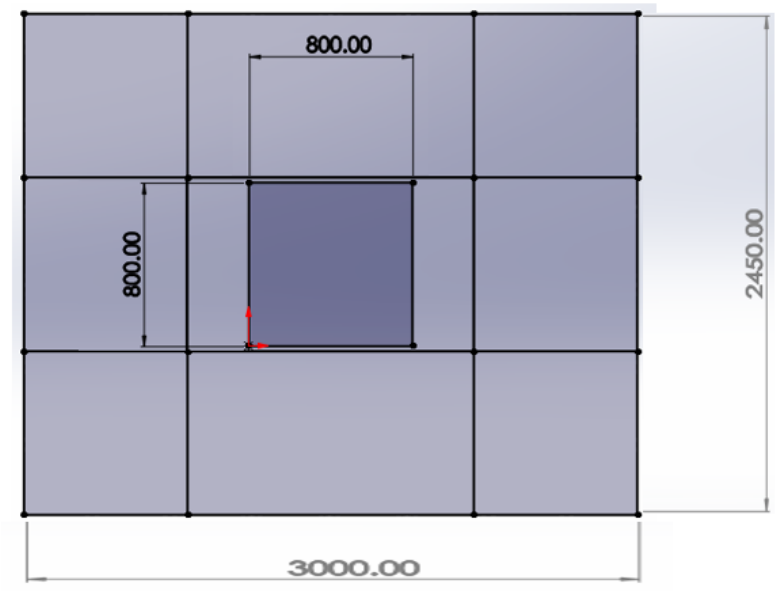


Figura 16: Espacio de visión total de la cámara (medidas en milímetros).

6.2.5. Velocidad crítica del tornillo ACME

La velocidad crítica [16] fue calculada a partir de la ecuación:

$$\omega_1 = \left(\frac{\pi}{l}\right)^2 \sqrt{\frac{gEI}{A\gamma}} \tag{1}$$

Donde ω_1 es la velocidad angular crítica para el tornillo, l es el avance del tornillo, E es el módulo de elasticidad del material, I es el momento de inercia, A es el área transversal del tornillo y γ es el peso específico del material. El resultado obtenido para la velocidad crítica fue de $\omega_1 = 1065.6rpm$. La mayoría de los motores a pasos adecuados para esta aplicación están limitados a $1000rpm$ como velocidad máxima. De igual manera, a esa velocidad o mayores, el tornillo comienza a verse afectado por los armónicos de las vibraciones. Por lo tanto, se toma la velocidad de $1000rpm$ como la velocidad crítica, la cual es utilizada para determinar la aceleración máxima permisible para el tornillo.

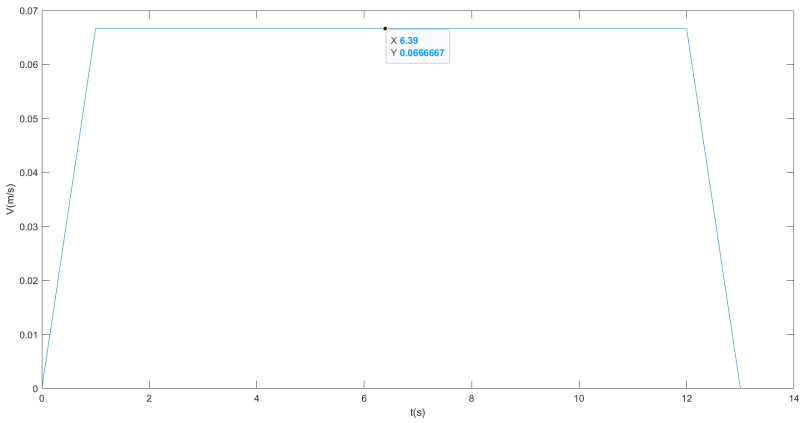


Figura 17: Perfil de velocidad del sistema.

En la Figura 17 se muestra el perfil de velocidad propuesto para una velocidad de operación deseada de $500rpm$. La distancia total del viaje es de $80cm$, por lo que el tiempo total desde el punto de inicio hasta el final de carreara es de $13s$.

6.2.6. Cálculo del par necesario para el movimiento

Para determinar la magnitud necesaria del par para vencer la fricción y la inercia de la carga es necesario tener en consideración distintos parámetros. Primero, la relación entre la inercia de la masa que está siendo trasladada y la inercia

reflejada está dada por:

$$J_{ref} = \frac{m}{N_s^2} \tag{2}$$

Donde $N_s^2 = 2\pi p$ es la relación de transmisión.

La inercia reflejada por el mecanismo a su eje de entrada está dada por:

$$J_{ref}^{trans} = J_{tornillo} + J_{carga \rightarrow in} + J_{carro \rightarrow in} = J_{tornillo} + \frac{1}{\eta N_s^2}\left(\frac{W_{carga} + W_{carro}}{g}\right) \tag{3}$$

Donde η es la eficiencia de la transmisión de potencia y $J_{tornillo}$ es la inercia del tornillo.

Es necesario incluir también la inercia que tiene el rotor del motor, la cual por lo general está especificada en la hoja de datos del fabricante. Además, también es necesario considerar la inercia del acoplador que se utiliza para transmitir el movimiento giratorio entre el eje del motor y el tornillo ACME [17]. Por lo tanto, la inercia total del sistema está dada por la siguiente ecuación:

$$J_{Total} = J_{motor} + J_{acoplador} + J_{ref}^{trans} \tag{4}$$

Las fuerzas exteriores que afectan al par de la carga reflejado en el motor tienen que sumarse para obtener el efecto total (incluyendo la fricción).

La fuerza de fricción se calcula con la ecuación

$$F_f = \mu \cdot (W_{carga} + W_{carro}) cos(\beta) \tag{5}$$

Donde β es el ángulo que existe entre la pista y la horizontal. De manera similar, la fuerza que actúa sobre el sistema por efectos de la gravedad puede encontrarse de

$$F_g = (W_{carga} + W_{carro}) \sin \beta \tag{6}$$

Así que la fuerza externa total que actúa sobre el sistema se determina a partir de

$$F_{ext} = F_f + F_g + F_p \tag{7}$$

Donde F_p son las fuerzas contra las que empuja el sistema en el proceso.

Así pues, el par de la carga reflejado en el motor está dado por

$$T_{carga \rightarrow in} = \frac{F_{ext}}{\eta N_s} \tag{8}$$

El par desarrollado durante la aceleración (valor pico) del sistema puede encontrarse a partir de

$$T_{aceleracion} = J_{Total} \cdot \ddot{\theta}_{motor} + T_{carga \rightarrow in} \tag{9}$$

El par durante el movimiento está dado por $T_r = T_{carga \rightarrow in}$ y el par durante la desaceleración cambia signos con el par necesario durante la aceleración del sistema, y queda como $T_{desaceleracion} = T_{carga \rightarrow in} - J_{Total} \cdot \ddot{\theta}_{motor}$.
Finalmente, el par continuo RMS se calcula a partir del par necesario en cada etapa del movimiento y es útil para determinar el par continuo necesario durante el tiempo total del movimiento. El par RMS se encuentra a partir de esta ecuación:

$$T_{RMS} = \sqrt{\frac{T_{aceleracion}^2 \cdot t_a + T_r^2 \cdot t_m + T_{desaceleracion}^2 \cdot t_d + T_{dw}^2 \cdot t_{dw}}{t_a + t_m + t_d + t_{dw}}} \tag{10}$$

Donde t_a, t_m, t_d y t_{dw} son el tiempo de aceleración, el tiempo durante el movimiento, el tiempo de desaceleración y el tiempo de mantenimiento, respectivamente.

Para calcular el par necesario para efectuar el movimiento, se utilizó un script de MATLAB con base en las ecuaciones mostradas anteriormente. El código del script se muestra en el Anexo 1.

Los resultados obtenidos que resultan relevantes son los que se presentan a continuación:

$$T_{aceleracion} = T_{pico} = 0.1708N \cdot m$$

$$T_{desaceleracion} = 0.051N \cdot m$$

$$T_r = 0.0537N \cdot m$$

$$T_{RMS} = 0.07N \cdot m$$

La gráfica del par calculado para cada etapa del movimiento se muestra en la Figura 18:

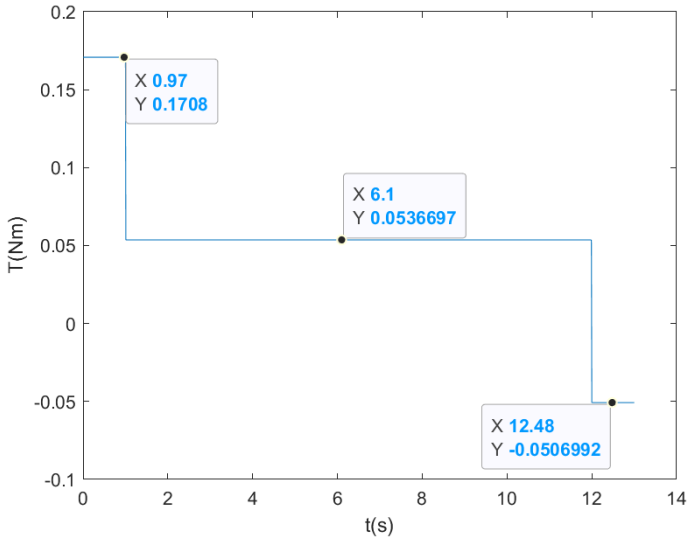


Figura 18: Gráfico de los pares desarrollados a lo largo del movimiento.

6.2.7. Vibraciones en la cámara

Al estar el sistema entero en constante movimiento, la cámara se ve expuesta a vibraciones provenientes de distintas fuentes. El efecto de las vibraciones impacta negativamente en el desempeño total del sistema, ya que haría que la captura de la imagen fuera mucho más complicada de obtener y procesar.

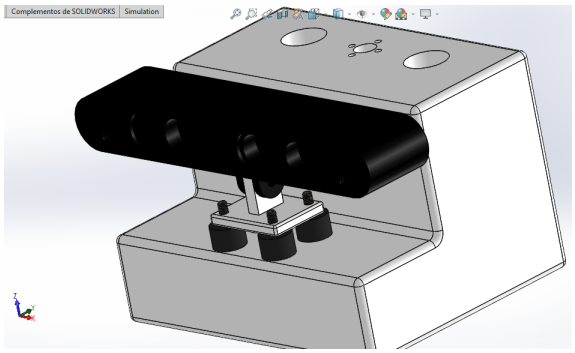


Figura 19: Montaje de amortiguadores pasivos para reducir el efecto de las vibraciones.

Para evitar esto, es necesario utilizar amortiguadores pasivos en el ensamble como se muestra en la Figura 19. El uso de este elemento proporciona una manera de mitigar el efecto de las vibraciones en la cámara, logrando así que la imagen capturada sea más clara y facilite el posterior procesamiento de la imagen.

De tal forma que al incluir estos amortiguadores, se puede modelar a la cámara como un sistema masa-resorte. Para colocar la cámara en el efector final, se remueve la base que viene por defecto para la ASUS Xtion Pro Live y se hace una base personalizada para esta aplicación. Esa base se sujetará en el carro del efector final utilizando soportes aisladores de vibración.

Otra forma de reducir las vibraciones en el sistema es utilizar la configuración de micropasos del motor a pasos. Esta opción se discute en la sección de Selección de motores.

6.2.8. Cálculo de esfuerzos

Para el cálculo de los esfuerzos se utiliza el script de MATLAB que se encuentra en el Anexo 1. Primero se realiza un análisis a los tornillos de potencia utilizados en los ejes de movimiento. El esfuerzo cortante τ debido al momento de torsión T_{pico} en la parte de afuera del cuerpo del tornillo es:

$$\tau = \frac{16T_{pico}}{\pi d_r^3} \tag{11}$$

Donde d_r es el diámetro menor de la rosca. El esfuerzo axial nominal está dado por:

$$\sigma = -\frac{4F}{\pi d_r^2} \tag{12}$$

Donde F es la fuerza de reacción provocada por la carga. El esfuerzo en el rodamiento con un hilo cargando $0.38F$

$$\sigma_B = -\frac{2(0.38F)}{\pi d_m(1)p} = -1.02MPa \tag{13}$$

Donde d_m es el diámetro de paso y p es el paso del tornillo. El esfuerzo de flexión en la raíz del hilo con un hilo cargando $0.38F$

$$\sigma_b = \frac{6(0.38F)}{\pi d_r(1)p} \tag{14}$$

Los esfuerzos principales son entonces:

$\sigma_x = 3.57MPa$	$\tau_{xy} = 0$
$\sigma_y = -2.087MPa$	$\tau_{yz} = 4.027MPa$
$\sigma_z = 0$	$T_{zx} = 0$

Y el esfuerzo máximo se encuentra a partir de:

$$\tau_{max} = \frac{\sigma_1 - \sigma_3}{2} = \frac{3.57 - (-5.20)}{2} = 4.38MPa \tag{15}$$

6.3. Selección de material

6.3.1. Estructura principal

El material necesario para construir la estructura principal del sistema Cartesiano debe de tomar en cuenta que el material debe de ser ligero para brindar la posibilidad de colocar el sistema en el techo de un laboratorio. Además, deben de procurarse la estética, el profesionalismo, el mantenimiento necesario del material y el periodo de vida estimado. Con base en esas premisas, se realizó la Tabla 2.

Tabla 2: *Selección de material para la estructura principal.*

Material	Costo	Peso	Disponibilidad	Estética	Consideraciones técnicas
Aluminio Estructural V-SLOT	20mmx60mmx1000mm: $542.60 MXN	Ligero	Media	5	Es de fácil montaje con uniones apernadas. Permite utilizar tuercas tipo T para montaje de elementos extra. Fácil maquinado adicional.
Perfil tubular galvanizado	25.4mmx50.8mmx6000mm: $348.76 MXN	Ligero	Alta	3	Se requieren uniones permanentes (soldadura). Mayor cantidad de procesos de manufactura. No requiere mantenimiento.
Madera	PANEL DE MDF 18 MM 1.22 X 2.44 M $529 MXN	Ligero	Alta	2	Fácil maquinado. Se pueden utilizar uniones no permanentes. Fácil montaje. Se requiere mantenimiento periódico.

La escala utilizada para valorar la estética del sistema según el material a utilizar es de 0 a 5, donde 0 es *Nada Estético* y 5 es *Muy Estético*.

Así, el material que presenta mayores ventajas y características requeridas para la realización del proyecto es el aluminio estructural con perfil tipo V-Slot. La gran

ventaja de utilizar este material es que permite realizar prototipos más portables, livianos, fáciles de ensamblar, resistentes y estéticos.

6.3.2. Acoplamiento de ejes

Para lograr la transmisión del movimiento, es necesario acoplar el eje del motor con el eje de transmisión. Para realizarlo, se contemplan dos posibilidades: utilizar un acoplamiento fijo o, uno flexible. La Tabla 3 fue utilizada para determinar el elemento a utilizar como acoplamiento.

Tabla 3: *Selección del acoplamiento para los ejes de movimiento.*

Nombre	Material	Costo	Disponibilidad	Consideraciones técnicas
Acoplamiento Flexible	Aluminio 7075	$100-$200 MXN	Alta	Al ser flexible, se mantiene la transmisión compensando la desviación radial y angular.
Acoplamiento rígido	Aluminio / Plástico	$50-$120 MXN	Alta	Se utilizan cuando el eje se posiciona con precisión tanto radial como angularmente, ya que no pueden compensar la desalineación.

Con base en el análisis anterior, se determina que la mejor opción es utilizar acoplamientos flexibles, ya que permiten cierta tolerancia ante la desalineación de los ejes en movimiento, lo que elimina esfuerzos indeseados en los elementos del ensamble y compensa el desalineamiento ya sea angular, excéntrico o combinado.

6.3.3. Selección de microcontrolador

El microcontrolador a utilizar tiene que ser rápido para poder enviar y recibir datos de manera eficaz y eficiente, debido a que el sistema se comunica inalámbricamente con la estación, que a su vez recibe datos de los vehículos que se están monitoreando. Se analizan las distintas posibilidades de utilizar tarjetas de desarrollo que sean fáciles de ensamblar (tamaño reducido) y que tengan facilidad de comunicarse de manera inalámbrica. A través de la tabla 4 se realiza un análisis de la mejor opción para el prototipo:

Tabla 4: *Selección del microcontrolador.*

Especificaciones \ Tarjeta	Arduino Nano	STM32F103C8T6	ESP32
CPU	ATmega328P	Arm Cortex-M3	Xtensa LX6
Núcleos	1	1	2*
Arquitectura	8 bits	32 bits	32 bits
Frecuencia CPU	16 MHz	72 MHz	160 MHz
Wireless	No.	No (requiere módulo externo).	WiFi, Bluetooth
RAM	2kB	20 kB	512 kB
Flash	32kB	64 kB	4 MB
Pines GPIO	14	37	36
SPI	1	2	4
I2C	1	2	2
I2S	0	0	2
UART	1	3	3
ADC Res/Ch	6*10-bit	10*12-bit	6*12-bit
Pines DAC	0	0	2
Precio	$70-$100 MXN	$ 70-$90 MXN (sin programador) $130 - $150 MXN (con programador)	$120 - $180 MXN
Dimensiones	39x13x12 mm	53x22.5 mm	51x27 mm

Comparando las distintas funcionalidades de cada tarjeta se determina que el mejor microcontrolador (denotado por μC) para este trabajo es el STM32F103C8T6 (de ahora en adelante, se referirá a este μC como STM32 solamente). El STM32 tiene varias funcionalidades adicionales con respecto a las otras tarjetas que resultan útiles para esta aplicación en particular.

Los timers del STM32 tienen un modo especial para leer encoders, lo cual acelera mucho el proceso de medición al dejar encargado al Hardware de esta tarea y liberando al μC de tener que realizar interrupciones para el conteo de pulsos.

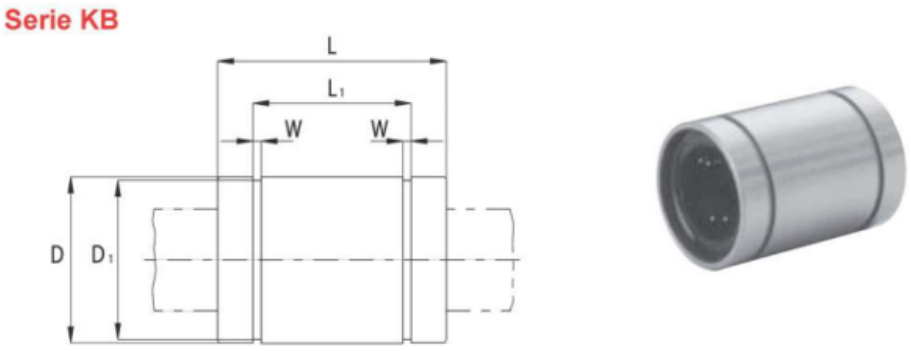
Además, tiene dos funcionalidades adicionales que se dejan como opcionales para este proyecto, las cuales son la capacidad que tiene el microcontrolador para ser programado por Bluetooth y la otra, que es posible instalar un sistema operativo en tiempo real para ejecutar todas las tareas del sistema con base en un itinerario programado. La primera funcionalidad adicional sirve para hacer más fácil al usuario dar mantenimiento o corregir el código del sistema, ya que

sólo bastaría con mover un pin de lugar a la hora de hacer la programación en lugar de desensamblar toda la parte que cubre al μC. La segunda, por su parte, brindaría aún más velocidad al sistema, permitiendo resolver esta problemática común para este tipo de sistemas de monitoreo.

Sin embargo, se reitera el carácter opcional de la adición de dichas funcionalidades al prototipo, ya que aumentan en un grado considerable la complejidad de la programación del mismo y podría no terminarse a tiempo la integración e implementación del sistema entero.

6.3.4. Selección de rodamientos lineales

La compra de los rodamientos lineales se hizo a través de internet y se hizo el rastreo del distribuidor hasta llegar a la compañía Rodavigo®. En su página de internet se encuentran los catálogos de productos y los cálculos recomendados para la selección.



Código	N° de hileras de bolas	Peso [N]	d [mm]	d Tolerancia [µm]	D [mm]	D Tolerancia [µm]	L [mm]	L Tolerancia [mm]
113KB0522	4	0.12	5	+8 ÷ 0	12	0 ÷ -8	22	0 ÷ -0.2
113KB0825	4	0.20	8	+8 ÷ 0	16	0 ÷ -8	25	0 ÷ -0.2
113KB1029	4	0.24	10	+8 ÷ 0	19	0 ÷ -8	29	0 ÷ -0.2
113KB1232	4	0.41	12	+8 ÷ 0	22	0 ÷ -9	32	0 ÷ -0.2
113KB1636	5	0.57	16	+9 ÷ -1	26	0 ÷ -9	36	0 ÷ -0.2
113KB2045	5	0.91	20	+9 ÷ -1	32	0 ÷ -11	45	0 ÷ -0.2
113KB2558	6	2.15	25	+11 ÷ -1	40	0 ÷ -11	58	0 ÷ -0.3
113KB3068	6	3.25	30	+11 ÷ -1	47	0 ÷ -11	68	0 ÷ -0.3
113KB4080	6	7.05	40	+13 ÷ -2	62	0 ÷ -13	80	0 ÷ -0.3
113KB50100	6	11.3	50	+13 ÷ -2	75	0 ÷ -13	100	0 ÷ -0.3
113KB60125	6	22.2	60	+13 ÷ -2	90	0 ÷ -15	125	0 ÷ -0.4

Figura 20: Medidas y número de parte del rodamiento lineal a utilizar.

Como se observa en la Figura 20, el rodamiento lineal cuenta con 4 hileras de bolas y es cerrado. La utilización de este tipo de rodamiento requiere que el eje

lineal que se utiliza sea endurecido y rectificado porque las bolas tienen un área de contacto de punto muy pequeña que permite una fricción de rodadura muy baja, y el tener un área de contacto pequeña significa que estos ejercen una gran fuerza sobre el eje.

Aplicar este tipo de rodamientos lineales aporta distintas ventajas como:

- Reducción de la fricción y el ruido.
- Capacidades de alta carga y larga vida útil.
- Estructura simple y alta fiabilidad.
- Alto rendimiento y capacidad de reemplazo.
- Lubricación fácil (se puede aplicar tanto aceite como grasa).

Código	L1 [mm]	L1 Tolerancia [mm]	W [mm]	D1 [mm]	máxima excentricidad [µm]	Juego radial tolerancia [µm]	Capacidad de carga [N]	
							C	C_0
113KB0522	14.5	0 ÷ -0.2	1.1	11.5	12	-5	210	270
113KB0825	16.5	0 ÷ -0.2	1.1	15.2	12	-5	270	410
113KB1029	22	0 ÷ -0.2	1.3	18	12	-5	370	470
113KB1232	22.9	0 ÷ -0.2	1.3	21	12	-7	520	790
113KB1636	24.9	0 ÷ -0.2	1.3	24.9	12	-7	590	910
113KB2045	31.5	0 ÷ -0.2	1.6	30.3	15	-9	880	1400
113KB2558	44.1	0 ÷ -0.3	1.85	37.5	15	-9	1000	1600
113KB3068	52.1	0 ÷ -0.3	1.85	44.5	15	-9	1600	2800
113KB4080	60.6	0 ÷ -0.3	2.15	59	17	-13	2200	4000
113KB50100	77.6	0 ÷ -0.3	2.65	72	17	-13	3900	8100
113KB60125	101.7	0 ÷ -0.4	3.15	86.5	20	-16	4800	10200

Figura 21: Especificaciones técnicas de carga.

Además, el tipo de rodamiento que provee el fabricante compensa automáticamente los errores de alineación de hasta 0,5 grados que se generan entre la carcasa y el eje sin una reducción de la capacidad de carga por presión en los extremos. Para el cálculo de la vida, se consideran los parámetros definidos en la Figura 21.

Se toma en cuenta que la carga perpendicular a los dos ejes del carro es de 196N como máximo (considerando 20kg de masa total). Además, se supone que

la carga está uniformemente repartida sobre los tres rodamientos del diseño. El carro hace recorridos de ida y vuelta en un tramo de $s = 0.80m$ considerando una frecuencia promedio de $n_s = 3\frac{carrerasdobles}{min}$.Este dato se obtiene considerando que se toman alrededor de 13 segundos para llevar al carro de un extremo a otro, sumando un total de 26 segundos por una carrera doble. En un minuto, se hacen $60/26 = 2.31$ carreras dobles, pero se redondean hacia arriba por la posibilidad de aumentar la velocidad del motor y para tomar en cuenta el peor caso posible.

Se considera que la duración de vida mínima sea de $L_H = 8000$ horas. La temperatura de servicio está entre 0℃ y 80℃. Para las condiciones de aplicación se toman en cuenta pocos golpes y vibraciones.

Ya que la carga está uniformemente repartida sobre los tres rodamientos lineales, se deberá considerar una carga para cada rodamiento de

$$F_m = \frac{196N}{3} = 65.33N$$

Donde F_m es la carga dinámica equivalente (para cada rodamiento). La duración de vida L de todo el recorrido, expresada en metros, se calcula de la siguiente manera:

$$L = 2 \cdot s \cdot n_s \cdot 60 \cdot L_H \tag{16}$$

Donde L es la duración de vida nominal en minutos, L_H es la duración de vida nominal en horas, s la longitud de carrera y n_s la frecuencia de carrera. El cálculo resulta en

$$L = 2 \cdot 0.80 \cdot 3 \cdot 60 \cdot 8000 = 23 \cdot 10^5 minutos$$

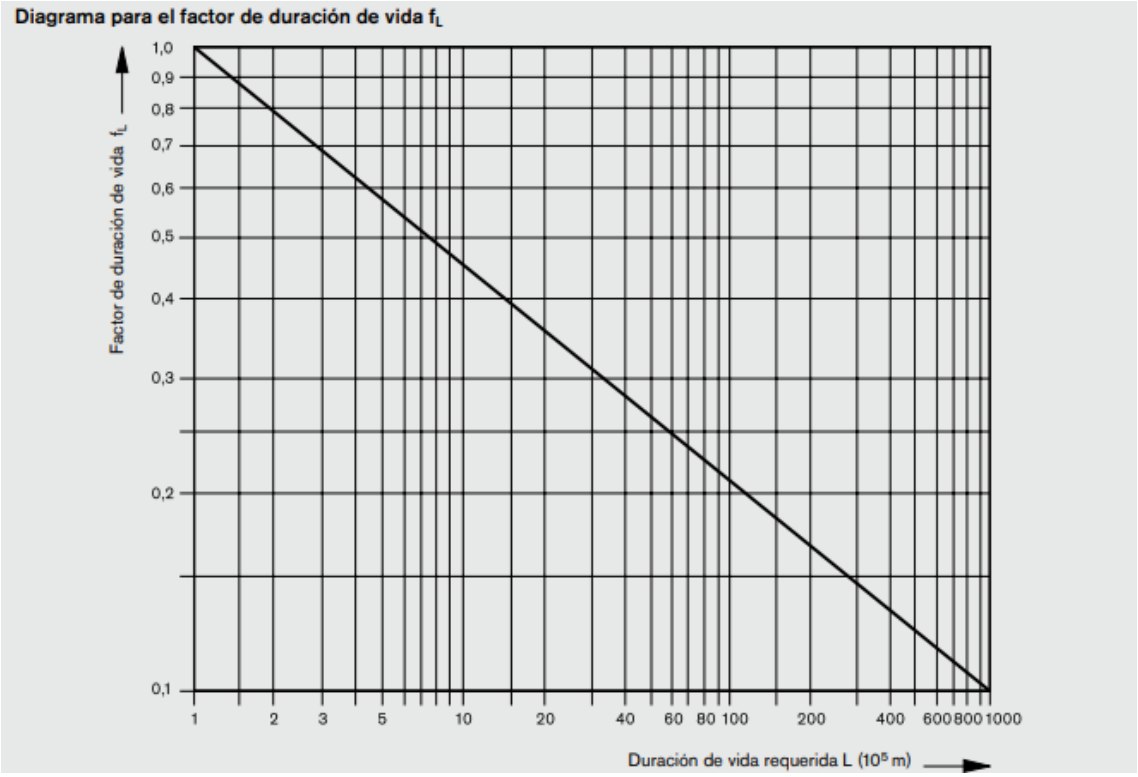


Figura 22: Diagrama para el factor de duración de vida f_L.

En la Figura 22 se muestra la influencia de la duración de vida, y con base en el cálculo realizado, se obtiene el factor $f_L = 0.35$.

La capacidad de carga requerida C_{req} se calcula a partir de

$$C_{req} = \frac{F_m}{f_H \cdot f_t \cdot f_s \cdot f_L} \tag{17}$$

Donde f_H es el factor de dureza, el cual se considera de 1 para el cálculo. El factor de temperatura f_t se obtiene de la Tabla 5. Finalmente, el factr de carrera corta f_s se considera con valor de 1.

Tabla 5: *Facto de temperatura de los rodamientos.*

Temperatura del rodamiento (°C)	100	125	150	175	200
Factor de temperatura f_t	1	0,92	0,85	0,77	0,70

La capacidad de carga requerida C_{req} resulta entonces:

$$C_{req} = \frac{65.33}{1 \cdot 1 \cdot 1 \cdot 0.35} = 186.65N$$

Por lo que el rodamiento 113KB1232 cumple perfectamente con la capacidad de carga requerida para la aplicación. Se utiliza este rodamiento en específico por las dimensiones de los carros y los ejes diseñados para el sistema Cartesiano. Así, la capacidad de carga del rodamiento seleccionado es de $C = 520N$. Para calcular la vida, el fabricante provee la siguiente fórmula:

$$L = \left(\frac{C}{F_m} \cdot f_H \cdot f_t \cdot f_s \right)^3 \cdot 10^5 \tag{18}$$

Por lo tanto, la vida nominal resulta en

$$L = \left(\frac{520N}{65.33} \cdot 1 \cdot 1 \cdot 1 \right)^3 \cdot 10^5 = 504 \cdot 10^5 minutos$$

La duración de vida convertida en horas de servicio se puede calcular según la fórmula:

$$L_h = \frac{L}{2 \cdot s \cdot n_s \cdot 60} \tag{19}$$

Las horas de servicio que tendrán de vida los rodamientos es de

$$L_h = \frac{504 \cdot 10^5}{2 \cdot 0.8 \cdot 3 \cdot 60} = 175 \cdot 10^3 horas$$

Lo cual sobrepasa el mínimo de 8000 horas establecidas al principio del diseño.

6.3.5. Selección de motores

Para la selección de los motores a utilizar, se siguió el criterio de selección mostrado en el Anexo 2 para el caso en que la carga está directamente acoplada al eje del motor [17].

Con base en ese método de selección de motores y los cálculos realizados a partir del script de MATLAB, el par pico obtenido para el eje y de movimiento tiene un valor de $T_{pico} = 0.17Nm$ que sin problemas puede redondearse a $T_{pico} = 0.2Nm$, ya que la metodología de diseño utilizada contempla el peor caso posible. En este tipo de diseño, es mejor redondear los valores hacia arriba. Para el par continuo, se tiene que $T_{RMS} = 0.07Nm$. En cuanto al eje x, los valores son $T_{pico} = 0.062Nm$ y $T_{RMS} = 0.03Nm$.

Para verificar, también se hizo el cálculo con base en las ecuaciones de par mostradas en el libro de Shigley [16], las cuales resultan en el par necesario para elevar la carga con tornillos ACME, dado por

$$T_R = \frac{Fd_m}{2}\left(\frac{l + \pi f d_m \sec\alpha}{\pi d_m - f l \sec\alpha}\right) \tag{20}$$

Donde α es el ángulo de avance y f es el coeficiente de fricción entre el tornillo y la tuerca. El par necesario para vencer la fricción del collarín se obtiene de

$$T_C = \frac{F f_c d_c}{2} \tag{21}$$

Donde f_c es el coeficiente de fricción entre el collarín y el tornillo y d_c es el diámetro del collarín. Los resultados de aplicar estas ecuaciones para cada eje del sistema da como resultado $T_{pico} = 0.21N \cdot m$ para el eje y, y $T_{pico} = 0.075N \cdot m$, para el eje x.

Se realizan ambos cálculos para diferenciar y comparar los criterios de diseño que se toman en cuenta. En este caso, tomar los resultados de los cálculos con el método expuesto por Shigley [16] se traduciría como una solución más conservadora, ya que contempla valores más altos del par. Dicha consideración permite tomar en cuenta el peor de los casos posibles durante el funcionamiento de la máquina.

Por otro lado, el cálculo con el método expuesto en el libro de Gürocack [17] ofrece una solución más flexible y considera una mayor cantidad de parámetros, como son la inercia del motor, el par generado por la carga al modelarse como una masa inercial equivalente, etc.

Por lo tanto, para este diseño se toman los valores calculados más altos para el valor de par T_{pico} y la selección de los motores se basa en esa información.

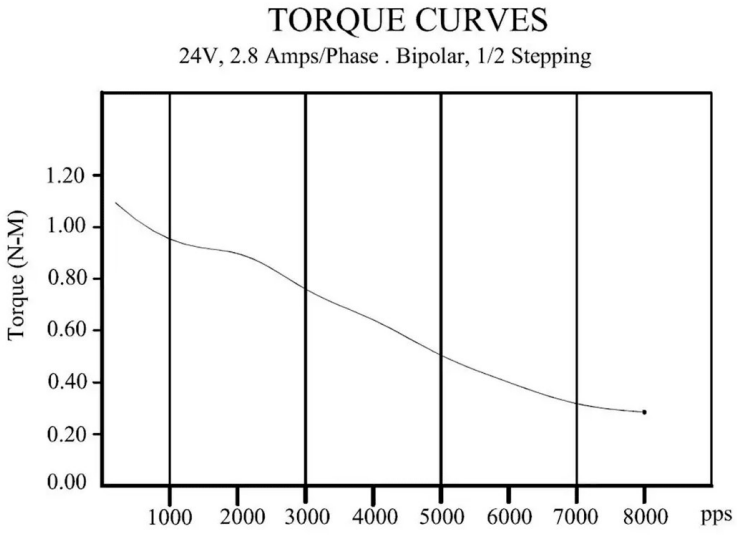


Figura 23: Curva par-velocidad del motor NEMA 23 fabricado por OpenBuilds®.

En la figura 23 se puede obsevar la curva par-velocidad que caracteriza al motor NEMA 23 que se propone utilizar para desplazar la carga a lo largo del eje y de movimiento. Los valores que se encuentran en el eje de las abscisas están representados en pulsos por segundo (pps) asumiendo que el motor está configurado en el modo de operación de medios pasos.

En este caso, el motor propuesto es un motor bipolar híbrido con 4 hilos. Se llama híbrido porque combina las características del motor a pasos de imán permanente y la del motor a pasos con reluctancia variable. Estos motores también son conocidos como “Motores de pasos con rotor de disco“ [18]. Por lo regular, estos motores tienen 100 segmentos segmentos de rotor, por lo que la distancia angular entre los polos es 3.6°. Por consiguiente, el ángulo de paso es la mitad de 3.6°, o sea 1.8°, ya que el rotor se mueve entre un polo y una zona neutral adyacente.

Tomando como ejemplo al motor NEMA 23 fabricado por OpenBuilds®, el motor requiere dar 200 pasos para girar el eje 360° (una vuelta completa) y, por lo tanto, el giro del eje por paso del motor es de 1.8° de movimiento angular. Por

otro lado, si se utiliza la configuración de $\frac{1}{2}$ paso, el motor requiere dar 400 pasos para girar 360°, por lo que el giro por paso es de $\frac{360}{400pasos} = 0.9$. Comprobando así lo establecido por Maloney [18] en su libro para motores de pasos con rotor de disco.

En primer lugar, es necesario describir la utilidad del uso del modo de operación por micropasos. Los motores a pasos pueden configurarse para que el movimiento giratorio tenga mayor resolución. Este aumento se refleja en la suavidad del movimiento de la carga que desplaza el eje, ya que cuando se utilizan más pasos por revolución, la conmutación entre los distintos estados del rotor son menos bruscos y existen menos vibraciones indeseadas en el eje de transmisión.

Sin embargo, conforme se aumenta la cantidad de pasos por revolución, también se aumenta la corriente que requiere el motor por devanado. Este aumento de corriente sucede porque el motor tiende a mantener el mismo par aunque la frecuencia sea mayor. Es decir, conforme se aumenta el número de pasos por revolución, la velocidad de giro tiene que ser mucho mayor para obtener el mismo avance, como se aprecia en la figura 24.

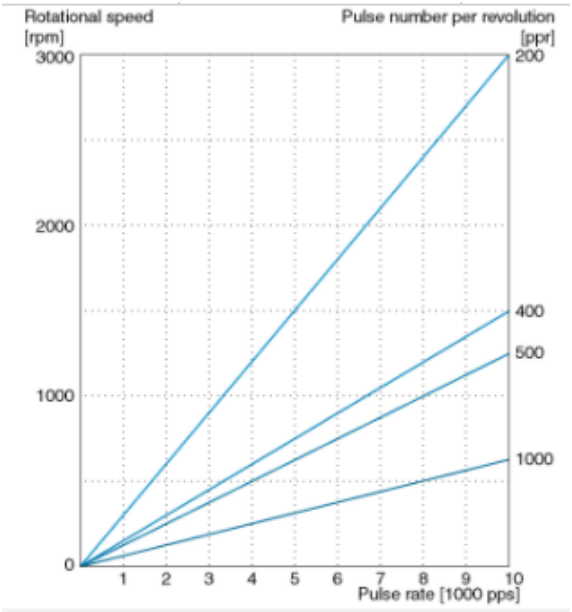


Figura 24: Gráfica de la frecuencia del pulso contra la velocidad de rotación.

Por consiguiente, utilizando los datos de la figura 23 se obtiene el valor del par con la velocidad deseada de movimiento según el perfil de movimiento obtenido en el diseño mecánico. El valor de la velocidad deseada es de 500rpm, por lo que haciendo la conversión de unidades

$$500\frac{rev}{min} \times \frac{1min}{60s} \times \frac{400pasos}{1rev} = 3333.33\frac{pasos}{s} = 3333.33pps$$

Buscando el valor obtenido en la curva de par-velocidad para el motor NEMA 23 (figura 23) , el valor aproximado del par entregado a dicha frecuencia es de $0.7N \cdot m$. Hay que recordar que el valor de par pico obtenido para el eje y de movimiento es $T_{pico} = 0.2N \cdot m$, y siguiendo el criterio de selección

$$T_{deseado} = 1.5 \quad T_{pico} = 1.5(0.2N \cdot m) = 0.3N \cdot m$$

Así pues, se tiene par de sobra que puede ser útil por los cambios que pueda experimentar la carga durante la operación del mecanismo. Además, se adquiere un margen más amplio para evitar fallas.

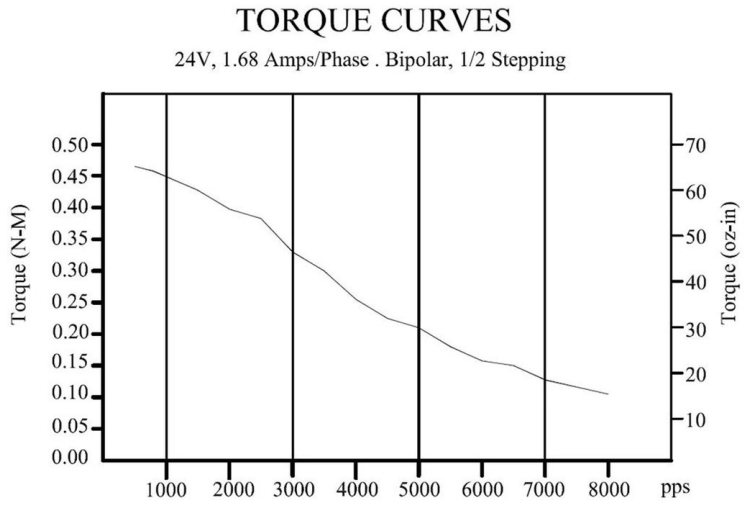


Figura 25: Curva par-velocidad del motor NEMA 17 fabricado por OpenBuilds®.

Ahora bien, para el motor del eje x, se propone utilizar el motor NEMA 17 fabricado por OpenBuilds®. La curva par-velocidad correspondiente a este motor se muestra en la figura 25. La velocidad deseada de operación es la misma que

para el motor pasado, por lo que se busca directamente el valor del par correspondiente a la frecuencia en el eje de las abscisas. El par otorgado por el motor a 500rpm es de aproximadamente $T = 0.30N \cdot m$. Según los resultados obtenidos del diseño mecánico, para el eje y se requiere un par pico $T_{pico} = 0.062N \cdot m$. Según el criterio de selección seguido

$$T_{deseado} = 1.5T_{pico} = 1.5(0.062N \cdot m) = 0.093N \cdot m \approx 0.1N \cdot m$$

Finalmente, se realiza el paso 5 del criterio de selección de motores, en donde se tiene que calcular la razón de inercia a partir de la siguiente ecuación:

$$J_R = \frac{J_{ejedelmotor} + J_{ref}}{J_motor} \tag{22}$$

Es decir, en el cociente se tiene la suma de todas las inercias presentes que son externas al motor. Por tanto, utilizando ecuación y los datos obtenido en el script de MATLAB, la razón de inercia tiene un valor de $J_R = 1.89$ para el motor en el eje x, y un valor de $J_R = 2.88$ para el motor en el eje y. Así, se cumple la condición $J_R \leq 5$ para los dos casos.

Por lo tanto, ambos motores son adecuados para utiizarse en el prototipo final para mover la carga en el efector final.

6.3.6. Selección de fuente de alimentación

Como fue planteado en los requerimientos funcionales, es deseable que el dispositivo esté disponible para que el usuario solamente conecte el dispositivo al tomacorriente y sea capaz de utilizarlo. Para ellos se tiene que seleccionar una fuente de poder adecuada para alimentar al sistema entero.

Sin embargo, analizando los componentes eléctricos y electrónicos que conforman al sistema, es mejor utilizar dos fuentes de alimentación independientes. Esto es porque se están utilizando motores, los cuales exigen niveles de corriente distintos y variables con respecto al tiempo según las exigencias que se le den para mover la carga. Por ello y por la característica inductiva de los motores es que utilizar la misma fuente para alimentar a la parte lógica resultaría en falsas

lecturas de valores lógicos, tanto ALTOS como BAJOS.

La solución propuesta parte de los requerimientos de los componentes seleccionados anteriormente en esta sección. El listado es el siguiente:

- Motor NEMA 17: 24V, 1.68A .
- Motor NEMA 23: 24V, 2.8A .
- Raspberry Pi: 5V, 3A .
- Parte lógica del sistema: 5V, al menos 1A.

Figura 26: Fuente de poder de 24V y 10A.

Para la alimentación de los motores, se propone utilizar una fuente de poder de 24V y 10A como la mostrada en la figura 26. Se selecciona de 10A porque requiere proveer la suma de las corrientes requeridas para operar los motores. Un criterio útil para seleccionar el valor de corriente deseado en la fuente es considerar el doble de la suma de las corrientes requeridas por los componentes a alimentar, para asegurar que los componentes eléctricos y electrónicos estén funcionando correctamente en todo momento.

Figura 27: Fuente de poder de 5V y 5A.

De manera similar, se selecciona una fuente de 5V que aporte la corriente suficiente para alimentar tanto a la Raspberry Pi como a la parte lógica de control en el circuito. Para ello, se selecciona un fuente de poder como la de la figura 27, la cual provee 5V y 5A de corriente.

6.4. Diseño eléctrico y electrónico

6.4.1. Diagrama de bloques

Para comprender bien el funcionamiento general del sistema y elaborar un diseño detallado sobre el mismo, siempre es de gran ayuda realizar un diagrama de bloques que ayude a establecer los componentes necesarios y las relaciones entre ellos para facilitar el proceso de diseño.

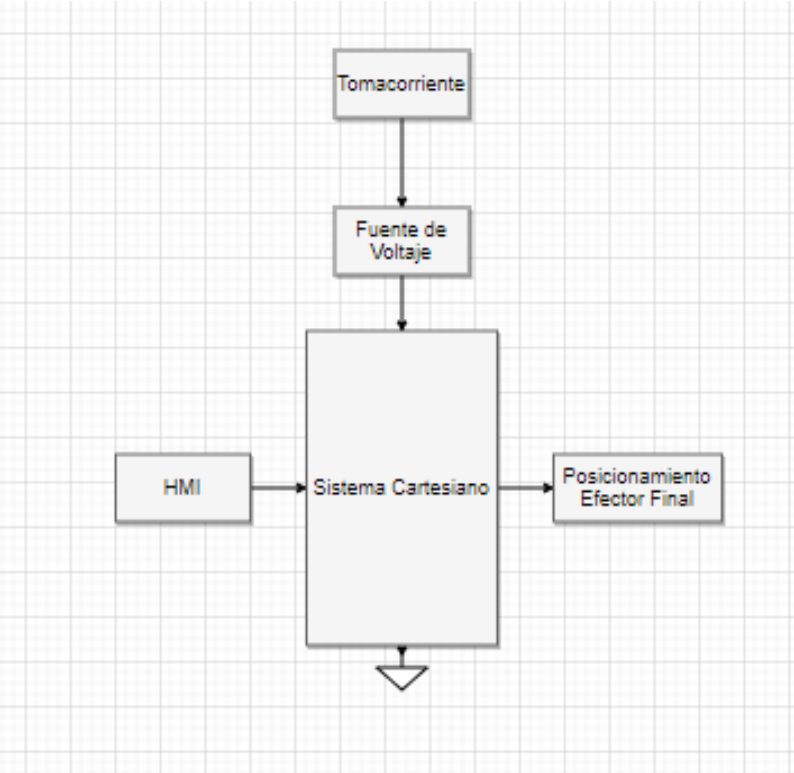


Figura 28: Diagrama de bloques general del sistema.

Como se puede apreciar en la Figura 28, la entrada al sistema es una posición introducida por el usuario a través de la HMI y la salida sería el posicionamiento del efector final en el plano $x - y$. El sistema en general está alimentado a partir de una fuente de poder conectada directamente a algún tomacorriente del laboratorio en el que se instale el sistema.

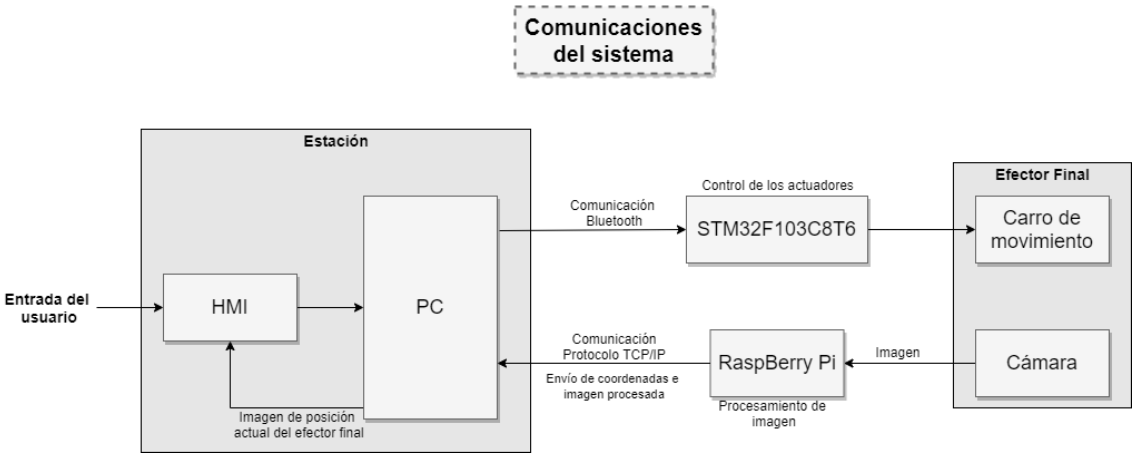


Figura 29: Diagrama de bloques de las comunicaciones y el envío y recepción de datos del sistema completo.

En la figura 29, puede observarse la configuración utilizada para las comunicaciones entre las distintas partes del sistema. En la parte más a la izquierda, se muestra la estación, que está conformada por la PC que despliega en el monitor la HMI, con la cual el usuario interactúa con el sistema Cartesiano.

Cuando el sistema se encuentra en el modo manual, el usuario ingresa el valor de posición deseado para el efector final a través de la HMI. Dicha coordenada se envía por medio de Bluetooth al microcontrolador que se encuentra en el robot Cartesiano, el cual se encarga de posicionar a la cámara RGB-D en la posición ingresada por el usuario.

En el efector final se tiene la cámara, la cual transmite la imagen que captura a la Raspberry Pi. La tarjeta Raspberry se encarga de hacer el procesamiento de imagen, y envía a la estación como resultado la imagen procesada que es la que el usuario puede observar en la HMI. Cuando el sistema está operando en el modo automático, la estación recibe de la Raspberry Pi la coordenada actual del vehículo analizado, habilitando así la realimentación servo-visual al sistema.

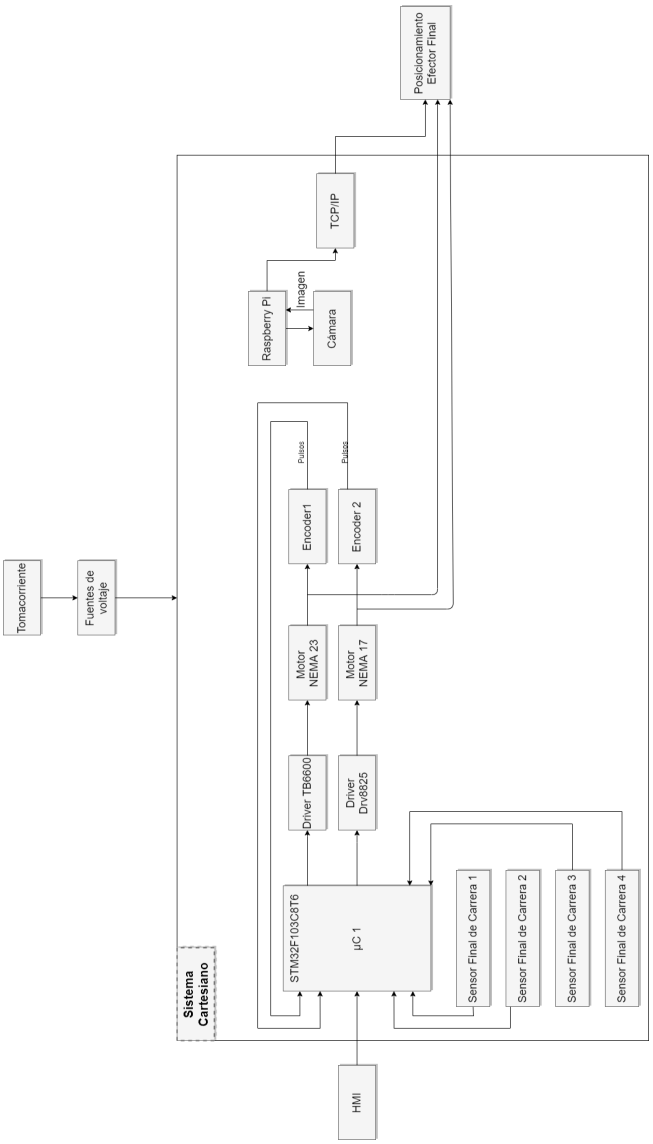


Figura 30: Diagrama de bloques desarrollado para el sistema Cartesiano.

En el diagrama de bloques mostrado en la figura 30, se puede observar un desarrollo más a detalle del funcionamiento del sistema Cartesiano. Por un lado se muestra la entrada del sistema, la cual es un valor que llega al μC por medio de Bluetooth. Se muestran las conexiones que tiene el microcontrolador con los demás elementos del sistema, que son los sensores finales de carrera, los motores con sus respectivos driver electrónicos y los encoders. En el otro extremo, se muestra la interacción entre la cámara, la Raspberry Pi y el resultado final del sistema.

6.4.2. Diagramas electrónicos

El diagrama que contiene el circuito eléctrico completo se encuentra en el Anexo 2, el cual incluye el μC, los motores, el módulo Bluetooth y los sensores.

En esta parte se analizan los diagramas por separado de los componentes utilizados, comenzando por los motores.

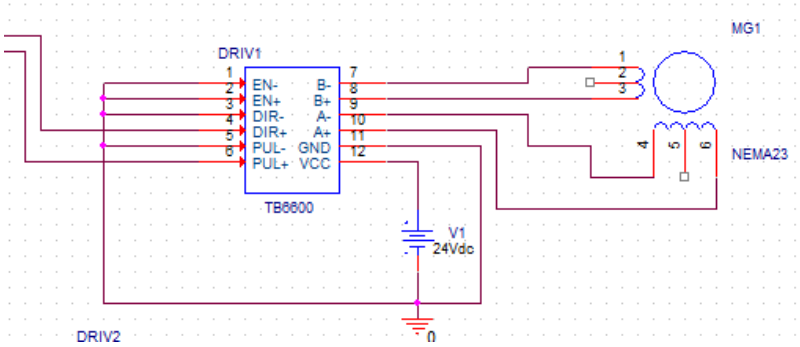


Figura 31: Conexiones del motor NEMA 23 sin mostrar el μC.

En la figura 31 se muestran las conexiones realizadas para operar el motor NEMA 23. Se conecta el driver TB6600 en configuración de cátodo común, es decir, haciendo comunes todas las entradas con signo negativo en la carcasa del driver. Se alimenta al driver con 24V, y consume hasta 2.9A pico según el grabado en la parte delantera del driver. La configuración de micropasos utilizada es el modo de operación de medios pasos. La dirección y la velocidad se controlan desde el μC por medio de pulsos desde los puertos GPIO. Posteriormente, se conectan las terminales del driver a los extremos de las bobinas y finalmente se puede utilizar el motor.

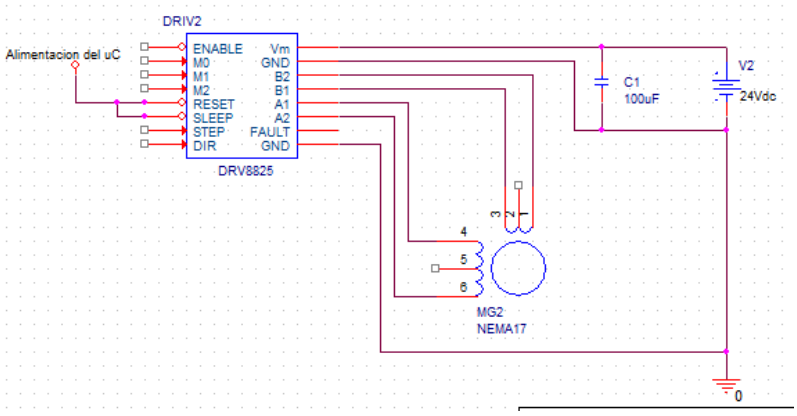


Figura 32: Conexiones del motor NEMA 17 sin mostrar el μC.

Las conexiones son similares para el caso del motor NEMA 17, sin embargo, el driver a utilizar es el DRV8825, el cual requiere que se conecte un capacitor de $100\mu F$ para filtrar el ruido eléctrico proveniente del motor que pueda afectar al IC.

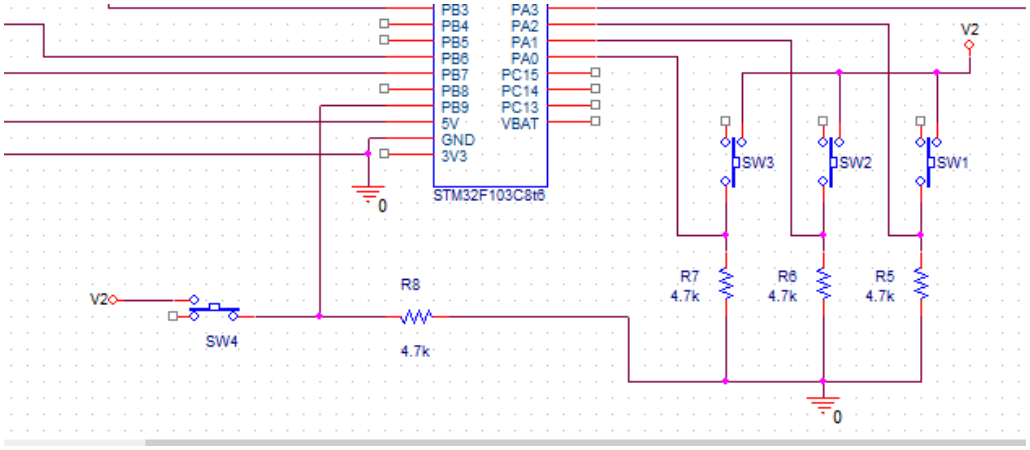


Figura 33: Conexiones de los interruptores de final de carrera al μC.

Los interruptores de final de carrera se conectan de la forma mostrada en la Figura 33. Se coloca un resistencia de pull-down para evitar falsos valores lógicos en la entrada del pin del microcontrolador, ya que los pines GPIO tienen un valor flotante en la entrada por defecto. Además, el interruptor de final de carrera es un interruptor SPDT, por lo que se conecta la terminal normalmente abierta para que el circuito se cierre cuando se haga contacto mecánicamente entre el carro y alguna parte del sistema Cartesiano.

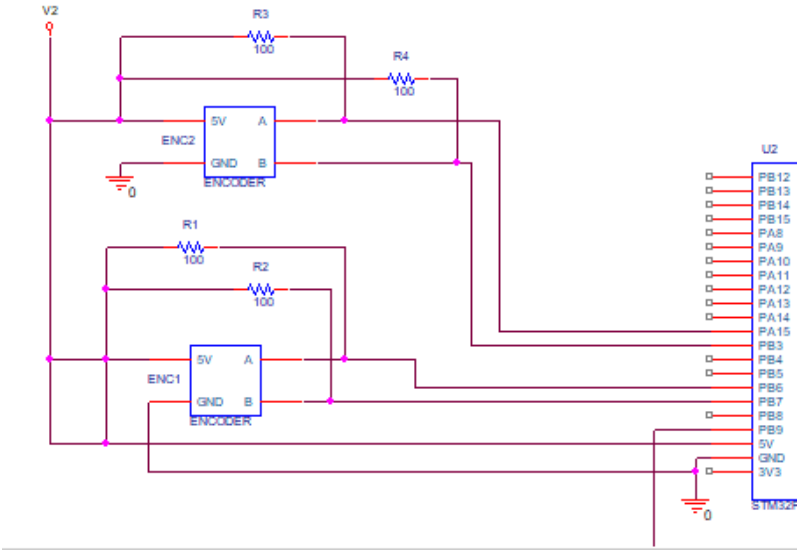


Figura 34: Conexiones de los encoders al μC.

En la Figura 35 se enfoca la conexión de los encoders al microcontrolador. Se conectan las fases del sensor a los pines PB6 y PB7 para el encoder 1, y a los pines PA15 y PB3 para el encoder 2. Estos pines corresponden a los canales de los timers de propósito general con los que cuenta el STM32. Este μC cuenta con una función adicional en los timers con la que se puede hacer el conteo de pulsos del encoder directamente desde el hardware al configurar el "Modo Encoder" en el registro necesario. El timer es de 16 bits por lo que la resolución de la lectura por vuelta es de $2^{16} - 1 = 65,535$ con desbordamiento automático del contador.

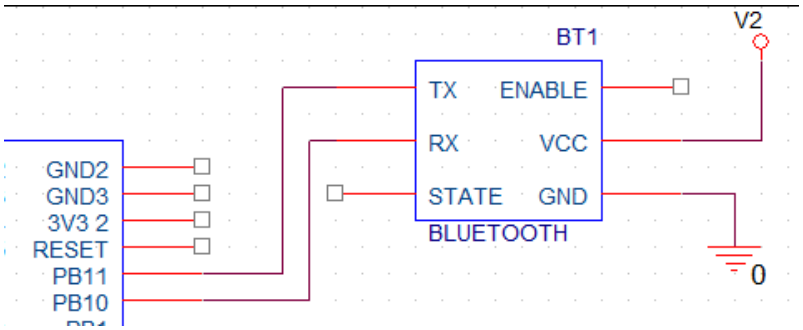


Figura 35: Conexión de dispositivo de comunicación Bluetooth, HC-05, al μC.

Por último, se muestra la conexión que se realiza del módulo Bluetooth HC-05, el cual se utiliza para recibir los valores deseados de posición del efector final provenientes de la PC en la estación. Los pines RX y TX del HC-05 se conectan a los pines PB10 y PB11, respectivamente. Dichos pines corresponden a la

conexión al USART3 del microcontrolador, por lo que es necesario establecer comunicación serial entre el módulo y el μC para obtener los datos requeridos para el funcionamiento del sistema.

6.5. Cinemática directa

El prototipo que se diseña en este trabajo es un robot Cartesiano, por lo que la categoría de sus eslabones es del tipo prismático (P). Este robot es de 2 grados de libertad (a partir de ahora referidos como gdl) ya que sólo se desplaza en el plano $x - y$. Por lo tanto, se puede definir al sistema como un robot Cartesiano de 2 gdl tipo PP.

La forma de modelar la cinemática del robot parte de poder obtener la posción de sus eslabones en el espacio de trabajo del dispositivo. De esa manera, se puede conocer la posición del efector final cuando los eslabones se posicionan según cierta configuración indicada.

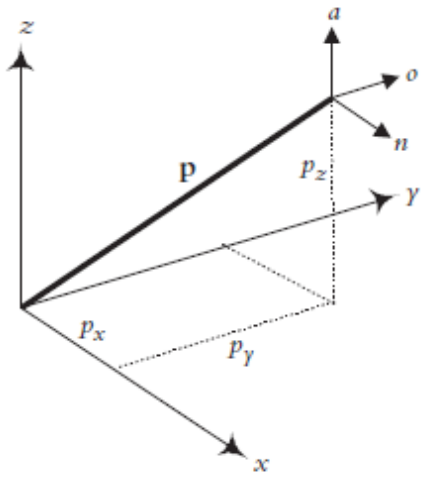


Figura 36: Coordenadas Cartesianas.

En el caso de este trabajo, como se trata de un robot Cartesiano y al no existir rotaciones en el sistema, la matriz de transformación que representa este movimiento al punto p es una matriz de traslación simple [19] que se muestra en la figura 36. El único parámetro de interés es la posición del origen del marco de

referencia, no su orientación.

La matriz de transformación que representa la ecuación de cinemática directa de la posición de la mano del robot en un sistema de coordenadas Cartesianas es:

$$T_p^R = T_{cart}(p_x, p_y, p_z) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

Sin embargo, se utiliza la convención de Denavit-Hartenberg (o sólo DH) para relacionar los marcos de referencia de los eslabones del mecanismo del robot de manera sistemática. Para mayor información, referirse al Anexo 3.

Con base en las definiciones referidas en el párrafo anterior, se procede a obtener la cinemática directa del robot.

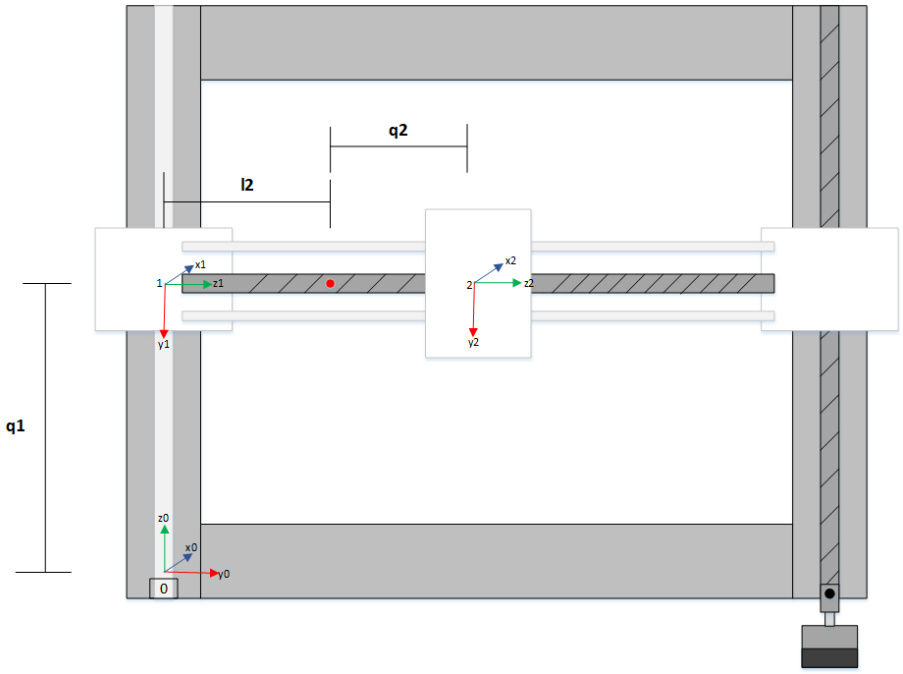


Figura 37: Determinación de los sistemas de referencia de los eslabones según la convención DH.

A partir de la definición de los sistemas de referencia para cada eslabón mostrados en la figura 37, se obtiene la siguiente tabla de parámetros DH:

Tabla 6: *Tabla de parámetros DH para el robot Cartesiano.*

	θ_i	d_i	a_i	α_i
1	0	q_1	0	$-\frac{\pi}{2}$
2	0	l_2+q_2	0	0

Se comienza por la esquina inferior izquierda, la cual ha sido definida como el 0 del sistema. Es en ese lugar en el que se colocan dos sensores de final de carrera para marcar la referencia o “home“ y ahí mismo es donde se coloca el primer marco de referencia con el eje z apuntando hacia arriba. Esta posición es ajustable para cuando el sistema se monte físicamente, ya que el sistema físico debe de ajustarse al diseño de cinemática directa, no al revés.

Por otro lado, al ser una junta prismática, las rotaciones no se consideran, sólo las traslaciones en el plano. En el marco de referencia 1 se coloca el eje z hacia la derecha, siguiendo la dirección del movimiento del segundo eslabón. Es de interés remarcar que se dejó un espacio entre el marco de referencia 2 y el 1, el cual está marcado en el diagrama como $l2$, el cual se deja con el propósito de tomar en cuenta la parte central del carro del efector final como origen de dicho marco de referencia. Es por eso que se compensa esa distancia en el parámetro d correspondiente al segundo eslabón.

Por tanto, la matriz de transformación homogénea para el primer par de marcos de referencia según las ecuaciones del Anexo 3 es

$$A_1 = T_0^1 = \begin{bmatrix} c_0 & -s_0 c_{\frac{\pi}{2}} & s_0 s_{\frac{\pi}{2}} & a_i c_0 \\ s_0 & c_0 c_{\frac{\pi}{2}} & -c_0 s_{\frac{-\pi}{2}} & a_i s_0 \\ 0 & s_{\frac{-\pi}{2}} & c_{\frac{\pi}{2}} & q_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & q_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Y la matriz de 1 a 2 está dada por:

$$A_2 = T_1^2 = \begin{bmatrix} c_0 & -s_0c_0 & s_0s_0 & a_ic_0 \\ s_0 & c_0c_0 & -c_0s_0 & a_is_0 \\ 0 & s_0 & c_0 & l_2 + q_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_2 + q_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Por ende, la matriz de transformación homogénea total se encuentra a partir de

$$T = A_1A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & q_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_2 + q_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & l_2 + q_2 \\ 0 & -1 & 0 & q_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6.6. Cinemática inversa

El cálculo de la cinemática inversa permite obtener la configuración de los eslabones del robot a partir de la posición deseada en el efector final, por lo que resulta útil para hacer un control cinemático.

La cinemática inversa se puede obtener a partir del cálculo previo de la cinemática directa a partir de la siguiente relación:

$$A_1A_2A_2^{-1} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times A_2^{-1} \tag{24}$$

Por lo que, de esta manera, es posible determinar el valor necesario de desplazamiento en los eslabones (q_1 y q_2) para llegar a las coordenadas deseadas. La matriz inversa de la matriz A_2 es:

$$A_2^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l_2 - q_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

El resultado se comprueba multiplicando A_2 por A_2^{-1}, resultando en la matriz identidad I.

De tal manera que al multiplicar A_2^{-1} por la matriz de parámetros **n**, **o**, **a** y **p**, e igualando con la matriz A_1 se tiene

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & q_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & -a_x(l_2 + q_2) + p_x \\ n_y & o_y & a_y & -a_y(l_2 + q_2) + p_y \\ n_z & o_z & a_z & -a_z(l_2 + q_2) + p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De tal forma que se obtienen las relaciones siguientes para la matriz **n**, **o**, **a**:

$$n_x = 1$$

$$a_x = 0$$

$$a_y = 1$$

$$a_z = 0$$

$$oz = -1$$

Y el vector P se iguala con la útlima columna de la matriz A_1, dando las ecuaciones siguientes:

$$q_1 = -a_z(l_2 + q_2) + p_z$$

$$p_x - a_x(l_2 + q_2) = 0$$

$$p_y - a_y(l_2 + 1_2) = 0$$

De tales resultados, se resuelve para las variables q_1 y q_2

$$q_2 = \frac{p_y}{a_y} - l_2 = p_y - l_2$$

$$q_1 = -a_z(l_2 + q_2) + p_z$$

Sustituyendo los valores de a_y y a_z, y derivando con respecto al tiempo, se tiene que:

$$\dot{q_1} = dp_z$$

$$\dot{q_2} = dp_y$$

Lo cual indica que la velocidad del eslabón 1 y 2 corresponden al cambio de posición con respecto al tiempo en las direcciones z e y, respectivamente. Esto es lógico ya que se está trabajando con uniones prismáticas y movimientos lineales. Lo relevante de este resultado es que indica en qué dirección se mueven los eslabones con respecto al marco de referencia establecido.

6.7. Jacobiano geométrico

El Jacobiano geométrico se obtiene a partir de la cinemática directa calculada anteriormente, el cual es utilizado para representar geométricamente a un mecanismo en el tiempo. Permite la conversión de movimientos diferenciales o velocidades de articulaciones individuales a movimientos diferenciales o velocidades de puntos de interés (por ejemplo, el efector final).

Tabla 7: *Determinación de los elementos del Jacobiano geométrico.*

	Prismatic	Revolute
Linear	$R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (d_n^0 - d_{i-1}^0)$
Rotational	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Para determinar los elementos dentro de la matriz que describe al Jacobiano J, es necesario utilizar la tabla 7, la cual indica cómo completar la matriz del Jacobiano dependiendo del tipo de unión entre los eslabones y el tipo de movimiento. Para el caso de uniones prismáticas y movimiento lineal en tres dimensiones, el Jacobiano se completa a partir de

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} R_0^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & R_1^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & R_2^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{d}_1 \\ \dot{d}_2 \\ \dot{d}_3 \end{bmatrix} \tag{25}$$

Las matrices R_{i-1}^{0} son las matrices de rotación desde el marco de referencia 0 hasta el marco de referencia $i-1$, donde $i = 1, 2, 3$. Para obtener esas matrices, basta con referirse al cálculo de la cinemática directa en la sección 6.5. Luego, se multiplica la matriz de rotación correspondiente por $[0\ 0\ 1]^T$ y el resultado se coloca en el índice correspondiente de la primer fila del Jacobiano. Posteriormente, se igualan los miembros de la ecuación para determinar las velocidades del sistema.

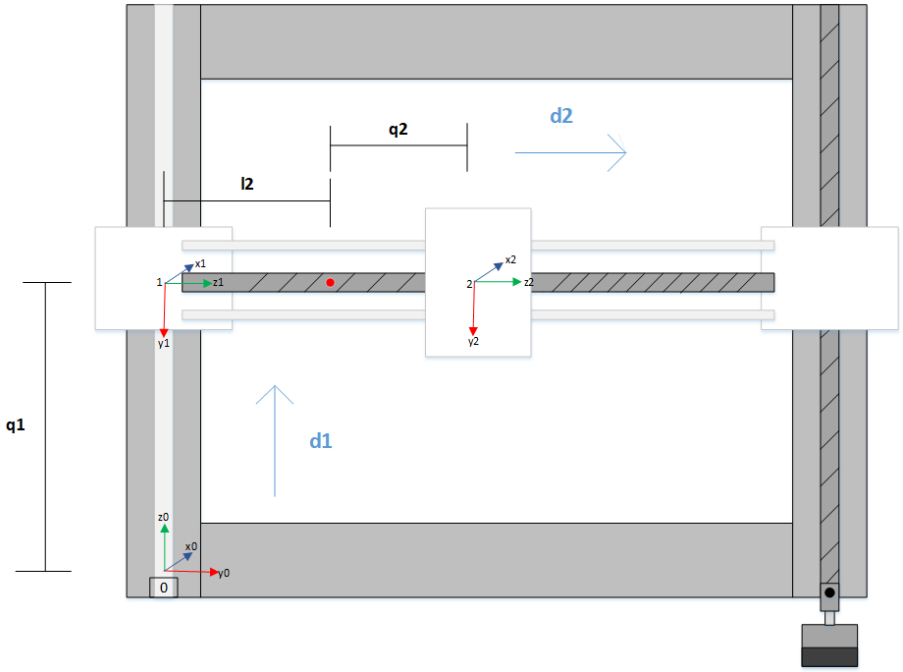


Figura 38: Determinación de las direcciones de movimiento para obtener las velocidades a partir del Jacobiano.

Con base en la figura 38, el Jacobiano geométrico para el robot Cartesiano resulta en

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ 0 \end{bmatrix}$$

Al hacer igualaciones, se obtiene:

$$\dot{x} = 0$$

$$\dot{y} = \dot{q}_2$$

$$\dot{z} = \dot{q}_1$$

Para verificar que el resultado tiene sentido, se tiene que obervar el marco de referencia 0 en la figura 38. En ese sistema de coordenadas, la componente z apunta hacia la dirección del movimiento d_1, por lo que la ecuación obtenida $\dot{z} = \dot{d}_1$ es correcta. Asimismo, en ese mismo eje de coordenadas,la componente y apunta hacia la dirección d_2, por lo que la ecuación $\dot{y} = \dot{d}_2$ también guarda sentido. En el caso de $\dot{x} = 0$ también tiene sentido porque el eje x apunta hacia la dirección z de un sistema tridimensional Cartesiano, y como el robot no tiene actuador en esa dirección, el desplazamiento y la velocidad son iguales a cero.

6.8. Control cinemático

El diagrama de bloques en el que se basa el diseño del control es el que se muestra en la Figura 39:

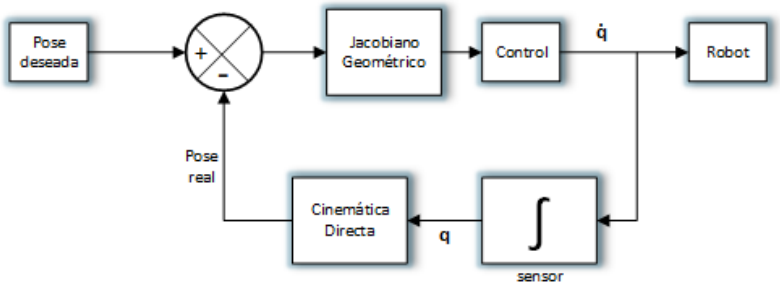


Figura 39: Diagrama de control cinemático.

La pose deseada contiene dos coordenadas, x y y. Éstas pueden ser introducidas por el usuario en el modo manual, o bien, ser enviadas a la estación a través del procesamiento de imagen de la Raspberry Pi en el modo automático de operación.

Como se muestra en la figura 29, la Raspberry Pi envía la imagen obtenida de la cámara, ya procesada, a la PC en la estación. Además de la imagen, también se reciben las coordenadas en donde se encuentra el vehículo de interés, lo cual genera el error de posición de la cámara. El código del procesamiento de imagen es proveído por el cliente y será integrado para su utilización con el prototipo final.

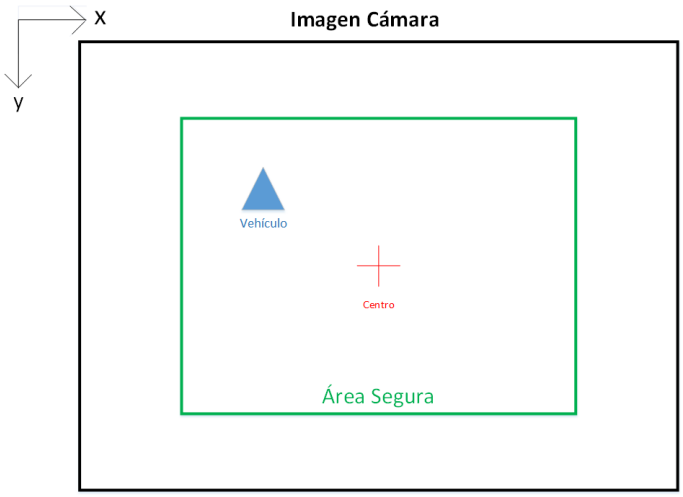


Figura 40: Ejemplo genérico de imagen procesada que recibe la PC.

La Figura 40 muestra una imagen ejemplo resultado del procesamiento de imagen del programa proveído por el cliente. Como se puede apreciar en la figura, la cámara contempla un área segura en la que puede operar sin problema el vehículo analizado. Sin embargo, cuando el vehículo sale de dicha área, la cámara necesita posicionar el centro de la imagen en el centro de la figura que identifica al vehículo autónomo. Por tanto, es necesario contemplar una forma de convertir ese error de distancia en píxeles a un error de distancia en metros (o centímetros).

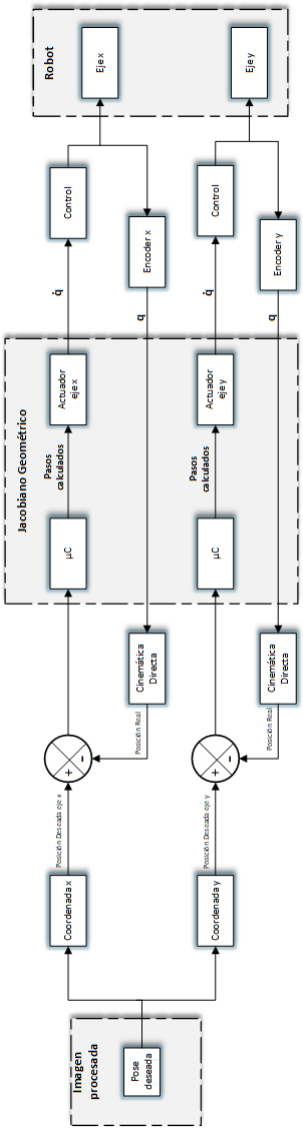


Figura 41: Control de lazo cerrado para el modo automático.

En la Figura 41 se encuentra el diagrama de bloques correspondiente al modelo de sistema de control de lazo cerrado para controlar la posición del efector final en el modo automático. Como se mostró anteriormente, el desarrollo del Jacobiano geométrico determina que la velocidad de los ejes en las direcciones de movimiento convenidas está dada por la velocidad de los eslabones del robot, por lo que esa velocidad proviene directamente del μC, el cual calcula el número de pasos necesarios a partir del error de posición.

Cabe destacar que hay un controlador por cada eje de movimiento. La velocidad del movimiento del eje en cuestión entra al controlador, cuya salida es la velocidad con la que se desplaza el eje y resulta en el posicionamiento del efector final en la posición deseada.

En cuanto al modo manual de operación del sistema, el control individual de los actuadores será de lazo abierto aprovechando la utilización de encoders en los extremos del tornillo de potencia. Se toma en cuenta que el motor a pasos no realiza el número de pasos exacto que le indica el controlador debido a factores como la fricción entre el tornillo y la tuerca, por lo que existe una pérdida de pasos durante la operación que puede generar un error de posición que se acumula y entre más tiemp pasa, se hace considerable. De tal forma que incorporar el uso de los encoders brinda la posibilidad de verificar que el número de pasos ejecutado corresponda al indicado por el controlador.

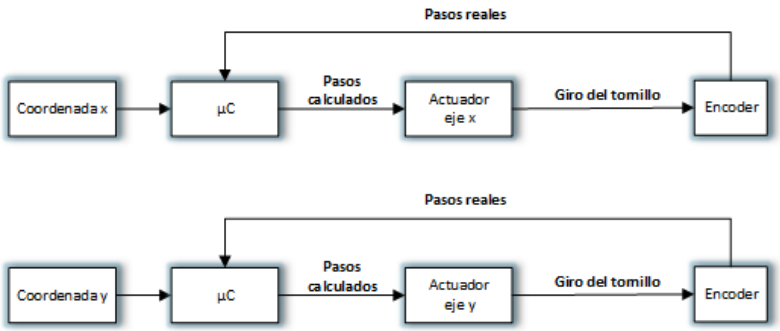


Figura 42: Control de lazo abierto para cada eje de movimiento.

Se muestra en la Figura 42 el diagrama de bloques del funcionamiento de este sistema de lazo abierto para el posicionamiento de la cámara en el modo manual de operación. Los diagramas de bloques para cada modo de operación se presentan a detalle en el Anexo 7.

El controlador recibe las coordenadas introducidas por el usuario a través de la HMI y utiliza estos datos para calcular el número de pasos necesario y la dirección de movimiento para posicionar al efector final en la posición deseada por el

usuario. Internamente en el programa se incluirá una condición en la que el movimiento del actuador no se detenga sino hasta que el número de pasos detectados por el encoder sea igual al número de pasos indicados por el controlador.

6.9. Diseño de PCB

Para el diseño de la PCB hay que tomar en cuenta dos aspectos importantes, que son el hecho de que se requieren altas corrientes para el funcionamiento de los motores, y el modo de obtención de las señales que entran al microcontrolador.

Tomando en cuenta primero al microcontrolador se proponen dos soluciones de implementación. La primera de ellas es utilizar dos tiras de pines hembra para conectar la tarjeta de desarrollo del STM32 directamente. Por otro lado, otra opción es utilizar un servicio de soldadura con tecnología de montaje superficial (SMT, por sus siglas en inglés) para utilizar el microcontrolador en la PCB con un sistema mínimo y entradas para el programador.

La cotización del servicio SMT se realiza con la empresa JLCPCB. Dicha compañía hace el ensamblaje de componentes SMD empezando desde $7 USD ($\approx$ $\$137.57$ MXN) y sumando una tarifa de montaje de $0.0015 USD ($\approx$ $\$0.029$ MXN) por unión. El precio resulta accesible y razonable por el acabado profesional que ofrece la empresa al utilizar su servicio.

El diseño de la PCB se realiza en la plataforma EasyEDA. La ventaja de utilizar esta plataforma es que está conectada directamente con la base de datos del *stock* de componentes de la empresa JLCPCB, por lo que se puede cotizar al momento de realizar el diseño y asegurar la disponibilidad de los componentes a ser utilizados.

Por tanto, el diseño que se realiza como prioritario es contemplando utilizar el servicio SMT de la empresa JLCPCB. Se deja como secundario el diseño en el

que se utiliza la tarjeta de desarrollo prefabricada, ya que es relativamente fácil por tener menor número de consideraciones necesarias.

Tabla 8: *Anchos de pista recomendados en una PCB según la corriente que requiera el driver de un motor.*

PCB TRACE WIDTHS				
Current (RMS or dc)	Trace width in 1 oz. copper		Trace width in 2 oz. copper	
	Outer layer (mm)	Inner layer (mm)	Outer layer (mm)	Inner layer (mm)
≤1 A	0.6	1.2	0.3	0.6
2.5 A	1	2	0.5	1
5 A	2.5	5	1.2	2.5
10 A	7	14	3.5	7

Ahora bien, se propone montar el driver para el motor NEMA 17 en la misma PCB que los componentes lógicos. Esto resulta en la necesidad de determinar anchos de pista distintos para los diferentes valores de corriente que se utilizan en el circuito. Como se puede observar en la Tabla 8, los anchos de pista recomendados varían en función del peso del cobre y de la corriente consumida por el circuito integrado utilizado como driver para el motor en cuestión [20].

La corriente especificada para el motor NEMA 17 según la hoja de datos del fabricante es de $1.68A$. Se debe considerar al menos el doble de corriente por las variaciones que puedan presentarse durante el uso del prototipo. Así que según las recomendaciones de la Tabla 8, como el doble de la corriente requerida es $1.68 \times 2 = 3.36$, un ancho de pista de 2.5mm en la capa exterior cumpliría con los requisitos térmicos de operación. Este ancho de pista se debe utilizar en todos los componentes a los que debe de llegar alrededor de 5A de corriente. Se puede utilizar un ancho de $0.8mm$ en todas las pistas conectadas a componentes lógicos.

El esquemático realizado para el diseño de la PCB se muestra en el Anexo 8, junto con el diseño del *layout*.

6.10. Simulaciones

Para simular la respuesta del motor ante la carga en el eje se utilizó un ejemplo del *toolbox* Simscape de MATLAB. Dicho modelo es un sistema de lazo abierto con un motor a pasos, el cual consta de un bloque que contiene todas las ecuaciones que modelan a un motor a pasos, simplemente se modifican las propiedades físicas del modelo por el valor real del motor a utilizar.

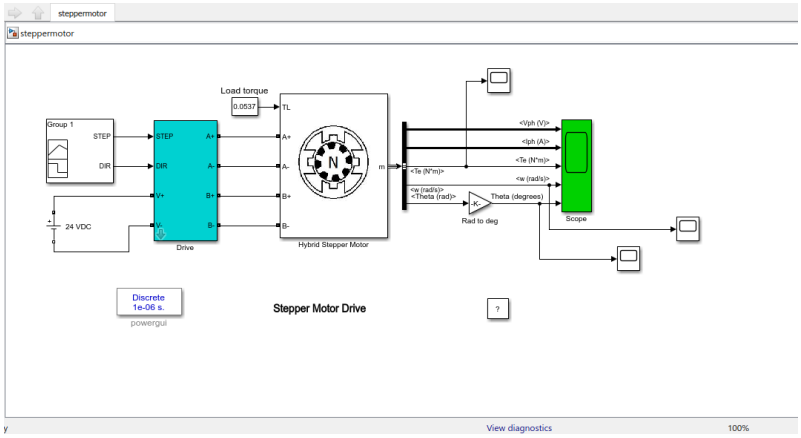


Figura 43: Diagrama de bloques en Simulink para la simulación del motor a pasos.

Se muestra en la Figura 43 el modelo en Simulink utilizado para la simulación del motor moviendo la carga. Por el lado izquierdo se tiene un constructor de señales, el cual sirve para generar las señales de dirección y de velocidad que entran al circuito integrado del driver del motor.

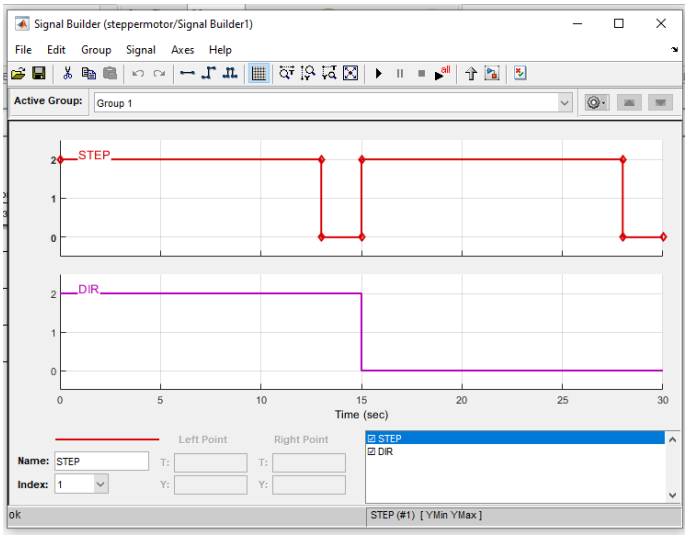


Figura 44: Configuración de simulación de las señales de entrada al driver.

La simulación se realiza tomando en cuenta que el tiempo del viaje completo de 80cm en la dirección de movimiento se diseña para 13 segundos. En este entendido, la señal que activa el movimiento a la velocidad especificada se enciende durante 13 segundos, se apaga durante 2 segundos más y finalmente, se enciende durante otros 13 segundos simulando el viaje de regreso. Como se puede ver en la Figura 44, la señal de dirección se invierte después de 13 segundos para indicar que el movimiento se realiza en la dirección contraria.

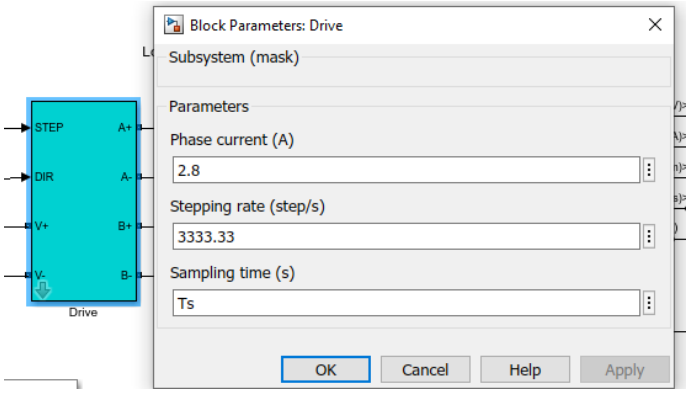


Figura 45: Configuración de los parámetros de simulación del driver.

Los parámetros para la configuración del driver se muestran en la figura 45. Se considera el valor de corriente que requiere el motor para operar y la velocidad en pasos por segundo establecida en el diseño.

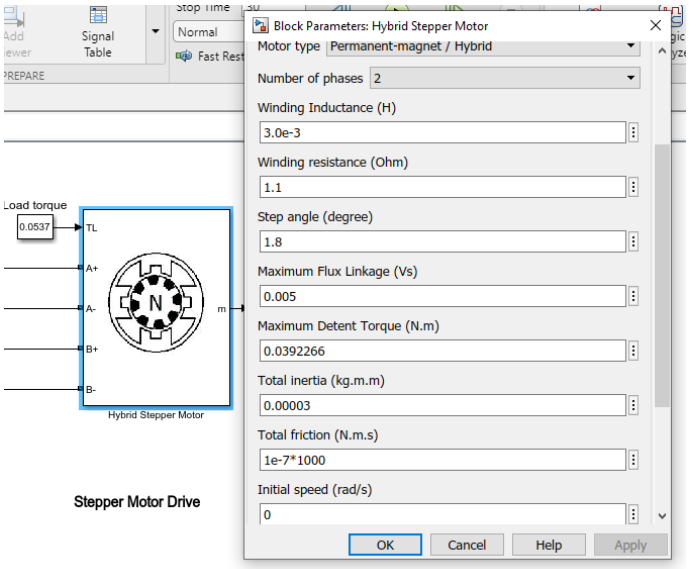


Figura 46: Configuración de los parámetros de simulación del motor.

Finalmente se introducen los datos proveídos por el fabricante en el modelo del motor, lo cual resuelve las ecuaciones internas del modelo para obtener en la salida el par eléctrico desarrollado, el voltaje y la corriente de fase, la velocidad angular del eje y la posición angular del mismo. Cabe destacar que este modelo permite incluir el efecto del par presente en la carga, cuyo valor es colocado en el modelo del motor paso a paso.

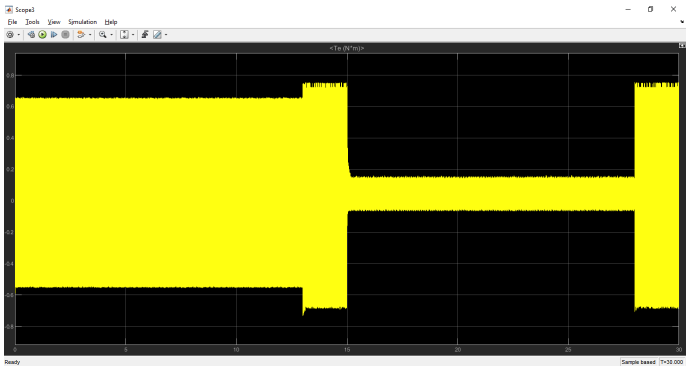


Figura 47: Gráfico del par desarrollado por el motor como resultado de la simulación.

En la Figura 47 puede observarse el cambio en el valor del par durante el movimiento hacia un lado y de regreso. La gráfica obtenida indica que el par pico desarrollado tiene un valor de aproximadamente $0.7N \cdot m$, lo cual cumple con los requisitos de par expuestos en la sección de selección de motores.

Para la simulación de los esfuerzos en los ejes que sirven como guía lineal, se cargó el modelo de la pieza en SolidWorks.

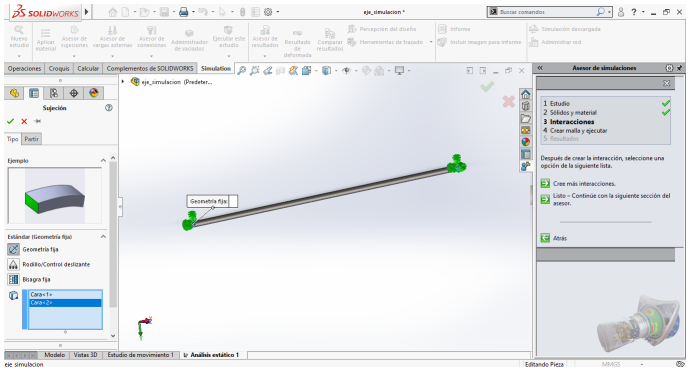


Figura 48: Carga del modelo e inicio del complemento Simulation.

La simulación se realiza con el complemento de SolidWorks: Simulation. El tipo de prueba que se hace es de tensión y con ella se obtienen tanto las deflexiones y la deformación, como el esfuerzo de Von Mises.

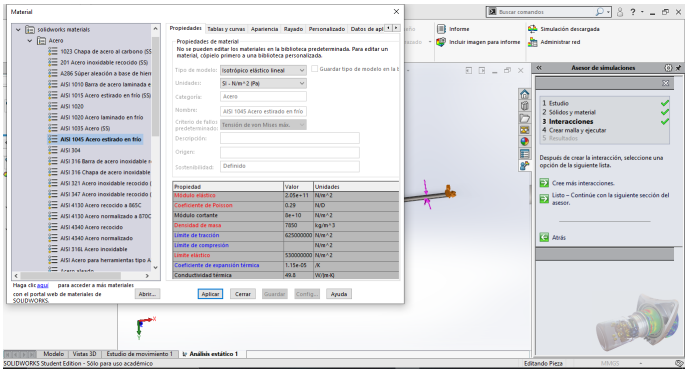


Figura 49: Configuración del material.

En la Figura 49 se observa la ventana en la que se selecciona el material de la pieza a analizar (Acero 1065) para de esta manera añadirle sus propiedades físicas.

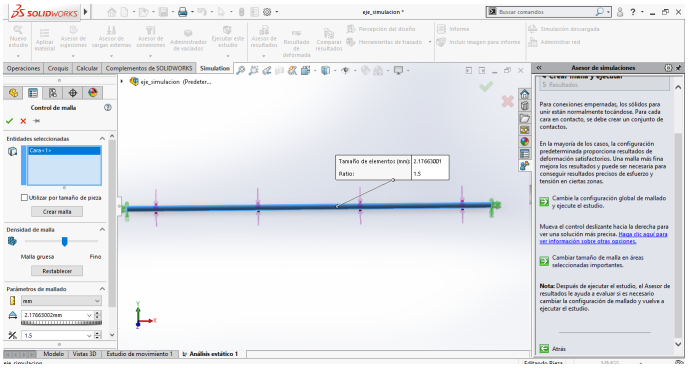


Figura 50: Determinación de la carga y soportes fijos.

Se aprecia en la Figura 50 la configuración de la carga que se aplica sobre el elemento para obtener como resultado el esfuerzo resultante. Además, se configuran como fijos los extremos debido a que la barra se encuentra sujeta de ese modo en el modelo CAD del prototipo.

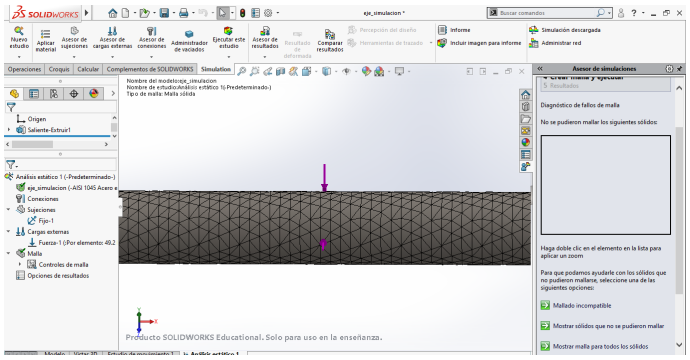


Figura 51: Configuración del mallado.

SolidWorks Simulation utiliza el método de análisis de elemento finito para obtener sus resultados. Por ello es necesario determinar un tamaño para la malla, el cual varía según la precisión que se busque en la simulación. En este caso, no es necesario utilizar polígonos de tamaño muy reducido, por lo que se deja el valor por defecto como se ve en la Figura 51.

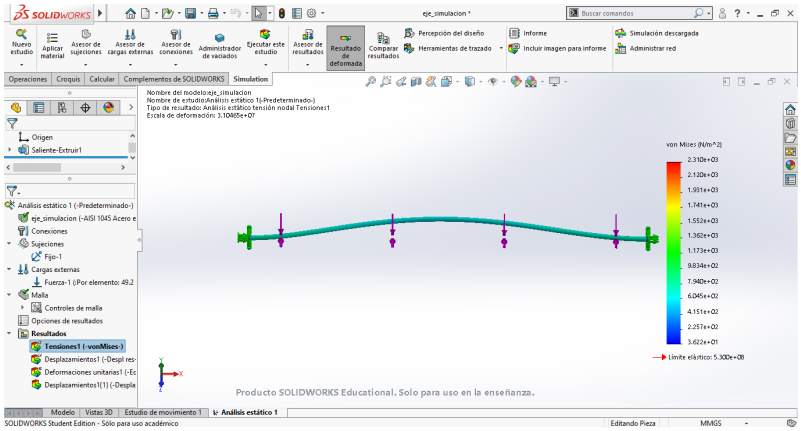


Figura 52: Resultados de la simulación.

Finalmente, se observan los resultados de la simulación en la Figura 52. Según el análisis, el esfuerzo máximo que sufre la barra es de $2.31kPa$, lo cual está dentro de un buen margen para la resistencia del acero 1065. Dicho esfuerzo se concentra en los extremos, lo cual conserva lógica debido a que es donde se tienen los soportes que sostienen a la barra.

6.11. Rediseño del efector final

El efector final del Cartesiano se había propuesto contemplando a la cámara fija a 90° paralela al piso. Sin embargo, para algunas pruebas en las que se tienen diferentes niveles de piso es necesario que la cámara puede moverse en los ángulos *pitch* y *roll*.

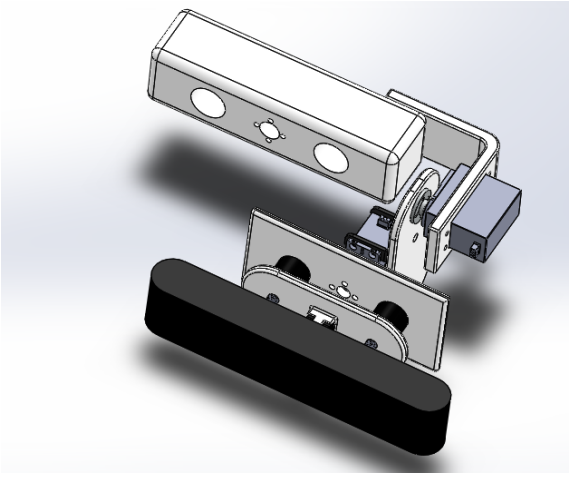


Figura 53: Diseño del efector final con servomotores para el control de la inclinación de la cámara.

Para lograr eso, se diseñó el mecanismo de la Figura 53. Se añadieron servomotores para controlar por PWM el ángulo en el que se posiciona la cámara.

6.12. Rediseño PCB

Debido a que se hizo un rediseño en el efector final para incluir el mecanismo con servomotores, también tuvo que cambiarse el diseño en la PCB.

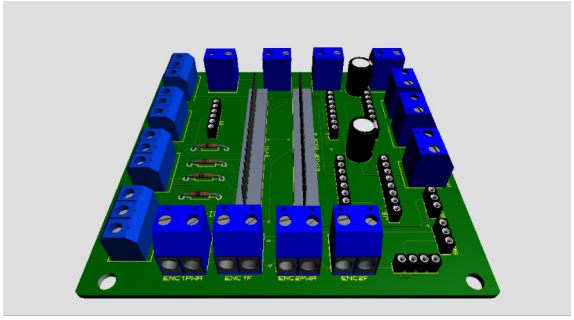


Figura 54: Diseño de la PCB para incorporar los servomotores.

En la figura 54 se puede observar el nuevo layout de la tarjeta. Otro de los factores que influyó en el cambio de la PCB fue el tiempo extra de diseño y de manufactura. Por ello se decidió cambiar el montaje del microcontrolador que previamente se estableció como de montaje superficial. En este nuevo diseño, se monta utilizando tiras de pines hembra para conectar las terminales de la tarjeta llamada “BluePill”.

6.13. Interfaz Humano-Máquina (HMI)

La Interfaz Humano-Máquina (HMI) se desplegará para el usuario a través de MATLAB, ya que en ese programa es más fácil integrar la parte de las comunicaciones en conjunto con el μC y la Raspberry Pi (Bluetooth y protocolo TCP/IP, respectivamente). Además, también permite comunicarse por medio de Bluetooth con los vehículos a escala a analizar.

Para el diseño de la HMI se tienen que establecer las funciones principales del programa:

- ◇ Inicio de sesión.
- ◇ Modo manual.
- ◇ Modo automático.
- ◇ Paro de emergencia.

El diagrama de clases correspondiente a estas funciones se encuentra en el Anexo 5. Estas funciones tienen que estar disponibles para que el usuario pueda operar el dispositivo con la mayor facilidad posible.

6.14. Código utilizado para el control del sistema

La programación del microcontrolador STM32 se hace en una máquina virtual de Ubuntu instalada en un sistema operativo Windows. Ésta se realiza sin usar algún IDE en particular, y con apoyo de las herramientas que provee la librería de código abierto llamada "libopencm3". El programador utilizado es un “dongle”

ST-link V2, el cual se conecta via USB a la computadora y tiene pines de conexión para conectar directamente con los pines del μC.

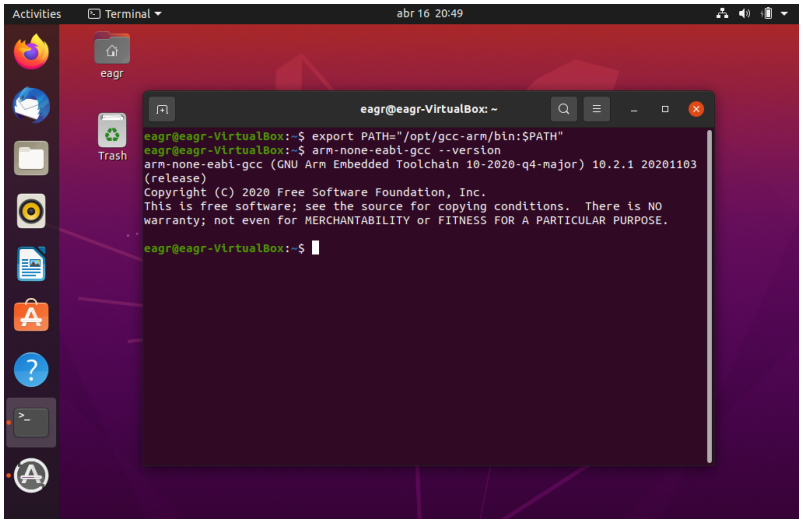


Figura 55: Compilación del código fuente a través del software libre libopencm3.

Para realizar el control se divide el programa principal en subprogramas con extensión .c como archivos fuente para el proyecto, de esta manera, se codifican por separado las tareas del control y se integran todas juntas en un archivo principal.

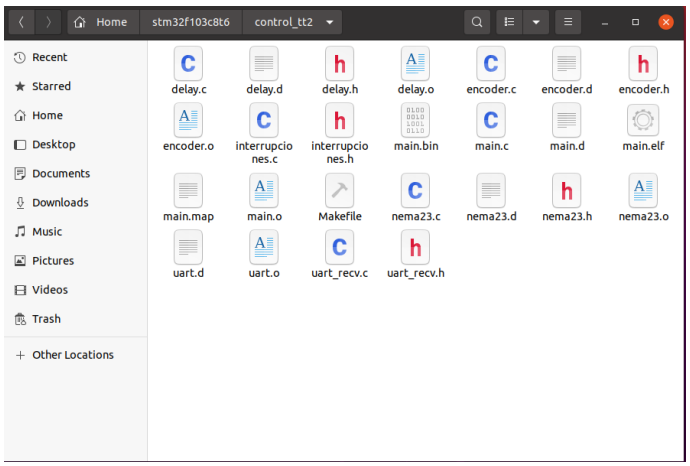


Figura 56: Carpeta que contiene los archivos fuente para el control del Cartesiano.

El código completo se encuentra en el Anexo 10, en el cual se muestran todos los archivos utilizados que se cargan al microcontrolador.

Para la programación del control del sistema, se toman en cuenta varios puntos. Lo primero a considerar fue el avance del tonillo ACME, el cual es de $l = 8mm$ (por vuelta). Entonces, si se quiere que avance 10cm (por ejemplo), primero se convierte la distancia deseada a milímetros tomando en cuenta que la entrada de datos del usuario en la interfaz de usuario es siempre en cm. Por lo tanto, siguiendo el ejemplo del avance de 10cm:

$$d_{deseada_{mm}} = 10cm \times 10\frac{mm}{cm} = 100mm$$

Lo siguiente es hacer el cálculo del número de vueltas necesarias en el tornillo para que el efector final llegue a esa distancia. Para ello, se realiza la siguiente operación:

,

$$n_{vueltas} = \frac{d_{deseada_{mm}}}{l} = \frac{100mm}{8mm} = 12.5vueltas$$

Por la divisón, la variable que almacena el número de vueltas calculadas debe ser de tipo flotante. No importa que el resultado sea decimal, la primer razón es la resolución que se tiene por el número de pasos que puede dar el motor por vuelta. El avance por paso es de $\pm.01mm$ por paso utilizando la configuración de micropasos de $\frac{1}{4}$ de paso (800 pasos por vuelta).

Después, se calcula el número de pasos que tiene que dar el motor. Como la configuración es de 800 pasos por vuelta, se realiza el siguiente cálculo:

$$pasos_{deseados} = n_{vueltas} \times 800 = 12.5 \times 800 = 10000pasos$$

Aunque $n_{vueltas}$ es flotante, al multiplicarlo por 800, el resultado siempre será un entero. Este valor se pasa al control del motor y éste gira hasta que el encoder detecte ese número de pasos con su correspondiente escalamiento en el número de pasos (explicado en la sección “Acondicionamiento De Señales“). En este ejemplo, se muestra el cálculo realizado por el μC:

$$pasos_{deseados_{encoder}} = pasos_{deseados} \times PPR_{ratio} = 10000 \times 3 = 30000pulsos$$

Donde PPR_{ratio} es igual a la razón entre los pulsos por vuelta que da el motor y los que entrega el encoder: $PPR_{ratio} = \frac{PPR_{encoder}}{PPR_{motor}} = 2400/800 = 3$.

Ya que se tiene el cálculo del número de pulsos que tiene que leer el encoder, el algoritmo utilizado para el control de la posición por medio de la lectura del encoder es el siguiente:

1. Se lee el valor actual del contador al que se encuentran conectadas las fases del encoder. Asumiendo que el valor es igual a 20 para el análisis, se almacena este valor en una variable, $pos_{inicial}$.

2. Ahora, el motor a pasos avanza el número de pasos calculados. Como existirán eventos de overflow en el contador del timer dependiendo de la posición en la que se encontraba inicialmente el encoder, se habilita previamente la interrupción para saber cuántas veces se desbordó el contador. Se llama a la variable $ovflw_{cnt}$.

3. Luego de terminar el movimiento, se lee nuevamente el valor actual del contador del timer y se almacena en la variable llamada pos_{final}. Se asume el valor como $pos_{final} = 65$

4. Ahora se hacen los cálculos. Como $pos_{inicial} = 20$, se sabrá cuántos pasos hubo hasta el overflow del timer (si es que hubo). Para el análisis, se asume que el desbordamiento del timer ocurrió tres veces : $ovflw_{cnt} = 3$. Del valor $pos_{inicial}$ al siguiente evento de overflow hay $2400 - 20 = 2380 pasos$. Después, quedan dos eventos que se leyeron en la cuenta. De los pasos obtenidos anteriormente hasta $ovflw_{cnt} - 1$ hay $2400 \times 2 = 4800 pasos$.

   Finalmente, del último evento de overflow a pos_{final} hay 65 pasos. Al terminar estas operaciones, se suman las cantidades para obtener la cantidad total de pasos leídos por el encoder en el movimiento real del Cartesiano:

$$pasos_{leidos} = 2380 + 4800 + 65 = 7245 pasos$$

   Sin embargo, se considera también el caso en el que no ocurrió ningún desbordamiento del timer, es decir, se revisa la condición if($ovflw_{cnt} > 0$).

Si la condición resulta en falso, no se completó una sola vuelta, por lo que la cantidad de pasos leídos se simplifica a

$$pasos_{leidos} = pos_{final}$$

5. Después de realizar el cálculo de los pasos obtenidos, se comparan los pasos leídos con los pasos calculados. Se tiene que obtener el error restando los pasos deseados a los pasos leídos por el encoder. Si el error es positivo, se mueve hacia enfrente (dir=1) el número de pasos calculados. Si el error es negativo, se mueve el motor hacia el lado contrario (dir=2) esa misma cantidad de pasos.

Finaliza ahí el algoritmo de control de posición manual. Para el modo automático se usa el control cinemático. Para ello, el controlador recibe por TCP/IP las coordenadas en centímetros a las que tiene que moverse. Ese será el valor deseado que entra en el sistema, y se prosigue a utilizar el mismo algoritmo que en la posición manual.
Para la recepción de los datos de posición por medio de Bluetooth, el procedimiento es el que se enuncia a continuación:

1. Primero se configura el UART en el μC para habilitar la comunicación serial con el módulo Bluetooth. El único canal que se necesita es el Tx en en el HC-05 y el Rx en el STM32 ya que solamente recibirá datos.

2. Del lado de la estación, primero se habilita la conexión a bluetooth del ordenador. Posteriormente, se empareja el dispositivo para que sea reconocible por la computadora y por MATLAB.

3. Una vez emparejado el dispositivo, se utiliza el comando $bluetooth()$ de MATLAB llenando el espacio en el paréntesis con el nombre del dispositivo al que se quiere conectar. En este caso, se conectará a dos módulos bluetooth desde la estación. La conexión a cada uno se realiza a través de la interfaz de usuario en donde al presionar el botón de “Conectar” se ejecuta un callback que utiliza la función $bluetooth()$ con el nombre de cada dispositivo. Se nombraron a los dispositivos como “cartesiano” y “Auto”.

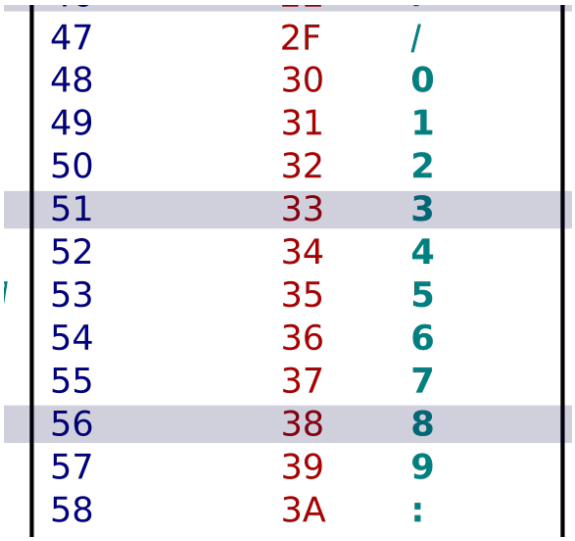
4. Una vez completada la conexión, se prosigue al envío de datos. El mensaje debe de contener varios datos de interés para el control del Cartesiano. La primer parte del mensaje contiene el modo de operación, el cual puede ser “manual” o “auto” (automático). Después, le prosiguen los datos en el siguiente orden:

```
1      mensaje = ...
           ',modo,dist_x,dir_x,dist_y,dir_y,grados_h,grados_v\n'
```

   En el mensaje se utiliza como *token* al caracter ‘ , ’, el cual sirve para separar la cadena en los diferentes valores para el control del Cartesiano. Como delimitadores, se utiliza el caracter ‘ , ’ para establecer el inicio de la cadena y se introduce un salto de línea ‘ \n ’ para saber que el mensaje ha terminado.

5. Ya enviado el mensaje, en el microcontrolador se usa la función incluida en la librería del firmware “$uart\_recv\_blocking()$” donde la salida de la función es un caracter. Lo que hace la función es que lee el mensaje transmitido desde el HC-05 al puerto del UART y realiza la lectura caracter por caracter mientras se almacena en un buffer. Posteriormente, se accede a la información del buffer utilizando la función “strtok( )” de C++ para separar la cadena. Para convertir los caracteres a enteros, se recorren los indices de la nueva subcadena de caracteres y a cada caracter se le hace la siguiente operación:

```
1      int_var = char_var - '\0'
```



47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:

Figura 57: Sección de la Tabla de código ASCII.

Para entender lo que realiza la operación anterior, se tiene que observar la sección de la Tabla de código ASCII en la figura 57. A la izquierda, en azul se tiene el valor decimal de los caracteres; en rojo, el valor hexadecimal; y en verde, el caracter que representa. Cuando se utiliza el operador ' - ' se realiza una operación entre enteros. Asumiendo que char_var = '8'. Como se realiza una resta, el compilador usa el valor entero de los caracteres: 56 (valor decimal del caracter '8') y 48 (valor decimal del caracter '0'). Al realizar la resta, obtenemos el resultado $int\_var = 56 - 48 = 8$ como número entero. La variable int_var es de tipo **int**.

6. Se realiza la conversión de caracteres a enteros tantas veces como sea necesario. Para obtener números de dos dígitos o más, se tienen que tomar los números individualmente y sumarlos como potencias de diez según su valor posicional.

Para posicionar a los servomotores en el ángulo deseado, se tuvieron que ajustar los valores de salida del Timer del STM32 de acuerdo con los límites físicos del mecanismo. Para ello, se siguieron los pasos mostrados a continuación:

1. Primero se tiene que tomar en cuenta el periodo de la señal PWM que requiere el servo para operar según los datos del fabricante. Para los motores

utilizados (MG995), se requiere un periodo de 20ms, es decir

$$f_{PWM} = \frac{1}{T_{PWM}} = \frac{1}{20ms} = 50Hz$$

2. El reloj interno del STM32 (SYSCLK) tiene una frecuencia de 72MHz. Internamente, el STM32 tiene un preescalador que conecta al SYSCLK con el bus APB1. Cuando se activa ese primer preescalador ($APB1\_PSC = 1$), la frecuencia del reloj del sistema se divide entre dos (salida de 36MHz). Después, el bus APB1 se conecta a los Timer a través de otro preescalador al que se le puede llamar preescalador "global". La salida del preescalador global multiplica por dos a la frecuencia de salida del bus APB1 ($36MHz \times 2 = 72MHz$) y esa es la frecuencia que entra a los preescaladores privados de los Timers.

3. Para facilitar el uso de los Timers, se utiliza el preescalador privado del TIM2 del STM32 para tener una frecuencia de 1MHz, esto es

$$f_{timer} = \frac{72MHz}{PSC} = \frac{72MHZ}{72} = 1MHz$$

4. Hay que tomar en cuenta la frecuencia que se requiere para controlar el servo dentro del cálculo del preescalador. Así, el preescalador deseado se calcula a partir de

$$PSC_{deseado} = \frac{f_{timer}}{f_{PWM}} = \frac{1MHz}{50Hz} = 20000$$

Así que en el código se utiliza la línea:

```
1       timer_set_prescaler(TIM2, 20000);
```

Como $f_{timer} = 1MHz$, se indica el ancho del pulso en microsegundos con la función:

```
1       timer_set_oc_value(TIM2, TIM2_OC1, OC_Value);
```

5. La resolución de los timers es de 16 bit (valores de 0 a 65535), pero según el periodo requerido para controlar al motor, el máximo valor teórico es

de 20000. Al introducir este valor en la salida del timer, el ciclo de trabajo de la señal es de 100 %, que corresponde a los 20ms que se definieron inicialmente. Según la hoja de datos del fabricante se requiere una señal de 1ms en alto para mover el servomotor a 0° y una señal de 2ms para mover el motor a 180°. Recordando que por los preescaladores elegidos, el valor que insertamos en la salida está en microsegundos, los valores de salida están limitados de la siguiente forma:

$$1000 \leq OC\_Value \leq 2000$$

6. De esta manera, el usuario ingresa en la interfaz la posición deseada en grados. Esos grados se convierten al OC_Value necesario haciendo una interpolación lineal en el código.

Para la conexión por medio de protocolo TCP/IP para recibir la posición actual del vehículo a escala, se utiliza MATLAB para la parte del cliente y a la Raspberry Pi como el servidor.

Para hacer la parte del cliente en MATLAB, primero se crea el objeto çliente"

```
1  cliente = tcpclient(direccion_servidor, puerto);
```

Para realizar la conexión, previamente se debió crear el servidor en la Raspberry Pi por medio de un socket. El servidor debe estar escuchando a través del mismo puerto. Ahora bien, cabe mencionar que el protocolo TCP/IP está diseñado para la confiable transmisión de datos, no necesariamente en tiempo real. Por ello mismo, es necesario realizar la lectura de los datos en paquetes.

El servidor se encontrará enviando datos de manera continua debido a que la lectura de la posición del vehículo a escala se realiza de esa forma. Esto llenará el buffer de transmisión de datos con el cliente. Para acceder al buffer y saber cuántos datos hay disponibles se recurre a la propiedad del objeto “NumBytesAvailable” con la línea de código

```
1   cantidad_de_datos= cliente.NumBytesAvailable;
```

Como el protocolo de comunicación utiliza delimitadores para determinar cuándo termina el mensaje, el número de bytes disponibles en el buffer siempre será el número total de caracteres del mensaje enviado por el servidor más un caracter correspondiente al delimitador. Para garantizar que la lectura sea continua, el cliente leerá del buffer mientras que el número bytes disponibles sea mayor que 0. Además, los datos se leen en paquetes de 10 y la posición se determina promediando las entradas.

6.15. Layout de la Interfaz Humano-Máquina

Para utilizar el Cartesiano, primero se necesita acceder al sistema. El cliente del proyecto solicitó que pudieran ingresar únicamente usuarios autorizados en los laboratorios de la universidad. Es por eso que primero se muestra una pantalla de inicio de sesión en la cual se validan los usuarios y contraseñas según lo incluido en un archivo de texto al que sólo tiene acceso el jefe de laboratorio.

Figura 58: Layout de la ventana de inicio de sesión.

En caso de que se quiera agregar un nuevo usuario, se cuenta con la función de registrar a un usuario. Para poder hacer el registro, primero se tiene que hacer un paso de autenticación ingresando la contraseña del laboratorio.

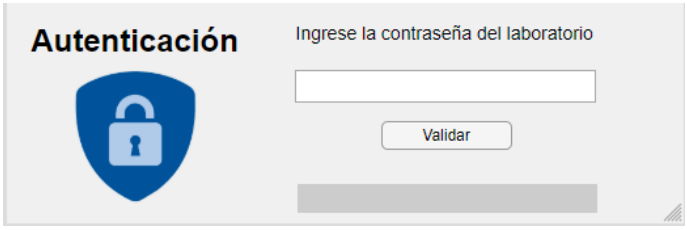


Figura 59: Layout de la ventana de autenticación para el registro de nuevo usuario.

Al ingresar credenciales válidas, aparece una ventana en la que se pueden introducir los datos para el registro del usuario.De este modo, el nuevo usuario también estará autorizado para utilizar la máquina.

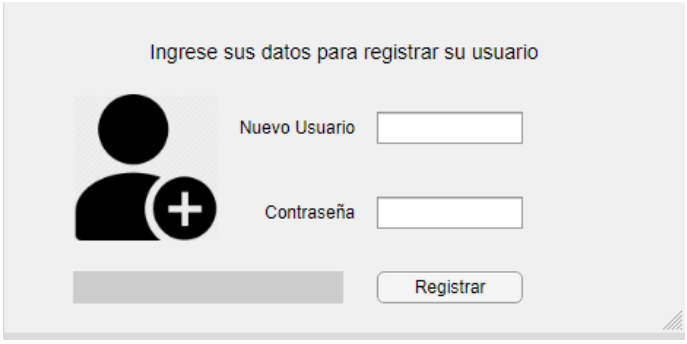


Figura 60: Layout de la ventana para registrar usuarios.

En caso de que las credenciales ingresadas sean incorrectas, se cuenta con un mensaje de error que alerta sobre la incompatibilidad entre el usuario y la contraseña ingresados.

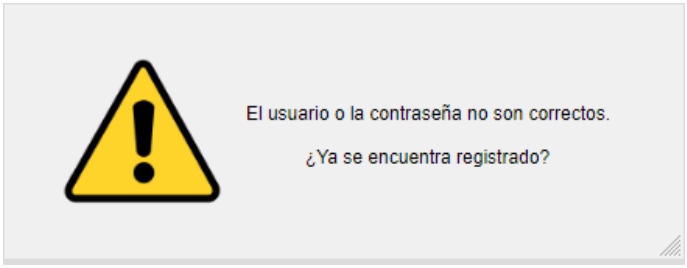


Figura 61: Layout de la ventana que alerta sobre un error en las credenciales ingresadas.

Ahora, bien. En la ventana principal se muestran los controles necesarios para utilizar todas las funciones del sistema Cartesiano. En el modo manual de operación, contamos con el posicionamiento directo por coordenadas de los eslabones

del Cartesiano así como del giro de la cámara. Además, se incluye un control para mover al vehículo a escala a una posición deseada. Se incluye también una gráfica en la cual se hace un mapeo 2D de la posición actual del vehículo dentro del sistema de coordenadas del sistema. Ése último dato es recibido por protocolo TCP/IP, proveniente de la Raspberry Pi sobre el Cartesiano.

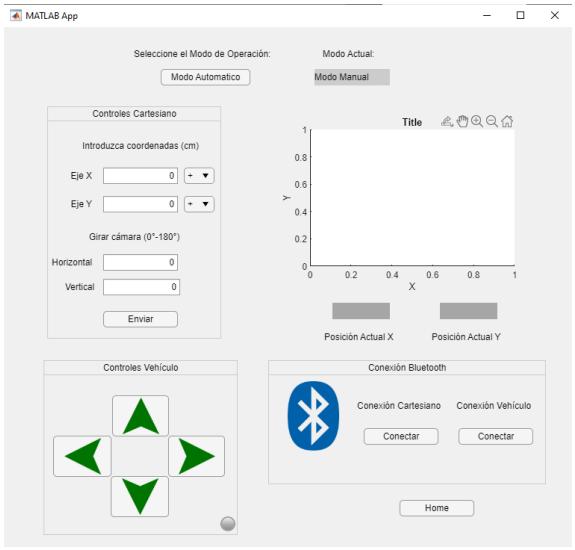


Figura 62: Layout de la ventana principal de control del Cartesiano.

Nótese también que se requiere establecer primero la comunicación inalámbrica entre los dispositivos para poder operar la máquina. Para ello se incluyó un panel en el cual se pueden activar y desactivar las conexiones a los moódulos Bluetooth tanto del Cartesiano como del vehículo a escala.

6.16. Conexión con la Raspberry Pi

Para acceder remotamente a la Raspberry, se utiliza el protocolo SSH. Para ello, desde la computadora de la estación se tiene que utilizar el programa Putty.

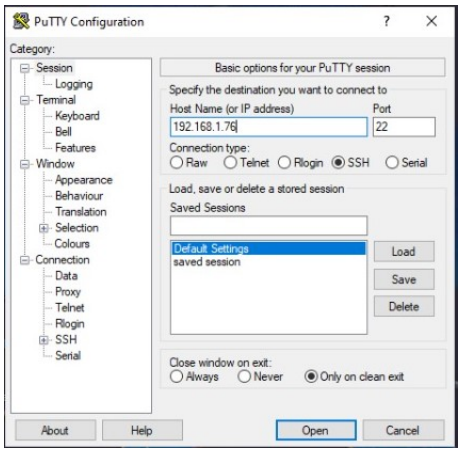


Figura 63: Configuración en el programa Putty para conexión remota.

Este tipo de conexión permite acceder a la terminal de la Raspberry Pi desde otro dispositivo sin necesidad de utilizar un monitor, teclado y ratón conectados a la tarjeta para controlar sus acciones. Para la conexión, se requirió configurar a la tarjeta con openSSH, librería que permite generar un servidor SSH para establecer la conexión. Posteriormente, se obtiene la dirección IP del dispositivo, y se habilita la conexión SSH por el puerto 22.

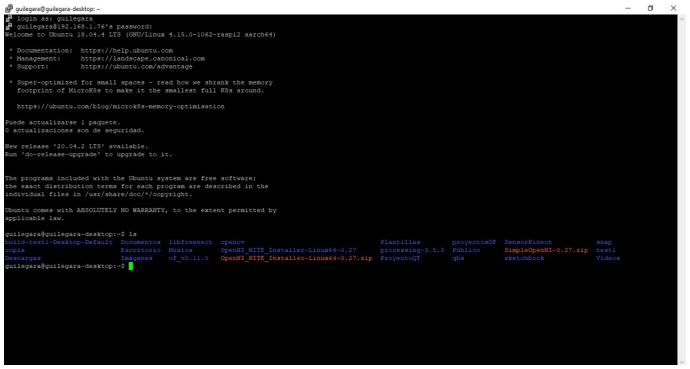


Figura 64: Terminal de la Raspberry Pi abierta desde la computadora en la estación.

Una vez realizada la configuración, se presiona el botón “Open“ en el cuadro de diálogo. Posteriormente, se solicita el nombre del usuario del dispositivo remoto y la contraseña. Tras haber introducido esos datos, se obtiene lo que puede apreciarse en la Figura 64. Así, se puede acceder a la imagen de la cámara sin

necesidad de enviarla a la interfaz. Cabe mencionar que si el equipo en la estación central utiliza Ubuntu como sistema operativo, no es necesario instalar Putty, ya que Ubuntu tiene SSH integrado.

6.17. Construcción de la estructura para sostener al Cartesiano

Se propuso un nuevo diseño en el cual se sostiene a la estructura principal por debajo para evitar que por el mismo peso se descuadren los elementos que conforman al sistema.

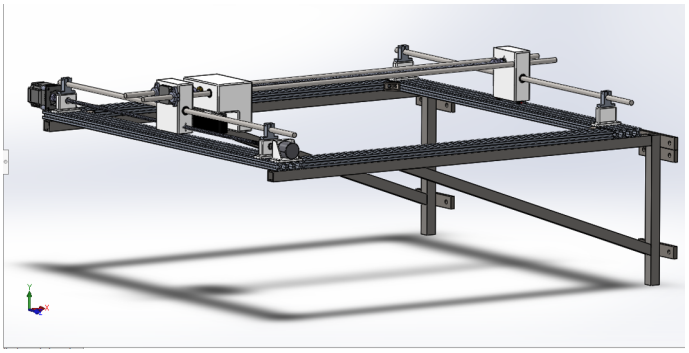


Figura 65: Nueva propuesta de diseño para sostener al sistema Cartesiano a la altura deseada.

Para ello, se realizó el diseño con PTR de $1\frac{1}{2} \times 1\frac{1}{2}$ pulgadas, calibre 14. Y solera para unir a una pared de concreto utilizando taquetes expansivos y tornillos adecuados para el concreto.

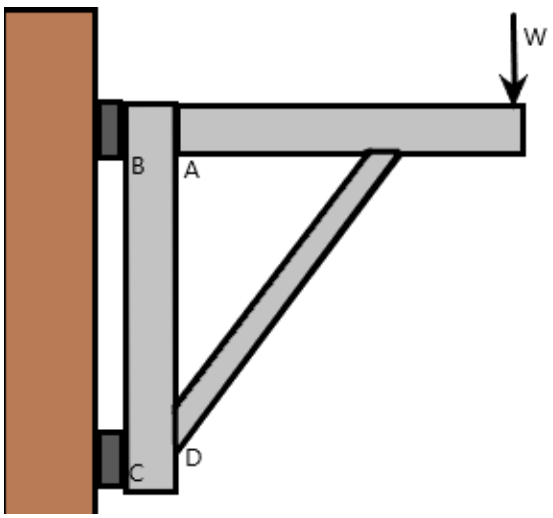


Figura 66: Diagrama para el cálculo de esfuerzos en las ménsulas.

En la Figura 66 se muestra un diagrama simplificado de la estructura de las ménsulas. La fuerza W es lo equivalente al peso máximo que tendrá que soportar la ménsula en ese punto. Esa fuerza es igual a la masa total de los componentes del sistema Cartesiano y la masa del PTR (para el cálculo de los taquetes) multiplicados por la constante de gravedad. Se considerará una masa de 15kg tomando en cuenta la masa máxima del sistema, entonces $W = (15kg)(9.8\frac{m}{s^2}) = 147N$. La fuerza W generará entonces el par máximo que se utiliza para el cálculo de la soldadura y la unión con el concreto.

El cálculo del par se obtiene con la distancia entre la fuerza W y el punto B, la cual es $d_{BW} = 950mm = 0.95m$. Entonces

$$M_B = (147N)(0.95m) = 139.65Nm$$

La fuerza que trata de extraer al tornillo de la unión depende de el par que se aplica al poner peso sobre el PTR y la distancia que existe entre el punto B y el punto en el que se coloca el tornillo en la solera. Esta distancia es $y_{tornillo} = 1.5cm = .015m$. Entonces, el cálculo está dado por

$$F_{tornillo} = \frac{M_B y_{tornillo}}{2{y_{tornillo}}^2} = \frac{(139.65)(.015)}{4(.015)^2} = 2.3kN$$

El 4 introducido en el denominador se coloca debido a que son 4 los tornillos que se unen a la solera en ese punto. Sin embargo, también se utiliza un par extra de tornillos en la parte inferior que ayudan al soporte de la estructura y repartiendo más el peso.

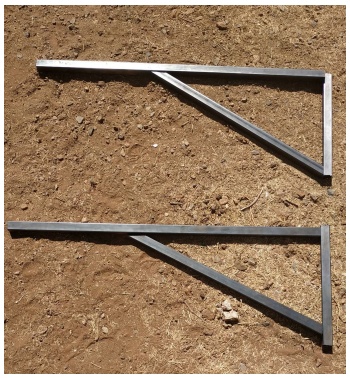


Figura 67: Construcción de las ménsulas que sirven de soporte al Cartesiano.

El corte del PTR se realizó con una esmeriladora. Posteriormente, se realizó el pulido de los perfiles para asegurar la correcta unión entre los elementos.

Figura 68: Corte y pulido de los PTR.

La soldadura de los perfiles se hizo con una soldadora inversora de 100A, la cual está configurada para trabajar en polarización directa entre 60-65A con un electrodo revestido E-6013 de 3/32".

Figura 69: Configuración de la máquina y técnica de soldadura empleada.

Para soldar los PTR se comenzó haciendo cordones de soldadura en filete. Sin embargo, algunas partes del material se perforaban por lo que se optó hacer la soldadura por punteo en algunas uniones.

Para poder colocar las ménsulas en la pared, se hicieron agujeros en la solera empleando una broca de acero de alta velocidad (HSS) de 3/8” (aproximadamente 10mm). No obstante, el filo de las brocas se desgastaba con facilidad y

era difícil hacer las perforaciones. Se optó por utilizar una broca de 1/4” de alta velocidad con recubrimiento de cobalto para perforar fácilmente el material sin desgastar el filo. Posteriormente, se empleó la broca de 3/8” para asegurar que entren los taquetes expansivos a través de la solera y hasta la pared.

Figura 70: Instalación de las ménsulas a la altura deseada para las pruebas.

6.18. Construcción del sistema Cartesiano

Para la construcción del Cartesiano, primero se cortaron los perfiles de aluminio 2060, los cuales se compraron con una medida de 1m de longitud. Si se dejaban de esa medida, las varillas lisas que se usan como guía lineal no alcanzaban a entrar bien en los carros del Cartesiano, por lo que tuvieron que cortarse para ajustar las medidas de los componentes.

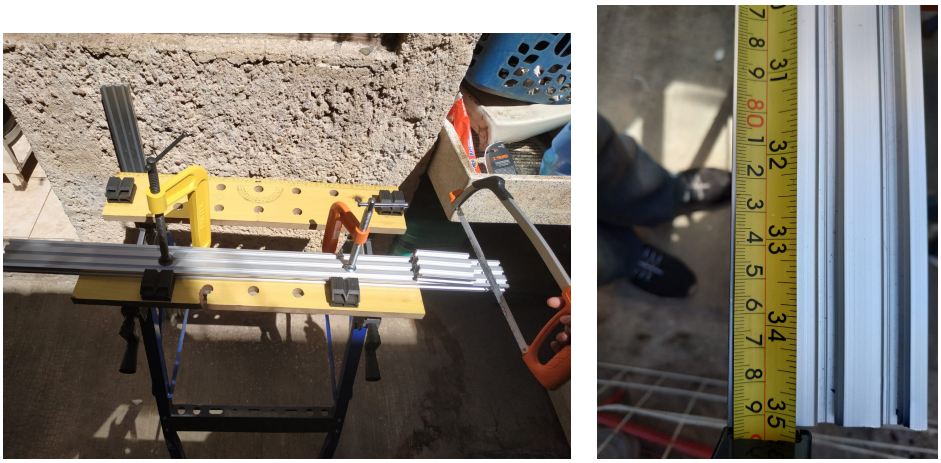


Figura 71: Ajuste de las medidas de los perfiles de aluminio 2060.

Como se muestra en la figura 71, el corte de los perfiles se hizo a 89cm para que las guías lineales entraran bien en los carros de desplazamiento de la carga del Cartesiano.

En cuanto a las piezas impresas en 3D, el método de unión utilizado para los rodamientos lineales fue por interferencia. Se hizo uso de una prensa tipo C (debido a las limitaciones de equipo y herramientas) para presionar el rodamiento contra el orificio del carro de la carga en el eje y, el cual por diseño es de un diámetro ligeramente menor al del rodamiento, para introducir el elemento y mantenerlo fijo.

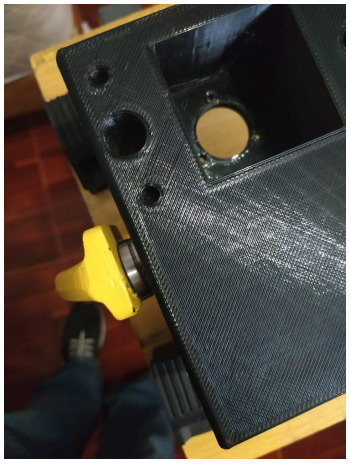


Figura 72: Unión del rodamiento con el carro de la carga en el eje y.

Posteriormente, se construyó el sistema de movimiento para el eje y empleando las piezas impresas en 3D para sostener al encoder y a los rodamientos.

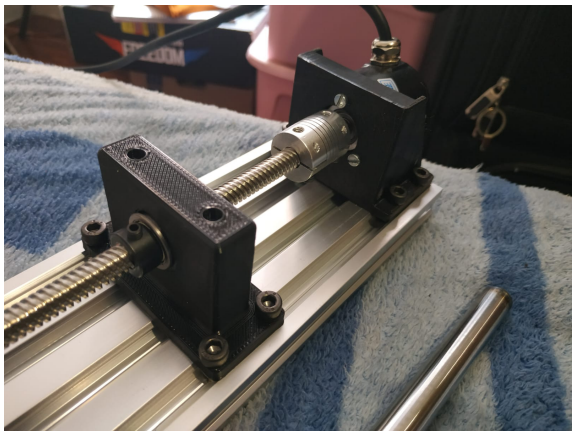


Figura 73: Submecanismo para el eje y de movimiento.

Luego, se añadieron las guías lineales junto con el motor NEMA 17 y el otro encoder para formar el eje x de movimiento, sin tomar en cuenta todavía al efector final.

Figura 74: Construcción del Cartesiano sin el efector final.

En la figura 74, se muestra que la estructura está elevada a cierta altura respecto al piso debido a que la placa que se utiliza para sostener al NEMA 23 sobresale del nivel del perfil de aluminio 2080. Por ello fue necesario ajustar esa altura para realizar las pruebas.

Figura 75: Efector final montado en el Cartesiano.

Se colocó el efector final en el Cartesiano. Se unieron dos rodamientos lineales por interferencia en cada uno de los orificios designados, como se aprecia en la Figura 75.

Figura 76: Montaje de la cámara para colocarla en el efector final.

En la figura 76 se muestra el ensamble de la cámara en el efector final para permitir la rotación de los servomotores para controlar la inclinación de la cámara.

Figura 77: Sistema Cartesiano montado en las ménsulas.

Finalmente, en la Figura 77 se muestra el ensamblaje en conjunto del Sistema Cartesiano con las ménsulas. El cual muestra el producto final mecánico.

6.19. Métodos de unión

Para unir los perfiles de aluminio se utilizan soportes esquina de 90° unidos al perfil con tornillos Allen M5 utilizando tuercas tipo T. La misma combinación se usa para unir los soportes fabricados por impresión 3D con los perfiles de aluminio.

Para hacer la unión entre piezas 3D se hace uso de tornillos autorroscantes M5 de 10mm y 16mm de largo. También se utilizan tornillos M3 de 10mm para unir el motor NEMA17 con el carro del eje x, igual que para el encoder colocado en el mismo eje de movimiento.

6.20. Acondicionamiento de señales

Para utilizar los interruptores de final de carrera, normalmente es necesario utilizar un filtro para deshacerse del ruido mecánico generado al presionarse el interruptor. Sin embargo, el fabricante de los interruptores incluyó un filtro RC para eliminar el rebote mecánico al accionarse el interruptor.

Figura 78: Módulos de interruptores de final de carrera acondicionados.

De esa manera, se asegura que las lecturas obtenidas a partir de las interrupciones del microcontrolador son correctas y no hay falsos estados en ALTO. Sin embargo, es pertinente realizar pruebas ya con el sistema montado para asegurar que siempre se obtiene la respuesta esperada ante el disparo de las interrupciones, así como el posicionamiento de “HOME“ cuando se quiere resetear la máquina.

Asimismo, para la utilización del driver del motor NEMA 23 con el STM32, fue necesario utilizar un convertidor de nivel lógico. Los driver TB6600 funcionan solamente recibiendo señales de 5V en los pines de control DIR y PUL. Los GPIO del STM32 trabajan a máximo 3.3V de salida. Para resolver este problema, se

proponen dos soluciones. Una es utilizar una resistencia de pull-up a 5V y configurar los GPIO como *open-drain*. La otra, es utilizar módulos de conversión de nivel lógico bidireccionales.

Para acondicionar la lectura de los encoder, se hizo una operación por software para hacer comparables los valores. El encoder, utilizando dos fases, entrega 2400 pulsos por revolución. En cambio, configurando el driver de los motores 1/4 de paso (micropasos), los motores dan 800 pulsos por revolución. La relación de PPR (Pulsos Por Revolución) entre dispositivos es de

$$PPR_{ratio} = \frac{PPR_{encoder}}{PPR_{motor}} = \frac{2400}{800} = 3$$

Este dato es importante para el momento de comparar por código el número de pasos deseados con el número de pasos reales que se dan. De esta manera se pretende evitar la pérdida de pasos por fricción y aumentar así la precisión del mecanismo. Para ello se contemplaron dos opciones:

1. Primera opción: Dividir los pulsos del encoder entre 3 para que sean la misma cantidad de pulsos por vuelta.

2. Segunda opción: Multiplicar por 3 a la variable que asigna los pasos que debe dar el motor para que sea comparable con los pasos del encoder. Así, se comparan los pasos que ha leído el encoder con el valor teórico. El motor avanza hasta que la lectura real iguale a la teórica.

Se consideró mejor opción la segunda ya que al hacerlo así no se le quita la resolución al encoder. Otra ventaja es que el proceso de cómputo requerido para la multiplicación es menor que para la división, así como la división puede resultar en valores con punto flotante que requieren ser redondeados. Por último, se le da prioridad a la lectura real del encoder, ya que el sistema puede perder pasos por la fricción del mecanismo.
En cuanto a los servomotores, éstos funcionan con los 3.3V del microcontrolador por lo que no fue necesario utilizar un convertidor de voltaje.

6.21. Programación del vehículo autónomo a escala

El vehículo a escala utilizado en este proyecto se construyó utilizando un módulo Bluetooth HC-05 para recibir la velocidad a la que deben de ir las llantas. También se usa un módulo L298N para control de giro de los motores. Por último, se cuenta con discos de plástico que funcionan como encoders para obtener la velocidad real de las llantas.

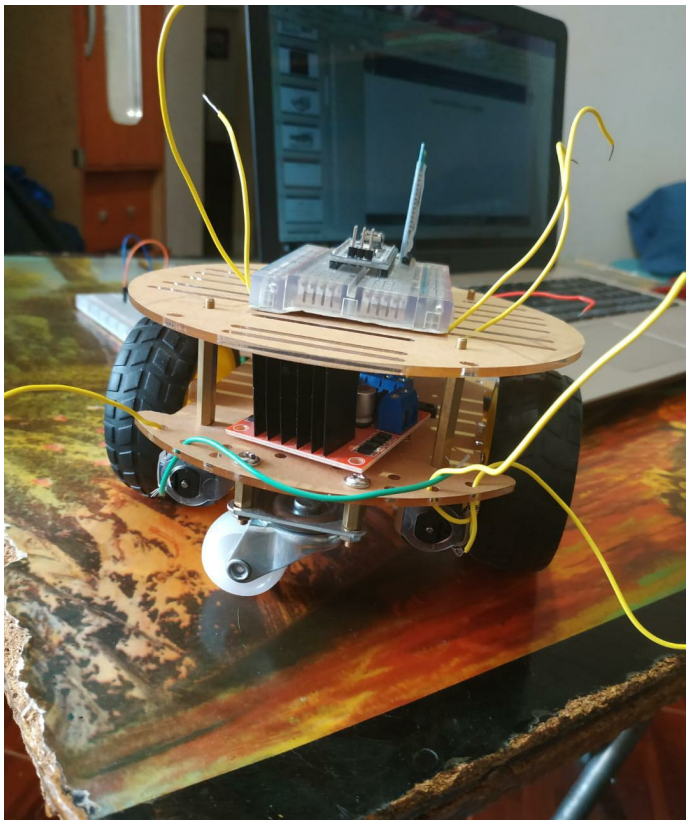


Figura 79: Construcción del vehículo a escala para las pruebas.

Dentro de los alcances del proyecto, no se incluye la implementación de la odometría para el control de posición del vehículo. Simplemente, se recibe el sentido de giro de los motores y una velocidad fija a la que debe de avanzar. Las interrupciones están programadas para calcular la velocidad, pero no se obtiene la posición con esos datos. El código utilizado para controlar al vehículo a escala se encuentra en el Anexo 12.

Capítulo 7. Análisis y validación de resultados

Por el momento, el sistema de control de posición por introducción manual en la HMI funciona correctamente en cuanto a la transmisión y recepción de datos. Asimismo, la pérdida de pasos en el tornillo durante la transmisión de movimiento por fricción se contrarresta de buena manera con el uso de los encoder. Éstos fueron de gran ayuda para el correcto posicionamiento del sistema Cartesiano en las coordenadas introducidas.

En cuanto a la construcción mecánica, la unión de los rodamientos lineales no fue la mejor, ya que es necesario que exista cierta holgura en la unión para el correcto ensamble del sistema. Este fallo en la construcción ocasiona que no se desplace correctamente el sistema lineal y exista error en el posicionamiento del efector final.

La desventaja de utilizar las ménsulas para posicionar al sistema Cartesiano a la altura necesaria es que el sistema es fijo. Como es necesario fijar el ensamble a la pared por medio de taquetes expansivos, la altura no se puede modificar una vez colocado el sistema, por lo que es inconveniente si se desea realizar pruebas con diferentes altura. Además, al colocarlo en la pared se pierde una parte del campo de visión de la cámara.

Las piezas impresas en 3D resultaron de buena calidad al haberse utilizado PETG para su fabricación. Tienen buena resistencia y durabilidad, y la principal ventaja es que, al ser el PETG un material flexible, se facilitan la manipulación y modificación de las piezas durante el ensamble del sistema.

El uso de acopladores flexibles resultó muy bien debido a que sí existe un desalineamiento entre los elementos del sistema de movimiento lineal debido a las

tolerancias de fabricación y construcción. El uso de estos acopladores permitió eliminar esos inconvenientes y operar de buena manera la transmisión de movimiento.

El layout de la HMI se ve muy saturado por las distintas funciones y opciones que requieren estar disponibles para el modo manual de operación, por lo que se necesitaría revisar una forma de presentar de manera más amigable estas opciones. Una opción sería utilizar menús desplegables como los que se muestran en las ventanas de los programas de Windows, o utilizar una ventana con pestañas similar a las disponibles en LabVIEW.

Referente a los motores, como se menciona en el Desarrollo del Proyecto, el STM32 no se podía utilizar directamente con el driver TB6600 y se propuso una solución utilizando una configuración distinta en los pines del microcontrolador. Sin embargo, la corriente que demandan los pines de control del TB6600, así como el nivel de voltaje, tienen un umbral lógico muy estrecho. La configuración open-drain no funcionó porque para alcanzar un voltaje de 5V en la salida del pin, la resistencia tenía que ser de un valor muy bajo (alrededor de 10Ω según las mediciones realizadas). Utilizar un resistor de valores pequeños es perjudicial para el μC por la alta corriente que deja pasar.

La solución para ese problema fue utilizar el driver DRV8825 también para el motor NEMA23, ya que el DRV8825 funciona con 3.3V y 5V directamente sin necesidad de añadir una interface. La desventaja de ello es que el driver opera siempre a los valores máximos y llega a tener muy altas temperaturas. Se ajustó el valor del voltaje de referencia del driver según la corriente máxima del motor (2.8A para el NEMA23 y 1.68A para el NEMA17) y con eso se garantiza que funcione el motor. Para mantener el driver a una temperatura más baja, se utilizan en conjunto un disipador colocado sobre el IC y un ventilador de 5V.

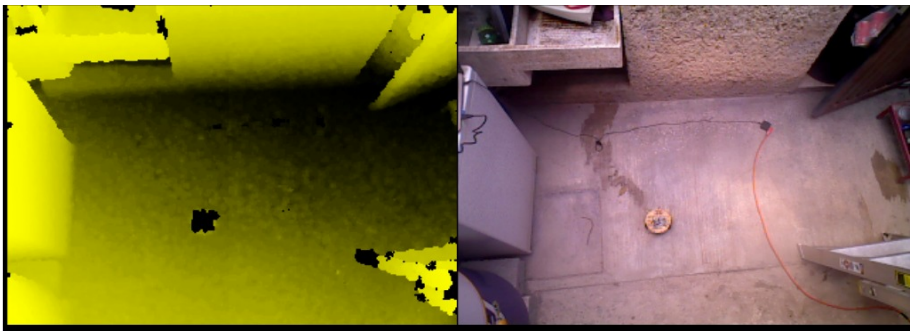


Figura 80: Imagen obtenida desde la cámara montada en el Cartesiano.

Se puede observar en la Figura 80 la imagen obtenida por la cámara vista desde la terminal. Se aprecia que la posición en la que se encuentra el vehículo a escala tiene un color distinto a lo demás en el histograma de profundidad, por lo cual el vehículo es detectable desde la altura a la que se encuentra el efector final del sistema.

Figura 81: Medida real del campo de visión fijo de la cámara.

Además, se realizaron medidas de la distancia que abarca tanto horizontal como verticalmente la cámara desde la altura fijada. Las medidas son las que se muestran en la Figura 81. En vertical, tomando como referencia los objetos en los extremos de la imagen 80 se midieron 213cm. En horizontal, se midieron aproximadamente 132cm. En contraste con el cálculo realizado inicialmente, el cual contemplaba que con la cámara a dos metros de altura, el rectángulo que forma el campo de visión de la ASUS sería de 2.22m$\times 1.66$m. Por lo que el error es de 9cm en vertical y 34cm en horizontal.

La diferencia mayor es en parte causada por el uso de las ménsulas, ya que el campo de visión está limitado por la pared que sostiene la estructura. Sin embargo, el campo de visión total de la cámara es de $3.02m \times 2.46m$, el cual es suficiente para los análisis y las pruebas que buscan realizarse con este sistema.

En cuanto a la parte de control, para ambas modalidades se utiliza el encoder como sensor para obtener el error en la posición. Si bien, en el modo manual de operación el control es a lazo abierto debido al principio de operación del motor a pasos, la programación del sistema incluye la corrección secuencial del número de pasos teóricos comparando con los pasos leídos por el encoder de cuadratura. Ese mismo algoritmo se utiliza para el lazo cerrado, pero en este caso la entrada proviene de la salida de la cámara.

Capítulo 8. Conclusiones

En cuanto al diseño de la estructura, ésta varía un poco respecto a lo planteado inicialmente. Se planteaba la posibilidad de instalar el sistema Cartesiano en el techo de uno de los laboratorios de la UPIIZ. Sin embargo, debido a las dificultades por la pandemia y cuestiones de permisos, se planteó realizar un diseño alternativo que brinde la posibilidad de posicionar el sistema en la altura deseada. Este sistema es posicionable con el uso de ménsulas que permiten que se coloque el prototipo en la pared, aunque sacrifique un poco el campo de visión total del sistema, sirve para realizar la demostración de la funcionalidad del prototipo.

Otro diseño alternativo que aún se encuentra en diseño es el de utilizar una configuración estilo “canasta de basketball“, el cual permite que el campo de visión total esté libre y que el sistema pueda posicionarse a 2m de altura como se planteó en un inicio. Sin embargo, esto implicaría un costo más elevado y requeriría una mayor cantidad de tiempo en la manufactura de los componentes. Por eso mismo, aún está en evaluación la viabilidad de esta alternativa.

Sobre la transmisión de potencia, el utilizar un tornillo ACME es una buena solución debido a que la distancia de viaje de los carros no es muy grande (menor a 1m) por lo que el tornillo no sufre deflexiones indeseadas por el tamaño. Asimismo, la velocidad máxima establecida para el tornillo es razonable ya que no se queda considerablemente atrás ante la velocidad de los vehículos autónomos analizados por la cámara.

La disposición física de los elementos del sistema Cartesiano puede mejorarse. Se eligió la configuración inicial contemplando que el sistema podría colocarse en el techo de un laboratorio de la UPIIZ. Sin embargo, tras las modificaciones pertinentes, la configuración pudo haber sido de maneras diferentes y quizá mejores.

A pesar de ello, el modo en que están posicionados físicamente los elementos del sistema es funcional.

Para lograr que el sistema se posicione en el espacio a partir de un control cinemático se optó por usar un sistema de control a lazo abierto para el modo manual, ya que el motor a pasos funciona bien en esta configuración. La pérdida de pasos que puede suceder por la fricción entre los elementos se compensa con la utilización de los encoders que se encuentran ya en el diseño, los cuales sirven para asegurarse de que el tornillo avanzó la cantidad de pasos que el μC determinó adecuada.

En cuanto al modo automático, como se utiliza el código proveído por el cliente, se obtiene una retroalimentación visual de la posición del vehículo analizado, lo cual hace que utilizar un control a lazo cerrado sea la mejor opción para posicionar el efector final en las coordenadas deseadas. Se utilizan las ecuaciones obtenidas de la cinemática directa e inversa para diseñar el control cinemático y posicionar los eslabones de manera correcta tomando en cuenta el marco de referencia inicial.

Las ventajas de utilizar el μC propuesto son que permite realizar el conteo de los pulsos de los encoders por medio del hardware, lo cual libera de carga al software y puede operar con mayor eficiencia. La clara desventaja es que es más difícil programar este tipo de microcontroladores, pero las especificaciones del mismo ayudan mucho a dar solución a los problemas del sistema diseñado, como lo son la velocidad de respuesta y la precisión.

Referente a la integración del sistema de visión con el sistema Cartesiano, la utilización de una Raspberry Pi brinda muchas ventajas. Se seleccionó este ordenador de placa simple porque permite utilizar la cámara RGB-D con librerías de código abierto integradas en conjunto: OpenNI, OpenCV, libfreenect. Los componentes de software necesarios se consiguen a través de repositorios de GitHub

en la web.

En resumen, las posibles mejoras de diseño radican en el aspecto mecánico. Tanto en la selección de elementos de máquina como en la configuración de la disposición física del dispositivo. Pese a ello, todos los elementos seleccionados durante el diseño son funcionales en conjunto y permiten monitorear vehículos a escala a través de la interfaz en la estación.

Capítulo 9. Trabajo a futuro

En cuanto a la utilización del microcontrolador para el control del Cartesiano mediante la recepción de datos vía Bluetooth, se pueden explotar al máximo las capacidades del STM32. Al ser un microcontrolador de 32 bits y con una velocidad considerable, puede aumentarse la velocidad de recepción hasta 115200 baudios sin problemas, incluso se podría experimentar con otras velocidades superiores. Asimismo, explorar la posibilidad de aumentar la resolución en las lecturas de los sensores y en el movimiento de los motores por medio de software. Otra implementación interesante sería la de aplicar control en tiempo real utilizando FreeRTOS.

Otro aspecto de mejora es temporizar la recepción de datos y el procesamiento de los mismos de acuerdo a los movimientos del sistema Cartesiano. Para evitar lecturas erróneas o con datos no significativos, se podría adaptar el envío y recepción de datos obtenidos de la cámara al tiempo que tarda el efector final en colocarse en la posición deseada. De esta manera, podrían evitarse errores durante el procesamiento de la imagen.

Referente a la construcción del soporte para el sistema Cartesiano, queda como trabajo a futuro el poder implementar una estructura para colocar al sistema en un laboratorio de la UPIIZ. Se optó por utilizar las ménsulas debido a que por la pandemia tenía que realizarse el trabajo desde casa y fue la opción más acertada por el espacio disponible. Sin embargo, para utilizar el sistema en la UPIIZ se tiene que realizar el diseño tomando en cuenta la construcción del laboratorio, idear una forma de sostener al sistema Cartesiano utilizando la estructura del techo o diseñando una estructura nueva para colocar al sistema a la altura deseada.

Para la transmisión de los datos de la cámara y mostrar la imagen, se aprovechó la instalación de Ubuntu en la RaspberryPi para incorporar ROS, el cual es un conjunto de librerías y herramientas para construir aplicaciones de robótica. La comunicación entre nodos, tema y suscriptor se realiza a través de un “master” y un cliente. Similar a la estructura de cliente-servidor, se puede hacer un uso más extensivo de las herramientas que provee ROS para hacer estudio y diseño de trayectorias, SLAM, etc.

Por último, existen varias mejoras que pueden hacerse a la GUI en MATLAB. La retroalimentación visual del modo de operación puede hacerse de una manera más intuitiva que la utilizada con etiquetas de texto. Además, en sistemas de tipo Cartesiano es común que para posicionar al efector final se pueda cambiar la posición individual de cada eje de movimiento de manera continua, contrario a lo utilizado para el proyecto. En la interfaz utilizada, el envío de los datos para el movimiento se hace en conjunto, ya que el mensaje enviado por Bluetooth contempla a todos los ejes de movimiento a la vez. Para poder hacer el movimiento individual, se puede modificar el envío de datos utilizando un botón por cada motor y cambiando la línea de caracteres que se envía respetando la estructura ya establecida para el mensaje que espera el microcontrolador para ejecutar el movimiento.

Fuentes de consulta

[1] INEGI. (2019) Seguridad pública y justicia: Accidentes de tránsito. [Online]. Available: https://www.inegi.org.mx/temas/accidentes/

[2] F. Arnéz and A. Villazón, "Virms: Un sistema de información vehicular y monitoreo de carreteras," *Investigación & Desarrollo*, vol. 2, no. 14, pp. 92–105, 2014.

[3] P. S. Hernández Chánez, "Sistemas inteligentes de transporte:situación actual y prospectiva," Feb 2014. [Online]. Available: https://repositorio.unam.mx/contenidos/431651

[4] "Sistema de posicionamiento global." [Online]. Available: https://www.gps.gov/spanish.php

[5] R. L. Gordon, *Intelligent Freeway Transportation Systems*. Springer US, 2010. [Online]. Available: https://doi.org/10.1007/978-1-4419-0733-2

[6] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020.

[7] G. Borenstein, *Making things see : 3D vision with Kinect, Processing, Arduino, and MakerBot*. Sebastopol: O'Reilly Media, 2012.

[8] S. J. D. Prince, *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 2012.

[9] A. Cherubini, F. Chaumette, and G. Oriolo, "Visual servoing for path reaching with nonholonomic robots," *Robotica*, vol. 29, no. 7, pp. 1037–1048, Apr. 2011. [Online]. Available: https://doi.org/10.1017/s0263574711000221

[10] G. Garcia, J. A. Corrales Ramon, J. Pomares, and T. Fernando, "Survey of visual and force/tactile control of robots for physical interaction in spain," *Sensors*, vol. 9, 12 2009.

[11] C. I. Esparza Castañeda, “Implementación de algoritmos de conducción autónoma en vehículos a escala 1:10,” Trabajo Terminal, Ingeniería Mecatrónica, UPIIZ, Zacatecas, 2005.

[12] O. O. V. Villegas, M. Nandayapa, and I. Soto, *Advanced Topics on Computer Vision, Control and Robotics in Mechatronics*. Springer International Publishing, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-77770-2

[13] G. Arechavaleta, A. Morales-Diaz, H. M. Perez-Villeda, and M. Castelan, “Hierarchical task-based control of multirobot systems with terminal attractors,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 334–341, Jan. 2017. [Online]. Available: https://doi.org/10.1109/tcst.2016.2549279

[14] “Award winning motion capture systems,” Jun 2020. [Online]. Available: https://www.vicon.com/

[15] “Motion capture systems.” [Online]. Available: https://optitrack.com/

[16] R. G. Budynas, J. K. Nisbett *et al.*, *Shigley’s mechanical engineering design*. McGraw-Hill New York, 2008, vol. 8.

[17] H. Gurocak, *Industrial motion control: motor selection, drives, controller tuning, applications*. John Wiley & Sons, 2015.

[18] T. J. Maloney, *Electrónica industrial moderna*. Pearson Educación, 2006.

[19] S. B. Niku, *Introduction to robotics: analysis, control, applications*. John Wiley & Sons, 2020.

[20] P. Millett, “Motor-driver pcb layout guidelines (part 1),” 2017. [Online]. Available: https://www.electronicdesign.com/power-management/article/21805384/motordriver-pcb-layout-guidelines-part-1

Anexo 1. Código utilizado para calcular el par necesario para el movimiento.

```
%Programa para calcular el torque necesario para mover la carga
%en el sistema Cartesiano

%Se limpia el espacio de trabajo y la ventana de comandos
clc
clear all

%**********************Datos sobre el motor************************

J_motor=3e-5; %Inercia del motor (buscar en datasheet) en kg-m^2
L_motor=0;       %Longitud del motor en metros
D_motor=0.00635;   %medida del diametro del eje del motor en metros


%******************Momento de inercia del encoder******************

J_encoder=1e-6; %buscar en la datasheet del encoder [kg-m^2]


%**********************Datos sobre el cople************************

di= .005;    %Diametro interior del cople (se toma en cuenta el que
             %va al eje del motor
do= .019;             %Diametro exterior del cople en metros
rho_cople= 2700;      %Densidad del material del cople en kg/m^3
L_cople= 0.025;       %Longitud del cople en metros
c1= di/2;             %radio interior
c2=do/2;              %radio exterior
J_cople = ((L_cople*rho_cople*pi)/2)*(c2^4-c1^4);%Inercia del ...
                                                 %cople[kg-m^2]

```

```
%**********************Datos tornillo ACME************************

Lx=0.99;                  %longitud del tornillo en metros
rho_tornillo=7800;        %densidad del material del tornillo(kg/m^3)
D_tornillo=0.008;         %diametro del tornillo en metros
r_tornillo=D_tornillo/2; %radio del tornillo en metros
l=0.008;                  %m/rev
p=1/l;                    %paso del tornillo
f = 0.16*2;               %coeficiente de friccion entre el tornillo
                          %y la tuerca.
                          %acero engrasado con tuerca de bronce p.404
                          %Shigley Novena edicion
alpha=atan(l/(pi*D_tornillo));   %angulo de helice en radianes
alpha = (alpha*180)/pi;          %angulo de helice en grados
beta=atan(f);                    %beta en radianes
beta=(beta*180)/pi;              %beta en grados
eta = tand(alpha)/tand(alpha+beta);    %eficiencia
N_tornillo= 2*pi*p;                    %relacion de transmision

%Inercia del tornillo [kg-m^2]
J_tornillo = (pi*Lx*rho_tornillo*r_tornillo^4)/2;

%************************Datos rodamiento*************************

f_c=0.15;                          %friccion del collarin
d_c=.0258;                        %diametro del collarin


%***************Pesos de los carros y de la carga*****************

m_carrox1 = 0.5;     %masa del carro en el eje x1
m_carrox2 = 0.5;     %masa del carro en el eje x2
m_carroy  = 0.5;     %masa del carro en el eje y
m_guias   = 1.07*2;  %masa de los ejes acerados en el eje y
m_tornilloy = .430;  %masa del tornillo en el eje 'y' en kilogramos
m_rod_tuercas = 0.026*4+0.1*2;   %masa de los rodamientos y
                                 %tuercas en el carro y

m_carga=.540+.045+.300+.055 ;   %masa de la carga en el efector final
```

```
70                                  %en kilos camara+rasp+tilt+servo
71
72  m_motory = 0.350;               %masa del motor en el eje y
73  m_encodery = 0.350;             %masa del encoder en el eje y
74  g= 9.81;                        %Valor de la gravedad
75  W_L=(m_carroy+m_guias+m_tornilloy+...
76      m_rod_tuercas+m_encodery+m_motory+m_carga)*g;  %peso de la carga
77  W_C=(m_carrox1+m_carrox2)*g;                       %peso del carro
78  W_T = W_L+W_C;                                     %peso total
79
80  %******************Calculo de la Inercia Total*********************
81
82  J_Lin=(1/(eta*N_tornillo^2))*(W_T/g);       %inercia de entrada
83                                              %de la carga
84
85  J_ref_trans = J_tornillo+J_Lin;             %Inercia reflejada por
86                                              %la transmision
87
88  J_Total=J_motor+J_encoder+J_cople+J_ref_trans;   %Inercia Total
89
90  %****************Calculo de las fuerzas externas*******************
91
92  F_f= f*W_T*cos(0);       %Fuerza de friccion (la inclinacion
93                           %respecto a lahorizontal es cero).
94  F_g=0;                   %Accion de la gravedad si es que se ...
       levanta                          %a la carga
95  F_p=0;                   %Fuerzas externas provocadas por procesos
96                           %del sistema
97  F_ext=F_g+F_f+F_p;       %Suma total de la accion de fuerzas ...
       exteriores
98
99  %***********Calculo del par necesario para el movimiento***********
100
101 T_load_in= F_ext/(eta*N_tornillo);   %par de la carga hacia el eje
102 v_deseada = 500;                     %velocidad deseada en rpm
103 L_sistema=0.8;                      %Longitud total del sistema ...
                                        en metros
104 v_m=v_deseada*(1/60)*(.008/1);      %velocidad deseada en m/s
105 a = 133.33e-3;                      %aceleracion en m/s^2
```

```
106 t_a=1;                                  %tiempo de aceleracion en ...
                                              segundos
107 t_d = t_a;                              %tiempo de desaceleracion en ...
                                           segundos
108 t_m = (L_sistema/v_m)-t_a;              %tiempo de movimiento estable
109                                         %en segundos
110 t_dw=0;                                 %tiempo de reposo en segundos
111 t_total = t_d+t_m+t_dw+t_a;             %tiempo total en segundos
112 w_m = v_deseada*(1/60)*(2*pi/1);        %velocidad deseada en rad/s
113 a_angular=w_m / t_a;                    %aceleracion angular en rad/s^2
114
115 T_C = (W_T*f_c*d_c)/2;                  %par necesario para vencer ...
       la                                    friccion del collarin
116 T_R= ((W_T*(D_tornillo-(p/2)))/2)* ( (l+pi*f*(D_tornillo-(p/2)))...
117     / (pi*(D_tornillo-(p/2))-f*l)) + T_C;   %par para mover la carga
118
119 %par necesario para la aceleraci n de la carga
120 T_aceleracion = J_Total*a_angular + T_load_in+T_C;
121
122 T_arranque = T_load_in;                  %par necesario para el arranque
123
124 %par desarrollado en la desaceleracion
125 T_desaceleracion = T_load_in - J_Total*a_angular;
126
127 T_dw=0;                 %par desarrollado en el movimiento de "reposo"
128 %par continuo necesario parael movimiento (RMS)
129 T_RMS = sqrt( ( ( (T_aceleracion^2)*t_a+(T_arranque^2)*t_m+...
130     (T_desaceleracion^2)*t_d+(T_dw^2)*t_dw ) / ...
131     (t_a+t_m+t_d+t_dw) ) );
132
133 %***************Graficos de par y perfil de velocidad**************
134
135 % t=0:0.01:t_total;
136 % for i=1:length(t)
137 %     if(t(i)≤t_a)
138 %         velocidad(i)= (v_m/t_a)*t(i);
139 %     elseif((t(i)>t_a)&&(t(i)<(t_a+t_m)))
140 %         velocidad(i)=v_m;
141 %     elseif((t(i)≥(t_a+t_m))&&(t(i)≤t_total))
```

```
142 %         velocidad(i)= ...
        (((0-v_m)/(t_total-(t_a+t_m)))*t(i))+(13/15);
143 %     end
144 % end
145 %
146 % plot(t,velocidad);
147 % for i=1:length(t)
148 %     if(t(i)≤t_a)
149 %         T(i)= T_aceleracion;
150 %     elseif((t(i)>t_a)&&(t(i)<(t_a+t_m)))
151 %         T(i)=T_arranque;
152 %     elseif((t(i)≥(t_a+t_m))&&(t(i)≤t_total))
153 %         T(i)= T_desaceleracion*-1;
154 %     end
155 % end
156 %
157 % plot(t,T);
158
159 %**********************Calculo de Esfuerzos***********************
160 cortante = (16*T_aceleracion)/(pi*0.006^3);
161 sigma_y= (-1*4*W_T)/(pi*0.006^2);
162 sigma_B = (-2*0.38*W_T)/(pi*0.007*1*0.002);
163 sigma_x = (6*0.38*W_T)/(pi*0.006*1*0.002);
164
165 sigma_1 = sigma_x;
166 sigma_2 = sigma_y/2 + sqrt(( ( -sigma_y/2 )^2 + cortante^2 ));
167 sigma_3 = sigma_y/2 - sqrt(( ( -sigma_y/2 )^2 + cortante^2 ));
168 sigma_prima = sqrt( ( ...
        ((sigma_1-sigma_2)^2+(sigma_2-sigma_3)^2+(sigma_3-sigma_1)^2)/ ...
        2 ) );
169 cortante_maximo = (sigma_1-sigma_3)/2 ;
```

Anexo 2. Criterio de selección de motores.

1. Encuentre la velocidad de funcionamiento ω_m del motor a partir del perfil de movimiento deseado para la carga. Calcule el par pico (T_{pico}) y el par RMS (T_{RMS}) que el motor debe de proveer. En este punto, el motor a utilizar es desconocido. Por tanto, utilice $J_m = 0$ en los cálculos del par por ahora.

2. Seleccione el motor más pequeño posible que pueda proporcionar estos valores de par a la velocidad de funcionamiento ω_m desde un catálogo de motores. La información del motor probablemente sea proveída a partir de curvas par-velocidad en formato tabular. Acepte la selección del motor si a la velocidad ω_m $T_{pico} \leq T_{PR}$ y $T_{RMS} \leq T_{CR}$.

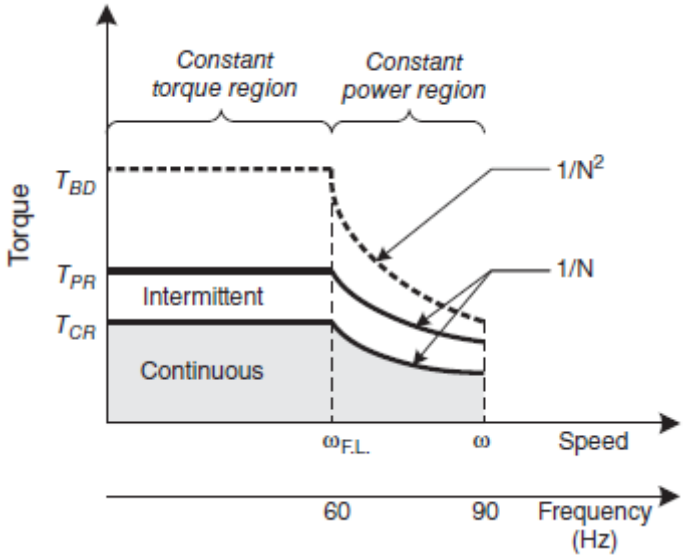


Figura 82: Curva par-velocidad genérica.

Se utilizará un driver eléctrico para hacer funcionar el motor. Por tanto, debe seleccionarse un driver adecuado que pueda proporcionar las corrientes necesarias para operar el motor en los pares pico y continuos. Si el driver tiene menos corriente de la que requieren los niveles de par, los pares T_{PR} y T_{CR} del motor deben reducirse (escalarse) linealmente, ya que el driver

aumentará la corriente y, por lo tanto, los pares del motor.

3. Recalcule T_{pico} y T_{RMS} con la inercia J_m del motor recién seleccionado.

4. Revise el criterio de selección del paso 2 para asegurarse de que el motor aún puede entregar el desempeño esperado con un cierto margen. En la práctica, un 30 % de margen para T_{RMS} y un 50 % de margen para T_{pico} son comunes. Es deseable que quede un poco de margen (capacidad de par extra) para realizar ajustes en caso de que las condiciones cambien durante la puesta en servicio de la máquina.

5. Calcule la razón de inercia J_R para asegurarse de que cumple con el criterio deseado. Típicamente, $J_R \leq 5$ es usado, pero esto también depende del desempeño deseado de la máquina. Si el criterio no se cumple, seleccione un motor con mayor inercia (un motor más grande) o reduzca la inercia de la carga y repita los paso 3 al 5.

Anexo 3. Diseño de la PCB.

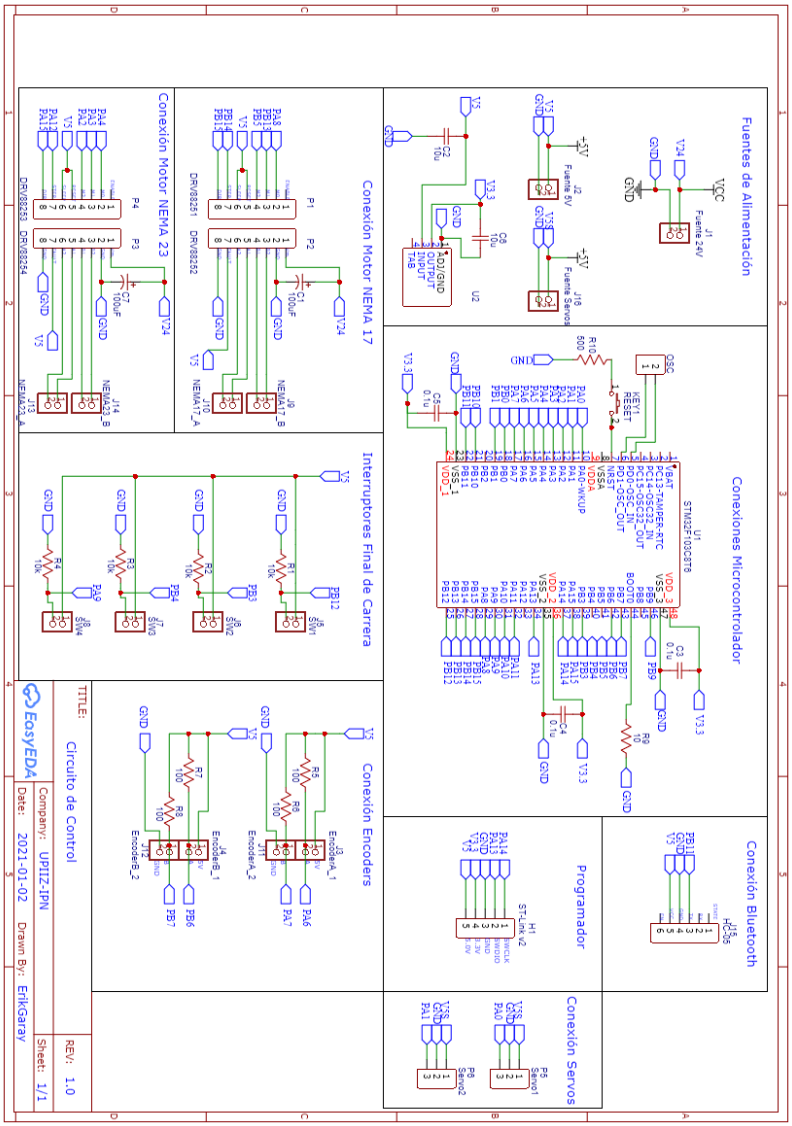


Figura 83: Esquemático utilizado para el diseño de la PCB.

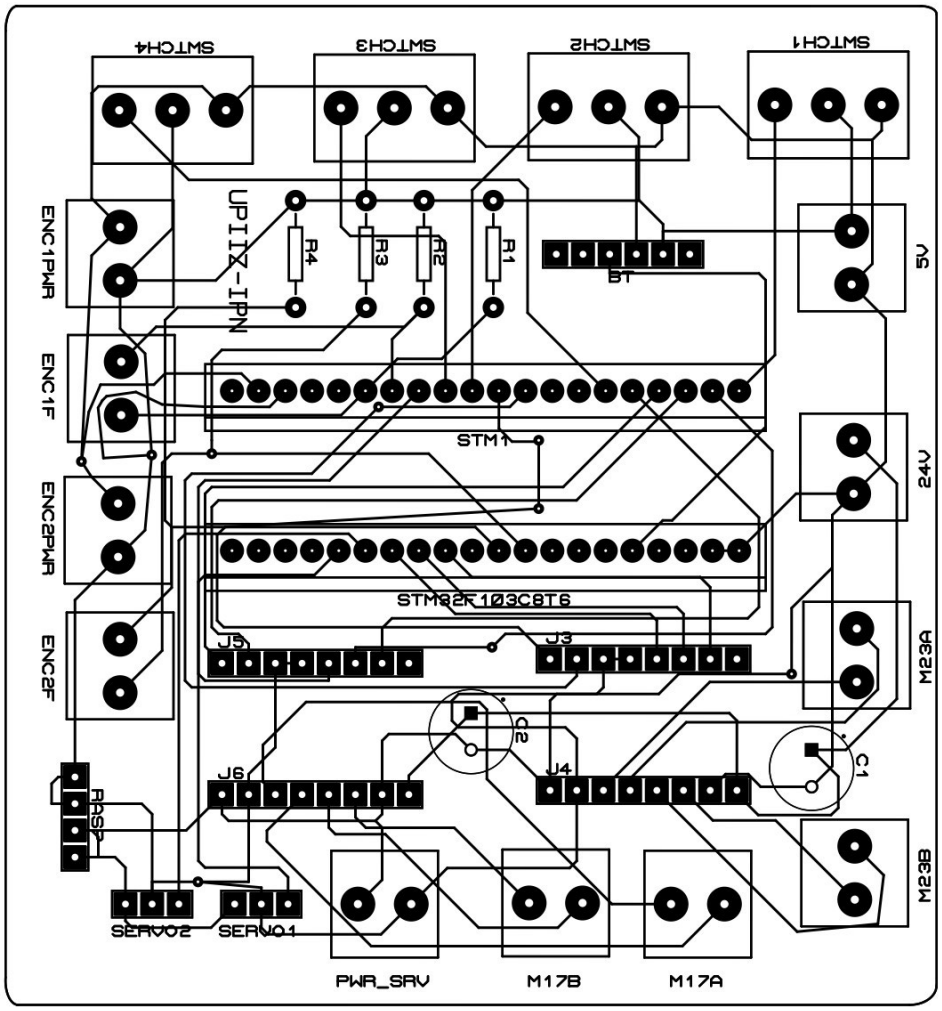


Figura 84: *Layout* y ruteo de las pistas.

Anexo 4. Convención de Denavit-Hartenberg.

En esta convención, cada transformación homogénea A_i es representada como un producto de cuatro transformaciones básicas

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \tag{26}$$

La ecuación anterior se desarrolla más a partir de la definición matemática de cada una de esas matrices. Por lo que la multiplicación de matrices que se realiza es

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\theta_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{27}$$

Resultando en una sola matriz luego de hacer la multiplicación de las cuatro transformaciones anteriores. La ecuación que contiene este resultado está dada por

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{28}$$

donde los cuatro parámetros θ_i, a_i, d_i, α_i son parámetros asociados con el eslabón i y la unión i. Las funciones seno y coseno son representadas por s y c, respectivamente, para ahorrar espacio.

Anexo 5. Diagrama de clases de la HMI.

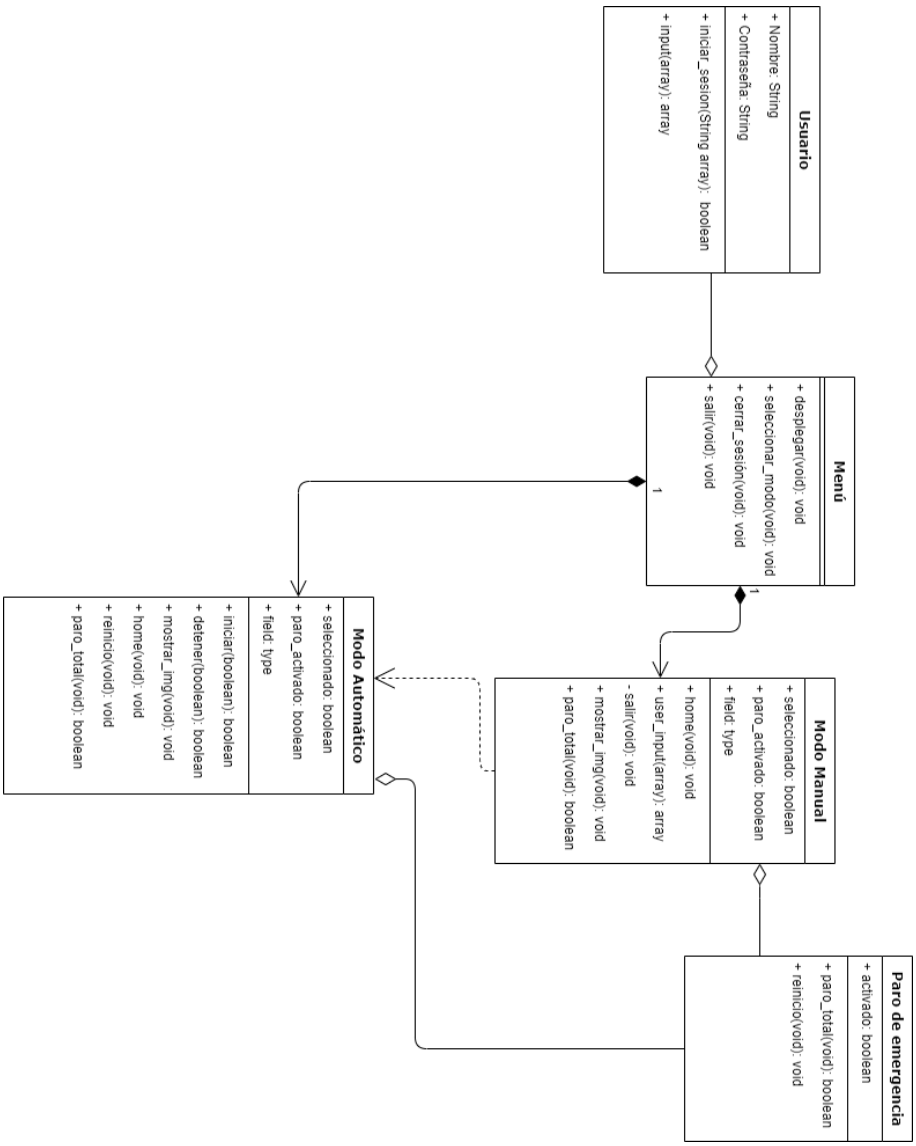


Figura 85: Diagrama de clases de la HMI.

Anexo 6. Diagramas de Bloques del Sistema de Control.

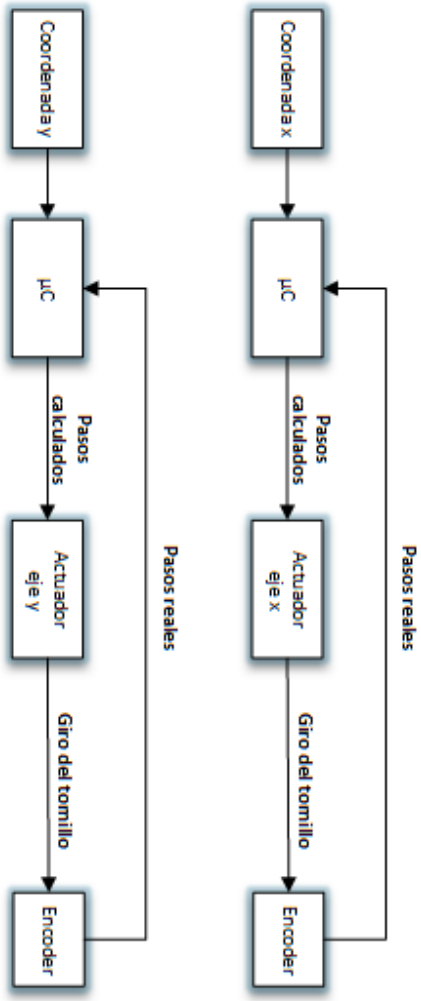


Figura 86: Diagrama de bloques de control a lazo abierto en el modo manual de operación.

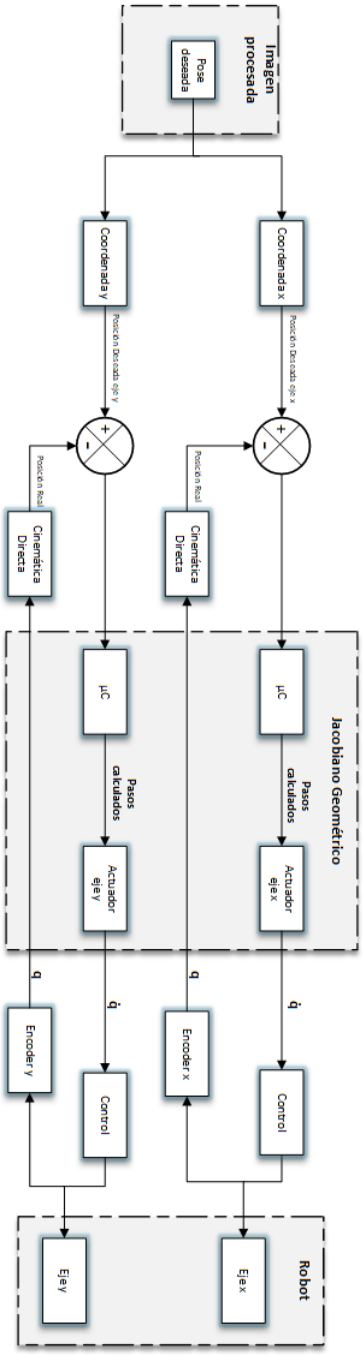
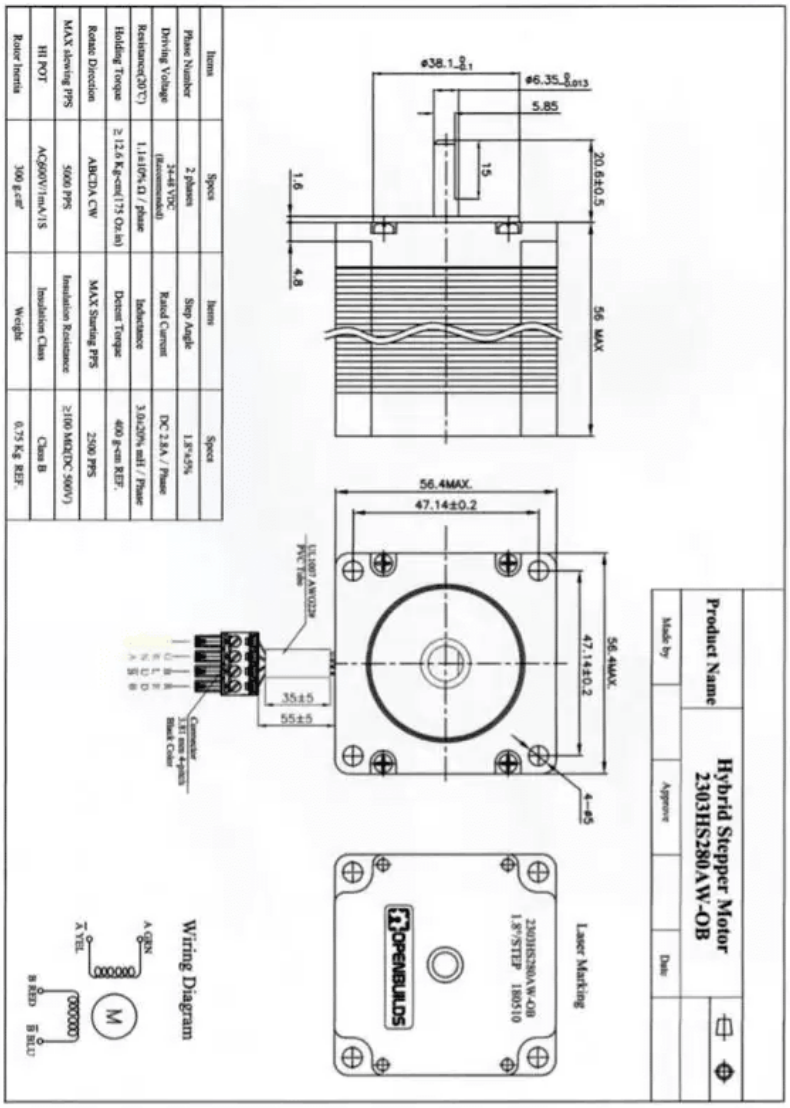


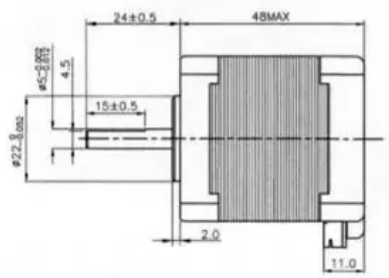
Figura 87: Diagrama de bloques de control a lazo cerrado en el modo automático de operación.

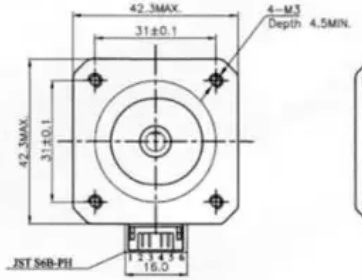
Anexo 7. Planos de piezas clave del ensamble mecánico.



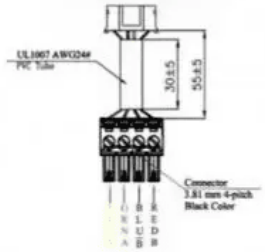
Items	Specs	Items	Specs
Phase Number	2 phases	Step Angle	1.8°±5%
Driving Voltage	24-48 VDC (Recommended)	Rated Current	DC 2.8A / Phase
Resistance(20℃)	1.1±10% Ω / phase	Inductance	3.0±20% mH / Phase
Holding Torque	≥ 12.6 Kg-cm(175 Oz in)	Detent Torque	400 g-cm REF.
Rotate Direction	ABCDA CW	MAX Starting PPS	2500 PPS
MAX slewing PPS	5000 PPS	Insulation Resistance	≥100 MΩ(DC 500V)
HI POT	AC600V/1mA/1S	Insulation Class	Class B
Rotor Inertia	300 g-cm²	Weight	0.75 Kg REF.

Product Name	Hybrid Stepper Motor 1704HS168A-OB	
Made by	Approve	Date

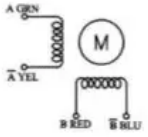




Laser Marking



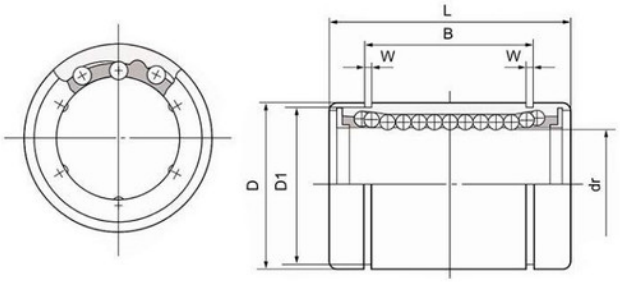
Wiring Diagram



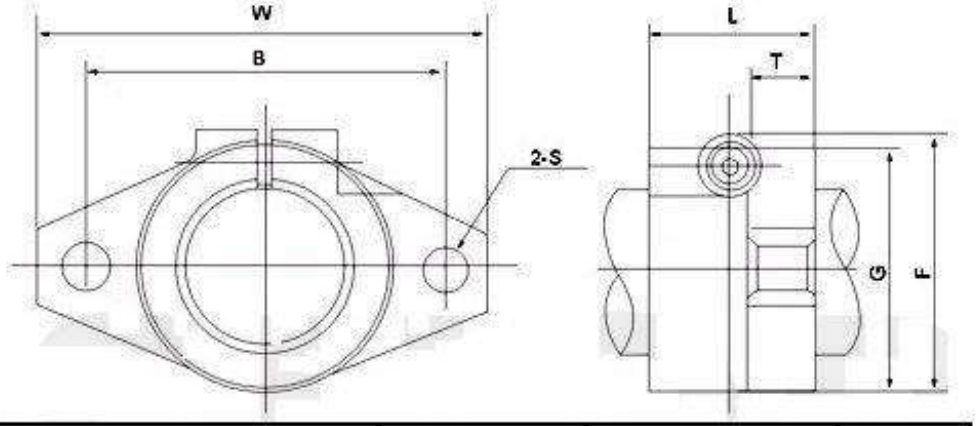
Items	Specs	Items	Specs
Phase Number	2 phases	Step Angle	1.8°±5%
Driving Voltage	12-24 VDC (Recommended)	Rated Current	DC 1.68 A / Phase
Resistance(20℃)	1.65±10% Ω / phase	Inductance	2.8±20% mH / Phase
Holding Torque	≥ 5.5 Kg-cm(77 Oz-in)	Detent Torque	0.2 Kg-cm REF.
Rotate Direction	ABCDA CW	MAX Starting PPS	2500 PPS
MAX slewing PPS	2500 PPS	Insulation Resistance	≥100 MΩ(DC 500V)
HI POT	AC600V/1mA/1S	Insulation Class	Class B
Rotor Inertia	68 g.cm²	Weight	0.35 Kg REF.

12mm
SHAFT BEARING
LM12UU

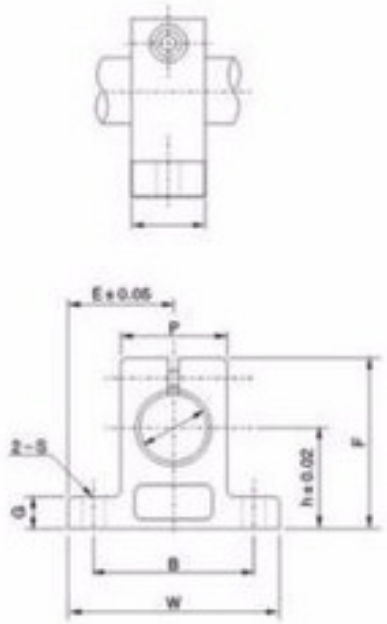
Nominal Shaft Diamter	Bearing No.	Ball Circuit	dr mm	D mm	L mm	B mm	W mm	D1 mm
8	LM8UU	4	8	15	24	17.5	1.1	14.3
10	LM10UU	4	10	19	29	22	1.3	18
12	LM12UU	4	12	21	30	23	1.3	20
16	LM16UU	5	16	28	37	26.5	1.6	27
20	LM20UU	5	20	32	42	30.5	1.6	30.5



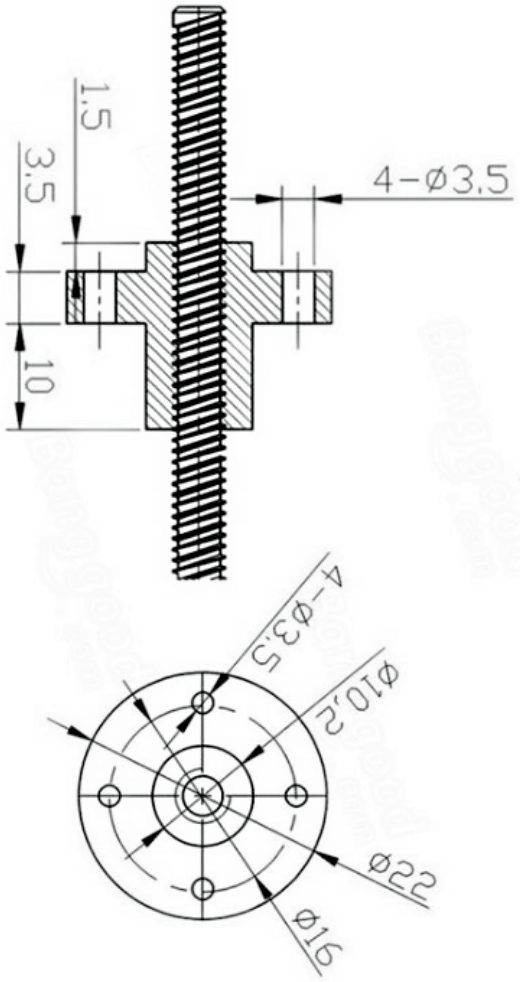
SHF12

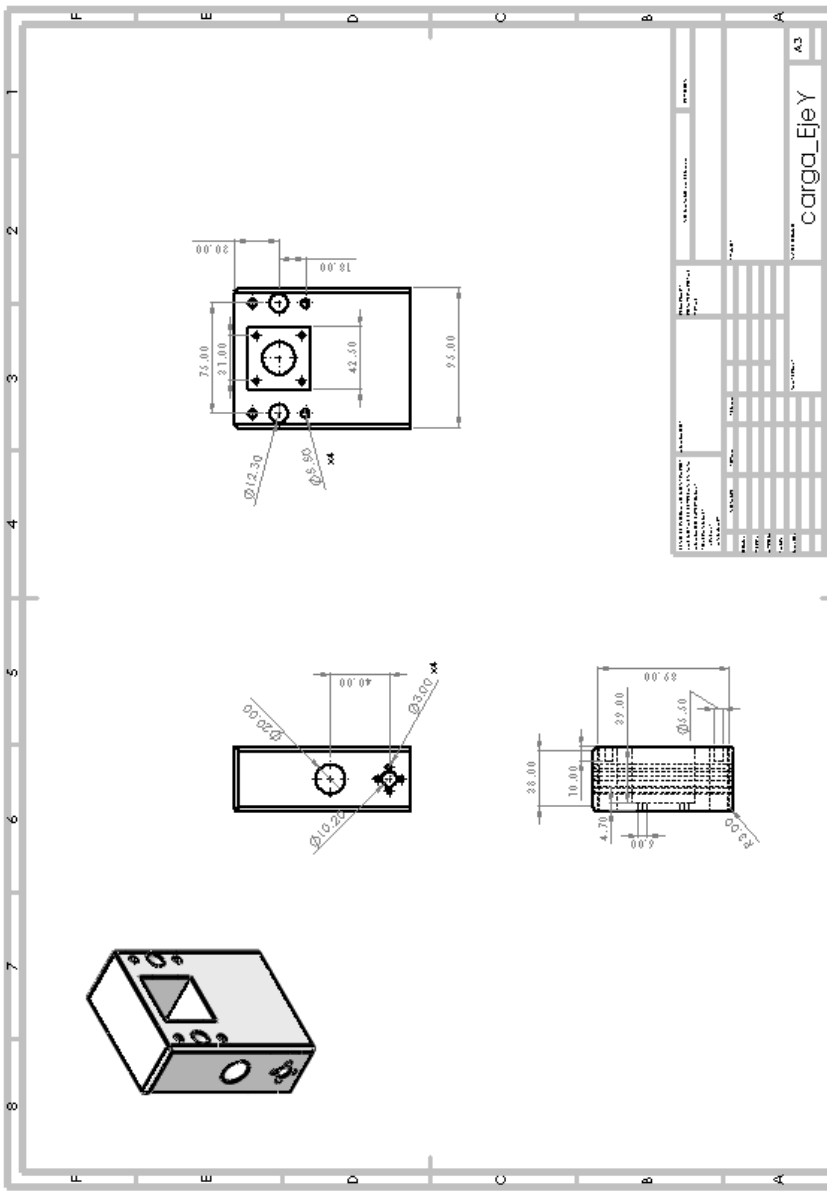


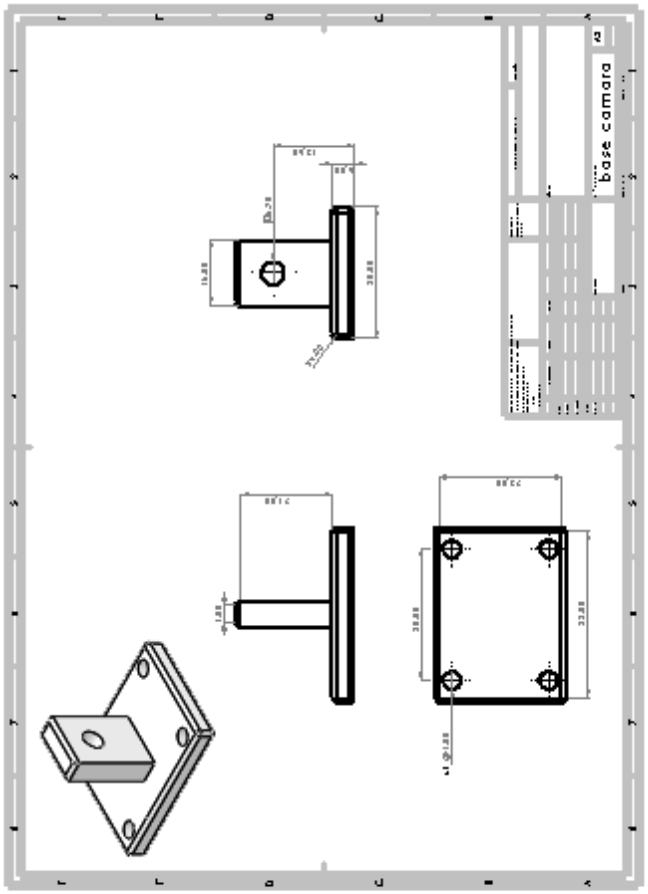
Model	Shaft Dia (mm)	Dimensions						Clamping bolt	Mounting Bolt (S)	Mass (g)
		W	L	T	F	G	B			
SHF12	12	47	13	7	28	25	36	M4	5.5 (M5)	20



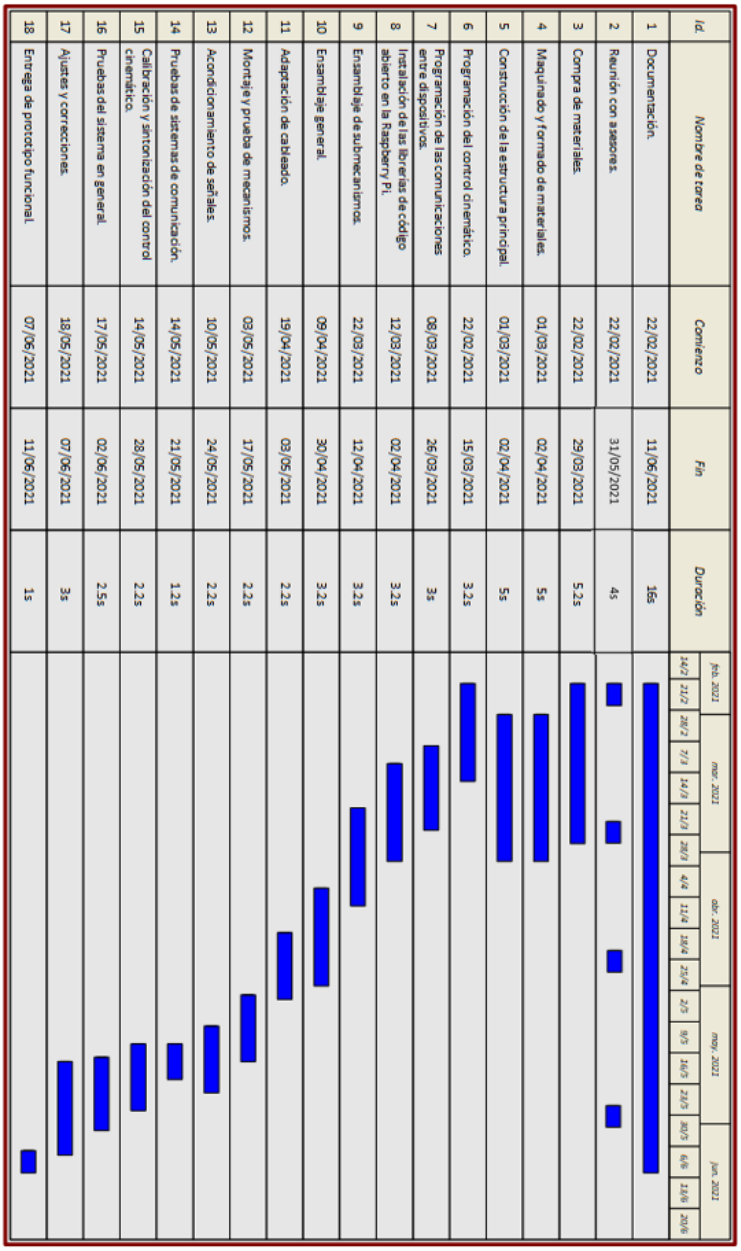
MODEL NO.	SHAFT DIAMETER	MAIN DIMENSIONS(mm)									DESIGNATION OF CLAMPING BOLT	DESIGNATION OF MOUNTING BOLT	WEIGHT (kg)
		h	E	W	L	F	G	P	B	S			
SK3	3	20	21	42	14	37.5	6	18	32	5.5	M4	M5	0.024
SK4	4	20	21	42	14	37.5	6	18	32	5.5	M4	M5	0.024
SK5	5	20	21	42	14	37.5	6	18	32	5.5	M4	M5	0.024
SK6	6	20	21	42	14	37.5	6	18	32	5.5	M4	M5	0.024
SK8	8	20	21	42	14	37.5	6	18	32	5.5	M4	M5	0.024
SK10	10	20	21	42	14	37.5	6	18	32	5.5	M4	M5	0.024
SK12	12	23	21	42	14	37.5	6	20	32	5.5	M4	M5	0.030
SK13	13	23	21	42	14	37.5	6	20	32	5.5	M4	M5	0.030
SK16	16	27	24	48	16	44	8	25	38	5.5	M5	M6	0.040
SK20	20	31	30	60	20	51	10	30	45	6.6	M6	M6	0.070
SK25	25	35	35	70	24	60	12	38	56	6.6	M6	M8	0.130
SK30	30	42	42	84	28	70	12	44	64	9	M8	M10	0.180
SK35	35	50	49	98	32	82	15	50	74	11	M8	M10	0.270
SK40	40	60	57	114	36	96	15	60	90	11	M10	M10	0.420
SK50	50	70	63	126	40	120	16	74	100	14	M12	M12	0.750
SK60	60	80	74	148	45	136	18	90	120	14	M12	M12	1.100


1.5
3.5
10
4-ø3.5
4-ø3.5
ø10.2
ø22
ø16


carga_Eje Y
A3


base camara

Anexo 8. Cronograma De Actividades Para Trabajo Terminal II



Id.	Nombre de tarea	Comienzo	Fin	Duración
1	Documentación.	22/02/2021	11/06/2021	16s
2	Reunión con asesores.	22/02/2021	31/05/2021	4s
3	Compra de materiales.	22/02/2021	29/03/2021	5.2s
4	Maquinado y formado de materiales.	01/03/2021	02/04/2021	5s
5	Construcción de la estructura principal.	01/03/2021	02/04/2021	5s
6	Programación del control cinemático.	22/02/2021	15/03/2021	3.2s
7	Programación de las comunicaciones entre dispositivos.	08/03/2021	26/03/2021	3s
8	Instalación de las librerías de código abierto en la Raspberry Pi.	12/03/2021	02/04/2021	3.2s
9	Ensamblaje de submecanismos.	22/03/2021	12/04/2021	3.2s
10	Ensamblaje general.	09/04/2021	30/04/2021	3.2s
11	Adaptación de cableado.	19/04/2021	03/05/2021	2.2s
12	Montaje y prueba de mecanismos.	03/05/2021	17/05/2021	2.2s
13	Acondicionamiento de señales.	10/05/2021	24/05/2021	2.2s
14	Pruebas de sistemas de comunicación.	14/05/2021	21/05/2021	1.2s
15	Calibración y sintonización del control cinemático.	14/05/2021	28/05/2021	2.2s
16	Pruebas del sistema en general.	17/05/2021	02/06/2021	2.5s
17	Ajustes y correcciones.	18/05/2021	07/06/2021	3s
18	Entrega de prototipo funcional.	07/06/2021	11/06/2021	1s

Anexo 9. Código utilizado para programar el control del Cartesiano con el μC STM32F103C8T6

Archivo de cabecera encoder1.h:

```
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/gpio.h>
#include <libopencm3/stm32/timer.h>
#include <libopencm3/cm3/nvic.h>
#define PPR_ENCODER 2400

void config_encoder(void);
int contar_pasos(void);
```

Archivo encoder1.c:

```
#include "encoder1.h"
//PIN     ENCODER
// PB7 -> A
// PB6 -> B

void config_encoder(void){
    // SYSCLK a 72Mhz
    //rcc_clock_setup_in_hse_8mhz_out_72mhz();

    // Se habilita el clock del TIM4.
    rcc_periph_clock_enable(RCC_TIM4);

    /*
        El encoder de cuadratura genera 2400 pulsos por vuelta.
    */
    timer_set_period(TIM4, PPR_ENCODER-1);

    /*
```

```
19          Modo encoder 3: Sube y baja el contador por cualquiera ...
                de los pulsos en
20          TI1FP1 o TI2FP2.
21      */
22      timer_slave_set_mode(TIM4, TIM_SMCR_SMS_EM3);
23
24      // Configuraci n de los Canales de Entrada TI1 y TI2:
25      timer_ic_set_input(TIM4, TIM_IC1, TIM_IC_IN_TI1);
26      timer_ic_set_input(TIM4, TIM_IC2, TIM_IC_IN_TI2);
27
28      // Se habilita la interrupci n por overflow
29      timer_enable_irq(TIM4, TIM_DIER_UIE);
30
31      // Se habilita el contador:
32      timer_enable_counter(TIM4);
33
34      //Se habilita la interrupci n desde el NVIC
35      nvic_enable_irq(NVIC_TIM4_IRQ);
36  }
37
38  int contar_pasos(void){
39
40      uint16_t pos = 0;
41      pos = timer_get_counter(TIM4);
42      return pos;
43
44  }
```

Archivo de cabecera encoder2.h:

```
1  #include <libopencm3/stm32/rcc.h>
2  #include <libopencm3/stm32/gpio.h>
3  #include <libopencm3/stm32/timer.h>
4  #include <libopencm3/cm3/nvic.h>
5  #define PPR_ENCODER2 2400
6
7  void config_encoder2(void);
8  int contar_pasos2(void);
```

Archivo encoder2.c:

```
#include "encoder2.h"
//PIN     ENCODER
// PA7 -> A
// PA6 -> B

void config_encoder2(void){
    // SYSCLK a 72Mhz
    //rcc_clock_setup_in_hse_8mhz_out_72mhz();

    // Se habilita el clock del TIM3.
    rcc_periph_clock_enable(RCC_TIM3);

    /*
        El encoder de cuadratura genera 2400 pulsos por vuelta.
    */
    timer_set_period(TIM3, PPR_ENCODER2-1); //Empieza en 0 el ...
       contador

    /*
        Modo encoder 3: Sube y baja el contador por cualquiera ...
           de los pulsos en
        TI1FP1 o TI2FP2.
    */
    timer_slave_set_mode(TIM3, TIM_SMCR_SMS_EM3);

    // Configuraci n de los Canales de Entrada TI1 y TI2:
    timer_ic_set_input(TIM3, TIM_IC1, TIM_IC_IN_TI1);
    timer_ic_set_input(TIM3, TIM_IC2, TIM_IC_IN_TI2);

    // Se habilita la interrupci n por overflow
    timer_enable_irq(TIM3, TIM_DIER_UIE);

    // Se habilita el contador:
    timer_enable_counter(TIM3);

    //Se habilita la interrupci n desde el NVIC
```

```
35      nvic_enable_irq(NVIC_TIM3_IRQ);
36  }
37
38  int contar_pasos2(void){
39
40      uint16_t pos = 0;
41      pos = timer_get_counter(TIM3);
42      return pos;
43
44  }
```

Archivo de cabecera uart_recv.h:

```
1  #include <libopencm3/stm32/gpio.h>
2  #include <libopencm3/stm32/rcc.h>
3  #include <libopencm3/stm32/usart.h>
4  #include <stdio.h>
5  void config_uart_recv(void);
6  char uart_getc(void);
7  int longitud_cadena(char cadena[]);
```

Archivo uart_recv.c:

```
1  #include <libopencm3/stm32/gpio.h>
2  #include "uart_recv.h"
3
4  void config_uart_recv(void){
5
6      rcc_periph_clock_enable(RCC_GPIOB);
7      rcc_periph_clock_enable(RCC_USART3);
8      gpio_set_mode(GPIOB,
9                GPIO_MODE_INPUT,
10               GPIO_CNF_INPUT_FLOAT,
11               GPIO10);
12     usart_set_baudrate(USART3,38400);
13     usart_set_databits(USART3,8);
14     usart_set_stopbits(USART3,USART_STOPBITS_1);
15     usart_set_mode(USART3, USART_MODE_RX);
```

```
16      usart_set_parity(USART3, USART_PARITY_NONE);
17      usart_set_flow_control(USART3, USART_FLOWCONTROL_NONE);
18      usart_enable(USART3);
19  }
20
21  char uart_getc(void){
22
23      char ch='\0';
24
25      ch= usart_recv_blocking(USART3);
26      return ch;
27  }
28
29  int longitud_cadena(char cadena[]){
30
31      int contador = 0;
32      while (cadena[contador] != '\0'){
33              contador++;
34      }
35      return contador;
36
37  }
```

Archivo de cabecera delay_us.h:

```
1  #include <libopencm3/stm32/rcc.h>
2  #include <libopencm3/stm32/gpio.h>
3  #include <libopencm3/cm3/systick.h>
4  #include <libopencm3/cm3/nvic.h>
5
6  void config_systick(void);
7  void delay_us(uint32_t us);
```

Archivo delay_us.c:

```
/*********************************
 * Instituto Politécnico Nacional *
 *            UPIIZ               *
 *     código por: Erik Garay      *
 *********************************/

 //Programa que hace un delay en microsegundos
 //parecido al de Arduino.


#include "delay_us.h"

volatile uint32_t micros;   //ticks de la última cuenta (cada 1 us)

void config_systick(void){

    systick_set_clocksource(STK_CSR_CLKSOURCE_AHB_DIV8);
    // Se carga el valor a cargar cuando el SysTick llega a 0
    //   (downflow):
        systick_set_reload(8); // 9M / 9 = 1M => T = us
        systick_interrupt_enable();
        systick_counter_enable();
}

void delay_us(uint32_t us){
    uint32_t tm = micros+us;
    while(micros<tm);
}

void sys_tick_handler(void){
    micros++;
}
```

Archivo de cabecera nema23.h:

```
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/gpio.h>

//*** Se definen las conexiones del STM32F103C8T6 con ***
//***                  el driver DRV8825              ***

/*  ********************************
    DRV8825        Bluepill
      DIR          A15
      STP          A12
      M0           A4
      M1           A3
      M2           A2
    ******************************** */

#define DIR_1   GPIO15
#define STP_1   GPIO12
#define M0_1    GPIO2
#define M1_1    GPIO3
#define M2_1    GPIO4
#define VELOCIDAD_MANUAL_Y 300

void config_pines(void);
void micropasos(void);
void mover_nema23_manual(int dir,int pasos);
void detener_nema23_manual(void);
```

Archivo nema23.c:

```
#include "nema23.h"
#include "delay_us.h"
#include "interrupciones.h"

volatile bool interr_y1, interr_y2, interr_x1, interr_x2;

void config_pines(void){
```

```
8
9       rcc_periph_clock_enable(RCC_GPIOA);
10      gpio_set_mode(GPIOA,
11                GPIO_MODE_OUTPUT_2_MHZ,
12                GPIO_CNF_OUTPUT_PUSHPULL,
13                DIR_1 | STP_1 | M0_1 | M1_1 | M2_1);
14  }
15
16  void micropasos(void){
17
18      gpio_clear(GPIOA,M0_1);
19      gpio_set(GPIOA,M1_1);
20      gpio_clear(GPIOA,M2_1);
21
22  }
23
24  void detener_nema23_manual(void){
25
26      gpio_clear(GPIOA, DIR_1);
27      gpio_clear(GPIOA, STP_1);
28
29  }
30
31  void mover_nema23_manual(int dir, int pasos){
32
33      int i=0;
34      switch(dir){
35
36          case 1:
37              gpio_set(GPIOA, DIR_1);
38
39              for(i=0; i<pasos; i++){
40                  if((interr_y1 == false) && (interr_y2 == false)){
41                      gpio_clear(GPIOA,STP_1);
42                      gpio_set(GPIOA,STP_1);
43                      delay_us(VELOCIDAD_MANUAL_Y);
44                  }
45
46                  else{
```

```
47                      detener_nema23_manual();
48
49                  }
50
51              }
52
53          break;
54
55          case 2:
56              gpio_clear(GPIOA, DIR_1);
57
58              for(i=0; i<pasos; i++){
59
60                  if((interr_y1 == false) && (interr_y2 == false)){
61                      gpio_clear(GPIOA,STP_1);
62                      gpio_set(GPIOA,STP_1);
63                      delay_us(VELOCIDAD_MANUAL_Y);
64                  }
65
66                  else{
67                      detener_nema23_manual();
68
69                  }
70
71              }
72
73          break;
74
75          default:
76              gpio_clear(GPIOA, DIR_1);
77              gpio_clear(GPIOA, STP_1);
78
79      }
80
81
82
83  }
84
85
```

```
86
87 void exti15_10_isr()
88 {
89     exti_reset_request(EXTI12);
90     interr_y1 = true;
91 }
92
93 void exti3_isr()
94 {
95     exti_reset_request(EXTI3);
96     interr_y2 = true;
97 }
```

Archivo de cabecera nema17.h:

```
1  #include <libopencm3/stm32/rcc.h>
2  #include <libopencm3/stm32/gpio.h>
3
4  //*** Se definen las conexiones del STM32F103C8T6 con ***
5  //***                  el driver DRV8825              ***
6
7  /*  ********************************
8      DRV8825        Bluepill
9        DIR          B15
10       STP          B14
11       M0           A8
12       M1           B13
13       M2           B5
14     ******************************** */
15
16 #define DIR_2   GPIO15
17 #define STP_2   GPIO14
18 #define M0_2    GPIO8
19 #define M1_2    GPIO13
20 #define M2_2    GPIO5
21 #define VELOCIDAD_MANUAL_X 300
22
23 void config_pines_17(void);
```

```
24 void micropasos_17(void);
25 void mover_nema17_manual(int dir,int pasos);
26 void detener_nema17_manual(void);
```

Archivo nema17.c:

```
1  #include "nema17.h"
2  #include "delay_us.h"
3  #include "interrupciones.h"
4  void config_pines_17(void){
5
6      rcc_periph_clock_enable(RCC_GPIOA);
7      gpio_set_mode(GPIOB,
8                GPIO_MODE_OUTPUT_2_MHZ,
9                GPIO_CNF_OUTPUT_PUSHPULL,
10               M0_2);
11
12     rcc_periph_clock_enable(RCC_GPIOB);
13     gpio_set_mode(GPIOB,
14               GPIO_MODE_OUTPUT_2_MHZ,
15               GPIO_CNF_OUTPUT_PUSHPULL,
16               DIR_2 | STP_2 | M1_2 | M2_2);
17 }
18
19 void micropasos_17(void){
20
21     gpio_clear(GPIOA,M0_2);
22     gpio_set(GPIOB,M1_2);
23     gpio_clear(GPIOB,M2_2);
24
25 }
26
27 void mover_nema17_manual(int dir, int pasos){
28
29     int i=0;
30     switch(dir){
31
32         case 1:
```

```
33              gpio_set(GPIOB, DIR_2);
34
35              for(i=0; i<pasos; i++){
36
37                  if((interr_y3 == false) && (interr_y4 == false)){
38                      gpio_clear(GPIOB,STP_2);
39                      gpio_set(GPIOB,STP_2);
40                      delay_us(VELOCIDAD_MANUAL_X);
41                  }
42              }
43
44          break;
45
46          case 2:
47              gpio_clear(GPIOB, DIR_2);
48
49                  for(i=0; i<pasos; i++){
50
51                  if((interr_y3 == false) && (interr_y4 == false)){
52                      gpio_clear(GPIOB,STP_2);
53                      gpio_set(GPIOB,STP_2);
54                      delay_us(VELOCIDAD_MANUAL_X);
55                  }
56              }
57
58          break;
59
60          default:
61              gpio_clear(GPIOB, DIR_2);
62              gpio_clear(GPIOB, STP_2);
63
64      }
65
66  }
67
68  void detener_nema17_manual(void){
69
70      gpio_clear(GPIOB, DIR_2);
71      gpio_clear(GPIOB, STP_2);
```

```
72
73 }
74
75 void exti4_isr()
76 {
77     exti_reset_request(EXTI4);
78     delay_ms(200);
79     interr_y3 = true;
80 }
81
82
83 void exti9_5_isr()
84 {
85     exti_reset_request(EXTI9);
86     interr_y4=true;
87 }
```

Archivo de cabecera interrupciones.h:

```
1 #include <libopencm3/stm32/rcc.h>
2 #include <libopencm3/stm32/exti.h>
3 #include <libopencm3/cm3/nvic.h>
4
5 void config_interrupciones(void);
```

Archivo interrupciones.c:

```
1 #include <libopencm3/stm32/rcc.h>
2 #include <libopencm3/stm32/gpio.h>
3 #include "interrupciones.h"
4
5 void config_interrupciones(void){
6
7     void config_interrupciones(void){
8
9     rcc_periph_clock_enable(RCC_AFIO);
10    gpio_set_mode(GPIOB, GPIO_MODE_INPUT, GPIO_CNF_INPUT_FLOAT, ...
          GPIO12);
```

```
11      gpio_set_mode(GPIOB, GPIO_MODE_INPUT, GPIO_CNF_INPUT_FLOAT, ...
            GPIO3);
12      gpio_set_mode(GPIOB, GPIO_MODE_INPUT, GPIO_CNF_INPUT_FLOAT, ...
            GPIO4);
13      gpio_set_mode(GPIOA, GPIO_MODE_INPUT, GPIO_CNF_INPUT_FLOAT, ...
            GPIO9);
14
15      //Habilitaci n de la interrupci n en el NVIC:
16      nvic_enable_irq(NVIC_EXTI15_10_IRQ);
17
18      exti_select_source(EXTI12,GPIOB);
19      exti_set_trigger(EXTI12, EXTI_TRIGGER_BOTH);
20      exti_enable_request(EXTI12);
21
22
23      nvic_enable_irq(NVIC_EXTI3_IRQ);
24
25      exti_select_source(EXTI3,GPIOB);
26      exti_set_trigger(EXTI3, EXTI_TRIGGER_RISING);
27      exti_enable_request(EXTI3);
28
29      nvic_enable_irq(NVIC_EXTI9_5_IRQ);
30
31      exti_select_source(EXTI9,GPIOA);
32      exti_set_trigger(EXTI9, EXTI_TRIGGER_BOTH);
33      exti_enable_request(EXTI9);
34
35      nvic_enable_irq(NVIC_EXTI4_IRQ);
36
37      exti_select_source(EXTI4,GPIOB);
38      exti_set_trigger(EXTI4, EXTI_TRIGGER_RISING);
39      exti_enable_request(EXTI4);
40
41  }
```

Archivo pwm.h:

```
1  #include <libopencm3/stm32/rcc.h>
```

```
2 #include <libopencm3/stm32/gpio.h>
3 #include <libopencm3/stm32/timer.h>
4
5 void configurar_pwm(void);
```

Archivo pwm.c:

```
1  #include "pwm.h"
2
3
4
5  void configurar_pwm(void){
6
7                  //Pin asociado al OC1
8      //Se configura el timer como PWM alineado al flanco.
9      rcc_periph_clock_enable(RCC_TIM2);
10     rcc_periph_clock_enable(RCC_AFIO);
11
12     rcc_periph_clock_enable(RCC_GPIOA);
13
14     gpio_primary_remap(
15     AFIO_MAPR_SWJ_CFG_JTAG_OFF_SW_OFF,
16     AFIO_MAPR_TIM2_REMAP_NO_REMAP);
17
18     //PA0 -> TIM2_CH1
19     gpio_set_mode(GPIOA,                 //Puerto correspondiente
20                GPIO_MODE_OUTPUT_50_MHZ,      //M xima velocidad ...
                      de switcheo
21                GPIO_CNF_OUTPUT_ALTFN_PUSHPULL,   //Funci n ...
                      alternativa
22                GPIO0);
23         //PA1 -> TIM2_CH2
24     gpio_set_mode(GPIOA,                 //Puerto correspondiente
25                GPIO_MODE_OUTPUT_50_MHZ,      //M xima velocidad ...
                      de switcheo
26                GPIO_CNF_OUTPUT_ALTFN_PUSHPULL,   //Funci n ...
                      alternativa
27                GPIO1);
```

```
28      timer_disable_counter(TIM2);
29      timer_set_mode(
30          TIM2,           //Timer general 2
31          TIM_CR1_CKD_CK_INT, //CLock interno como fuente
32          TIM_CR1_CMS_EDGE,   //modo alineado al flanco
33          TIM_CR1_DIR_UP      //Indistinto, esto es ignorado
34          );
35
36      /*  Seteamos la cuenta del Timer
37
38      */
39
40      timer_set_prescaler(TIM2,72);
41      timer_enable_preload(TIM2);
42      timer_continuous_mode(TIM2);
43      timer_set_period(TIM2, 20000);  // 72M/2/10000 = 3,6khz
44
45      timer_disable_oc_output(TIM2, TIM_OC1);
46      //Se configuran las salidas del timer
47      timer_set_oc_mode(TIM2, TIM_OC1, TIM_OCM_PWM1);
48      timer_enable_oc_output(TIM2, TIM_OC1);
49      //timer_enable_counter(TIM2);
50
51      timer_disable_oc_output(TIM2, TIM_OC2);
52      //Se configuran las salidas del timer
53      timer_set_oc_mode(TIM2, TIM_OC2, TIM_OCM_PWM1);
54      timer_enable_oc_output(TIM2, TIM_OC2);
55      timer_enable_counter(TIM2);
56  }
```

Archivo main.c:

```
1  #include <libopencm3/stm32/rcc.h>
2  #include <libopencm3/stm32/gpio.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include "delay_us.h"
```

```
7  #include "nema23.h"
8  #include "nema17.h"
9  #include "uart_recv.h"
10 #include "uart_send.h"
11 #include "encoder1.h"
12 #include "encoder2.h"
13 #include "interrupciones.h"
14
15 #define AVANCE 8    //El avance de los tornillos es de 8mm/rev
16 #define PPR_MOT 800 //Pasos por vuelta de los motores; config. ...
       1/4 de paso
17 #define PPR_RATIO 3    //Raz n de proporcionalidad entre la ...
       resoluci n del motor
18             //y del encoder
19 void setup(void);
20
21
22 volatile int ovrflw_cnt=0;
23 volatile int ovrflw_cnt2=0;
24
25 void setup(void){
26     rcc_clock_setup_in_hse_8mhz_out_72mhz(); //SYSCLK a 72MHz
27     config_pines();
28     config_pines_17();
29     config_systick();
30     //config_interrupciones();
31     config_uart();
32     config_uart_recv();
33     config_encoder();
34     config_encoder2();
35     micropasos();
36     micropasos_17();
37
38 }
39
40
41 int main(void) {
42
43     // Se configuran los perif ricos
```

```
44      setup();
45
46      //********* DECLARACION DE VARIABLES ***************
47
48      // Variables necesarias para el BT
49      char dato;
50      char buffer[50];
51      const char s[2] = ",";
52      char *t1,*t2,*t3,*t4,*t5,*t6,*t7;
53      int i=0;
54      int v1d = 0, v2d = 0;
55      int distd_23 = 0, distd_17 = 0;     //distancias a las que ...
           se desean mover los carros
56      int v1=0, dir_23=0;
57      int v2=0, dir_17=0;
58      int pwm_servo1 = 0, pwm_servo2=0;
59      //int n=0;
60
61
62
63      // Aqu  se declaran las variables que tienen que ver con el ...
           Motor NEMA23
64
65      int dist_mm_23 = 0;         //se convierte la distancia a ...
           mil metros
66      float vueltas_deseadas_23 = 0;  //se calculan las vueltas ...
           necesarias para alcanzar
67                          //la distancia deseada
68      int pasos_23 = 0;           //se calculan los pasos que debe ...
           dar el motor
69
70      //************************************************
71
72      // Aqu  se declaran las variables que tienen que ver con el ...
           Motor NEMA17
73
74      int dist_mm_17 = 0;         //se convierte la distancia a ...
           mil metros
```

```
75      float vueltas_deseadas_17 = 0;  //se calculan las vueltas ...
            necesarias para alcanzar
76                          //la distancia deseada
77      int pasos_17 = 0;           //se calculan los pasos que debe ...
            dar el motor
78
79      //*********************
80
81      // Aqu  se declaran las variables que tienen que ver con el ...
            encoder 1
82      uint16_t pos_inicial=0, pos_final;
83      int pasos_antes_primer_ovfl=0, pasos_antes_ultimo_ovflw=0;
84      int pasosd_encoder, pasos_leidos, error_pos;
85      //*************************************************
86
87      // Aqu  se declaran las variables que tienen que ver con el ...
            encoder 2
88      uint16_t pos_inicial2=0, pos_final2;
89      int pasos_antes_primer_ovfl2=0, pasos_antes_ultimo_ovflw2=0;
90      int pasosd_encoder2, pasos_leidos2, error_pos2;
91      //*************************************************
92
93
94      while(true){
95
96          //Primero se recibe por BT el mensaje que contiene
97          //el modo en que debe operar la m quina.
98
99          //COMIENZA LA RECEPCI N DE DATOS
100         dato= uart_getc();
101
102         while(dato!='\n'){
103
104             buffer[i] = dato;
105             i++;
106             dato=uart_getc();
107         }
108
109         buffer[i]='\0';
```

```
110         i=0;
111
112         //Se separa el mensaje recibido para obtener la ...
                informaci n//
113         //token = strtok(buffer,s);
114         t1 = strtok(buffer,s);
115         t2 = strtok(NULL,s);
116         t3 = strtok(NULL,s);
117         t4 = strtok(NULL,s);
118         t5 = strtok(NULL,s);
119         t6 = strtok(NULL,s);
120         t7 = strtok(NULL,s);
121
122
123         while(t1[i] != '\0'){
124             uart_putc(t1[i]);
125             i++;
126         }
127         uart_putc('\r');
128         uart_putc('\n');
129         uart_putc('\0');
130         t1[0]='\0';
131         i=0;
132
133         //Se obtiene la distancia deseada del Eje y
134         while(t2[i] != '\0'){
135             uart_putc(t2[i]);
136             i++;
137         }
138
139         uart_putc('\r');
140         uart_putc('\n');
141         uart_putc('\0');
142
143         //Como la distancia es en cm, solo son dos n meros
144         //los que se reciben. Se convierten los char a int
145
146         if(i==2){
147             distd_23 = ((t2[0]-'0')*10) + (t2[1]-'0');
```

```
148             t2[0]='\0';
149             i=0;
150         }
151 
152         else if(i==1){
153             distd_23 = (t2[0]-'0');
154             t2[0]='\0';
155             i=0;
156         }
157 
158         //Se obtiene la direcci n de movimiento en el eje y
159 
160         while(t3[i] != '\0'){
161             uart_putc(t3[i]);
162             i++;
163         }
164         uart_putc('\r');
165         uart_putc('\n');
166         uart_putc('\0');
167         dir_23 = t3[0]-'0';
168         t3[0] = '\0';
169         i=0;
170 
171 
172 
173         //Se obtiene la distancia deseada del Eje x
174 
175         while(t4[i] != '\0'){
176             uart_putc(t4[i]);
177             i++;
178         }
179         uart_putc('\r');
180         uart_putc('\n');
181         uart_putc('\0');
182 
183         if(i==2){
184             distd_17 = ((t4[0]-'0')*10) + (t4[1]-'0');
185             t4[0]='\0';
186             i=0;
```

```
187         }
188
189         else if(i==1){
190             distd_17 = (t4[0]-'0');
191             t4[0]='\0';
192             i=0;
193         }
194
195         //Se obtiene la direcci n de movimiento del Eje x
196         while(t5[i] != '\0'){
197             uart_putc(t5[i]);
198             i++;
199         }
200         uart_putc('\r');
201         uart_putc('\n');
202         uart_putc('\0');
203         dir_17= t5[0]-'0';
204         t5[0] = '\0';
205         i=0;
206
207         //Se obtiene el pwm del servo 1
208         while(t6[i] != '\0'){
209             uart_putc(t6[i]);
210             i++;
211         }
212         uart_putc('\r');
213         uart_putc('\n');
214         uart_putc('\0');
215
216         if(i==3){
217             pwm_servo1 = ((t6[0]-'0')*100) + ((t6[1]-'0')*10) + ...
                    (t6[2]-'0');
218             t6[0]='\0';
219             i=0;
220         }
221
222         else if(i==2){
223             pwm_servo1 = ((t6[0]-'0')*10) + (t6[1]-'0');
224             t6[0]='\0';
```

```
225             i=0;
226         }
227 
228         else if(i==1){
229             pwm_servo1 = (t6[0]-'0');
230             t6[0]='\0';
231             i=0;
232         }
233 
234         //Se obtiene el pwm del servo 2
235 
236         while(t7[i] != '\0'){
237             uart_putc(t7[i]);
238             i++;
239         }
240         uart_putc('\r');
241         uart_putc('\n');
242         uart_putc('\0');
243 
244         if(i==3){
245             pwm_servo2 = ((t7[0]-'0')*100) + ((t7[1]-'0')*10) + ...
                    (t7[2]-'0');
246             t7[0]='\0';
247             i=0;
248         }
249 
250         else if(i==2){
251             pwm_servo2 = ((t7[0]-'0')*10) + (t7[1]-'0');
252             t7[0]='\0';
253             i=0;
254         }
255 
256 
257         else if(i==1){
258             pwm_servo2 = (t7[0]-'0');
259             t7[0]='\0';
260             i=0;
261         }
262 
```

```
263         // FINALIZA LA RECEPCI N DE DATOS
264
265
266         //Ahora, se calculan los pasos que debe de dar el motor ...
                NEMA23
267         dist_mm_23 = distd_23*10; //convierte de cm a mm
268         vueltas_deseadas_23 = (dist_mm_23/AVANCE);
269         pasos_23 = vueltas_deseadas_23*PPR_MOT;
270         pasosd_encoder = pasos_23 * PPR_RATIO;
271
272
273         pos_inicial = contar_pasos();
274
275         mover_nema23_manual(dir_23,pasos_23);
276
277         pos_final = contar_pasos();
278
279
280         if(ovrflw_cnt > 0){
281             pasos_antes_primer_ovfl = PPR_ENCODER-pos_inicial;
282             pasos_antes_ultimo_ovflw = PPR_ENCODER*(ovrflw_cnt-1);
283             pasos_leidos =  ...
                    pasos_antes_primer_ovfl+pasos_antes_ultimo_ovflw ...
                                    +pos_final;
284         }
285         else{
286             pasos_leidos = pos_final;
287         }
288
289         ovrflw_cnt=0;
290
291         if(pasos_leidos != pasosd_encoder){
292
293             error_pos = pasosd_encoder - pasos_leidos;
294
295             if(error_pos!=0){
296
297                 if(dir_23 == 1){
298                     dir_23 = 2;
```

```
299                     mover_nema23_manual(dir_23, error_pos);
300                 }
301                 else if(dir_23 == 2){
302                     dir_23=1;
303                     mover_nema23_manual(dir_23, error_pos);
304                 }
305 
306 
307             }
308 
309         }
310 
311 
312 
313         //Ahora, se calculan los pasos que debe de dar el motor ...
                NEMA17
314         dist_mm_17 = distd_17*10; //convierte de cm a mm
315         vueltas_deseadas_17 = (dist_mm_17/AVANCE);
316         pasos_17 = vueltas_deseadas_17*PPR_MOT;
317         pasosd_encoder2 = pasos_17 * PPR_RATIO;
318 
319 
320         pos_inicial2 = contar_pasos2();
321 
322         mover_nema17_manual(dir_17,pasos_17);
323 
324         pos_final2 = contar_pasos2();
325 
326 
327         if(ovrflw_cnt2 > 0){
328             pasos_antes_primer_ovfl2 = PPR_ENCODER-pos_inicial;
329             pasos_antes_ultimo_ovflw2 = PPR_ENCODER*(ovrflw_cnt2-1);
330             pasos_leidos2 =  ...
                    pasos_antes_primer_ovfl2+pasos_antes_ultimo_ovflw2 ...
                                    +pos_final2;
331         }
332         else{
333             pasos_leidos2 = pos_final2;
334         }
```

```
335
336             ovrflw_cnt2=0;
337
338             if(pasos_leidos2 != pasosd_encoder2){
339
340                 error_pos2 = pasosd_encoder2 - pasos_leidos2;
341
342                 if(error_pos2!=0){
343
344                     if(dir_17 == 1){
345                         dir_17 = 2;
346                         mover_nema17_manual(dir_17, error_pos);
347                     }
348                     else if(dir_17 == 2){
349                         dir_17=1;
350                         mover_nema17_manual(dir_17, error_pos);
351                     }
352
353
354                 }
355
356             }
357
358
359         }
360
361 }
362
363 void tim3_isr(void)
364 {
365     timer_clear_flag(TIM3, TIM_SR_UIF); // Se baja el flag de la ...
          interrupci n
366     ovrflw_cnt2++;
367 }
368
369 void tim4_isr(void)
370 {
371     timer_clear_flag(TIM4, TIM_SR_UIF); // Se baja el flag de la ...
          interrupci n
```

```
372      ovrflw_cnt++;
373  }
```

Archivo Makefile:

```
1  ####################################
2  # PROJECT
3  ####################################
4  BINARY      = main
5  SRCFILES    = main.c delay_us.c nema23.c nema17.c uart_recv.c ...
       encoder1.c encoder2.c pwm.c interrupciones.c
6
7  all: elf bin
8  include ../Makefile.incl
9
10 #END
```

Anexo 10. Código utilizado para la interfaz de usuario en AppDesigner de MATLAB

Las funciones que se han utilizado como scripts de MATLAB para fungir como callbacks en la interfaz de usuario, son las que se muestran debajo.

Función escribir_txt.m:

```
function [mensaje,bandera]=escribir_txt(usr,pwd)
ii=1;
dividir = ["";""];
fid = fopen('usuarios.txt','a+t');

if fid<0
    fprintf('Error abriendo archivo\n');
    return;
end

linea = fgets(fid);
while ischar(linea)
    datos(ii) = {linea};
    conv_string(ii) = string(datos{ii});
    dividir = [dividir split(conv_string(ii),',')];
    linea = fgets(fid);
    ii=ii+1;
end

usuarios = dividir(1,:);
tam= length(usuarios);
usuarios = usuarios(3:tam);

pwds = dividir(2,:);
pwds= pwds(3:tam);
pwds=strtrim(pwds);

id1 = find(usuarios==usr, 1);
```

```
id2 = find(pwds==pwd, 1);

if ((isempty(id1)== 1) && (isempty(id2)==1))
    fprintf(fid,"%s,%s\n",usr,pwd);
    bandera=1;
    mensaje = "Usuario Registrado Exitosamente";
else
    mensaje = "Este usuario ya existe";
    bandera=2;
end


fclose(fid);

end
```

Archivo leer_txt.m:

```
function [usuarios, pwds] = leer_txt(archivo)

ii=1;
dividir = [""];""];
fid = fopen(archivo,'rt');

if fid<0
    fprintf('Error abriendo archivo\n');
    return;
end

linea = fgets(fid);
while ischar(linea)
    datos(ii) = {linea};
    conv_string(ii) = string(datos{ii});
    dividir = [dividir split(conv_string(ii),',')];
    linea = fgets(fid);
    ii=ii+1;
end

```

```
21 usuarios = dividir(1,:);
22 tam= length(usuarios);
23 usuarios = usuarios(3:tam);
24
25 pwds = dividir(2,:);
26 pwds= pwds(3:tam);
27 pwds=strtrim(pwds);
28 %fprintf('\n');
29 fclose(fid);
30 end
```

Archivo validar_login.m:

```
1  function bandera = validar_login(usr,pwd)
2
3  [usrs_registrados, pwds_registradas] = leer_txt('usuarios.txt');
4  id1 = find(usrs_registrados==usr, 1);
5  id2 = find(pwds_registradas==pwd, 1);
6
7  if ((isempty(id1)== 1) || (isempty(id2)==1))
8      bandera= 0;
9  elseif (id1 ≠ id2)
10     bandera= 0;
11 else
12     bandera = 1;
13 end
14
15 end
```

Archivo conectar_BT.m:

```
1  function [dispositivo, msj_conect]=conectar_BT()
2      lista_dispositivos = bluetoothlist;
3      nombre_dispositivo = "Auto2";
4      status_valido = "Ready to connect";
5      canal=1;
6
7      C= table2cell(lista_dispositivos);
```

```
8       [r,c] = size(C);
9       ids = C(1:r,1);
10      status = C(1:r,c);
11
12      ids_str = cellstr(ids);
13      status_str = cellstr(status);
14
15      indice_id = ids_str == nombre_dispositivo;
16      %indice_status= find(status_str == status_valido);
17
18      if strcmp(status_str(indice_id), status_valido)
19          dispositivo = bluetooth(nombre_dispositivo,canal);
20          msj_conect = "Conexion exitosa";
21      else
22          dispositivo = 0;
23          msj_conect = "Ocurrio un error al conectar el ...
                dispositivo BlueTooth. Asegurese de emparejar los ...
                dispositivos.";
24      end
25
26
27
28  end
```

Archivo desconectar_BT.m:

```
1  function desconectar_BT(nombre_dispositivo)
2  nombre = nombre_dispositivo;
3  clear nombre
4  end
```

Archivo limpiar_campos_login.m

```
function limpiar_campos_login(app)

app.Usr.Value = '';
app.Pwd.Value = '';

end
```

Las siguientes secciones de código muestran el código empleado para generar las Apps de MATLAB que se muestran como Interfaz Humano-Máquina.

Archivo gui_GPS.mlapp:

```
classdef gui_GPS < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                matlab.ui.Figure
        PwdLabel                matlab.ui.control.Label
        Pwd                     wt.PasswordField
        LoginexitosoLamp        matlab.ui.control.Lamp
        LoginexitosoLabel       matlab.ui.control.Label
        Rgs                     matlab.ui.control.Button
        Login                   matlab.ui.control.Button
        iconoLogin              matlab.ui.control.Image
        Usr                     matlab.ui.control.EditField
        UsuarioEditFieldLabel   matlab.ui.control.Label
        proyTitulo              matlab.ui.control.Label
        Label                   matlab.ui.control.Label
        IPN                     matlab.ui.control.Label
        logoUPIIZ               matlab.ui.control.Image
        logoIPN                 matlab.ui.control.Image
    end


    properties (Access = public)
```

```
24
25      end
26
27      methods (Access = public)
28
29      end
30
31
32      % Callbacks that handle component events
33      methods (Access = private)
34
35          % Button pushed function: Login
36          function LoginButtonPushed(app, event)
37              usr_ingresado = app.Usr.Value;
38              pwd_ingresada = app.Pwd.Value;
39              bandera = validar_login(usr_ingresado, pwd_ingresada);
40              if bandera == 1
41                  app.LoginexitosoLamp.Color = 'g';
42                  principal
43                  limpiar_campos_login(app);
44                  app.UIFigure.Visible = 'off';
45                  %app.LoginexitosoLamp.Color = '#ebede4';
46              else
47                 app.LoginexitosoLamp.Color  = 'r';
48                 limpiar_campos_login(app);
49                 NoMatch_message
50              end
51          end
52
53          % Button pushed function: Rgs
54          function RgsButtonPushed(app, event)
55              autenticacion_registro
56
57          end
58      end
59
60      % Component initialization
61      methods (Access = private)
62
```

```
63          % Create UIFigure and components
64          function createComponents(app)
65
66              % Create UIFigure and hide until all components are ...
                    created
67              app.UIFigure = uifigure('Visible', 'off');
68              app.UIFigure.Position = [100 100 640 480];
69              app.UIFigure.Name = 'MATLAB App';
70
71              % Create logoIPN
72              app.logoIPN = uiimage(app.UIFigure);
73              app.logoIPN.Position = [1 383 149 85];
74              app.logoIPN.ImageSource = 'logo_ipn.png';
75
76              % Create logoUPIIZ
77              app.logoUPIIZ = uiimage(app.UIFigure);
78              app.logoUPIIZ.Position = [519 382 83 98];
79              app.logoUPIIZ.ImageSource = 'logo_upiiz_sin_fondo.jpg';
80
81              % Create IPN
82              app.IPN = uilabel(app.UIFigure);
83              app.IPN.HorizontalAlignment = 'center';
84              app.IPN.FontSize = 14;
85              app.IPN.Position = [211 437 219 44];
86              app.IPN.Text = 'Instituto Polit cnico Nacional';
87
88              % Create Label
89              app.Label = uilabel(app.UIFigure);
90              app.Label.HorizontalAlignment = 'center';
91              app.Label.Position = [134 420 374 22];
92              app.Label.Text = 'Unidad Profesional ...
                    Interdisciplinaria de Igenier a campus Zacatecas';
93
94              % Create proyTitulo
95              app.proyTitulo = uilabel(app.UIFigure);
96              app.proyTitulo.HorizontalAlignment = 'center';
97              app.proyTitulo.Position = [173 394 295 22];
98              app.proyTitulo.Text = 'Sistema de Emulaci n de GPS ...
                    en Entornos Cerrados';
```

```
99
100             % Create UsuarioEditFieldLabel
101             app.UsuarioEditFieldLabel = uilabel(app.UIFigure);
102             app.UsuarioEditFieldLabel.HorizontalAlignment = 'right';
103             app.UsuarioEditFieldLabel.Position = [209 230 47 22];
104             app.UsuarioEditFieldLabel.Text = 'Usuario';
105
106             % Create Usr
107             app.Usr = uieditfield(app.UIFigure, 'text');
108             app.Usr.Position = [271 230 110 22];
109
110             % Create iconoLogin
111             app.iconoLogin = uiimage(app.UIFigure);
112             app.iconoLogin.Position = [292 279 58 48];
113             app.iconoLogin.ImageSource = 'login.jpg';
114
115             % Create Login
116             app.Login = uibutton(app.UIFigure, 'push');
117             app.Login.ButtonPushedFcn = createCallbackFcn(app, ...
                    @LoginButtonPushed, true);
118             app.Login.Position = [209 139 100 22];
119             app.Login.Text = 'Iniciar Sesi n';
120
121             % Create Rgs
122             app.Rgs = uibutton(app.UIFigure, 'push');
123             app.Rgs.ButtonPushedFcn = createCallbackFcn(app, ...
                    @RgsButtonPushed, true);
124             app.Rgs.Position = [330 139 100 22];
125             app.Rgs.Text = 'Registrarse';
126
127             % Create LoginexitosoLabel
128             app.LoginexitosoLabel = uilabel(app.UIFigure);
129             app.LoginexitosoLabel.HorizontalAlignment = 'right';
130             app.LoginexitosoLabel.Position = [246 75 90 22];
131             app.LoginexitosoLabel.Text = ' Login  exitoso?';
132
133             % Create LoginexitosoLamp
134             app.LoginexitosoLamp = uilamp(app.UIFigure);
135             app.LoginexitosoLamp.Position = [351 75 20 20];
```

```
136             app.LoginexitosoLamp.Color = [0.8 0.8 0.8];
137
138             % Create Pwd
139             app.Pwd = wt.PasswordField(app.UIFigure);
140             app.Pwd.Position = [271 185 110 25];
141
142             % Create PwdLabel
143             app.PwdLabel = uilabel(app.UIFigure);
144             app.PwdLabel.Position = [199 186 68 22];
145             app.PwdLabel.Text = 'Contrase a';
146
147             % Show the figure after all components are created
148             app.UIFigure.Visible = 'on';
149         end
150     end
151
152     % App creation and deletion
153     methods (Access = public)
154
155         % Construct app
156         function app = gui_GPS
157
158             % Create UIFigure and components
159             createComponents(app)
160
161             % Register the app with App Designer
162             registerApp(app, app.UIFigure)
163
164             if nargout == 0
165                 clear app
166             end
167         end
168
169         % Code that executes before app deletion
170         function delete(app)
171
172             % Delete UIFigure when app is deleted
173             delete(app.UIFigure)
174         end
```

```
175     end
176 end
```

Archivo autenticacion_registro.mlapp:

```
1  classdef autenticacion_registro < matlab.apps.AppBase
2
3      % Properties that correspond to app components
4      properties (Access = public)
5          UIFigure       matlab.ui.Figure
6          Label          matlab.ui.control.Label
7          ValidarButton  matlab.ui.control.Button
8          PwdLabel       matlab.ui.control.Label
9          PasswordField  wt.PasswordField
10         ImagenLabel    matlab.ui.control.Label
11         Imagen         matlab.ui.control.Image
12     end
13
14     % Callbacks that handle component events
15     methods (Access = private)
16
17         % Code that executes after component creation
18         function startupFcn(app)
19             app.Label.Visible = 'off';
20         end
21
22         % Button pushed function: ValidarButton
23         function ValidarButtonPushed(app, event)
24             txt_ingresado = app.PasswordField.Value;
25             if  strcmp(txt_ingresado,'LabControlUpiiz1')
26                 app.Label.HandleVisibility = 'off';
27                 Registrar_Usuario
28                 delete(app)
29             else
30                 app.Label.Visible = 'on';
31                 app.Label.FontColor = 'r';
32                 app.Label.Text = 'La contrase a ingresada es ...
                       incorrecta';
```

```
33                  app.PasswordField.Value = '';
34              end
35          end
36      end
37
38      % Component initialization
39      methods (Access = private)
40
41          % Create UIFigure and components
42          function createComponents(app)
43
44              % Create UIFigure and hide until all components are ...
                    created
45              app.UIFigure = uifigure('Visible', 'off');
46              app.UIFigure.Position = [100 100 517 168];
47              app.UIFigure.Name = 'MATLAB App';
48
49              % Create Imagen
50              app.Imagen = uiimage(app.UIFigure);
51              app.Imagen.Position = [41 9 117 119];
52              app.Imagen.ImageSource = 'secure-authentication.png';
53
54              % Create ImagenLabel
55              app.ImagenLabel = uilabel(app.UIFigure);
56              app.ImagenLabel.HorizontalAlignment = 'center';
57              app.ImagenLabel.FontName = 'Arial';
58              app.ImagenLabel.FontSize = 24;
59              app.ImagenLabel.FontWeight = 'bold';
60              app.ImagenLabel.Position = [17 119 164 44];
61              app.ImagenLabel.Text = 'Autenticaci n';
62
63              % Create PasswordField
64              app.PasswordField = wt.PasswordField(app.UIFigure);
65              app.PasswordField.Position = [221 93 230 25];
66
67              % Create PwdLabel
68              app.PwdLabel = uilabel(app.UIFigure);
69              app.PwdLabel.FontSize = 14;
70              app.PwdLabel.Position = [222 135 235 22];
```

```
71              app.PwdLabel.Text = 'Ingrese la contrase a del ...
                    laboratorio';
72
73              % Create ValidarButton
74              app.ValidarButton = uibutton(app.UIFigure, 'push');
75              app.ValidarButton.ButtonPushedFcn = ...
                    createCallbackFcn(app, @ValidarButtonPushed, true);
76              app.ValidarButton.Position = [287 57 100 22];
77              app.ValidarButton.Text = 'Validar';
78
79              % Create Label
80              app.Label = uilabel(app.UIFigure);
81              app.Label.BackgroundColor = [0.8 0.8 0.8];
82              app.Label.HorizontalAlignment = 'center';
83              app.Label.Position = [223 9 229 22];
84              app.Label.Text = '';
85
86              % Show the figure after all components are created
87              app.UIFigure.Visible = 'on';
88          end
89      end
90
91      % App creation and deletion
92      methods (Access = public)
93
94          % Construct app
95          function app = autenticacion_registro
96
97              % Create UIFigure and components
98              createComponents(app)
99
100             % Register the app with App Designer
101             registerApp(app, app.UIFigure)
102
103             % Execute the startup function
104             runStartupFcn(app, @startupFcn)
105
106             if nargout == 0
107                 clear app
```

```
108             end
109         end
110
111         % Code that executes before app deletion
112         function delete(app)
113
114             % Delete UIFigure when app is deleted
115             delete(app.UIFigure)
116         end
117     end
118 end
```

Archivo Registrar_Usuario.mlapp:

```
1  classdef Registrar_Usuario < matlab.apps.AppBase
2
3      % Properties that correspond to app components
4      properties (Access = public)
5          window3                     matlab.ui.Figure
6          registerIcon                matlab.ui.control.Image
7          NuevoUsuarioEditFieldLabel  matlab.ui.control.Label
8          new_usr                     matlab.ui.control.EditField
9          ContraseaEditFieldLabel     matlab.ui.control.Label
10         pwd                         matlab.ui.control.EditField
11         RegistrarButton             matlab.ui.control.Button
12         title                       matlab.ui.control.Label
13         Label                       matlab.ui.control.Label
14     end
15
16     % Callbacks that handle component events
17     methods (Access = private)
18
19         % Button pushed function: RegistrarButton
20         function RegistrarButtonPushed(app, event)
21             nuevo_usuario = app.new_usr.Value;
22             contra = app.pwd.Value;
23             [mensaje, bandera] = escribir_txt(nuevo_usuario,contra);
24             if bandera == 1
```

```
25                  app.Label.FontColor='g';
26                  app.Label.Text = mensaje;
27              else
28                  app.Label.FontColor='r';
29                  app.Label.Text = mensaje;
30              end
31
32          end
33      end
34
35      % Component initialization
36      methods (Access = private)
37
38          % Create UIFigure and components
39          function createComponents(app)
40
41              % Create window3 and hide until all components are ...
                    created
42              app.window3 = uifigure('Visible', 'off');
43              app.window3.Position = [100 100 466 227];
44              app.window3.Name = 'Registrar Nuevo Usuario';
45
46              % Create registerIcon
47              app.registerIcon = uiimage(app.window3);
48              app.registerIcon.Position = [48 64 100 100];
49              app.registerIcon.ImageSource = 'Register.png';
50
51              % Create NuevoUsuarioEditFieldLabel
52              app.NuevoUsuarioEditFieldLabel = uilabel(app.window3);
53              app.NuevoUsuarioEditFieldLabel.HorizontalAlignment = ...
                    'right';
54              app.NuevoUsuarioEditFieldLabel.Position = [156 130 ...
                    85 22];
55              app.NuevoUsuarioEditFieldLabel.Text = 'Nuevo Usuario';
56
57              % Create new_usr
58              app.new_usr = uieditfield(app.window3, 'text');
59              app.new_usr.Position = [256 130 100 22];
60
```

```
61              % Create ContraseaEditFieldLabel
62              app.ContraseaEditFieldLabel = uilabel(app.window3);
63              app.ContraseaEditFieldLabel.HorizontalAlignment = ...
                    'right';
64              app.ContraseaEditFieldLabel.Position = [173 72 68 22];
65              app.ContraseaEditFieldLabel.Text = 'Contrase a';
66
67              % Create pwd
68              app.pwd = uieditfield(app.window3, 'text');
69              app.pwd.Position = [256 72 100 22];
70
71              % Create RegistrarButton
72              app.RegistrarButton = uibutton(app.window3, 'push');
73              app.RegistrarButton.ButtonPushedFcn = ...
                    createCallbackFcn(app, @RegistrarButtonPushed, ...
                    true);
74              app.RegistrarButton.Position = [256 21 100 22];
75              app.RegistrarButton.Text = 'Registrar';
76
77              % Create title
78              app.title = uilabel(app.window3);
79              app.title.HorizontalAlignment = 'center';
80              app.title.FontSize = 14;
81              app.title.Position = [96 182 275 22];
82              app.title.Text = 'Ingrese sus datos para registrar ...
                    su usuario';
83
84              % Create Label
85              app.Label = uilabel(app.window3);
86              app.Label.BackgroundColor = [0.8 0.8 0.8];
87              app.Label.FontColor = [1 0 0];
88              app.Label.Position = [48 21 185 22];
89              app.Label.Text = '';
90
91              % Show the figure after all components are created
92              app.window3.Visible = 'on';
93          end
94      end
95
```

```
96      % App creation and deletion
97      methods (Access = public)
98
99          % Construct app
100         function app = Registrar_Usuario
101
102             % Create UIFigure and components
103             createComponents(app)
104
105             % Register the app with App Designer
106             registerApp(app, app.window3)
107
108             if nargout == 0
109                 clear app
110             end
111         end
112
113         % Code that executes before app deletion
114         function delete(app)
115
116             % Delete UIFigure when app is deleted
117             delete(app.window3)
118         end
119     end
120 end
```

Archivo NoMatch_message.mlapp:

```
1  classdef NoMatch_message < matlab.apps.AppBase
2
3      % Properties that correspond to app components
4      properties (Access = public)
5          window2       matlab.ui.Figure
6          warning_icon  matlab.ui.control.Image
7          msg2          matlab.ui.control.Label
8          msg1          matlab.ui.control.Label
9      end
10
```

```
11      % Component initialization
12      methods (Access = private)
13
14          % Create UIFigure and components
15          function createComponents(app)
16
17              % Create window2 and hide until all components are ...
                    created
18              app.window2 = uifigure('Visible', 'off');
19              app.window2.Position = [100 100 436 163];
20              app.window2.Name = 'Error de Inicio de Sesi n';
21
22              % Create warning_icon
23              app.warning_icon = uiimage(app.window2);
24              app.warning_icon.Position = [25 21 128 123];
25              app.warning_icon.ImageSource = 'warning_icon.png';
26
27              % Create msg2
28              app.msg2 = uilabel(app.window2);
29              app.msg2.HorizontalAlignment = 'center';
30              app.msg2.Position = [152 76 242 36];
31              app.msg2.Text = 'El usuario o la contrase a no son ...
                    correctos.';
32
33              % Create msg1
34              app.msg1 = uilabel(app.window2);
35              app.msg1.HorizontalAlignment = 'center';
36              app.msg1.Position = [191 55 163 22];
37              app.msg1.Text = ' Ya  se encuentra registrado?';
38
39              % Show the figure after all components are created
40              app.window2.Visible = 'on';
41          end
42      end
43
44      % App creation and deletion
45      methods (Access = public)
46
47          % Construct app
```

```
48          function app = NoMatch_message
49
50              % Create UIFigure and components
51              createComponents(app)
52
53              % Register the app with App Designer
54              registerApp(app, app.window2)
55
56              if nargout == 0
57                  clear app
58              end
59          end
60
61          % Code that executes before app deletion
62          function delete(app)
63
64              % Delete UIFigure when app is deleted
65              delete(app.window2)
66          end
67      end
68  end
```

Archivo principal.mlapp:

```
1  classdef principal < matlab.apps.AppBase
2
3      % Properties that correspond to app components
4      properties (Access = public)
5          UIFigure                    matlab.ui.Figure
6          HomeButton                  matlab.ui.control.Button
7          ConnBluetoothPanel          matlab.ui.container.Panel
8          ConVehEx                    matlab.ui.control.Label
9          ConCartesianoEx             matlab.ui.control.Label
10         ConVehButton                matlab.ui.control.Button
11         ConCartesianoButton         matlab.ui.control.Button
12         ConVehculoLabel             matlab.ui.control.Label
13         ConCartesianoLabel          matlab.ui.control.Label
14         Image                       matlab.ui.control.Image
```

```
15          ControlesVehiculoPanel    matlab.ui.container.Panel
16          Lamp                      matlab.ui.control.Lamp
17          Izquierda                 matlab.ui.control.StateButton
18          Atras                     matlab.ui.control.StateButton
19          Derecha                   matlab.ui.control.StateButton
20          DeFrente                  matlab.ui.control.StateButton
21          PosicinActualYLabel       matlab.ui.control.Label
22          PosicinActualXLabel       matlab.ui.control.Label
23          Label_2                   matlab.ui.control.Label
24          Label                     matlab.ui.control.Label
25          ControlesCartesianoPanel  matlab.ui.container.Panel
26          DirY                      matlab.ui.control.DropDown
27          DirX                      matlab.ui.control.DropDown
28          Giro                      matlab.ui.control.Label
29          Vertical                  matlab.ui.control.NumericEditField
30          VerticalEditFieldLabel    matlab.ui.control.Label
31          Horizontal                matlab.ui.control.NumericEditField
32          HorizontalEditFieldLabel  matlab.ui.control.Label
33          EnviarButton              matlab.ui.control.Button
34          EjeY                      matlab.ui.control.NumericEditField
35          EjeYEditFieldLabel        matlab.ui.control.Label
36          EjeX                      matlab.ui.control.NumericEditField
37          EjeXEditFieldLabel        matlab.ui.control.Label
38          coordTitle                matlab.ui.control.Label
39          Current                   matlab.ui.control.Label
40          CurrentTitle              matlab.ui.control.Label
41          Title                     matlab.ui.control.Label
42          Modo                      matlab.ui.control.Button
43          UIAxes                    matlab.ui.control.UIAxes
44      end
45
46
47      properties (Access = public)
48
49          cartesiano % Objeto bluetooth del cartesiano
50          auto %Objeto bluetooth del auto a escala
51          servidor %Objeto del servidor protocolo TCP/IP
52      end
53
```

```
54
55      % Callbacks that handle component events
56      methods (Access = private)
57
58          % Code that executes after component creation
59          function startupFcn(app)
60              app.ConnBluetoothPanel.Visible = 'off';
61              app.ControlesCartesianoPanel.Visible = 'off';
62              app.ControlesVehiculoPanel.Visible = 'off';
63              app.UIAxes.Visible = 'off';
64              app.PosicinActualXLabel.Visible = 'off';
65              app.PosicinActualYLabel.Visible = 'off';
66              app.Label.Visible= 'off';
67              app.Label_2.Visible = 'off';
68              app.ConCartesianoEx.Visible = 'off';
69              app.ConVehEx.Visible = 'off';
70          end
71
72          % Button pushed function: Modo
73          function ModoButtonPushed(app, event)
74              %app.servidor = server = tcpserver("0.0.0.0",30000);
75              texto = app.Modo.Text;
76
77              switch (texto)
78                  case 'Modo Manual'
79                      app.Modo.Text = 'Modo Automatico';
80                      app.Current.Text = 'Modo Manual';
81                      app.ControlesCartesianoPanel.Visible = 'on';
82                      app.ConnBluetoothPanel.Visible = 'on';
83                      app.ControlesVehiculoPanel.Visible = 'on';
84                      app.UIAxes.Visible = 'on';
85                      app.PosicinActualXLabel.Visible = 'on';
86                      app.PosicinActualYLabel.Visible = 'on';
87                      app.Label.Visible= 'on';
88                      app.Label_2.Visible = 'on';
89                      app.UIAxes.Position = [372,338,321,245];
90                      app.PosicinActualXLabel.Position = ...
                            [430,271,99,22];
```

```
91                      app.PosicinActualYLabel.Position = ...
                            [574,271,99,22];
92                      app.Label.Position= [441,306,77,22];
93                      app.Label_2.Position = [585,306,77,22];
94                  case 'Modo Automatico'
95                      app.Modo.Text = 'Modo Manual';
96                      app.Current.Text = 'Modo Automatico';
97                      app.ControlesCartesianoPanel.Visible = 'off';
98                      app.ConnBluetoothPanel.Visible = 'off';
99                      app.ControlesVehiculoPanel.Visible = 'off';
100                     app.UIAxes.Visible = 'on';
101                     app.PosicinActualXLabel.Visible = 'on';
102                     app.PosicinActualYLabel.Visible = 'on';
103                     app.Label.Visible= 'on';
104                     app.Label_2.Visible = 'on';
105                     app.UIAxes.Position = [140,200,500,350];
106                     app.PosicinActualXLabel.Position = ...
                            [250,90,99,22];
107                     app.PosicinActualYLabel.Position = ...
                            [500,90,99,22];
108                     app.Label.Position= [255,120,77,22];
109                     app.Label_2.Position = [505,120,77,22];
110             end
111
112         end
113
114         % Button pushed function: ConCartesianoButton
115         function ConCartesianoButtonPushed(app, event)
116             app.cartesiano = bluetooth("cartesiano",1);
117             app.ConCartesianoEx.Visible = 'on';
118         end
119
120         % Button pushed function: ConVehButton
121         function ConVehButtonPushed(app, event)
122             app.auto = bluetooth("Auto",1);
123             app.ConVehEx.Visible = 'on';
124         end
125
126         % Button pushed function: EnviarButton
```

```
127         function EnviarButtonPushed(app, event)
128             mover_x= num2str(app.EjeX.Value);
129             mover_y= num2str(app.EjeY.Value);
130             servo_horizontal= num2str(app.Horizontal.Value);
131             servo_vertical= num2str(app.Vertical.Value);
132 
133             seleccionX = app.DirX.Value;
134             if strcmpi(seleccionX,'+')
135                 dir1='1';
136             elseif strcmpi(seleccionX,'-')
137                 dir1='2';
138             end
139 
140             seleccionY = app.DirY.Value;
141             if strcmpi(seleccionY,'+')
142                 dir2='1';
143             elseif strcmpi(seleccionY,'-')
144                 dir2='2';
145             end
146 
147             tok = ',';
148             salto= '\n';
149             mensaje = ...
                    strcat(tok,'manual',tok,mover_x,tok,dir1,tok,mover_y,tok,dir2,tok,s
150             writeline(app.cartesiano,mensaje);
151         end
152 
153         % Value changed function: DeFrente
154         function DeFrenteValueChanged(app, event)
155             boton1_val = app.DeFrente.Value;
156             boton2_val = app.Derecha.Value;
157             boton3_val = app.Atras.Value;
158             boton4_val = app.Izquierda.Value;
159 
160             if ((boton1_val == 1) && (boton2_val == 0) && ...
                    (boton3_val == 0) && (boton4_val == 0))
161                 app.Lamp.Color = 'g';
162                 mensaje = 'adelante';
163                 writeline(app.auto,mensaje);
```

```
164             else
165                 app.Lamp.Color = 'r';
166             end
167 
168         end
169 
170         % Value changed function: Derecha
171         function DerechaValueChanged(app, event)
172             boton2_val = app.Derecha.Value;
173             boton1_val = app.DeFrente.Value;
174             boton3_val = app.Atras.Value;
175             boton4_val = app.Izquierda.Value;
176 
177              if ((boton1_val == 0) && (boton2_val == 1) && ...
                     (boton3_val == 0) && (boton4_val == 0))
178                 app.Lamp.Color = 'g';
179                 mensaje = 'derecha';
180                 writeline(app.auto,mensaje);
181             else
182                 app.Lamp.Color = 'r';
183             end
184 
185         end
186 
187         % Value changed function: Atras
188         function AtrasValueChanged(app, event)
189             boton3_val = app.Atras.Value;
190             boton1_val = app.DeFrente.Value;
191             boton2_val = app.Derecha.Value;
192             boton4_val = app.Izquierda.Value;
193 
194              if ((boton1_val == 0) && (boton2_val == 0) && ...
                     (boton3_val == 1) && (boton4_val == 0))
195                 app.Lamp.Color = 'g';
196                 mensaje = 'atras';
197                 writeline(app.auto,mensaje);
198             else
199                 app.Lamp.Color = 'r';
200             end
```

```
201
202         end
203
204         % Value changed function: Izquierda
205         function IzquierdaValueChanged(app, event)
206             boton4_val = app.Izquierda.Value;
207             boton1_val = app.DeFrente.Value;
208             boton2_val = app.Derecha.Value;
209             boton3_val = app.Atras.Value;
210
211             if ((boton1_val == 0) && (boton2_val == 0) && ...
                    (boton3_val == 0) && (boton4_val == 1))
212                 app.Lamp.Color = 'g';
213                 mensaje = 'izquierda';
214                 writeline(app.auto,mensaje);
215             else
216                 app.Lamp.Color = 'r';
217             end
218
219         end
220
221         % Close request function: UIFigure
222         function UIFigureCloseRequest(app, event)
223             delete(app)
224         end
225     end
226
227     % Component initialization
228     methods (Access = private)
229
230         % Create UIFigure and components
231         function createComponents(app)
232
233             % Create UIFigure and hide until all components are ...
                    created
234             app.UIFigure = uifigure('Visible', 'off');
235             app.UIFigure.Position = [100 100 761 696];
236             app.UIFigure.Name = 'MATLAB App';
```

```
237             app.UIFigure.CloseRequestFcn = ...
                    createCallbackFcn(app, @UIFigureCloseRequest, true);
238
239             % Create UIAxes
240             app.UIAxes = uiaxes(app.UIFigure);
241             title(app.UIAxes, 'Title')
242             xlabel(app.UIAxes, 'X')
243             ylabel(app.UIAxes, 'Y')
244             zlabel(app.UIAxes, 'Z')
245             app.UIAxes.PlotBoxAspectRatio = [1.4974358974359 1 1];
246             app.UIAxes.Position = [372 338 321 245];
247
248             % Create Modo
249             app.Modo = uibutton(app.UIFigure, 'push');
250             app.Modo.ButtonPushedFcn = createCallbackFcn(app, ...
                    @ModoButtonPushed, true);
251             app.Modo.Position = [211 619 120 22];
252             app.Modo.Text = 'Modo Manual';
253
254             % Create Title
255             app.Title = uilabel(app.UIFigure);
256             app.Title.Position = [175 649 215 25];
257             app.Title.Text = 'Seleccione el Modo de Operaci n:';
258
259             % Create CurrentTitle
260             app.CurrentTitle = uilabel(app.UIFigure);
261             app.CurrentTitle.Position = [428 650 79 22];
262             app.CurrentTitle.Text = 'Modo Actual: ';
263
264             % Create Current
265             app.Current = uilabel(app.UIFigure);
266             app.Current.BackgroundColor = [0.8 0.8 0.8];
267             app.Current.Position = [417 619 101 22];
268             app.Current.Text = '';
269
270             % Create ControlesCartesianoPanel
271             app.ControlesCartesianoPanel = uipanel(app.UIFigure);
272             app.ControlesCartesianoPanel.TitlePosition = ...
                    'centertop';
```

```
273             app.ControlesCartesianoPanel.Title = 'Controles ...
                    Cartesiano';
274             app.ControlesCartesianoPanel.Position = [59 280 233 ...
                    311];
275
276             % Create coordTitle
277             app.coordTitle = uilabel(app.ControlesCartesianoPanel);
278             app.coordTitle.HorizontalAlignment = 'center';
279             app.coordTitle.Position = [44 247 162 22];
280             app.coordTitle.Text = 'Introduzca coordenadas (cm)';
281
282             % Create EjeXEditFieldLabel
283             app.EjeXEditFieldLabel = ...
                    uilabel(app.ControlesCartesianoPanel);
284             app.EjeXEditFieldLabel.HorizontalAlignment = 'right';
285             app.EjeXEditFieldLabel.Position = [26 207 34 22];
286             app.EjeXEditFieldLabel.Text = 'Eje X';
287
288             % Create EjeX
289             app.EjeX = uieditfield(app.ControlesCartesianoPanel, ...
                    'numeric');
290             app.EjeX.Position = [75 207 100 22];
291
292             % Create EjeYEditFieldLabel
293             app.EjeYEditFieldLabel = ...
                    uilabel(app.ControlesCartesianoPanel);
294             app.EjeYEditFieldLabel.HorizontalAlignment = 'right';
295             app.EjeYEditFieldLabel.Position = [26 169 34 22];
296             app.EjeYEditFieldLabel.Text = 'Eje Y';
297
298             % Create EjeY
299             app.EjeY = uieditfield(app.ControlesCartesianoPanel, ...
                    'numeric');
300             app.EjeY.Position = [75 169 100 22];
301
302             % Create EnviarButton
303             app.EnviarButton = ...
                    uibutton(app.ControlesCartesianoPanel, 'push');
```

```
304             app.EnviarButton.ButtonPushedFcn = ...
                    createCallbackFcn(app, @EnviarButtonPushed, true);
305             app.EnviarButton.Position = [75 15 100 22];
306             app.EnviarButton.Text = 'Enviar';
307
308             % Create HorizontalEditFieldLabel
309             app.HorizontalEditFieldLabel = ...
                    uilabel(app.ControlesCartesianoPanel);
310             app.HorizontalEditFieldLabel.HorizontalAlignment = ...
                    'right';
311             app.HorizontalEditFieldLabel.Position = [0 91 60 22];
312             app.HorizontalEditFieldLabel.Text = 'Horizontal';
313
314             % Create Horizontal
315             app.Horizontal = ...
                    uieditfield(app.ControlesCartesianoPanel, ...
                    'numeric');
316             app.Horizontal.Position = [75 91 100 22];
317
318             % Create VerticalEditFieldLabel
319             app.VerticalEditFieldLabel = ...
                    uilabel(app.ControlesCartesianoPanel);
320             app.VerticalEditFieldLabel.HorizontalAlignment = ...
                    'right';
321             app.VerticalEditFieldLabel.Position = [18 58 45 22];
322             app.VerticalEditFieldLabel.Text = 'Vertical';
323
324             % Create Vertical
325             app.Vertical = ...
                    uieditfield(app.ControlesCartesianoPanel, ...
                    'numeric');
326             app.Vertical.Position = [75 58 103 22];
327
328             % Create Giro
329             app.Giro = uilabel(app.ControlesCartesianoPanel);
330             app.Giro.HorizontalAlignment = 'center';
331             app.Giro.Position = [53 125 127 22];
332             app.Giro.Text = 'Girar c mara (0  -180  )';
333
```

```
334             % Create DirX
335             app.DirX = uidropdown(app.ControlesCartesianoPanel);
336             app.DirX.Items = {'+', '-'};
337             app.DirX.Position = [183 207 41 22];
338             app.DirX.Value = '+';
339 
340             % Create DirY
341             app.DirY = uidropdown(app.ControlesCartesianoPanel);
342             app.DirY.Items = {'+', '-'};
343             app.DirY.Position = [183 169 41 22];
344             app.DirY.Value = '+';
345 
346             % Create Label
347             app.Label = uilabel(app.UIFigure);
348             app.Label.BackgroundColor = [0.651 0.651 0.651];
349             app.Label.HorizontalAlignment = 'center';
350             app.Label.Position = [441 306 77 22];
351             app.Label.Text = '';
352 
353             % Create Label_2
354             app.Label_2 = uilabel(app.UIFigure);
355             app.Label_2.BackgroundColor = [0.651 0.651 0.651];
356             app.Label_2.HorizontalAlignment = 'center';
357             app.Label_2.Position = [585 306 77 22];
358             app.Label_2.Text = '';
359 
360             % Create PosicinActualXLabel
361             app.PosicinActualXLabel = uilabel(app.UIFigure);
362             app.PosicinActualXLabel.Position = [430 271 99 22];
363             app.PosicinActualXLabel.Text = 'Posici n Actual X';
364 
365             % Create PosicinActualYLabel
366             app.PosicinActualYLabel = uilabel(app.UIFigure);
367             app.PosicinActualYLabel.Position = [574 271 99 22];
368             app.PosicinActualYLabel.Text = 'Posici n Actual Y';
369 
370             % Create ControlesVehiculoPanel
371             app.ControlesVehiculoPanel = uipanel(app.UIFigure);
372             app.ControlesVehiculoPanel.TitlePosition = 'centertop';
```

```
373             app.ControlesVehiculoPanel.Title = 'Controles ...
                    Veh culo';
374             app.ControlesVehiculoPanel.Position = [54 17 260 234];
375
376             % Create DeFrente
377             app.DeFrente = uibutton(app.ControlesVehiculoPanel, ...
                    'state');
378             app.DeFrente.ValueChangedFcn = ...
                    createCallbackFcn(app, @DeFrenteValueChanged, true);
379             app.DeFrente.Icon = 'up_arrow.png';
380             app.DeFrente.Text = '';
381             app.DeFrente.Position = [92 132 76 55];
382
383             % Create Derecha
384             app.Derecha = uibutton(app.ControlesVehiculoPanel, ...
                    'state');
385             app.Derecha.ValueChangedFcn = createCallbackFcn(app, ...
                    @DerechaValueChanged, true);
386             app.Derecha.Icon = 'right_arrow.png';
387             app.Derecha.Text = '';
388             app.Derecha.Position = [167 78 76 55];
389
390             % Create Atras
391             app.Atras = uibutton(app.ControlesVehiculoPanel, ...
                    'state');
392             app.Atras.ValueChangedFcn = createCallbackFcn(app, ...
                    @AtrasValueChanged, true);
393             app.Atras.Icon = 'down_arrow.png';
394             app.Atras.Text = '';
395             app.Atras.Position = [90 24 78 55];
396
397             % Create Izquierda
398             app.Izquierda = uibutton(app.ControlesVehiculoPanel, ...
                    'state');
399             app.Izquierda.ValueChangedFcn = ...
                    createCallbackFcn(app, @IzquierdaValueChanged, ...
                    true);
400             app.Izquierda.Icon = 'left_arrow.png';
401             app.Izquierda.Text = '';
```

```
402             app.Izquierda.Position = [13 78 78 55];
403
404             % Create Lamp
405             app.Lamp = uilamp(app.ControlesVehiculoPanel);
406             app.Lamp.Position = [237 5 20 20];
407             app.Lamp.Color = [0.651 0.651 0.651];
408
409             % Create ConnBluetoothPanel
410             app.ConnBluetoothPanel = uipanel(app.UIFigure);
411             app.ConnBluetoothPanel.TitlePosition = 'centertop';
412             app.ConnBluetoothPanel.Title = 'Conexi n Bluetooth';
413             app.ConnBluetoothPanel.Position = [355 85 372 166];
414
415             % Create Image
416             app.Image = uiimage(app.ConnBluetoothPanel);
417             app.Image.Position = [16 45 89 93];
418             app.Image.ImageSource = 'bluetooth_logo.png';
419
420             % Create ConCartesianoLabel
421             app.ConCartesianoLabel = ...
                    uilabel(app.ConnBluetoothPanel);
422             app.ConCartesianoLabel.Position = [119 94 118 22];
423             app.ConCartesianoLabel.Text = 'Conexi n Cartesiano';
424
425             % Create ConVehculoLabel
426             app.ConVehculoLabel = uilabel(app.ConnBluetoothPanel);
427             app.ConVehculoLabel.Position = [252 94 106 22];
428             app.ConVehculoLabel.Text = 'Conexi n Veh culo';
429
430             % Create ConCartesianoButton
431             app.ConCartesianoButton = ...
                    uibutton(app.ConnBluetoothPanel, 'push');
432             app.ConCartesianoButton.ButtonPushedFcn = ...
                    createCallbackFcn(app, ...
                    @ConCartesianoButtonPushed, true);
433             app.ConCartesianoButton.Position = [128 53 100 22];
434             app.ConCartesianoButton.Text = 'Conectar';
435
436             % Create ConVehButton
```

```
437             app.ConVehButton = uibutton(app.ConnBluetoothPanel, ...
                    'push');
438             app.ConVehButton.ButtonPushedFcn = ...
                    createCallbackFcn(app, @ConVehButtonPushed, true);
439             app.ConVehButton.Position = [255 53 100 22];
440             app.ConVehButton.Text = 'Conectar';
441 
442             % Create ConCartesianoEx
443             app.ConCartesianoEx = uilabel(app.ConnBluetoothPanel);
444             app.ConCartesianoEx.BackgroundColor = [0.651 0.651 ...
                    0.651];
445             app.ConCartesianoEx.HorizontalAlignment = 'center';
446             app.ConCartesianoEx.FontColor = [0 1 0];
447             app.ConCartesianoEx.Position = [148 24 60 22];
448             app.ConCartesianoEx.Text = '';
449 
450             % Create ConVehEx
451             app.ConVehEx = uilabel(app.ConnBluetoothPanel);
452             app.ConVehEx.BackgroundColor = [0.651 0.651 0.651];
453             app.ConVehEx.FontColor = [0 1 0];
454             app.ConVehEx.Position = [275 24 59 22];
455             app.ConVehEx.Text = '';
456 
457             % Create HomeButton
458             app.HomeButton = uibutton(app.UIFigure, 'push');
459             app.HomeButton.Position = [531 42 100 22];
460             app.HomeButton.Text = 'Home';
461 
462             % Show the figure after all components are created
463             app.UIFigure.Visible = 'on';
464         end
465     end
466 
467     % App creation and deletion
468     methods (Access = public)
469 
470         % Construct app
471         function app = principal
472 
```

```
473             % Create UIFigure and components
474             createComponents(app)
475 
476             % Register the app with App Designer
477             registerApp(app, app.UIFigure)
478 
479             % Execute the startup function
480             runStartupFcn(app, @startupFcn)
481 
482             if nargout == 0
483                 clear app
484             end
485         end
486 
487         % Code that executes before app deletion
488         function delete(app)
489 
490             % Delete UIFigure when app is deleted
491             delete(app.UIFigure)
492         end
493     end
494 end
```

Anexo 11. Código utilizado para el control del vehículo a escala

Se utilizó un Arduino Nano para el control del vehículo. El siguiente código muestra la recepción de las variables de velocidad a través del módulo Bluetooth y la asignación de las mismas a los motores del vehículo.
Archivo recibir_velocidad_BT.ino:

```
#include <SoftwareSerial.h>

#define ENA 5
#define IN1 6
#define IN2 7

SoftwareSerial Auto(2,3);

String s = "";
char c = 0;
bool nueva_linea= false;
int pin_led1= 5;
int pin_led2= 6;
void setup() {
  pinMode(ENA,OUTPUT);
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  Serial.begin(9600);
  Auto.begin(38400);

}

void loop() {

  while(Auto.available()){

      c = Auto.read();
```

```
28        if(c=='\r'){
29
30            continue;
31
32        }
33        else if(c=='\n'){
34          nueva_linea=true;
35          break;
36        }
37        else{
38          s=s+c;
39        }
40    }
41
42    if(nueva_linea){
43      Serial.print("Longitud de la cadena: ");
44      Serial.println(s.length());
45      Serial.print("String encontrado: ");
46      Serial.println(s);
47
48      char s_array[s.length()];
49      s.toCharArray(s_array, s.length());
50
51      char* mensaje = strtok(s_array,",");
52      Serial.print("Mensaje: ");
53      Serial.println(mensaje);
54
55      char* ch_velocidad = strtok(NULL,",");
56      Serial.print("Velocidad: ");
57      Serial.println(ch_velocidad);
58
59      char* ch_direccion = strtok(NULL,",");
60      Serial.print("Direccion: ");
61      Serial.println(ch_direccion);
62
63        int valor_pwm = atoi(ch_velocidad);
64        int direccion = atoi(ch_direccion);
65
66      if(valor_pwm>255){
```

```
67        valor_pwm=255;
68      }
69      else if(valor_pwm<0){
70        valor_pwm=0;
71      }
72
73  //    if(estado == 1){
74  //      digitalWrite(pin_led2,HIGH);
75  //    }
76  //    else{
77  //      digitalWrite(pin_led2,LOW);
78  //    }
79
80      switch(direccion){
81
82            case 1:
83                  digitalWrite(IN1, HIGH);
84                  digitalWrite(IN2, LOW);
85                  break;
86           case 2:
87                  digitalWrite(IN1, LOW);
88                  digitalWrite(IN2, HIGH);
89                  break;
90
91          default:
92                  digitalWrite(IN1, LOW);
93                  digitalWrite(IN2, LOW);
94      }
95
96      analogWrite(ENA,valor_pwm);
97
98      s="";
99      nueva_linea=false;
100   }
```

Archivo prueba_motor_y_encoder_Auto_1.ino:

```
1  /*************************************************************
```

```
 * Instituto Politécnico Nacional (IPN)                    *
 * Unidad Profesional Interdisciplinaria de Ingeniería Campus ...
    *Zacatecas (UPIIZ)                                      *
*Código por  Erik Alejandro Garay Rodríguez                 *
************************************************************/
#define N 20                    //Número de ranuras en el encoder
#define DIAMETRO 6.6            //Diámetro de la llanta en ...
    centímetros
#define CONTADOR_TICKS 3    //Número de ticks para cálculo de ...
    velocidad
#define TAM 10              //Tamaño del vector del cálculo de ...
    promedio
#define ENCODER_R 2       //Pin de conexión del encoder derecho.
#define PWM1 6          //Pin 6 arduino al pin 2 del L293D
#define PWM2 5          //Pin 5 arduino al pin 6 del L293D
//PWM mínimo en el motor derecho es de 90 (valor ADC)

volatile unsigned muestreo_actual=0;
volatile unsigned muestreo_anterior=0;
volatile unsigned ΔMuestreo;

//*****************Variables del motor derecho*****************//

volatile unsigned muestreoActualInterrupcionR   = 0;      ...
    //Variables de definición del tiempo de interrupción y ...
    cálculo de la velocidad del motor derecho
volatile unsigned muestreoAnteriorInterrupcionR = 0;
double ΔMuestreoInterrupcionR = 0;

double frecuenciaR = 0;                         //Frecuencia de ...
    interrupción llanta R.
double Wr = 0;                                  //Velocidad ...
    angular R
double Vr = 0;                                 //Velocidad lineal
int CR=0;                                     //contador Ticks
float vectorR[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};  //vector de ...
    almacenamiento de datos para promedio del tiempo de ...
    interrupciones

```

```
31
32 void setup(){
33
34   attachInterrupt(digitalPinToInterrupt(ENCODER_R), encoder_r, ...
        FALLING);
35   //attachInterrupt(digitalPinToInterrupt(ENCODER_L), encoder_l, ...
        FALLING);
36   Serial.begin(9600);
37 }
38
39 void encoder_r(){                                    //Funci n de ...
     interrupci n del encoder de la llanta derecha
40
41   CR++;
42
43   if(CR == CONTADOR_TICKS){
44
45     float media = 0;
46     ΔMuestreoInterrupcionR = muestreoActualInterrupcionR - ...
          muestreoAnteriorInterrupcionR; //diferencia de tiempos ...
          de interrupciones de ticks del motor
47
48     for(int i = 0; i<TAM-1;i++){
49       vectorR[i] = vectorR[i+1]; ...
                                                  //Se rellena el ...
            vector para calcular el promedio posteriormente
50     }
51     vectorR[TAM-1]=ΔMuestreoInterrupcionR; ...
                                  // ltimo  dato del vector
52
53     for(int i = 0; i<TAM;i++){
54       media = vectorR[i] + media;
55     }
56     media=media/TAM;
57     ΔMuestreoInterrupcionR = media; ...
                                       //se reemplaza por el ...
          valor de su media
58
```

```
59      frecuenciaR = 1000/ ΔMuestreoInterrupcionR;                    ...
           //frecuencia de interrupci n
60      muestreoAnteriorInterrupcionR = muestreoActualInterrupcionR; ...
              //Se actualiza el tiempo de interrupci n anterior.
61      CR=0; ...
                                                                       ...
           //Reinicio del contador.
62    }
63  }
64
65  void loop(){
66    muestreo_actual = millis();
67    muestreoActualInterrupcionR= millis();
68    ΔMuestreo=(double) muestreo_actual - muestreo_anterior;
69
70    if(ΔMuestreo≥k){
71      haha
72    }
73
74    Wr = CONTADOR_TICKS*((2*PI)/N)*frecuenciaR;                    ...
          //frecuencia angular rad/s
75    Vr = Wr*(DIAMETRO/2);                                          ...
          //velocidad lineal cm/s
76
77    analogWrite(PWM1,80);                                         ...
          //la llanta derecha avanza hacia el frente
78    analogWrite(PWM2,0);
79
80    Serial.println(Vr);                                            // ...
          se muestra la velocidad lineal en la llanta derecha.
81
82  }
```