UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

DISEÑO Y CONSTRUCCIÓN DE UN ROBOT PARALELO TIPO DELTA CONTROLADO CON TIVA TM4C123G

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO ELÉCTRICO ELECTRÓNICO

P R E S E N T A:

IAN MICHEL CRUZ SORIA

ASESOR:

DR. PATRICIO MARTÍNEZ ZAMUDIO

Ciudad Nezahualcóyotl, Estado de México; 2020

Agradecimientos

A mi familia por ser inspiración, ejemplo, pilares y raíces, por inculcarme valores, por apoyarme y soportarme diario; a mis padres por ser ejemplo de esfuerzo, responsabilidad, trabajo y constancia, por su creatividad e inventiva, a mi hermano por su gran apoyo incondicional y ánimo, a mi hermana por su ánimo y aspiraciones, a mi tía por apoyar mis estudios y formación, a mis abuelos por ser ejemplo de vida digna y haber criado personas de bien con gran voluntad.

A mis maestros, compañeros y amigos por su apoyo y buenos recuerdos.

Al Mtro. De Matías Aguilar por ser mentor y guía, por su apoyo, conocimientos y enseñanzas que permitieron hacer realidad este trabajo.

Al Dr. Patricio Martínez Zamudio por ser guía, por su apoyo y conocimientos que permitieron documentar el trabajo.

Al Mtro. Juan Gastaldi Pérez, M. en I. Fernando Macedo Chagolla, Ing. Ramón Patiño Rodríguez, Ing. Oscar Guadalupe Moreno Espinoza, integrantes del Comité de Sinodales por su tiempo y comentarios en la revisión de esta tesis.

A los profesores que permitieron un crecimiento en mi como estudiante, por su apoyo, retos y conocimientos, al Ing. Luna Escorza Pablo Prócoro, Ing. Ramírez Mora José Manuel, Mtro. Arturo Ocampo Álvarez, Dr. Ismael Díaz Rangel, Dr. Sergio Guillermo Torres Cedillo, Ing. Ivan Leos Santiago, Dr. Miguel A. Padilla Castañeda, Mtro. Rafael Antonio Márquez Ramírez, Ing. Julián Zúñiga Navarrete.

Resumen

En este trabajo de tesis se presenta el prototipo piloto funcional de un robot paralelo delta, el sistema embebido de control, una interfaz de usuario; se presenta el diseño mecánico y análisis cinemático de la posición con el fin de obtener las ecuaciones que rigen el movimiento del efector final de la plataforma móvil; dentro de un entorno grafico se desarrolla una plataforma virtual de simulación para validar el diseño y las ecuaciones de cinemática inversa, aprobando o rechazando la construcción del prototipo; se muestra el diseño y manifactura de PCB generado conforme al hardware electrónico (sensores y actuadores) definido por las propiedades y tareas proyectadas del robot; se realiza la caracterización del hardware para conocer su comportamiento e introducirlo en el algoritmo de control; en el microcontrolador se programa un algoritmo de control de lazo cerrado de tipo proporcional, el cual involucra los sensores de posición angular como realimentación y controla la posición angular del actuador de cada brazo como salida; además se desarrolla una interfaz gráfica de usuario (GUI) con propiedades y restricciones de acuerdo al espacio de trabajo de la plataforma, la cual se comunica con el microcontrolador del robot; la GUI se implementa en un módulo externo como interfaz hombre máquina (HMI), la cual brinda al usuario un medio de interacción amigable con el robot, permitiéndole ingresar un punto de interés dentro de los límites de movimiento del robot paralelo tipo delta.

DISEÑO Y CONSTRUCCIÓN DE UN ROBOT PARALELO TIPO DELTA CONTROLADO CON TIVA TM4C123G

Índice

Tabla de Figuras

INTRODUCCIÓN

Historia de la robótica

Una gran muestra de la capacidad de inventiva del hombre es la creación de los autómatas, en los cuales se ha intentado reproducir artificialmente el movimiento de los seres vivos.

Algunas de las primeras ideas plasmadas sobre autómatas son: la Ilíada, con la mención de trípodes semovientes creados por Hefesto; la leyenda de Arquitas de Tarento y su pájaro mecánico propulsado por aire comprimido (400 a.C.); la *Pneumática,* Herón describe la eolípila, primera máquina de vapor, precursora de las turbinas de vapor modernas (Fig. 1); el sirviente automático de Alberto Magno (siglo XII) el cual acuño el termino *androide*. [1]

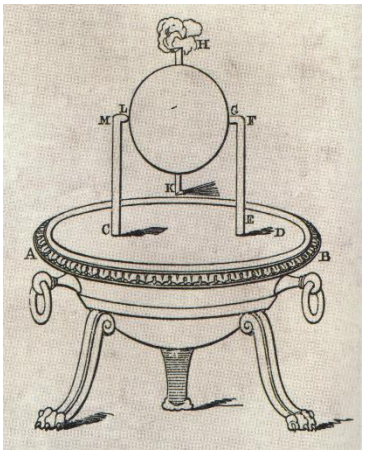


Fig. 1 La eolípila construida por Herón

Leonardo Da Vinci (Siglo XV) construyó autómatas de renombre, algunos son un carro mecanizado y un león mecánico móvil, al cual se le abría el pecho (Fig. 2), posteriormente en la creación de autómatas resaltan nombres como: Jannello Torriani de Cremona; Salomon de Caus; Christiaan Huygens; Jacques de Vaucanson al cual se le recuerda como el más célebre inventor de autómatas, algunas de sus creaciones más reconocidas son, el flautista androide que tocaba doce melodías, el pato artificial que bebía, comía y digería (Fig. 3); Pierre Jaquet-Droz construyo tres autómatas, los cuales se conservan en el museo suizo Neuchâtel, son conocidos como el escritor, el músico y el dibujante (Fig. 4).

Fig. 2 El carro y el leon de Leonardo

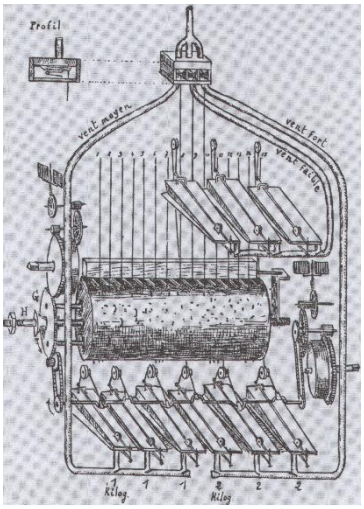


Fig. 3 El mecanismo del Flautista de Vaucanson

Fig. 4 El androide-escritor de Pierre Jacquet-Droz

El término *robot* surgió en la fantasía, se usó por primera vez en 1921 por Karel Čapek en su obra Rossum Universal Robots (RUR), tiene su origen de la palabra checa *robota* que significa trabajo duro o trabajo forzoso. Inicialmente se empleó para designar mecanismos con acciones complejas e incluso a algunos sistemas de relojería.

En la actualidad los robots abundan en la ciencia ficción, el cine y llegan a tener presencia en múltiples sectores de la actividad humana como son: agricultura, salud, producción industrial, hasta exploración, por mencionar algunos.

El uso de robots permite la manipulación de objetos en áreas de difícil acceso o en condiciones peligrosas, como zonas radioactivas, exploraciones subacuáticas y del espacio, evitando arriesgar la integridad del ser humano.

De acuerdo con la definición dada en 1980 por *Robot Institute of America (RIA):*

"Un robot es un manipulador multifuncional reprogramable, diseñado para mover materiales, partes herramientas o dispositivos especializados a través de movimientos programados para la ejecución de una variedad de tareas." [2]

"La robótica es una disciplina científica que aborda la investigación y desarrollo de una clase particular de sistemas mecánicos, denominados *robots manipuladores*, diseñados para realizar una amplia variedad de aplicaciones industriales, científicas, domésticas y comerciales." [2]

La robótica es una labor interdisciplinaria que hace uso de tres disciplinas para desarrollar los componentes de cualquier sistema robótico, las cuales son: Ingeniería Mecánica, Ingeniería Electrónica y Ciencias de la Computación.

"Actualmente existe una gran variedad de robots con diversas estructuras geométricas y mecánicas que definen su funcionalidad y aplicación."(Fig. 5) [2]

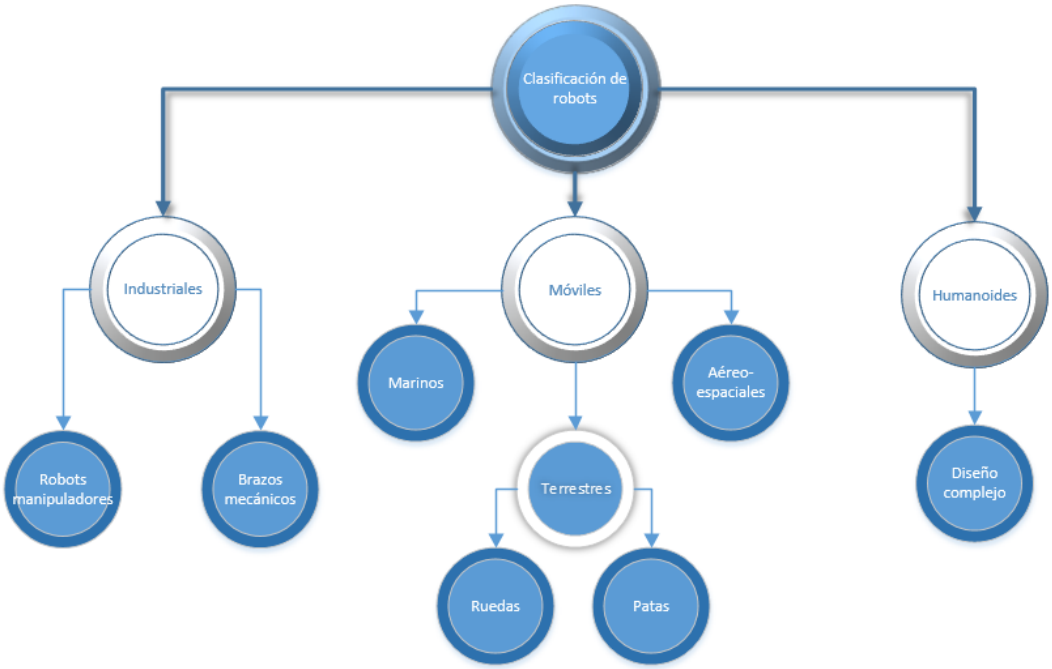


Fig. 5 Clasificación general de los robots

Funcionamiento general de un robot

Un robot se conforma de múltiples elementos, la acción conjunta de todos ellos permite al robot operar correctamente y llevar a cabo las tareas encomendadas, los elementos indispensables de un robot pueden ser agrupados dentro de tres subsistemas (Fig. 6) como propone Subir Kumar.[3]

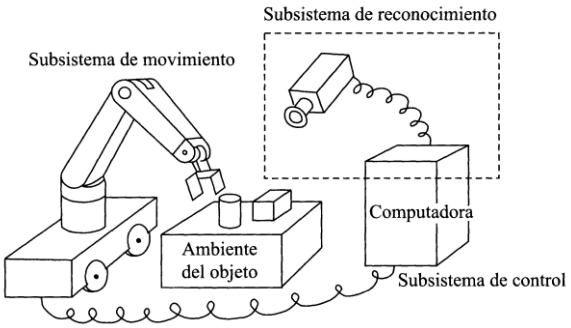


Fig. 6 Subsistemas de robots

Los subsistemas que se proponen son:

Subsistema de Movimiento
Subsistema de Reconocimiento
Subsistema de Control

Subsistema de Movimiento

El subsistema de movimiento comprende el cuerpo físico del robot (llamado manipulador), el cual se conforma de eslabones conectados por articulaciones (pares cinemáticos más comunes: prismático y revoluta); mediante actuadores (motores eléctricos, pistones hidráulicos o neumáticos) acoplados a una transmisión (banda, cadena, engranes, eslabones) se provee de movilidad, capacidad de carga y rigidez al manipulador, al accionarlos de forma conjunta se logra mover correctamente al efector final (mano, garra mecánica, herramienta de soldadura o corte), la selección de componentes y materiales del robot está en función de las características de aplicación y condiciones del entorno de trabajo a las que estará sometido (Fig. 7)[3].

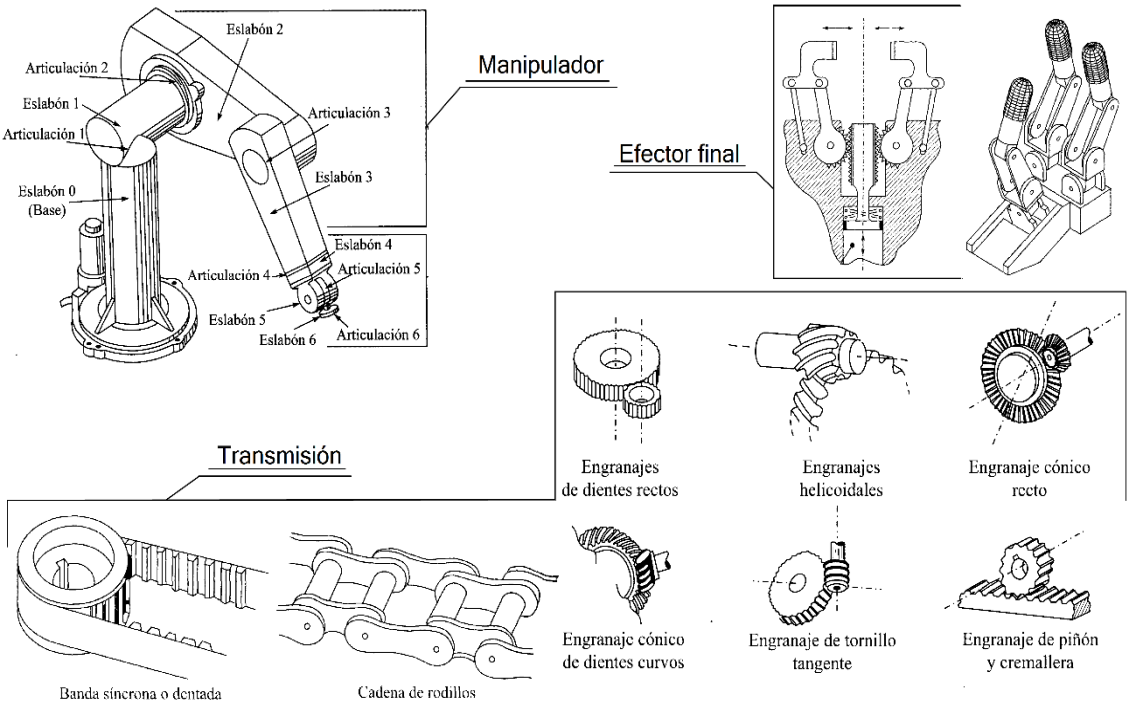


Fig. 7 Componentes del Subsistema de Movimiento

Clasificación de manipuladores industriales

Dentro de la categoría de manipuladores industriales existen tres tipos de estructuras, las cuales son seriales, paralelos e híbridos

Para continuar es necesario definir que, una cadena cinemática es un ensamble de eslabones conectados por articulaciones uno seguido de otro.

La estructura serial se conforma por una cadena cinemática de lazo abierto; lo cual significa que, partiendo del eslabón conectado a la base fija del manipulador, existe un único camino para llegar al extremo final de la cadena (efector final).[4, 5]

En la estructura paralela la cadena cinemática forma uno o más lazos cerrados, lo cual significa que, todos los eslabones están conectados por al menos dos caminos distintos, de modo que es posible recorrer las cadenas del manipulador que se entrelazan, partiendo de un eslabón y regresando al mismo por otro camino; las cadenas cinemáticas suelen ser simétricas y coinciden con el efector final del manipulador. [4, 5]

La estructura hibrida comprende una combinación de las estructuras serial y paralela.

La Fig. 8 muestra a la izquierda una estructura de tipo serial con 6GDL, en medio una estructura de tipo paralelo con 3GDL y a la derecha una estructura de tipo híbrida con 5GDL. [4]

Los grados de libertad (GDL) de un manipulador indica la cantidad de variables linealmente independientes con las cuales se define la configuración del manipulador.

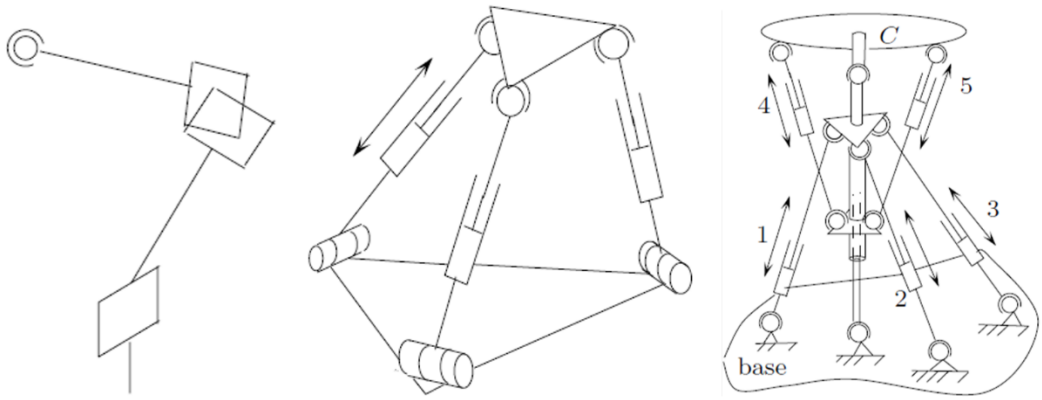


Fig. 8 Estructuras de manipuladores

Los robots paralelos, industrialmente se implementan para asistir tareas dentro de procesos repetitivos, monótonos, continuos, que requieren de un tiempo de trabajo prolongado y por lo general no manipulan cargas pesadas, ni objetos muy grandes.[4]

Existen dos grupos de estructuras paralelas, planar y espacial; los primeros considerados plataformas móviles con 3GDL, suelen estar compuestos por articulaciones rotacionales permitiendo traslaciones de la plataforma móvil a lo largo de *X*, *Y*, además de contar con un giro alrededor del eje *Z* del sistema de la base móvil con ángulo ϑ; la Fig. 9 muestra los distintos tipos de articulaciones que pueden integrar un manipulador planar. [4]

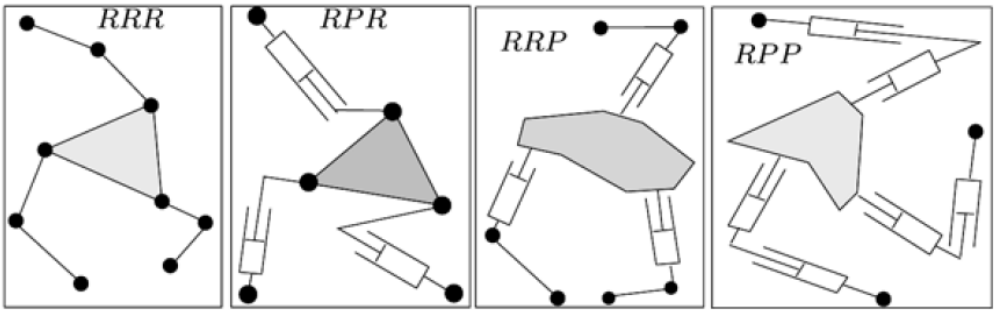


Fig. 9 Arquitecturas distintas del grupo paralelo planar

El grupo de los manipuladores paralelos espaciales cuentan con 3, 4, 5 y 6 grados de libertad, los cuales comúnmente corresponden al número de brazos con los que cuenta el manipulador; a continuación, se muestran algunos de los manipuladores paralelos emblemáticos.

La configuración paralela tipo *Delta* es un *manipulador paralelo espacial*, consta de tres cadenas cinemáticas independientes (*brazos*) idénticos a una disposición de 120 grados entre sí, los cuales vinculan la base superior fija con la plataforma inferior móvil. La interacción entre las posiciones angulares de los brazos determina la posición puntual del efector final dentro del espacio cartesiano tridimensional [6]; por lo general cada cadena cinemática dispone de un actuador, el volumen o área de trabajo suele ser amplio, los robots paralelos se caracterizan por poseer gran rigidez, velocidad y capacidad de carga en comparación con su tamaño. [4]

La Fig. 10 muestra un robot industrial paralelo tipo delta nombrado *FlexPicker*, fabricado por la compañía *ABB*, el cual consta de 4 brazos, 3 de ellos externos con arquitectura RSS que permiten al efector final efectuar traslaciones en el espacio y uno interno que permite una rotación del efector final en el eje Z, orientado de manera perpendicular a la base móvil del manipulador; existen distintos modelos de la familia *IRB 360*, a continuación se detallan las características generales de dicha gama de robots, obtenidas de su hoja de especificaciones; cuentan con un diámetro del área de trabajo que va desde 800 hasta 1600 mm, además cuentan con una altura de trabajo que va de 200mm hasta 305 mm; la capacidad de carga con la que cuentan es de hasta 1kg o 8 kg según el modelo, los cuales operan a velocidades altas de hasta 1400mm/s, logrando mantener una buena precisión; el robot pesa 140 kg, utiliza una alimentación de corriente alterna de 200 hasta 600 Volts a 60 Hz; en un ciclo con una carga de 1kg consume 0.447 kW. Las aplicaciones industriales de este robot son trabajos denominados *pick and place* en montaje y embalaje.

Fig. 10 Manipulador FlexPicker de la compañía ABB

Objetivo

Generar un manipulador paralelo tipo delta, que logre ser un prototipo piloto funcional, resolver la cinemática inversa de este, diseñar su control y la electrónica necesaria, con el fin de aplicar y comprobar los conocimientos de métodos teóricos y prácticos obtenidos a lo largo de la carrera.

Objetivos particulares

i. Modelar la cinemática inversa del robot paralelo tipo delta
ii. Construir un manipulador paralelo tipo delta
iii. Generar el control de los actuadores del robot
iv. Diseñar tarjetas electrónicas

Justificación

Presentar un sistema integral básico, con el fin de acercar la tecnología al interesado, consolidando una herramienta sencilla y estructurada, apoyada tanto de métodos teóricos como prácticos.

Metas

1.- Análisis Cinemático de la posición

a) Generar diseño mecánico de la plataforma en software de CAD

b) Realizar análisis de la posición haciendo uso de Transformaciones Homogéneas

2.- Simulación y selección de componentes

a) Simular el mecanismo e implementar ecuaciones en el algoritmo de control

b) Manufacturar el manipulador y caracterizar sus elementos

c) Seleccionar actuadores, sensores y el controlador

3.- Caracterización, algoritmos e integración

a) Caracterizar los componentes electrónicos

b) Adaptar algoritmos de control, programación y elaboración de hardware necesario

c) Integrar el sistema y generar una HMI

4.- Pruebas y Resultados

Alcances

El siguiente trabajo exige diseñar un manipulador paralelo de configuración delta, posteriormente realizar el análisis cinemático de la posición, con el objetivo de obtener las ecuaciones que rigen el movimiento del efector final de la plataforma; una forma eficiente de validar dichas ecuaciones es simular la plataforma e implementar un algoritmo de control, apoyado de una interfaz gráfica de usuario.

Al validar las ecuaciones de movimiento, se podrá proceder a construir el prototipo con materiales económicos, procesos replicables y de fácil acceso.

Las propiedades de la plataforma definen los actuadores y sensores a seleccionar, obligando a adaptar el sistema de control a dichos elementos, también se deben considerar las características del micro-controlador en el cual se pretende programar dicho algoritmo de control, debe ser capaz de calcular y emitir las señales requeridas para posicionar correctamente el efector final de la plataforma.

Se contempla usar un módulo externo programado con una interfaz gráfica, por la cual el usuario logre proveer de coordenadas e instrucciones al controlador del sistema robótico.

1 GENERALIDADES

1.1 Antecedentes

El primer manipulador paralelo esférico fue propuesto por Gwinnet en 1931, la plataforma estaba dirigida a la industria cinematográfica como simulador de movimiento (Fig. 11), por desgracia nunca fue construida (le fue otorgada la patente US1789680).

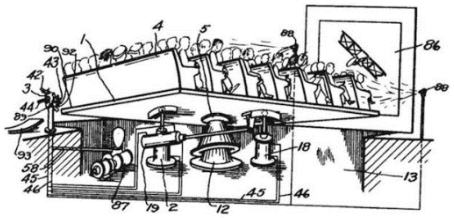


Fig. 11 Propuesta de simulador por Gwinnet

Años después se conocería la primera aplicación industrial de un robot paralelo, Pollard en 1940 diseñó un manipulador de cinco barras (Fig. 12), el cual tenía la finalidad de pintar automáticamente vehículos usando pintura en aerosol (le fue otorgada la patente US2286571) [7].

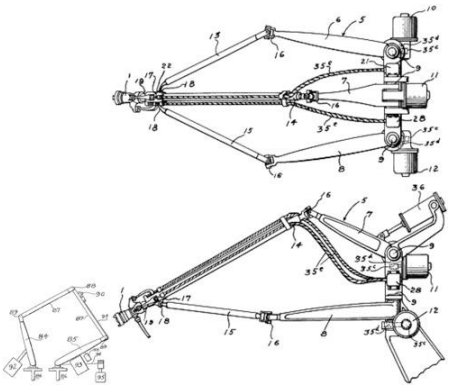


Fig. 12 Manipulador de cinco barras diseñado por Pollard

En 1955 se terminó la construcción de la "Máquina de prueba de neumáticos universal" (Fig. 13) atribuida al doctor Eric Gough, el hexápodo de Gough simulaba cargas a las que se enfrentan los neumáticos de aeronaves durante el aterrizaje, con la finalidad de conocer las propiedades de los neumáticos, lo hacía sometiéndolos a cargas combinadas aplicando fuerza en cualquier dirección; el invento publicado por Gough es uno de los manipuladores paralelos más reconocidos [4, 8].

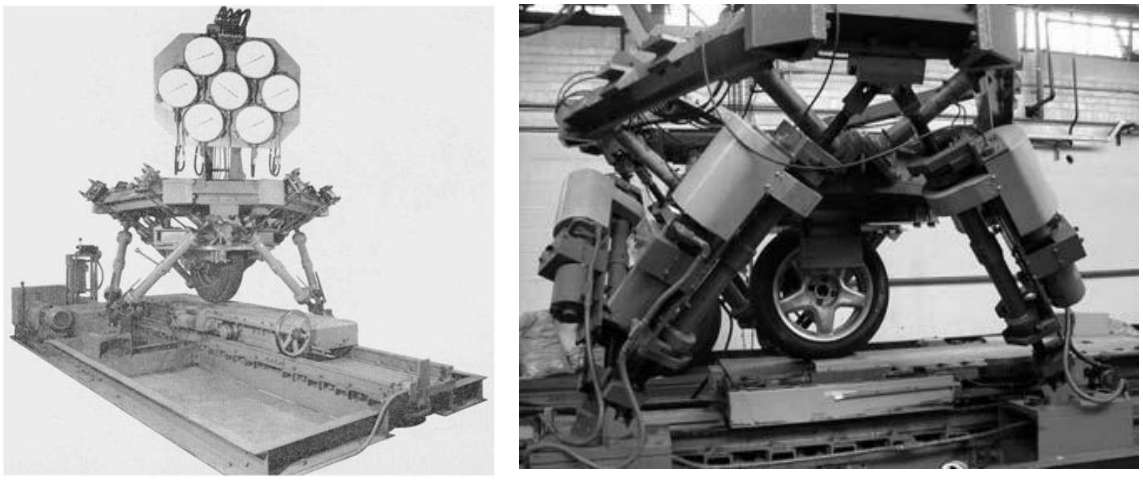


Fig. 13 Máquina de Prueba de "Neumáticos Universal" y "Universal Rig" diseñados por Gough

Stewart en 1965 publicó un artículo teórico en el cual describió una plataforma de seis grados de libertad en el que incluyo análisis de movilidad y singularidad, la plataforma analizada tenia propósitos que van desde aplicaciones espaciales hasta simuladores de vuelo, también menciona la posibilidad de desarrollar nuevos tipos de máquinas herramienta; el artículo describe la plataforma con seis extremidades extensibles (gatos hidráulicos como actuadores), tres de ellos (miembros primarios) proporcionaban movimiento conectando la plataforma móvil con la fija, los tres restantes (miembros secundarios) proporcionaban orientación conectando la base fija con el extremo de los miembros primarios (Fig. 14). Dicho artículo ha sido de los más reconocidos académicamente [9].

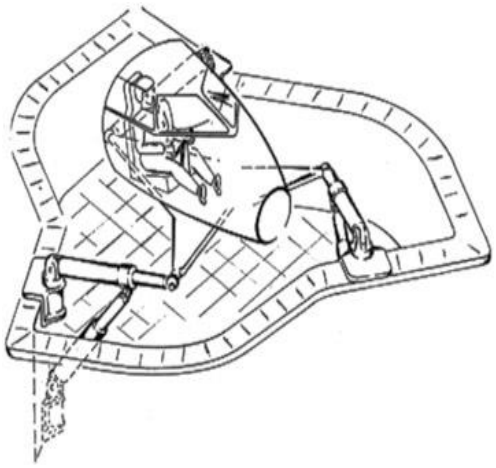


Fig. 14 Propuesta de plataforma de Stewart

Mecanismos basados en hexápodo octaédrico son conocidos universalmente como plataformas de Gough-Stewart, ninguno tuvo el interés de obtener una patente por dicha invención, En 1967 a Klaus Cappel le fue otorgada la primera patente referente al hexápodo octaédrico (Fig. 15) y fue el primer simulador de vuelo comercialmente disponible [7].

Fig. 15 Simulador de vuelo de Klaus Cappel

En la década de los 80's (1990, 1991) Reymond Clavel diseñó el renombrado *DELTA4*, robot paralelo de 4 grados de libertad que cuenta con un amplio espacio de trabajo, fue pensado para asistir trabajos en industrias de electrónica, alimentos, farmacéutica, han sido utilizados industrialmente en trabajos de pick and place y medicamente como dispositivos de asistencia quirúrgica; el robot propuesto por Clavel consiste en cuatro brazos, tres de ellos externos que proporcionan la posición del efector final y un brazo central interno que proporciona la orientación del mismo mediante un giro (Fig. 16); dicha configuración ha logrado aceleraciones de hasta 50 G en ambientes experimentales y dentro de ambientes industriales 12 G; a Clavel le fue otorgada la patente de su invención en 1990, le fue entregado el Golden Robot Award, patrocinado por la empresa ABB Flexible Automation y algunas empresas se hicieron de los permisos para comercializar su robot [6].

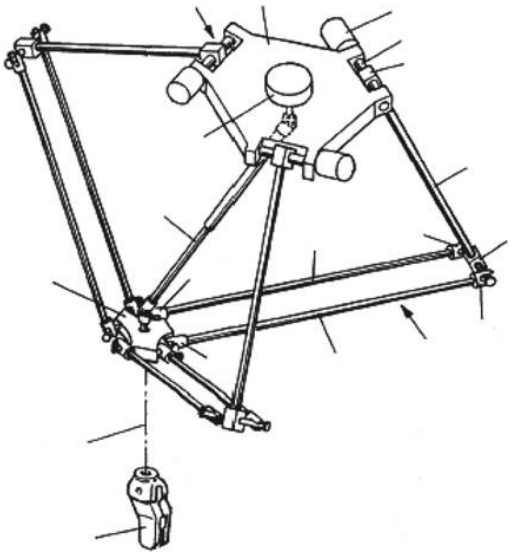


Fig. 16 Robot DELTA4 patentado por Clavel

Pierrot (1999) introduce el manipulador paralelo espacial nombrado *H4* (Fig. 17)con la finalidad de ejecutar tareas de pick and place a gran velocidad, comparándolo con la arquitectura *Delta* de Clavel, se conservó el número de brazos y GDL, la diferencia radica en la distribución de los brazos, ya que el *H4* omite el brazo central de la configuración *Delta* e introduce el cuarto brazo de forma externa; tiempo después Adept Technology© lanzaría el robot Quattro capaz de ejecutar 240 ciclos por minuto, considerado a la fecha como el robot más rápido dentro de las tareas de pick and place.

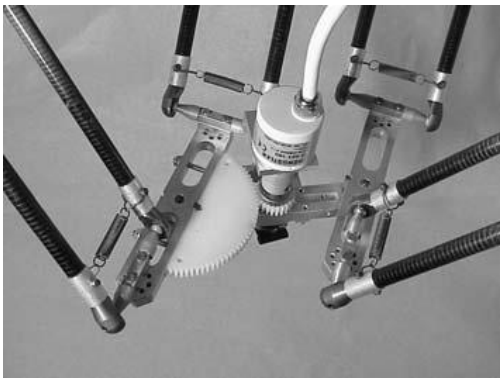


Fig. 17 Efector final del H4 diseñado por Pierrot

Continuando con la experimentación de configuraciones, en la década de los 90`s también se desarrollaron manipuladores paralelos esféricos, uno de los más populares es el nombrado Ojo de águila diseñado por Gosselin (Fig. 18), en el cual sus ejes convergen en un punto, convirtiéndose en su centro de rotación; tenía como finalidad la rápida orientación de una cámara; dicho manipulador registró altas aceleraciones angulares de 20,000°/s². Posteriormente Kim y Tsai desarrollaron el manipulador paralelo cartesiano con 3 GDL desacoplados, mecanismo espacial basado en cadenas seriales utilizando actuadores lineales, este tipo de manipuladores poseen ventajas en cuanto a la simplicidad de la cinemática y el control [4, 7].

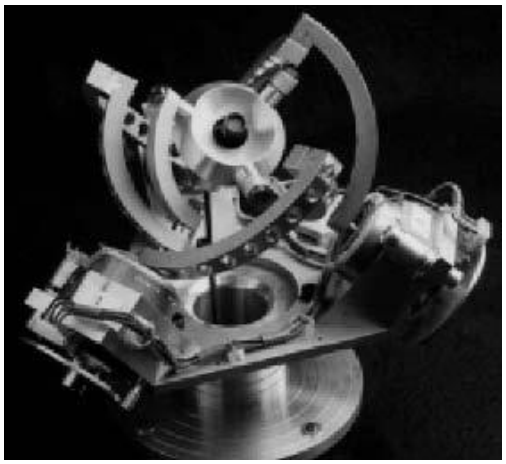


Fig. 18 Manipulador esférico diseñado por Gosselin

Como se ha visto los manipuladores paralelos ofrecen grandes ventajas y se espera que puedan superar los límites de espacio de trabajo y manipulación de cargas que hasta ahora se han observado.

1.2 Sistemas embebidos y su aplicación en un robot paralelo

Utilizando como referencia [10-12] se pueden obtener las siguientes definiciones.

Un sistema embebido (SE) es un dispositivo electrónico de propósito específico dedicado al procesamiento de información, el cual está integrado a un sistema mecánico o eléctrico más grande.

Los sistemas embebidos actualmente están integrados en automóviles (ADAS - Advanced Driver Assistance System), trenes, aviones, equipos de telecomunicaciones, control, médicos y de fabricación; su implementación está dirigida a controlar procesos físicos con el uso de unidades de micro controlador (UMC), los cuales suelen tener periféricos (hardware) o interfaces de usuario como botones, palancas, volantes, pedales o GUI, de modo que, las tareas que ejecuta la UMC sean invisibles para el usuario o que apenas pueda reconocer el procesamiento que se lleva a cabo dentro del mismo sistema; el software especializado que está programado dentro de la UMC posibilita la ejecución de las tareas del SE, el cual se conoce como software embebido.

El software embebido es un algoritmo integrado con modelos y restricciones definidos por las características del proceso físico a los cuales está dirigida la implementación del SE.

Debido a que la estructura y aplicación del software embebido se enfoca en administrar tareas específicas, es posible escribir manualmente el código, teniendo en cuenta el flujo de datos para optimizar la ejecución de los procesos; la UMC debe contar con características que le permitan al software operar sin complicaciones o adaptar este al hardware de manera eficiente, manteniendo un balance de la relación costo/rendimiento.

El término Sistema Ciber Físico (CPS) enfatiza el vínculo entre el sistema embebido y el entorno al que está integrado, ya que la combinación de hardware (sensores y actuadores) y software (programa) del cual dispone el SE permite conectarse e interactuar de manera continua con el mundo físico.

Un claro ejemplo de Sistemas Embebidos/Ciber Físicos son los Robots, los cuales introducen aspectos mecánicos importantes del *sistema de movimiento* dentro del *sistema de control*, además el controlador apoyado del software embebido logra procesar la información adquirida por el hardware que compone al *sistema de reconocimiento* para procesarla y actuar con base en los modelos y requerimientos del sistema físico, permitiendo realizar una tarea específica.

Las características de los Sistemas Embebidos varían respecto a la tarea que desempeñan, a pesar de ello suelen contar con algunas de las siguientes:

- Dimensiones pequeñas
- Suministro de energía propio.
- Bajo consumo de energía
- Capacidad de procesamiento y memoria reducida respecto a sistemas de propósito general, computadoras personales (PC)

Para desarrollar este trabajo de tesis hay que considerar los subsistemas de movimiento, reconocimiento y de control que, permitirán automatizar el movimiento del manipulador de manera controlada.

1.2.1 Subsistema de Reconocimiento

El subsistema de reconocimiento se encarga de recopilar información con la finalidad de conocer el estado del robot, de los objetos a manipular o del entorno de trabajo, según sea el enfoque y el tipo de sensores utilizados, ya que los sensores pueden medir distintas magnitudes físicas según su construcción; el subsistema permite el acondicionamiento de la señal de los sensores, mediante filtros y convertidores analógico-digital (ADC), los cuales interpretan los niveles de voltaje de la señal analógica de los sensores para comunicar dicha información de manera digital al controlador del robot; la conversión de una señal analógica se muestra en Fig. 19.

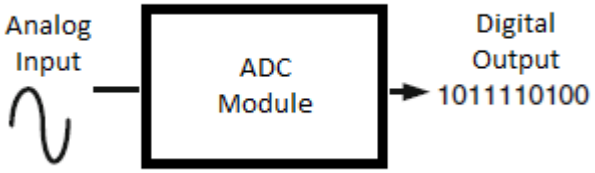


Fig. 19 Diagrama de digitalización de señal provista por un sensor

1.2.2 Subsistema de Control

El subsistema de control se conforma de un controlador encargado de recibir la información del subsistema de reconocimiento, procesar la información y emitir señales digitales específicas; las señales de salida del controlador están se deben acoplar a un convertidor digital - analógico (DAC), de modo que traducen los pulsos digitales emitidos por el controlador a señales analógicas equivalentes, las cuales son de baja intensidad, por lo que es necesario utilizar una etapa de potencia con la finalidad de amplificar las señales analógicas, permitiendo accionar los actuadores del manipulador, logrando un correcto posicionamiento del manipulador; la figura Fig. 20 muestra un módulo de control de la empresa ABB.

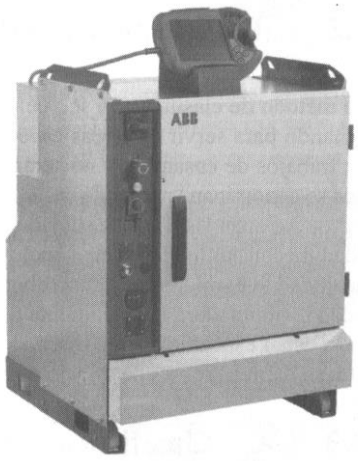


Fig. 20 Módulo de control de ABB

De acuerdo con lo anterior, el subsistema de control es el conjunto que integra tanto al subsistema de movimiento como al subsistema de reconocimiento, su diseño e implementación deben estar enfocados a permitir la interacción de los elementos de cada subsistema; de acuerdo con las ventajas que ofrecen actualmente los micro controladores, el controlador suele incluir uno o más convertidores analógico digital (ADC), por lo que el subsistema de reconocimiento no requiere ser muy sofisticado e incluso se comprende únicamente por sensores (hardware) que envían al controlador las señales adquiridas; el hardware (sensores y actuadores) debe responder a las necesidades del subsistema de movimiento y estar dentro del rango de operación del entorno, manteniendo un margen de seguridad.

Es de suma importancia el subsistema de movimiento ya que dentro del controlador deben programarse los modelos físicos del entorno, en este trabajo dichos modelos son dados por las ecuaciones de posición angular de cada articulación actuada, las cuales se obtienen al resolver el problema de la cinemática inversa; con el objetivo de lograr posicionar la plataforma móvil sobre algún punto definido por el usuario mediante coordenadas, se debe diseñar el software de procesamiento implementando un control de lazo cerrado como se muestra en [13], el cual mueve de manera controlada al sistema robótico hasta alcanzar el objetivo calculado, para ello, el controlador calcula la diferencia (error) entre el valor de entrada derivado del modelo físico (valor deseado) y el valor de realimentación adquirido de los sensores (valor medido); dicha diferencia corrige y proporciona una salida apropiada a los actuadores; el proceso se realimenta y ejecuta continuamente hasta alcanzar el valor deseado.

Dentro de este trabajo, el diagrama del sistema de control implementado en el robot delta se muestra en Fig. 21; el diagrama muestra el flujo de información del proceso de control actuando en el sistema robótico, utilizando la información de los sensores como realimentación de la variable física (posición, velocidad) y colocando los modelos o trayectoria del manipulador como objetivo de la configuración de entrada (deseada)**.**

Por lo anteriormente expuesto es necesario obtener las ecuaciones de posición que rigen el movimiento del robot.

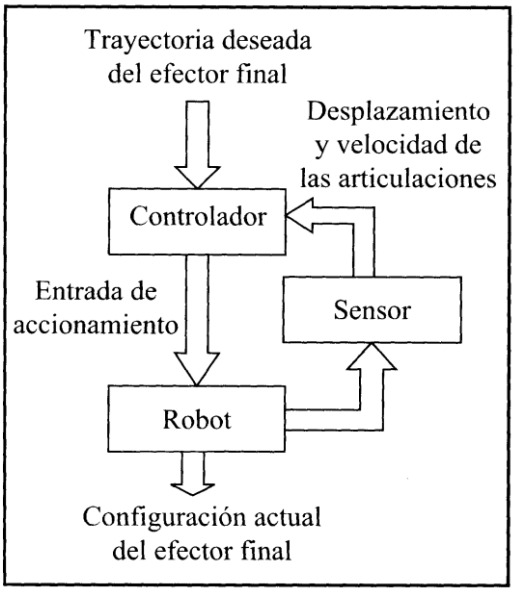


Fig. 21 Diagrama de control del manipulador

2 ANÁLISIS DE LA POSICIÓN POR CINEMÁTICA INVERSA

2.1 Marco Teórico

2.1.1 Cinemática

La cinemática estudia el movimiento de un sistema mecánico sin tomar en cuenta los componentes de fuerza o torque que causan dicho movimiento, considera la geometría de los elementos, sus restricciones, y la localización de los cuerpos en el espacio (posición y orientación) [2, 5, 14].

El manipulador (cuerpo físico del robot) se conforma por eslabones conectados mediante articulaciones, también llamados pares cinemáticos o juntas; el movimiento relativo de los eslabones de un manipulador se permite de acuerdo con las restricciones físicas definidas por la geometría de cada par cinemático [3, 5, 14] la Fig. 22 muestra los seis distintos tipos de articulaciones y su respectivo grado de libertad (GDL); generalmente son utilizadas únicamente las juntas de rotación o revoluta y prismáticas. [5, 14]

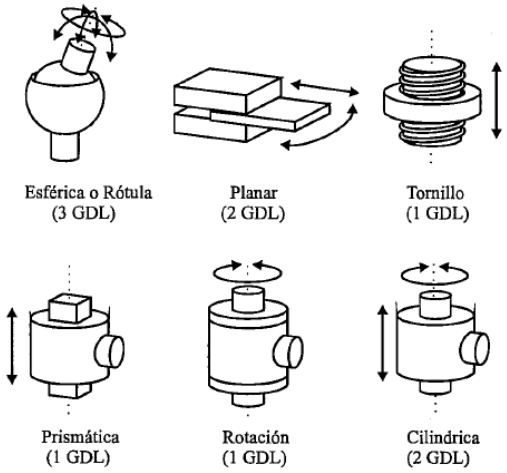


Fig. 22 Tipos de articulaciones

La cadena cinemática de un manipulador se compone por articulaciones y eslabones interconectados; un manipulador paralelo consta de múltiples cadenas cinemáticas, el tipo de juntas empleadas en cada cadena cinemática define la arquitectura del manipulador y a su vez define la configuración de posición/orientación que es capaz de adoptar el efector final del manipulador, ya que el efecto de movimiento que provee cada tipo de articulación es distinto como se describe en [3, 5].

Para un análisis más práctico de la cinemática de un manipulador, cada eslabón se considera como un cuerpo rígido, lo cual significa que sus dimensiones son constantes en cualquier instante; cada eslabón debe tener asociado un sistema de referencia cartesiano derecho *N* con los vectores unitarios i_n, j_n, k_n; el espacio tridimensional tiene su origen representado por el sistema de coordenadas inercial, fijo o universal *O* (con vectores unitarios i_0, j_0, k_0); para definir la posición de un eslabón dentro del espacio tridimensional, el origen del sistema de referencia asociado al eslabón debe ser referido al origen del sistema inercial, representando los parámetros de traslación del eslabón con las componentes x, y, z del vector de posición *p* de dimensión 3x1, logrando definir en el espacio cartesiano tridimensional la posición del cuerpo rígido, sin considerar la orientación del mismo como se muestra en la Fig. 23 [5, 14, 15].

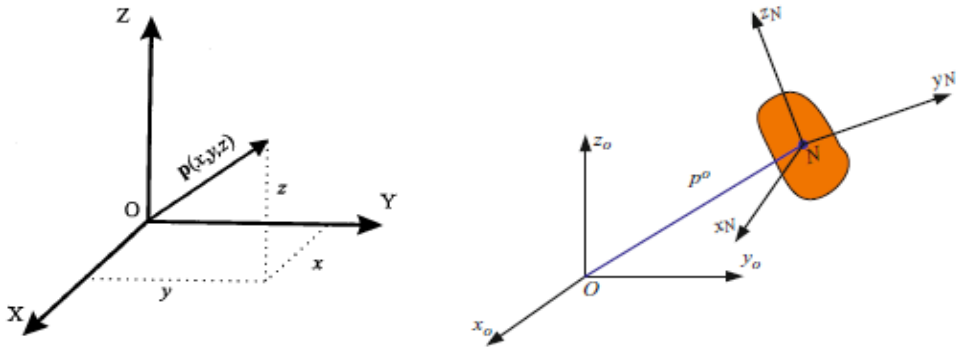


Fig. 23 Sistema cartesiano inercial y de referencia del cuerpo rígido, definido por el vector p

Las matrices de rotación con dimensión 3x3 permiten conocer la orientación de un sistema de referencia rotado un ángulo, respecto a un eje especifico del sistema de referencia anterior; para conocer la orientación de un cuerpo rígido en el espacio tridimensional del manipulador, es necesario considerar la relación con los sistemas de referencia anteriores por los cuales se encuentra definido el sistema del cuerpo rígido; la Fig. 24 muestra el efecto que genera en los sistemas de referencia la implementación de las matrices básicas de rotación, las cuales se obtienen por el método de cosenos directores [5, 14, 15].

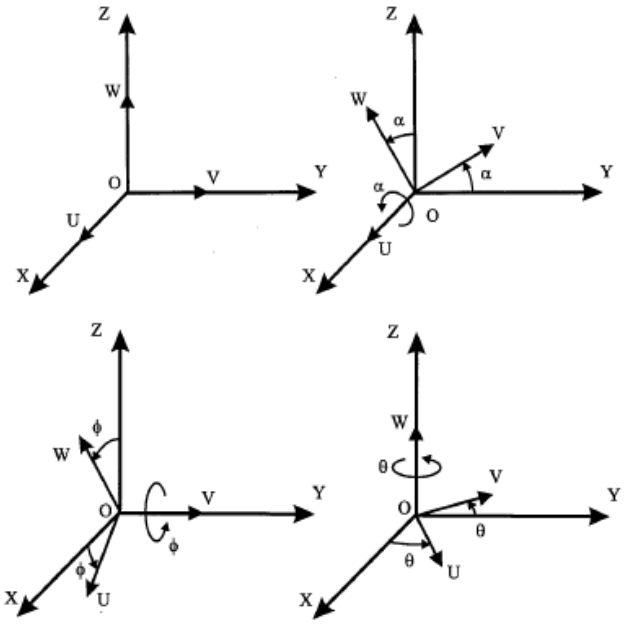


Fig. 24 Orientación del sistema de referencia del cuerpo rígido respecto al sistema fijo

La forma de la matriz de rotación global con dimensión 3x3 que define completamente la orientación de un cuerpo rígido, cambia según la secuencia de rotaciones seguida debido a que las matrices de rotación no son conmutativas; la matriz que resulta de rotar sobre los tres ejes de un sistema de referencia, se conforma de tres parámetros ($\alpha, \varphi, \vartheta$) [5, 14].

2.2 Matrices de Transformación Homogénea

Los parámetros de posición y orientación describen la localización del cuerpo rígido; las matrices de transformación homogénea con dimensión 4x4 involucran las operaciones de traslación y rotación en una sola matriz, la cual se compone de 4 elementos, *Matriz de Rotación* con dimensiones 3x3; *Vector de Traslación* con dimensión 3x1; *Transformación de perspectiva* con dimensión 1x3; *Factor de Escala*, este último elemento generalmente se considera con valor unitario y la transformación de perspectiva nula; lo anterior descrito se muestra en la Fig. 25; las matrices de transformación homogénea permiten que un vector definido en un sistema de referencia dado, pueda ser representado en otro sistema de referencia [2, 5, 14].

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{p}_{3x1} \\ \mathbf{f}_{1x3} & \mathbf{w}_{1x1} \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ \text{Perspectiva} & \text{Escalado} \end{bmatrix}$$

Forma general

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{p}_{3x1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ 0 & 1 \end{bmatrix}$$

Asignación de valores

Fig. 25 Componentes de la matriz de transformación homogénea

Las matrices de transformación homogénea de traslación pura y rotación pura en cada uno de los ejes de un sistema de referencia (X, Y, Z) se muestran en la Fig. 26.

Traslación

$$\mathbf{T}_{z1}(x) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{z2}(y) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{z3}(z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotación

$$\mathbf{T}_{z4}(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta_x & -s\theta_x & 0 \\ 0 & s\theta_x & c\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{z5}(\theta_y) = \begin{bmatrix} c\theta_y & 0 & s\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -s\theta_y & 0 & c\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{z6}(\theta_z) = \begin{bmatrix} c\theta_z & -s\theta_z & 0 & 0 \\ s\theta_z & c\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 26 Matrices de transformación homogénea

2.3 Diseño Mecánico del Robot

Se utilizó el algoritmo de diseño que muestra la Fig. 27 para seleccionar una arquitectura y proponer un diseño correspondiente a la parte mecánica del robot; fue necesario analizar distintas arquitecturas de manipuladores paralelos como RUS, RUU, RR-(4R)-R, las cuales se diseñaron utilizando un software de CAD; al analizar el movimiento de los ensambles de cada propuesta, se consideraron los mejores aspectos de cada manipulador y buscando obtener el diseño con la mayor libertad de movimiento posible (además de reducir el número de piezas), se realizaron modificaciones a la geometría de los eslabones considerando piezas y medidas disponibles en el mercado a un costo accesible; ya que el objetivo del robot es ejecutar tareas con fines demostrativos, la inversión económica debe responder a las necesidades y funciones de operación del mismo [4, 5, 16].

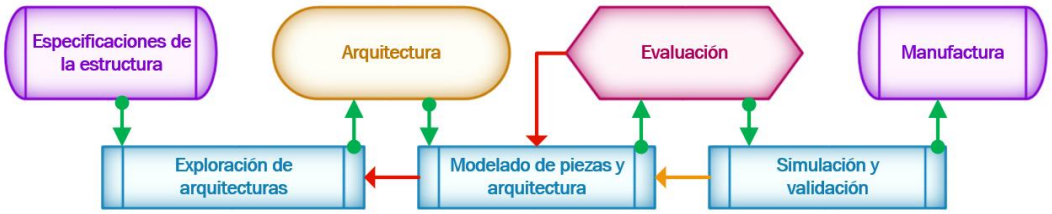


Fig. 27 Algoritmo de diseño del manipulador paralelo

2.3.1 Arquitectura del manipulador RR-(4R)-R

La arquitectura del manipulador conocido como *University of Maryland Manipulator* empleada en el presente trabajo, se describe en [5, 6] como dos plataformas o bases triangulares equiláteras fija y móvil, las cuales se conectan por tres cadenas paralelas idénticas RR-(4R)-R, separadas 120 grados entre sí; cada cadena cinemática se compone de dos segmentos principales *Brazo* y *Antebrazo*, este último es un paralelogramo de cuatro barras conectadas por articulaciones de revoluta (4R), el cual se encuentra conectado al eslabón *Brazo* y a la *Base Móvil* del manipulador; a las articulaciones consideradas como activas se les acopla un actuador, los cuales se montan directamente a la *Base fija*. Esta arquitectura se describe como un modelo simplificado, variante del delta original introducido por Clavel, la Fig. 28 muestra un diagrama simplificado del que se muestra en [6], se puede apreciar que, en medio de la *Base fija* se encuentra el origen del sistema de coordenadas fijo o inercial *O* con vectores unitarios (i_0, j_0, k_0); el manipulador permite traslaciones de la *Base móvil* en el espacio cartesiano tridimensional definido por los ejes X, Y, Z, utilizando únicamente articulaciones de tipo rotacional; para referirnos a cada articulación, como se muestra en el esquema se sustituyó la notación q_n por la inicial del tipo de articulación, en este caso particular todas las articulaciones son rotacionales, por lo que se nombran como *R1, R2, R3, R4, R5*.

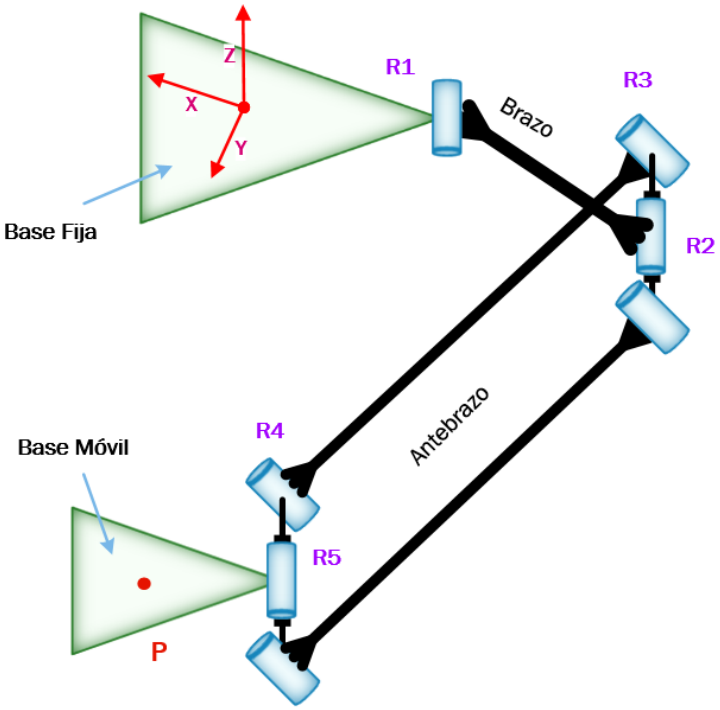


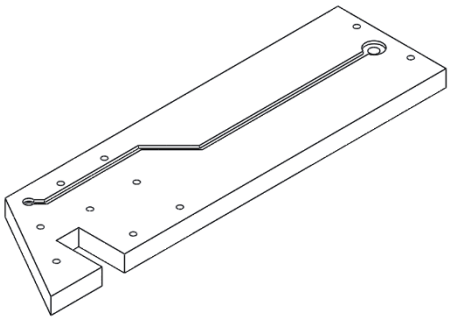
Fig. 28 Arquitectura del manipulador

2.3.2 Diseño mecánico de las piezas

A continuación, se muestran las piezas que conforman al manipulador paralelo con arquitectura 3 RR-(4R)-R previamente señalada.

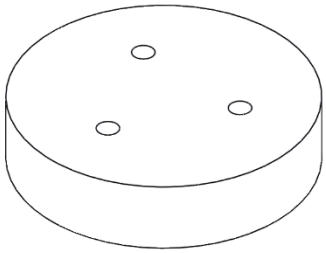
Base A

Es una de las piezas que constituyen a la base fija del manipulador, en las cuales se atornillan las piezas *soporte de brazo* que prevén de orientación y firmeza a cada uno de los brazos independientes del manipulador.

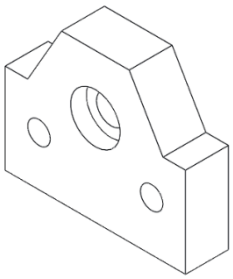


Unión de base fija

Esta pieza permite ensamblar la base fija del manipulador, ya que en ella se atornillan las tres piezas tipo *Base A* requeridas para generar la estructura de la base fija del manipulador.

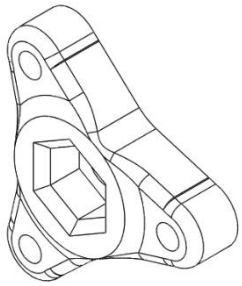


Soporte de brazo



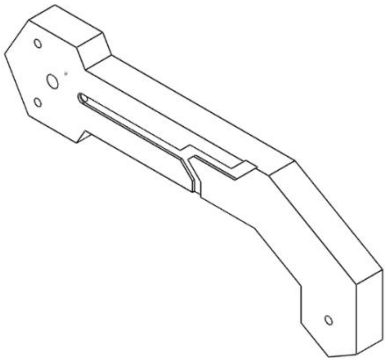
Esta pieza brinda soporte y permite el movimiento de la pieza *Brazo*, se ancla a la *Base A* formando parte de la base fija del manipulador, en esta pieza surge el primer componente del par cinemático de tipo rotacional correspondiente a *R1*, la cual se muestra en esquema de la arquitectura.

Unión de brazo



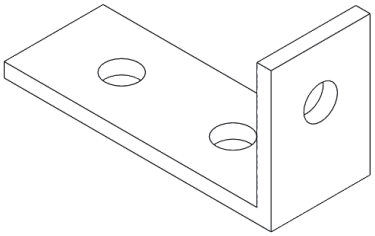
Esta pieza se ensambla al eslabón *Brazo* utilizando los tres puntos externos de apoyo, coloca la pieza *Brazo* sobre el eje de acción del *Soporte de brazo*, siendo la segunda componente del par cinemático de tipo rotacional *R1*.

Brazo



Esta pieza conforma el primer eslabón de cada cadena cinemática del manipulador, en sus extremos deben atravesar los ejes de acción de las articulaciones rotacionales *R1* y *R2* como se muestra en esquema de la arquitectura.

Junta



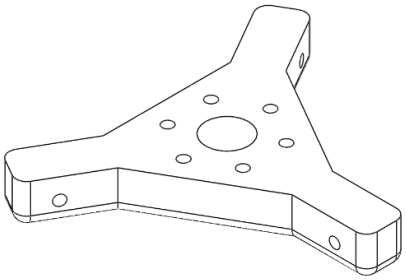
Este tipo de pieza se ensambla a los extremos del eslabón *Antebrazo* y según sea el extremo, se ensambla a las piezas *Brazo* o *Base móvil*, formando las articulaciones (*R2* y *R3*) o (*R4* y *R5*); de acuerdo a la geometría de esta pieza los ejes de acción son perpendiculares entre sí y coincidentes en un punto, sirviendo como una junta universal según [5].

Antebrazo

A este eslabón se ensamblan dos piezas *Junta*, formando los pares cinemático-rotacionales *R3* y *R4* en su respectivo extremo; el paralelogramo -(4R)- de cada cadena se forma con dos piezas de antebrazo y sus respectivos ensambles.

Base Móvil

Esta pieza une las tres cadenas cinemáticas del manipulador ensamblado, la posición de esta pieza depende de la orientación de las articulaciones activas de las tres cadenas paralelas, sus ejes de acción corresponden a *R5* de cada cadena.



Las piezas anteriormente descritas comprenden al manipulador, la Fig. 29 muestra su ensamble; el diseño seleccionado tiene un rango de movimiento de ± 50 grados, superior al de las articulaciones universales y esféricas convencionales, las cuales cuentan normalmente con ±15 grados y algunas especiales con ± 30 o ± 45 grados [4], la Fig. 30 muestra los límites de las articulaciones rotacionales implementadas en el manipulador, correspondientes a *R3* y *R4* según el esquema de la arquitectura, mientras que la Fig. 31 muestra los alcances del manipulador.

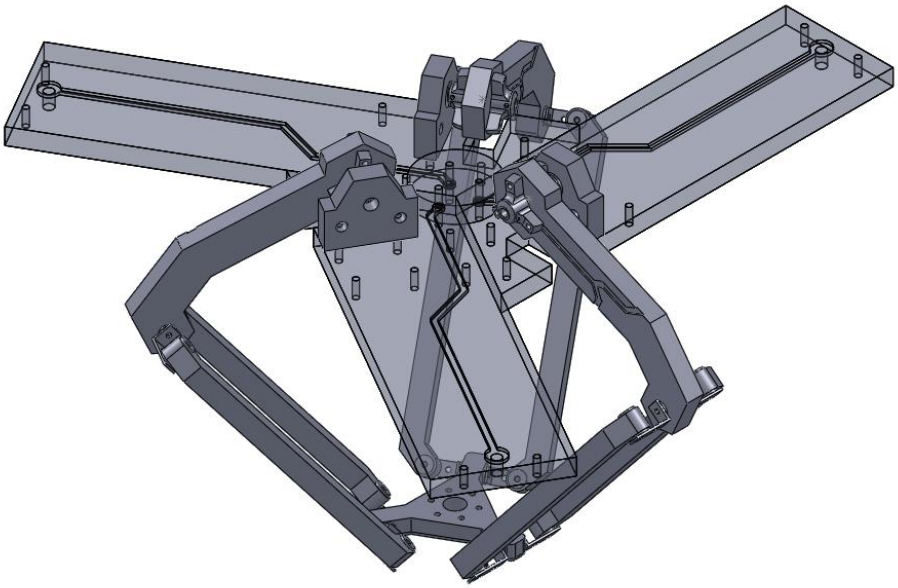


Fig. 29 Ensamble del manipulador Delta

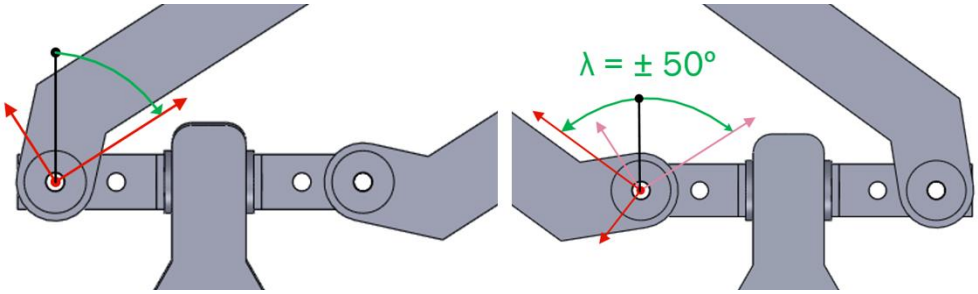


Fig. 30 Rango de movimiento máximo de articulaciones rotacionales

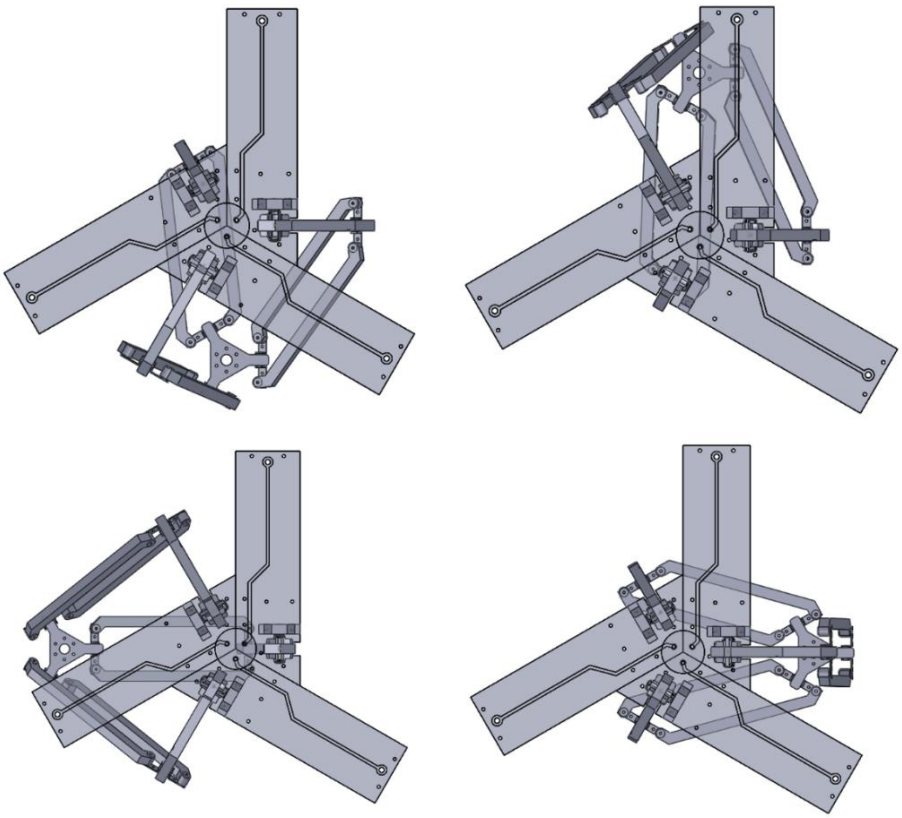


Fig. 31 Desplazamientos máximos del manipulador

2.4 Cinemática del Manipulador con MTH

Dentro del presente trabajo, las matrices de transformación homogénea (MTH) son empleadas para modelar la cinemática del manipulador con arquitectura 3 RR-(4R)-R, lo cual se logra obteniendo la secuencia de transformaciones que describen la relación entre los elementos de una de las cadenas cinemáticas del manipulador, ya que las tres cadenas paralelas que lo componen son simétricas; siguiendo la geometría y relaciones del ensamble, se selecciona el tipo de transformación (rotación / traslación), cada transformación genera un nuevo sistema de referencia *local* el cual está asociado a un eslabón con origen $OXYZ_n$, el cual describe las rotaciones o traslaciones relativas al sistema de referencia anterior; en el caso de rotación pura el origen del sistema local se conserva, uno de los ejes del sistema local y anterior sobre el cual realiza el giro deben estar alineados con el eje de acción de la articulación R_n, de manera que se logre orientar uno de los ejes restantes del sistema local con el cuerpo del siguiente eslabón; en el caso de traslación pura, el sistema local se desplaza sobre uno de los ejes del sistema anterior a través del eslabón hasta situar el origen del sistema local sobre uno de los puntos que definen la geometría del cuerpo o sobre uno de los puntos del eje de acción de una articulación R_n; las figuras A, B y C muestran de forma gráfica la secuencia de los sistemas de referencia que definen el modelo cinemático de una cadena del manipulador.

La secuencia de transformaciones homogéneas de la i-ésima cadena cinemática del manipulador inicia con la aplicación de una operación o transformación de rotación del sistema local *OXYZ1* un ángulo δ_1 sobre el eje *Z0* del sistema inercial *OXYZ0*, a fin de orientar el eje X_1 con el cuerpo del primer eslabón de la cadena cinemática, la cual se encuentra a una disposición angular δ_1 respecto del sistema inercial; posteriormente se aplican dos transformaciones de traslación, la primer transformación realiza un desplazamiento del sistema local *OXYZ2* una distancia dz_2 sobre el eje Z_1, mientras que la segunda transformación realiza un desplazamiento *dx3* del sistema local *OXYZ3* sobre X_2; una vez posicionado el sistema local en el eje de acción de la articulación activa o actuada *R1*, se aplica una transformación de rotación del sistema *OXYZ4* asociado al cuerpo del eslabón *Brazo*, un ángulo $\vartheta 4i$ sobre el eje Y_3; se realizan dos operaciones de traslación sobre los ejes X_4 y Z_5 con un desplazamiento *dx5* y *dz6* respectivamente, atravesando la geometría del eslabón *Brazo* para lograr situar el sistema *OXYZ6* sobre el eje de acción de la articulación rotacional pasiva *R2*, el cual es paralelo al eje de acción de *R1*; para continuar la geometría del ensamble, se aplica una transformación de traslación, desplazando al sistema local *OXYZ7* una distancia *dy7* sobre el eje Y_6, posicionando al sistema sobre la superficie de una cara lateral del eslabón *Brazo*; lo anterior descrito se muestra gráficamente en la ilustración Fig. 32.

$$T_{0,3\,i} = T_{z6}(\delta_{1\,i})T_{z3}(d_{z2i})T_{z1}(d_{x3i}) \tag{1}$$

$$T_{3,6\,i} = T_{z5}(\theta_{4\,i})T_{z1}(d_{x5i})T_{z3}(d_{z6i}) \tag{2}$$

$$T_{6,7\,i} = T_{z2}(d_{y7}) \tag{3}$$

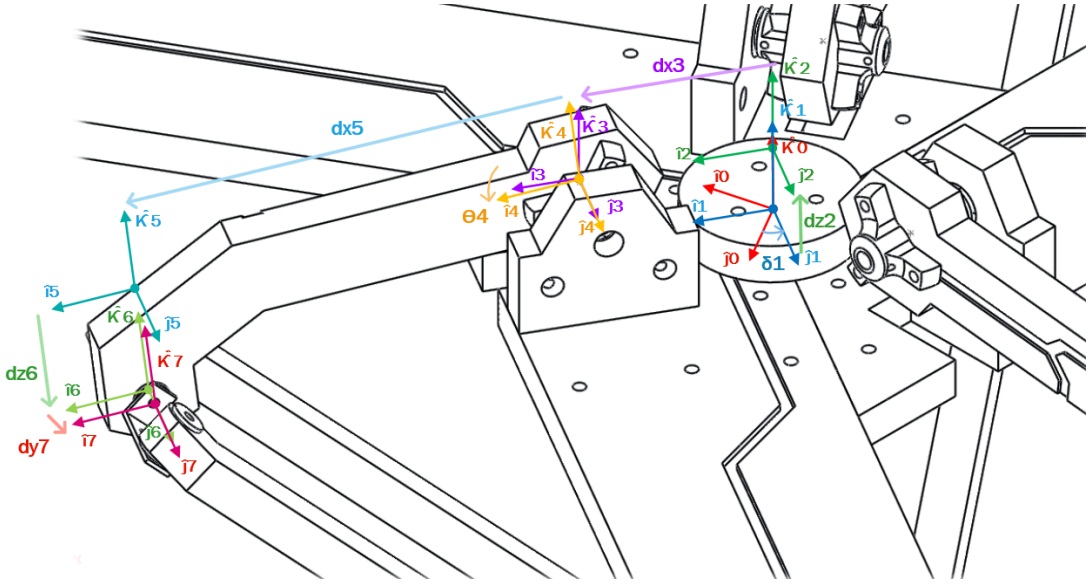


Fig. 32 Secuencia de transformaciones 1 de la i-ésima cadena del manipulador

Partiendo del sistema *OXYZ7*, el cual se encuentra posicionado en el eje de acción de *R2* y sobre la cara del eslabón *Brazo*, se aplica una transformación de rotación al sistema local *OXYZ8* asociado a la pieza *Junta* con un ángulo $\vartheta 8$ sobre el eje Y_7, orientando a X_8 con el eje de acción de *R3*; para posicionar al sistema *OXYZ9* sobre el eje de acción de *R3* se aplica una transformación de traslación sobre el eje Y_8 con un desplazamiento *dy9*; posteriormente se aplica una transformación de rotación del sistema *OXYZ10* sobre el eje X_9 con un ángulo $\vartheta 10$, a fin de orientar al eje Y_{10} con el cuerpo del eslabón *Antebrazo*; se realiza una operación de traslación sobre Y_{10} una distancia *dy11*, posicionando al sistema *OXYZ11* sobre el eje de acción de *R4*; se aplica una transformación de rotación al sistema *OXYZ12* con un ángulo $\vartheta 12$ sobre X_{11} para orientar al eje Y_{12} con el eje de acción de *R5*; se aplica una transformación de traslación con un desplazamiento *dy13* sobre el eje Y_{12} con el fin de posicionar al sistema *OXYZ13* en las caras de ensamble entre la pieza *Junta* y la pieza *base móvil* (ver Fig. 33).

$$T_{7,9\,i} = T_{z5}(\theta_{8\,i})T_{z2}(d_{y9i}) \tag{4}$$

$$T_{9,11\,i} = T_{z4}(\theta_{10\,i})T_{z2}(d_{y11i}) \tag{5}$$

$$T_{11,13\,i} = T_{z4}(\theta_{12\,i})T_{z2}(d_{y13i}) \tag{6}$$

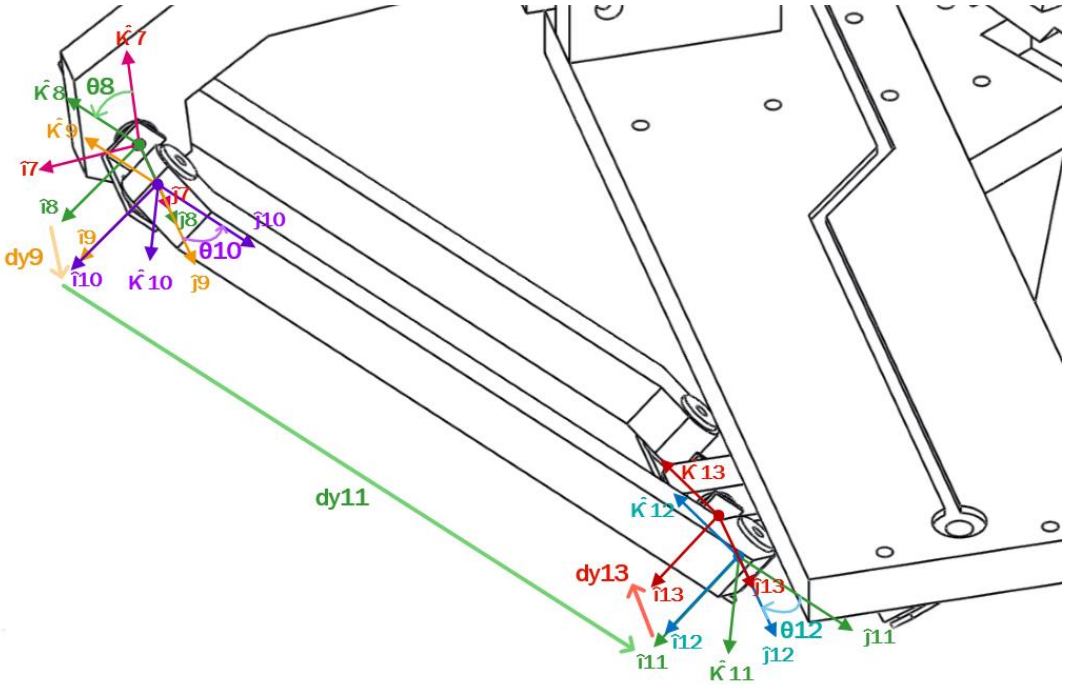


Fig. 33 Secuencia de transformaciones 2 de la i-ésima cadena del manipulador

Con la finalidad de orientar el sistema local *OXYZ14* con la *Base Móvil* del manipulador, a dicho sistema se aplica una transformación de rotación con un ángulo $\vartheta 14$ respecto al sistema local de la pieza *Junta* sobre el eje Y_{13}; posteriormente se aplican tres transformaciones de traslación a través de la pieza *Base Móvil* para llevar un sistema de referencia al punto *P* (posición del efector final del manipulador), el cual se encuentra en el centro de la *Base Móvil*; la primer traslación realiza un desplazamiento *dy15* en el eje Y_{14}, posicionando el sistema *OXYZ15* en el centro de un extremo de la *Base Móvil*, al cual se ensambla el conjunto de eslabones que conforma una cadena del manipulador; la segunda traslación realiza un desplazamiento *dx16* sobre el eje X_{15}, posicionando el sistema *OXYZ16* en el centro de la *Base Móvil*; la tercer traslación realiza un desplazamiento del sistema local *OXYZ17* una distancia *dz17* en el eje Z_{16} del sistema de referencia anterior, posicionando el sistema local *OXYZ17* en el punto *P*; Finalmente se aplica una transformación de traslación en los ejes *X, Y, Z* partiendo del sistema inercial con desplazamientos de acuerdo a las componentes (x, y, z) del punto *P*, posicionando el sistema *OP* en el punto *P* (centro de la *Base Móvil*), además se aplica una transformación de rotación sobre el eje *Zp* con un ángulo $\delta 17$ para alinear el sistema *OP* con el sistema *OXYZ17* de la i-ésima cadena cinemática (ver Fig. 34).

$$T_{13,15\,i} = T_{z5}(\theta_{14\,i})T_{z2}(d_{y15i}) \tag{7}$$

$$T_{15,17\,i} = T_{z1}(d_{x16i})T_{z3}(d_{z17i}) \tag{8}$$

$$T_{0,p} = T_{z1}(xp)T_{z2}(yp)T_{z3}(zp) \tag{9}$$

$$T_{p,17\,i} = T_{z6}(\delta_{17\,i}) \tag{10}$$

$$T_{11,17\,i} = T_{11,13\,i}T_{13,15\,i}T_{15,17\,i} \tag{11}$$

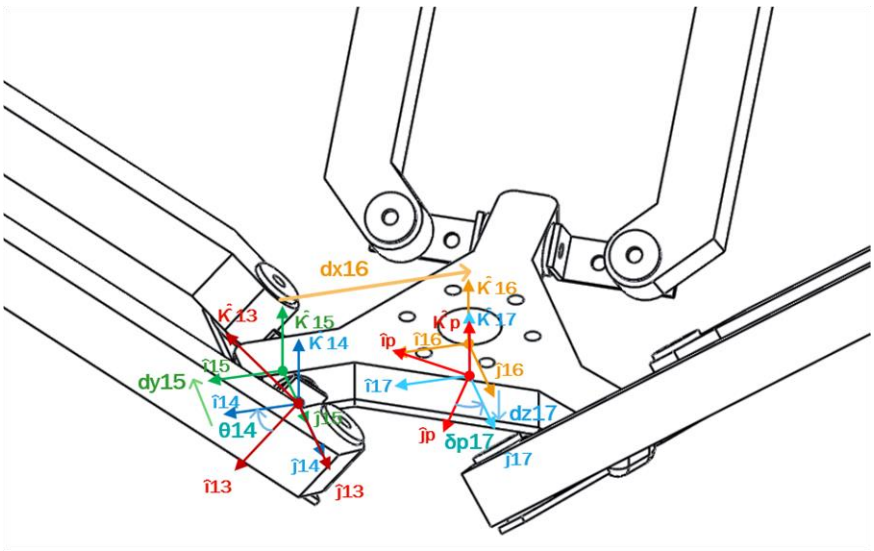


Fig. 34 Secuencia de transformaciones 3 de la i-ésima cadena del manipulador

La cinemática inversa se centra en obtener los valores de las variables articulares de cada cadena cinemática del manipulador, ya que la configuración de las mismas permite posicionar y orientar al efector final del manipulador; las soluciones cerradas (ecuaciones obtenidas mediante cinemática inversa) permiten conocer el valor angular de la respectiva articulación q_n, introduciendo como parámetros la localización deseada del efector final ($x, y, z, \alpha, \varphi, \vartheta$), lo cual es esencial para el control de posición de los manipuladores paralelos; en este trabajo únicamente se consideran los parámetros de posición del vector de localización (x, y, z), sin tomar en cuenta los parámetros de orientación del mismo; al observar las transformaciones anteriormente descritas se puede notar que las variables articulares incógnitas son ($\vartheta 4i, \vartheta 8i, \vartheta 10i, \vartheta 12i, \vartheta 14i$), las cuales son articulaciones pasivas a excepción de $\vartheta 4i$, que es una articulación activa [4, 14].

Mediante cinemática inversa, por practicidad únicamente se obtiene la ecuación algebraica de la variable articular actuada $\vartheta 4i$; se busca obtener una ecuación algebraica ya que posibilita sustituir los valores de longitud de eslabones y los ángulos de construcción de cada brazo para obtener directamente la ecuación del ángulo $\vartheta 4$ correspondiente a la i-ésima cadena cinemática del manipulador.

2.5 Solución del Ángulo θ_{4i} Mediante Cinemática Inversa

A diferencia del apartado anterior, los nombres *Rn* de en este apartado refieren un vector y no a una articulación q_n.

Se especifican los vectores de una cadena cinemática de acuerdo a la secuencia de transformaciones homogéneas aplicadas, en la Fig. 35 se muestran dichos vectores además del vector *Rp*, el cual define la posición de la base móvil respecto al sistema de referencia inercial, el cual construye un lazo cerrado para el análisis de cualquiera de las tres cadenas cinemáticas del manipulador.

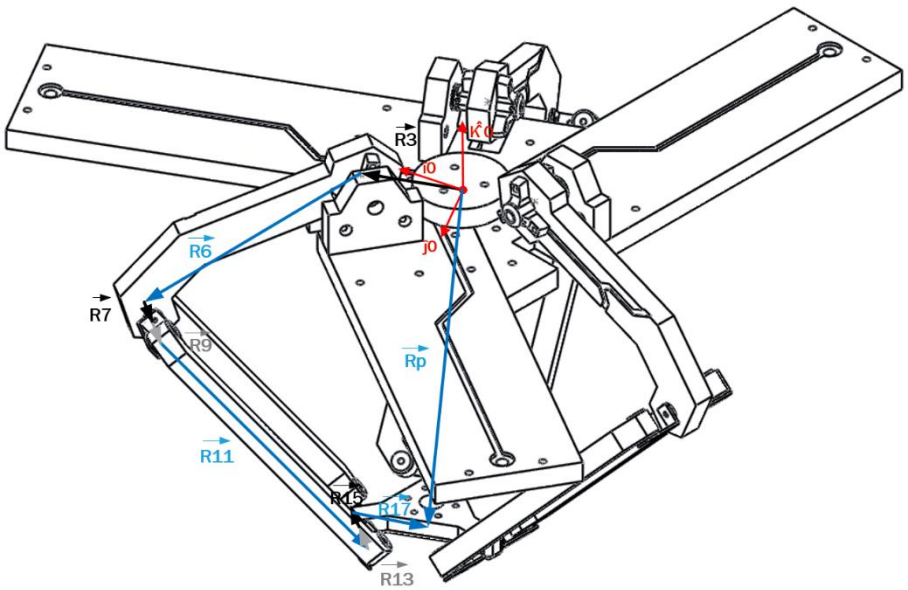


Fig. 35 Lazo vectorial para la i-ésima cadena cinemática del manipulador

Especificar un vector R_p permite generar una ecuación de lazo cerrado; teniendo en consideración que cada extremo del eslabón *Antebrazo* son puntos donde intersectan dos ejes de acción de las variables articulares incógnitas de las piezas *Junta*, buscando eliminar dichas incógnitas, se genera la ecuación de lazo cerrado en un extremo del eslabón *Antebrazo*.

$$R_{3i} + R_{6i} + R_{7i} + R_{9i} + R_{11i} = R_p + R_{17i} + R_{15i} + R_{13i} \tag{12}$$

Ya que existe una igualdad entre los vectores de lazo, se puede despejar el vector R_{11i} de la ecuación (12), el cual hace referencia al eslabón *Antebrazo*; de acuerdo a su geometría, es posible conocer la magnitud constante de dicho vector, obteniendo la ecuación.

$$R_{11i} = R_p + R_{17i} + R_{15i} + R_{13i} - (R_{3i} + R_{6i} + R_{7i} + R_{9i}) \tag{13}$$

Se representa la ecuación (13) utilizando las MTH, involucrando a todas las variables articulares

$$R_{11i} = N = T_{0p}T_{p17i}T_{1711i}n - (T_{03i}T_{36i}T_{67i}T_{79i}n) \tag{14}$$

$$T_{1711i} = T_{1117i}{}^{-1} \tag{15}$$

$$n = [0, 0, 0, 1]^T \tag{16}$$

Dónde T_{1711i} representa la matriz inversa de la secuencia de transformaciones homogéneas que van de *11* a *17* de modo que los vectores R_{13}, R_{15} y R_{17}, tendrán sentido contrario; *n* es un vector que al realizar la operación de producto punto con alguna MTH, permite considerar únicamente la *4ta* columna correspondiente al vector de posición de dicha MTH.

Se calcula la magnitud del vector R_{11i}; se sustituye R_{11i} por el valor d_{y11i} del eslabón *Antebrazo* de acuerdo a la geometría del mismo; con el fin de eliminar varias incógnitas, se obtiene el producto de elevar al cuadrado la ecuación, de la misma forma se obtiene el producto de pre multiplicar la matriz *N* con dimensión 4x1 por la transpuesta del mismo vector *N* (1x4), resultando un elemento (dimensión 1x1); agrupando y simplificando las componentes que involucran al seno y al coseno de $\vartheta 4i$, se obtienen las siguientes ecuaciones.

$$||R_{11i}|| = ||(T_{0p}T_{p17i}T_{1711i} - T_{03i}T_{36i}T_{67i}T_{79i})n|| = ||N|| \tag{17}$$

$$d_{y11i} = ||(T_{0p}T_{p17i}T_{1711i} - T_{03i}T_{36i}T_{67i}T_{79i})n|| = ||N|| \tag{18}$$

$$(d_{y11i})^2 = N^T N \tag{19}$$

$$
\begin{aligned}
(d_{11\,i})^2 = {} & d_{x16i}^{\,2} + 2\,c(\delta_{17\,i})\,c(\delta_{1\,i})\,d_{x16i}\,d_{x3i} + 2\,s(\delta_{17\,i})\,s(\delta_{1\,i})\,d_{x16i}\,d_{x3i} + d_{x3i}^{\,2} + d_{x5i}^{\,2} - \\
& 2\,c(\delta_{1\,i})\,s(\delta_{17\,i})\,d_{x3i}\left(d_{y13i} + d_{y15i}\right) + 2\,c(\delta_{17\,i})\,s(\delta_{1\,i})\,d_{x3i}\left(d_{y13i} + d_{y15i}\right) + \\
& \left(d_{y13i} + d_{y15i}\right)^2 + 2\,c(\delta_{1\,i})\,s(\delta_{17\,i})\,d_{x16i}\left(d_{y7i} + d_{y9i}\right) - \\
& 2\,c(\delta_{17\,i})\,s(\delta_{1\,i})\,d_{x16i}\left(d_{y7i} + d_{y9i}\right) + 2\,c(\delta_{17\,i})\,c(\delta_{1\,i})\left(d_{y13i} + d_{y15i}\right)\left(d_{y7i} + d_{y9i}\right) + \\
& 2\,s(\delta_{17\,i})\,s(\delta_{1\,i})\left(d_{y13i} + d_{y15i}\right)\left(d_{y7i} + d_{y9i}\right) + \left(d_{y7i} + d_{y9i}\right)^2 + d_{z6i}^{\,2} - \\
& 2\,c(\delta_{17\,i})\,d_{x16i}\,xp - 2\,c(\delta_{1\,i})\,d_{x3i}\,xp + 2\,s(\delta_{1\,i})\left(d_{y7i} + d_{y9i}\right)xp + \\
& xp^2 - 2\,s(\delta_{1\,i})\,d_{x3i}\,yp - 2\,c(\delta_{17\,i})\left(d_{y13i} + d_{y15i}\right)yp - \\
& 2\,c(\delta_{1\,i})\left(d_{y7i} + d_{y9i}\right)yp + yp^2 + 2\,s(\delta_{17\,i})\left(\left(d_{y13i} + d_{y15i}\right)xp - d_{x16i}\,yp\right) + \\
& s(\theta_{4\,i})\left(2\,c(\delta_{17\,i})\,c(\delta_{1\,i})\,d_{x16i}\,d_{z6i} + 2\,s(\delta_{17\,i})\,s(\delta_{1\,i})\,d_{x16i}\,d_{z6i} - \right. \\
& \quad 2\,c(\delta_{1\,i})\,s(\delta_{17\,i})\left(d_{y13i} + d_{y15i}\right)d_{z6i} + 2\,c(\delta_{17\,i})\,s(\delta_{1\,i})\left(d_{y13i} + d_{y15i}\right)d_{z6i} - \\
& \quad \left. 2\,c(\delta_{1\,i})\,d_{z6i}\,xp - 2\,s(\delta_{1\,i})\,d_{z6i}\,yp + 2\left(d_{x3i}\,d_{z6i} - d_{x5i}\left(d_{z17i} + d_{z2i} - zp\right)\right)\right) + \\
& c(\theta_{4\,i})\left(2\,c(\delta_{17\,i})\,c(\delta_{1\,i})\,d_{x16i}\,d_{x5i} - 2\,c(\delta_{1\,i})\,s(\delta_{17\,i})\,d_{x5i}\left(d_{y13i} + d_{y15i}\right) + \right. \\
& \quad 2\,c(\delta_{17\,i})\,s(\delta_{1\,i})\,d_{x5i}\left(d_{y13i} + d_{y15i}\right) - 2\,c(\delta_{1\,i})\,d_{x5i}\,xp + \\
& \quad \left. 2\left(d_{x3i}\,d_{x5i} + s(\delta_{1\,i})\,d_{x5i}\left(s(\delta_{17\,i})\,d_{x16i} - yp\right) + d_{z6i}\left(d_{z17i} + d_{z2i} - zp\right)\right)\right) + \\
& \left(d_{z17i} + d_{z2i} - zp\right)^2
\end{aligned}
\tag{20}
$$

La ecuación anterior se obtiene la forma:

$$
A_{1\,i}c\theta_{4\,i} + B_{1\,i}s\theta_{4\,i} + C_{1\,i} = (d_{y11i})^2 \tag{21}
$$

$$
A_{1\,i}c\theta_{4\,i} + B_{1\,i}s\theta_{4\,i} + (C_{1\,i} - (d_{y11i})^2) = 0 \tag{22}
$$

$$
A_{1\,i}c\theta_{4\,i} + B_{1\,i}s\theta_{4\,i} + D_{1\,i} = 0 \tag{23}
$$

Donde:

$$
\begin{aligned}
A_{1\,i} = {} & 2\,c(\delta_{17\,i})\,c(\delta_{1\,i})\,d_{x16i}\,d_{x5i} - 2\,c(\delta_{1\,i})\,s(\delta_{17\,i})\,d_{x5i}\left(d_{y13i} + d_{y15i}\right) + \\
& 2\,c(\delta_{17\,i})\,s(\delta_{1\,i})\,d_{x5i}\left(d_{y13i} + d_{y15i}\right) - 2\,c(\delta_{1\,i})\,d_{x5i}\,xp + \\
& 2\left(d_{x3i}\,d_{x5i} + s(\delta_{1\,i})\,d_{x5i}\left(s(\delta_{17\,i})\,d_{x16i} - yp\right) + d_{z6i}\left(d_{z17i} + d_{z2i} - zp\right)\right)
\end{aligned}
\tag{24}
$$

$$
\begin{aligned}
B_{1\,i} = {} & 2\,c(\delta_{17\,i})\,c(\delta_{1\,i})\,d_{x16i}\,d_{z6i} + 2\,s(\delta_{17\,i})\,s(\delta_{1\,i})\,d_{x16i}\,d_{z6i} - \\
& 2\,c(\delta_{1\,i})\,s(\delta_{17\,i})\left(d_{y13i} + d_{y15i}\right)d_{z6i} + 2\,c(\delta_{17\,i})\,s(\delta_{1\,i})\left(d_{y13i} + d_{y15i}\right)d_{z6i} - \\
& 2\,c(\delta_{1\,i})\,d_{z6i}\,xp - 2\,s(\delta_{1\,i})\,d_{z6i}\,yp + 2\left(d_{x3i}\,d_{z6i} - d_{x5i}\left(d_{z17i} + d_{z2i} - zp\right)\right)
\end{aligned}
\tag{25}
$$

$$
\begin{aligned}
D_{1\,i} = {} & d_{x16i}^{\,2} + d_{x3i}^{\,2} + d_{x5i}^{\,2} - d_{y11i}^{\,2} - 2\,s(\delta_{17\,i} - \delta_{1\,i})\,d_{x3i}\,d_{y13i} + \\
& d_{y13i}^{\,2} - 2\,s(\delta_{17\,i} - \delta_{1\,i})\,d_{x3i}\,d_{y15i} + 2\,d_{y13i}\,d_{y15i} + d_{y15i}^{\,2} + \\
& 2\,s(\delta_{17\,i} - \delta_{1\,i})\,d_{x16i}\,d_{y7i} + d_{y7i}^{\,2} + 2\,s(\delta_{17\,i} - \delta_{1\,i})\,d_{x16i}\,d_{y9i} + 2\,d_{y7i}\,d_{y9i} + \\
& d_{y9i}^{\,2} + 2\,c(\delta_{17\,i} - \delta_{1\,i})\left(d_{x16i}\,d_{x3i} + \left(d_{y13i} + d_{y15i}\right)\left(d_{y7i} + d_{y9i}\right)\right) + d_{z17i}^{\,2} + \\
& 2\,d_{z17i}\,d_{z2i} + d_{z2i}^{\,2} + d_{z6i}^{\,2} - 2\,c(\delta_{1\,i})\,d_{x3i}\,xp + 2\,s(\delta_{17\,i})\,d_{y13i}\,xp + \\
& 2\,s(\delta_{17\,i})\,d_{y15i}\,xp + 2\,s(\delta_{1\,i})\,d_{y7i}\,xp + 2\,s(\delta_{1\,i})\,d_{y9i}\,xp + xp^2 - \\
& 2\,s(\delta_{17\,i})\,d_{x16i}\,yp - 2\,s(\delta_{1\,i})\,d_{x3i}\,yp - 2\,c(\delta_{1\,i})\,d_{y7i}\,yp - 2\,c(\delta_{1\,i})\,d_{y9i}\,yp + \\
& yp^2 - 2\,c(\delta_{17\,i})\left(d_{x16i}\,xp + \left(d_{y13i} + d_{y15i}\right)yp\right) - 2\,d_{z17i}\,zp - 2\,d_{z2i}\,zp + zp^2
\end{aligned}
\tag{26}
$$

La ecuación algebraica de la articulación actuada *ϑ4i* del modelo cinemático del manipulador está dada por la siguiente ecuación.

$$\theta_{4\,i} = \arctan\left(\frac{B_{1\,i}}{A_{1\,i}}\right) \pm \arccos\left(\frac{D_{1\,i}}{\sqrt{A_{1\,i}{}^2 + B_{1\,i}{}^2}}\right) \tag{27}$$

La programación de las ecuaciones y desarrollo anterior se llevó a cabo utilizando el software Wolfram Mathematica; el desarrollo de la ecuación trascendental para obtener *ϑ4i* se muestra en el Apéndice A de la tesis de *Shair Mendoza Flores* con nombre "*Análisis Cinemático y Dinámico de un Robot Delta de 3 Grados de Libertad*".

Se obtienen las ecuaciones de las variables articulares actuadas de cada cadena cinemática (*ϑ41, ϑ42* y *ϑ43*), al sustituir los valores algebraicos de la ecuación (27) por los valores geométricos de cada eslabón y ángulos de construcción de la n-ésima cadena cinemática, los cuales se muestran en la *Tabla 1*; cada ecuación *ϑ4n* está en función de la posición, al introducir como parámetros las coordenadas x, y, z del punto *P*, el cual representa la posición del efector final, la ecuación arroja el valor angular que debe adoptar dicho brazo para posicionar la *base móvil* en *P*.

$d_{z2i}=4$ cm	$d_{x16i}=-6$ cm
$d_{x3i}=8.5$ cm	$d_{z17i}=-0.9525$ cm
$d_{x5i}=20$ cm	$\delta_{11}=0°$
$d_{z6i}=-5$ cm	$\delta_{12}=120°$
$d_{y7i}=1.1525$ cm	$\delta_{13}=240°$
$d_{y9i}=2.814375$ cm	$\delta_{171}=0°$
$d_{y11i}=30$ cm	$\delta_{172}=120°$
$d_{y13i}=-2.814375$ cm	$\delta_{173}=240°$
$d_{y15i}=-1.1525$ cm	

Tabla 1 Datos geométricos del manipulador paralelo delta

Para obtener una trayectoria continua rectilínea es necesario considerar el *punto inicial* del efector final o de partida de la trayectoria y el *punto de final* de la trayectoria o de interés (Horizontes); con el fin de suavizar el movimiento del efector final a lo largo de la trayectoria recta, se utiliza un perfil de velocidad conocido como *campana de gauss*, al aplicarlo a la ecuación de la recta de *i* hasta *t*, donde *t* es el tiempo de término de la trayectoria e *i* es un tiempo intermedio, se obtiene una distribución de puntos de ruta, en los cuales existe un cambio mínimo en los extremos de la trayectoria y un desplazamiento máximo a la mitad de la misma; la siguiente ecuación determina la posición del i-ésimo punto de ruta de la trayectoria continua rectilínea suavizada que inicia en *pi* y termina en *pf*.

$$\mathrm{Rtp} = \mathrm{pi} + \left(10\left(\frac{i}{t}\right)^3 - 15\left(\frac{i}{t}\right)^4 + 6\left(\frac{i}{t}\right)^5\right)(\mathrm{pf} - \mathrm{pi}) \tag{28}$$

A continuación se realizan dos pruebas con el fin de visualizar el comportamiento de cada articulación activa ($\vartheta 4$) a lo largo de la trayectoria recta; al resolver el polinomio del perfil de velocidad (donde t=30) y aplicarlo a la ecuación de la recta, se introducen las coordenadas (x, y, z) de cada punto de ruta de la trayectoria como parámetros de las ecuaciones de $\vartheta 4_n$, se obtienen los valores articulares (posición angular) que deben adoptar los tres brazos del manipulador, para que el efector final pueda recorrer la trayectoria recta, los valores de posición angular de cada brazo se muestran en la Fig. 36.

Ya que la configuración conjunta de las articulaciones activas define la posición del efector final, entre más puntos de ruta existan será mejor la resolución de cada gráfica y por lo tanto el movimiento del manipulador a lo largo de la trayectoria; los puntos de horizonte de las pruebas son:

- *Punto inicial: Prueba 1 (25, 0, 28) ; Prueba 2 (0, -20, 28)*
- *Punto final: Prueba 1 (-25, 0, 28) ; Prueba 2 (0, 20, 28)*

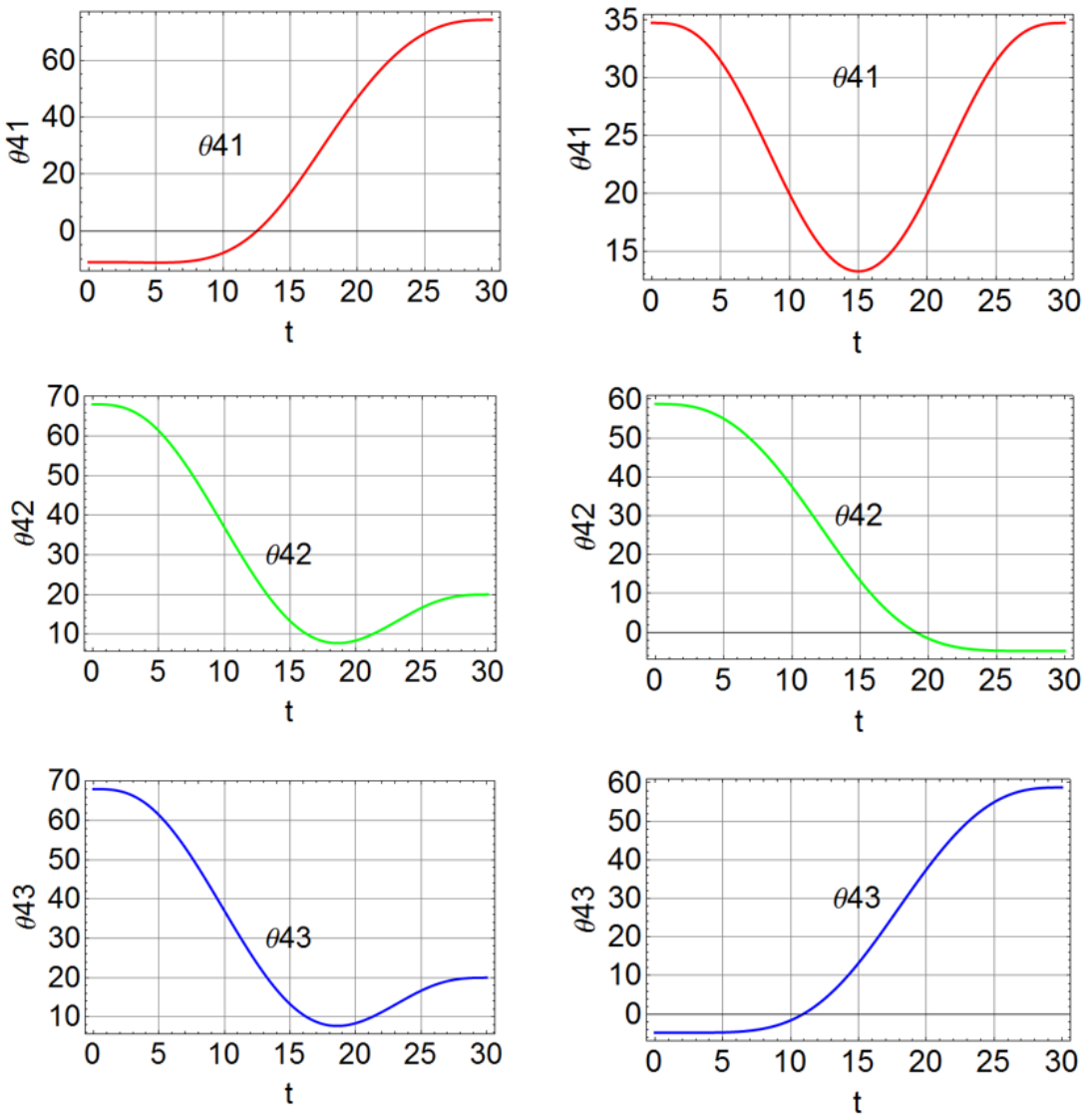


Fig. 36 Graficas con valores angulares de ϑ4 de las pruebas

3 SELECCIÓN DE COMPONENTES DEL PROTOTIPO

3.1 Simulación del Prototipo en Unity3D

La finalidad de simular el prototipo es: comprobar el correcto funcionamiento del mecanismo, validar las ecuaciones cinemáticas de la posición mediante la implementación de un algoritmo de control, que logre el movimiento controlado de la plataforma, permitiendo visualizar los recorridos y la información monitoreada.

Unity cuenta con un motor gráfico en el cual se pueden introducir modelos 3D, además cuenta con una interfaz amigable, y controles sencillos enfocados a desarrollo de video juegos, lo cual indica una programación orientada a objetos haciendo más sencilla y versátil la comunicación con el usuario; existe una gran comunidad de usuarios de Unity, así como foros y personas que comparten sus creaciones, código, modelos, o juegos.

3.1.1 Ensamble del manipulador Delta dentro del entorno de simulación

Para simular el mecanismo, las piezas del modelo diseñadas previamente se deben convertir a formato *fbx* antes de poder introducirlas a Unity.

Se agregan al ambiente de trabajo las piezas que conforman una cadena completa de brazo del mecanismo, como se muestra en Fig. 37.

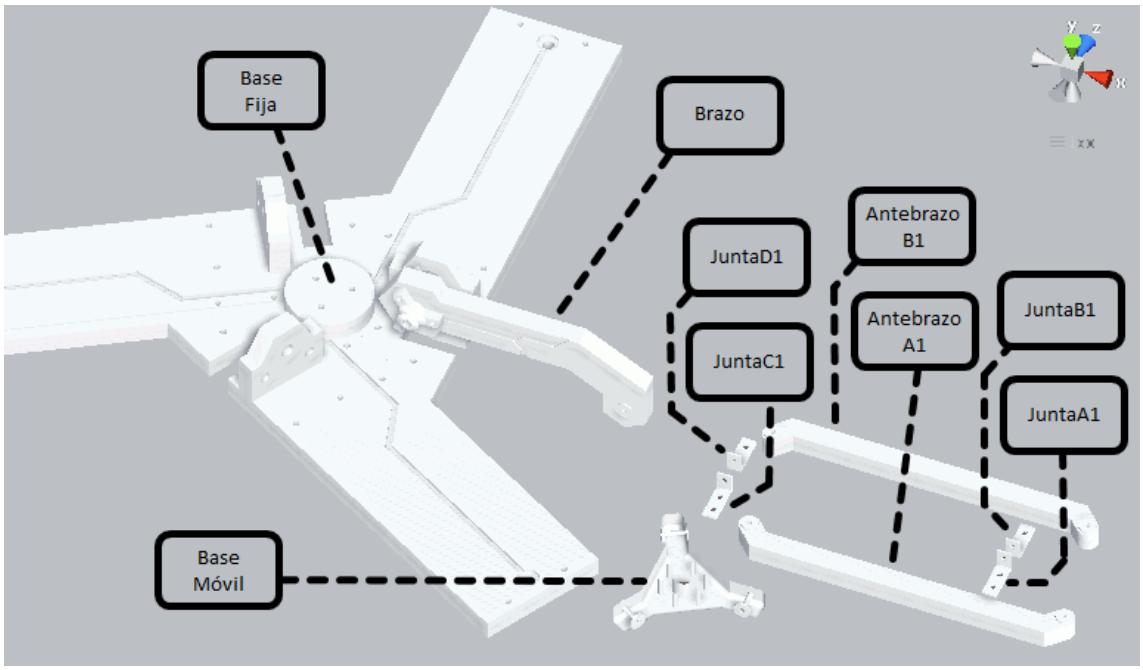


Fig. 37 Elementos importados al entorno de Unity

La pieza *Brazo* se debe alinear con el realce de la pieza *Base Fija*, para ello se traslada manualmente haciendo coincidir el eje de acción del brazo con el barreno central del realce de la base fija; a cada pieza con excepción de *Base Fija* y *Brazo*, se le añade el componente *Hinge Joint* para conectar los elementos, el cual crea flechas que indican la relación entre cuerpos y la alineación que tendrán las piezas del mecanismo al momento de correr la escena, como se muestra en la Fig. 38; las flechas deben estar alineadas respecto a los ejes de acción de cada articulación [17] ; la conexión de las piezas se realiza de la siguiente forma:

a) JuntaA1 con Brazo
b) JuntaB1 con Brazo
c) AntebrazoA1 con JuntaA1
d) AntebrazoB1 con Junta B1
e) JuntaC1 con AntebrazoA1
f) JuntaD1 con AntebrazoB1

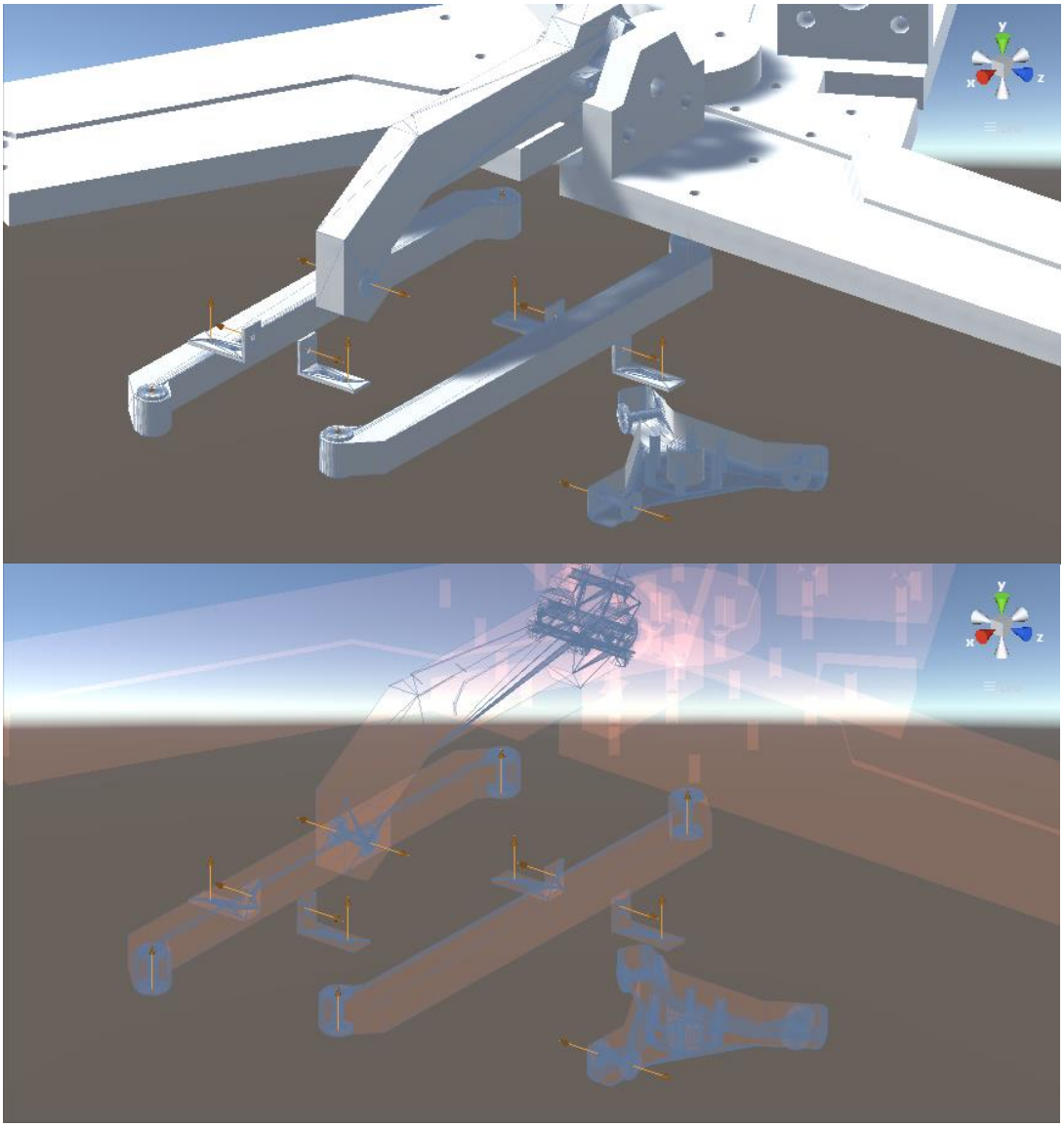


Fig. 38 Ensamble de una cadena completa de brazo

Una vez que se alinean los eslabones de la cadena se agrupan y copian, formando tres grupos, como se muestra en Fig. 39; se renombran las piezas, de modo que correspondan con el número de grupo de su respectiva cadena, se conecta la *Base móvil* con las juntas C y D de cada cadena, por lo que la base móvil es la única que contiene más de una componente *Hinge Joint*.

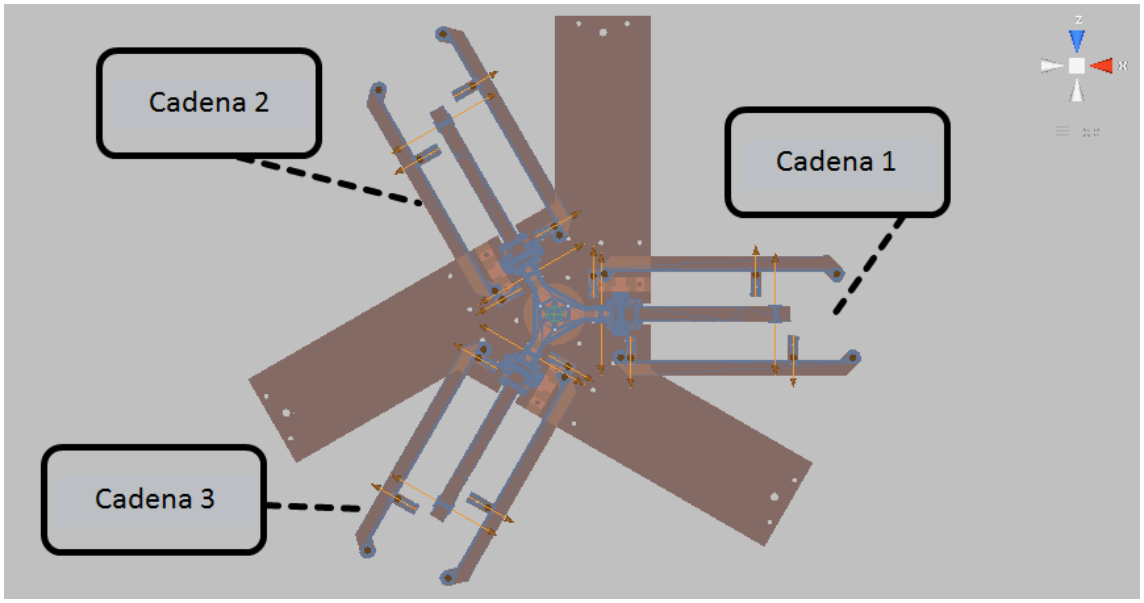


Fig. 39 Vista inferior del modelo completo relacionado

Al terminar de relacionar los eslabones, se corre la escena, observando el mecanismo ensamblado por completo, como se muestra en Fig. 40, el cual se encuentra en una posición fija debido a la inmovilidad de las piezas *Brazo* de cada cadena.

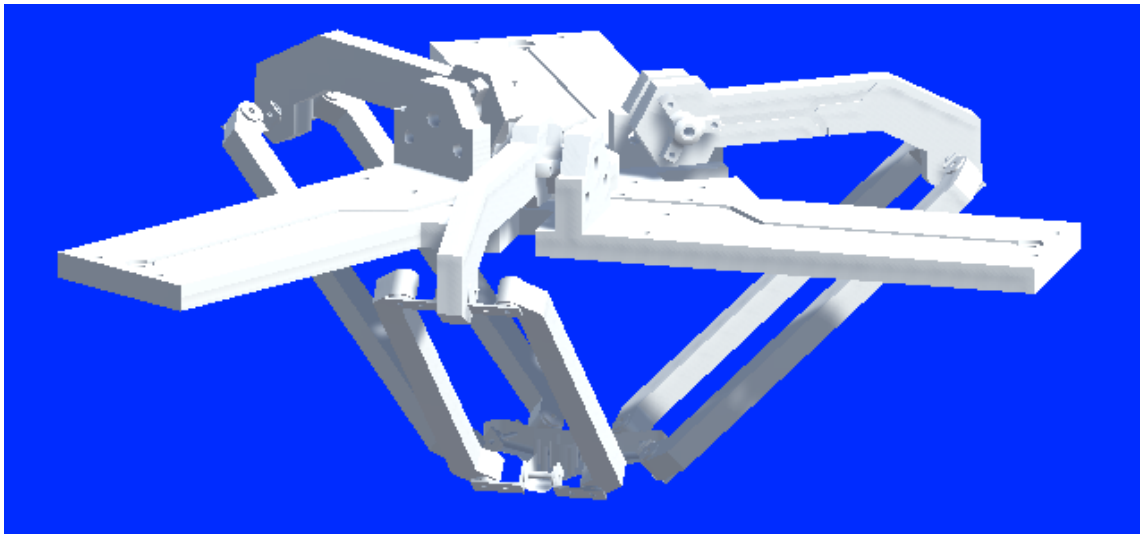


Fig. 40 Manipulador paralelo ensamblado

3.1.2 Funciones básicas y controles de la simulación

Antes de generar el algoritmo que logre controlar el manipulador virtual, se deben especificar las funciones a cumplir por la plataforma dentro de la simulación, además de planear el modo de introducir información al algoritmo de control para poder operar,

Funciones y necesidades básicas de la simulación:

a) Posicionamiento de la plataforma móvil hacia un punto de interés
b) Entorno de selección ágil del punto de interés
c) Monitoreo de coordenadas
d) Visualización controlada de la plataforma virtual

Las funciones básicas de la simulación requieren la consolidación de una interfaz gráfica de usuario (GUI) con diversos elementos, la cual permita proveer de información al algoritmo de control de la plataforma virtual.

El primer panel cuenta con una figura que representa el área de trabajo del modelo virtual, mediante un clic dentro del área de trabajo se selecciona el punto de interés a alcanzar por la *Base Móvil* [18] ; además cuenta con tres controles deslizantes, su desplazamiento individual permite modificar una componente del punto de interés sobre un eje de coordenadas.

Existe una relación entre los controles deslizantes y el área de trabajo, el control deslizante que rige al eje *z* genera un cambio en el área de trabajo como se muestra en Fig. 41, estableciendo límites para la acción de los ejes *x, y*, los cuales constan de un control deslizante con auto-corrección de valor en caso de exceder los límites máximos establecidos por el área de trabajo, corrigiendo el punto de interés.

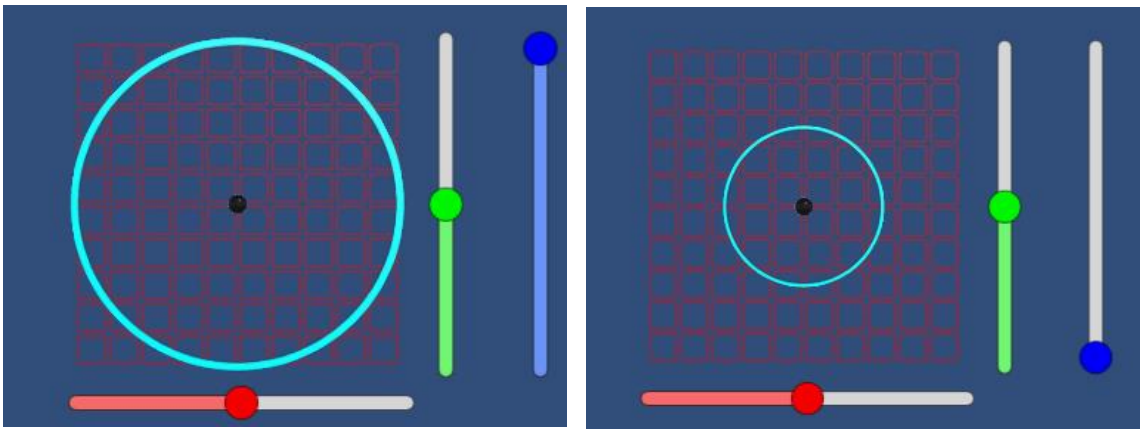


Fig. 41 Controles gráficos de posicionamiento

El segundo panel tiene como propósito visualizar las coordenadas del punto de interés mediante el uso de cuadros de texto, como se muestra en Fig. 42, además cuenta con un botón que confirma las coordenadas y comunica al algoritmo la posición de interés establecida por el primer panel.

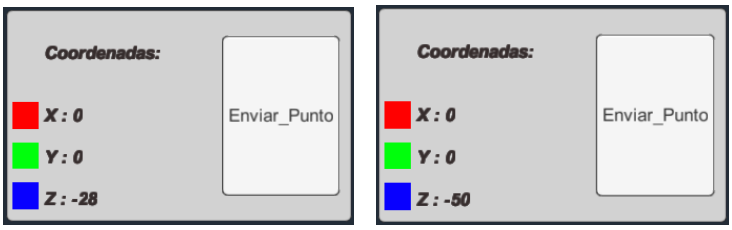


Fig. 42 Panel de visualización y ejecución de coordenadas

El tercer panel se centra únicamente en visualizar el modelo virtual de la plataforma, logrando desplazamientos horizontales y verticales de la cámara regulados mediante el uso de botones y un control deslizante; incluye la función de restaurar la vista principal, como se muestra en la figura Fig. 43.

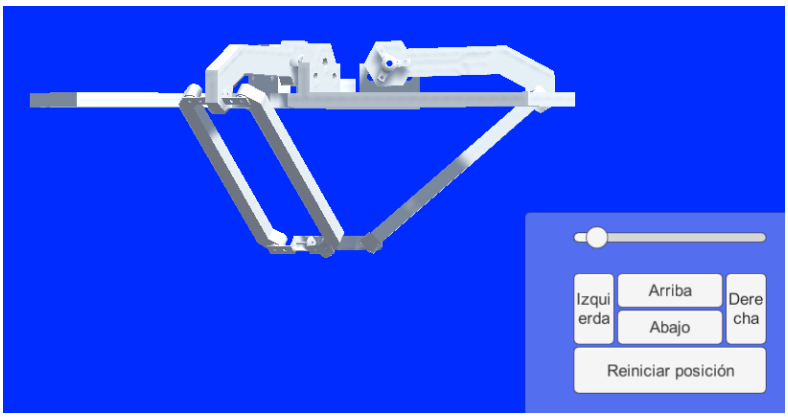


Fig. 43 Panel de visualización del modelo virtual

Se genera una distribución de los tres paneles y cámaras en una sola escena, como se muestra en Fig. 44 de manera que se tenga un panorama acondicionado para visualizar una simulación integral y transmitir información al algoritmo de control.

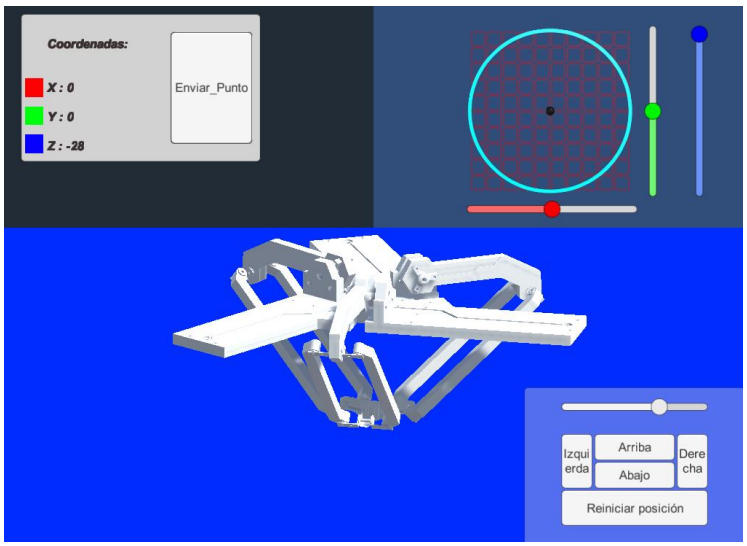


Fig. 44 Entorno de simulación e integración de GUI en ventana única

3.1.3 Implementación de algoritmo de control en Unity3D

Se genera un objeto vacío en la escena y se le agrega un script con el siguiente algoritmo en forma código.

El algoritmo que permite la simulación se muestra en la Fig. 45 y se describe a continuación.

Al correr la escena el código inicializa las variables que se mencionan a lo largo de la descripción, se colocan los brazos en una posición angular inicial, posicionando la plataforma móvil en un punto inicial guardado en la variable *pi* de tipo Vector3; la función *FixedUpdate ()* se representa dentro del diagrama de flujo, sus puntos de repetición están marcados por una letra *A* en el diagrama de flujo.

El primer panel de la GUI tiene como función permitir seleccionar las coordenadas del punto de interés, haciendo uso del área de trabajo o de los controles deslizantes, el segundo panel de la GUI permite visualizar las coordenadas del punto de interés y al dar clic en el botón Enviar_Punto tiene la función de ejecutar el algoritmo de color verde mostrado en el diagrama de flujo, el cual asigna las coordenadas del punto de interés a la variable de tipo Vector3 *pf* (punto final); se asigna el valor de 1 a la variable de tipo punto flotante ***i***; se asigna el valor de verdadero (true) a la variable booleana *calcular_PosAng*, solicitando el cálculo de las posiciones angulares de cada brazo correspondientes al i-ésimo punto de ruta de la trayectoria, que parte de un punto inicial *pi*, hasta llegar a un punto final *pf*.

La función *FixedUpdate()* del algoritmo inicialmente evalúa el estado de la variable *calcular_PosAng*, en caso de ser verdadera (true) el algoritmo calcula el i-ésimo punto de ruta, para ello se considera, el número máximo de puntos intermedios de la trayectoria y el perfil de velocidad, por lo que se resuelve su función polinomial y guarda el resultado en la variable de tipo punto flotante *funcPol*, el resultado de la función polinomial del perfil de velocidad se multiplica por la diferencia entre los puntos final *pf* e inicial *pi* que describe el movimiento rectilíneo de la trayectoria, al resultado se le suma el punto de partida *pi* punto inicial, otorgando las coordenadas del i-ésimo punto de ruta a alcanzar por la plataforma móvil del manipulador, las coordenadas del punto se guardan en la variable Vector3 *wp*.

Haciendo uso de las ecuaciones de movimiento obtenidas mediante cinemática inversa, se calcula la posición angular que requiere alcanzar cada brazo para lograr situar a la plataforma móvil sobre el punto de ruta *wp*, sus coordenadas (*x, y, z*) se ingresan como parámetros a las ecuaciones cinemáticas y estas devuelven valores de posición angular para su respectivo brazo (en radianes, se convierten a grados), el cálculo se guarda en las variables de punto flotante *gradF1, gradF2, gradF3*.

Posteriormente se hace una lectura de la posición angular actual de cada brazo, se guarda la información en las variables de punto flotante *gradI1, gradI2, gradI3*, los valores exigen una corrección ya que la lectura que se obtiene de Unity son ángulos únicamente positivos (0 a 360 grados) y los valores que arrojan las ecuaciones de movimiento de la cinemática inversa son tanto positivos como negativos de acuerdo con el orden de las transformaciones homogéneas que se siguieron en la cinemática del robot.

Se obtiene la diferencia entre posiciones angulares para el n-ésimo brazo $gradI_n$ y $gradF_n$ (actual y deseada, respectivamente), guardando el resultado en la respectiva variable $difP_n$.

A las variables de tipo punto flotante z_1, z_2, z_3, le son asignados los valores de la posición angular actual de cada brazo, guardada anteriormente en la variable $gradI_n$ del n-ésimo brazo; se le asigna el valor de falso a la variable booleana *calcular_PosAng* con la intención de que no se repita el procedimiento anteriormente descrito hasta que los brazos alcancen las posiciones angulares calculadas y se solicite un nuevo cálculo del punto de ruta i+1, para el siguiente periodo de la trayectoria.

En cada ciclo que Unity ejecuta la función *FixedUpdate ()*, a las variables z_n de posición angular del n-ésimo brazo se les suma la diferencia angular respectiva anteriormente calculada $difP_n$ multiplicada por *Time.deltaTime* como ganancia, permitiendo ver el movimiento de la plataforma; de esta forma el algoritmo adquiere propiedades de un control proporcional; a la orientación del n-ésimo brazo se le asigna el nuevo valor de z_n, la posición angular ya sumada.

Posteriormente se compara la diferencia angular $difP_n$ del n-ésimo brazo con cero, para identificar la dirección de giro, además se compara el valor de z_n posición angular (ya sumada) actual del n-ésimo brazo con la posición angular final (deseada) $gradF_n$, con la finalidad de conocer si el brazo logro igualar o exceder la posición calculada para el i-ésimo punto de ruta; en caso de que algún brazo cumpla la combinación de ambas condiciones, se le asigna el valor de cero a la correspondiente variable $difP_n$ por lo que no volverá a generar ningún cambio en la posición angular del n-ésimo brazo en posteriores ciclos de ejecución de la función *FixedUpdate()*.

Consiguientemente se evalúa si las diferencias angulares de todos los brazos son iguales a cero sobre entendiendo que han igualado o excedido la posición angular deseada, en caso de ser cierto, se evalúa el valor de la variable *i*.

En caso de que *i* sea igual o mayor al número de periodos totales en los que se segmenta la trayectoria, se actualiza el valor de la posición inicial *pi*, se le asigna el valor del punto final *pf*, indicando el término del recorrido de la trayectoria, posicionando de la base móvil del manipulador sobre el punto de interés *pf* y preparando la posición inicial para el cálculo que implica recibir posteriormente un nuevo punto de interés ingresado por el usuario mediante la interfaz gráfica de usuario GUI.

En caso de que el valor de *i* no sea mayor al número máximo de puntos de ruta que definen la trayectoria, el valor de la variable *i* incrementa unitariamente, además se establece como verdadero (true) el valor de la variable *calcular_PosAng* indicando la necesidad de realizar un nuevo cálculo de un punto de ruta con la función polinómica del perfil de velocidad y ecuaciones de movimiento, para obtener nuevos objetivos de posiciones angulares a alcanzar para cada uno de los brazos, del periodo actualizado *i*, ya que no ha terminado el recorrido de la trayectoria que lleva a la plataforma móvil del manipulador hacia el punto de interés.

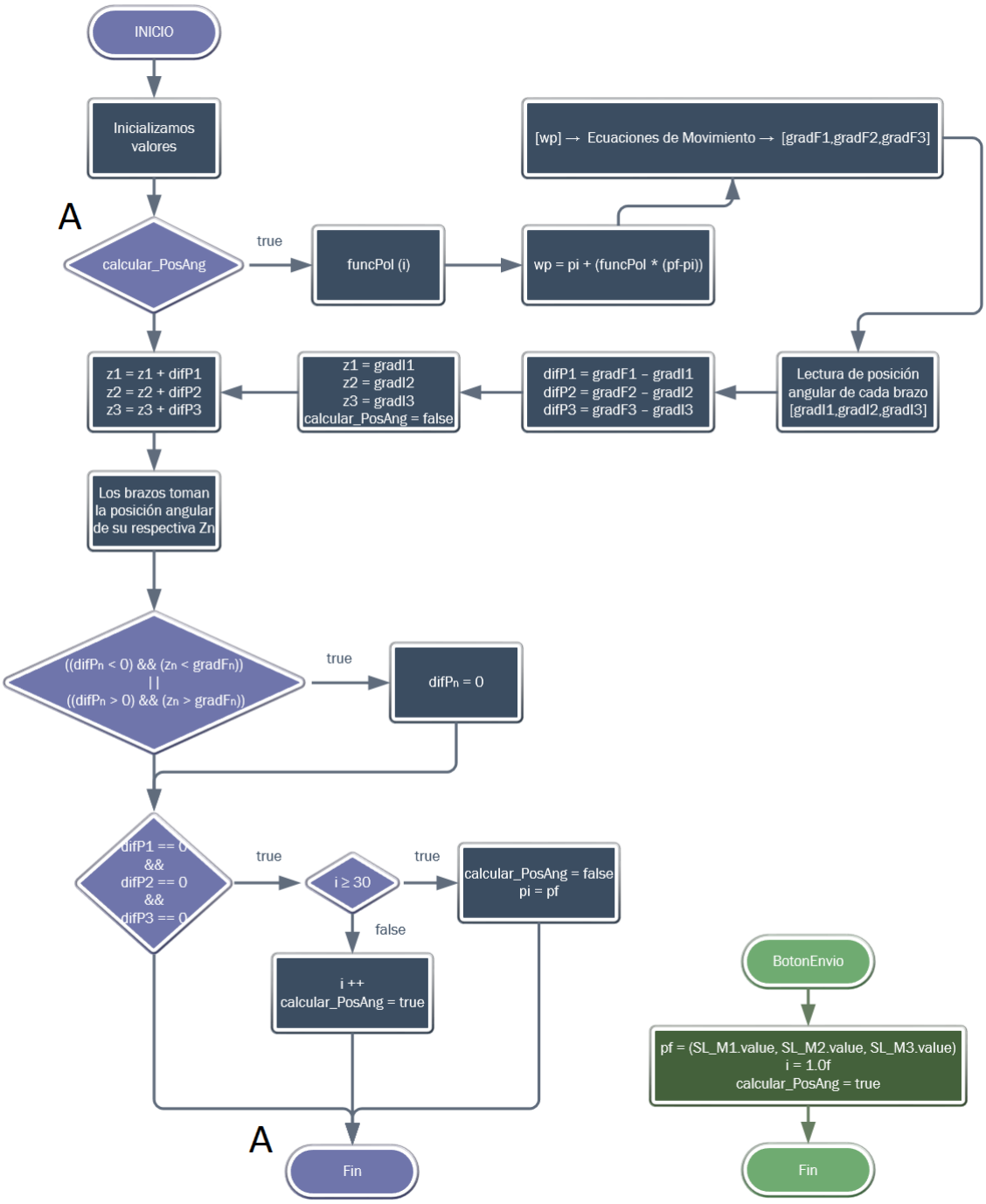


Fig. 45 Diagrama de flujo del algoritmo de la simulación

Dentro del entorno de desarrollo de Unity se agrega un efecto de luz al modelo de la plataforma móvil con la finalidad de visualizar su trayectoria, con el mismo propósito se instancia una esfera cada vez que se realiza el cálculo de un nuevo punto de ruta [17].

Al correr la escena y probar la funcionalidad de la simulación es de notar que a pesar de que sigue correctamente la trayectoria recta existe perturbación en la base móvil cuando la posición en *z* es máxima y los brazos del manipulador se encuentran extendidos, esto se debe a que las *Juntas A* y *B* de cada cadena de brazo no están unidas, cada junta gira de forma independiente, des coordina los elementos siguientes de cada cadena, tal efecto no se logra apreciar en el modelo generado dentro del software de CAD en el cual se diseñaron y ensamblaron las piezas; la plataforma muestra la necesidad de diseñar una pieza de unión para las *Juntas A* y *B*, forzando a que ambas piezas adopten una misma rotación.

Se genera el ejecutable de la escena y al correrla se coloca en el punto inicial (-15, 0, -35.5) y se selecciona el punto de interés o final (15, 0, -35.5), la trayectoria que genera se muestra en la Fig. 46 utilizan diferentes colores y las esferas aparecen en el instante en que los brazos llegan a la posición angular deseada y se realiza el cálculo de la posición angular para el siguiente punto de ruta, se observa que el tiempo de recorrido entre esferas es el mismo, efectivamente la función polinomial del perfil de velocidad suaviza el movimiento al inicio y al final del recorrido de la trayectoria; la ilustración Fig. 47 muestra una vista distinta.

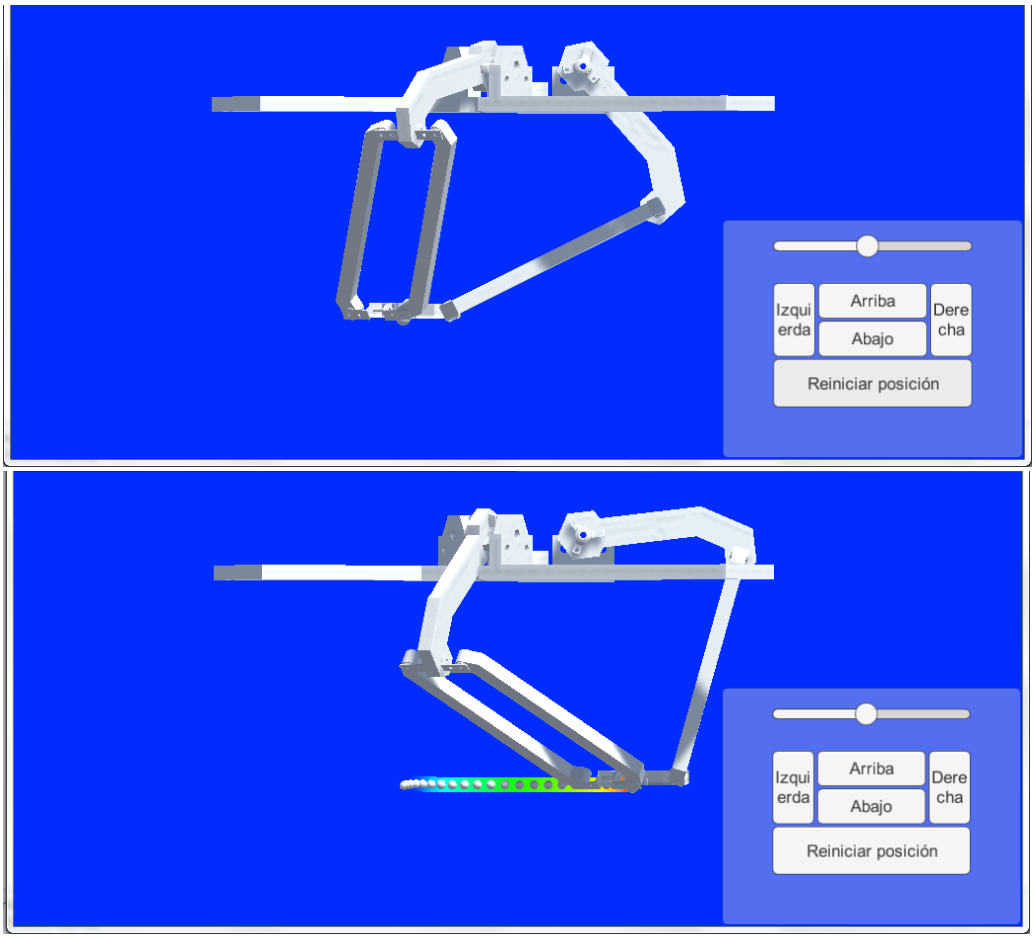


Fig. 46 Vista frontal de la simulación

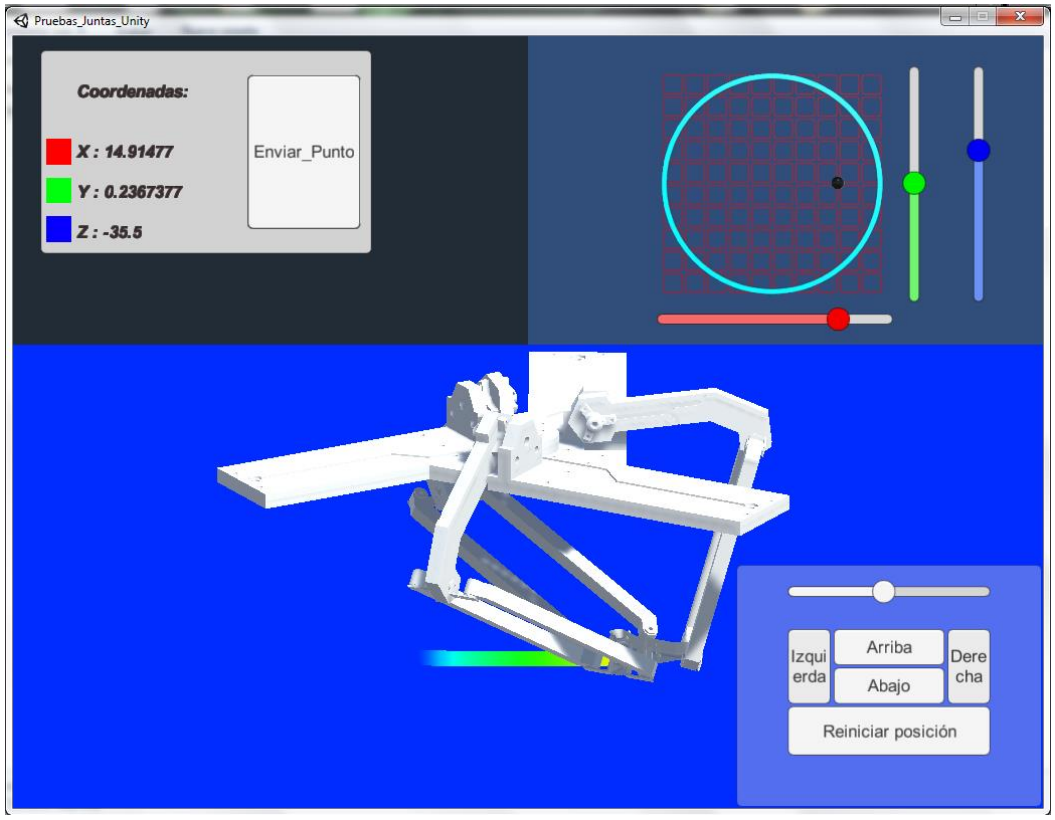


Fig. 47 Vista inclinada de la simulación

3.2 Construcción y Caracterización de Elementos

Una vez validadas las ecuaciones de movimiento para cada brazo y el mecanismo mediante la visualización de la trayectoria recta de un punto inicial a un punto de interés dentro de la simulación, se tiene la certidumbre de que el mecanismo y el algoritmo de control son funcionales, por lo que se procede a manufacturar y caracterizar las piezas que conforman un brazo del manipulador.

La manufactura de las piezas requirió utilizar corte CNC ya que es un proceso replicable, el material elegido que constituye a los eslabones del manipulador es madera con ¾ de pulgada de espesor, se eligió el material con base en su bajo costo.

También fue requerida la manifactura aditiva, utilizando una impresora 3D *Ender3* se generaron las piezas utilizando PLA como material de impresión, las piezas se imprimieron configurando los parámetros de impresión a 0.3 milímetros para el espesor de cada capa, una tolerancia de +0.4 milímetros de ampliación en el diámetro de barrenos, considerando la expansión del material; la temperatura de la cama y extrusor fueron configuradas según especifica el fabricante para trabajar el material.

La caracterización de las piezas es necesaria para llevar a cabo un estudio de movimiento del manipulador, considerando un factor se seguridad de dos, la relevancia del estudio de movimiento del sistema sobredimensionado, radica en encontrar y asegurar el torque necesario con el cual deben contar los actuadores para levantar los brazos del manipulador con el doble de su peso original, obteniendo mayor margen de operación de carga, dicho parámetro de torque cobra gran relevancia al momento de seleccionar actuadores.

La caracterización consiste en obtener la masa de cada pieza fabricada, las cuales componen a una cadena del manipulador e ingresar los datos al modelo dentro del software de CAD, la *Tabla2* muestra el nombre de las piezas móviles, su respectiva masa y el material que las conforma.

Tabla 2 Caracterización de elementos

Nombre de pieza	Masa [gr]	Material
Brazo	150	Madera
Antebrazo	65	Madera
Rondana	2.5	Metal
Tuerca de seguridad	2.5	Metal
Tornillo 3/16 * 1½ pulgada	5	Metal
Junta	15	Metal
Base móvil	180	PLA
Unión brazo	7	PLA
Unión junta A	4	PLA
Unión junta B	7	PLA

3.2.1 Obtención de Torque con factor de seguridad 2

A cada pieza se le introduce la información de masa respectiva; ya que el software de CAD considera datos de masa, los reparte en el volumen y geometría del elemento, arrojando como resultado la densidad, posteriormente se reescribe la densidad multiplicada por el factor de seguridad dentro del ensamble completo del manipulador; al aplicar gravedad en dirección de la plataforma móvil y asignar el giro del actuador con la intención de hacer que la plataforma móvil se acerque a la base fija, definiendo inicialmente 1Nm como parámetro de torque para cada junta actuada del brazo, se puede notar que los brazos y la plataforma móvil caen; al incrementar paulatinamente dicho parámetro y correr el cálculo de la simulación dentro del software de CAD, se puede observar que la base móvil desciende cada vez más lentamente.

La base móvil haciende cuando el torque se acerca a los 2Nm, por lo que se considera como el torque necesario para mover el manipulador con un factor de seguridad de 2.

3.3 Planeación de Funciones del Robot

Las funciones a ejecutar del robot son:

a) Posicionar la plataforma móvil hacia un punto de interés.
b) Posicionar la plataforma móvil hacia un punto de calibración (HOME).

La parte electrónica del robot comprende actuadores, sensores y un controlador para implementar un sistema de control de lazo cerrado como se muestra en [13]; la Fig. 48 muestra un esquema general del sistema atendiendo los requerimientos de las funciones a y b, se describe a continuación; la principal característica con la que deben contar los actuadores es el torque, considerando un factor de seguridad de dos; los actuadores deben estar acoplados a una etapa de potencia que amplifique las señales emitidas por el controlador digital, para brindar de movilidad controlada a la plataforma.

Los sensores tienen la tarea de obtener información del estado del robot, por lo que debe transmitir hacia el controlador las señales correspondientes de posición angular e indicar el límite de movimiento de cada brazo en la función de calibración (*HOME*).

El controlador digital debe recibir el punto de interés introducido por el usuario, realizar los cálculos de la cinemática y almacenar la posición angular que los brazos deben adoptar en cada segmento de la trayectoria, además el controlador debe contar con suficientes puertos de entrada-salida acondicionados para obtener lecturas de cada sensor y emitir las señales correspondientes a las etapas de potencia, logrando accionar los actuadores, posicionando la base móvil del manipulador en el punto de interés o llevándola a la posición de calibración.

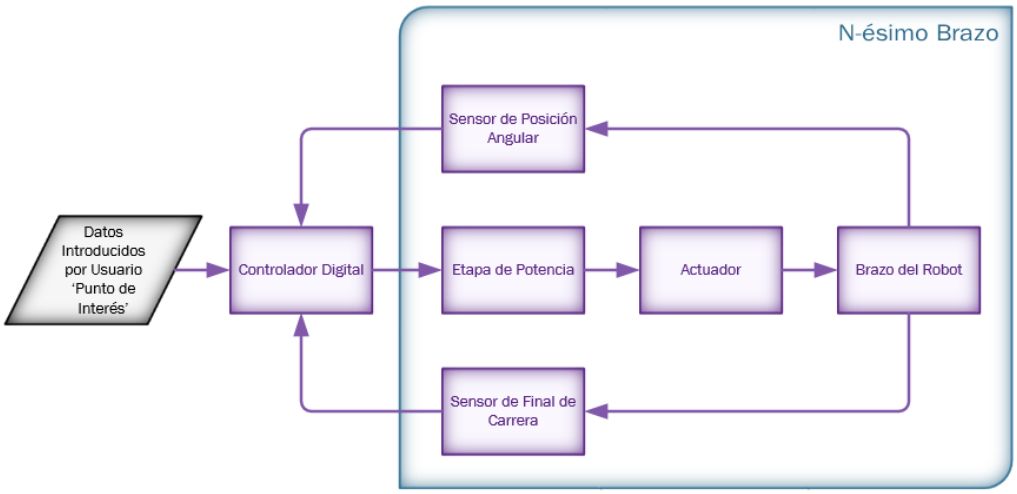


Fig. 48 Diagrama de funcionamiento del sistema robótico

Una vez definidas las funciones y elementos del prototipo piloto se proceden a seleccionar los componentes que logren satisfacer dichas necesidades; al considerar un componente se debe buscar su hoja de especificaciones, para conocer datos de torque, temperatura, voltaje y corriente de operación, entre otros.

3.4 Selección de Actuadores

Para una correcta selección de actuadores, se deben comparar las necesidades o propiedades del prototipo piloto con las especificaciones que proporciona el fabricante en las hojas de datos del actuador considerado; únicamente se consideraron actuadores eléctricos, descartando neumáticos e hidráulicos, por precio, instalación y control. El criterio de selección para los actuadores fue:

3.4.1 Consideraciones

Torque y precisión de posicionamiento

Etapa de potencia y dimensiones

Fuente de alimentación

El prototipo únicamente busca posicionar la plataforma móvil del manipulador en un punto de interés, no busca alcanzar velocidades o aceleraciones altas, además debe tener la capacidad de mantener una posición angular constante, por lo que un motor *hibrido* o *a pasos* es una buena opción; las etapas de potencia de dichos motores tienen dimensiones considerables, por lo que se debe valorar su instalación sobre el prototipo o construir un módulo externo (Ver Tabla 3); El uso de motores híbridos debe utilizar una fuente de alimentación relativamente potente, ya que se debe considerar el consumo de tres actuadores operando simultáneamente [3].

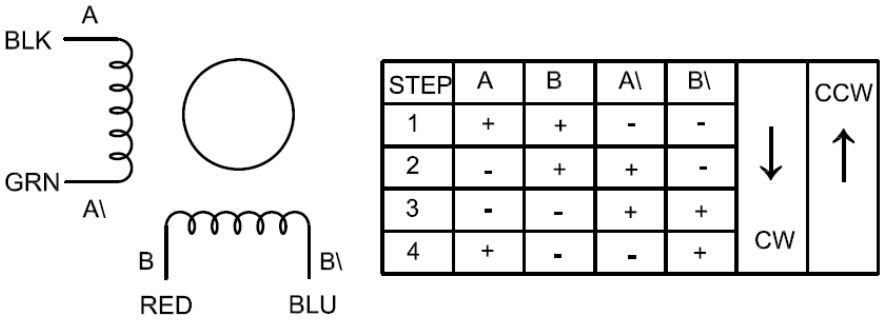
Tabla 3 Selección de actuadores y controladores (drivers)

COMPONENTE	ESPECIFICACIONES
	17HS15-1684S-PG5 • Motor a pasos Bipolar • Corriente por fase: 1.68 • Torque sostenido del motor sin caja de engranes: 0.36Nm • Ángulo de paso sin caja de engranes: 1.8° • Temperatura máxima: 80°C • Caja reductora de engranes planetarios con relación 5:1
	DRV8825 • Voltaje de alimentación: 8.2V a 45V • Corriente máxima en cada salida: 2.5Amp a 24V y 25°C • Resolución máxima de pasos: 1/32 • Corriente de operación ajustable • Voltaje lógico de operación 3.3V a 5V

El conjunto seleccionado que cumple con el torque requerido, precisión de posicionamiento, una etapa de potencia de fácil implementación, tamaño reducido y una fuente de alimentación de potencia razonable, es el motor NEMA 17 con caja reductora de engranes planetarios con relación 5:1, el cual se puede acoplar a las salidas del módulo DRV8825 que energiza y regula la corriente que suministra en secuencias a las bobinas del motor para generar el giro de la flecha.

El movimiento del motor está en relación a la aplicación de los pulsos, la Fig. 49 muestra un esquema básico de las bobinas internas del motor a pasos y la secuencia de pulsos que requieren las bobinas del motor para generar un desplazamiento angular de paso completo, la dirección de giro depende del orden con el cual se energizan las bobinas, ya que genera un movimiento con dirección a las manecillas del reloj o en contrasentido, las imágenes fueron obtenidas de la hoja de especificaciones del motor.

La construcción interna de un motor a pasos hibrido con imanes permanentes y armadura multidientes, se muestra en la Fig. 50, además de la conexión interna de las bobinas del estator; el flujo de corriente que atraviesa a las bobinas produce un campo magnético en los dientes del estator que atrae o repele el campo magnético de los dientes de la armadura que conforman al rotor, logrando desplazarlo angularmente; el polo magnético en los dientes de la bobina del estator, se invierte al invertir el flujo de corriente que atraviesa a la bobina [3, 19].



STEP	A	B	A\	B\
1	+	+	-	-
2	-	+	+	-
3	-	-	+	+
4	+	-	-	+

Fig. 49 Diagrama de conexión del motor a pasos bipolar y secuencia de activación

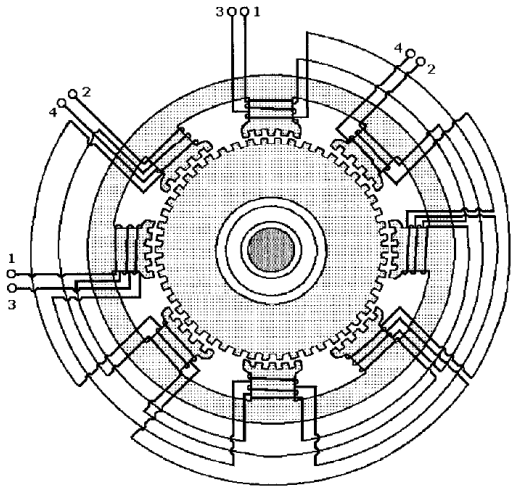


Fig. 50 Diagrama de conexión interna del motor hibrido (sección transversal)

El DRV8825 necesita de únicamente dos señales para controlar las secuencias de movimiento requeridas por el motor a pasos, logrando simplificar el algoritmo de control comparado con el uso convencional de un *puente H* que sirve solo para amplificar las señales de entrada; la primer señal de entrada indica la dirección del movimiento, mediante el estado del pin (alto o bajo) se logra controlar el sentido de giro del motor; la segunda señal de entrada es de paso, mediante un flanco de subida se le indica al driver que debe hacer un cambio en la secuencia de pulsos de salida, de manera que se energicen las bobinas correctamente para hacer girar al motor, la secuencia coordinada de pulsos de salida las genera automáticamente, considerando siempre el estado del pin de dirección; el DRV8825 cuenta con pines que al polarizarlos utilizando las combinaciones detalladas en la hoja de datos del driver, permite configurar la resolución de paso del motor hasta en 32 partes como máximo generando micro pasos; lo que significa que cada pulso enviado al pin de paso (step), permite al motor desplazarse angularmente una fracción del paso completo previamente establecido por hardware; el uso de micro pasos e incremento de la resolución compromete el torque del motor, ya que se suministra a las bobinas una fracción de la corriente que se envía en la configuración de pasos completos como se muestra en la Fig. 51; el campo magnético que se genera en el estator del motor disminuye, reduciendo la fuerza del efecto de repulsión/atracción que provee de movimiento al motor.

La implementación del módulo DRV8825 requiere conocer la corriente de operación del motor, para calibrar la corriente de salida hacia las bobinas por debajo del valor de operación para no dañar el motor; la relación de salida de corriente por fase, es del doble del voltaje de referencia medido en el potenciómetro integrado a la placa del driver; las señales de entrada (dirección y paso) del DRV8825 requieren de tiempos de espera mínimos, menor a dos microsegundos para su correcto funcionamiento, los parámetros anteriormente establecidos se indican en la hoja de especificaciones y se muestran en la Fig. 52.

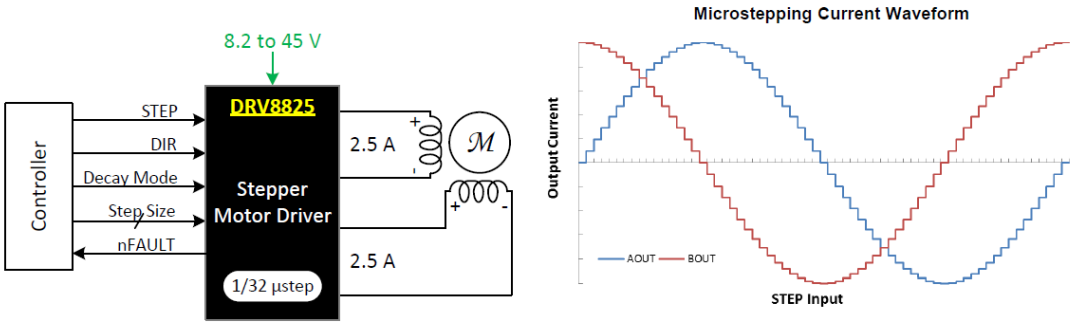


Fig. 51 Diagrama de conexión del DRV8825 y grafica de corriente configurado a 32 micro pasos

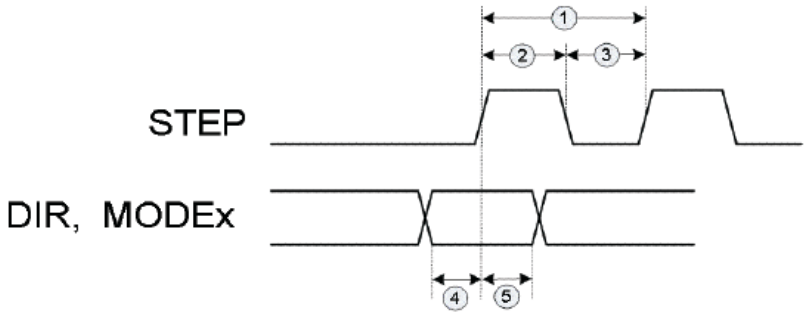


Fig. 52 Diagrama de tiempos de espera para señales de entrada del DRV8825

Se consideraron como actuadores, motores de corriente directa, brushless trifásicos con trenes de engranes, servomotores, motores a pasos NEMA 17, 23, 24, actuadores lineales acoplados con mecanismo de 4 barras; Etapas de potencia, puentes H, ESC, drivers a4988, DRV8825, TB6600, M542T; Fuentes de alimentación conmutadas de 60W hasta 400W; Los componentes se descartaron ya que algunos excedían las especificaciones de torque, de corriente en la etapa de potencia, incrementando requerimientos de la fuente de alimentación e instalación, elevando considerablemente el precio.

3.5 Selección de Sensores

En la selección de sensores únicamente se consideran sensores internos para conocer el estado del robot y satisfacer las necesidades de cada función planteada, no se requiere de sensores externos que reconozcan el área de trabajo, los objetos a manipular ni sus características; las funciones planteadas coinciden en la necesidad de obtener información de posición angular de cada brazo del robot, además se debe contar con un sensor de final de carrera para identificar cuando los brazos logren alcanzar una posición angular previamente establecida *HOME*, con la finalidad de calibrar los sensores y ajustar las variables de posición angular cuando el usuario lo indique. El criterio de selección para los sensores fue:

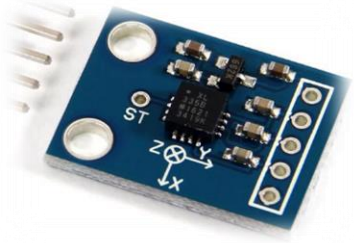
3.5.1 Consideraciones

Características de operación

Desgaste y dimensiones

Adaptabilidad y montaje

Tabla 4 Selección de sensores

COMPONENTE	ESPECIFICACIONES
	H2010 • Sensor óptico de herradura • Voltaje de operación: 3.3V a 5V • Corriente de colector: 20mA • Salida digital de presencia
	ADXL335 • Acelerómetro de 3 ejes • Voltaje de operación: 1.8V a 3.6V • Corriente de operación: 350µA • Salidas analógicas por cada eje

El sensor seleccionado H2010 óptico servirá como indicador de final de carrera, no sufre desgaste ya que no tiene contacto con alguna otra pieza, el sensor se conforma de un diodo emisor de luz infrarroja y un elemento receptor foto sensible que capta el haz de luz, se encuentran montados a los extremos de una misma pieza, posicionados de frente en una vista directa, es fácil de instalar y únicamente requiere de una etapa de acondicionamiento de señal para la posterior adquisición por el controlador digital [2].

El funcionamiento del sensor óptico requiere de una pieza que atraviese por la abertura del sensor, para permitir o interrumpir el paso del haz de luz del led que se hace incidir sobre el opto transistor, indicando la presencia o ausencia de la condición monitoreada, la configuración seleccionada se muestra en la Fig. 53, al realizar mediciones de voltaje en el colector del opto transistor, el voltaje es máximo cuando se interrumpe el haz de luz, ya que se encuentra inactiva la base del opto transistor e impide el paso de la corriente a través del mismo, al permitir el paso del haz de luz hay un cambio de voltaje en el colector, la corriente atraviesa al opto transistor hacia el emisor y llega a tierra directamente.

El sensor llega a ser susceptible al ruido eléctrico, por lo que se requiere de una etapa de acondicionamiento de señal como se muestra en la Fig. 54, para ello se implementa un comparador de voltaje con un amplificador operacional UA741CN, convirtiendo la señal de voltaje analógica del sensor en una señal digital, posteriormente el controlador muestrea la señal digital (OUTPUT) emitida por el filtro comparador para obtener el estado de la condición monitoreada.

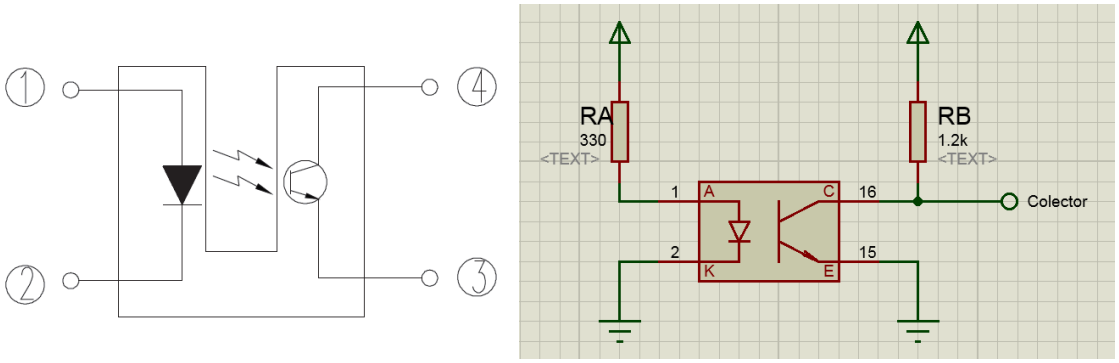


Fig. 53 Diagrama del sensor óptico de herradura y diagrama de pruebas analógico

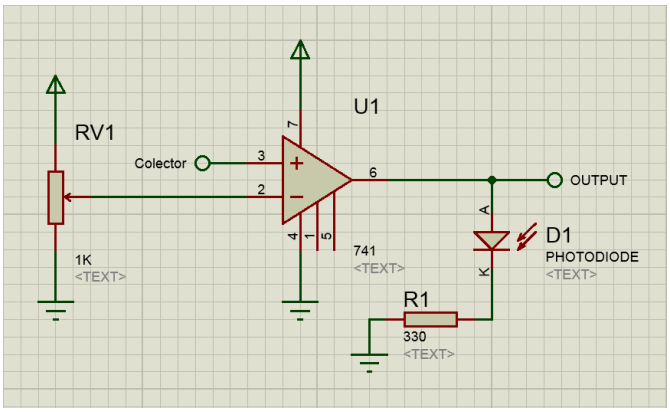


Fig. 54 Circuito comparador de voltaje con salida digital y led indicador

El sensor seleccionado para obtener información de posición angular es el módulo GY-61, es fácil de montar, cuenta con dos barrenos para fijarlo con algún elemento de soporte, no existe contacto con alguna otra pieza, por lo que no genera rozamiento ni desgaste; el módulo contiene el circuito integrado ADXL335, que mide la aceleración dinámica producto del movimiento, vibraciones y la aceleración estática con apoyo de la gravedad; el sensor es de bajo consumo y aporta como salidas señales analógicas que indican la inclinación del componente sobre los 3 ejes de acción (X,Y,Z) como se muestra en la Fig. 55 (obtenida de la hoja de especificaciones del componente), las cuales son simétricas en la mitad del giro sobre cada eje, además hay que destacar que el sensor se requiere caracterizar, es necesario conocer la ecuación que define su comportamiento ya que no es lineal.

La implementación del módulo GY-61 se basa en adquirir información de aceleración estática, detectando el ángulo de inclinación del eslabón *Brazo* sobre el cual se debe montar como se muestra en Fig. 56; únicamente se requiere de una señal del sensor ya que el brazo del manipulador se mueve angularmente sobre un eje, la señal analógica emitida por el sensor debe ser traducida mediante un ADC para que el controlador digital pueda procesar la información utilizando la ecuación característica de la señal del eje del sensor, con la finalidad de conocer la posición angular de cada brazo actuado, dicha medida convencionalmente es proporcionada por un encoder incremental o absoluto.

El nivel de BIAS para los ejes X, Y, es de 1.5V ±0.15 Volts; el nivel de BIAS del eje Z es de 1.5V ±0.3Volts, cuando el sensor se encuentra en la posición que indica cero g; el ancho de banda de muestreo del sensor tiene un rango que va de 0.5Hz a 1600Hz para los ejes X, Y; un rango de 0.5Hz a 550Hz para el eje Z.

Las pruebas de este componente electrónico requieren el uso de un nivelador o algún otro instrumento para calibrar el cero físicamente del módulo del sensor, ya que es el punto inicial de donde parten las pruebas y se comienza a tomar lecturas, hasta llegar a los límites de cada señal del sensor.

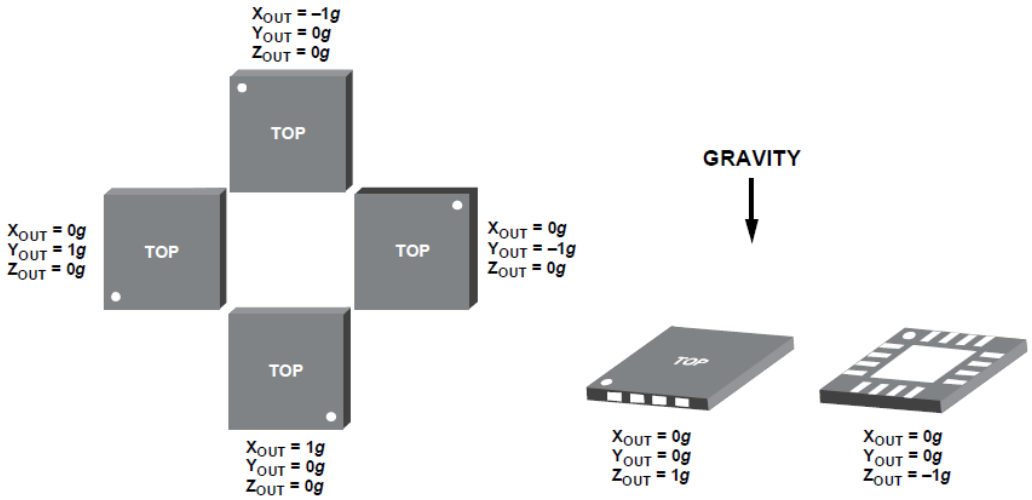


Fig. 55 comportamiento del ADXL335 de aceleración estática

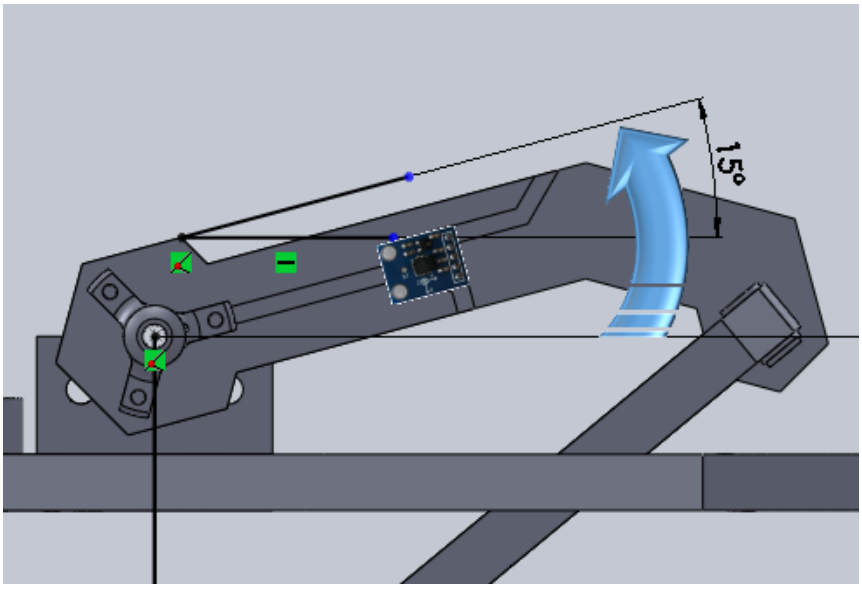


Fig. 56 Acelerómetro GY-61 montado sobre el eslabón Brazo

3.6 Selección de Controlador

El controlador digital debe tener la velocidad de procesamiento para atender las señales proporcionadas por los sensores y emitir señales en respuesta para accionar los actuadores, por lo que el controlador además de tener gran velocidad de procesamiento debe contar con la cantidad de pines de entrada / salida que requiere el total de actuadores y sensores.

De los controladores sobre los cuales se tiene conocimiento y cumple con las necesidades planteadas es la placa de evaluación de Texas Instruments que se detalla a continuación.

3.6.1 Tarjeta electrónica de Texas Instruments TIVA - TM4C123G

El LaunchPad *EK-TM4C123GXL* es una placa de evaluación de Texas Instruments, la cual cuenta con el micro controlador *TM4C123GH6PM* basado en la arquitectura *ARM Cortex-M4F*; la placa de evaluación es de bajo costo, posee un reloj configurable hasta 80MHz, la placa cuenta con 40 pines distribuidos en 2 filas dobles de conectores extendidos tipo hembra-macho (nombrados J1, J2, J3, J4), que conectan a los pines GPIO del controlador (para conectar hardware adicional), los cuales varios se consideran como periféricos (UART, I2C, SPI) según la distribución mostrada en la Fig. 57 obtenida del sitio oficial de energía https://energia.nu/pinmaps/ek-tm4c123gxl/; la placa también cuenta con un indicador de encendido; LED RGB de gran brillo y programable; interruptor de selección de conector USB (con distinto propósito y alimentación); par de interruptores (botones dedicados al usuario); interruptor de reinicio (reset); bus serie universal (USB micro-B) principal y secundario, el cual permite acceder a diversas funciones para programar al controlador, además de proveer su alimentación, cuenta con una interfaz de depuración en circuito (ICDI); lo anterior descrito se muestra en Fig. 58. [20]

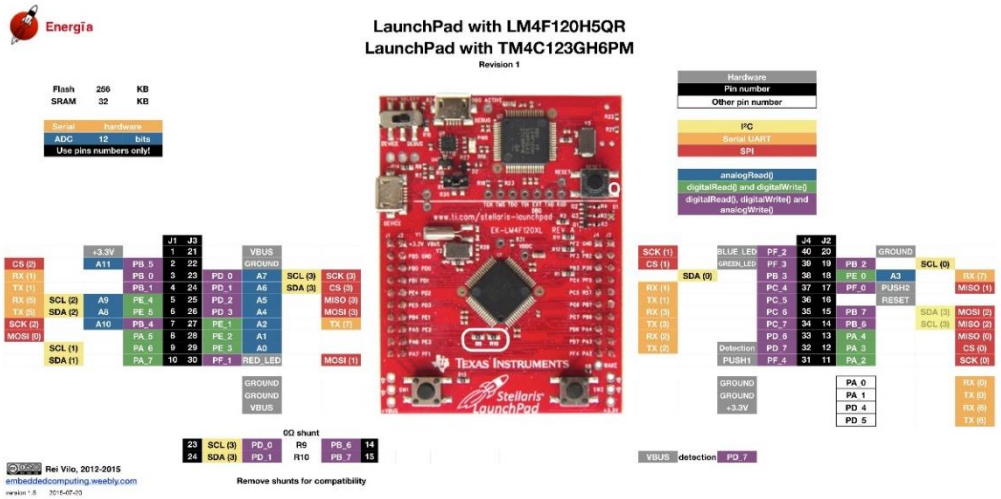


Fig. 57 Mapa de pines GPIO y disposición de periféricos

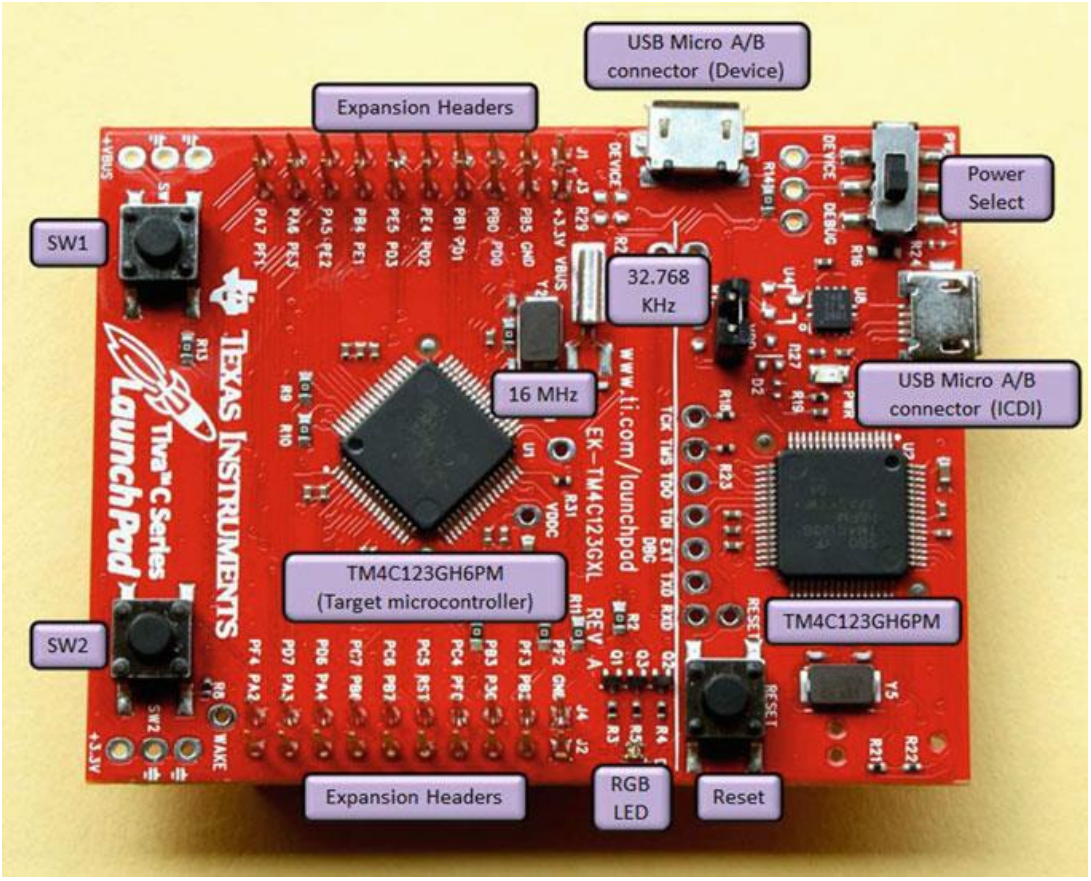


Fig. 58 Elementos de la placa de evaluación Tiva TM4C123G LaunchPad

La placa consta de dos micro controladores; el primero ejecuta las instrucciones de la programación realizada por el usuario y tiene la interacción con el hardware de la placa de desarrollo; el segundo esta pre programado para actuar como interfaz de depuración (ICDI), permitiendo monitorear al primero bajo ciertas condiciones.

Existen placas de expansión de hardware conocidas como placas hijas (shields), las cuales generalmente se utilizan para desarrollar sistemas embebidos, la conexión de dichas placas se realiza apilando una placa sobre otra considerando la disposición de los conectores (headers) y su numeración; el ensamble de shields permite integrar hardware de soporte a la placa de evaluación sin realizar cableados, además de agregar en poco espacio elementos de entrada o salida como pantallas LCD, sensores, teclados, entre otros; los diseños de las placas pueden ser prefabricados (**BoosterPacks**) o personalizados (creados por el usuario). [20]

Para referiremos a la placa de desarrollo anteriormente descrita utilizaremos el nombre de *TIVA*

ARM Cortex – M4 es un procesador de 32 bits, de bajo consumo de energía y alto rendimiento de procesamiento con arquitectura Harvard, el cual cuenta con: unidades de cálculo que mejoran el rendimiento de las operaciones lógicas/aritméticas; un conjunto de instrucciones amplio; opciones de energía que permiten al procesador entrar en distintos modos de suspensión, los cuales puede interrumpir y cambiar a modo normal al detectar alguna interrupción; la arquitectura del procesador permite ser monitoreado en tareas de depuración; sus bancos de memoria y rutas de datos son de 32 bits; tiene una latencia de interrupción baja, siendo posible atender hasta 240 interrupciones externas; además cuenta con un set de instrucciones extenso, la Fig. 59 muestra la arquitectura Cortex-M4. [21]

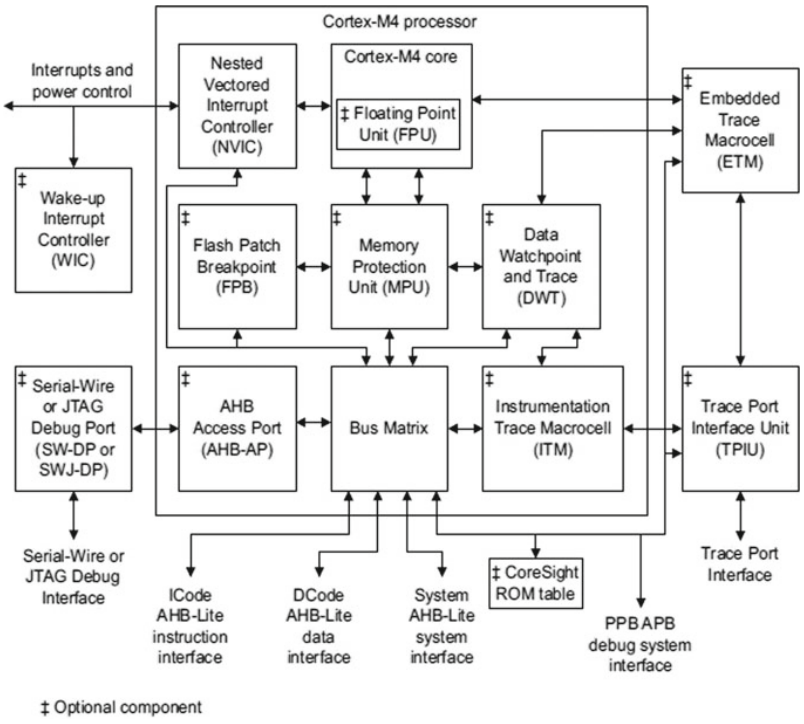


Fig. 59 Diagrama de la arquitectura Cortex-M4

Se optó por realizar la programación de la tarjeta en el entorno de desarrollo integrado (IDE) llamado *Energía* mostrado en la Fig. 60, el cual utiliza un lenguaje de programación con funciones y bibliotecas muy sencillas de implementar. [22]

Fig. 60 Logo y entorno de desarrollo de Energía

4 IMPLEMENTACIÓN DEL SISTEMA

4.1 Caracterización de Componentes Seleccionados

La caracterización es necesaria ya que el comportamiento de los componentes físicos puede variar respecto a las especificaciones dadas por el fabricante en las hojas de datos de dichos elementos, para conocer el margen de error es necesario caracterizar cada uno de los componentes electrónicos; generalmente se desea que exista una relación constante entre la magnitud física medida y la respuesta del componente electrónico, de manera que una variación en la magnitud física corresponda proporcionalmente a una variación constante de voltaje o corriente como es el caso de los sensores; para conocer el comportamiento de un componente se deben realizar pruebas con la finalidad de comparar los resultados de la medición de la respuesta del elemento seleccionado ante un estímulo con los resultados de algún material de referencia de cierta jerarquía que se encuentre correctamente calibrado [23]; posteriormente mediante métodos numéricos obtener la ecuación característica que rige el comportamiento del elemento electrónico [24].

Con la finalidad de obtener un registro de las lecturas adquiridas por los sensores y visualizar dicha información, se recurre a la creación de interfaces programadas dentro del IDE *Processing* mostrado en la Fig. 61, el cual está enfocado a la programación de entornos visuales de carácter artístico [25]; la biblioteca de nombre *ControlP5* provee al IDE con objetos que sirven como controles o indicadores gráficos, con los cuales se puede interactuar, además cuenta con una amplia variedad de elementos y ejemplos de cada uno como se muestra en la Fig. 62 (obtenida de http://www.sojamo.de/libraries/controlP5/#features), su integración al código en Processing resulta muy sencilla.

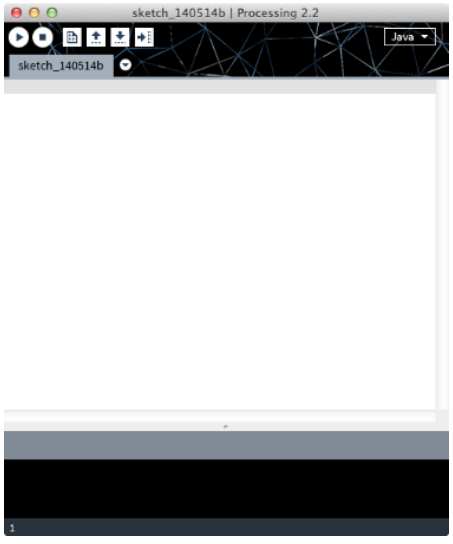


Fig. 61 Entorno de programación

Fig. 62 Controles e indicadores de ControlP5

4.1.1 Caracterización de actuadores

Con el fin de precisar la relación de la caja reductora de engranes planetarios se construye un banco de pruebas para el motor, el cual requiere diseñar y cortar en madera un soporte e imprimir en 3D una pieza simple en PLA que embone con la flecha de la caja de engranes, con espesor suficiente para lograr interrumpir el paso de la luz infrarroja del sensor de cerradura.

Se carga a la Tiva un programa encargado de emitir vía serial con una velocidad de transmisión de 9600 baudios en cada interrupción del sensor de cerradura, la información de un contador que incrementa de acuerdo al número de pulsos que se emiten hacia la etapa de potencia DRV8825, para mover la flecha del motor nema 17 con caja reductora, por lo tanto se obtiene una lectura del contador en cada vuelta que complete la flecha de la caja de engranes planetarios del motor, dicha información se muestra en el monitor serial del IDE de Energía [26].

Al probar el uso de los motores a pasos en conjunto con los DRV8825 controlados por la Tiva, la cual se encarga de emitir los pulsos de dirección y paso, correspondientes para controlar los motores [27]; provocó problemas en los puertos USB de la computadora, el cual suministra la energía necesaria a la Tiva y sirve de comunicación para obtener los resultados vía serial.

El problema se resolvió opto acoplando las señales de control de la Tiva hacia los DRV8825 (Dir, Step), ya que de esta forma se separan las tierras de la fuente destinada a la alimentación de los motores y la tierra del puerto USB que alimenta al controlador (Tiva), para lograrlo se hizo uso de dos circuitos integrados LTV847, completando el número de señales requeridas para el control de los actuadores; la configuración de opto acoplamiento requiere conectar en serie al pin de señal del controlador una resistencia de protección de 220 ohm para el led a la entrada del opto acoplador, que estimula la base del opto transistor, el cual requiere de un transistor de tipo NPN conectado en arreglo Darlington, amplificando la señal de salida opto acoplada, las conexiones anteriores son requeridas para cada señal destinada al control de los actuadores, el opto acoplamiento de las señales permite usar de manera independiente ambas fuentes [28]; la primer fuente de alimentación se provee vía USB (5v) para energiza al controlador y al sensor; la segunda fuente entrega 12 Volts a 5 Amperes, la cual se debe conectar al pin de alimentación de potencia del DRV8825 (VMOT), además se requiere de un regulador de voltaje de 5 volts para alimentar la parte lógica del DRV8825 y alimentar el colector del arreglo Darlington de cada opto acoplador; la Fig. 63 muestra el diagrama de bloques del sistema; la placa prototipo que se empleó en la caracterización con el arreglo anteriormente descrito se muestra en la Fig. 64; el sistema de pruebas conectado a la placa prototipo se muestra en la Fig. 65.

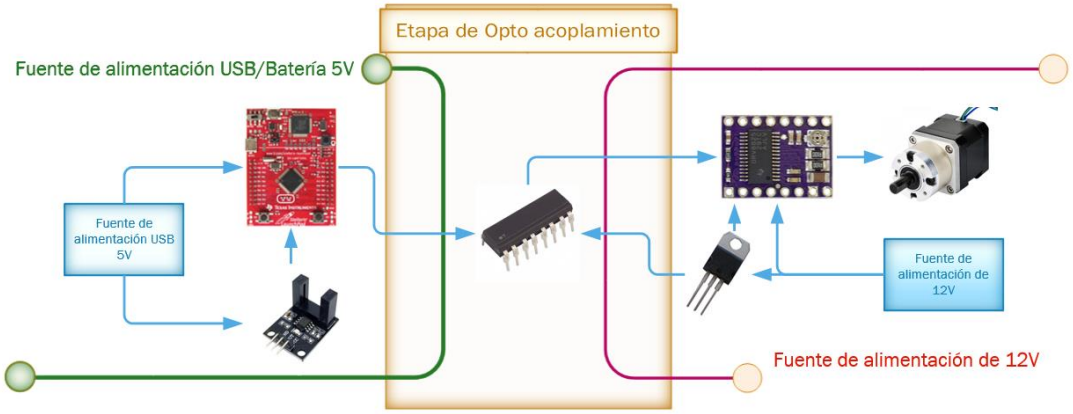
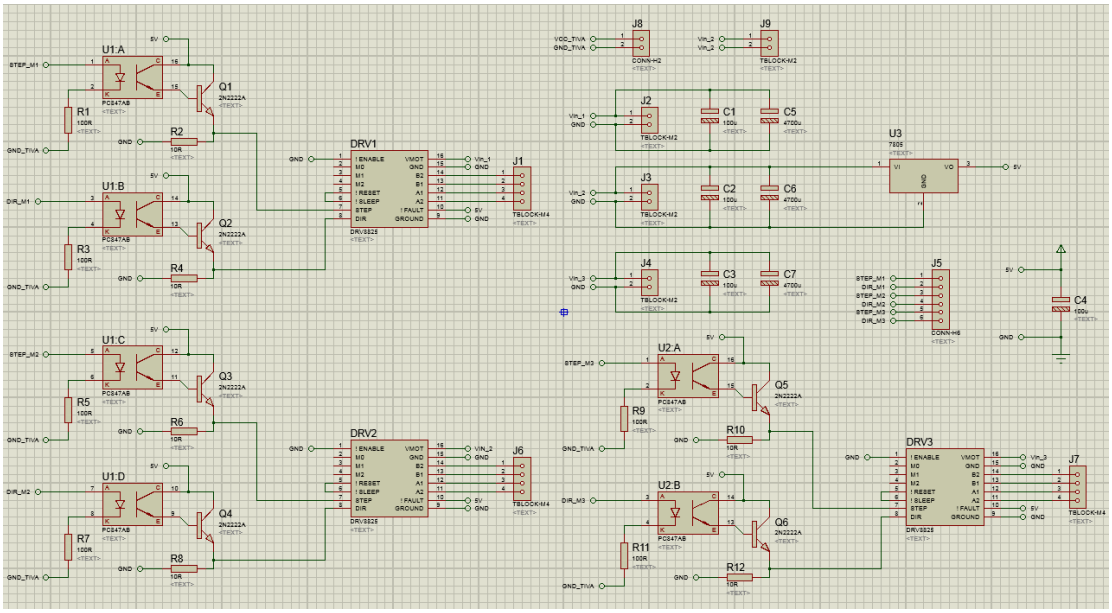
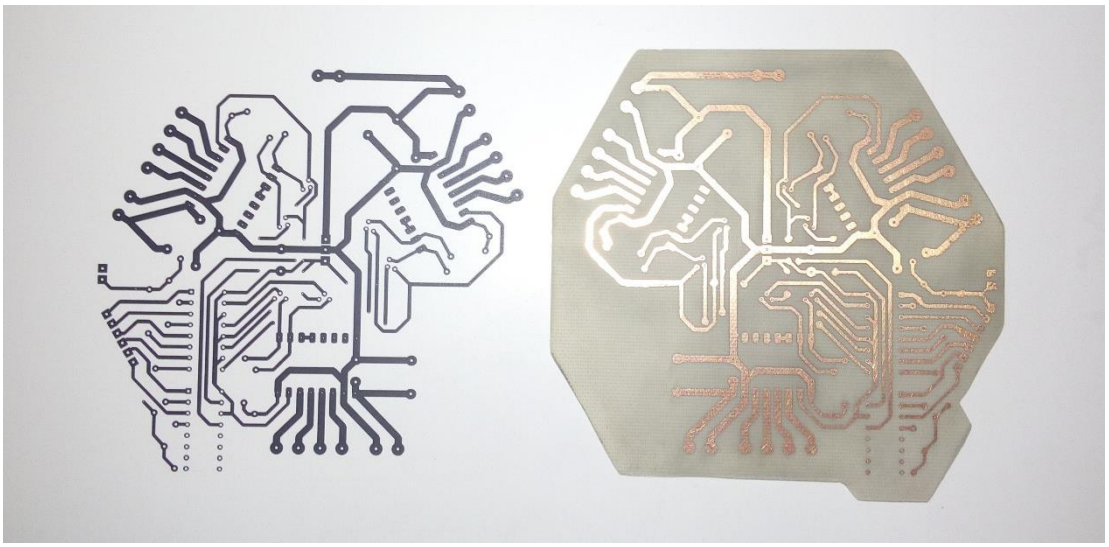


Fig. 63 Diagrama a bloques del sistema de caracterización del actuador





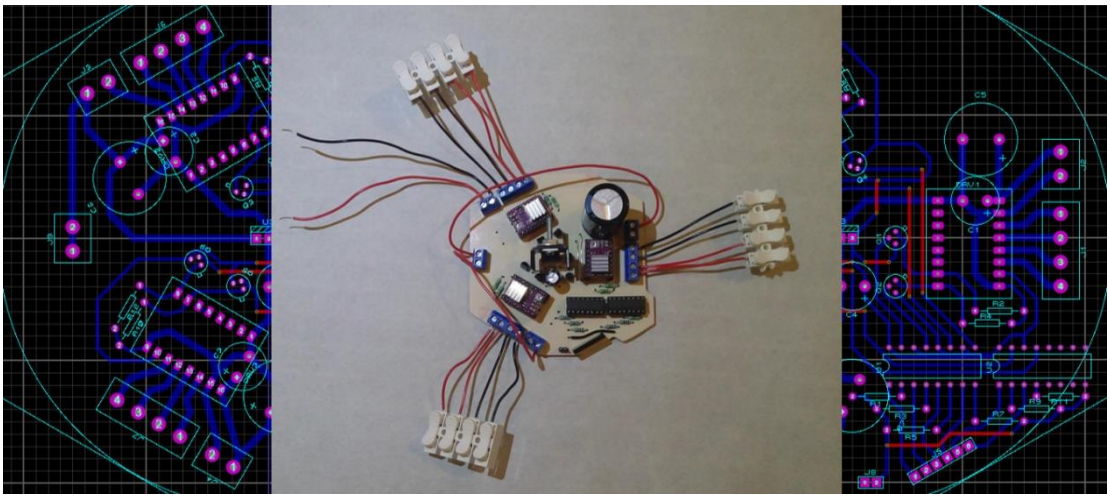


Fig. 64 Esquemático y diseño PCB de prototipo del módulo de potencia

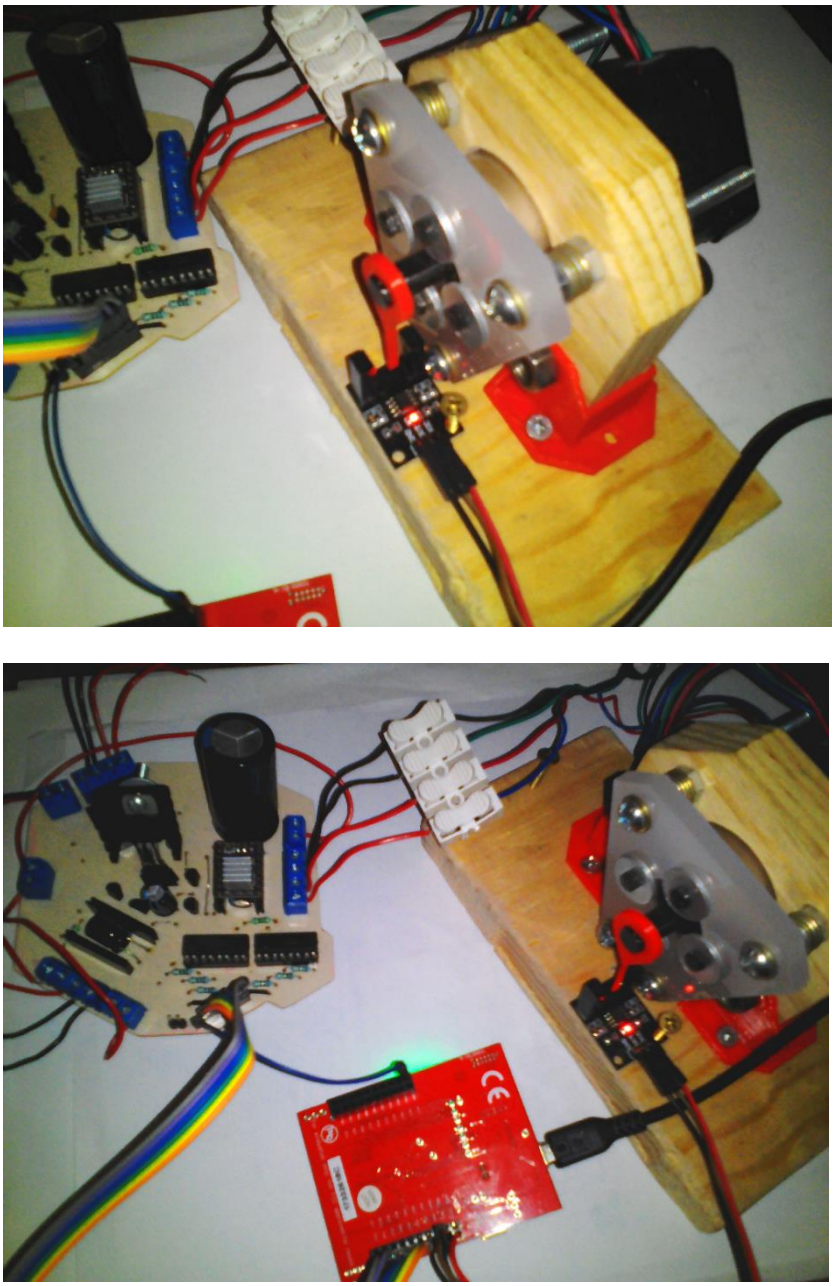


Fig. 65 Sistema de pruebas para caracterización del actuador

La información de la prueba se visualiza utilizando el monitor serial del IDE de Energía, los datos obtenidos indican que se requieren 1036 pasos para que la flecha del sistema de engranes planetarios acoplado al motor NEMA 17 logre dar un giro completo; la relación de transmisión 5.18 : 1 del actuador se obtiene al dividir 1036 entre los 200 pasos que el motor sin caja de engranes necesita para completar una vuelta; la resolución angular de paso del actuador se obtiene al dividir 360 grados de una circunferencia entre los 1036 pasos necesarios para completar una vuelta, por lo tanto el desplazamiento angular en cada paso es de 0.34749 grados.

4.1.2 Caracterización de sensores

Para realizar la caracterización del sensor se requiere ensamblar un banco de pruebas.

De acuerdo con la estrategia de montaje del módulo GY-61 sobre el eslabón *Brazo* del robot, utilizando el ADC con 12 bits de resolución de la Tiva se tomaron lecturas analógicas de las señales de salida *X, Y, Z*, del módulo para identificar la señal que detecta el ángulo de inclinación del eslabón [29]; la señal *Y* proporciona dicha información, respecto a la orientación de montaje del módulo sobre el eslabón, al analizar las lecturas, la señal del sensor mostro un comportamiento no lineal, para obtener su ecuación característica se consideró lo siguiente.

Ya que el motor nema 17 sin caja reductora de engranes cuenta con un desplazamiento angular constante de 1.8 grados por paso, según su hoja de especificaciones, se decidió utilizar como material de referencia para caracterizar el modulo GY-61.

La prueba de caracterización del sensor GY-61 tiene como finalidad obtener lecturas del sensor en cada una de las posiciones angulares que permite alcanzar la resolución de paso del motor, con los DRV8825 configurados a pasos completos, en un rango de 0 a -90 grados y de 0 a 90 grados; se genera una interfaz que registra los valores en una tabla, la cual relaciona la posición angular de la flecha del motor y las lecturas obtenidas por el ADC de la Tiva (estimulo-respuesta); la prueba requiere montar el módulo (acelerómetro) en el extremo de la flecha del motor, lo cual se logra mediante una pieza impresa en 3D [23].

Para obtener el registro y apoyo visual de las lecturas del módulo GY-61, así como un control sobre la posición angular del motor, se genera una interfaz gráfica en Processing con un entorno visualmente simple como muestra la Fig. 66, la cual se constituye por controles e indicadores en los cuales se muestra el valor de los datos obtenidos por el ADC del controlador (Tiva), además se encarga de recibir del usuario la solicitud para posicionar angularmente al motor sobre un ángulo especifico (múltiplos de 1.8, debido al paso del motor), para posteriormente indicar al controlador la dirección y el número de pulsos que debe emitir a los motores; la interfaz cuenta con botones y comandos para agilizar la prueba, el primero ejecuta instrucciones que permiten emitir la solicitud del usuario de posición angular del motor hacia el controlador; el segundo permite guardar la información de la posición angular actual del motor y la información de los datos muestreados del sensor obtenidos por el controlador (Tiva).

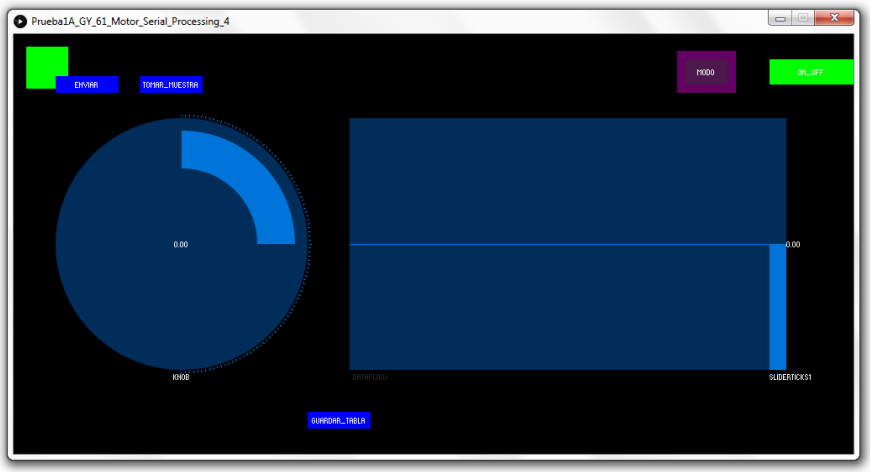


Fig. 66 Interfaz de adquisición de datos para caracterización del módulo GY-61

El programa que se carga en la TIVA se encarga de tomar lecturas constantes periódicamente de la señal *Y* del módulo GY-61 mediante el convertidor analógico digital que tiene integrado, envía los datos del ADC al puerto serie con una velocidad de comunicación de 9600 baudios, además, por la misma vía recibe información a modo de instrucciones con el número de pulsos que debe emitir a los drivers (Dir, Step) de la placa prototipo, para accionar correctamente al motor NEMA 17 y alcanzar la posición angular solicitada según lo indique el usuario dentro de la interfaz en Processing, el diagrama de bloques del sistema se muestra en la Fig. 67.

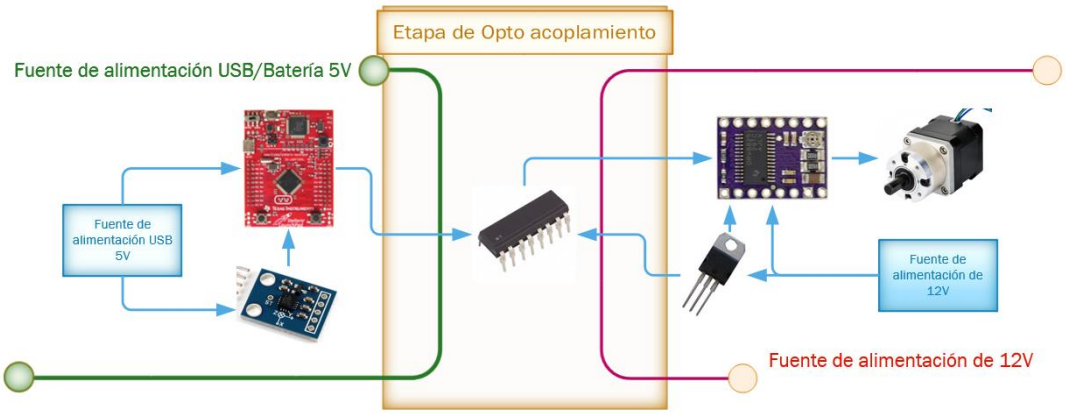


Fig. 67 Diagrama a bloques del sistema de caracterización del sensor

Al terminar de obtener los datos y dar clic al botón *Guardar Tabla* de la interfaz genera un archivo de Excel de tipo *csv* con el contenido de los datos registrados en cada posición angular de la flecha del motor [30]; teniendo un total de 100 posiciones del motor diferentes de -90 a 90 grados y al menos 3 lecturas del ADC en cada una de las posiciones, ya que al obtener los datos de manera experimental es necesario considerar la posibilidad de errores [24].

Una vez que se generó el archivo de Excel con las columnas de datos, el archivo guarda en Excel con un formato diferente, se convierte a *texto delimitado por tabulaciones*, el archivo de texto se guarda en la carpeta (Documentos → MATLAB) ya que dicho archivo será leído e ingresado a MATLAB haciendo uso del código que se muestra en la Fig. 68.

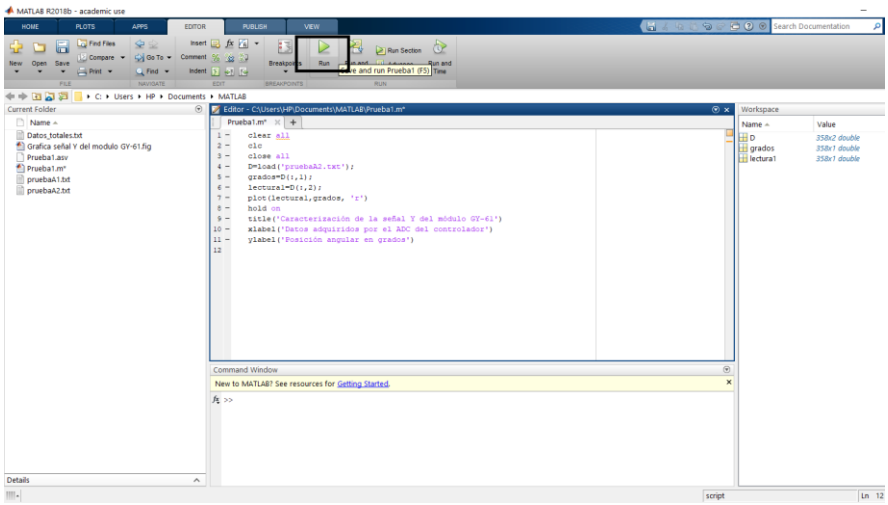


Fig. 68 Entorno de MATLAB y código de programación

Al correr el código, la información del archivo de texto es introducida en MATLAB como una matriz de dimensión n x 2 (filas x columnas), los datos de cada columna se grafican en cada eje, logrando obtener una gráfica que muestra la relación estimulo–respuesta, con los encabezados de la variable medida *datos adquiridos por el ADC del controlador*, variable real *posición angular en grados*; el comportamiento no lineal de la señal Y del módulo GY-61 se muestra en Fig. 69.

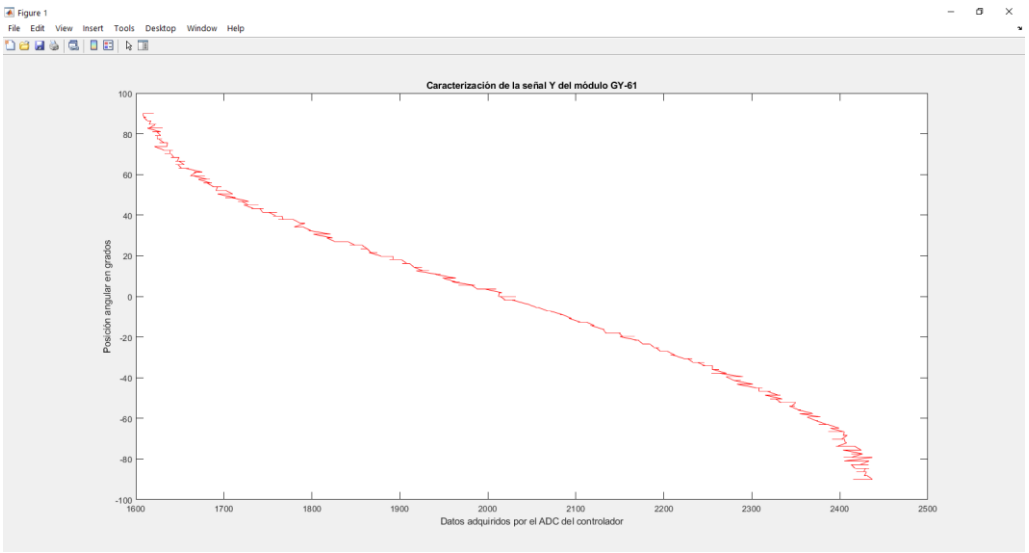


Fig. 69 Grafico en MATLAB con valores obtenidas por la interfaz en Processing

El despliegue de la figura que muestra la gráfica de la información tabulada, contiene un panel de herramientas, entre sus opciones, se encuentra la llamada *Basic Fitting*, al dar clic en dicha opción se abre una ventana con un grupo de opciones que permiten obtener la ecuación característica del módulo GY-61 que describe el comportamiento de la señal *Y* visualizado en la gráfica, como se muestra en la Fig. 70.

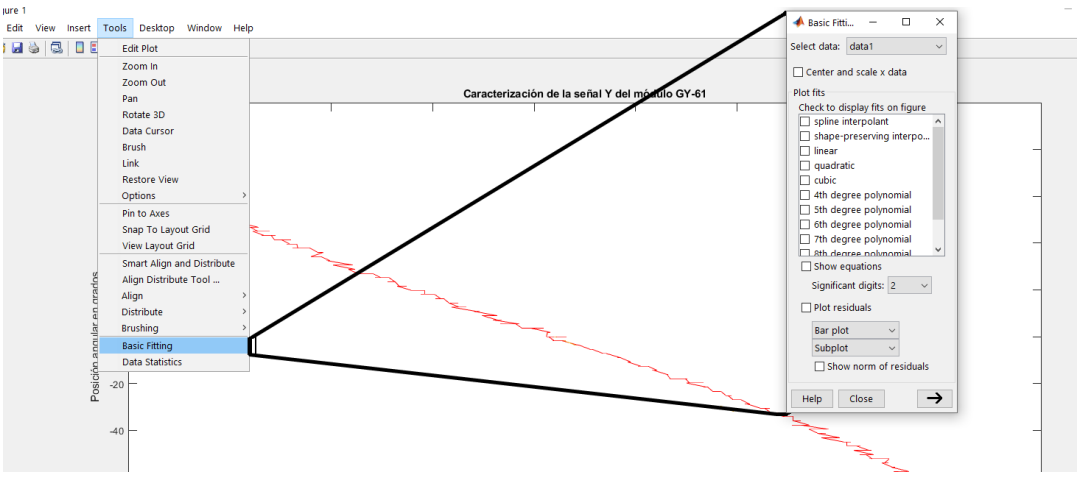


Fig. 70 Ventana de ajuste básico desplegado del panel de herramientas

Al seleccionar alguna de las opciones que se muestran en la ventana, se obtiene un trazo de acuerdo con el tipo de función y orden **m** del polinomio seleccionado; al marcar las primeras tres opciones y analizar el trazo, se puede observar en la Fig. 71, que es necesario considerar otras opciones de polinomio con orden superior, ya que el trazo muestra mucho error en las zonas señaladas.

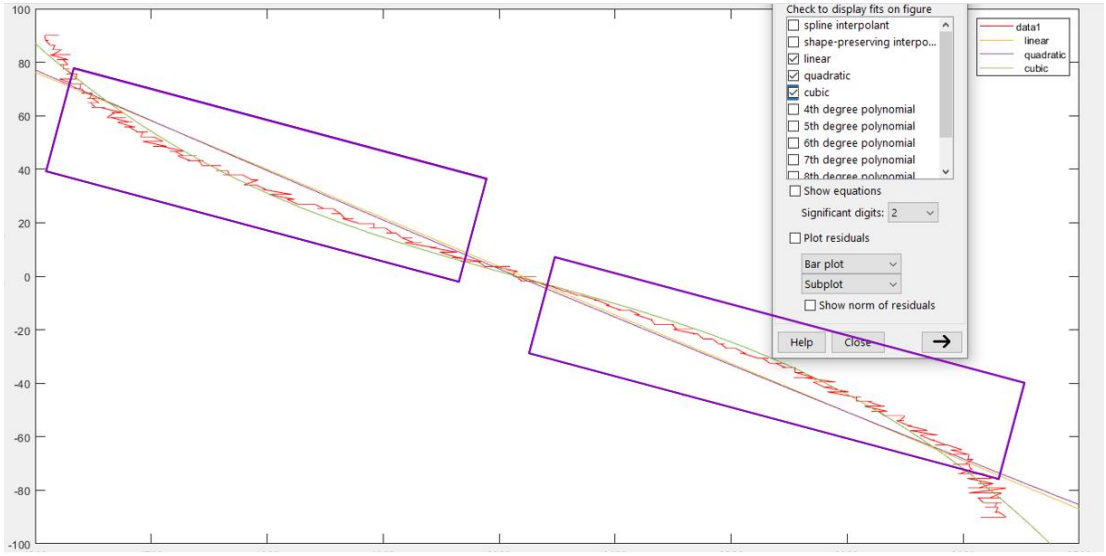


Fig. 71 Errores de coincidencia en las opciones de trazos lineal, cuadrático y cubico

El error es menor, pero sigue presente de igual manera con siguientes las dos opciones siguientes como se muestra en Fig. 72, las zonas marcadas señalan el error ya que los trazos no tocan algunos puntos que describen el comportamiento de la gráfica, entre mayor sea el orden o grado del polinomio, el trazo logra ajustarse mejor al comportamiento de la señal obtenida del GY-61

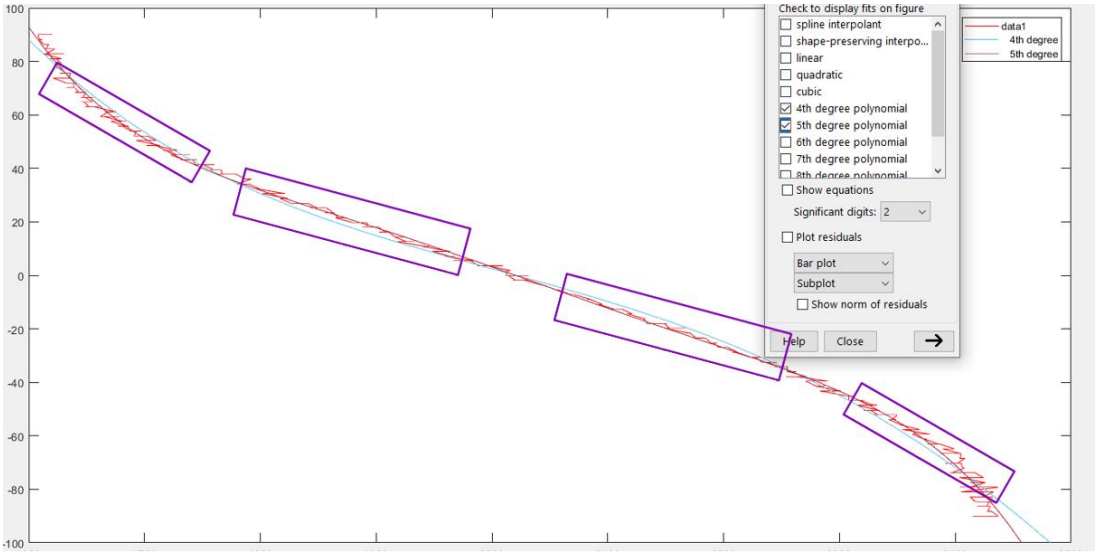


Fig. 72 Errores de coincidencia en graficas del polinomio de cuarto y quinto grado

El trazo del polinomio de sexto orden describe de manera óptima el comportamiento de la señal, se observa en la Fig. 73 el ajuste del trazo entre los puntos que conforman a la gráfica original, de manera que se encuentra la ecuación característica de la señal Y del sensor GY-61; al dar clic en el botón con la flecha que apunta a la derecha (siguiente) se muestran las constantes necesarias y la plantilla para programar el polinomio de grado 6.

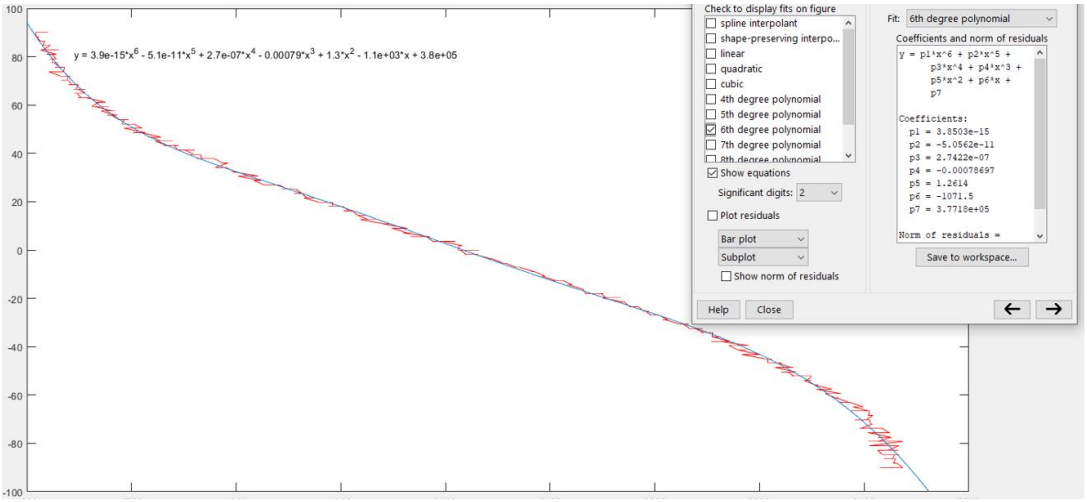


Fig. 73 Polinomio de sexto grado coincidente a la gráfica original de la señal del GY-61

Al implementar el polinomio de sexto grado (ecuación característica) en el controlador, establecer el cero usando un nivelador, tomar lecturas del ADC hacia los extremos del sensor y transmitir los datos a la interfaz en Processing, guardando en tablas los datos e introduciendo en MATLAB la información para graficarla (Ver Fig. 74), se obtiene el comportamiento lineal del ángulo dado por la ecuación característica del sensor respecto a la posición angular del motor NEMA 17 (material de referencia); el sistema de pruebas para la caracterización del acelerómetro modulo GY-61 acoplado al eje del motor a pasos se muestra en Fig. 75.

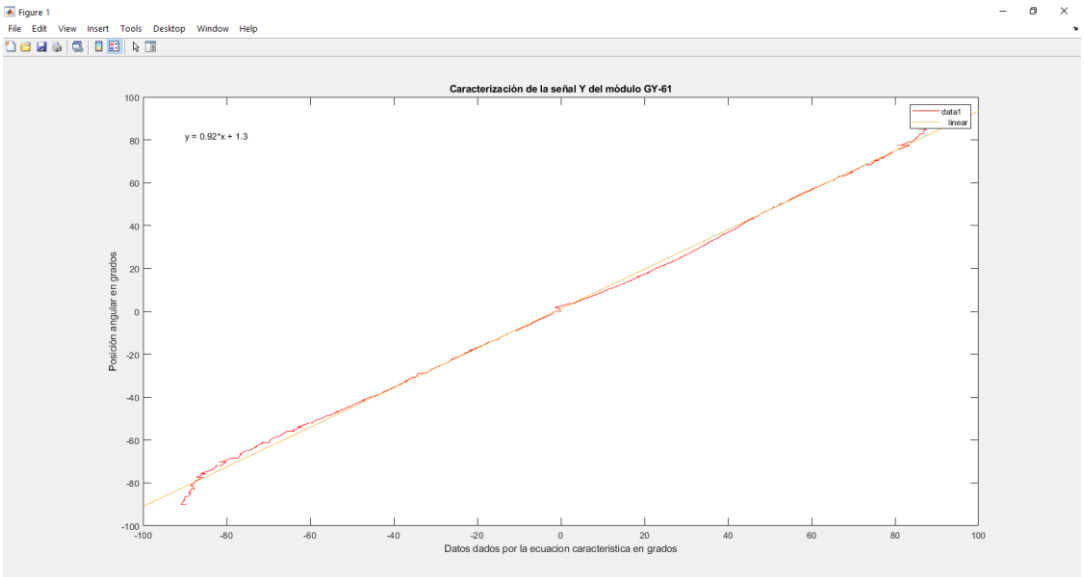


Fig. 74 Grafica con datos de la implementación del polinomio de sexto orden

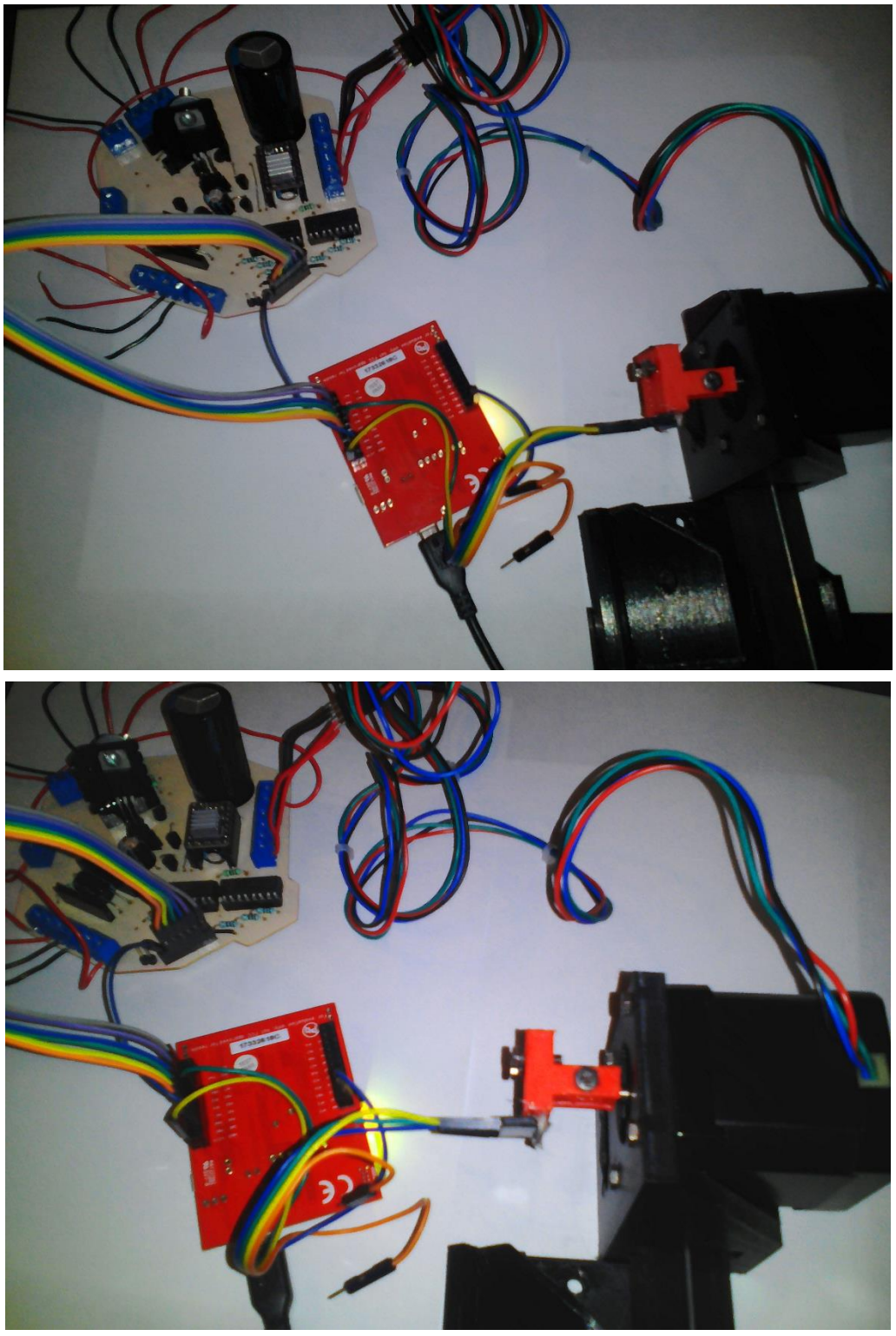


Fig. 75 Conexión del sistema de pruebas para caracterizar el sensor GY-61

4.1.3 Manufactura de hardware y elementos de soporte

Una vez probados y caracterizados los componentes electrónicos se propone la estructura que integra el subsistema electrónico del robot en un solo bloque (sistema embebido), complementando con el módulo de comunicación bluetooth HC-05 como se muestra en Fig. 76, para controlar a distancia el robot delta.

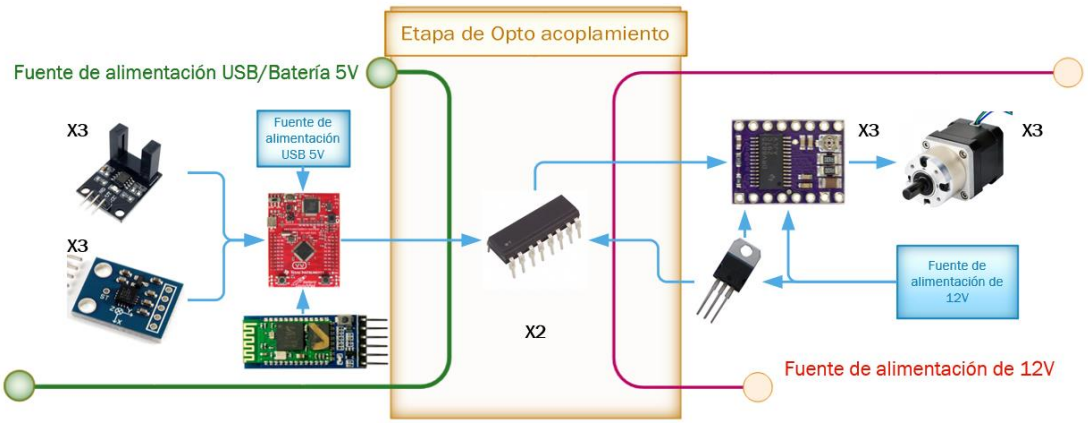


Fig. 76 Diagrama a bloques del subsistema eléctrico del Robot

Para lograr entender la interrelación entre los elementos que comprenden al sistema embebido, así como la importancia e implementación de cada elemento, el hardware que comprende al subsistema electrónico es dividido en 4 niveles de placas:

- Placa de reconocimiento
- Placa de control (Tiva)
- Placa de opto acoplamiento
- Placa de potencia para actuadores

El diseño del sistema embebido contempla la distribución de los componentes en los distintos niveles de placas como se muestra en la Fig. 77.

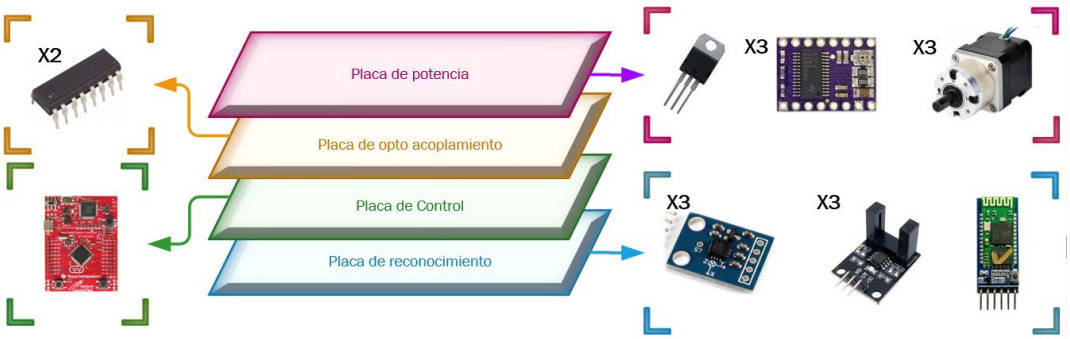


Fig. 77 Diagrama de interacción entre niveles de placas electrónicas

Se debe colocar un ventilador apuntando a los módulos DRV8825 ya que se calientan al estar en funcionamiento como se pudo notar en las pruebas de caracterización; se opta por usar el Shield CNC de Arduino como *placa de potencia*, debido a la distribución de drivers, integración de pines dedicados a la configuración de micro pasos y puertos de salida definidos hacia las fases del motor; al momento de montar los DRV8825 se deben identificar los pines A1, A2, B1, B2 del driver, los cuales deben estar orientados y tener inmediatamente a un costado los pines de salida del Shield CNC de Arduino, como muestra la Fig. 78.

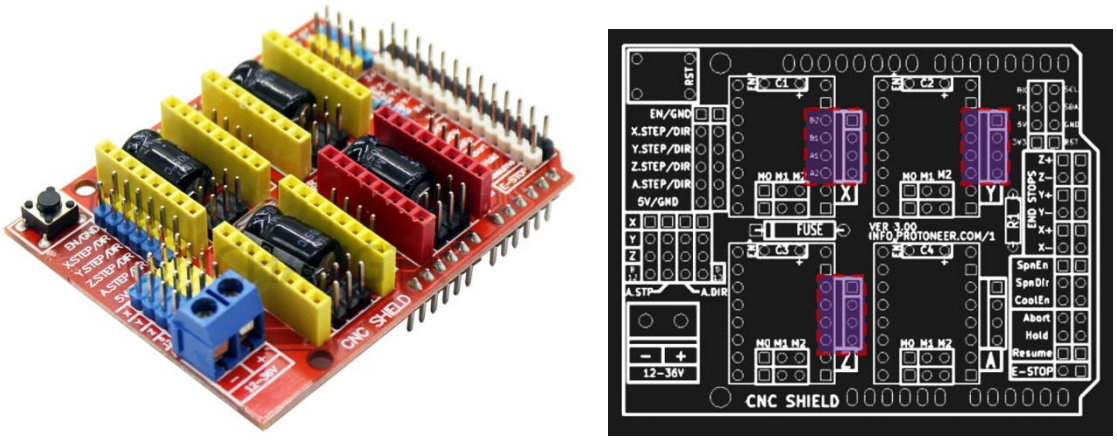


Fig. 78 Shield CNC Arduino (placa de potencia) y mascara de componentes con salidas señaladas

La *placa de opto acoplamiento* cuenta con los circuitos integrados LTV847 para opto acoplar las señales de control de la Tiva y direccionar las salidas opto acopladas a los pines dirección y paso para cada motor (X, Y, Z) marcados en el Shield CNC como muestra la Fig. 79.

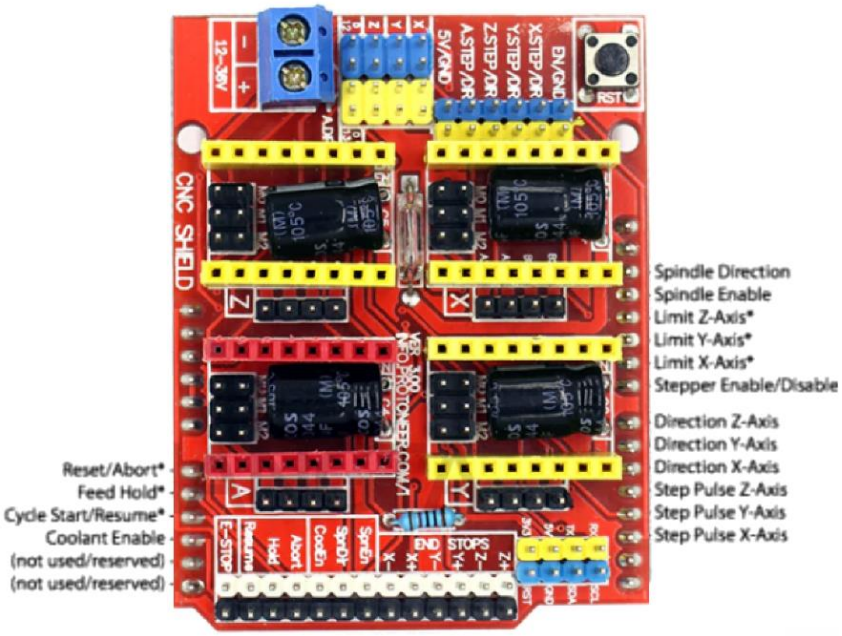
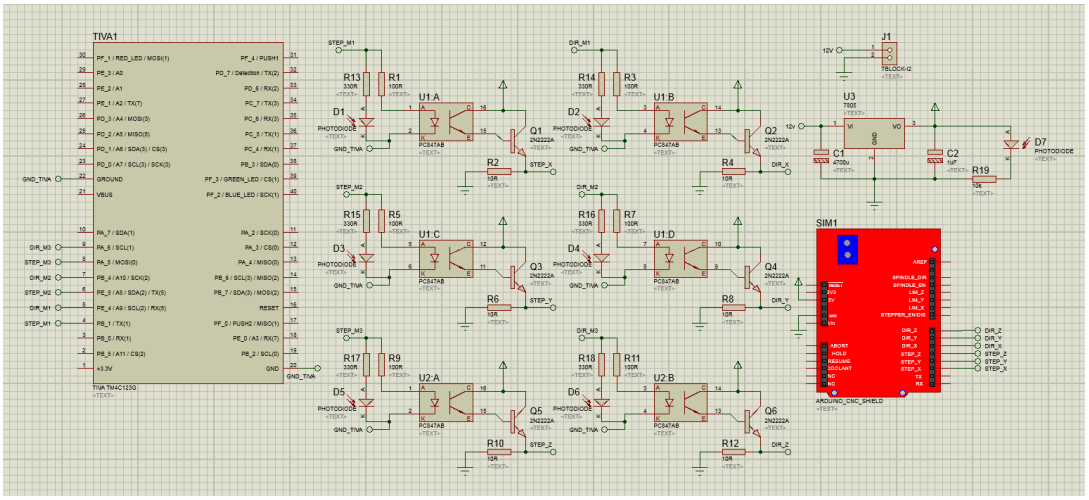


Fig. 79 Distribución de pines (Pin Map) del Shield CNC

El uso del Shield CNC Arduino obliga a que el diseño de la *placa de opto acoplamiento*, logre empatar la disposición de pines de la Tiva con la distribución de pines del Shield CNC como se muestra en la Fig. 80.



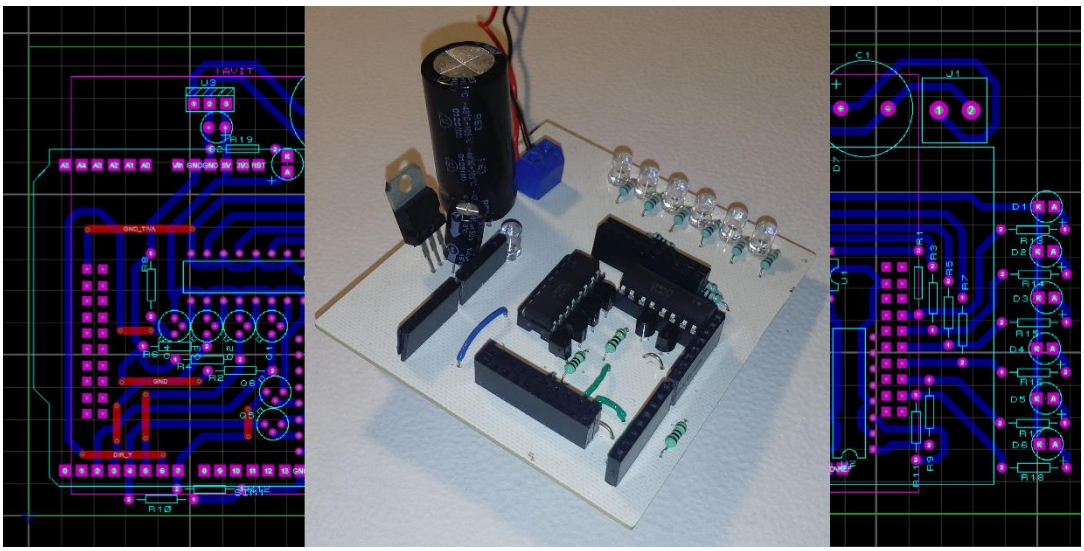


Fig. 80 Esquemático y diseño PCB de placa de opto acoplamiento

La *placa de reconocimiento* a través de los conectores tipo kk-254, borneras y pines hembra recibe las señales de los módulos GY-61, sensores ópticos de cerradura y bluetooth respectivamente, conectando dichas señales a los pines GPIO de la Tiva, la cual a su vez proporciona la alimentación de operación de 3.3Volts y tierra para energizar a los componentes.

La *placa de reconocimiento* cuenta con un jumper que permite seleccionar el UART Serial al cual va a responder la Tiva, cuando no está conectado responde a las señales del puerto USB (Serial) y cuando está conectado responde a las señales del Bluetooth HC-05 (Serial2), los pines de recepción y transmisión del módulo Bluetooth (BT) se conectan de manera cruzada con los pines UART Serial2 de la Tiva, *BT_TX* con *Tiva_RX2*(pin33) y *BT_RX* con *Tiva_TX2*(pin32) de la Tiva.

La *placa de reconocimiento* brinda el espacio requerido por los conectores de los sensores y por los comparadores de voltaje implementados con el circuito integrado *UA741CN* (OPAM) que sirven como etapa de acondicionamiento de la señal, las cuales son requeridas por cada opto transistor que forma parte del sensor de cerradura; la distribución de los elementos que se encuentran conectados a las borneras son el cátodo del led IR → naranja, tierra → blanco, colector → café, la Fig. 81 muestra la placa de reconocimiento.

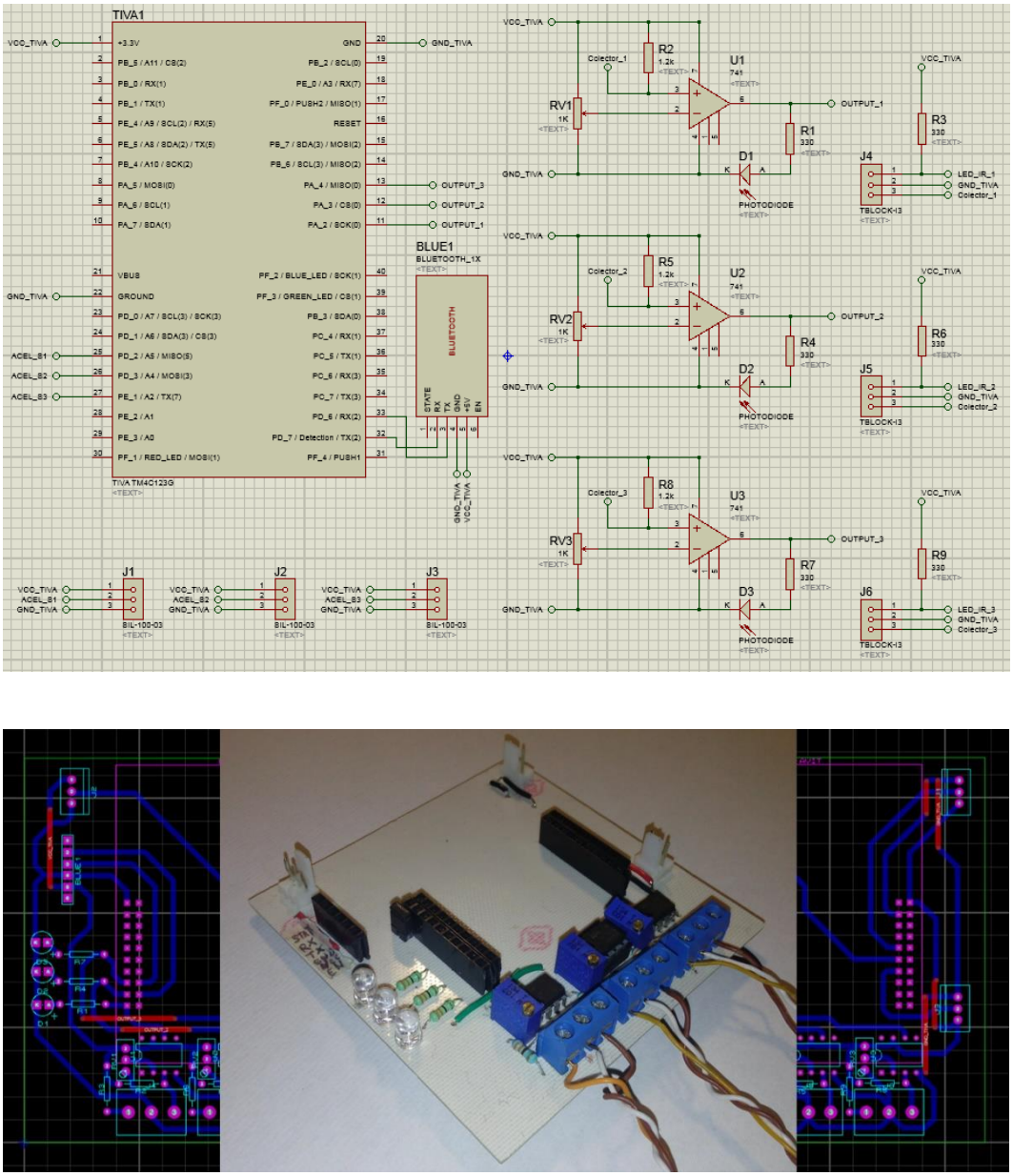


Fig. 81 Esquemático y diseño PCB de placa de reconocimiento

Las placas se ensamblan apilando cada placa sobre otra, iniciando por la *placa de reconocimiento* y terminando con la *placa de potencia*, de manera que en un tamaño aceptable se integra el sistema electrónico completo del robot delta como se muestra en la Fig. 82.

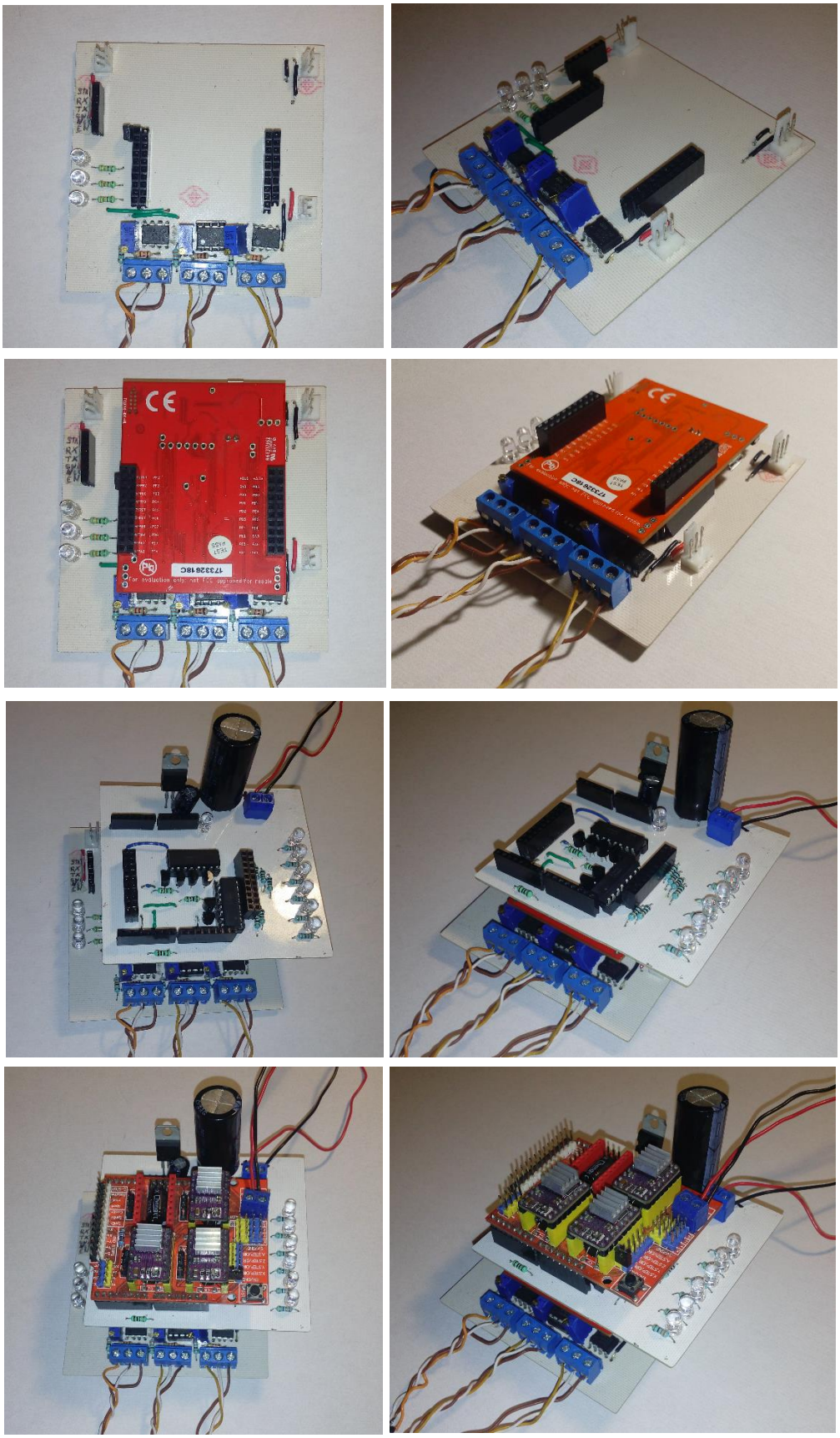


Fig. 82 Ensamble del sistema electrónico por nivel de placas

Se conecta el Bluetooth y los acelerómetros GY-61 a la *placa de reconocimiento* como se muestra en Fig. 83.

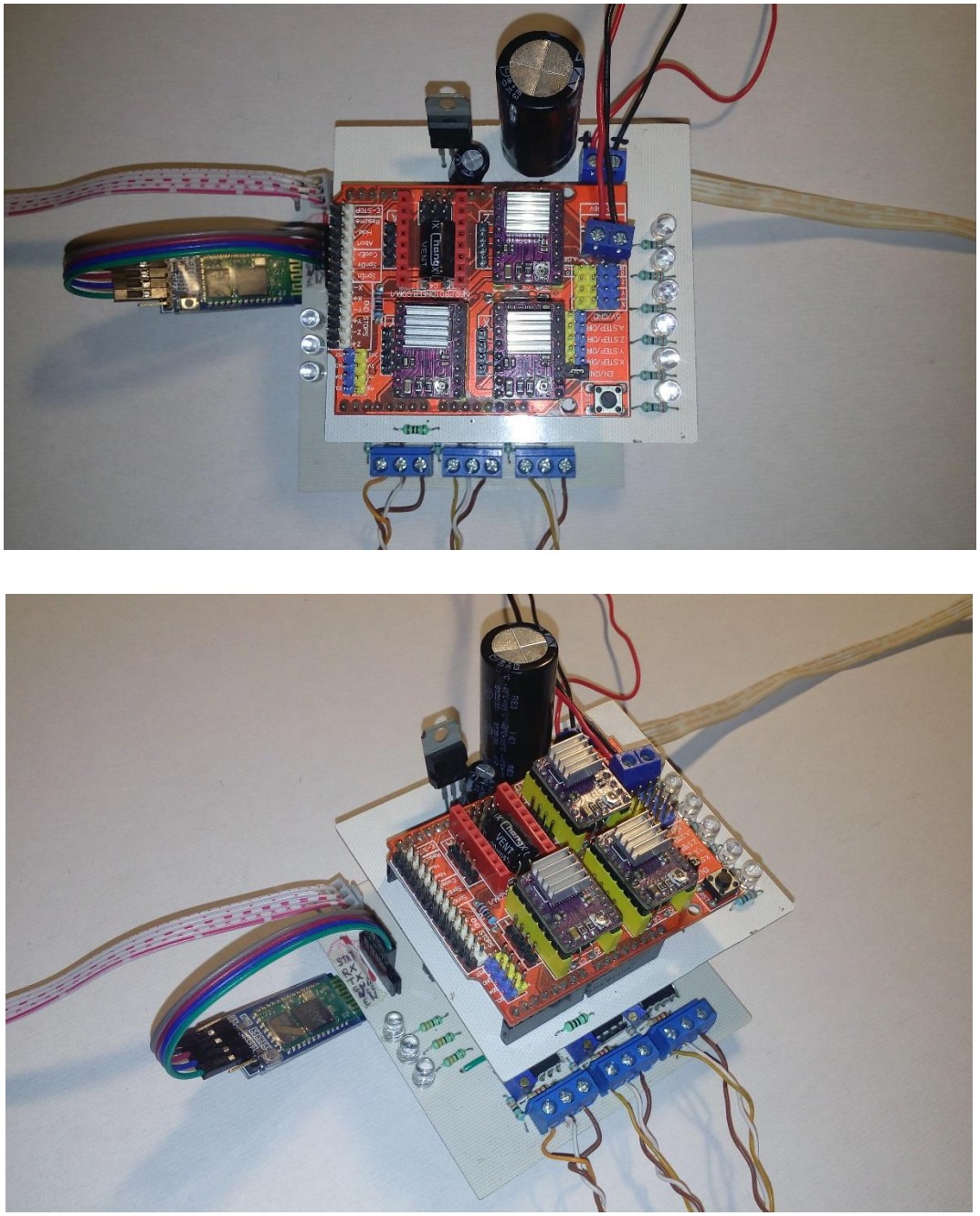


Fig. 83 Ensamble de hardware del sistema electrónico

Para encapsular, proteger y montar el sistema electrónico sobre el robot, se genera un soporte con piezas de acrílico, el cual se conforma por una pieza triangular de acrílico que cuenta con un ventilador de 12 Volts apuntando a los DRV8825 montados en la placa de potencia, con la finalidad de disipar el calor producido durante el funcionamiento del robot (Fig. 84); la pieza triangular y ventilador se montan sobre tres postes externos que brindan estructura y rigidez al conjunto de dos piezas de acrílico idénticas, las cuales se encuentran separadas por una tuerca de seguridad colocada en cada poste externo; cada poste externo es un tornillo de 3/16*3 ½ pulgadas (diámetro*largo), en medio de las dos placas de acrílico idénticas se insertan de manera uniforme y distribuida tres tornillos de ¼ * 3 pulgadas, los cuales sirven de conexión con los tornillos que ensamblan la base fija del manipulador, permitiendo montar el subsistema electrónico sobre el robot, la Fig. 85 muestra carcasa de acrílico ensamblado.

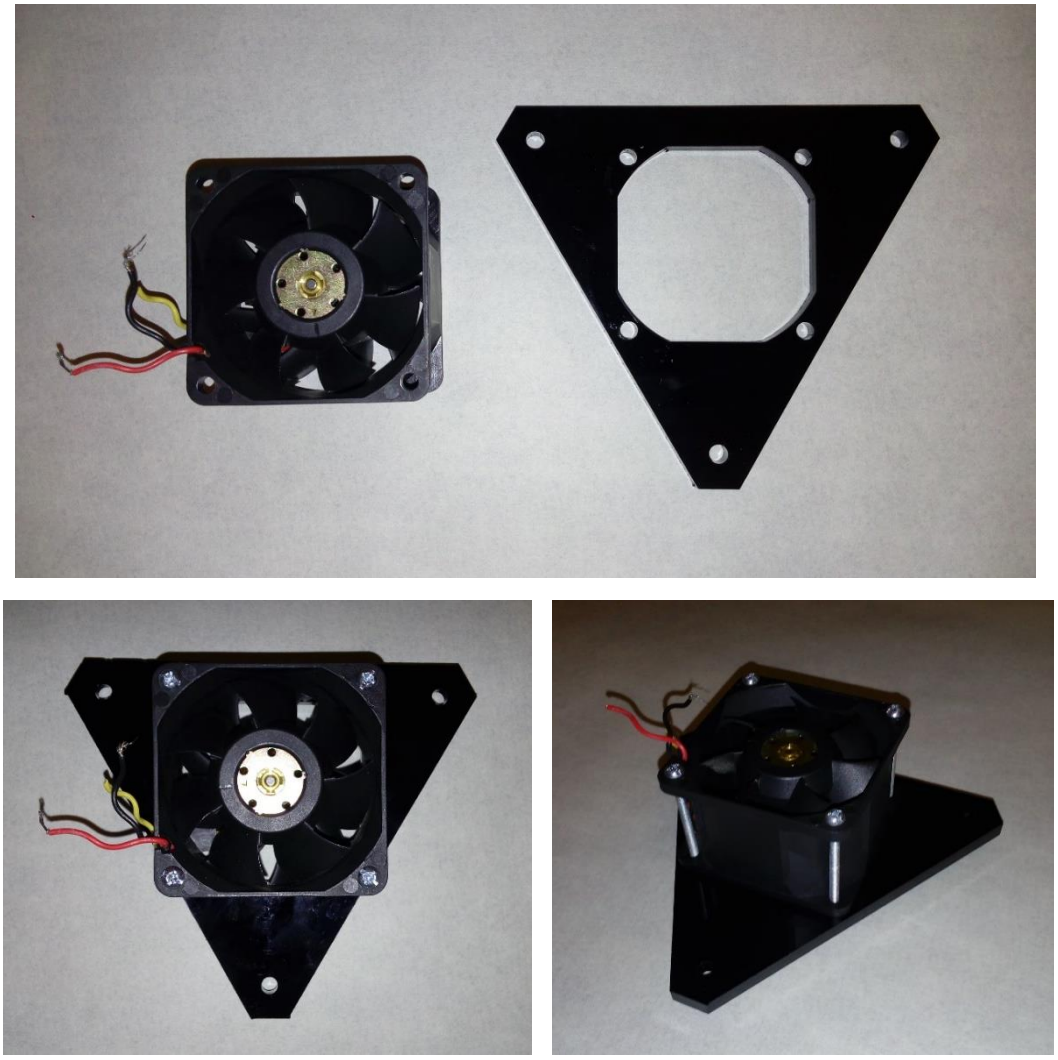


Fig. 84 Ventilador y pieza triangular de acrílico

Fig. 85 Ensamble del soporte de acrílico

Una vez montadas las placas del sistema electrónico en la protección de acrílico, es posible instalar el sistema embebido sobre el robot, permitiendo una distribución directa de las conexiones para recibir información de los sensores y transmitir señales de potencia a los actuadores, además se logra una ventilación de la etapa de potencia y la visualización de cada etapa del sistema robótico (Ver Fig. 86).

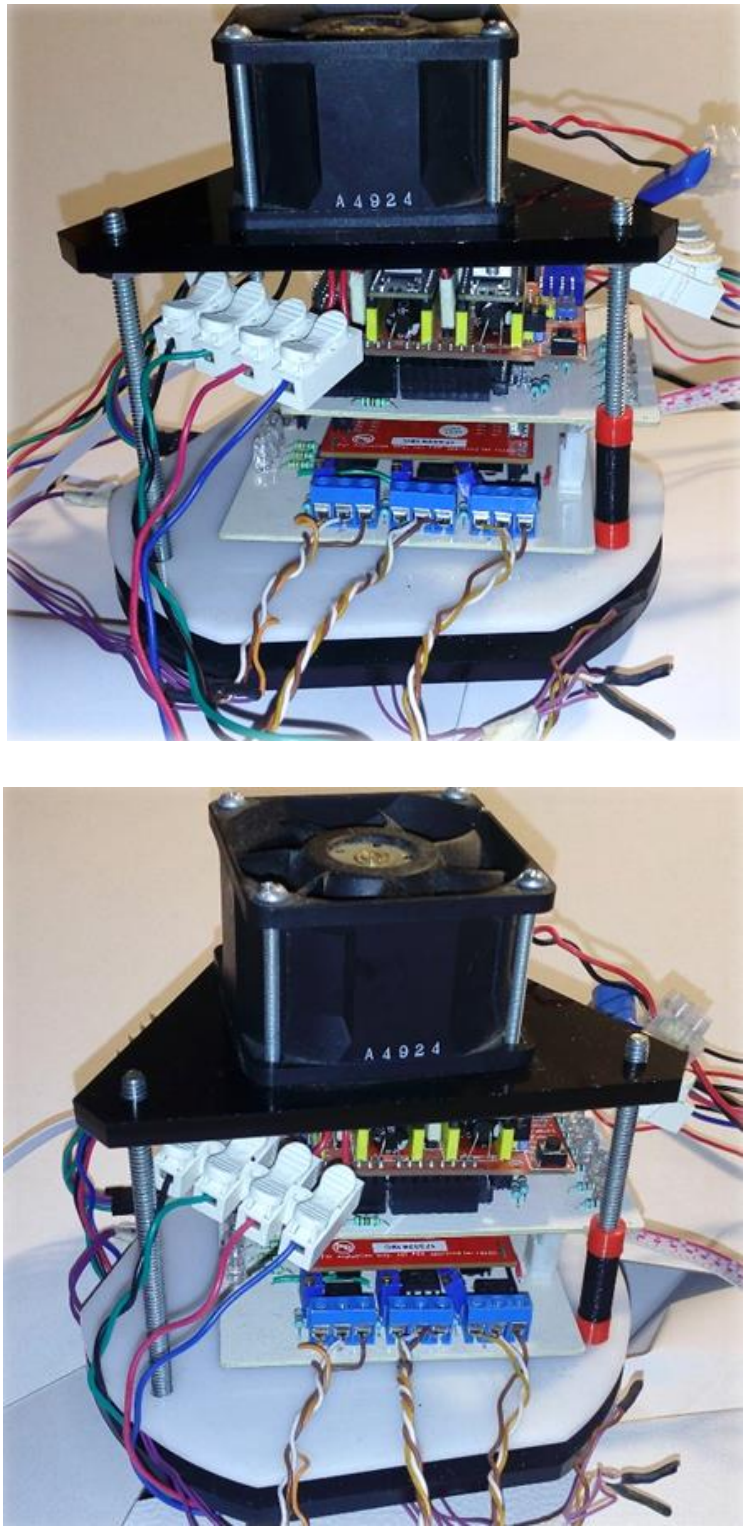


Fig. 86 Sistema embebido completamente integrado

4.2 Ensamble del Prototipo Piloto

Con la finalidad de exponer mejor el proceso de ensamble de manera organizada, se divide en boques, las piezas de la plataforma se ensamblan utilizando únicamente de tornillos estándar, los tornillos que son milimétricos son aquellos que anclan las piezas de *Soporte* de cada brazo a las piezas *BaseA*.

El primer bloque conforma la base fija del manipulador, tres piezas *BaseA* se unen por una pieza central en forma de círculo mediante tornillos de ¼ * 3 pulgadas (diámetro * largo); se coloca un tornillo de ⅜ * 3 pulgadas al extremo de cada pieza, en el cual se atornillan las *Patas de soporte* compuestas por tubular cuadrado y una tuerca de ⅜ de pulgada soldada a un extremo, con la finalidad de poder nivelar la base fija del robot considerando terrenos irregulares Fig. 87.

Fig. 87 Ensamble de la base fija (estructura) del manipulador

El segundo bloque involucra las piezas móviles del manipulador, inicialmente a una pieza *Soporte* le es incrustado un balero 608zz, por el cual cruza un tornillo de 5/16 * 5 pulgadas *eje del brazo*, al cual se le colocan dos tuercas de seguridad, permitiendo una separación de ½ pulgada entre el *Soporte* y la cabeza del tornillo; a través del tornillo se añade una pieza de impresión 3D *Unión Brazo*, la cual cubre a la segunda tuerca de seguridad, posteriormente se incrusta el *Brazo* del manipulador; partiendo del *Brazo* se repite el orden de las piezas en sentido contrario, se coloca una *Unión Brazo* y dos tuercas de seguridad, terminando con un balero incrustado en el *Soporte* el cual coincide con en el *eje del brazo*.

Posteriormente en ambos lados del extremo del *Brazo* se ensamblan dos piezas *Junta* mediante un tornillo de 3/16 * 1½ pulgadas, en cada cara de ambas piezas *Junta* debe colocarse una rondana, y una tuerca de seguridad al extremo del tornillo que cruza a las piezas; la pieza de impresión en 3D *Unión_Junta_B* impide el movimiento independiente de las juntas, la cual es ensamblada a las juntas colocando en cada extremo tornillos de 3/16 * ¾ pulgadas con su tuerca de seguridad como se muestra en la Fig. 88.

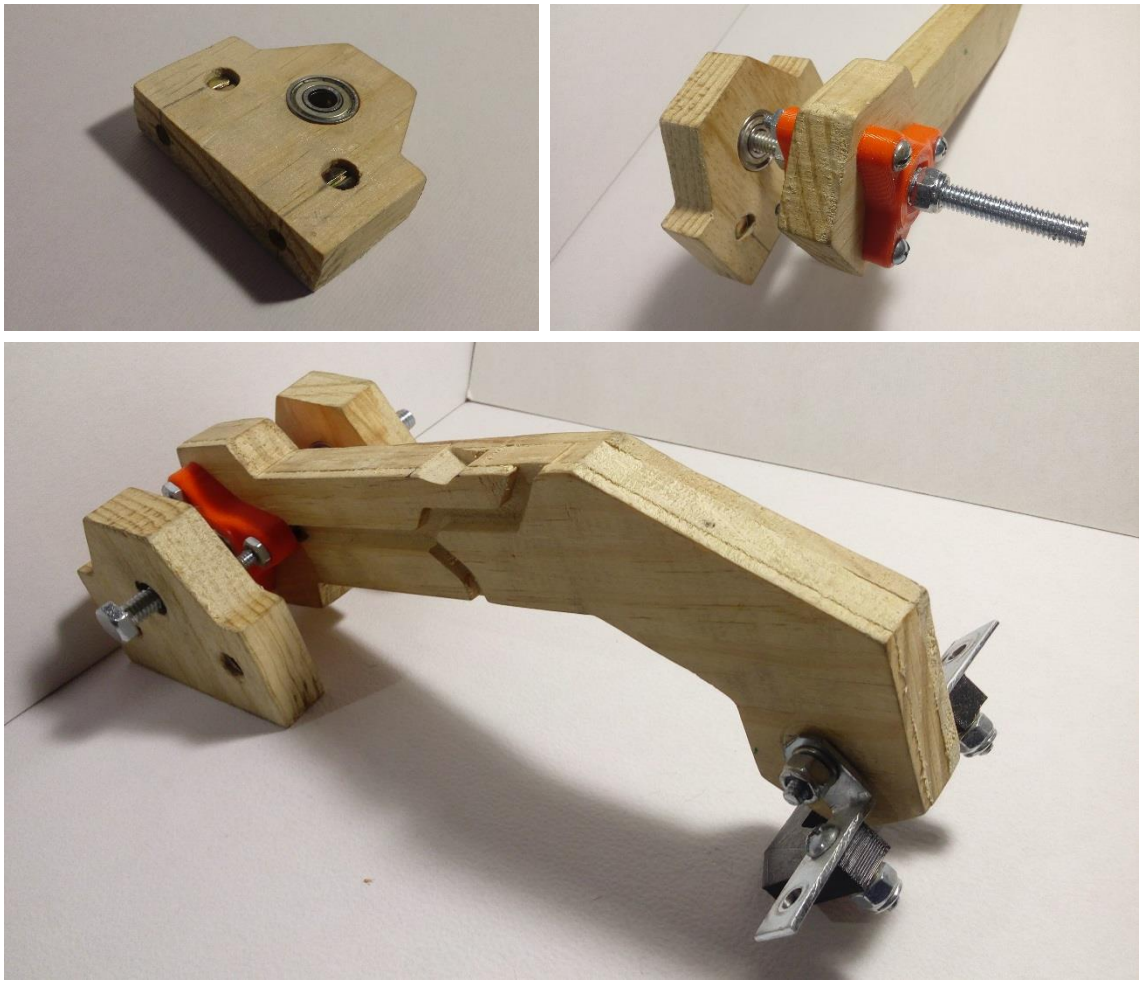


Fig. 88 Ensamble de soportes y brazo

El tercer bloque (Fig. 89) se conforma de la pieza *base móvil* del manipulador al cual se le ensamblan dos piezas *Junta* en cada extremo de la geometría mediante un tornillo de 3/16 * 1½ pulgada, incluyendo una rondana en cada cara de ambas juntas y una tuerca de seguridad, además se le coloca una pieza de impresión 3D *Unión_Junta_A* ensamblada con tornillos de 3/16 * ¾ de pulgada en el cuerpo de las juntas, con la finalidad de lograr el movimiento idéntico de ambas juntas.

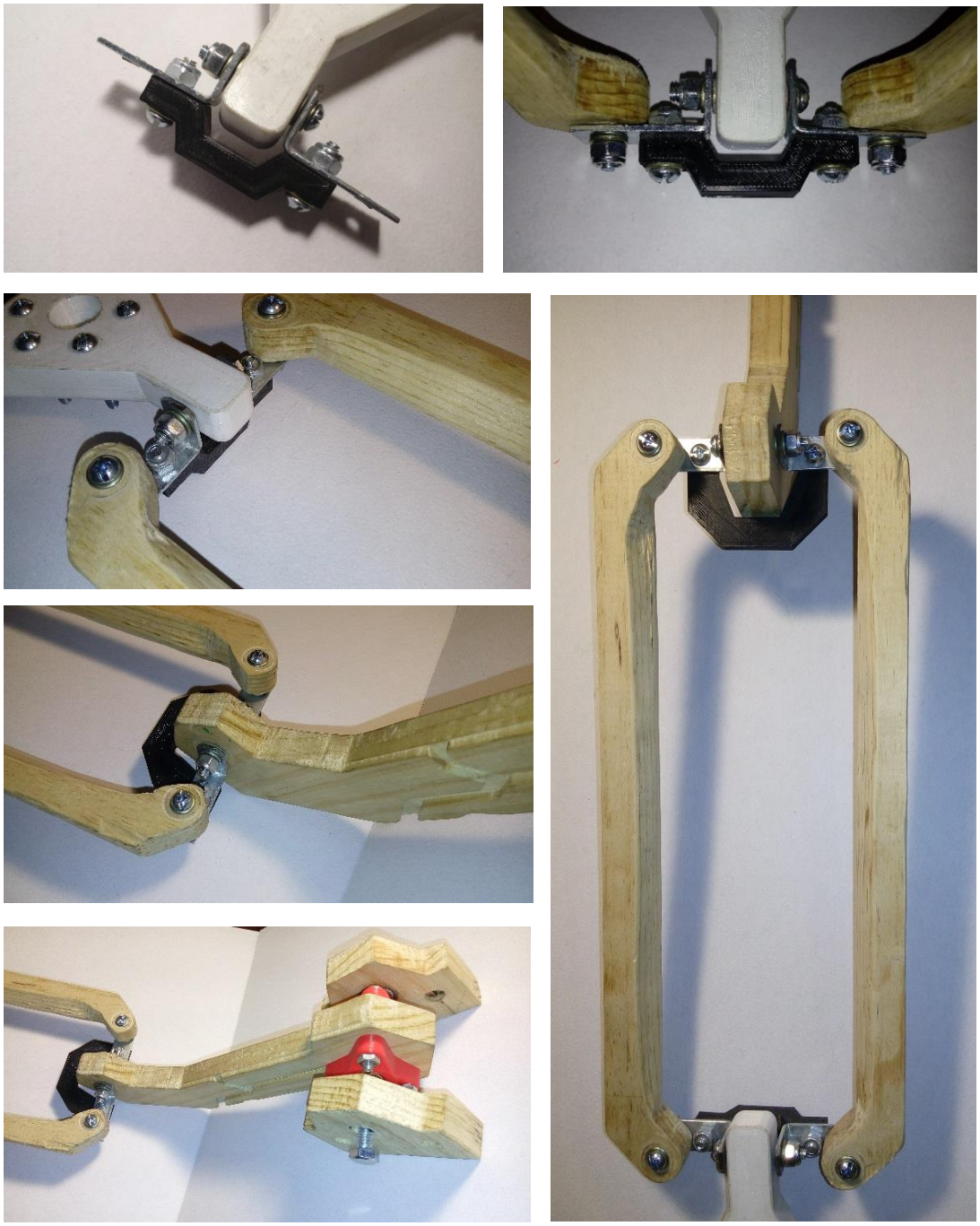


Fig. 89 Ensamble de una cadena de brazo a la plataforma móvil

La cadena de brazo ya ensamblado se ancla a la plataforma fija del manipulador mediante tornillos de 6 * 45mm y tuercas circulares; también se ancla el actuador Nema 17 con caja reductora de engranes 5:1, acoplado a la flecha o *eje del Brazo* utilizando un cople de 8mm, sujetado con opresores de 3mm; los sensores ópticos se colocan en una pieza 3D blanca debajo del cople del motor de modo que el giro de la pieza 3D negra que abraza al cople logre interrumpir el haz de luz del led, permitiendo que el opto transistor cambie el estado de la señal de salida hacia la placa de reconocimiento como se muestra en la Fig. 90.

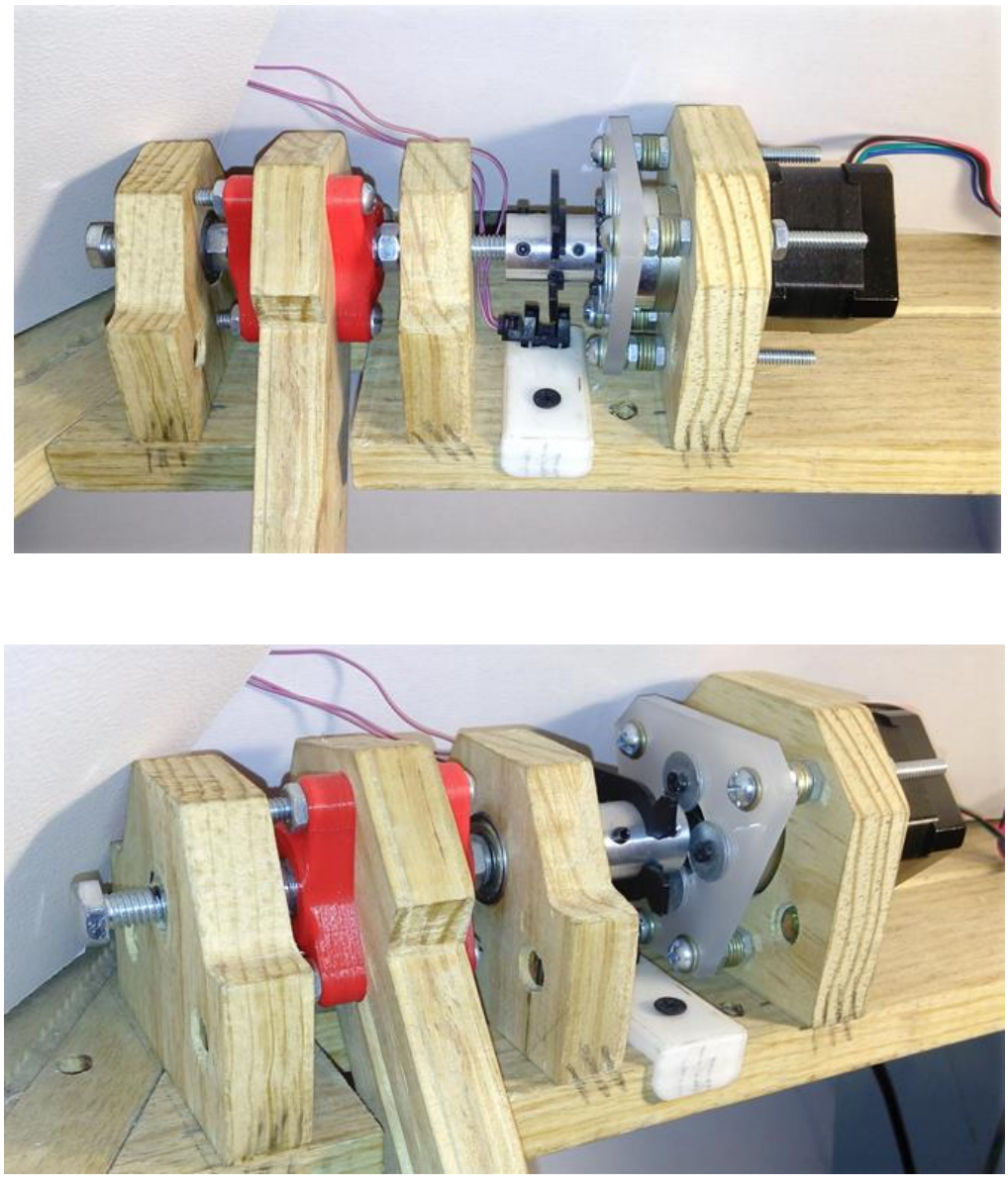


Fig. 90 Ensamble del brazo a la base fija

El acelerómetro GY-61 se monta a la pieza *brazo* utilizando una pieza impresa en 3D en forma de pinza, además se utilizan conectores de modo único de conexión para alimentar al módulo y transmitir la información a la tarjeta de reconocimiento como se muestra en Fig. 91.

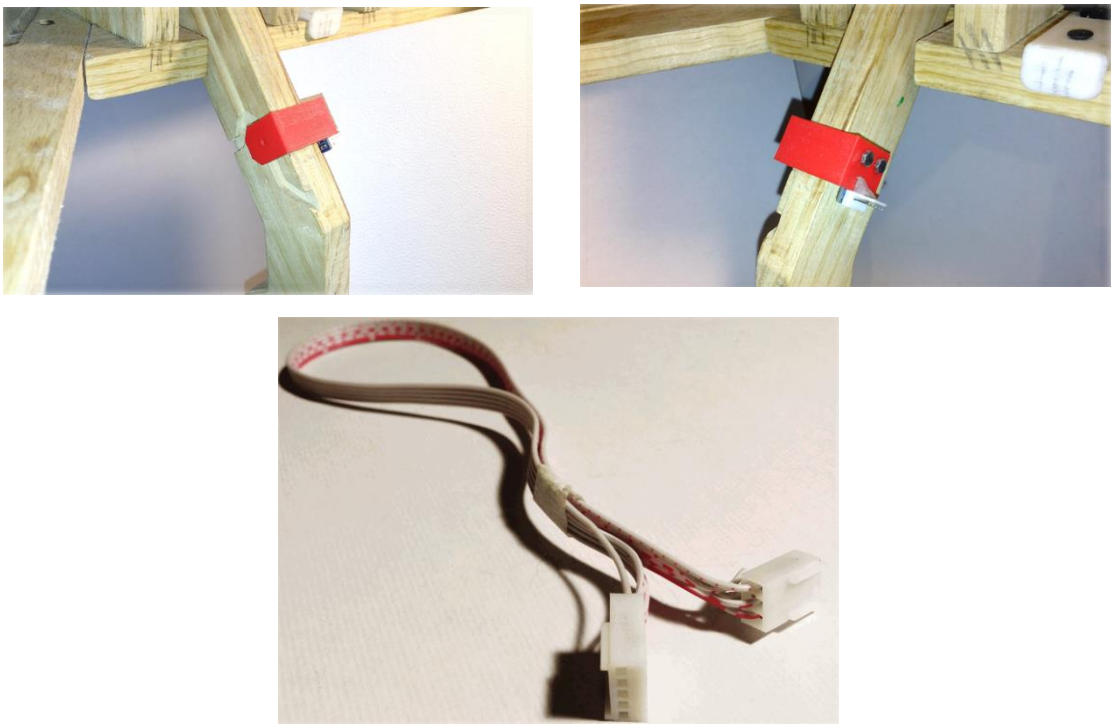


Fig. 91 Montaje del módulo GY-61 sobre la pieza Brazo

Los mismos bloques a partir del segundo se repiten como pasos a seguir hasta obtener las tres cadenas del manipulador ensambladas, de manera que la plataforma móvil se ensambla como se muestra en la Fig. 92.

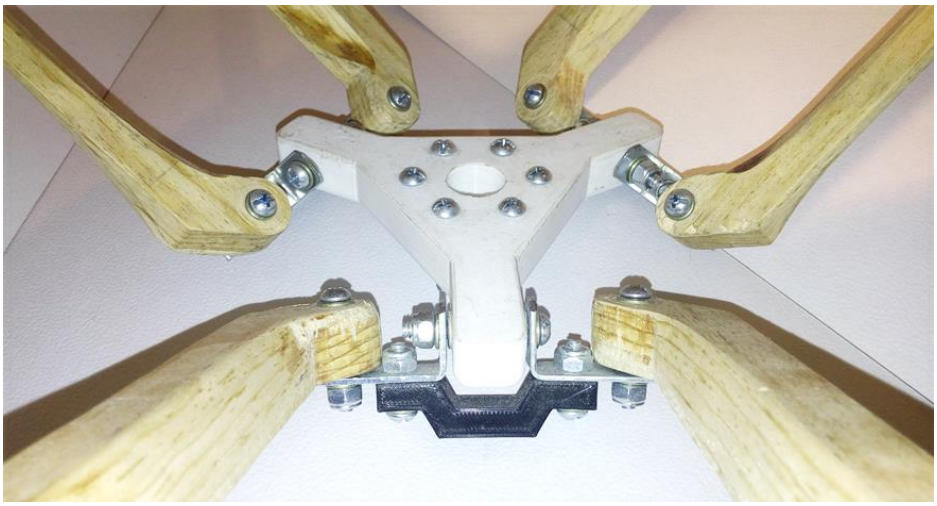


Fig. 92 Plataforma móvil ensamblada por tres cadenas de brazos

Para ensamblar la electrónica sobre la plataforma se deben atornillar los separadores para tornillo de ¼ en los tornillos que soportan a la base fija del manipulador y posteriormente atornillarlos al bloque de electrónica de modo que el bloque se encuentre sobre el manipulador como muestra la Fig. 93.

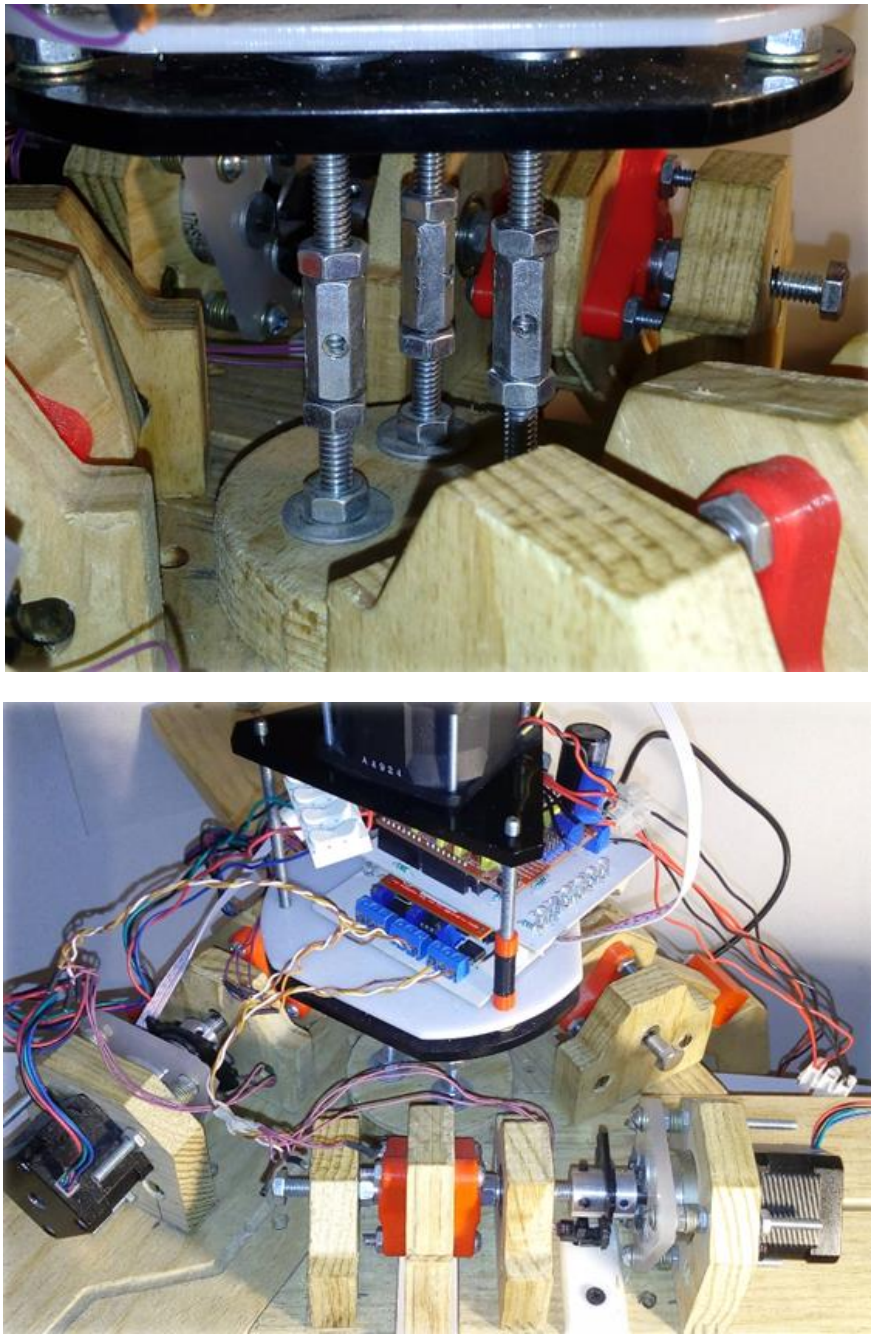


Fig. 93 Instalación del sistema embebido sobre el manipulador

En la Fig. 94 se muestra el ensamble del robot completo, el cual integra las partes del mecanismo y los componentes electrónicos que brindan información al controlador.

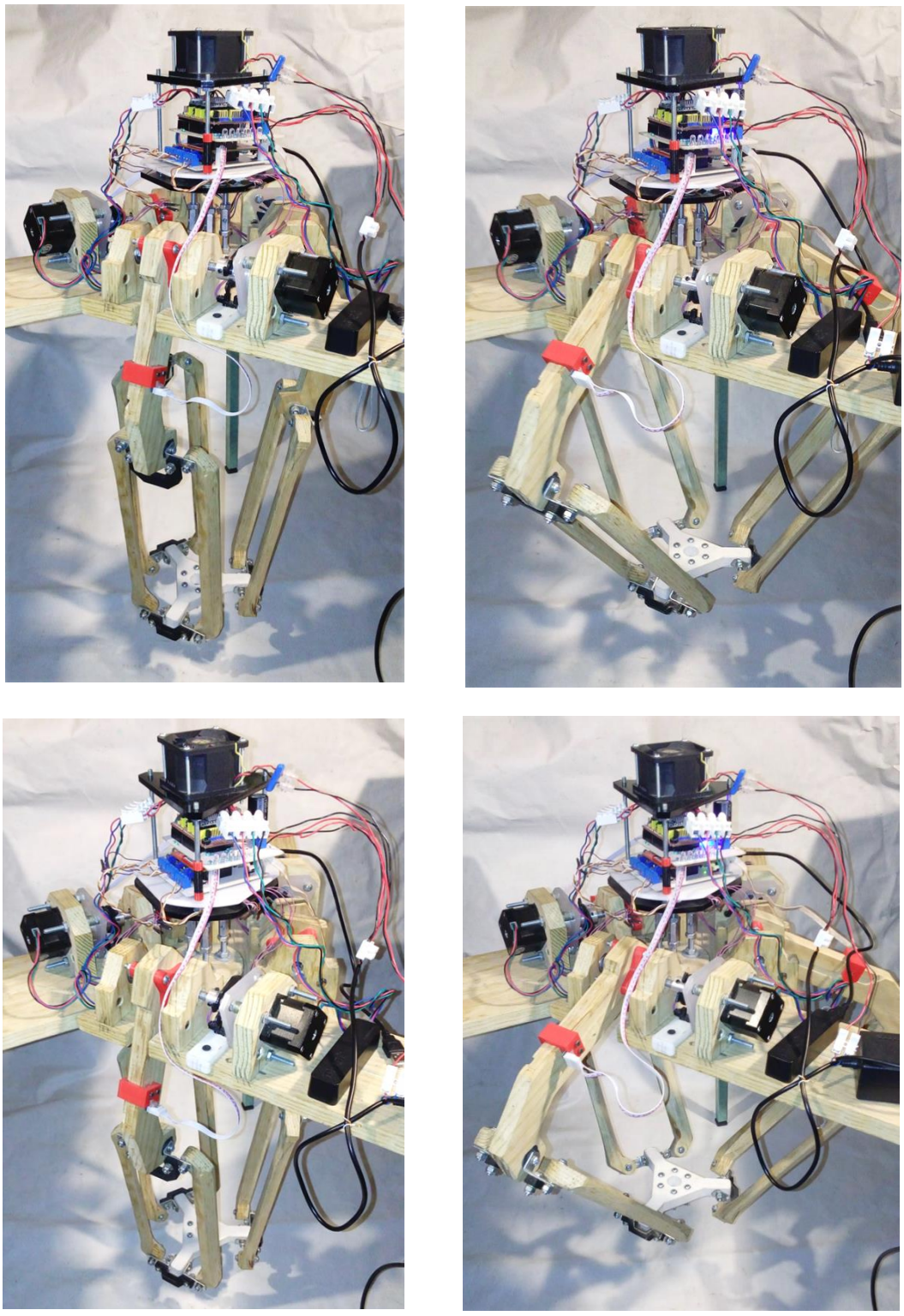


Fig. 94 Vista frontal del hardware del robot completamente integrado

Es importante resaltar que el robot puede mantener una posición fija al conectar la placa de opto acoplamiento y la placa de potencia a la fuente de 12 Volts a 5 Amperes, independientemente del funcionamiento de la placa de control; distintas vistas del robot se muestran en la Fig. 95 y Fig. 96.

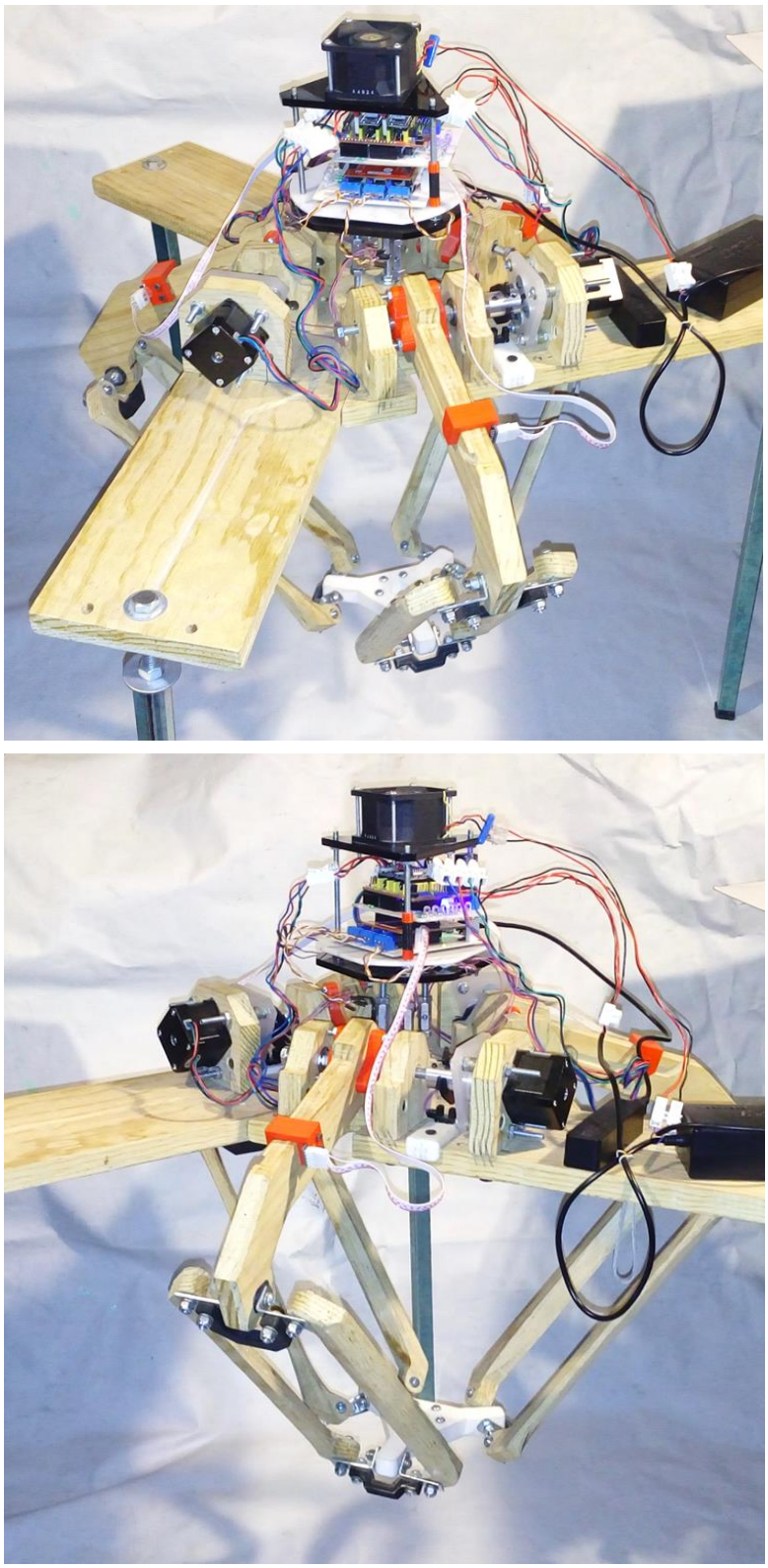


Fig. 95 Vista inclinada frontal del hardware del robot completamente integrado

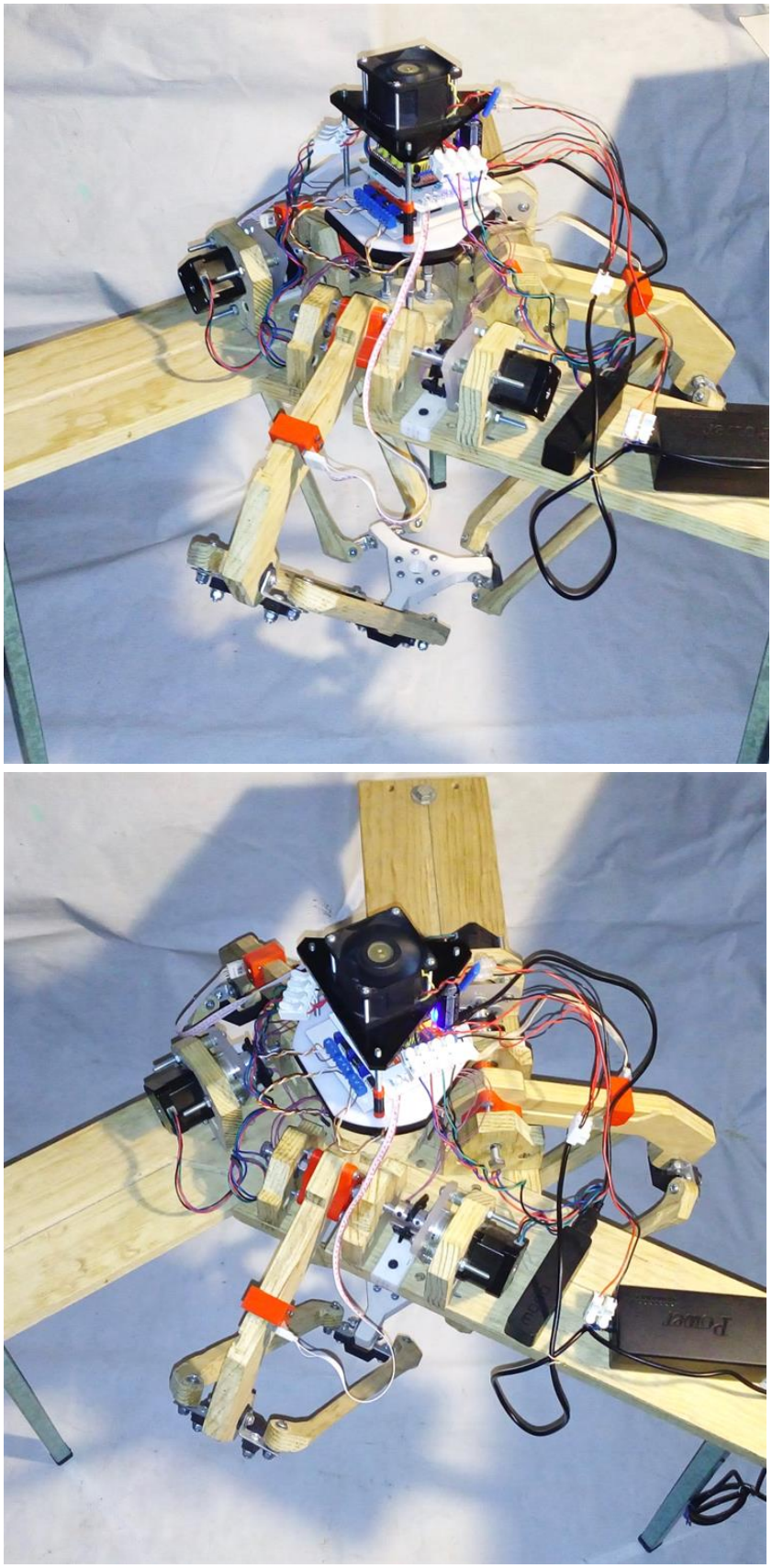


Fig. 96 Vista superior inclinada del hardware del robot

4.3 Integración del Sistema

El algoritmo cargado al controlador TM4C123G de la placa de control del sistema electrónico del robot, se describe a continuación para que el lector logre tener una mejor comprensión del algoritmo, así como del funcionamiento de la interfaz gráfica de usuario y de la implementación del mismo dentro de un sistema embebido para lograr el control remoto del prototipo piloto, el cual debe incluir las funciones planeadas de movimiento hacia un punto de interés y calibración.

4.3.1 Algoritmo de control en TIVA

Inicialmente el algoritmo cargado en el controlador obtiene información del puerto Serial seleccionado y se guarda la información en el arreglo de variables de tipo byte de nombre *BufferSerial* en la posición *contSerial*.

Cada vez que el algoritmo recibe un dato del puerto serie, el contador de nombre *contSerial* se evalúa, de modo que, si el valor de *contSerial* es mayor o igual a 8, indica que ha obtenido al menos 9 bytes del puerto serial, se manda llamar una función de nombre *AnalizarBuffer*; en caso contrario, si *contSerial* es menor a 8, únicamente se incrementa en 1 el valor de *contSerial* y regresa al inicio, para continuar leyendo el puerto serial.

Por lo anteriormente descrito, la función inicial del algoritmo resulta ser un bucle que consiste en leer el puerto serial hasta obtener 9 bytes, para ejecutar la función *AnalizarBuffer* como muestra la Fig. 97.

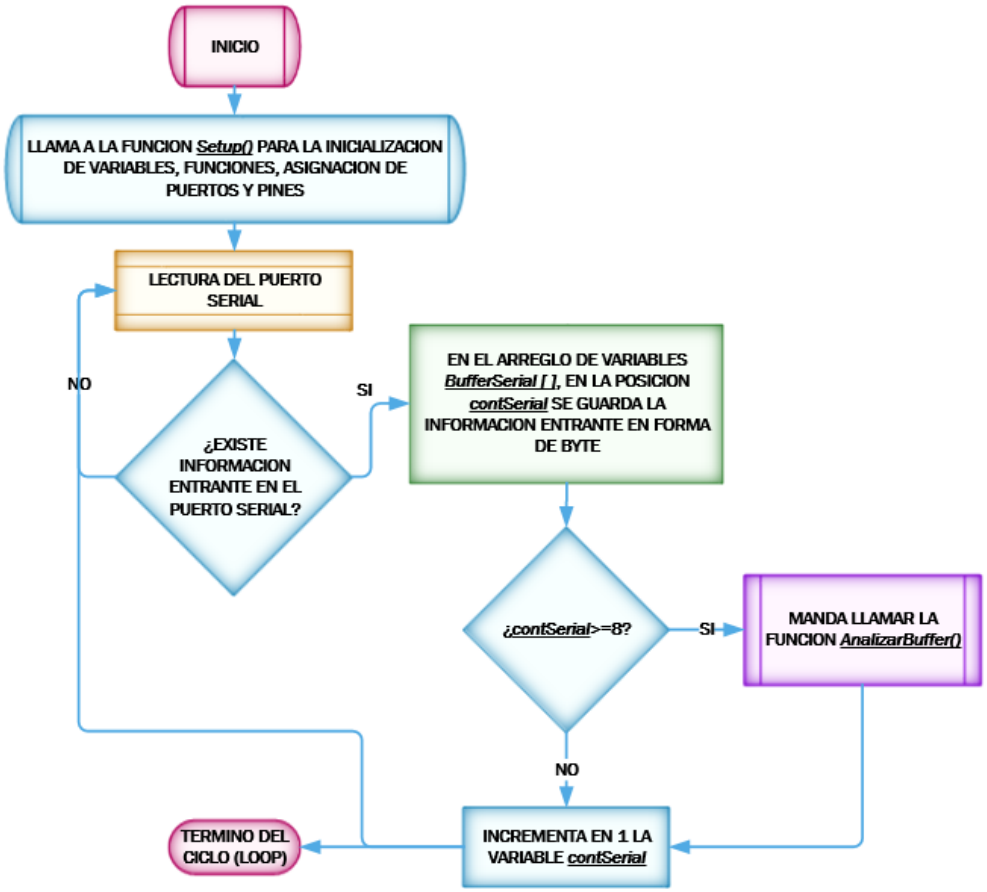


Fig. 97 Bucle inicial del algoritmo de control del robot

La función *AnalizarBuffer* tiene como objetivo validar la información guardada en el arreglo de variables de tipo byte de nombre *BufferSerial*.

La información guardada en *BufferSerial* debe contar con una cabecera, lo cual implica que debe contener valores específicos en las posiciones *contSerial* -1 y -2 para poder tomar en cuenta el paquete de datos obtenidos y guardados en posiciones anteriores; en caso de validar los valores en las posiciones indicadas y por lo tanto validar la información contenida en *BufferSerial*, se procede a llamar la función *ObtenerDatos*, dando como parámetro el valor contenido en *BufferSerial* en la posición *contSerial* (ultimo byte adquirido del puerto Serie) como se muestra en la Fig. 98.

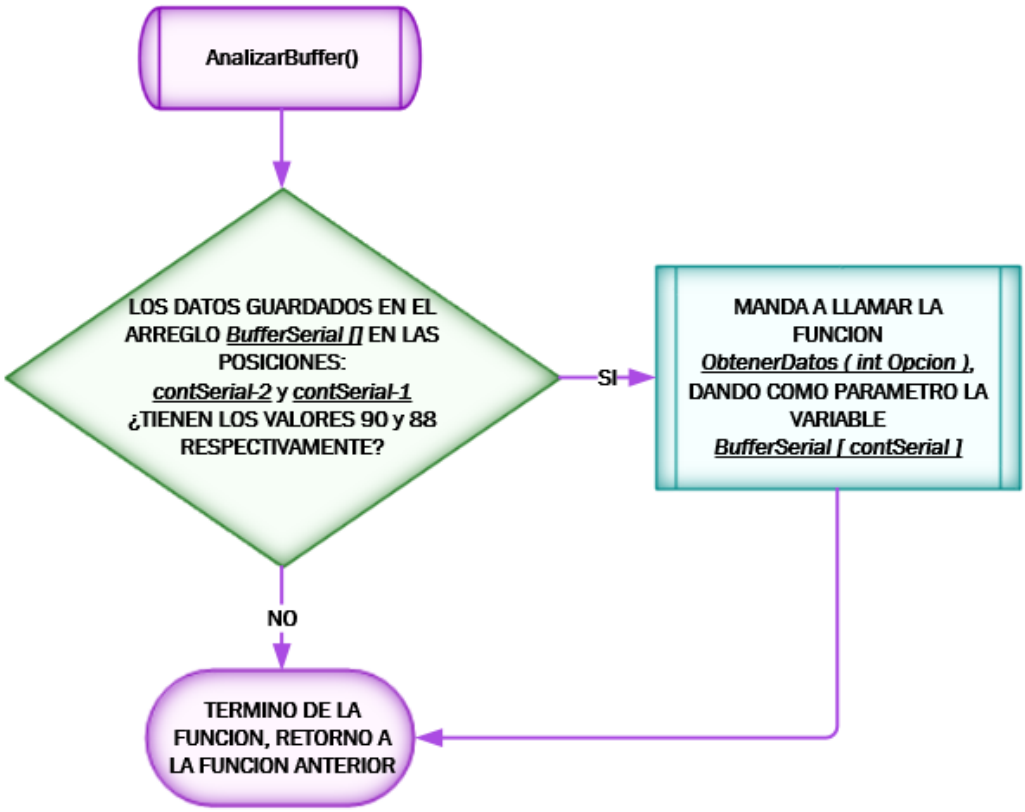


Fig. 98 Función AnalizarBuffer validación de información mediante cabecera

La función *ObtenerDatos* guarda el valor de *BufferSerial* posición *contSerial* (parámetro de la función) en la variable local de nombre *Opción*; si el valor de la variable *Opción* logra coincidir con algún valor previamente establecido para cada caso, el robot ejecuta una de las funciones planteadas en el capítulo anterior, los casos establecidos para el control del robot se describen a continuación.

El primer caso corresponde al valor 65 de la variable local *Opción*, indica que el usuario solicita que el robot se desplace hacia un punto de interés; el segundo caso requiere que la variable *Opción* contenga el valor de 66, indica que el robot debe calibrarse, por lo que debe regresar a un punto conocido como *HOME*; en caso de que el valor de *Opción* no logre coincidir con el valor de alguno de los casos anteriores, se ejecuta *default*, lo cual implica que el algoritmo termina saliendo de la función y regresa al inicio del programa para continuar obteniendo lecturas del puerto serial; el algoritmo descrito se muestra en la Fig. 99.

Fig. 99 Función ObtenerDatos selección de caso

Dependiendo de la función que va a desempeñar el robot los valores guardados en el arreglo de bytes *BufferSerial* son procesados de manera distinta, ya que tienen fines diferentes.

En el primer caso (65) los valores guardados en el arreglo *BufferSerial* se interpretan como componentes (*XYZ*) del punto de interés como se muestra en la Fig. 100, para posteriormente realizar los cálculos de cinemática inversa, logrando conocer y guardar los ángulos que requiere adoptar cada brazo, de modo que la plataforma móvil logre alcanzar el punto de interés con una trayectoria rectilínea, utilizando una campana de gauss como perfil de velocidad, suavizando el movimiento al inicio y termino de la trayectoria.

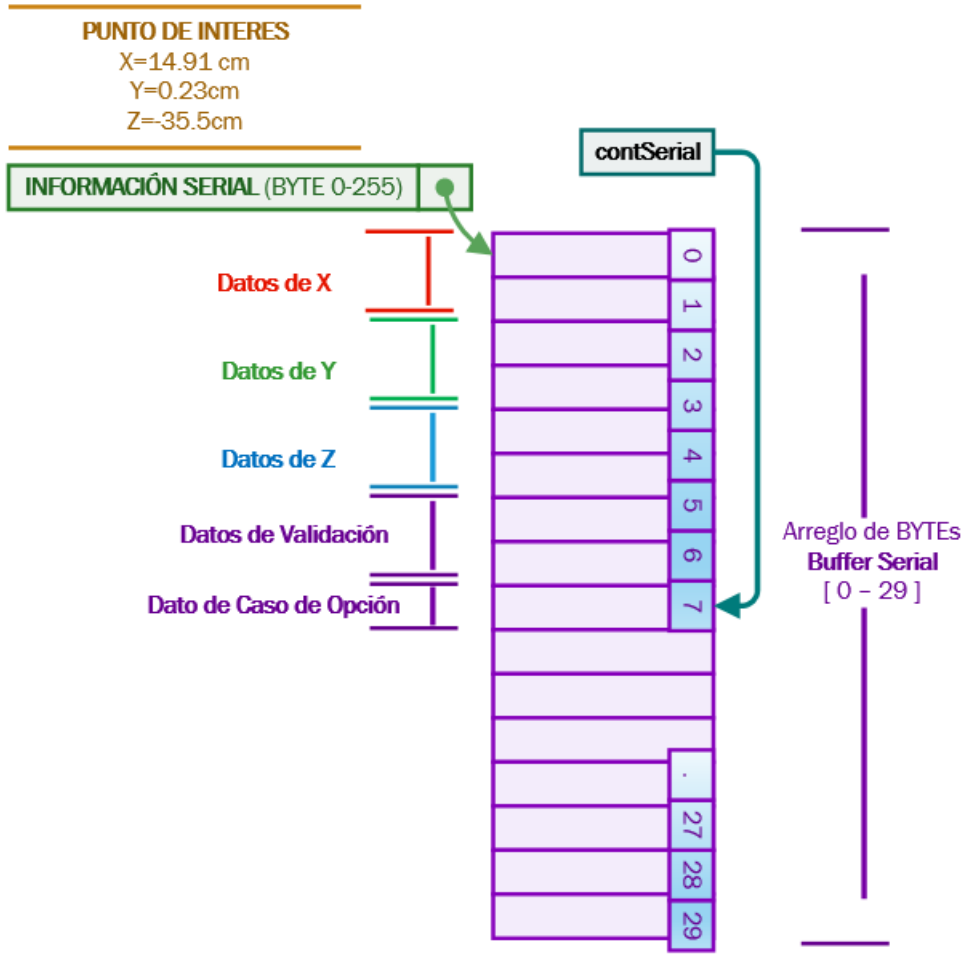


Fig. 100 Representación de información e interpretación del caso de movimiento

En el segundo caso (66), indica que el robot debe calibrarse, el robot actúa sin necesidad de información adicional guardada en *BufferSerial* como muestra la Fig. 101; cada actuador del brazo se moverá con la finalidad de acercar la plataforma móvil a la plataforma fija, hasta que la pieza 3D colocada en el cople de cada brazo logre interrumpir el haz de luz del límite de carrera, reestableciendo y ajustando el valor del error del sensor de los brazos para su posterior movimiento hacia un punto de interés.

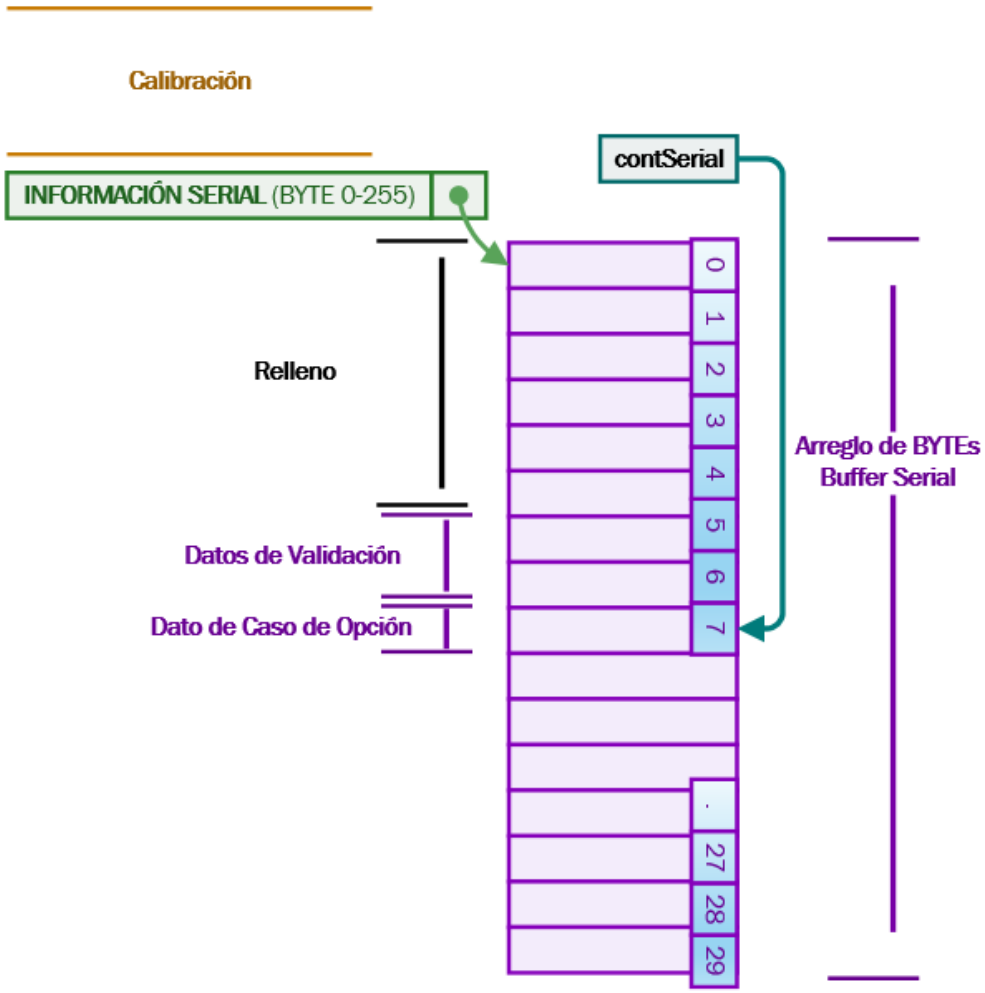


Fig. 101 Representación de información e interpretación del caso de calibración

Algoritmo de traslado de plataforma móvil al punto de interés a detalle (caso 65).

En la Fig. 102 se muestra el diagrama de flujo del algoritmo que se describe a continuación; se utiliza la información contenida en el arreglo *BufferSerial* para obtener las componentes x, y, z del punto de interés dadas por el usuario, dichas variables son guardadas en el arreglo punto final pf; se manda a llamar la función *BorrarBuffer*, ya que se han utilizado todos los elementos que requiere el caso 65; El algoritmo entra a un ciclo con una función de control *For* que inicia en 0 y termina en el número de periodos de la trayectoria o de puntos de ruta máximos, el ciclo permite calcular los puntos de ruta distribuidos sobre la trayectoria rectilínea que va de la posición actual del manipulador al punto de interés, además el ciclo calcula las posiciones angulares que requiere adoptar cada brazo para que la plataforma móvil logre posicionarse sobre los puntos de ruta hasta llegar al punto de interés.

El punto de ruta del i-ésimo periodo se calcula al resolver el polinomio característico del perfil de velocidad conocido como *campana de gauss* para el i-ésimo periodo, el resultado se guarda en la variable funcPol, la cual se introduce en la ecuación de la trayectoria rectilínea, obteniendo las componentes del i-ésimo punto de ruta, guardando los resultados en las variables *xp yp zp*.

Al obtener las coordenadas del i-ésimo punto de ruta, el ciclo permite calcular el valor de la posición angular que requiere adoptar cada motor, lo cual requiere utilizar la respectiva ecuación de movimiento del brazo, obtenida mediante cinemática inversa; cada destacar que las ecuaciones de cada brazo son diferentes, y todas requieren como parámetro las componentes *xp yp zp* del i-ésimo punto de ruta, obteniendo como resultado el valor de la posición angular para el i-ésimo periodo de la trayectoria, los resultados se guardan en su respectivo arreglo de variables de tipo flotante de nombre *THETA41, THETA42* y *THETA43*, cada arreglo consta de 30 posiciones, de acuerdo con los 30 puntos de ruta o periodos totales en los que se secciona la trayectoria.

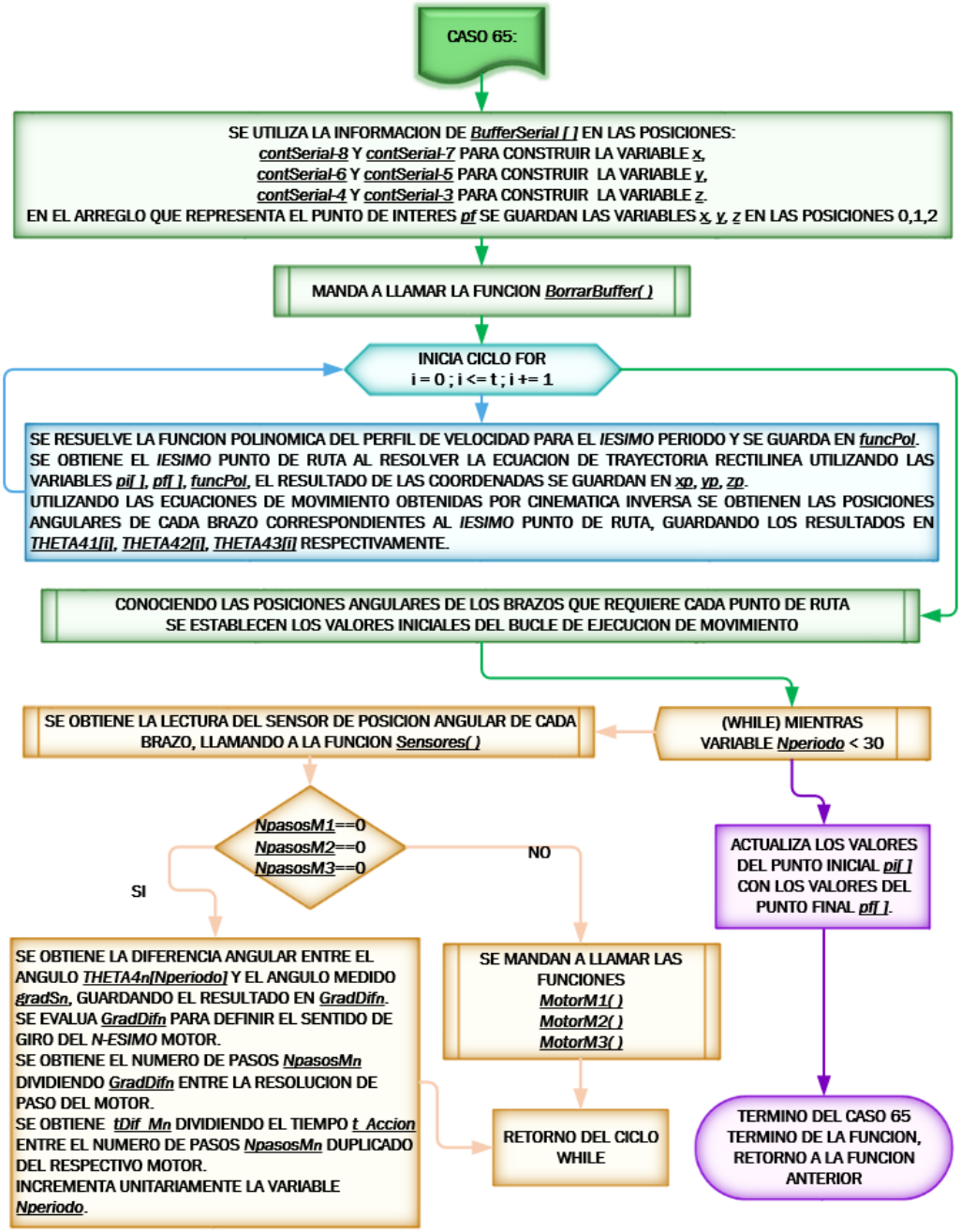


Fig. 102 Diagrama de flujo del algoritmo del caso de traslación caso Opción = 65

Al terminar de calcular los ángulos de cada brazo requeridos para alcanzar los puntos de ruta de la trayectoria, se le asigna el valor de cero a la N-ésima variable *NpasosMn*, anticipando el comienzo del ciclo *while* que permite la ejecución del movimiento de los actuadores.

En este punto, el algoritmo entra en un bucle que continua hasta que el número de periodos *Nperiodo* obtenga el valor de 30; inicialmente *Nperiodo* tiene un valor de cero, el cual incrementa unitariamente al haber ejecutado el número de pasos calculados para que cada brazo alcance la posición angular *THETA4n* del i-ésimo punto de ruta; el ciclo involucra el cálculo y la ejecución de pasos, por lo que se implementa un control de lazo cerrado proporcional [13, 31]; cada que el ciclo *while* se repite, trata de obtener información de los sensores, en cada nuevo cálculo de pasos se obtiene un valor distinto de la ganancia *P*, la cual se obtiene al resolver la diferencia de las posición angulares del brazo, medida por los sensores y posición deseada *THETA4n*; el ciclo termina al haber alcanzado cada uno de los puntos de ruta, logrando que la plataforma móvil alcance el punto de interés solicitado por el usuario, lo anterior se explica detalladamente a continuación.

Antes de ejecutar el movimiento de los motores para alcanzar el i-ésimo punto de ruta, se requiere llamar la función *Sensores*, la cual obtiene lecturas de los módulos GY-61, se procesa utilizando la ecuación característica del sensor, considera el error y guarda la información en la respectiva variable *gradSn*; lo anterior es requerido para calcular la diferencia entre el ángulo actual medido dado por *gradSn* y el ángulo calculado *THETA4n* para posicionar la plataforma móvil sobre el i-ésimo punto de ruta, dicha diferencia se guarda en la variable de tipo flotante de nombre *GradDifn* correspondiente a cada brazo, dicha diferencia se divide entre la resolución angular de paso del motor con tren de engranes, obteniendo así el número de pasos requeridos para alcanzar la posición angular calculada *THETA4n*, se guardan los resultados en la variable *NpasosMn*; el recorrido entre cada punto de ruta debe ejecutarse en un tiempo definido, su valor se establece desde el inicio del algoritmo con el nombre de *t_Accion*, dicho tiempo es dividido entre el doble de la variable *NpasosMn*, obteniendo el tiempo que debe tardar el cambio del estado lógico de la señal de control periódica *StepMn* para lograr la ejecución esperada por parte del robot (referenciar libro), dicho tiempo se guarda en la variable de nombre *t_DifMn*; para cada motor, el estado de la señal de control *DirMn* que define el sentido de giro del motor se asigna respecto al signo (positivo o negativo) de la variable *GradDifn*.

Una vez obtenido en microsegundos (µs) el tiempo correspondiente a la mitad del periodo de la señal de control *t_DifMn* y el número de ciclos *NpasosMn* que deben enviarse a los drivers de cada actuador para que cada brazo logre alcanzar la posición angular calculada *THETA4n*, logrando posicionar la plataforma móvil sobre el i-ésimo punto de ruta en un tiempo establecido; el algoritmo se encarga de ejecutar las acciones de movimiento con las funciones *MotorMn* de cada actuador y la función *Sensores*, ambas funciones se explican detalladamente a continuación.

Dentro de la función *MotorMn* como se muestra en la Fig. 103 inicialmente se evalúa el valor de la respectiva variable *NpasosMn* para si quiera considerar el cambio de estado lógico de la señal de control *StepMn*, ya que, si es mayor a cero indica que aún faltan pasos por ejecutar, posteriormente se asigna a la variable *tInit_Mn* el valor del contador interno de la *TIVA* en microsegundos utilizando la función *micros*, se procede a comparar la variable *t_DifMn* con la diferencia entre las variables *tInit_Mn* y *tTerm_Mn*, teniendo en cuenta que si tal diferencia no es mayor a *t_DifMn*, indicando que no es tiempo de ejecutar un cambio de estado en la señal de control, la función *MotorMn* termina y regresa a la función anterior; en caso de que la diferencia logre ser mayor a *t_DifMn* se evalúa el estado lógico de la señal, en caso de contar

con un estado bajo (LOW), se asigna un estado alto (HIGH) al estado de la señal; en caso contrario, entendiendo que la señal cuenta con un estado alto, se le asigna un estado bajo y se realiza el decremento unitario sobre el valor de la variable *NpasosMn*, indicando el término de un periodo de la señal; al termino de ambos casos se asigna a la variable *tTerm_Mn* el valor actual del contador en microsegundos interno de la *TIVA* utilizando la función *micros***,** renovando el ciclo de espera que requiere un flanco o cambio en el estado de la señal de control[32].

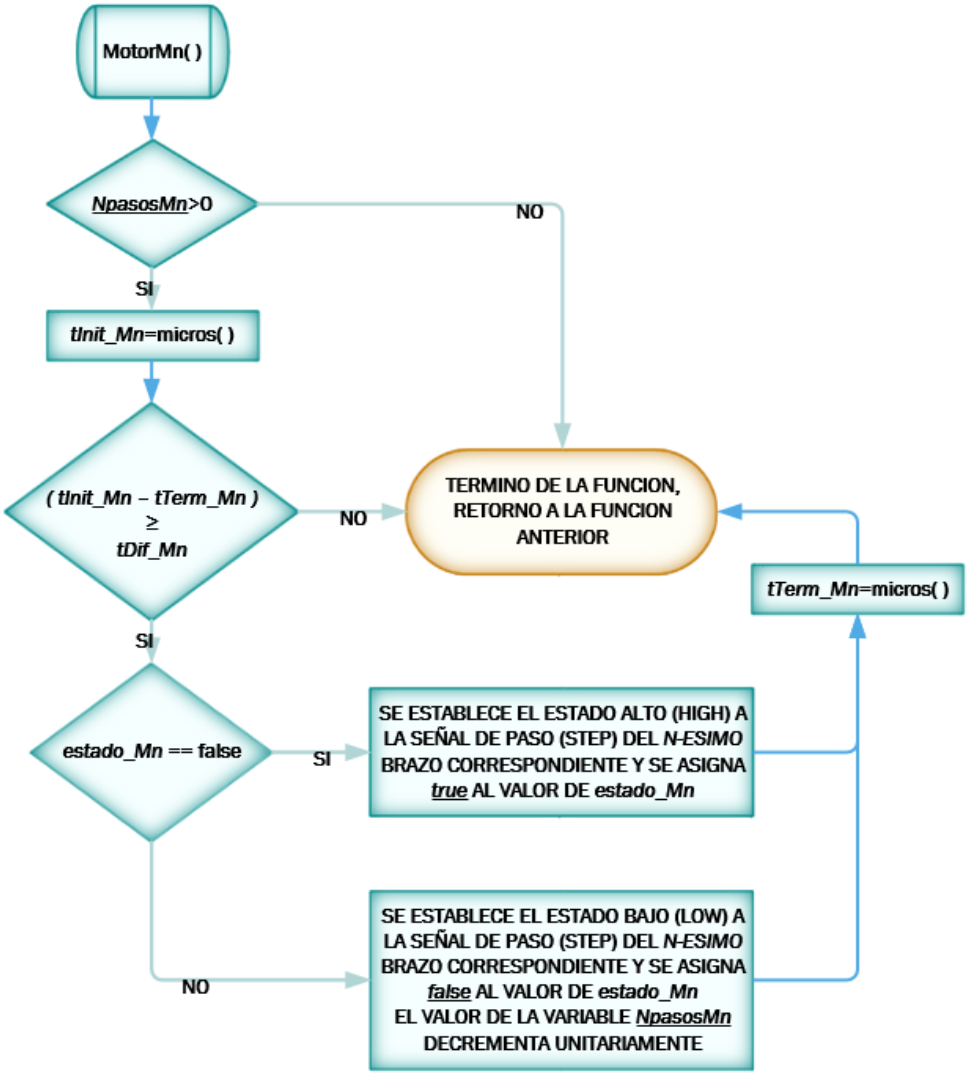


Fig. 103 Diagrama de flujo de la función MotorMn

El propósito de la función *Sensores* es tomar lecturas de la señal *Y* de los módulos GY-61 para conocer la posición angular de cada brazo, la Fig. 104 muestra el diagrama de flujo del algoritmo que se describe a continuación; inicialmente utilizando la función *micros* se asigna el valor del contador interno de la *TIVA* a la variable *tInit_ADC*; posteriormente se compara el valor de *tDif_ADC* con la diferencia entre las variables *tInit_ADC* y *tTerm_ADC*; de modo que, si *tDif_ADC* es mayor que el resultado de la diferencia de variables, el algoritmo sale de la función *Sensores* sin tomar lecturas, ya que los tiempos no se encuentran dentro del rango de la frecuencia de muestreo valida, que especifica la hoja de datos del sensor *ADXL335* del módulo GY-61; en caso de que *tDif_ADC* sea mayor que la diferencia entre las variables *tInit_ADC* y *tTerm_ADC*, las variables en el arreglo *busADCn* se recorren para que la lectura ADC del sensor se guarde en la última posición del arreglo *busADCn*.

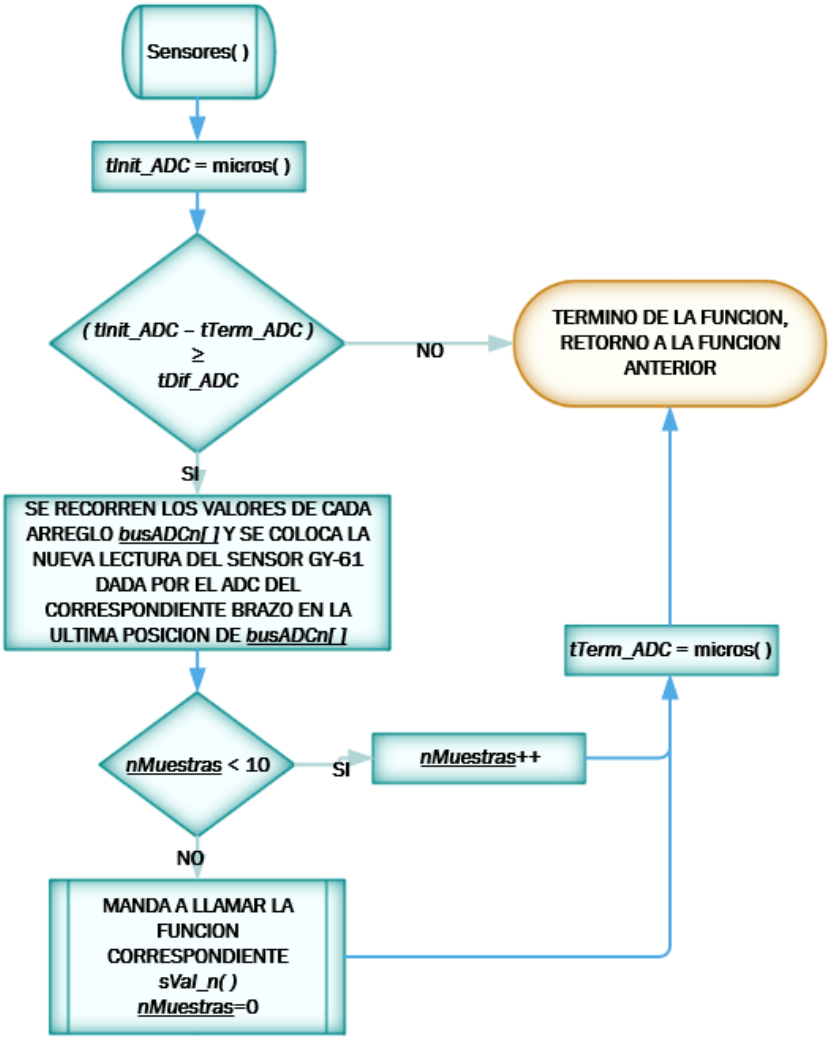


Fig. 104 Diagrama de flujo de la función Sensores

En la caracterización del módulo GY-61, con el apoyo de las lecturas graficadas dentro de la interfaz, se pudo notar mucha oscilación en el comportamiento del sensor al momento de cambiar de posición angular, con la finalidad de atenuar las variaciones mediante el algoritmo, es necesario promediar lecturas del sensor, obteniendo un único valor; por lo cual se evalúa el valor de la variable *nMuestras*, ya que, al haber guardado 10 muestras de cada sensor se manda a llamar la función *sVal_n* y reinicia el valor de *nMuestras*; en caso de no obtener 10 lecturas del ADC, únicamente se incrementa el valor de *nMuestras*; al termino de ambos casos se reestablece el valor de *tTerm_ADC* con el contador interno *micros* de la *TIVA*.

Dentro de la función *sVal_n* como se muestra en la Fig. 105, inicialmente se promedian 10 lecturas obtenidas por el ADC de la TIVA contenidas en el respectivo arreglo de variables *busADCn*, para obtener un único valor; posteriormente el resultado se introduce en la función polinomial que caracteriza el comportamiento de la señal *Y* del sensor, obteniendo un valor entre -90 y 90 grados, dicho resultado forma parte de un arreglo de nombre *ValPromSn*, guardando el resultado en la última posición, el cual se promedia arrojando el valor de la posición angular actual del brazo obtenida por el modulo GY-61, se guarda en la variable *gradSn*[33].

A la variable *gradSn* se le debe restar un error *ErrMn*, el cual es definido en el caso 66 por la función de calibración, por lo que antes de solicitar que el robot alcance un punto de interés es necesario que el usuario calibre el robot; la Fig. 106 muestra de manera gráfica el procesamiento descrito anteriormente sobre la adquisición de información del sensor.

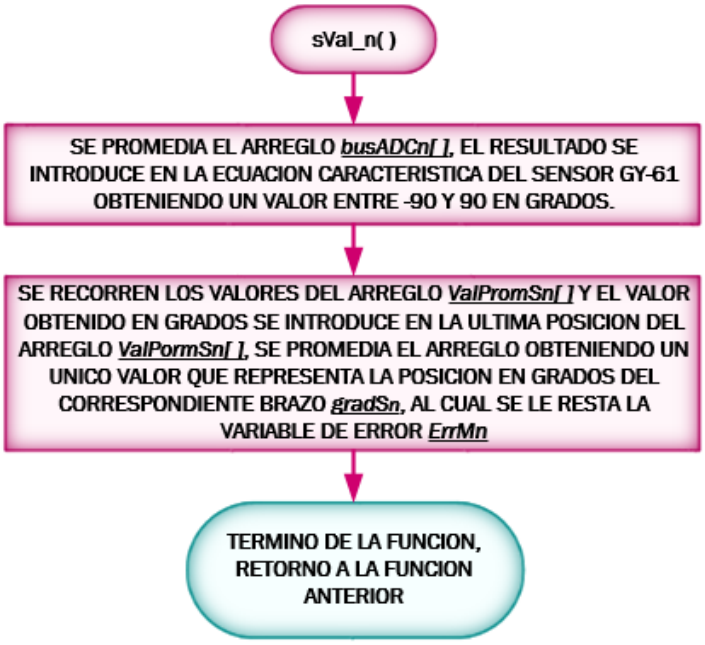


Fig. 105 Diagrama de flujo de la función sVal_n

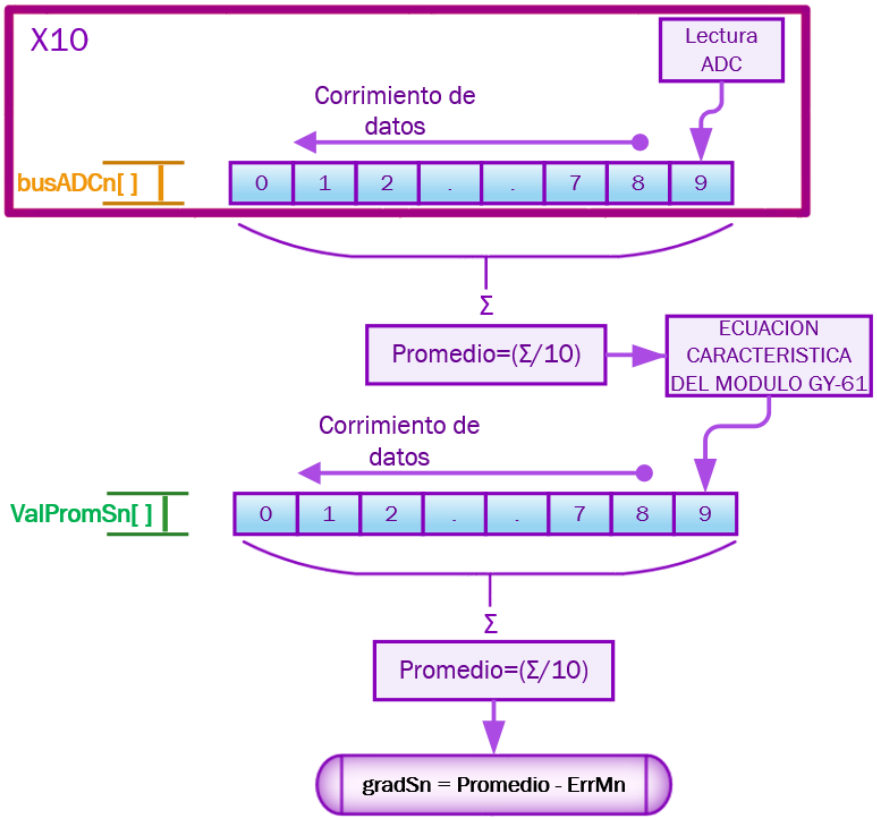


Fig. 106 Procesamiento de la señal Y del sensor GY-61

El ciclo while termina cuando los pasos de cada periodo se han ejecutado, lo cual implica haber recorrido todos los puntos de ruta de la trayectoria rectilínea y posicionar la plataforma móvil sobre el punto de interés, el algoritmo asigna los valores de las componentes del punto de interés o final arreglo *pf* a los valores del punto inicial del arreglo *pi*; termina saliendo de las funciones y regresa a la función principal que toma lecturas del puerto serial.

Algoritmo de calibración del robot (caso 66)

La Fig. 107 muestra el diagrama de flujo del algoritmo del caso calibración, en el cual la variable *Opción* contiene el valor de 66, Inicialmente los estados de las señales de control *DirMn* son asignados con la intención de elevar los brazos del robot, de manera que la plataforma móvil se acerque a la plataforma fija o base del robot.

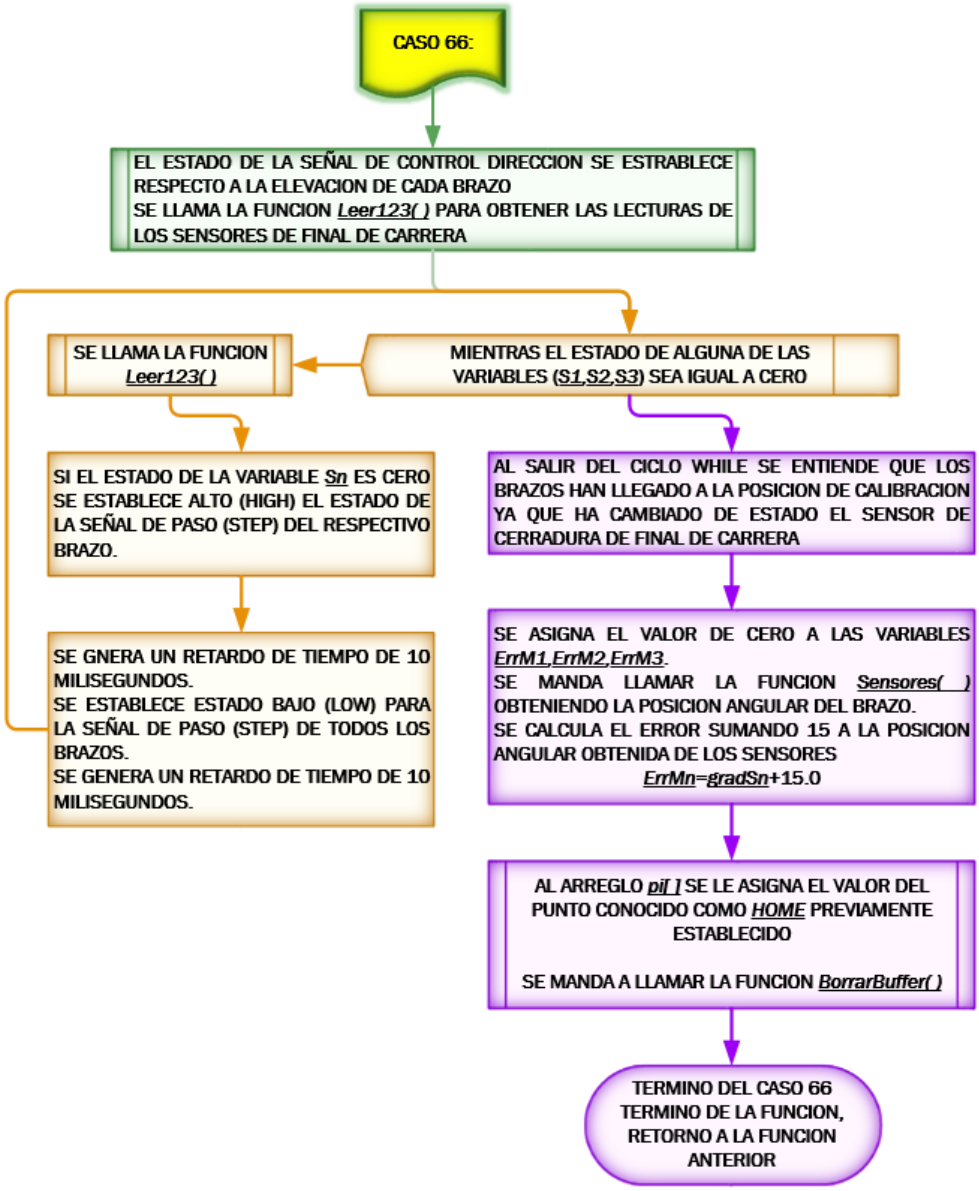


Fig. 107 Diagrama de flujo del algoritmo caso 66 (calibración)

Se manda a llamar la función *Leer123* para conocer el estado de los sensores ópticos de final de carrera colocados debajo de la flecha de cada motor a pasos con tren de engranes, esto prepara las variables *S1, S2, S3* para entrar en el ciclo while que permite ejecutar los pasos hasta llegar a la posición angular establecida para *HOME*; dentro del ciclo while, se manda a llamar la función *Leer123* y se evalúa el estado de los sensores, si el sensor del N-ésimo brazo no se ha visto interrumpido por la pieza 3D colocada sobre el cople (activando la señal del sensor), se asigna el estado alto a la señal de control *StepMn*, accionando al motor; en caso contrario, simplemente continua con el algoritmo sin generar el flanco de subida en la señal *StepMn*, se genera un retardo, se asigna el estado bajo a la señal de control *StepMn* y nuevamente se asigna un retardo; lo anterior continua hasta que todos los brazos se encuentren en la posición angular de calibración, posicionando la plataforma móvil sobre el punto *HOME*; la Fig. 108 muestra el diagrama de flujo de la función *Leer123*.

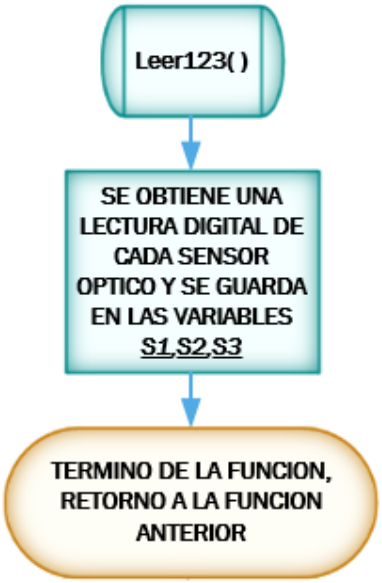


Fig. 108 diagrama de flujo de función Leer123

Al terminar el movimiento de los brazos y salir del ciclo while, se preparan las variables *ErrMn* asignándoles el valor de cero; se manda a llamar varias veces la función *Sensores* para obtener la medición de la posición angular de cada brazo; el valor de *ErrMn* se obtiene al sumar 15 a la variable *gradSn***,** ya que la posición angular de calibración es de -15 grados de acuerdo con las transformaciones seguidas en la cinemática del manipulador; posteriormente se muestra un ejemplo que expone la funcionalidad de la acción de calibración; a las componentes del punto inicial se les asignan las componentes (x, y, z) de la posición de calibración *HOME* (0,0,-15.73) en centímetros, las coordenadas son obtenidas utilizando el software de CAD en donde fueron generadas las piezas del manipulador paralelo al posicionar angularmente los brazos a 15 grados por encima de la plataforma móvil como se muestra en la Fig. 109; el caso 66 termina llamando la función *BorrarBuffer*, preparando al robot para atender una solicitud de movimiento hacia un punto de interés (caso 65), el algoritmo termina saliendo de las funciones y regresa a la función principal que toma lecturas continuas del puerto serial.

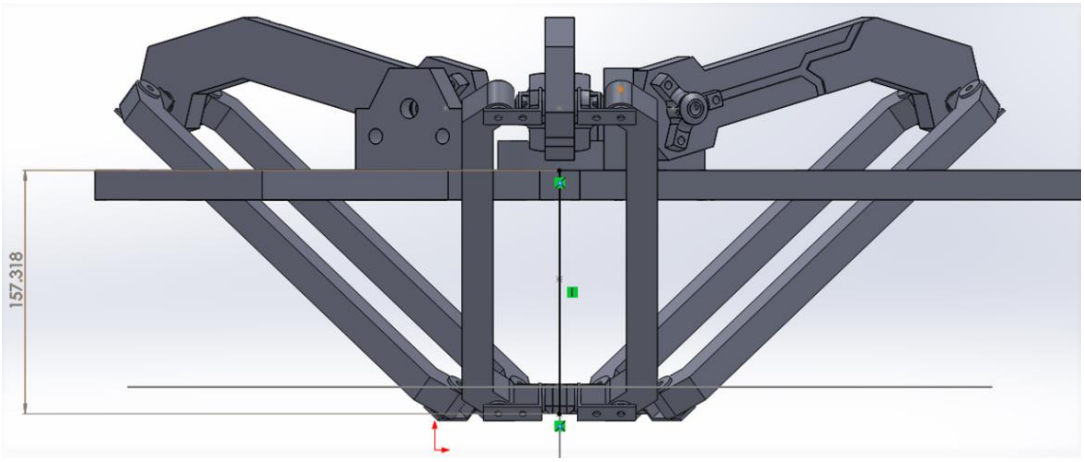
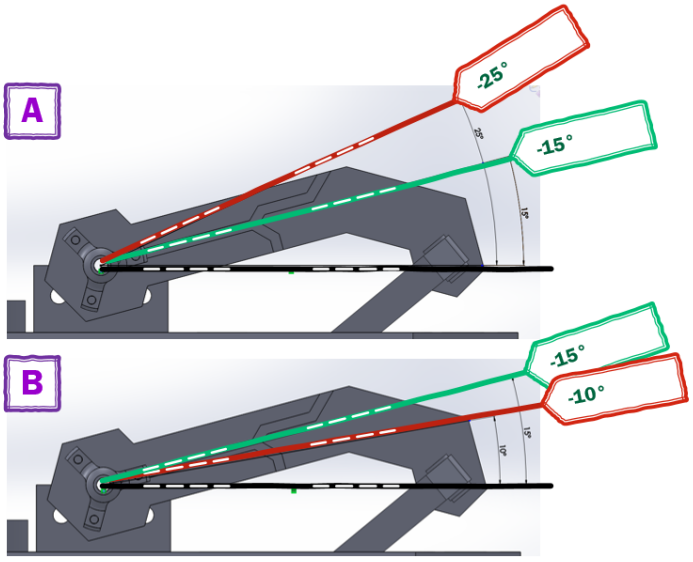


Fig. 109 Altura de calibración

En la siguiente descripción se exagera un ejemplo con la intención de hacer notar el proceso de la función de calibración, la Fig. 110 muestra dos casos (A y B) en los cuales se observan dos pestañas rojo y verde con valores, las cuales representan la posición angular medida (GY-61) y la real – física (*HOME*) respectivamente, difiriendo entre si ambos valores, los casos exponen dos situaciones, la primera muestra a la variable medida siendo mayor que la variable real y la segunda muestra a la variable medida siendo menor que la variable real; en la figura se puede observar que el *valor medido gradSn* es el valor promedio de los datos contenidos en el arreglo *ValPromSn*; también se puede notar que al sustituir el valor del error *ErrMn* en la *ejecución de movimiento* se obtiene como resultado -15 en ambos casos, logrando corregir el valor medido por los respectivos módulos GY-61; ya que el caso 66 mide el error y el 65 ejecuta la acción de movimiento hacia un punto de interés contemplando dicho error, el robot requiere ser calibrado antes de ejecutar una solicitud de movimiento.



Medición del Error caso 66

Valor medido gradSn + 15 = ErrM

A	B
-25 + 15 = -10	-10 + 15 = 5

Ejecución de movimiento caso 65

Valor medido gradSn - ErrM = gradSn

A	B
-25 – (-10) = -15	-10 - 5 = -15

Fig. 110 Ejemplo de la operación de calibración

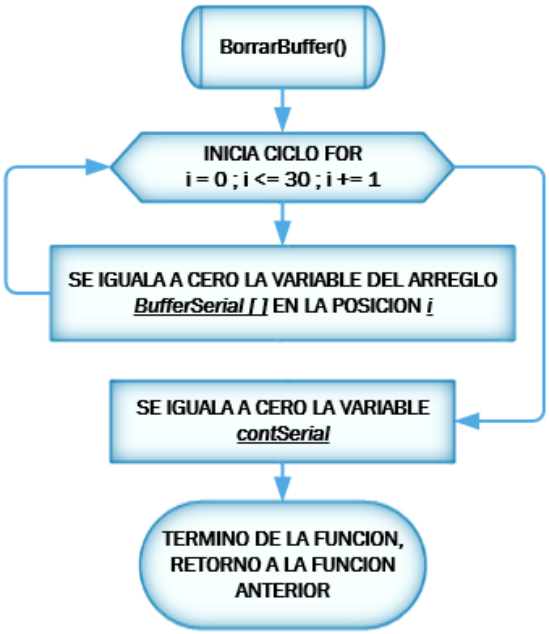


Fig. 111 diagrama de flujo de la función BorrarBuffer

La Fig. 111 muestra el diagrama de flujo de la función *BorrarBuffer* que es descrita a continuación y se utilizada en los casos 65, 66, default; inicialmente entra en un ciclo *for* que inicia en 0 y termina en 30, que es la dimensión del arreglo *BufferSerial*, asignando el valor de cero a las variables de *BufferSerial* en la i-ésima posición; la función termina con la asignación del valor de la variable contSerial, reestableciendo el puntero de *BufferSerial* a cero.

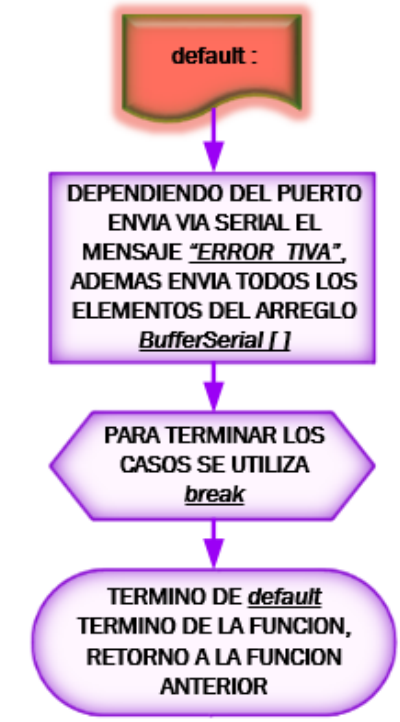


Fig. 112 Diagrama de flujo del caso default

La Fig. 112 muestra el diagrama de flujo del caso *default*, el cual se manifiesta cuando existe un error, dado que, no hay coincidencia con ningún caso por parte de la variable *Opción*; se envía vía serial el mensaje *error_TIVA* al monitor serial del usuario, así como los elementos contenidos en el arreglo *BufferSerial*, de modo que el usuario pueda identificar la falla en la información serial que se ha enviado hacia el robot; la función termina saliendo de las funciones y regresa al bucle principal, el cual busca recibir datos del puerto serial para posteriormente interpretar las instrucciones.

4.3.2 Interfaz de usuario

Processing apoyado de la biblioteca *ControlP5* permite desarrollar un entorno gráficamente simple con elementos de control e indicadores configurables mediante código, lo cual permite al usuario una interacción con el robot; la Fig. 113 muestra los elementos de la interfaz gráfica de usuario (GUI) del robot delta; en la parte superior derecha de la ventana, se encuentra un botón que cambia el estado de la comunicación serial, activando o desactivando las funciones de la interfaz de acuerdo al estado de la comunicación serial, en la parte superior izquierda se observa un indicador que cambia de color de acuerdo al estado de la comunicación serial; en la parte central de la ventana se encuentra un control deslizante (slider) de dos dimensiones, nombrado *s2D*, mediante el cual, el usuario al posicionar el puntero dentro del panel, establece las componentes *X,Y* del punto de interés, a su costado derecho se encuentra un control deslizante simple en el cual el usuario establece la componente *Z* del punto de interés; en la parte central a la derecha de la ventana, se encuentran dos botones encargados de enviar al robot las instrucciones vía serial para ejecutar dos funciones diferentes; al dar clic sobre el botón *CALIBRAR*, el algoritmo de la interfaz se encarga de enviar 6 bytes sin contenido específico, posteriormente envía la cabecera de validación y el valor *66* para que el controlador identifique la solicitud de calibración; al dar clic sobre el botón *ENVIAR* el algoritmo desglosa las componentes del punto de interés enviando inicialmente la parte entera y después la parte fraccionaria de cada componente (X,Y,Z) logrando enviar 6 bytes, posteriormente se envía la cabecera de validación y el valor *65* para que el robot logre identificar la solicitud de movimiento hacia el punto de interés.

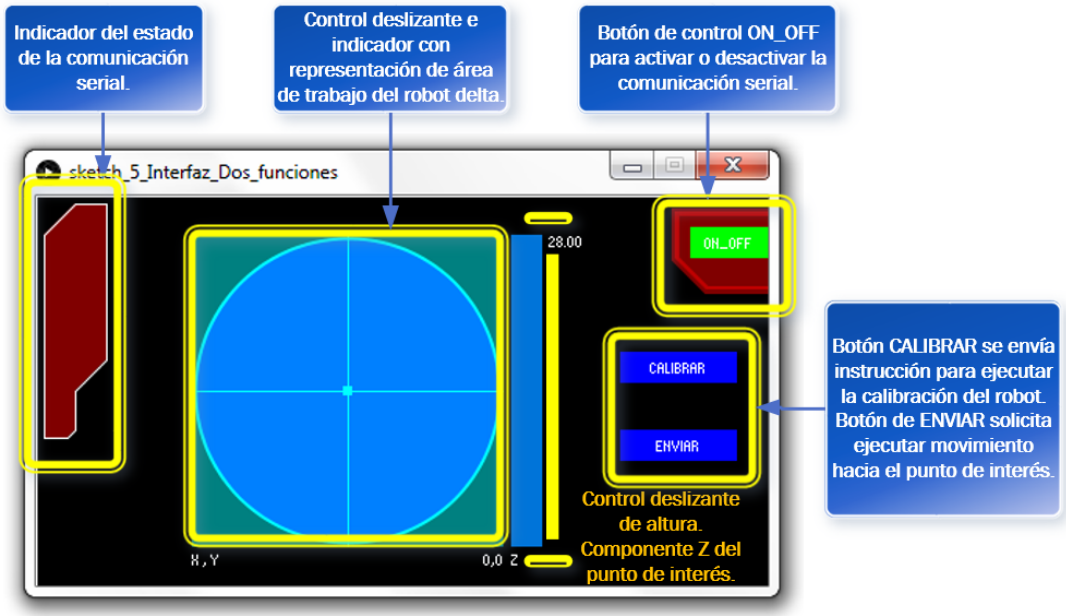


Fig. 113 Ventana de interfaz del robot delta

A continuación se explica el procesamiento de datos que se realiza por el algoritmo de la interfaz al dar clic en el botón enviar; inicialmente se obtiene la parte entera del valor de la componente en turno (x, y, z), lo cual se logra al convertir el tipo de la variable de tipo punto flotante (float) a tipo entero (int), despreciando la parte fraccionaria de la componente; se evalúa si el valor de la componente es positivo o negativo, ya que, enseguida se envía el valor absoluto de la variable entera y en caso de ser negativa se le suma 128 para activar el 8vo bit del byte con longitud de 8bits; en caso de ser positivo, únicamente se envía el valor absoluto; lo anterior no afecta el valor de la variable ya que no puede haber ningún valor mayor a 100, debido a que las componentes del punto de interés se encuentran dadas en centímetros.

Posteriormente el algoritmo de la interfaz envía la parte fraccionaria de la componente en turno de la siguiente manera, al valor absoluto de la componente en turno se le resta el valor entero obtenido anteriormente, el resultado se multiplica por 100 y se convierte en una variable entera, dicha variable se envía vía serial, de modo que se ha enviado la componente en turno completa; a modo de ejemplo se muestran dos casos para enviar el valor de una componente vía serial; en caso de que el valor de la componente sea 15.42, el primer byte que se envía es 15 y posteriormente 42; en caso de querer enviar -15.42, inicialmente se envía 143 (15+128) y posteriormente 42, el controlador que recibe la información y procesa los datos y rápidamente reconstruye las componentes del punto de interés, para ejecutar el movimiento del caso 65.

Con el fin de conocer el volumen de trabajo del robot, utilizando el software de CAD en el cual se diseñó y ensamblo el manipulador, se colocan a distintas alturas una serie de círculos delimitados por el alcance de la plataforma móvil, se genera un archivo de texto separado por tabulaciones que contenga en una columna los datos de la altura y en otra el valor del radio de cada circunferencia que se puede observar en la Fig. 114.

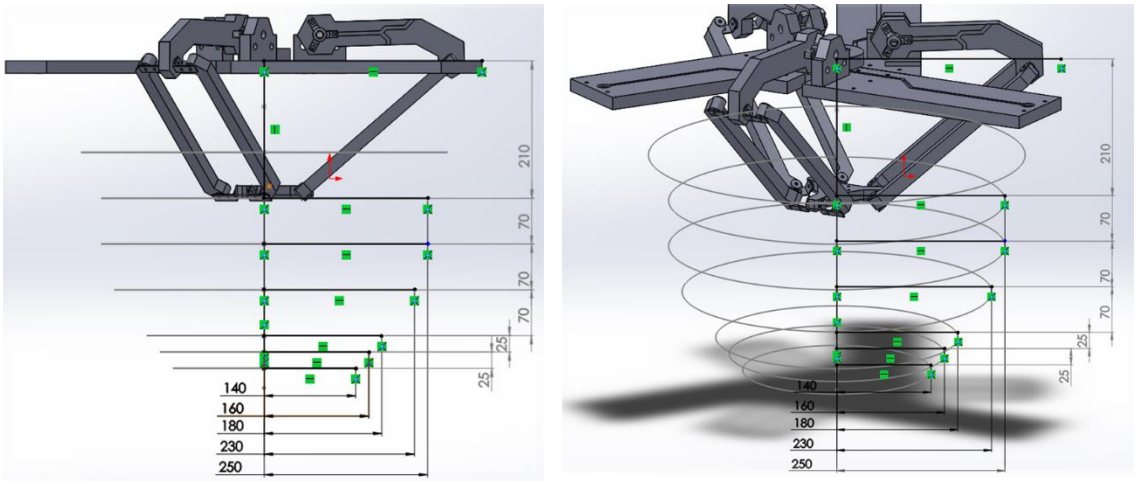


Fig. 114 Límites máximos de movimiento del robot

Se puede notar en las figuras anteriores la relación entre la variable de altura y radio, ya que, al aumentar la altura del manipulador, el área de acción disminuye; utilizando la función *BasicFitting* en *Matlab*, se obtiene la ecuación que caracteriza el comportamiento del área de acción como se muestra en la Fig. 115.

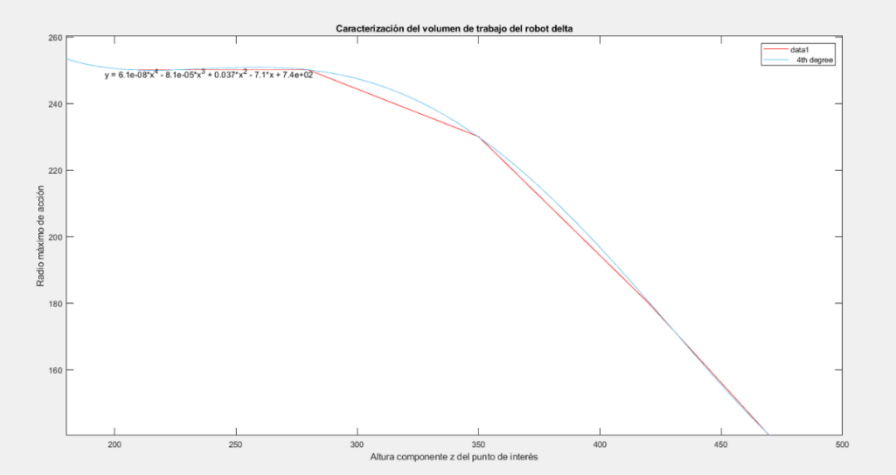


Fig. 115 Grafica de comportamiento del radio de acción máximo contra la altura Z

Dentro de la interfaz, el control deslizante que permite al usuario establecer el valor de la componente Z del punto de interés, es utilizado como parámetro de la ecuación característica, al resolverla se obtiene el valor del radio máximo de acción *Rz*, dicho valor se asigna como radio a la circunferencia que representa el límite de movimiento de la plataforma móvil sobre la altura establecida, la cual se observa dentro del control deslizante *s2D*; el algoritmo de la interfaz restringe los puntos de interés (peticiones del usuario) según los límites de la circunferencia, además constantemente obtiene la hipotenusa de las componentes X, Y del slider *s2D*, al realizar el cálculo de la raíz cuadrada de la suma de las componentes elevadas al cuadrado, guardando el resultado en la variable *R*, seguido de ello se comparan los valores obtenidos de *R* con *Rz* y en caso de que *R* sea menor a *Rz*, la coordenada seleccionada es válida, ya que los valores X,Y se encuentran dentro del área de acción del robot; en caso contrario (*R* mayor a *Rz*) el puntero del slider *s2D* regresa a la última coordenada valida como se muestra en la Fig. 116 y Fig. 117.

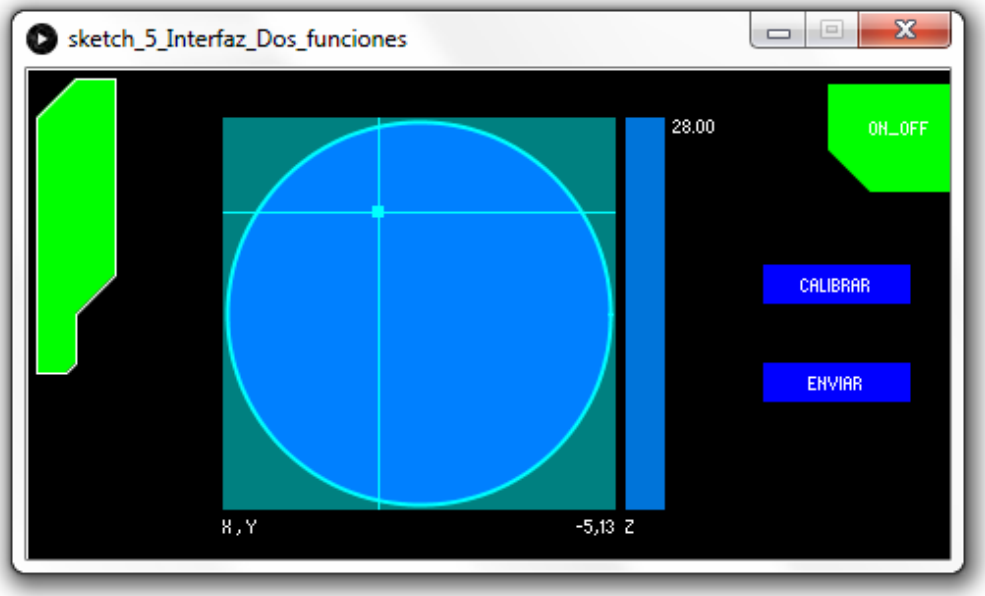


Fig. 116 Interfaz punto de interés -5,13 (X, Y), con altura 28 (Z)

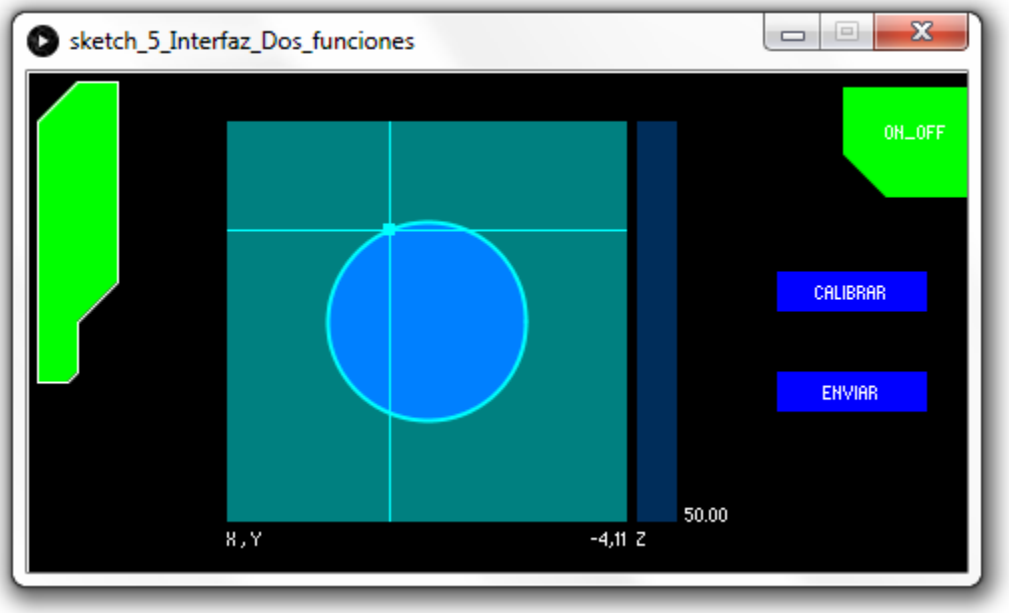


Fig. 117 Interfaz punto de interés -4,11 (X, Y), con altura 50 (Z)

La Fig. 118 muestra el esquema de comunicación del usuario, la interfaz dentro de una PC, la validación del punto de interés solicitado y la transmisión de información a la TIVA utilizando un cable USB.

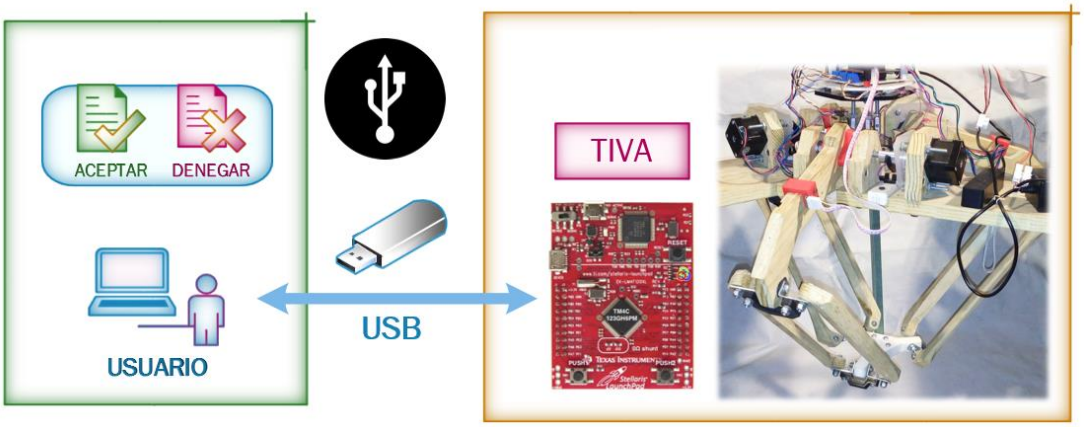


Fig. 118 Diagrama de interacción de la interfaz GUI con el robot

4.3.3 Sistema embebido HMI

Se desarrolló una interfaz Hombre Maquina (HMI) contemplando algunos parámetros básicos como calibración y ejecución que se describen en [34, 35], permitiendo simplificar la interacción con el robot y evitando errores humanos como plantea [36], ya que, al servir como puente de comunicación entre el robot y el usuario, procesa las solicitudes antes de transmitir las señales correspondientes al robot para ejecutar alguna acción, la Fig. 119 muestra el diagrama de interacción con la interfaz y el robot.

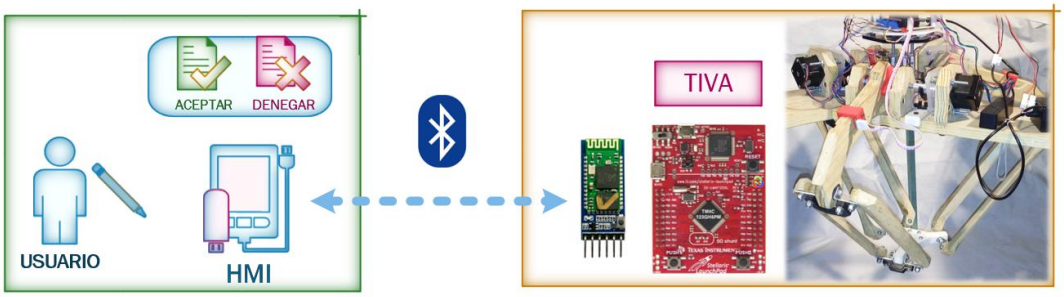


Fig. 119 Diagrama de interacción de la interfaz HMI con el robot

La HMI propuesta únicamente cuenta con el modo de programación en línea para un control manual [34, 37], la cual no cuenta con un monitoreo de información; su desarrollo se describe a continuación.

Con la finalidad de obtener un acercamiento a lo que se conoce como teach pendant [34, 36, 37], se implementa la interfaz anteriormente mostrada en una Raspberry pi 3 modelo B [38, 39], la cual es considerada una computadora con las siguientes características:

- Frecuencia de procesamiento 1.2GHz
- Pines de propósito general entrada/salida: 40
- Ranura para memoria micro SD
- Puerto de alimentación micro USB
- Puerto HDMI
- Puertos USB

Inicialmente a la Raspberry (Fig. 120) se le instala el sistema operativo Raspbian como se muestra en [38, 40, 41]; con el objetivo de consolidar un sistema portátil, se utiliza la pantalla táctil de 3.5 pulgadas que se muestra en la Fig. 121.

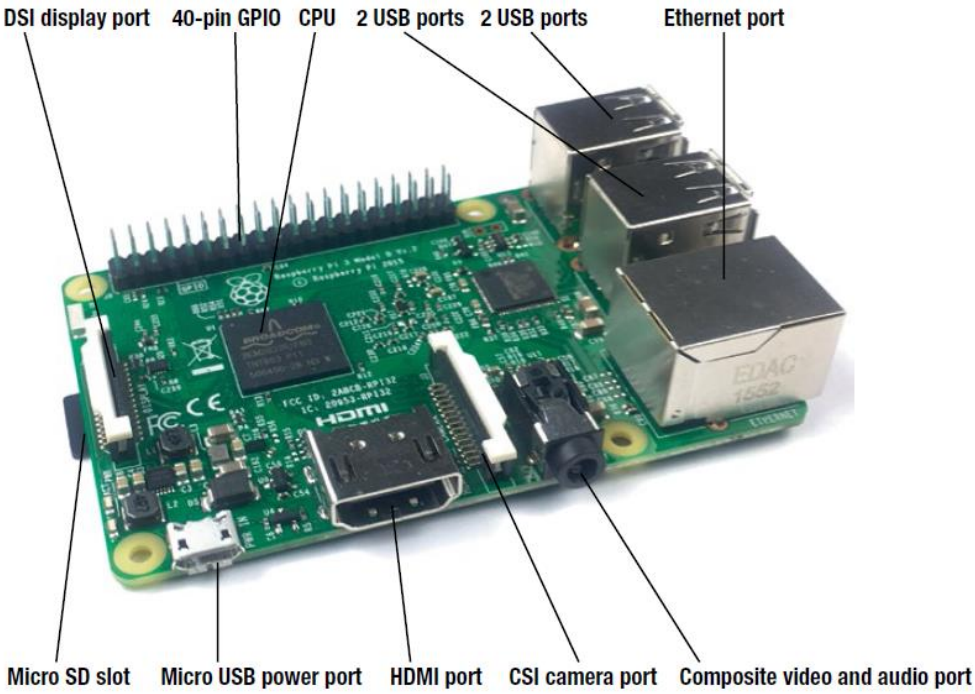


Fig. 120 Diagrama de componentes de la placa Raspberry Pi 3B

Fig. 121 Pantalla de 3.5 pulgadas para Raspberry Pi 3B

La Raspberry Pi 3B conectada a la pantalla táctil se integra en una carcasa de acrílico, como se muestra en Fig. 122.

Fig. 122 Módulo de interfaz

Para hacer que la Raspberry muestre la imagen en la pantalla de 3.5 pulgadas, usando el teclado virtual se ingresa el siguiente código en la terminal:

sudo rm -rf LCD-show

git clone https://github.com/goodtft/LCD-show.git

chmod -R 755 LCD-show

cd LCD-show/

sudo ./LCD35-show

Para mostrar la imagen en el puerto HDMI utilizando una pantalla se ingresa el siguiente código en la terminal

chmod -R 755 LCD-show

cd LCD-show/

sudo ./LCD-hdmi

El módulo de interfaz se alimenta con un banco de baterías de 10000mAh, con salida USB que proporciona 5 Volts a 2.4 Amper de corriente directa; adicionalmente se le conecta un mouse inalámbrico a un puerto USB para obtener una opción adicional de interacción con la computadora; el módulo completo que contempla la tarjeta Raspberry conectada a la pantalla táctil de 3.5 pulgadas, integrada en una carcasa y conectada a la batería se muestra en Fig. 123.

Fig. 123 Modulo remoto de interfaz integrado

Para poder vincular el bluetooth del robot delta con el módulo de interfaz, se requiere que la TIVA se conecte a una batería con capacidad de 2200mAh, con salida USB de 5 Volts a 1 Amper para alimentar al módulo bluetooth como se muestra en las Fig. 124 y Fig. 125.

Fig. 124 Banco de baterías

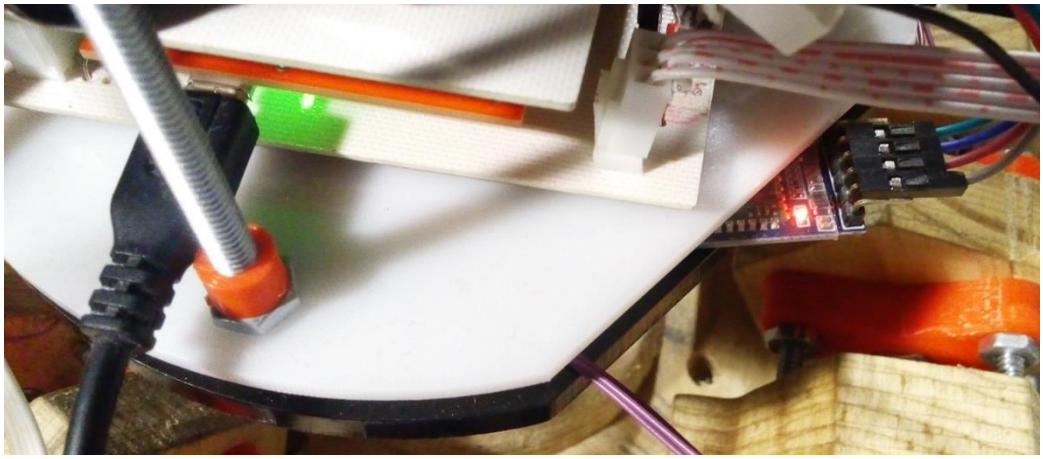


Fig. 125 Controlador TIVA y modulo bluetooth energizados

Al conectar el módulo bluetooth del robot y el módulo remoto de interfaz, se debe introducir por única vez dentro de la terminal de Raspberry el comando *hcitool scan* para detectar los dispositivos bluetooth cercanos al módulo de interfaz, tal comando arroja el nombre y la dirección del dispositivo con el formato 11: 22: 33: 44: 55: 66 [42]; al conocer la dirección del dispositivo, lo anterior mencionado ya no es necesario para los posteriores usos del robot; para vincular el módulo de interfaz con el módulo bluetooth, se debe introducir el comando *rfcomm*, seguido de la dirección del dispositivo bluetooth AA:BB:CC:DD:EE:FF, logrando establecer una comunicación de puerto serie [43], el vínculo correctamente establecido se puede confirmar observando el cambio en el ritmo de parpadeo del indicador del módulo bluetooth del robot; la terminal con el código de creación de puerto serie se muestra en la Fig. 126.

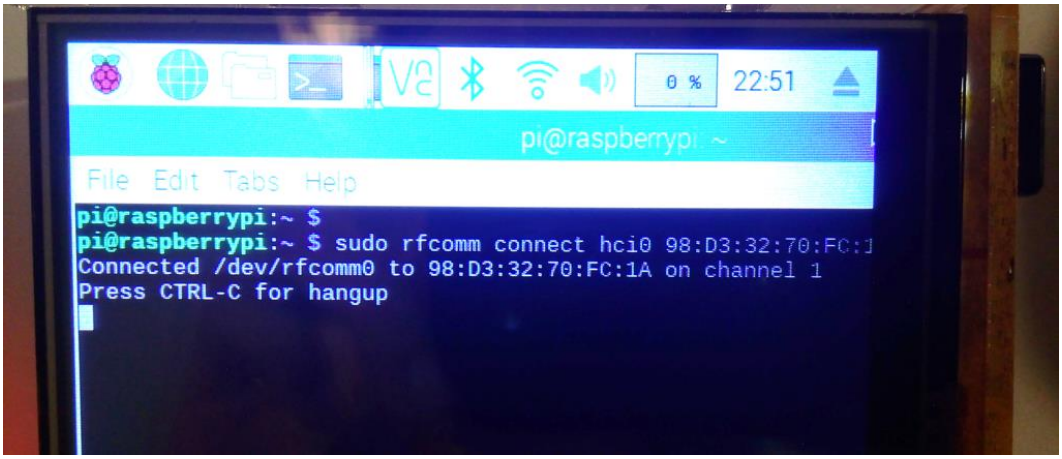


Fig. 126 Terminal de Raspberry con comandos de puerto serie

Al instalar Processing con el comando referido en la página web de Processing https://pi.processing.org/download/ y abrir un sketch nuevo (Fig. 127), el nombre del puerto se puede obtener utilizando la función Serial.list () en un sketch distinto de Processing.

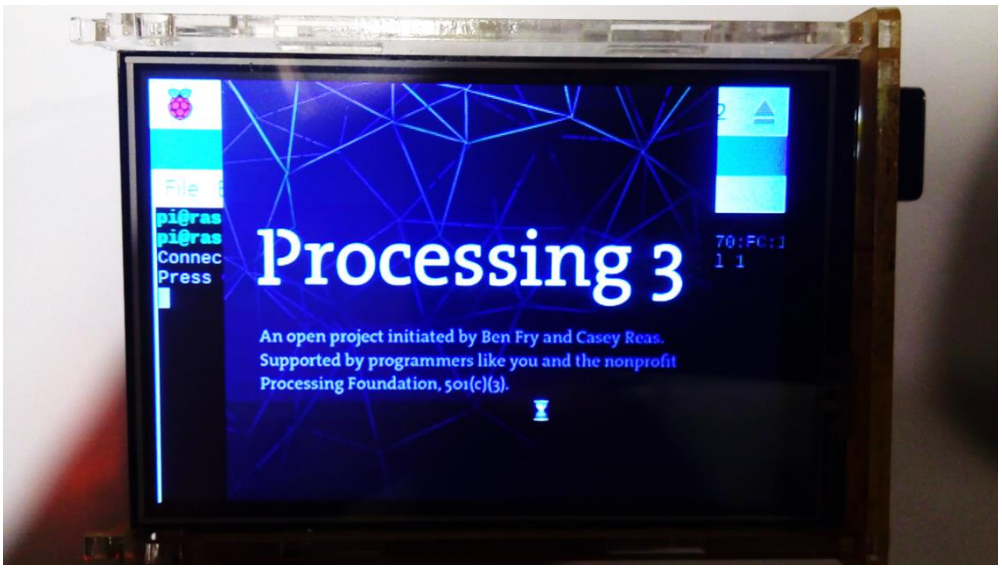


Fig. 127 Inicio de Processing con Raspberry

Se abre el proyecto de la interfaz (GUI) mostrada anteriormente, se cambia el nombre *"COM x"* del puerto serial, por el nombre del puerto serie creado *"/dev/rfcomm0"* (Fig. 128) y se corre el sketch, logrando vincular la GUI dentro del módulo remoto de interfaz (Raspberry) con el módulo bluetooth del robot como muestra la Fig. 129.

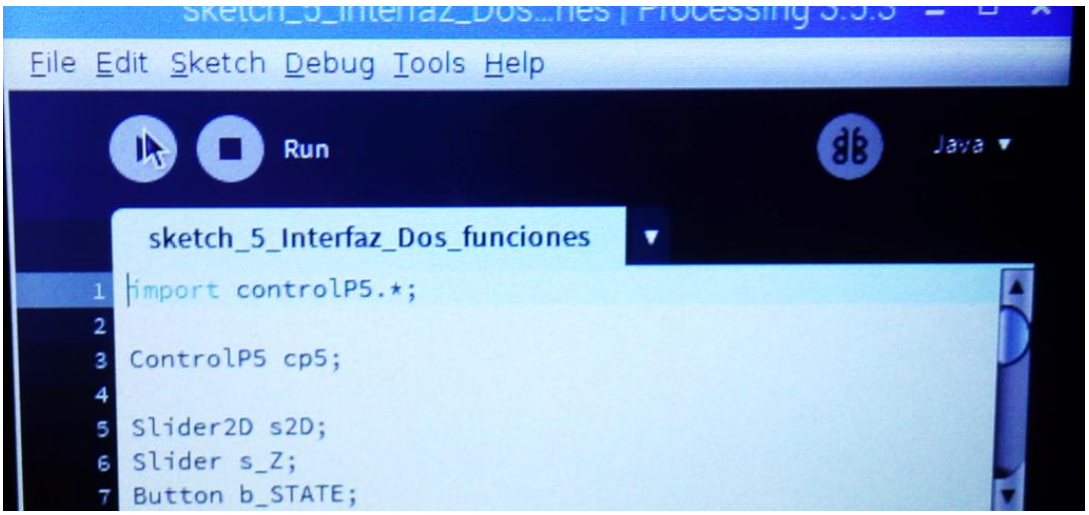


Fig. 128 Sketch de la Interfaz utilizando Raspberry

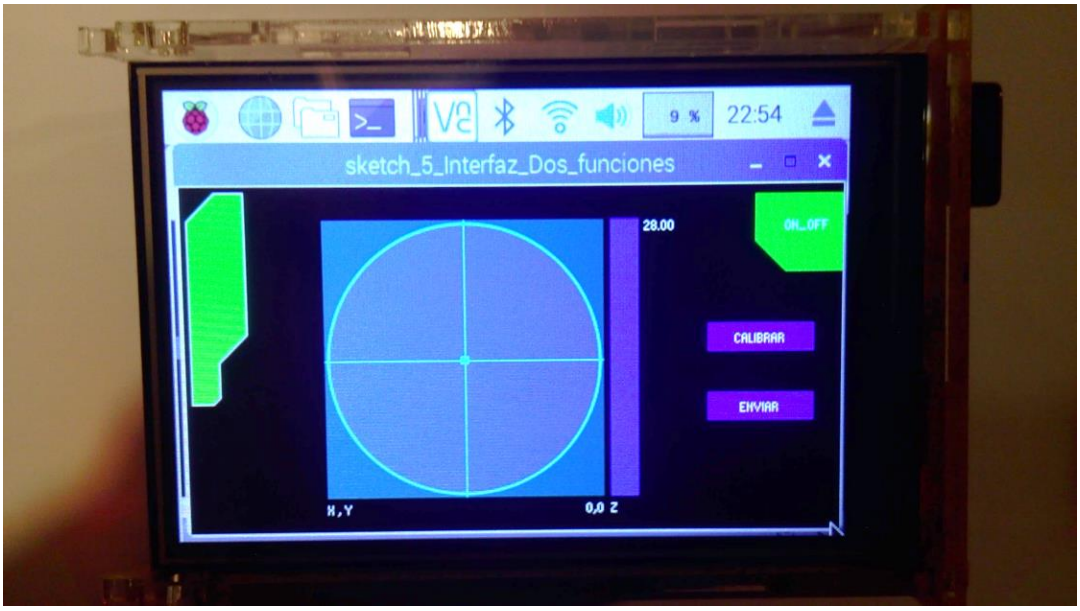


Fig. 129 Interfaz GUI implementada en modulo remoto

5 PRUEBAS Y RESULTADOS

5.1 Pruebas

Las pruebas consisten en solicitar que el robot ejecute las funciones planteadas de calibración y movimiento hacia un punto de interés; la prueba de calibración consiste en llevar cada brazo a la posición angular de 15 grados correspondiente al punto *HOME*, la calibración del robot debe llevarse a cabo inicialmente en cada puesta en marcha del robot; la prueba de movimiento contempla que el robot sea capaz de posicionar la plataforma móvil del robot sobre distintos puntos de interés solicitados por el usuario, las pruebas se describen a continuación.

Las pruebas de movimiento se basan en el posicionamiento de la plataforma móvil sobre el punto de interés con un desplazamiento rectilíneo a través de los ejes X, Y, Z individualmente, sobre dos alturas distintas sin llegar al límite máximo del área de trabajo marcada en la HMI; las alturas son seleccionadas con base en el espacio de trabajo, obteniendo los siguientes casos de puntos a alcanzar con la estructura *Numero de caso) (partiendo del punto inicial) Hacia→ (un punto final).*

1) (Desconocido) → (HOME) Calibración
2) (25,0,-28) → (-25,0,-28) Desplazamiento en X
3) (0,-20,-28) → (0,20,-28) Desplazamiento en Y
4) (0,0,-28) → (0,0,-39) Desplazamiento en Z
5) (-20,0,-39) → (20,0,-39) Desplazamiento en X
6) (0,20,-39) → (0,-20,-39) Desplazamiento en Y

5.2 Resultados

A continuación, se muestran las figuras del robot partiendo del punto inicial hasta llegar al punto final con tres vistas distintas de cada uno de los casos anteriores, tentativamente *1_dimétrica, 2_ligeramente inclinada* y *3_frontal*.

El escenario en el cual se desarrollaron las pruebas se muestra en Fig. 130.

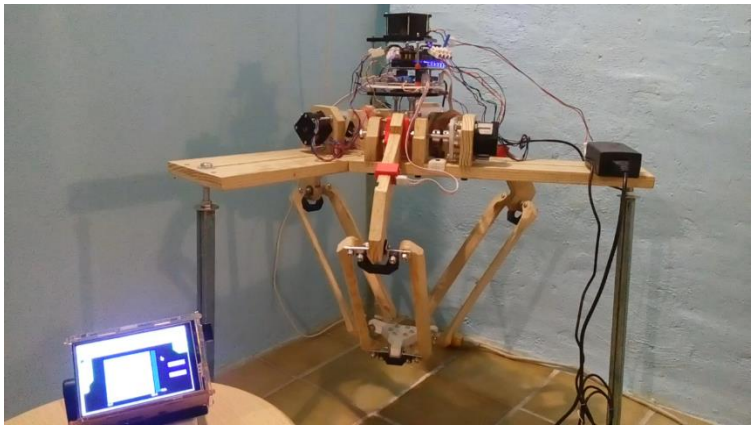


Fig. 130 Robot paralelo tipo delta y HMI en escenario de pruebas

Resultados de la prueba *1) Ejecución de la función de calibración*, la cual lleva a la plataforma móvil de cualquier punto hacia el punto HOME, posicionando cada brazo a 15 grados (Ver figuras Fig. 131, Fig. 132, Fig. 133).

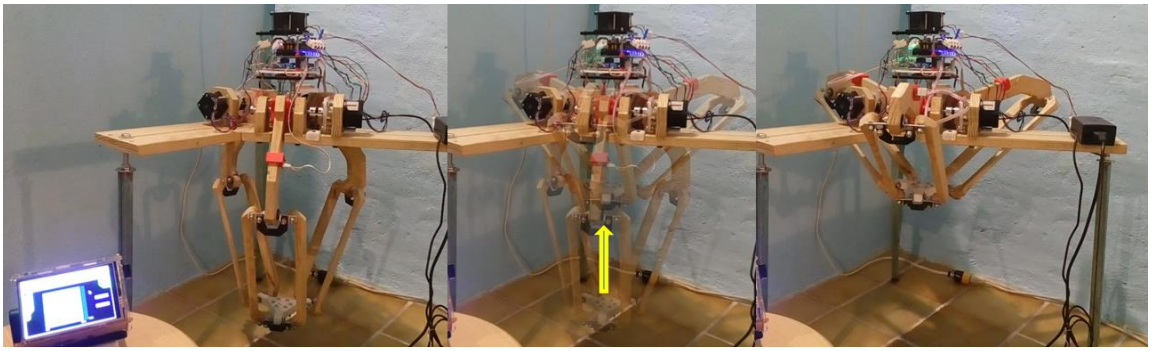


Fig. 131 Resultado de la acción de calibración caso 1 Vista 1

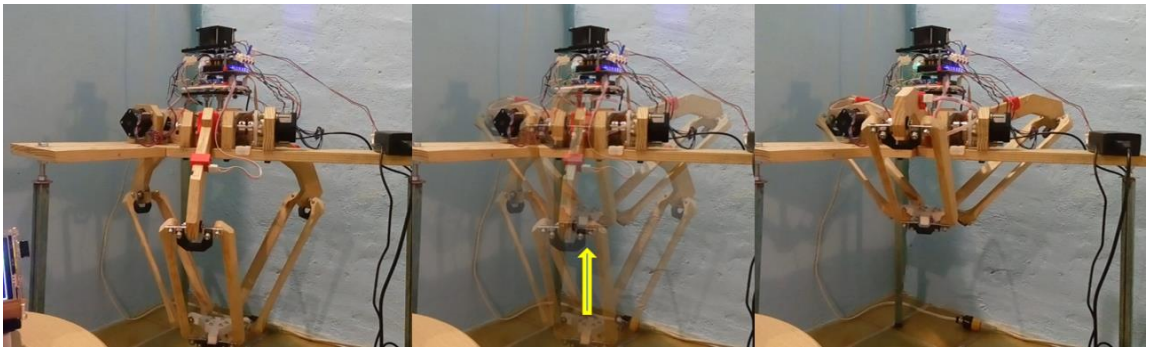


Fig. 132 Resultado de la acción de calibración caso 1 Vista 2

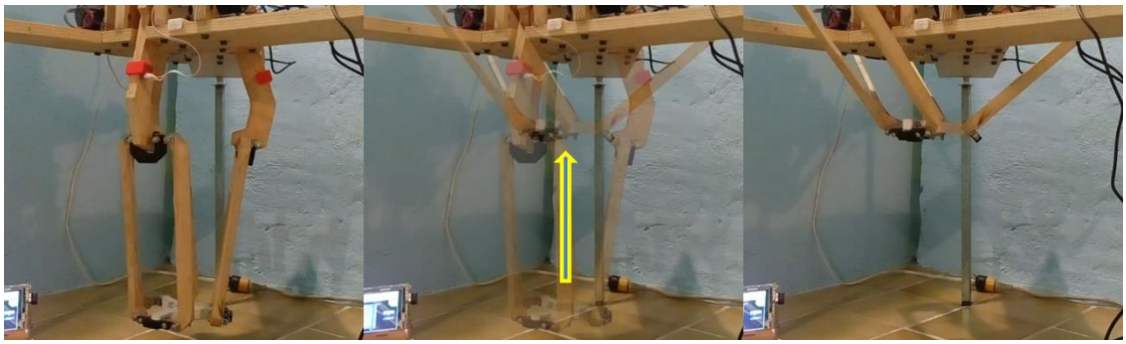


Fig. 133 Resultado de la acción de calibración caso 1 Vista 3

Resultados de la prueba *2) Desplazamiento en X sobre altura de -28*; partiendo del punto inicial (25, 0,-28), hacia el punto final (-25, 0,-28); ver figuras Fig. 134, Fig. 135, Fig. 136.

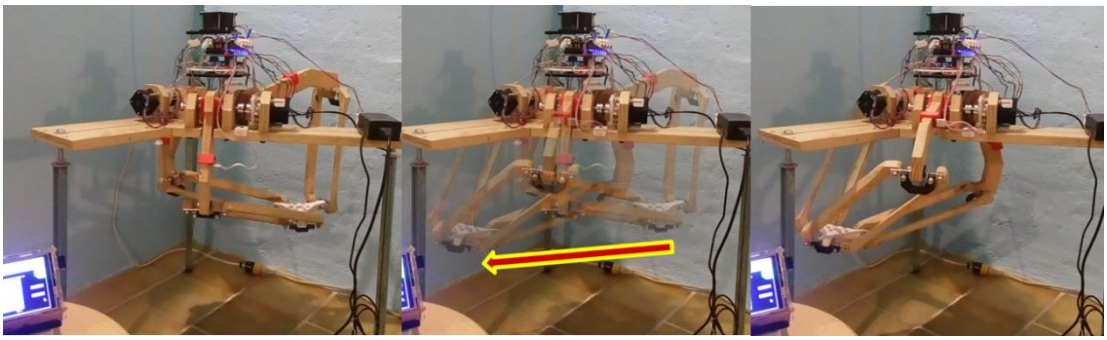


Fig. 134 Resultado de la acción de movimiento caso 2 Vista 1

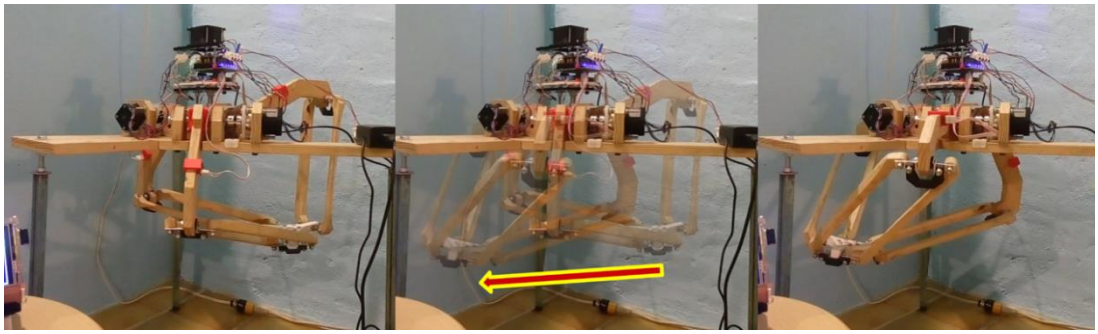


Fig. 135 Resultado de la acción de movimiento caso 2 Vista 2

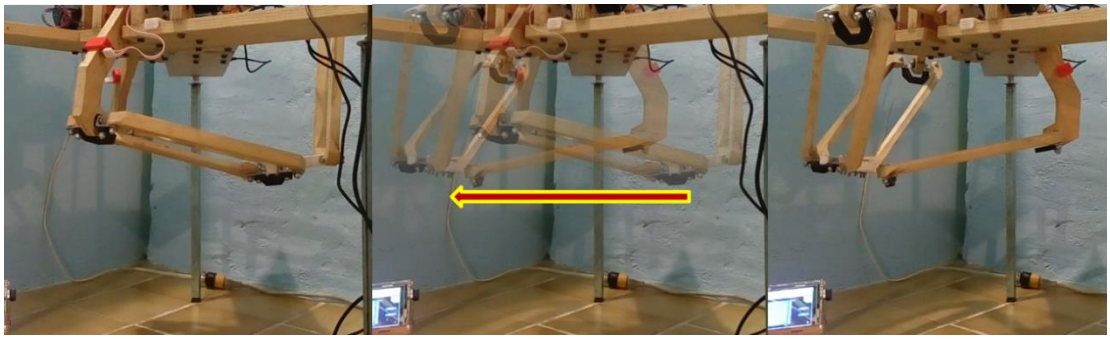


Fig. 136 Resultado de la acción de movimiento caso 2 Vista 3

Resultados de la prueba *3) Desplazamiento en Y sobre altura de -28;* partiendo del punto inicial (0, -20,-28), hacia el punto final (0, 20,-28); ver figuras Fig. 137, Fig. 138, Fig. 139.

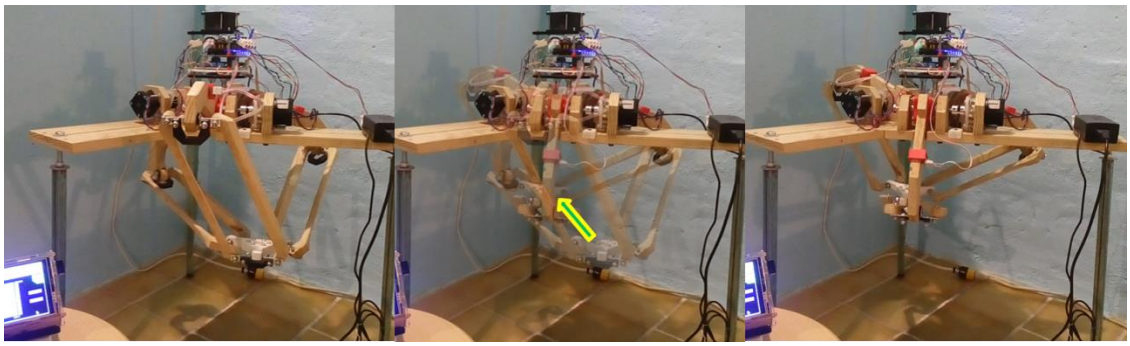


Fig. 137 Resultado de la acción de movimiento caso 3 Vista 1

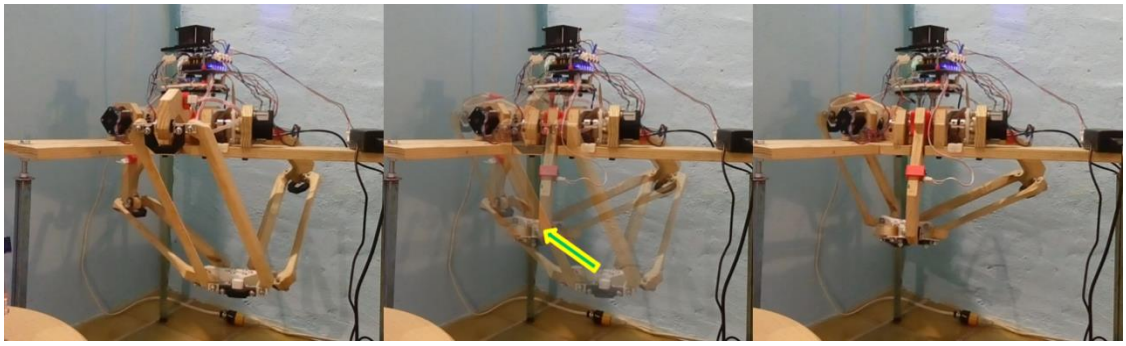


Fig. 138 Resultado de la acción de movimiento caso 3 Vista 2

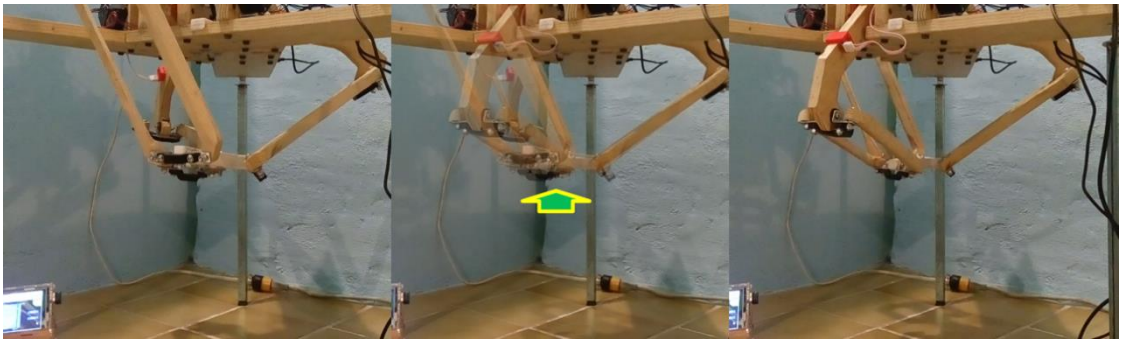


Fig. 139 Resultado de la acción de movimiento caso 3 Vista 3

Resultados de la prueba *4) Desplazamiento en Z*; partiendo del punto inicial (0, 0,-28), hacia el punto final (0, 0,-39); ver figuras Fig. 140, Fig. 141, Fig. 142.

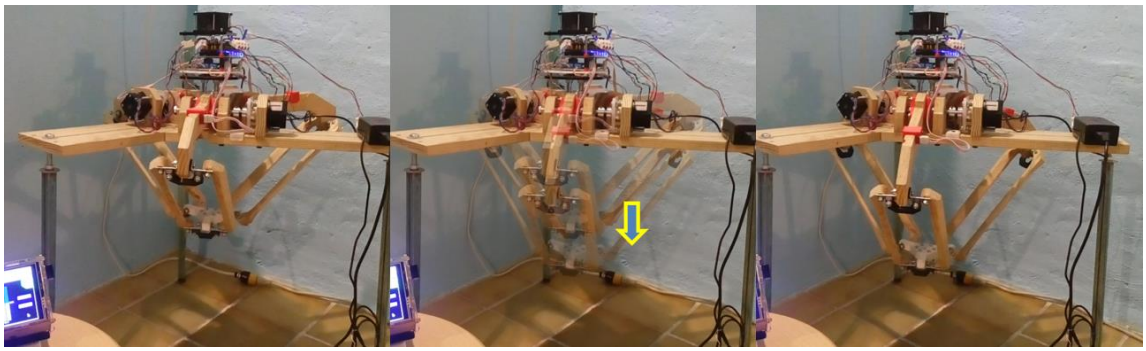


Fig. 140 Resultado de la acción de movimiento caso 4 Vista 1

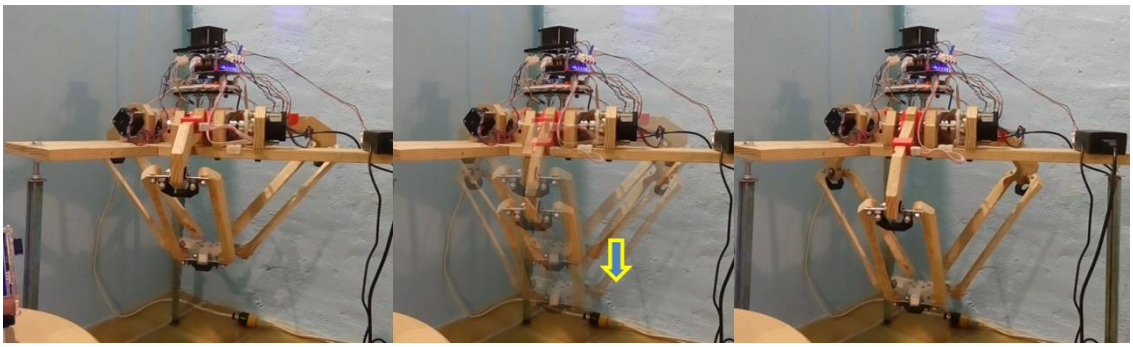


Fig. 141 Resultado de la acción de movimiento caso 4 Vista 2

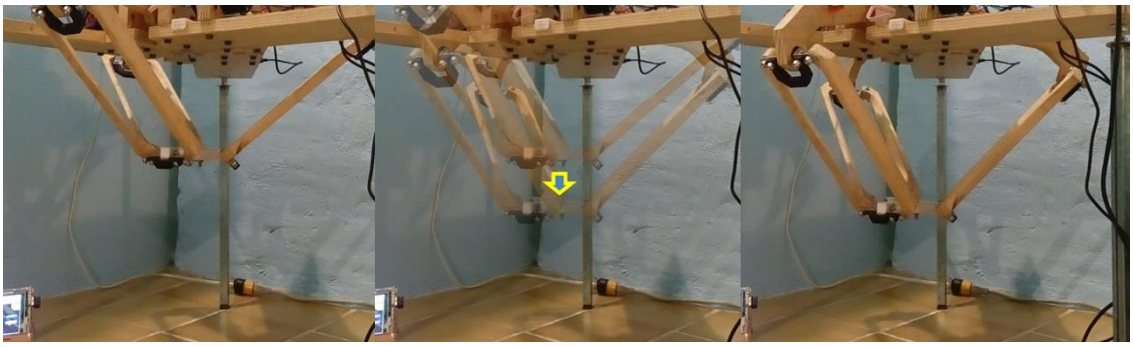


Fig. 142 Resultado de la acción de movimiento caso 4 Vista 3

Resultados de la prueba *5) Desplazamiento en X sobre altura de -39;* partiendo del punto inicial de (-20, 0,-39), hacia el punto final (20, 0,-39); ver figuras Fig. 143, Fig. 144, Fig. 145.

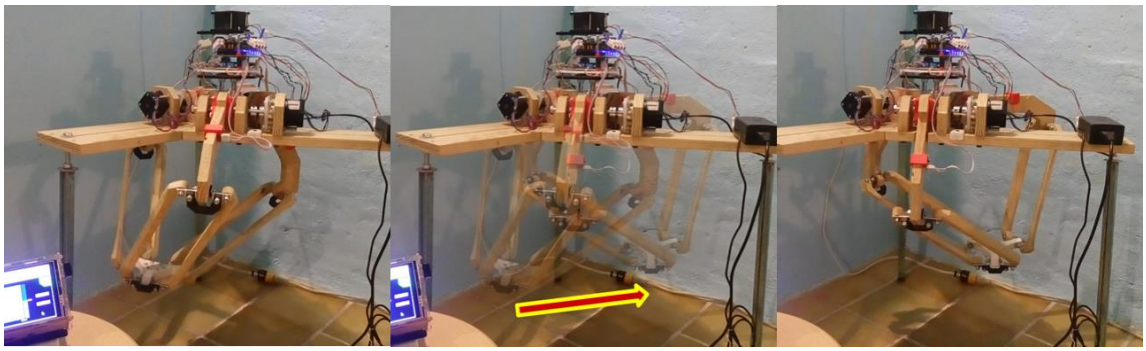


Fig. 143 Resultado de la acción de movimiento caso 5 Vista 1

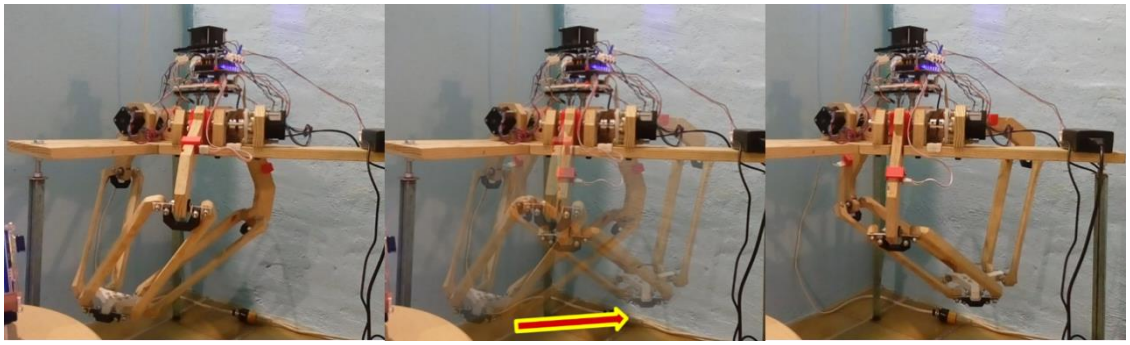


Fig. 144 Resultado de la acción de movimiento caso 5 Vista 2

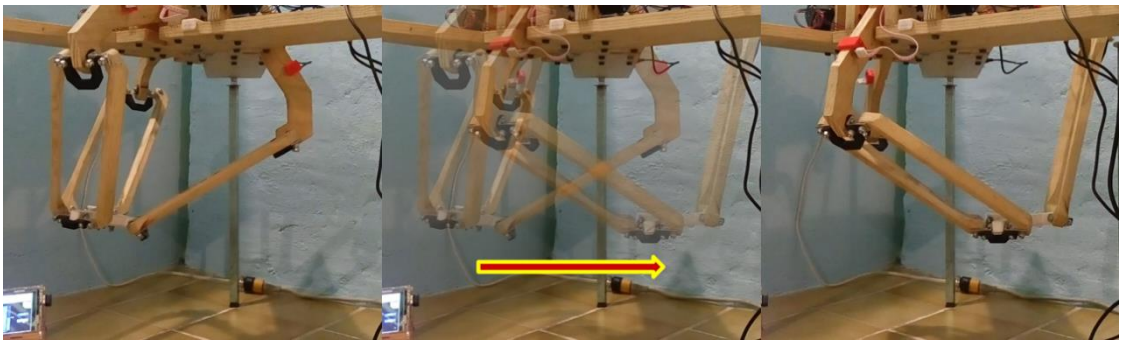


Fig. 145 Resultado de la acción de movimiento caso 5 Vista 3

Resultados de la prueba *6) Desplazamiento en Y sobre altura de -39*; partiendo del punto inicial (0, 20,-39), hacia el punto final (0, -20,-39); ver figuras Fig. 146, Fig. 147, Fig. 148.

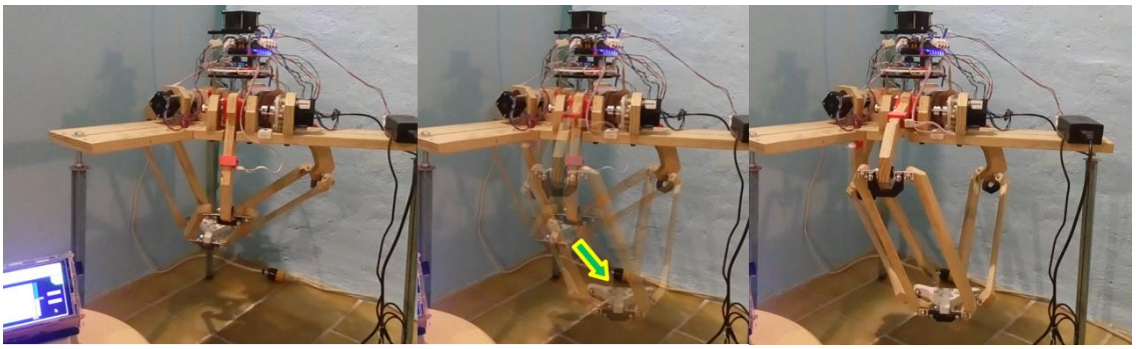


Fig. 146 Resultado de la acción de movimiento caso 6 Vista 1

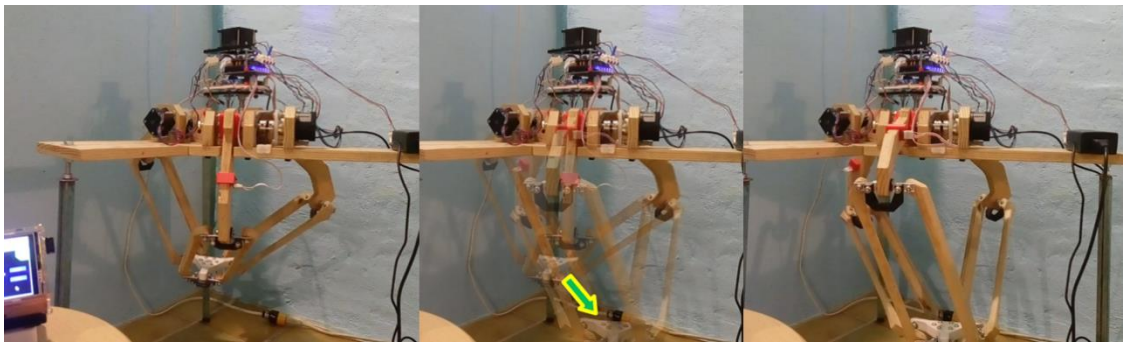


Fig. 147 Resultado de la acción de movimiento caso 6 Vista 2

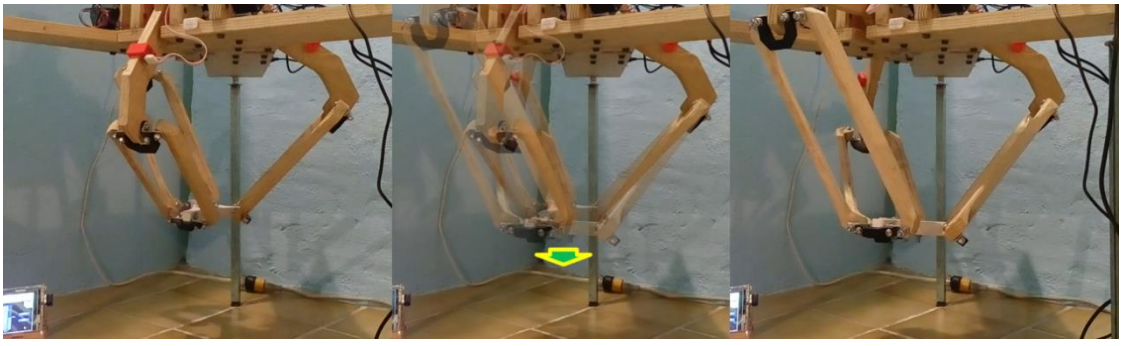


Fig. 148 Resultado de la acción de movimiento caso 6 Vista 3

CONCLUSIONES

Se generó un manipulador paralelo tipo delta, que logra ser un prototipo piloto funcional, se validó la cinemática inversa de este, se diseñó su control y la electrónica necesaria, con el fin de aplicar y comprobar los conocimientos, métodos teóricos y prácticos obtenidos a lo largo de la carrera de Ingeniería Eléctrica Electrónica, así como probar estos conocimientos en una pequeña rama de la ingeniería mecánica, que es un robot paralelo delta y cuya dificultad es poder accionar las tres cadenas cinemáticas de manera simultánea.

Se presenta un sistema integral básico, con el fin de acercar la tecnología a la comunidad en Ingeniería de Facultad de Estudios Superiores Aragón, consolidando una herramienta sencilla y estructurada, apoyada tanto de métodos teóricos como prácticos alcanzando las siguientes metas:

El análisis cinemático de la posición; la generación del diseño mecánico de la plataforma en software de CAD; la realización del análisis de la posición haciendo uso de Matrices de Transformación Homogénea; la simulación del mecanismo e implementación de ecuaciones en el algoritmo de control; la manufactura del manipulador y caracterización de sus elementos; la selección de los actuadores, sensores y controlador; la elaboración de hardware necesario; la programación y adaptación del algoritmo de control; la integración de elementos del sistema robótico y la generación de una HMI.

En la parte de resultados se cuenta con un prototipo funcional, inicialmente las ecuaciones de posición de cada articulación activa (resultado de la solución del problema cinemático inverso), permitieron validar el mecanismo, validar el algoritmo de control de lazo cerrado simulado en Unity y programarlo en la TIVA para su posterior implementación dentro del prototipo piloto, resultando funcional, el posicionamiento de la plataforma móvil del robot delta sobre el punto de interés se cumple en todos los casos mostrados como se observa en el capítulo 5 pruebas y resultados.

El diseño de la electrónica requirió conocimientos de instrumentación, control, dibujo electrónico, programación y una visión amplia del funcionamiento del robot para integrarlos en un prototipo funcional; el diseño de cada nivel de placa está pensado para atender un subsistema específico del robot, el correcto funcionamiento del robot solo es posible con la acción conjunta de las placas.

TRABAJOS A FUTURO

Se pretende utilizar materiales ligeros, rediseñar el manipulador e implementar rodamientos en cada articulación rotacional pasiva, de modo que actúen estrictamente con forme a los ejes de acción del modelo cinemático, ya que los problemas observados en el posicionamiento de la plataforma móvil son derivados del juego mecánico del manipulador.

Se planea sustituir el módulo de acelerómetro GY-61, por un encoder acoplado a la flecha del brazo, para monitorear continuamente la posición angular de cada brazo, resolviendo el problema del movimiento seccionado; dicha modificación requiere rediseñar la placa correspondiente al subsistema de reconocimiento y adaptar el código cargado a la TIVA.

Es posible evolucionar la programación de la HMI y del sistema embebido TIVA, para implementar la programación fuera de línea, la preparación, enseñanza y reproducción de rutinas, así como el control de robots simulados mediante la HMI, en la cual se debe introducir un botón de paro general de emergencia.

Cabe mencionar que los cálculos de las ecuaciones de posición arrojados por la TIVA son menos precisos en comparación con el cálculo realizado en Wolfram Mathematica.

La ejecución del movimiento a lo largo de la trayectoria recta se nota seccionado, lo cual se debe al tiempo de espera que se requiere para adquirir una lectura del módulo de acelerómetro GY-61, ya que detecta oscilaciones que genera el giro del motor en cada paso.

REFERENCIAS

[1] M. Taddei and M. Lisa, *Los robots de Leonardo*: TIKAL EDICIONES, 2010.

[2] F. R. Cortés, *Robótica: Control de Robots Manipuladores*: Marcombo, 2011.

[3] S. K. Saha, *Introducción a la robótica*: McGraw Hill, 2010.

[4] J. P. Merlet, *Parallel Robots*: Springer Netherlands, 2006.

[5] L. W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*: Wiley, 1999.

[6] J. Gallardo-Alvarado, "Delta Robot," in *Kinematic Analysis of Parallel Manipulators by Algebraic Screw Theory*, ed Cham: Springer International Publishing, 2016, pp. 301-316.

[7] J. Gallardo-Alvarado, "An Overview of Parallel Manipulators," in *Kinematic Analysis of Parallel Manipulators by Algebraic Screw Theory*, ed Cham: Springer International Publishing, 2016, pp. 19-28.

[8] J. Gallardo-Alvarado, "Gough's Tyre Testing Machine," in *Kinematic Analysis of Parallel Manipulators by Algebraic Screw Theory*, ed Cham: Springer International Publishing, 2016, pp. 255-280.

[9] J. Gallardo-Alvarado, "The Original Stewart Platform," in *Kinematic Analysis of Parallel Manipulators by Algebraic Screw Theory*, ed Cham: Springer International Publishing, 2016, pp. 281-299.

[10] H. Hidaka, "Introduction," in *Embedded Flash Memory for Embedded Systems: Technology, Design for Sub-systems, and Innovations*, H. Hidaka, Ed., ed Cham: Springer International Publishing, 2018, pp. 1-6.

[11] A. Holt and C.-Y. Huang, "Introduction," in *Embedded Operating Systems: A Practical Approach*, ed London: Springer London, 2014, pp. 1-12.

[12] P. Marwedel, "Introduction," in *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*, ed Dordrecht: Springer Netherlands, 2011, pp. 1-19.

[13] "ARM® Cortex®-M4 Other Peripherals Programming," in *Practical Microcontroller Engineering with ARM® Technology*, ed, pp. 805-926.

[14] A. Barrientos, A. B. Cruz, L. F. Peñín, and C. Balaguer, *Fundamentos de robótica*: McGraw-Hill, 2007.

[15] Z. Xiong, C. Zhuang, and J. Wu, "Rigid-Body Motions," in *Handbook of Manufacturing Engineering and Technology*, A. Y. C. Nee, Ed., ed London: Springer London, 2015, pp. 1717-1776.

[16] R. L. Norton, *DISEÑO DE MAQUINARIA*: McGraw-Hill Interamericana de España S.L., 2009.

[17] S. Blackman, "Physics and Special Effects," in *Beginning 3D Game Development with Unity 4: All-in-One, Multi-Platform Game Development*, ed Berkeley, CA: Apress, 2013, pp. 391-449.

[18] S. Blackman, "Message Text," in *Beginning 3D Game Development with Unity 4: All-in-One, Multi-Platform Game Development*, ed Berkeley, CA: Apress, 2013, pp. 451-481.

[19] N. Sclater and N. Chironis, *Mechanisms and Mechanical Devices Sourcebook, Fourth Edition*: Mcgraw-hill, 2007.

[20] D. V. Gadre and S. Gupta, "Tiva C Series LaunchPad," in *Getting Started with Tiva ARM Cortex M4 Microcontrollers: A Lab Manual for Tiva LaunchPad Evaluation Kit*, ed New Delhi: Springer India, 2018, pp. 27-31.

[21] D. V. Gadre and S. Gupta, "ARM Cortex-M4 Core and Tiva C Series Peripherals," in *Getting Started with Tiva ARM Cortex M4 Microcontrollers: A Lab Manual for Tiva LaunchPad Evaluation Kit*, ed New Delhi: Springer India, 2018, pp. 13-26.

[22] A. Fernandez and D. Dang, "Chapter 4 - Meet Energia—a Software Development Environment," in *Getting Started with the MSP430 Launchpad*, A. Fernandez and D. Dang, Eds., ed Oxford: Newnes, 2013, pp. 21-34.
[23] H. Czichos, "Testing," in *Measurement, Testing and Sensor Technology: Fundamentals and Application to Materials and Technical Systems*, ed Cham: Springer International Publishing, 2018, pp. 25-42.
[24] A. K. Gupta, "Regression and Model Fitting," in *Numerical Methods using MATLAB*, ed Berkeley, CA: Apress, 2014, pp. 107-118.
[25] I. Greenberg, D. Xu, and D. Kumar, "Diving into the Shallow End," in *Processing: Creative Coding and Generative Art in Processing 2*, ed Berkeley, CA: Apress, 2013, pp. 1-32.
[26] D. V. Gadre and S. Gupta, "Universal Asynchronous Receiver and Transmitter (UART)," in *Getting Started with Tiva ARM Cortex M4 Microcontrollers: A Lab Manual for Tiva LaunchPad Evaluation Kit*, ed New Delhi: Springer India, 2018, pp. 151-167.
[27] "ARM® Cortex®-M4 Parallel I/O Ports Programming," in *Practical Microcontroller Engineering with ARM® Technology*, ed, pp. 433-546.
[28] V. Protopopov, "Photoreceivers," in *Practical Opto-Electronics: An Illustrated Guide for the Laboratory*, ed Cham: Springer International Publishing, 2014, pp. 77-110.
[29] D. V. Gadre and S. Gupta, "Analog to Digital Converter (ADC)," in *Getting Started with Tiva ARM Cortex M4 Microcontrollers: A Lab Manual for Tiva LaunchPad Evaluation Kit*, ed New Delhi: Springer India, 2018, pp. 183-209.
[30] I. Greenberg, D. Xu, and D. Kumar, "Expressive Power of Data," in *Processing: Creative Coding and Generative Art in Processing 2*, ed Berkeley, CA: Apress, 2013, pp. 149-185.
[31] L. G. Gutiérrez, *Instrumentación básica de medida y control*: AENOR, 2010.
[32] "Stepper," in *Arduino™ Sketches*, ed, pp. 253-260.
[33] P. SZNAJDLEDER, *Algoritmos a Fondo: - Con implementaciones en c y java*: Alfaomega Grupo Editor, 2017.
[34] Y. Gao, Y. Su, W. Dong, W. Wang, Z. Du, X. Gao*, et al.*, "U-Pendant: A universal teach pendant for serial robots based on ROS," in *2014 IEEE International Conference on Robotics and Biomimetics, IEEE ROBIO 2014, December 5, 2014 - December 10, 2014*, Bali, Indonesia, 2014, pp. 2529-2534.
[35] M. Gao, L. Ye, and Y. Yan, "Design and implementation of teach pendant for six degrees of freedom industrial robot," in *5th International Conference on Instrumentation and Measurement, Computer, Communication, and Control, IMCCC 2015, September 18, 2015 - September 20, 2015*, Qinhuangdao, China, 2015, pp. 1929-1933.
[36] H. Roibu, D. Popescu, M.-M. Abagiu, and N.-G. Bizdoaca, "Human-machine interfaces for robotic system control," in *2018 International Conference on Applied and Theoretical Electricity, ICATE 2018, October 4, 2018 - October 6, 2018*, Craiova, Romania, 2018, p. Association for Support in Engineering Education; Ministry of Research and Innovation.
[37] I. Mehta, K. Bimbraw, R. G. Chittawadigi, and S. K. Saha, "A teach pendant to control virtual robots in Roboanalyzer," in *1st International Conference on Robotics and Automation for Humanitarian Applications, RAHA 2016, December 18, 2016 - December 20, 2016*, Kerala, India, 2016.
[38] G. Hart-Davis, "Choosing Raspberry Pi Hardware," in *Deploying Raspberry Pi in the Classroom*, ed Berkeley, CA: Apress, 2017, pp. 17-54.
[39] A. Pajankar, "Introduction to Single Board Computers and Raspberry Pi," in *Raspberry Pi Image Processing Programming: Develop Real-Life Examples with Python, Pillow, and SciPy*, ed Berkeley, CA: Apress, 2017, pp. 1-24.
[40] J. Cicolani, "An Introduction to Raspberry Pi," in *Beginning Robotics with Raspberry Pi and Arduino: Using Python and OpenCV*, ed Berkeley, CA: Apress, 2018, pp. 17-45.
[41] J. R. Strickland, "Meet the Raspberry Pi," in *Raspberry Pi for Arduino Users: Building IoT and Network Applications and Devices*, ed Berkeley, CA: Apress, 2018, pp. 35-61.

[42] "Bluetooth Tools in Linux," in *Bluetooth Essentials for Programmers*, A. S. Huang and L. Rudolph, Eds., ed Cambridge: Cambridge University Press, 2007, pp. 181-192.

[43] "Other Platforms and Environments," in *Bluetooth Essentials for Programmers*, A. S. Huang and L. Rudolph, Eds., ed Cambridge: Cambridge University Press, 2007, pp. 157-180.

Anexos

Datasheet tarjeta electrónica de evaluación TM4C123G

Tiva™ C Series TM4C123G LaunchPad Evaluation Board

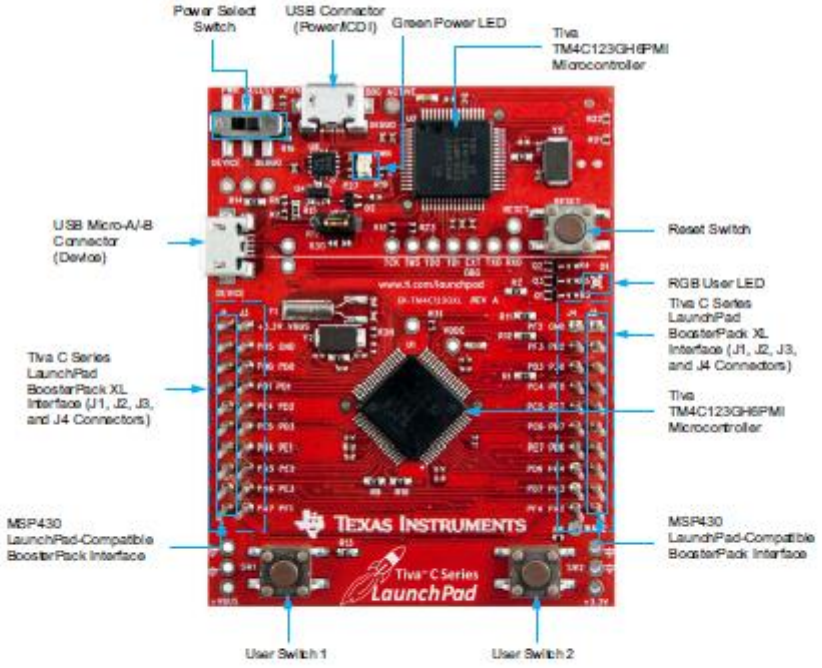
User's Guide

Literature Number: SPMU296
April 2013

Board Overview

The Tiva™ C Series TM4C123G LaunchPad Evaluation Board (EK-TM4C123GXL) is a low-cost evaluation platform for ARM® Cortex™-M4F-based microcontrollers. The Tiva C Series LaunchPad design highlights the TM4C123GH6PMI microcontroller USB 2.0 device interface, hibernation module, and motion control pulse-width modulator (MC PWM) module. The Tiva C Series LaunchPad also features programmable user buttons and an RGB LED for custom applications. The stackable headers of the Tiva C Series TM4C123G LaunchPad BoosterPack XL interface demonstrate how easy it is to expand the functionality of the Tiva C Series LaunchPad when interfacing to other peripherals on many existing BoosterPack add-on boards as well as future products. Figure 1-1 shows a photo of the Tiva C Series LaunchPad.

Figure 1-1. Tiva C Series TM4C123G LaunchPad Evaluation Board



1.1 Kit Contents

The Tiva C Series TM4C123G LaunchPad Evaluation Kit contains the following items:

- Tiva C Series LaunchPad Evaluation Board (EK-TM4C123GXL)
- On-board In-Circuit Debug Interface (ICDI)
- USB micro-B plug to USB-A plug cable
- README First document

1.2 Using the Tiva C Series LaunchPad

The recommended steps for using the Tiva C Series TM4C123G LaunchPad Evaluation Kit are:

1. **Follow the README First document included in the kit.** The README First document will help you get the Tiva C Series LaunchPad up and running in minutes. See the Tiva C Series LaunchPad web page for additional information to help you get started.
2. **Experiment with LaunchPad BoosterPacks.** A selection of Tiva C Series BoosterPacks and compatible MSP430™ BoosterPacks can be found at the TI MCU LaunchPad web page.
3. **Take your first step toward developing an application with Project 0 using your preferred ARM tool-chain and the Tiva C Series TivaWare Peripheral Driver Library.** Software applications are loaded using the on-board In-Circuit Debug Interface (ICDI). See Chapter 3, ***Software Development***, for the programming procedure. The TivaWare for C Series Peripheral Driver Library Software Reference Manual contains specific information on software structure and function. For more information on Project 0, go to the Tiva C Series LaunchPad wiki page.
4. **Customize and integrate the hardware to suit an end application.** This user's manual is an important reference for understanding circuit operation and completing hardware modification.

You can also view and download almost six hours of training material on configuring and using the LaunchPad. Visit the Tiva C Series LaunchPad Workshop for more information and tutorials.

1.3 Features

Your Tiva C Series LaunchPad includes the following features:

- Tiva TM4C123GH6PMI microcontroller
- Motion control PWM
- USB micro-A and micro-B connector for USB device, host, and on-the-go (OTG) connectivity
- RGB user LED
- Two user switches (application/wake)
- Available I/O brought out to headers on a 0.1-in (2.54-mm) grid
- On-board ICDI
- Switch-selectable power sources:
  - ICDI
  - USB device
- Reset switch
- Preloaded RGB quickstart application
- Supported by TivaWare for C Series software including the USB library and the peripheral driver library
- Tiva C Series TM4C123G LaunchPad BoosterPack XL Interface, which features stackable headers to expand the capabilities of the Tiva C Series LaunchPad development platform
  - For a complete list of available BoosterPacks that can be used with the Tiva C Series LaunchPad, see the LaunchPad web page.

1.4 BoosterPacks

The Tiva C Series LaunchPad provides an easy and inexpensive way to develop applications with the TM4C123GH6PM microcontroller. Tiva C Series BoosterPacks and MSP430 BoosterPacks expand the available peripherals and potential applications of the Tiva C Series LaunchPad. BoosterPacks can be used with the Tiva C Series LaunchPad or you can simply use the on-board TM4C123GH6PM microcontroller as its processor. See Chapter 2 for more information.

Build your own BoosterPack and take advantage of Texas Instruments' website to help promote it! From sharing a new idea or project, to designing, manufacturing, and selling your own BoosterPack kit, TI offers a variety of avenues for you to reach potential customers with your solutions.

1.5 Specifications

Table 1-1 summarizes the specifications for the Tiva C Series LaunchPad.

Table 1-1. EK-TM4C123GXL Specifications

Parameter	Value
Board supply voltage	4.75 V_{DC} to 5.25 V_{DC} from one of the following sources: • Debugger (ICDI) USB Micro-B cable (connected to a PC) • USB Device Micro-B cable (connected to a PC)
Dimensions	2.0 in x 2.25 in x 0.425 in (5.0 cm x 5.715 cm x 10.795 mm) (L x W x H)
Break-out power output	• 3.3 V_{DC} (300 mA max) • 5.0 V_{DC} (depends on 3.3 V_{DC} usage, 23 mA to 323 mA)
RoHS status	Compliant

Datasheet módulo controlador de motores a pasos DRV8825

DRV8825 Stepper Motor Controller IC

1 Features

- PWM Microstepping Stepper Motor Driver
  - Built-In Microstepping Indexer
  - Up to 1/32 Microstepping
- Multiple Decay Modes
  - Mixed Decay
  - Slow Decay
  - Fast Decay
- 8.2-V to 45-V Operating Supply Voltage Range
- 2.5-A Maximum Drive Current at 24 V and T_A = 25°C
- Simple STEP/DIR Interface
- Low Current Sleep Mode
- Built-In 3.3-V Reference Output
- Small Package and Footprint
- Protection Features
  - Overcurrent Protection (OCP)
  - Thermal Shutdown (TSD)
  - VM Undervoltage Lockout (UVLO)
  - Fault Condition Indication Pin (nFAULT)

2 Applications

- Automatic Teller Machines
- Money Handling Machines
- Video Security Cameras
- Printers
- Scanners
- Office Automation Machines
- Gaming Machines
- Factory Automation
- Robotics

3 Description

The DRV8825 provides an integrated motor driver solution for printers, scanners, and other automated equipment applications. The device has two H-bridge drivers and a microstepping indexer, and is intended to drive a bipolar stepper motor. The output driver block consists of N-channel power MOSFET's configured as full H-bridges to drive the motor windings. The DRV8825 is capable of driving up to 2.5 A of current from each output (with proper heat sinking, at 24 V and 25°C).

A simple STEP/DIR interface allows easy interfacing to controller circuits. Mode pins allow for configuration of the motor in full-step up to 1/32-step modes. Decay mode is configurable so that slow decay, fast decay, or mixed decay can be used. A low-power sleep mode is provided which shuts down internal circuitry to achieve very low quiescent current draw. This sleep mode can be set using a dedicated nSLEEP pin.

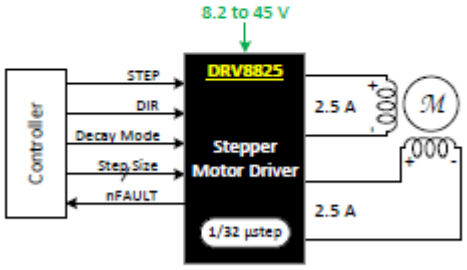
Internal shutdown functions are provided for overcurrent, short circuit, under voltage lockout and over temperature. Fault conditions are indicated via the nFAULT pin.

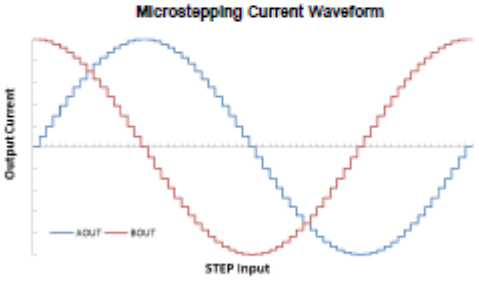
Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
DRV8825	HTSSOP (28)	9.70 mm × 6.40 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

4 Simplified Schematic





Datasheet sensor acelerómetro ADXL335

ANALOG DEVICES

Small, Low Power, 3-Axis ±3 *g* Accelerometer

ADXL335

FEATURES

- 3-axis sensing
- Small, low profile package
  - 4 mm × 4 mm × 1.45 mm LFCSP
- Low power : 350 µA (typical)
- Single-supply operation: 1.8 V to 3.6 V
- 10,000 *g* shock survival
- Excellent temperature stability
- BW adjustment with a single capacitor per axis
- RoHS/WEEE lead-free compliant

APPLICATIONS

- Cost sensitive, low power, motion- and tilt-sensing applications
  - Mobile devices
  - Gaming systems
  - Disk drive protection
  - Image stabilization
  - Sports and health devices

GENERAL DESCRIPTION

The ADXL335 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The product measures acceleration with a minimum full-scale range of ±3 *g*. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.

The user selects the bandwidth of the accelerometer using the C_X, C_Y, and C_Z capacitors at the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for the X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis.

The ADXL335 is available in a small, low profile, 4 mm × 4 mm × 1.45 mm, 16-lead, plastic lead frame chip scale package (LFCSP_LQ).

FUNCTIONAL BLOCK DIAGRAM

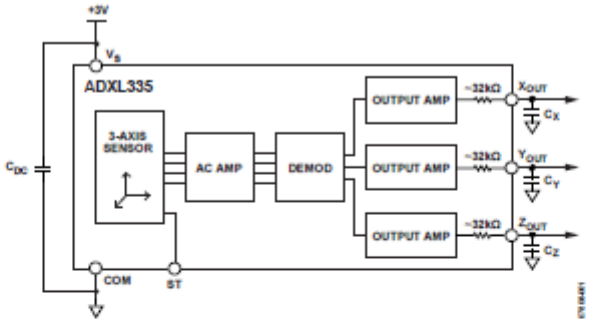


Figure 1.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113

Datasheet Motor a pasos con caja de engranes planetarios 5:1