Universidad Nacional Autónoma de México

FACULTAD DE INGENIERÍA

Diseño y Control de un brazo robótico de 3 grados de libertad.

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO ELÉCTRICO Y ELECTRÓNICO

P R E S E N T A:

LUIS ROMÁN HERNÁNDEZ TORRES

Director de tesis:

DR. PAUL ROLANDO MAYA ORTÍZ

MÉXICO, D. F. **2015**

Jurado:

PRESIDENTE: DR. MARCO ANTONIO ARTEAGA PEREZ

VOCAL: DR. PAUL ROLANDO MAYA ORTIZ

SECRETARIO: DR. EDMUNDO GABRIEL ROCHA COZATL

1ER. SUPLENTE: DR. MARCOS ANGEL GONZALEZ OLVERA

2DO. SUPLENTE: M.I. JAVIER PLIEGO JIMENEZ

Esta tesis fue elaborada en el Laboratorio de Robótica del Edificio de Estudios de Posgrado de la Facultad de Ingeniería; a cargo del Dr. Paul Rolando Maya Ortíz.

Tutor de tesis:

Dr. Paul Rolando Maya Ortíz.

A mi madre, Rosa María

A mi padre, Jorge

A mis hermanos, Tania y Jorge

Agradecimientos

A la Universidad Nacional Autónoma de México por ofrecerme tanto en mi paso como estudiante, de la cual me siento orgulloso en pertenecer.

Al Dr. Paul Rolando Maya Ortíz quien me aceptó en primer lugar para desempeñar como tesista, le agradezco la oportunidad que me brindo en trabajar con usted.

Finalmente, a todas las personas que me acompañaron durante mi formación profesional sin ellos no hubiera sido posible este logro, pero principalmente agradecerle a mis padres por darme la oportunidad de conocer nuevas fronteras y ver como el mas pequeño crece junto a sus hermanos. Por esto y mas... ¡Gracias!

Índice de figuras

Índice de cuadros

Índice general

Capítulo 1

Introducción

Cuando se piensa en robótica, lo primero que se viene a la mente son los robots, pero lo que muchos no saben es que hay mas de la robótica que solo eso. La robótica no es solo la creación de robots, sino la ciencia y la tecnología detrás de su fabricación. Esta ciencia es la mente detrás de su diseño, producción y programación.

La robótica[1] es una rama de la ciencia que se encarga de asegurar que los robots puedan realizar y desempeñar las mismas tareas realizadas por seres humanos que requieren del uso de la inteligencia. En otras palabras, la robótica en todo sentido es la mente detrás de la operación de los robots.

Los robots pueden ser de diferentes diseños, todo depende de la función que vayan a realizar. Lo que si se conoce son las diferentes características que pueden poseer, entre estas encontramos:

- La precisión que tienen a la hora de realizar una acción o movimiento.
- La capacidad de carga, en kilogramos que el robot puede manejar.
- El grado de libertad que tienen con sus movimientos.
- El sistema de coordenadas que especifica a que direcciones se realizarán sus movimientos y posiciones. Estas pueden ser coordenadas cartesianas (x,y,z), cilíndricas, al igual que polares.
- La programación de cada robot o el poder de aprendizaje que cada uno tiene.

Los robots pueden ser de diferentes tipos, cuyas clasificaciones, dependiendo del tipo de sección asociada se pueden observar en la Figura 1.1.

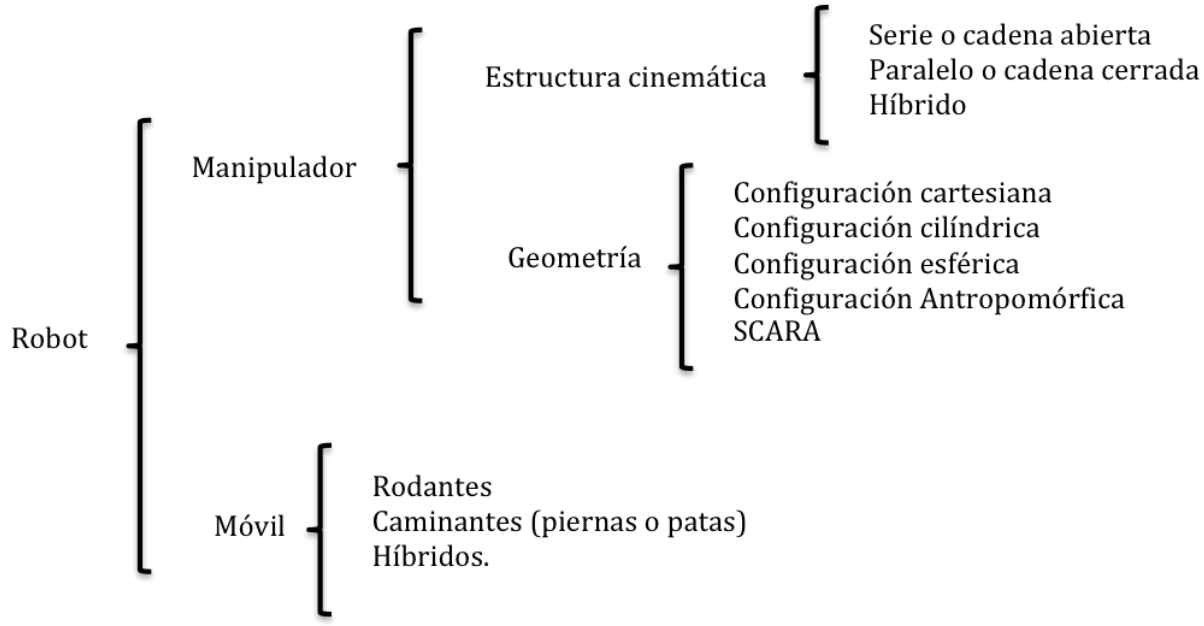


Figura 1.1: Tipos de Robot

1.1. Antecedentes

A medida que hay avances tecnológicos, la investigación científica requiere de equipo capaz de responder a las necesidades y exigencias de los investigadores, ingenieros y estudiantes que día con día trabajan en ciencia y tecnología aplicada. Para el campo de la Robótica principalmente se requieren estudiantes para el manejo de hardware y software especializado en la adquisición de datos que les permita fortalecer conocimientos del área de robots industriales.

Es de suma importancia que los algoritmos desarrollados en laboratorios de investigación sean probados en un equipo que permita validar teorías, por lo que se busca tener plataformas de fácil implementación. A su vez, por motivos prácticos, es conveniente el uso de

equipo existente y su modernización mediante software y hardware actualizado.

Actualmente en el Laboratorio de Robótica del Edificio de Posgrado de la Facultad de Ingeniería tienen un robot de 3 grados de libertad con configuración Antropomorfica, el Robot es conocido como Omni Phantom[2]. En el Robot Omni Phantom se han implementado controladores para el seguimiento de trayectorias. Este Robot esta diseñado para realizar háptica y actividades de teleoperación. Si al Robot Omni Phantom se le llegará a dañar un componente interno o externo, se complica el intercambiar la pieza por una nueva, ya que el fabricante desarrolla sus propias piezas y son difíciles de adquirir, por lo que se tendría que conseguir un nuevo robot.

En el Laboratorio de Control de Robots Industriales de la Facultad de Ingeniería de la UNAM existe un Robot Mentor, este robot tiene 5 grados de libertad en configutación Antropomórfica. El Robot Mentor tiene la desventaja de que el usuario no puede implementar controladores sobre el, debido a que internamente ya tiene un controlador y esta diseñado para que el usuario solo pueda proporcionar las posiciones articulares deseadas para que el efector final se desplace.

1.2. Motivación

La principal motivación de este trabajo es mostrar la importancia de crear software y hardware libre para el diseño y control de un brazo robótico para fines didácticos, así como las ventajas y desventajas que tiene el uso de componentes del Robot.

Para el robot de 3 grados de libertad de esta tesis se tomaron como base algunos robots que ya estan desarrollados, como es el caso del Robot Omni Phantom y el Robot Mentor. Los 2 robots mencionados tienen un arreglo cinemático del tipo Antropomórfico, la diferencia son los grados de libertad que contiene cada uno de ellos. La idea parte de poder crear un Robot de bajo costo, para que los alumnos del Laboratorio de Control Industriales de la Facultad de Ingeniería puedan implementar los conceptos vistos en la teoría y comprobar sus resultados en la práctica.

Actualmente el Robot Mentor es utilizado por los alumnos de la licenciatura en Ingeniería Eléctrica y Electrónica. En él se realizan pruebas de los conceptos básicos del área, por lo que la implementación de nuevos robots permitirá comprender con mayor facilidad los conceptos de Control de Robots Industriales.

Es conveniente que los robots utilizados por estudiantes cuenten con un medio real en donde sea posible generar trayectorias con algunos obstáculos fijos que restrinjan el espacio de trabajo del Robot.

1.3. Objetivos de la tesis

El objetivo general de este trabajo es el diseño, construcción y análisis del comportamiento de un robot de 3 grados de libertad. El diseño e implementación de un Robot de 3 grados de libertad para el laboratorio de Control de Robots Industriales, servirá de base para la comprensión de algoritmos de control para los alumnos de Licenciatura en Ingeniería Eléctrica y Electrónica.

Los objetivos particulares en el trabajo de tesis se encuentran:

- La implementación de la cinemática directa y cinemática inversa mediante el uso de interfaces en MATLAB.
- Realizar un controlador PID para el seguimiento de trayectorias.
- Utilizar un método de sintonización para el controlador PID.
- Simular y probar su comportamiento en el Robot a realizar.

1.4. Planteamiento del problema

En el diseño del robot es necesario verificar que tipos de motores son de gran conveniencia utilizar para la manipulación del mismo, además de que se pueden implementar distintas formas de enlazar los motores con una computadora, ya sea con microcontroladores o controladores desarrollados por algunos fabricantes.

Para el desarrollo del robot se deben determinar las dimensiones de cada articulación y eslabón, con respecto a los motores que se utilizarán. El diseño de este robot es nuevo, por lo que se debe analizar su comportamiento mediante las cinemáticas y dinámicas que actuán en cada una de las articulaciónes.

1.5. Logros y aportaciones

En la realización de este trabajo se utilizaron conceptos vistos a lo largo de la licenciatura, como Modelado de Sistemas Físicos, Fundamentos de Control, Control Avanzado, Electrónica, Programación, etc.

Los elementos principales que se utilizaron en el Robot son 3 servomotores Dynamixel AX-12, Material en MDF, circuitos electricos y un microcontrolador Atmega 2560. Estos sirven de entorno para el robot en donde se pueden planear trayectorias. Uno de los principales logros de este trabajo es el diseño de un brazo liviano y pequeño.

Mediante la cinemática directa e inversa se obtuvieron las ecuaciones que determinan el comportamiento cinemático del efector final, es decir, permite mapear la posición de cada articulación en un sistema coordenado. El modelo dinámico es necesario para poder simular el comportamiento del robot, y aproximarse al control.

La aportación de esta tesis es el desarrollo de un Robot Antropomórfico, tomando en cuenta los primeros 3 grados de libertad que dan la posición del brazo, en este caso no se toman en cuenta los ultimos 3 grados de libertad que representan la muñeca esférica. Se desarrolla un prototipo para el laboratorio de Control de Robots Industriales, esto permite que los estudiantes puedan tener mayor uso de material didactico para comprender los conceptos de la materia.

En el caso de este trabajo permite que el usuario pueda desplazar el brazo respecto al análisis de la cinemática directa e inversa y el desarrollo de trayectorias del robot, mediante el uso de interfaces en MATLAB & SIMULINK, debido a que los robots industriales adquiridos están restringidos a que solo puedan ser manipulados mediante interfaces propietarias. También se disminuye el costo de equipo el robot desarrollado en esta tesis, debido a que normalmente la mayoría de los fabricantes de robots industriales diseñan equipo de alto costo para el publico en general.

1.6. Organización del trabajo

El presente trabajo está organizado en cinco capítulos. En el Capítulo 2 se explica la debida asignación para la aplicación del algoritmo de Denavit-Hartenberg, para la cinemática directa e inversa del robot. Consecutivamente se hace el análisis del modelo dinámico del robot

de 3 grados de libertad, tomando en cuenta las fuerzas que actúan sobre cada articulación del robot. Obtenidos los datos necesarios del Robot, se simula su comportamiento con un controlador PID mediante el uso de trayectorias en MATLAB & SIMULINK.

El Capítulo 3 esta centrado en el diseño del robot, primero se describen todos los componentes que integran al robot que se desarrolló, así como el funcionamiento que tiene cada uno de ellos. Después se explica el diseño de los circuitos eléctricos necesarios para la adquisición de datos de los motores, el diseño de la estructura mecánica mediante el análisis de los motores a ocupar, así como una breve explicación de como se desarrolló el software para el microcontrolador y MATLAB & SIMULINK. Finalmente se hace un análisis de costos de producción de todo el robot.

El Capítulo 4 esta enfocado a la implementación y resultados, en este capítulo se presentan pruebas del robot, mediante el uso de la cinemática directa e inversa, las sintonización del controlador PID y la generación de trayectorias mediante un control PID. Finalmente en el capítulo 5 se presentan las conclusiones del trabajo de investigación.

Capítulo 2

Análisis Cinemático y Dinámico del Robot de 3 GDL

Existen distintos tipos de Robots Industriales, el que se analizará en este trabajo de tesis es el Robot Antropomorfico de 3 grados de libertad. Todas las articulaciones del robot son de tipo rotacional, estas permiten desplazar la posición del efector final a un punto localizado en el espacio de trabajo.

Para analizar el comportamiento de un robot, el primer paso es la obtención de los modelos cinemáticos, el cual, proporciona información sobre el movimiento del robot considerando únicamente sus características geométricas e ignorando las fuerzas y pares que lo generan.

2.1. Cinemática Directa

Para determinar las ecuaciones cinemáticas, se utilizará el método matricial propuesto por Denavit y Hartenberg[3] en 1955. El objetivo que tiene la cinemática directa es poder determinar la posición del efector final respecto a un sistema base, con ecuaciones en función únicamente de las variables articulares del robot, mientras que la cinemática inversa es útil para determinar las variables articulares cuando se conoce la posición del efector final respecto a un sistema base.

El establecimiento de los sistemas coordenados se asignan en base a la convención del algoritmo de Denavit-Hartenberg como se observa en la Figura 2.1.

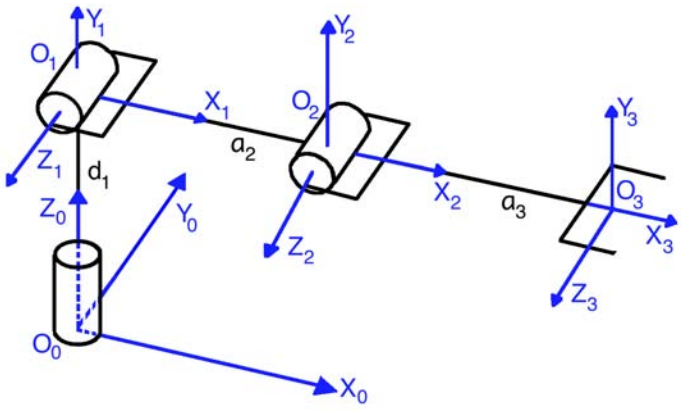


Figura 2.1: Establecimiento de sistemas coordenados y ángulos de las articulaciones

Mediante la asignación de los sistemas coordenados, se obtienen los parámetros que van a representar la movilidad de cada articulación del robot. La obtención de los parámetros se observa en la Tabla 2.1.

Eslabón	a_i	α	d_i	θ_i
1	0	$\pi/2$	d_1	θ_1^*
2	a_2	0	0	θ_2^*
3	a_3	0	0	θ_3^*

Tabla 2.1 Parámetros Denvit-Hartenberg del robot

donde,
θ_i: Es el ángulo medido del eje $\mathbf{x}_{i-1}$ al eje $\mathbf{x}_i$ teniendo como eje de giro a $\mathbf{z}_{i-1}$.
d_i es la distancia del sistema coordenado i-ésimo$_{-1}$ hasta la intersección de los ejes $\mathbf{z}_{i-1}$ y $\mathbf{x}_i$ medida sobre $\mathbf{z}_{i-1}$.
a_i es la distancia del sistema coordenado i-ésimo hasta la intersección de $\mathbf{z}_{i-1}$ y $\mathbf{x}_i$ medida sobre $\mathbf{x}_i$.
α_i es el ángulo medido del eje $\mathbf{z}_{i-1}$ al eje $\mathbf{z}_i$, tomando como eje de giro al eje $\mathbf{x}_i$.

Tomando en cuenta las restricciones de la representación de Denavit-Hartenberg se puede representar cualquier desplazamiento o rotación con las transformaciones homogéneas $\mathbf{A}_i$ de cada eslabón:

$$\mathbf{A}_i = \begin{bmatrix} {}^0\mathbf{R}_i & {}^0\mathbf{O}_i \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

donde, ${}^0\mathbf{R}_i$ es una matriz de rotación, que representa la rotación del sistema coordenado i-ésimo, respecto al sistema base y ${}^0\mathbf{O}_i$ representa la traslación del sistema coordenado i-ésimo, respecto al sistema base. Estas matrices son necesarias para obtener la matriz de inercia del robot.

Durante el desarrollo de este trabajo se obtuvieron las matrices homogéneas del robot, están dadas por:

$$\mathbf{A}_1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$\mathbf{A}_2 = \begin{bmatrix} c_2c_1 & s_2c_1 & s_1 & a_2c_2s_1 \\ c_2s_1 & s_2s_1 & c_1 & a_2c_2c_1 \\ s_2 & c_2 & 0 & a_2s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$\mathbf{A}_3 = \begin{bmatrix} c_3c_1 & -s_3c_1 & s_1 & (a_3c_{23} + a_2c_2)c_1 \\ c_3s_1 & -s_3s_1 & -c_1 & (a_3c_{23} + a_2c_2)s_1 \\ s_3 & c_3 & 0 & a_3s_{23} + a_2s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

De (2.4) se obtienen las ecuaciones para la Cinemática Directa:

$$x = (a_3c_{23} + a_2c_2)c_1 \tag{2.5}$$

$$y = (a_3c_{23} + a_2c_2)s_1 \tag{2.6}$$

$$z = a_3s_{23} + a_2s_2 + d_1 \tag{2.7}$$

donde, x, y, z son la posición del efector final sobre el sistema coordenado base (x_0, y_0, z_0).

Para poder hacer el análisis del robot se obtuvo el par máximo de cada uno de los motores que actúan en las articulaciones, esto depende del tamaño de cada eslabón, los cuales son los que se tomarán en cuenta para el diseño del robot, por lo que d_1: 11.5 [cm], a_2: 11.0 [cm] y a_3: 16.5 [cm]. [1]

[1]Nota: La información del Robot diseñado se encuentran en el Diseño mecánico en las pag. 50-53

2.2. Cinemática Inversa

El objetivo de la cinemática inversa es determinar las variables articulares en función únicamente de la posición del efector final. En general, es más complicado que la cinemática directa, ya que no siempre puede obtenerse una solución analítica cerrada a este problema. Sin embargo, se puede obtener una solución para el problema cinemático inverso mediante un enfoque geométrico.

Para obtener las variables articulares es conveniente hacer proyecciones sobre planos para comprender su comportamiento. En este sentido, la solución para θ_1 resulta sencilla de apreciar, así que la primer variable articular que se obtendrá es θ_1. Para esto basta con hacer una proyección de x, y, z, (Posición del efector final) sobre el plano $\mathbf{x}_0$ y $\mathbf{y}_0$, nótese que no hay relación del efector en $\mathbf{z}_0$ con θ_1.

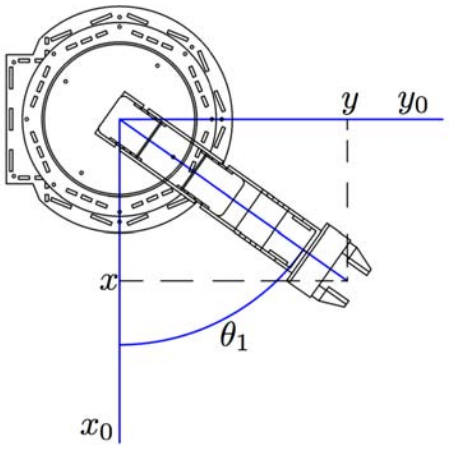


Figura 2.2: Vista del plano $\mathbf{x}_0$ y $\mathbf{y}_0$, para la obtención de θ_1

De la Figura 2.2 se obtiene:

$$\theta_1 \;=\; \text{atan2}(\text{y}, \text{x}) \tag{2.8}$$

Para poder determinar θ_2 y θ_3 se hace una proyección sobre el plano $\mathbf{z}_1$; $\mathbf{y}_1$, tal que sea más evidente la trayectoria que pueden describir los eslabones del manipulador. Es importante mencionar que se obtendrá primero θ_2, ya que θ_3 está en función de θ_2.

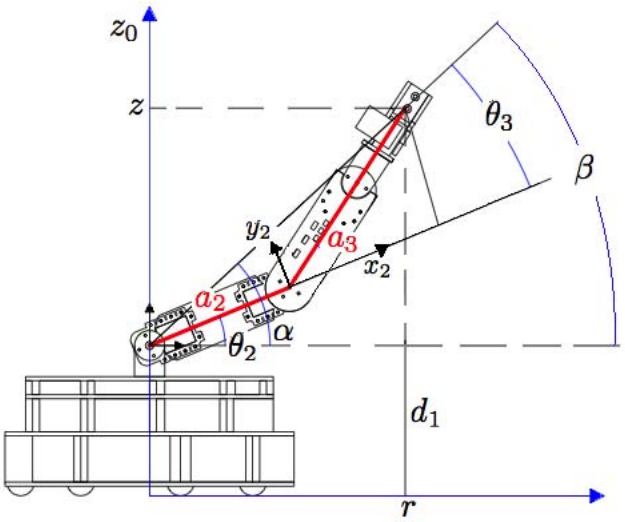


Figura 2.3: Proyección sobre el plano $\mathbf{z}_0$ y $\mathbf{y}_0$, para mostrar la obtención de θ_2 y θ_3

Para la obtención de θ_2 y θ_3, se puede apreciar mejor si se hace un corte de un plano en Z y la posición que se encuentre el efector final entre X y Y, como se observa en la Figura 2.4.

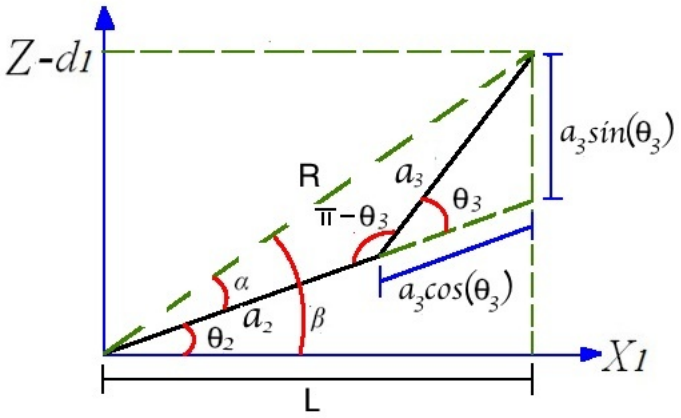


Figura 2.4: Asignación del sistema coordenado para θ_2 y θ_3

De la Figura 2.4 se observa que θ_2 es la resta de $\beta - \alpha$ donde β es el ángulo formado por los segmentos $Z_c - d_1$ y L; α es el ángulo formado por los segmentos L y $a_2 + a_3 \cos\theta_3$, para determinar θ_2 se utilizarán relaciones trigonométricas obtenidas a partir del triángulo formado por los segmentos de la Figura 2.4.

Para obtener θ_2:

$$\beta = atan2(Z_c - d_1, L) \tag{2.9}$$

$$\alpha = atan2(a_3 \sin(\theta_3), a_3 \cos(\theta_3) + a_2) \tag{2.10}$$

$$\theta_2 = \beta - \alpha \tag{2.11}$$

Sustituyendo (2.10) y (2.11) en (2.12), se obtiene que θ_2:

$$\theta_2 = atan2(Z_c - d_1, L) - atan2(a_3 sin(\theta_3), a_3 \cos(\theta_3) + a_2) \tag{2.12}$$

Para obtener θ_3 se toma en cuenta el triángulo imaginario formado por a_2 y a_3 de la Figura 2.4:

$$R^2 = a_2^2 + a_3^2 + 2a_2 a_3 \cos(\pi - \theta_3) \tag{2.13}$$

Despejando $\cos(\pi - \theta_3)$ de (2.14) se tiene:

$$\cos(\pi - \theta_3) = \frac{a_2^2 + a_3^2 - R^2}{2a_2 a_3} \tag{2.14}$$

Simplificando (2.15):

$$\cos(\theta_3) = \frac{R^2 - a_2^2 - a_3^2}{2a_2 a_3} = D \tag{2.15}$$

De (2.16) de D, se obtiene θ_3, tomando en cuenta que se desea controlar con codo arriba(-):

$$\theta_3 = atan2(-\sqrt{1 - D^2}, D) \tag{2.16}$$

2.3. Modelo Dinámico

El modelo dinámico puede ser formulado por medio de cantidades de energía. El modelo de un sistema mecánico rígido, con n grados de libertad, puede ser obtenido usando el método de Euler-Lagrange. Este método proporciona las ecuaciones de estado en forma explicita y esta pueden ser utilizadas para analizar y diseñar estrategias de control. Para implementar estrategias de control es muy importante tener en cuenta todas las fuerzas que interactúan con el robot, como lo son las inercias, fuerzas de Coriolis, centrífugas y el efecto de la gravedad sobre cada uno de los eslabones.

Para modelar el Robot se utilizarán las ecuaciónes de Euler - Lagrange, estas ecuaciones hacen un balance de energías de la dinámica, tanto cinetica como potencial, este análisis se puede representar como un lagrangiano de un sistma mecánico dado como:

$$\mathcal{L} = \mathcal{K} - \mathcal{P}$$

donde K es la energía cinética y P es la energía potencial, por lo tanto la ecuación Euler-Lagrange para cada grado de libertad del manipulador es:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \;\; i = 1, ..., n$$

Estas ecuaciones pueden escribirse en forma matricial como:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{D}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \tag{2.17}$$

donde la matriz $\mathbf{H}(\mathbf{q})\ \epsilon\ \mathbb{R}^{nxn}$ es la matriz de inercia del robot, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\ \epsilon\ \mathbb{R}^{nxn}$ es la matriz de Coriolis, $\mathbf{D}\ \epsilon\ \mathbb{R}^{nxn}$ es una matriz diagonal con los coeficientes de fricción y $\mathbf{g}\ \epsilon\ \mathbb{R}^{n}$ es el efecto de las fuerzas gravitacionales ejercidas sobre cada articulación.

En el desarrollo del análisis del robot se tomó como base el modelo dinámico de la Puesta en marcha del robot Omni Phantom de Sensable[2], debido a que tiene caracteristicas similares al desarrollado en este trabajo.

2.3.1. Momento de inercia del robot

El momento de inercia depende de la geometría del cuerpo y de la posición del eje de giro. La inercia es la propiedad de los materiales a oponerse al cambio en el movimiento. En el caso mas general, la inercia rotacional debe representarse por medio de un conjunto de momentos de inercia y componentes que forman el tensor de inercia. Para el caso de este trabajo los tensores de inercia se consideraron como una varilla rígida, por lo que los tensores de inercia del robot son:

$$\mathbf{I}_1 = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_1 \end{bmatrix}$$

$$\mathbf{I}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_2 \end{bmatrix}$$

$$\mathbf{I}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & I_3 & 0 \\ 0 & 0 & I_3 \end{bmatrix}$$

$I_n = \frac{m_n l_n^2}{12}$, donde n representa el número de eslabón correspondiente.

El cálculo de la matriz de inercia del robot se obtiene mediante la energía cinética que cada articulación genera, ya que dependen únicamente de la masa y de su velocidad. La representación de la *Matriz de inercia de un robot*[1] para n grados de libertad se representa como:

$$\mathbf{H(q)} = \sum_{i=1}^{n} m_i \mathbf{J}_{vci}^T \mathbf{J}_{vci} + \mathbf{J}_{\omega \mathbf{ci}}^{\mathbf{T}} \, {}^{\mathbf{0}}\mathbf{R}_{\mathbf{ci}} \, \mathbf{I}_{\mathbf{ci}} \, {}^{\mathbf{0}}\mathbf{R}_{\mathbf{ci}}^{\mathbf{T}} \, \mathbf{J}_{\omega_{\mathbf{c}}\mathbf{i}} \tag{2.18}$$

Donde:
m_i es la masa de cada eslabón
$\mathbf{J}_{vci}$ es el Jacobiano de velocidad lineal del eslabón i
$\mathbf{J}_{\omega ci}$ es el Jacobiano de velocidad angular del eslabón i
${}^0\mathbf{R}_i$ Matrices de rotación del sistema coordenado i respecto al sistema base
$\mathbf{I}_{ci}$ Tensor de inercia.

Del Robot Antropomórfico se realizo el análisis necesario, tomando como base las matrices de velocidad cinemática del Robot Omni Phantom, por lo que se obtuvieron como resultado:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{2.19}$$

De la matriz $\mathbf{H}(\mathbf{q})$, se definieron los parámetros b_1, b_2 y b_3 como se observa en la ecuación (2.20), (2.21) y (2.22):

$$b_1 = m_3 O_{c3} + \frac{1}{12} m_3 a_3^2; \tag{2.20}$$

$$b_2 = a_2 m_3 O_{c3} \tag{2.21}$$

$$b_3 = m_2 O_{c2} + a_2^2 m_3 + \frac{1}{12} m_2 a_2^2 \tag{2.22}$$

La matriz $\mathbf{H}(\mathbf{q})$ esta definida como:
$h_{11} = b_1 c_3^2 + 2 b_2 c_2 c_3 + b_3 c_2^2$
$h_{12} = 0$
$h_{13} = 0$
$h_{21} = 0$
$h_{22} = 2 b_2 s_2 s_3 + 2 b_2 c_2 c_3 + b_1 + \theta_3$
$h_{23} = b_2 s_2 s_3 + b_2 c_2 c_3 + b_1$
$h_{31} = 0$
$h_{32} = b_2 s_2 s_3 + b_2 c_2 c_3 + b_1$
$h_{33} = b_1$

2.3.2. Coriolis y fuerzas centrifugas

La fuerza de Coriolis ocurre cuando un objeto, al desplazarse sobre cualquier sistema rotacional sufre una aceleración adicional producida por una fuerza perpendicular al movimiento. El resultado que provoca al objeto, es una desviación de su recorrido que da lugar a una trayectoria curva.

En cambio las fuerzas centrífugas son aquellas que restringen a un cuerpo a rotar alrededor de un punto, se alejan del centro de masa del movimiento circular uniforme. Para obtener las fuerza de Coriolis y fuerzas centrífugas, se utilizarán los símbolos de Christoffel:

$$\mathbf{C}_{kij} := \frac{1}{2} \left(\frac{\partial h_{ij}}{\partial q_k} + \frac{\partial h_{ik}}{\partial q_j} + \frac{\partial h_{kj}}{\partial q_i} \right) \tag{2.23}$$

La matriz $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ puede ser fácilmente calculada a partir de la matriz de inercia. La matriz de Coriolis está dada por:

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{2.24}$$

donde cada elemento de $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ es:

$c_{11} = -s_2\dot{q}_2(b_3c_2 + b_2c_3) - b_1s_2c_3\dot{q}_3 - b_2s_3c_2\dot{q}_3$
$c_{12} = b_3s_2c_2\dot{q}_1 - b_2s_2c_3\dot{q}_1$
$c_{13} = -b_1s_3c_3\dot{q}_1 - b_2c_2s_3\dot{q}_1$
$c_{21} = b_3s_2c_2\dot{q}_1 + b_2s_2c_3\dot{q}_1$
$c_{22} = b_2(c_2s_3 - s_2c_3)(\dot{q}_2 - \dot{q}_3)$
$c_{23} = -b_2(c_2s_3 - s_2c_3)(\dot{q}_2 + \dot{q}_3)$
$c_{31} = b_1s_3c_3\dot{q}_1 + b_2c_2s_3\dot{q}_1$
$c_{32} = 2b_2(c_2s_3 - s_2c_3)\dot{q}_2$
$c_{33} = 0$

2.3.3. Coeficientes de Fricción

Para obtener un mejor análisis se deben de tomar en cuenta las fuerzas que se oponen al movimiento de las articulaciones. En los sistemas mecánicos la fricción viscosa es un elemento que permite que un objeto se mueva con mayor amortiguamiento. En la representación matricial para el modelo, los coeficientes serán los elementos de la diagonal principal de la matriz **D** .

$$\mathbf{D} = \begin{bmatrix} C_{f1} & 0 & 0 \\ 0 & C_{f2} & 0 \\ 0 & 0 & C_{f3} \end{bmatrix} \tag{2.25}$$

Los coeficientes de fricción se puede obtener de la hoja de datos de los motores, si el fabricante no la especifica, este se puede calcular como:

$\mathbf{Cf} = \frac{Max.Par[Nm]}{Max.Velocidad[rad/seg]}$

2.3.4. Fuerzas gravitacionales del robot

Las fuerzas gravitacionales del robot se obtienen del efecto generado por la energía potencial. Para conocer el vector $\mathbf{g}(\mathbf{q})$, se obtiene primero el vector de constantes gravitacionales y la energía potencial de cada eslabón del robot.

$$g = \begin{bmatrix} 0 & 0 & 9.8 \end{bmatrix} \left[\frac{m}{s^2}\right] \tag{2.26}$$

La Energía Potencial para n eslabones se representa como:

$$\mathbf{M} = \sum_{i=0} m_i g^T r_{ci} \tag{2.27}$$

donde m_i es la masa de cada eslabón, $\mathbf{g}$ es un vector con la constantes gravitacional $(9.81\frac{m}{s^2})$ en su componente en z_0, referido al sistema base y r_{ci} es el vector posición para cada instante de cada centro de masa respecto al sistema base. Del resultado de la energía potencial del robot se obtiene el vector de fuerzas gravitacionales dado por:

$$\mathbf{g(q)} = \begin{bmatrix} \frac{\partial P}{\partial q_1} \\ \frac{\partial P}{\partial q_2} \\ \frac{\partial P}{\partial q_3} \end{bmatrix} = \begin{bmatrix} 0 \\ (m_2 l_{c2} + a_2 m_3) g c_2 \\ m_3 l_{c3} g c_3 \end{bmatrix} \tag{2.28}$$

2.4. Control PID para el Robot

En la actualidad existen varios métodos de control conocidos, sin embargo, en la industria los controladores PD y PID son los que suelen ser ocupados, debido a que tienen un gran desempeño y facilidad de implementación en los distintos procesos industriales.

Para comprobar el análisis de la cinemática directa, cinemática inversa y el modelo dinámico del Robot se utilizo un controlador PID, debido a su facil implementación y no se necesario incorporar parte del modelo dinámico en el controlador.

$$\tau = K_p \Delta q + K_d \frac{d}{dt} \Delta q + K_i \int_0^t \Delta q dt + g(q) \tag{2.29}$$

Las ganancias del controlador PID con compensación de gravedad son K_p, K_i, K_d y $g(q)$ como se muestra en la ecuación (4.1).

- La ganancia K_p, representa un cambio presente en la salida del controlador, permite generar una compensación en la medición.
- La ganancia K_i, da una respuesta proporcional a la integral del error. Esta acción elimina el error en estado estacionario, provocado por la acción proporcional. Su implementación da como resultado un mayor tiempo de establecimiento, una respuesta más lenta y el periodo de oscilación es mayor que en el caso de la acción proporcional.
- La ganancia K_d, da una respuesta proporcional a la derivada del error (velocidad de cambio del error). Añadiendo esta acción de control a las anteriores se disminuye el exceso de sobreoscilaciones.

- La compensación de gravedad $g(q)$, se define en la ecuación (2.28), depende de las posiciones articulares del robot.

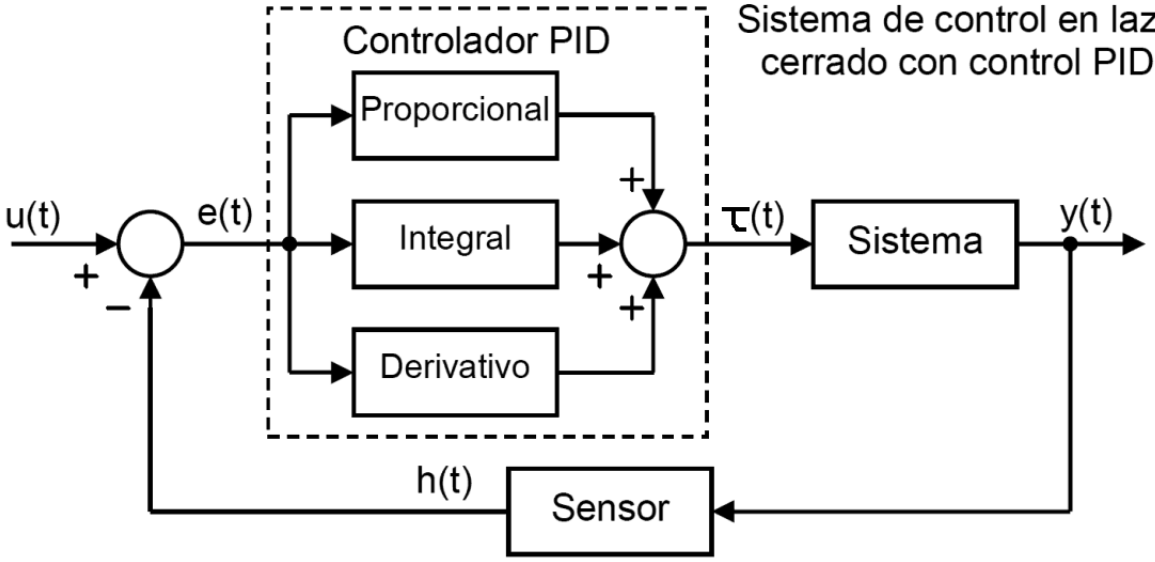


Figura 2.5: Diagrama de bloques de un sistema en lazo cerrado

En la Figura 2.5 se ve diagrama de bloques del sistema en lazo cerrado con un controlador PID, donde el bloque de sistema representa el modelo dinámico del Robot, en la simulación la salida del modelo dinámico representa los datos sensados. Para el caso de este trabajo la entrada $u(t)$ es representada como las posiciones articulares deseadas y $y(t)$ representa la salida del sistema.

2.4.1. Generación de trayectorias

La generación de trayectorias es uno de los aspectos básicos del desarrollo de robots industriales. Permite al robot poder desplazarse de un lugar a otro de manera segura, a partir de un modelo de obstáculos que lo rodean y a un camino, ya calculado. Los estudios en generación de trayectorias son importantes debido a que son la base del desplazamiento de un robot. Para esto, en primer lugar, se debe de calcular una trayectoria. Ésta puede ser calculada por distintos métodos dependiendo del algoritmo utilizado. Una vez calculada la trayectoria, debe ser realizada por el robot real, lo que lleva a un problema de incertidumbre en su ejecución.

Es recomendable simular el comportamiento del robot antes de probarlo en uno real, debido a que se pueden generar fallas en el control. En la simulación se busca obtener un comportamiento ideal del robot, ya que un seguimiento de trayectoria exacto se logra cuando el error entre las trayectorias de referencia y las reales es cero. En un robot real esto no suele pasar y lo que se busca es hacer este error tan cercano a cero como sea posible, tomando como base los datos de la simulación.

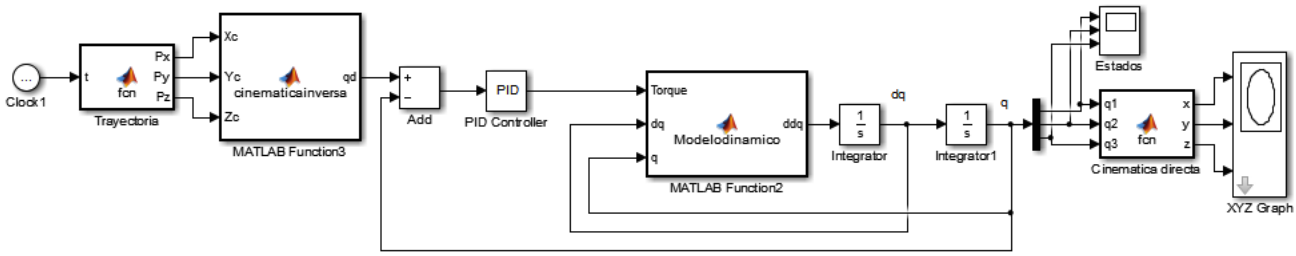


Figura 2.6: Modelo del robot para el seguimiento de trayectoria

En la Figura 2.6 se observa el diagrama de bloques utilizado en SIMULINK que representa el sistema en lazo cerrado del controlador y el modelo dinámico del Robot. Para el robot se tomaron condiciones iniciales de las posiciones articulares como $q = [0; 0.5; -1.2]$ y se considero que cuando el tiempo es cero la velocidad en cada una de las articulaciones es cero.

2.4.1.1. Seguimiento de trayectoria circular

En el diseño de trayectorias se optó por evaluar el robot con una trayectoria circular. Este tipo de trayectoria permite describir un comportamiento sencillo del robot, cuando el robot termina de generar la circunferencia se repite la misma trayectoria en el espacio de trabajo hasta que concluya el tiempo de simulación . Para poder generar una trayectoria circular se programan las ecuaciones paramétricas de una circunferencia, para el caso de este trabajo la trayectoria se diseño para el plano **XY**, y para el eje **Z** se realiza un corte en una altura que el robot pueda alcanzar.

En las ecuaciones paramétricas se debe especificar las proporciones que debe tomar la trayectoria, ya que si se hace muy grande el robot no alcanzará el punto deseado y puede que se generen singularidades. Para este trabajo las ecuaciones paramétricas de la circunferencia son representadas en el plano XY y el eje Z.

Para el plano XY se consideraron las siguientes ecuaciones:

$$x = 19.5 + r\cos(\omega t)[cm] \tag{2.30}$$

$$y = 19.5 + r\cos(\omega t)[cm] \tag{2.31}$$

donde,
r: representa el radio de la circunferencia, para este trabajo se tomo un radio de 6.5 [cm].
ω: es la frecuencia a la que el robot se desplaza en la trayectoria, para que el robot lo realice a una velocidad regular se tomó como $0.5[rad/seg]$.

Para el eje Z se hizo un corte en:

$$z = 12[cm] \tag{2.32}$$

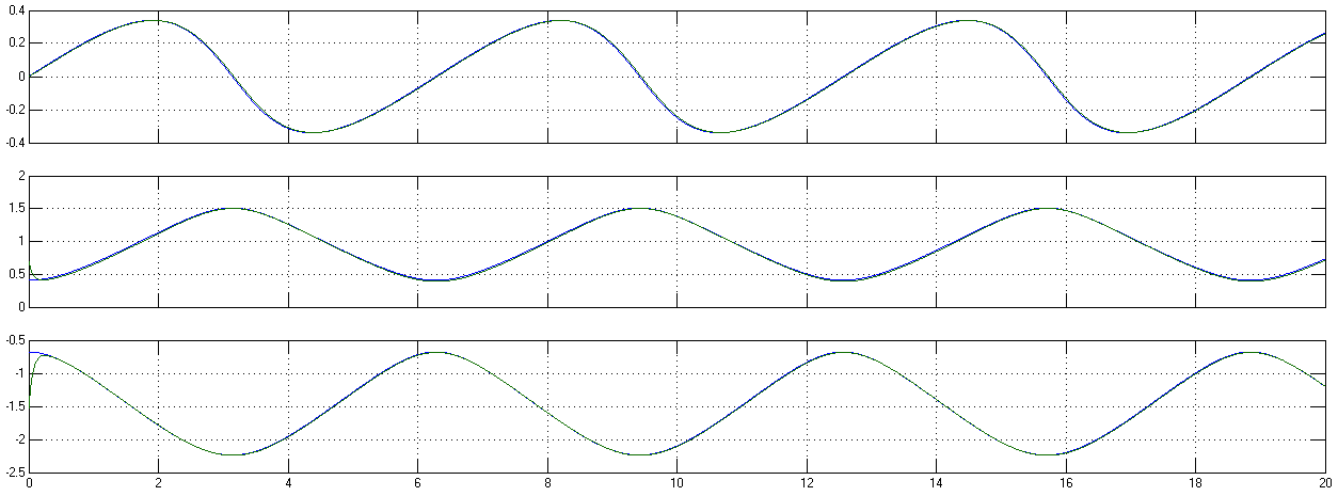


Figura 2.7: Comportamiento de los estados del Robot

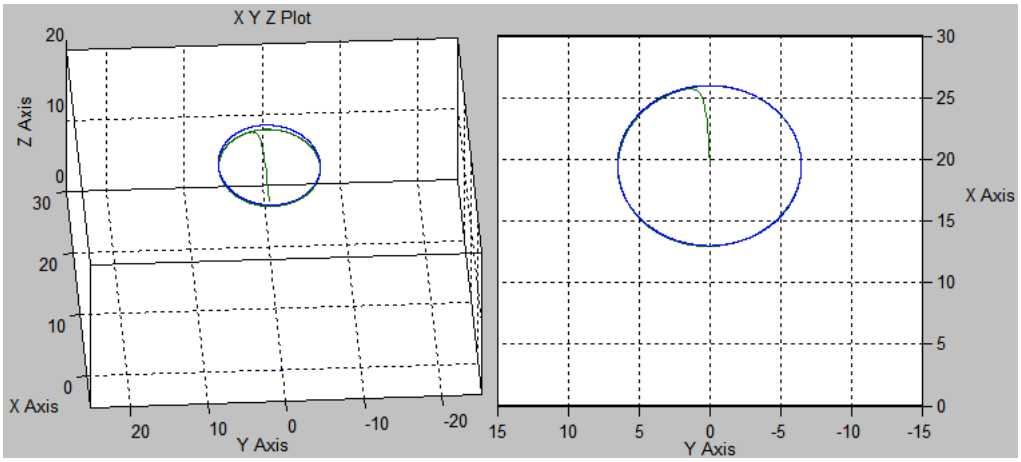


Figura 2.8: Grafica de la trayectoria en el sistema coordenado

De las Figuras 2.7 y 2.8 se pueden observar de color azul la trayectoria que debe de seguir el efector final del robot y la de color verde es la que generó el robot con respecto al análisis realizado de las cinemáticas y dinámicas del robot.

2.4.1.2. Seguimiento de trayectoria mediante parametrización

Existen distintas formas de generar trayectorias, como se observó en el seguimiento de trayectoria circular, es necesario enviarle al robot posiciones deseadas mediante ecuaciones que representan la trayectoria, también se puede hacer uso del efector final durante la ejecución de la trayectoria. Para el caso de este seguimiento de trayectoria se desea trasladar un objeto desde un punto del espacio de trabajo a otro punto.

En el seguimiento circular solo se ingresa una ecuación para cada eje coordenado, pero es posible enviar una trayectoria mediante la parametrización de ecuaciones como se muestra a continuación.

En la trayectoria del Robot se definen los siguientes ecuaciones:

r=20, r representa una distancia máxima a la que el brazo llegará, este es similar al del seguimiento de trayectoria circular.
w=0.5, w representa la velocidad a la que va realizar la trayectoria.
t: representa el tiempo real.
$t_1 = t - 1$, en esta ecuacion se hace una diferencia para reducir el tiempo un segundo.

$t_2 = w * t_1$, este es un multiplo del tiempo menos uno.
Pinza=0, el valor de la pinza varía en donde 0 se encuentra totalmente abierta y cuando esta en 100 se cierra completamente.

En el Robot se tomarón condiciones iniciales igual a x=20, y=0, z=2 y Pinza =50. Las condiciones iniciales permaneceran mientras el tiempo t_2 sea menor a cero. En este caso el Robot ya sujeto la pieza a trasladar.

Cuando el tiempo t_2 sea mayor a cero y menor a 3.1416, empezara a generar un semicurculo en el plano XY.

$x = r * \sin(t_2)$
$y = r * \cos(t_2)$

Para el eje Z si el tiempo es mayor a cero y menor a 1.57 empezará a alzar el brazo hasta llegar al tiempo t2=1.57.

$z = 8.912655 * (t_2) + 2$

Al llegar al tiempo t2 el efector final alcanza una altura de 16[cm] con respecto a Z_0.

Para el eje Z si el tiempo es mayor a 1.57 y menor a 3.1416 empezará a bajar el brazo hasta llegar al tiempo t2=3.1416.

$z = -8.912655 * (t_2) + 30$

Al finalizar el brazo deberá permanecer estable en la posición x=0, y=−20, z=2 y Pinza=0. Finalmente en este punto es cuando el efector final suelta la pieza que se desplazo y termina la trayectoria.

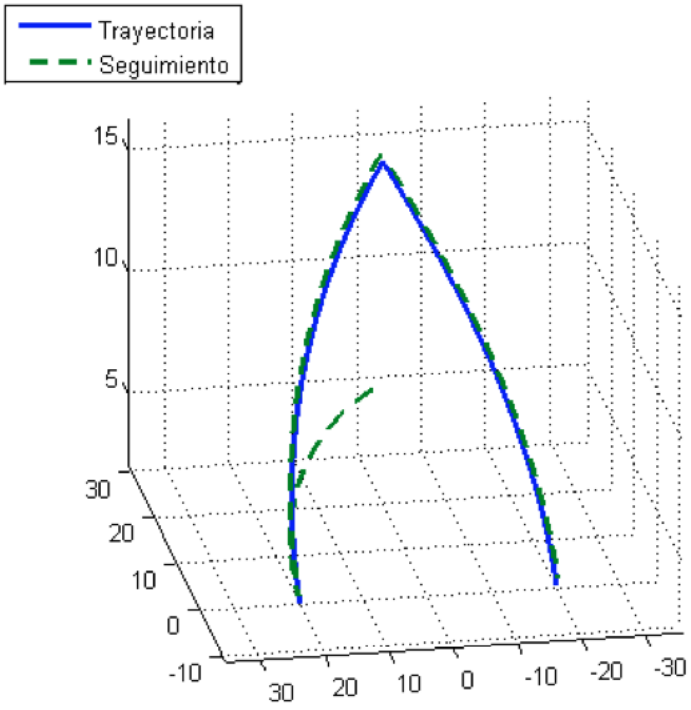


Figura 2.9: Grafica del seguimiento de trayectoria mediante parametrización en el sistema coordenado

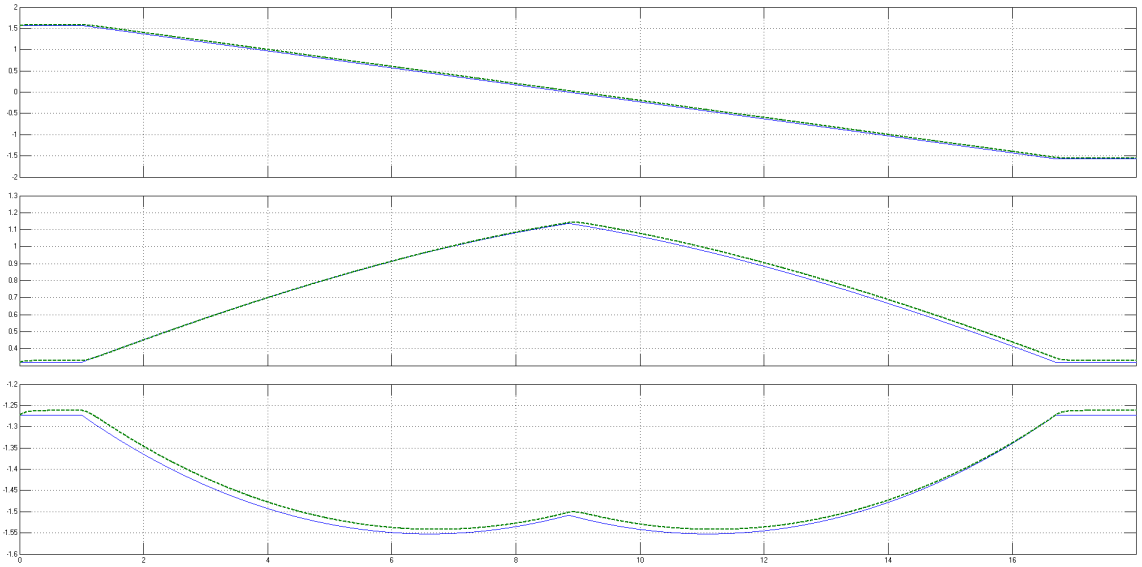


Figura 2.10: Graficas de los estados del seguimiento de trayectoria mediante parametrización

En la Figura 2.9 se observa con línea punteda la trayectoria generada por el Robot en el sistema coordenado, en el se ve que toma las condiciones iniciales del sistema coordenado hasta llegar a las condiciones finales. También se puede notar que la trayectoria en el eje Z va mostrando una pendiente que empieza a incrementar y al llegar a la mitad de la trayectoria decrece , la trayectoria en Z forma un triangulo, es preferible que se generen trayectorias mas suaves. Si se desea realizar una trayectoria mas suave se dificulta un poco mas al obtener las ecuaciones de la trayectoria

En la Figura 2.10 se observa el seguimiento de trayectoria visto desde cada posicion articular, donde la trayectoria punteada es la trayectoria que generada por el robot.

Se puede ver que existen distintas formas de generar trayectorias, algunas se calculan con métodos númericos, debido a que le dan un mejor comportamiento al robot.

Capítulo 3

Diseño del Robot de 3 GDL

Tanto la estructura mecánica como lógica de un robot, son parte fundamental en la construcción del mismo. En este capítulo se detallaran los diseños en hardware y en software propuestos para dar solución al problema planteado, al igual que algunas alternativas observadas en el transcurso de la investigación.

3.1. Componentes del Robot

En el diseño de un robot se debe conocer el material necesario para poder realizar una tarea específica. Para el diseño se debe tomar en cuenta dos partes principales: el diseño físico (hardware) y el diseño lógico (software).

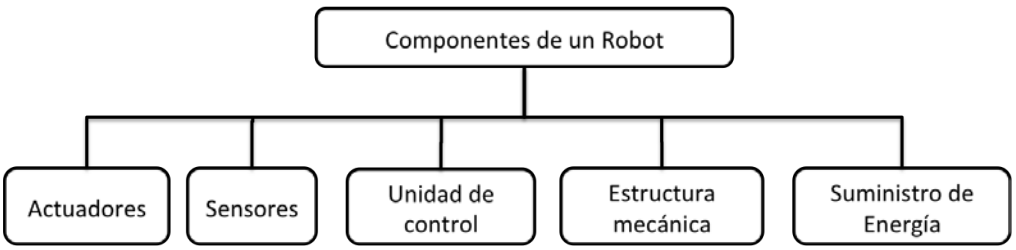


Figura 3.1: Diagrama de componentes

En la Figura 3.1 se observan todos los componentes que conforman el Robot, en donde algunos elementos del robot están destinados a obtener información (sensores y comunicación), otros al procesamiento (control) y otros más generan respuestas al medio (estructura mecánica y actuadores). El diseño físico esta compuesto por los actuadores, sensores y la

estructura mecánica, y el diseño lógico esta compuesto de toda la comunicación del Robot y el control del mismo.

3.1.1. Estructura mecánica

La estructura mecánica es el esqueleto del diseño mecánico, y da soporte a todos los elementos y define los movimientos que el robot podrá realizar. En el desarrollo del proyecto se considera el número de grados de libertad (GDL) que contendrá el brazo robótico, así como la distribución y orientación de cada uno de ellos a lo largo de la estructura mecánica.

El primer aspecto es la cantidad de grados de libertad[4] del robot. Mientras más grande sea la cantidad de grados de libertad, aumenta la cantidad de movimientos que el mismo pueda realizar, generando redundancias, lo que provoca una mayor complejidad en el modelo dinámico del robot.

Como se ha mencionado con anterioridad el robot realizado es un robot antropomórfico como se muestra en la Figura 3.2.

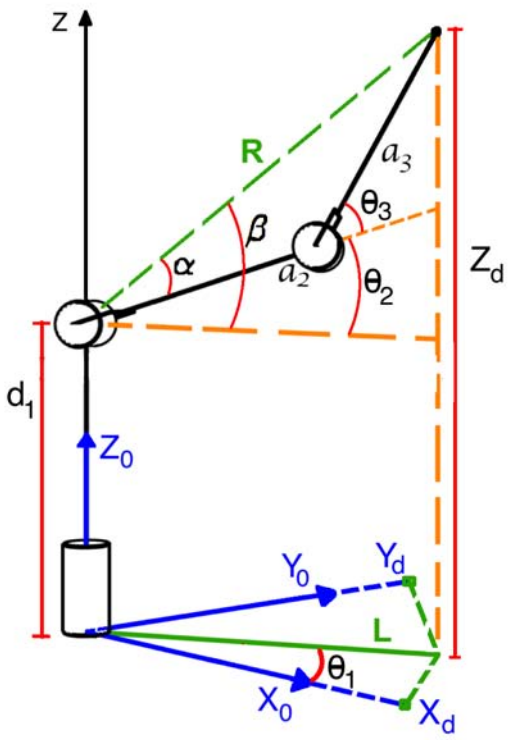


Figura 3.2: Modelo del Robot Antropomorfico

Debido a que el robot desarrollado en este trabajo no contiene muñeca esférica, no permite que al efector final se le pueda asignar alguna orientación deseada.

Mecánicamente, un robot está formado por una cadena cinemática[5] abierta o cerrada dependiendo del tipo de configuración del mismo. La constitución física de la mayor parte de los robots industriales guarda ciertos rasgos antropomórficos, por lo que en ocasiones, para hacer referencia a los distintos elementos que componen el robot, se usan términos como cuerpo, brazo, codo y muñeca.

En este trabajo la representación de los eslabones del robot son el brazo y antebrazo. Se utilizaron articulaciones de tipo rotacional, ya que son las utilizadas en el arreglo cinemático del robot antropomórfico.

3.1.2. Actuadores y Sensores

Los actuadores son los elementos encargados de promocionar el movimiento a las articulaciones del robot. En la actualidad existen una variedad de actuadores que se caracterizan por la forma en la que reciben la energía para realizar el movimiento, los actuadores pueden ser del tipo eléctrico, hidráulico o neumático.

Para el caso del robot de esta tesis se utilizan actuadores electromecánicos, el cual son representados mediante el uso de motores eléctricos. Estos se emplean para robots de pequeña y mediana escala, debido a que no requieren de tanta velocidad ni potencia. Los actuadores electromecánicos utilizan la energía eléctrica para que el robot ejecute sus movimientos. Existen de distintos tipos de motores:

- ***Motores eléctricos de corriente continua.*** Estos se utilizan para proporcionar movimientos giratorios en los que no se requiere mucha precisión.
- ***Motores paso a paso.*** Estos permiten controlar de forma precisa el ángulo de giro del motor, haciendo que el motor se coloque en un posición determinada. Para el control de estos motores se requiere un circuito electrónico de control.
- ***Servomotor de RC(Radio-Control).*** Es un motor de corriente continua que tiene la capacidad de ser controlado a alguna posición deseada. Es capaz de ubicarse en cualquier posición dentro de un rango de operación (generalmente de 180º) y mantenerse estable en dicha posición. Los servomotores son muy utilizados en robots de pequeña escala, por la característica que integran tanto el motor como la electrónica de control y la mecánica de reducción en un solo dispositivo.

Por otro lado los sensores, son los encargados de recoger la información del entorno y enviarla a la unidad de control para su procesamiento. Los sensores se pueden clasificar en dos tipos dependiendo de la función que realicen.

- ***Sensores externos.*** Toman datos del entorno(como sensor de voltaje que mide la fuente de alimentación, para ver si no supera el rango de operación de los motores y sensor de temperatura, para evitar el sobrecalentamiento).
- ***Sensores internos.*** Controlan el propio funcionamiento del robot(como sensor de posición y sensor de velocidad).

Los actuadores electromecánicos se utilizan principalmente en robots con articulaciones rotacionales. En los robots es muy común el uso de los servomotores como es el caso de este trabajo, debido a que tienen una facilidad de integración.

3.1.2.1. Servomotor AX-12

Para las articulaciones del robot se utilizaron los servomotores Dynamixel AX-12A de ROBOTIS, cuyas principales características se pueden observar en la hoja de datos del servomotor[6]. El servomotor Dynamixel AX-12A puede ser llevado a posiciones angulares específicas al enviar una señal codificada. Mientras el servomotor reciba la misma señal codificada mantendrá la posición angular del piñón hasta que la señal codificada sea distinta, la posición angular del piñón cambiará.

Los servomotores Dynamixel pueden llegar a ser mejores que algunos servomotores de RC. Los servomotores Dynamixel tienen internamente un microcontrolador, el cual adquiere la información necesaria de los sensores y permite que el usuario mediante la comunicación serial pueda configurar algunos parámetros como el par, la posición, etc. En cambio los servomotores de RC internamente tienen una serie de circuitos integrados que le permiten adquirir los datos y envía o recibe datos mediante una señal PWM.

Figura 3.3: Servomotor AX-12

El servomotor Dynamixel AX-12A está compuesto de una serie de sensores, una memoria EEPROM[7] y una memoria RAM[8] interconectados con el microcontrolador. Está integrado de un potenciómetro encargado de medir la posición del servomotor con respecto al voltaje, entre otros se encuentra un sensor de temperatura que permite que el servomotor no se queme por sobrecalentamiento, un sensor de voltaje que se encarga de medir la tensión del mismo para que no haya un sobrepaso. Si el sensor de voltaje y el sensor de temperatura tienen un cambio que pueda afectar al servomotor, el microcontrolador bloquea el funcionamiento del motor hasta que se corrija el error.

ROBOTIS en su diseño de los servomotores ocupan un microcontrolador[9] Atmega8 como se observa en la Figura 3.4, en el es fundamental el uso de una memoria EEPROM y una memoria RAM. Las memorias EEPROM y RAM se encargan de enviar y almacenar los datos del motor para ser transferidos al microcontrolador después de que la alimentación sea conectada o desconectada respectivamente.

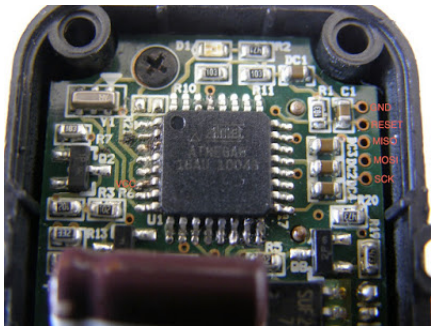


Figura 3.4: Microcontrolador del servomotor AX-12

El fabricante maneja una gama de servomotores que tienen algunas diferencias entre sí, como es el par que puede soportar uno con respecto al otro, el tamaño, peso, precio, tipo de conector, etc. El servomotor Dynamixel AX-12A es uno de los de gama más baja por lo que soporta 1.5 [Nm] de par con una alimentación de voltaje que se recomienda de 11 V.

Cada servomotor tiene 2 conectores, donde cada conector tiene 3 canales como se muestra en la Figura 3.5. El primer canal representa la tierra o GND, el segundo canal es la alimentación de voltaje de aproximadamente 11V, el tercer canal representa la entrada o salida de datos; el otro conector se utiliza cuando se desea utilizar más de un solo servomotor como es el caso de este trabajo. El tipo de conector depende del servomotor que se utilizará, debido a que los que son de gama mayor suelen tener un conector con 4 puertos, estos conectores son diseñados por el propio fabricante.

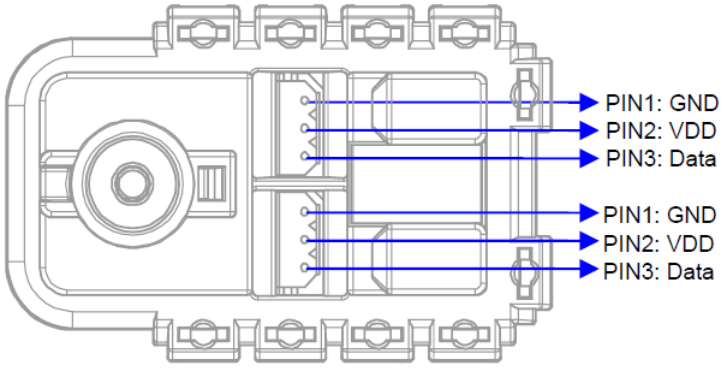


Figura 3.5: Diagrama de Conexión del servomotor Dynamixel AX-12

El servomotor AX-12 tiene la facilidad de modificar algunos parámetros del mismo, por lo que se utiliza un programa en específico conocido como RoboPlus[10]. Para la configuración del servomotor es necesario tener un dispositivo de ROBOTIS conocido como USB2Dynamixel, ya que tiene compatibilidad con el software RoboPlus.
La USB2Dynamixel es un dispositivo que se utiliza para operar los servomotores Dynamixel desde un ordenador, mediante la comunicación serial. Este se conecta al puerto USB del ordenador y a los conectores 3P y 4P que estan asignados a los servomotores. El uso de la USB2Dynamixel es restringido, esto se debe a que tiene sus propias librerías que son compatibles con MATLAB, LabView, Visual Studio C++ y Visual Studio C#.

Para poder realizar un buen diseño del robot se debe de saber las ventajas y desventajas que tiene el uso de los servomotores AX-12A, como se observa en la Tabla 3.1.

Ventajas	Desventajas
El servomotor AX-12 es el mas liviano, pequeño y barato del fabricante	El servomotor AX-12 es el que genera menos par
No es necesario modificar algún mecanismo interno para obtener las lecturas de los sensores.	Se requiere mandar una cadena de bytes para que el servomotor realice una instrucción
Su configuración permite conectar varios servomotores entre sí(como máximo se pueden conectar 254 servomotores)	Para obtener respuesta de un sensor, se envia una instrucción al servomotor, recibiendo una cadena de bytes por lo que los ultimos 2 bytes representan el valor deseado.
Tiene un indicador, cuando se activa bloquea el funcionamiento del servomotor, debido a que ocurrio un error de comunicación o problemas en alguna parte del servomotor.	El servomotor utiliza un potenciometro como sensor de posición, la señal medida del potenciometro es mas ruidosa comparada con el codificador incremental.
Existe la posibilidad de mover varios servomotores al mismo tiempo sin recibir una respuesta de ellos.	El fabricante no da mucha información para realizar codigo abierto sobre los servomotores.
El cableado es muy sencillo: ocupa 2 cables de alimentación y uno de comunicación UART.	
Tiene una resolución de 0.3 grados, puede moverse de una posición 0 hasta 300 grados.	
No necesita una etapa de potencia para alimentar al servomotor	

Tabla 3.1 Ventajas y Desventajas del Servomotor AX-12A

Los servomotores se pueden configurar en modo Articulación o en modo Rueda como se explica en la Tabla 3.2, las configuraciones realizadas en este trabajo se observan en el Apéndice A. Configuracion de Servomotores Dynamixel en RoboPlus.

Modo Articulación	Modo Rueda
Mueve el eje del motor hasta llegar a un ángulo deseado y velocidad deseada	Existe la posibilidad de girar el eje del motor a un cierto par, dándole el sentido de giro
No es necesario obtener datos de la posición	Se puede adquirir datos en tiempo real de la posición del eje del motor o de algún sensor interno del servomotor
Realiza un control en lazo abierto	Se puede realizar control en lazo cerrado
Actuá de 0º a 300º, donde toma como resolución de 0 a 1023	El potenciometro va de 0º a 300º y los 60 grados restantes son punto muerto de la lectura de datos por lo que suele representarlo como 0º
Tiene un par que puede variar de 0 a 1.5 Nm, donde toma como resolución de 0 a 1023(el fabricante lo tomá como velocidad, pero en si es una relación con el par de giro del motor)	El par del motor en sentido anti-horario va de 0 a 1.5 Nm, tomando como resolución de 0 a 1023 cuentas. El par del motor en sentido horario va de 0 a 1.5Nm tomando como resolución de 1024 a 2047, en donde 1024 representa el giro en cero y 2047 el máximo giro

Tabla 3.2 Configuración en el giro del motor

[2] En la Tabla 3.2 se muestra una breve explicación de los modos de configuración del motor para poder realizar control en lazo abierto y cerrado que es lo que se desea realizar en este trabajo de tesis.

Un problema de este servomotor es que no contiene un sensor de velocidad. Normalmente para obtener la velocidad, realiza una derivada númerica, es decir, realiza una tasa de variación de la posición y el tiempo. Por lo tanto, cuando uno desea implementar control se facilita el diseñar control lineal y se dificulta el generar control no lineal, ya que se desea un valor preciso de la velocidad. Si se desea obtener un mejor resultado, se puede diseñar un observador de velocidad para cada articulación, lo cual no esta incluido en este trabajo.

3.1.2.2. Servomotor HITEC HS-422

El servomotor HITEC HS-422 es un servomotor de RC controlado mediante una señal PWM, con un intervalo de movimiento de 0º hasta 180º. Este tipo de servomotor se utiliza en el efector final del Robot, requiere una fuente de alimentación de 5 [V].

[2]Nota: En el diseño del Robot se obtuvo que la Resolución para el giro del motor en modo Rueda o modo DC corresponde al Par del motor, la resolución es de 0 a 1023 en sentido anti-horario y 1024 a 2047 en sentido horario y el par de 0 a 1.5 Nm.

Los robots manipuladores suelen llevar algún dispositivo que se une a la muñeca del robot para realizar una tarea determinada. Se pueden dividir en dos tipos:

- ***Pinzas.*** Estas son diseñadas para la manipulación, transporte y unión de objetos.
- ***Herramientas.*** Realizan funciónes específicas (soldadores, atornilladores, pistola de pintura, etc.).

Para fines de este trabajo se decidió usar una pinza Little Grip Kit de marca HITEC (ver Figura 3.6), la cual permite tomar objetos, normalmente una pieza de trabajo. La pinza es controlada con un servomotor HS-422 de la marca HITEC para poder agarrar un objeto de hasta 3cm de ancho. En este caso no es necesario adquirir datos, debido a que la pinza solo se utilizará para sujeción.

Figura 3.6: Pinza Little Grip Kit

3.1.3. Unidad de control

La unidad de control es el encargado de analizar la información que les mandan los sensores, tomar decisiones y dar ordenes para que las realicen los actuadores.

La unidad de control se puede representar de dos formas:

- ***Mediante un circuito electrónico que puede ser programable.*** Este sistema de control permite construir pequeños robots sin necesidad de cables de conexión con un ordenador.
- ***Mediante ordenador.*** Este es mas utilizado en máquinas que no realizan desplazamientos, ya que la conexión por cable con el ordenador dificultaría su movilidad.

Para el desarrollo de la unidad de control se implementó con un microcontrolador Atmega2560 de Atmel, el cual tiene compatibilidad con el software Arduino[11]. El uso de este microcontrolador provee un compromiso entre flexibilidad y bajo costo, en este se programa gran parte del funcionamiento del hardware, mediante el lenguaje de programación en C.

El microcontrolador Atmega2560 por sí solo no se puede comunicar con los motores, por lo que se necesita adicionar un circuito integrado 74LS241, este se encarga de enviar y recibir la información necesaria por un mismo canal. El tipo protocolo que maneja es Half duplex asíncrono de comunicación serie, y el tipo de cableado que ocupa es mediante el cableado Daisy Chain.

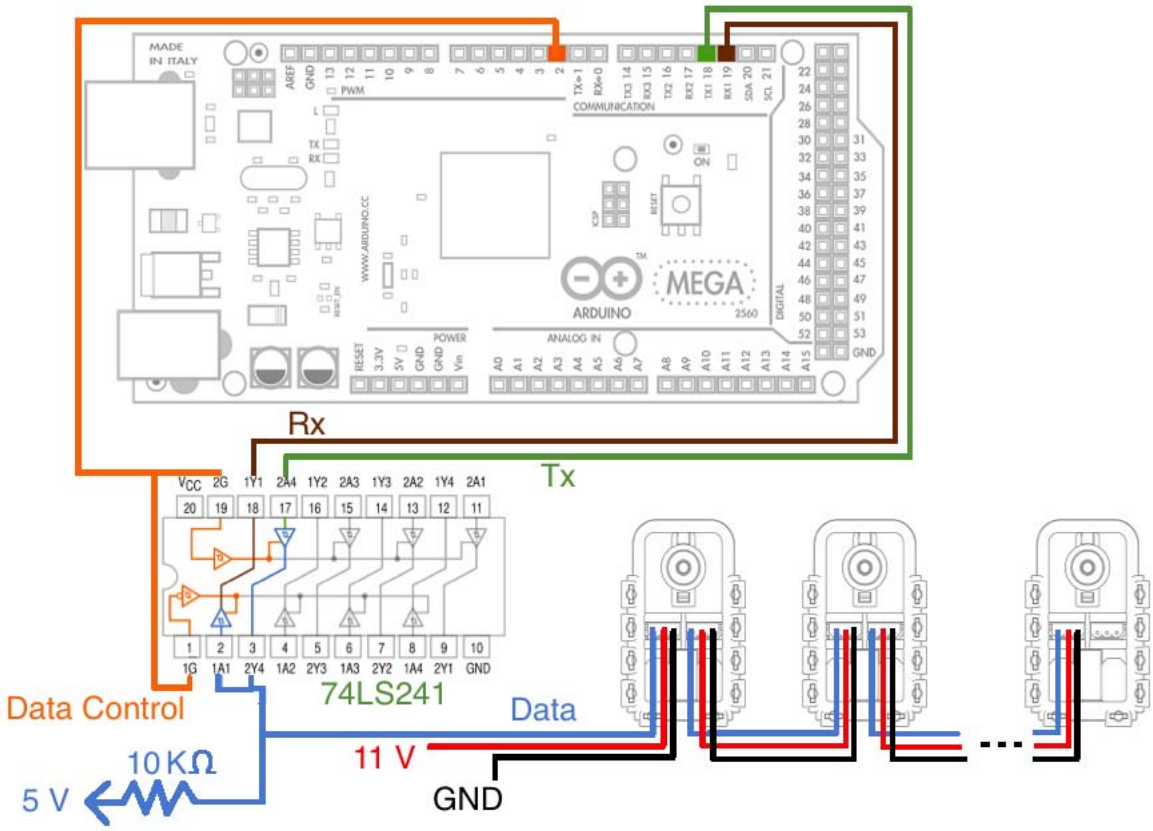


Figura 3.7: Microcontrolador con circuito integrado 74LS241

No es necesario usar más de un circuito integrado 74LS241, ya que los motores tienen la disponibilidad de conectarse entre sí, mediante el tipo de cableado Daisy Chain como se

muestra en la Figura 3.7.

Los servomotores para recibir información del microcontrolador, es necesario enviarle una cadena de datos específica para el tipo de instrucción que se desea realizar, en este trabajo solo se usaron instrucciones para la lectura de posición y escritura de par. Para poder realizar algun tipo de instrucción, se debe verificar la tabla de control del Servomotor AX-12 localizada en los manuales del Servomotor[12]. Las intrucciones que realiza el microcontrolador se pueden consultar en el "Apéndice B. Codigo de Instrucciones del microcontrolador" y si se desea mas información de como se generan las instrucciones se puede observar en "Apéndice D. Ejemplos de Instrucciones del Servomotor Dynamixel AX 12 y AX 18".

3.1.4. Suministro de energía

Los robots son máquinas electromecánicas que necesitan de un suministro de energía eléctrica que alimenta la unidad de control y los motores eléctricos.

En este trabajo los componentes principales que requieren una fuente de alimentación son los motores eléctricos que representan cada articulación del robot, la unidad de control y los circuitos integrados que contenga el robot. Para la selección de la fuente de alimentación se debe verificar cuanto consume cada componente, tomando en cuenta que cada motor puede consumir como máximo 0.9 [A], la unidad de control y los circuitos integrados consumen aproximadamente 0.3 [A]. Para fines de este trabajo se utiliza un adaptador de voltaje de 12 [V] a 3[A], mostrado en la Figura 3.8.

Figura 3.8: Adaptador de voltaje 12[V] a 3[A]

3.2. Diseño Electrónico

Para el diseño electrónico del robot se utilizo una herramienta de diseño de circuitos impresos (Printed Circuit Board, PCB), esto facilita la creación de placas electrónicas para poder tener una mejor comunicación y fuente de alimentación entre los dispositivos. En este trabajo se utilizo Proteus v.8 ISIS/ARES, debido a que es fácil de desarrollar circuitos electricos.

Los componentes electrónicos que ocupan los circuitos electricos son los siguientes:

1 74LS241
2 resistencia 10Kohms
1 LM7805
4 Header para arduino 6 pines
1 Header para arduino 4 pines
2 Molex de 2 vias con los componentes macho, hembra y conector
2 Molex de 3 vias con los componentes macho, hembra y conector
1 LM350
3 capacitores cerámico 100nF
2 capacitores electrolíticos 10uF
2 diodos 1N4007
1 resistencia 2.7 KOhms
1 resistencia 150 Ohms
2 Disipadores de calor para reguladores de voltaje
1 metro de cable negro calibre 22
1 metro de cable rojo calibre 22
1 metro de cable amarillo calibre 22
1 conector dupont 3 vias
1 metro de soldadura
1 cautín
1 placa fenolica
1 cloruro ferrico
1 tira de conectores para dupont hembra para soldar
1 tira de conectores para dupont macho para soldar
1 Jack hembra para Arduino
3 Bornes de 2 vías
1 Switch de boton para prender o apagar el robot

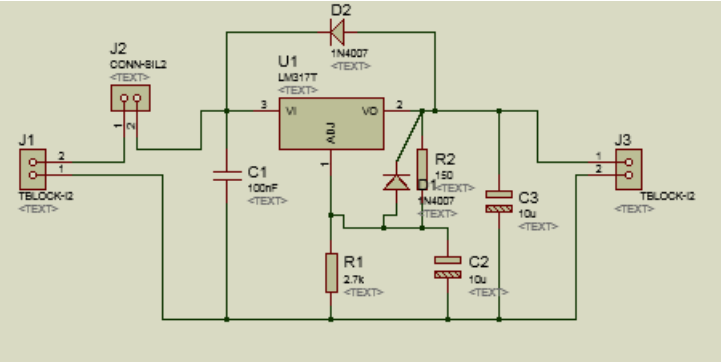


Figura 3.9: Circuito Regulador de Voltaje a 11[V]

En la Figura 3.9 se muestra este circuito ya que el fabricante recomienda usar los motores a 11[V], por lo que se usa una fuente de alimentación de 12[V], esto permite regular el voltaje, los motores se pueden utilizar desde 9[V] hasta 14[V].

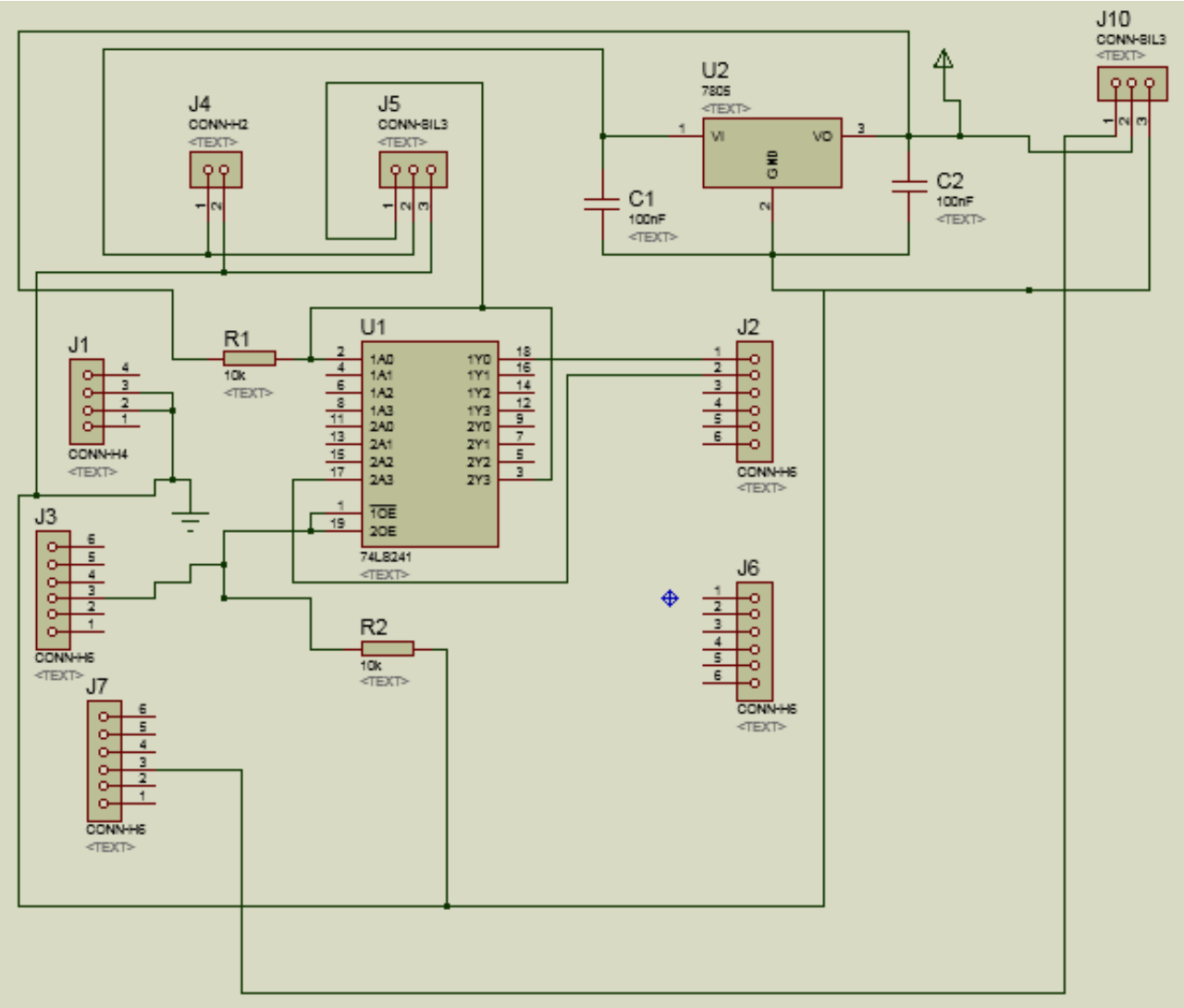


Figura 3.10: Circuito de comunicación

En la Figura 3.10 se muestra un circuito que sirve para sobre ponerse en el arduino mega como un Shield, por lo que hay componentes sueltos en el diagrama. Este circuito permite tener la transmisión y recepción de datos por un mismo canal sin tener colisiones de datos y fallas en la comunicación.

Para poder crear una comunicación entre los actuadores y una computadora, se diseño un circuito electrónico cuyo diagrama de conexiones se muestra en la Figura 3.7. El circuito electrónico se puede observar en la Figura 3.11, este circuito esta adaptado para ser integrado

en el microcontrolador atmega 2560.

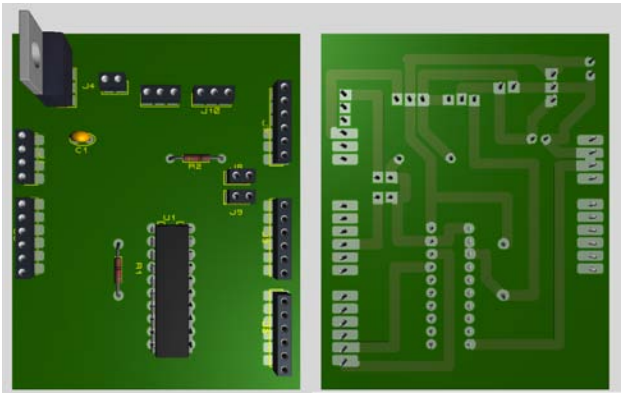


Figura 3.11: Circuito Electrónico del 74LS241

3.3. Diseño Mecánico

Para el diseño mecánico del robot se utilizó una herramienta de diseño asistido por computadora (Computer Aided Design- CAD), ya que facilita el modelado de componentes con las medidas correspondientes, permitiendo un ahorro en recursos, tanto económicos como en tiempo. En la actualidad existe una gran variedad de herramientas CAD, las cuales se pueden elegir dependiendo de las necesidades y los recursos disponibles. En este caso se utilizó AutoCAD 2015 debido a que la versión de estudiante esta disponible para varios sistemas operativos, lo que lo convierte en una herramienta accesible.

Las partes del robot se consideraron en base al tamaño del servomotor Dynamixel AX-12, tratando de copiar la morfología de un brazo humano. Por lo que se definieron algunas distancias para cada parte del mismo.

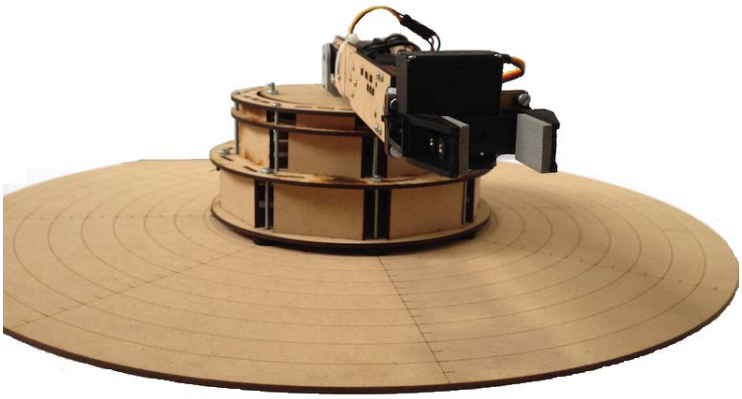
El diseño del robot se dividió en varios componentes principales: cuerpo del robot, la cintura, el hombro, el brazo, el codo, el antebrazo y la pinza. La cadena cinemática de un robot esta compuesta de los eslabones y articulaciones que hay en el mismo. En este trabajo los eslabones del robot son representados como el brazo y el antebrazo, y las articulaciones representan cada uno de los servomotores del robot que son conocidos como la cintura, el hombro y el codo.

El diseño del robot requirió realizar diversas modificaciones durante el desarrollo de este

trabajo de tesis, debido principalmente a que el peso total del robot y las fuerzas que deben aplicar los motores requerían de un diseño más resistente.

Durante el proceso de construcción del robot se eligió el material MDF como material para prototipos, por ser ligero y de bajo costo. Para el robot principal está planeado usar acrílico, ya que es un material ligero, resistente y un buen aislante eléctrico.

A



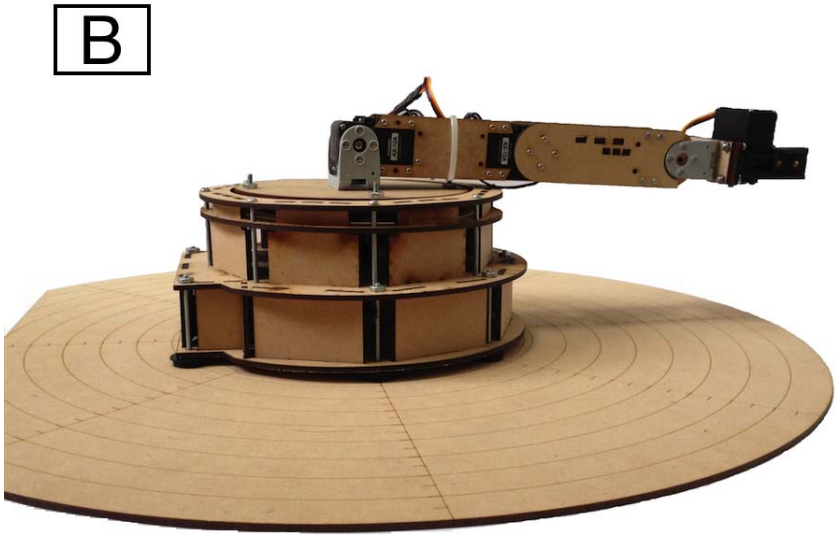


Figura 3.12: Prototipo del Robot con vista A y B

En la Figura 3.12 se muestra la vista del Robot, en A se observa el robot visto desde enfrente y en B se observa una vista lateral del Robot ubicada en la posición cero.

En el análisis de fuerzas que hay en cada articulación se observó que el hombro del robot debe ejercer más par que las otras articulaciones. Para el cálculo del par se tomó la decisión de realizar un brazo robótico que tuviera una extensión máxima de 30[cm], apartir de la segunda articulación hasta terminar el efector final o pinza como se ve en la Figura 3.12 en el caso B.

Para realizar el cálculo del par se debe saber lo siguiente. Cual es el par máximo que puede soportar el servomotor. El par se cálcula como M=F*d.

donde,

M: es el momento de torsión o par.
F: La fuerza que debe soportar o en este caso es el peso del brazo.
d: la distancia de todo el brazo.

Realizando los cálculos necesarios del Robot, sabiendo que el servomotor puede soportar

como máximo 1.5 [N*m]. El brazo contruido en material MDF tiene una masa de 257[g] por lo que su peso es de 2.521[N], el brazo tiene una distancia total de 0.3 [m]. El par resultante es:

$$M = (2.521[N])(0.3[m]) = 0.7564[Nm] \tag{3.1}$$

En la ecuación (3.1) se observa el par máximo que ejerce el brazo cuando se encuentra en posición horizontal. Esto equivale a casi la mitad del par que soporta el motor. Esto permite dar a conocer que la pinza del robot puede sostener un objeto con masa máxima de 200[g]. Si el servomotor se le aumenta una carga de 200[g] ejercería un par aproximado a 1.3744 [N*m].

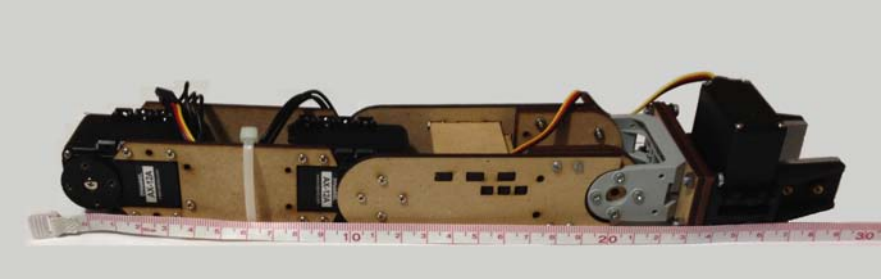


Figura 3.13: Extension del brazo

En la Figura 3.13 se observa la extensión completa del brazo, la cual como se calculo anteriormente es de 30[cm] de longitud.

3.3.1. Diseño del Cuerpo del Robot

En el diseño del cuerpo del robot se tomaron en cuenta todos los dispositivos que van a controlar a los servomotores. La cintura tiene integrada un servomotor, este permitirá que el brazo se desplace en alguna posición ubicada en el plano XY. Este servomotor es de suma importancia, este soporta la mayor parte del peso del brazo robótico. En esta articulación el problema que se genera es que el peso cae sobre él, provocando que cuando hay un cambio de posición hay mucha fricción para poder girar el servomotor. Para poder minimizar la fricción se optó por utilizar una Base cuadrada con rodamientos de 10cmx10cm, el cual permite decrementar la fricción.

Figura 3.14: Base cuadrada con rodamientos

La parte de electrónica del robot ocupa la mayor parte del cuerpo del robot, se encuentra en la parte interior del mismo, mientras que la fuente de alimentación es conecta externamente al cuerpo del robot, al igual que el cable USB que se comunicará con la computadora.

Para la manipulación del brazo robótico se diseñó un espacio de trabajo, esto permite tener una mejor ubicación del efector final.

Figura 3.15: Espacio de trabajo

La estructura del cuerpo del robot esta compuesta de una serie de piezas diseñadas en AutoCAD. Estas piezas se diseñaron para que alcanzára el espacio necesario para todos los componentes, y con una forma sencilla de construcción. El cuerpo del robot esta sujeto con 23 tornillos, 7 se utilizan para aderir el brazo con una superficie rígida, 8 son utilizados para sujetar la base cuadrada con rodamientos en el servomotor Dynamixel AX-12.

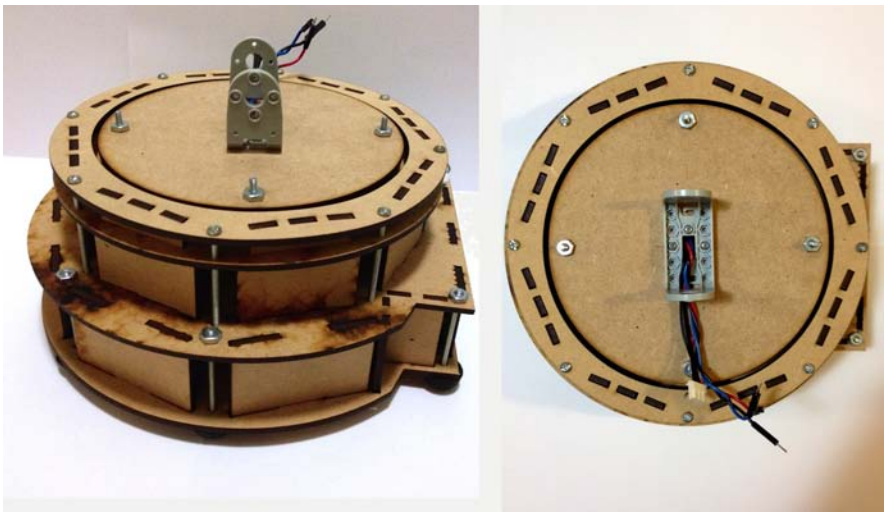


Figura 3.16: Cuerpo del Robot

3.3.2. Diseño del brazo

La sección del brazo ocupa una parte de la extensión del robot, este es la unión del hombro con el brazo y éste mismo con el codo.

Para el diseño del brazo se tomó en cuenta la forma de los servomotores AX-12, ya que contiene una serie de orificios para atornillarse con otras piezas diseñadas por el fabricante o por el usuario. Las piezas del brazo estan diseñadas para que embone el hombro y el codo, entre ellos se genera una distancia de 11 [cm].

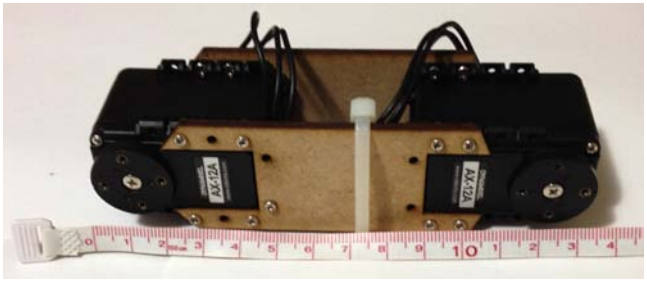


Figura 3.17: Brazo del Robot

3.3.3. Diseño del antebrazo

El antebrazo está sujeto con el codo, por lo que no es necesario el cálculo del momento de inercia. Esto se debe a que en este trabajo se utilizarán el mismo tipo de servomotor para cada articulación y el par que se generá en el codo es menor al del hombro.

Para el diseño del antebrazo se tomó en cuenta la forma del eje del servomotor AX-12 localizado en el codo del robot. La ventaja del eje del servomotor es que es de fácil implementación para piezas de sujeción y a diferencia del diseño del brazo, este va a ser el que se va encargar de sujetar la pinza. El fabricante del servomotor AX-12 tiene incluido algunos aditamentos conocidos como FP04-F2 con tornillos M2 DP4.0 que permiten la sujeción de otras piezas como se muestra en la Figura 3.18.

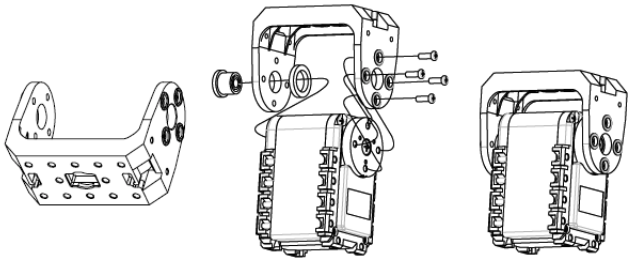


Figura 3.18: FP04-F2

Para el diseño de las piezas se observaron el tamaño específico para cada componente con el servomotor y con los accesorios necesarios para sujetar el motor, teniendo una distancia entre el eje del servomotor y el extremo del FP04-F2 de 17[cm] de largo.

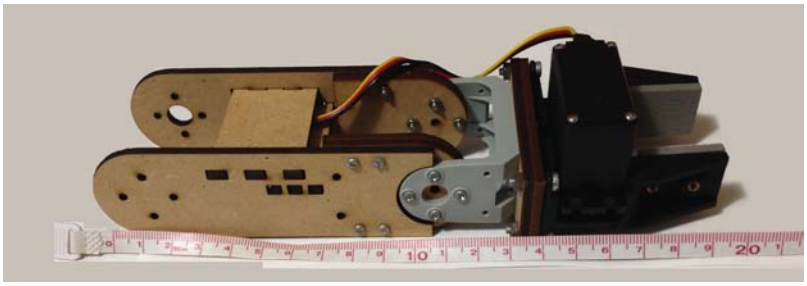


Figura 3.19: Antebrazo del Robot

3.4. Diseño del Software

Durante el proceso de este trabajo se optó por usar MATLAB & SIMULINK, ya que se pueden crear aplicaciones mediante el uso de lenguaje escrito y gráfico. La comunicación entre la computadora y los servomotores se utilizo el microcontrolador Atmega 2560, ya que el uso de la USB2Dynamixel para MATLAB esta restringido por la librería y sólo permite mandar y obtener una respuesta a la vez.

Para poder programar el microcontrolador Atmega 2560 se utilizó el software Arduino, este software se basa en el lenguaje de programación C. Actualmente existe una librería en Arduino desarrollada para poder controlar los servomotores AX-12 y AX-18 de Dynamixel, la desventaja fue que no contenia todas las funciones necesarias para el proyecto. Por lo tanto, a la librería se le agregaron funciones para poder mover todos los motores al mismo tiempo.

El enlace entre los motores y el microcontrolador se realiza mediante la comunicación UART, como se muestra en la Figura 3.7. Para poder desarrollar el código para que el robot pueda realizar las instrucciones se tomó en cuenta el uso de diagramas de flujo, como se observa en la Figura 3.20 y 3.21.

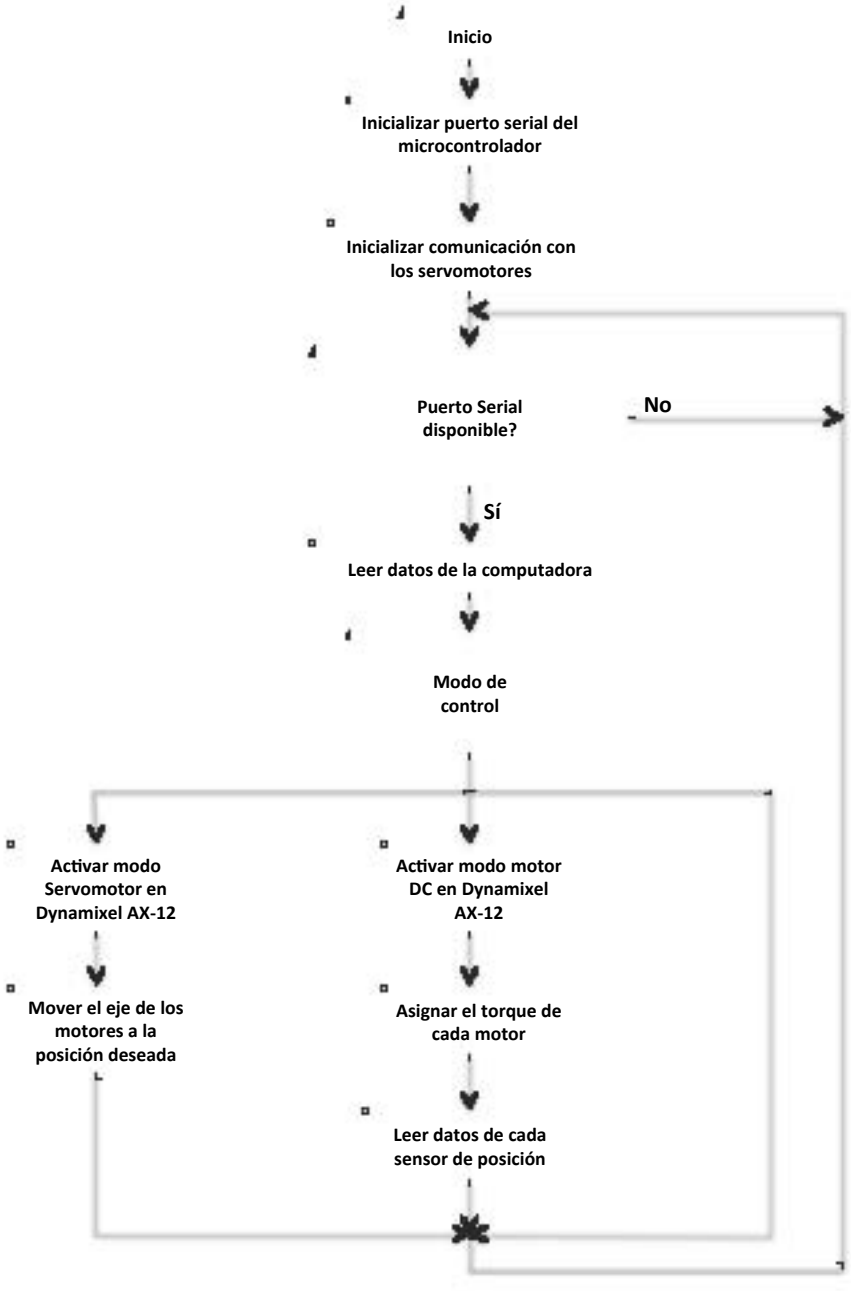


Figura 3.20: Diagrama flujo para el Microcontrolador

En la Figura 3.20 se observa una explicación sencilla de como se diseño el código que fue desarrollado para el microcontrolador para poder ejecutar las distintas tareas que realizará

el robot.

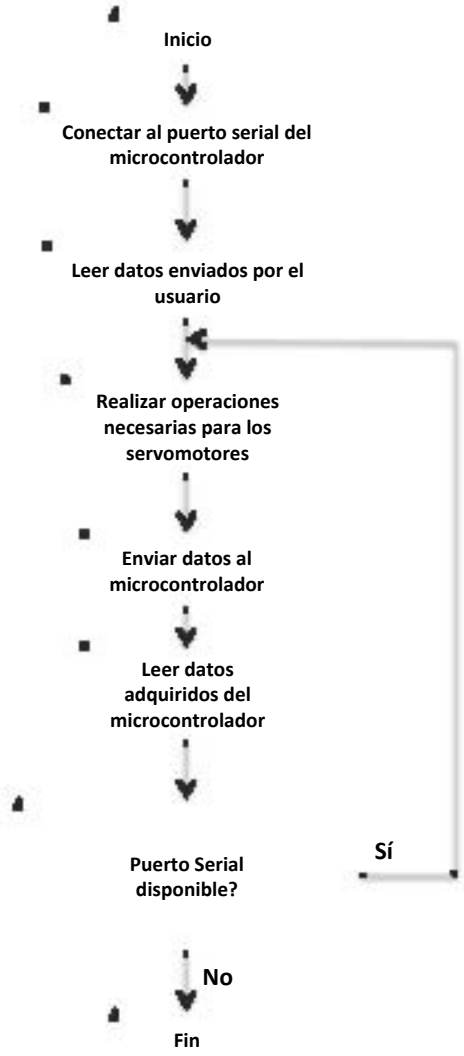


Figura 3.21: Diagrama flujo para MATLAB

En la Figura 3.21 se observa el diagrama de flujo que se tomó en cuenta para el desarrollo del control en MATLAB & SIMULINK.

3.5. Costos de producción del Robot

Los costos de producción son los gastos necesarios para mantener un proyecto, línea de procesamiento o un equipo en funcionamiento, este se encuentra representado por dos elementos básicos, que también son denominados como elementos de producción:

- ***Materia prima.*** Esta es la base con la que se realiza un producto y sin esta no puede realizarse el objetivo de la empresa y es catalogado como el principal costo de producción. Es mediante el manejo de la materia prima que produce un producto completamente diferente al material inicial.
- ***Mano de obra.*** Este siempre es necesaria para la transformación y creación del producto final, el cual se realizará con la materia prima

Para el desarrollo del Robot se obtuvieron los siguientes costos de producción:

Material	Costo aproximado
Material MDF 3mm 1.22mx 2.44m	$125.00 MXN
Circuitos integrados y componentes electronicos	$200.00 MXN
7 Tornillos de 1/8 de pulgada a 4[cm] de largo	$15.00 MXN
8 Tornillos de 1/8 de pulgada a 5[cm] de largo	$15.00 MXN
8 Tornillos de 1/8 de pulgada aprox. 1[cm] de largo	$10.00 MXN
24 Tuercas para tornillos de 1/8 de pulgada	$15.00 MXN
20 Tornillos M2 (2mm) de 1.2[cm] de largo	$70.00 MXN
4 Tornillos M2 (2mm) de 1.6[cm] de largo	$15.00 MXN
4 Tornillos (2.8mm) de 1.2[cm] de largo y tuercas	$15.00 MXN
7 Soporte de goma atornillable	$40.00 MXN
1 Base cuadrada embalerada de 10x10cm	$50.00 MXN
3 Servomotores AX-12 y sus aditamentos	$2,400.00 MXN
1 Servomotores AX-18	$1,920.00 MXN
1 Pinza Little Grip Kit(con 2 servos HS-422)	$ 707.00 MXN
1 Arduino Mega 2560 Generico	$300.00 MXN
1 Adaptador de Voltaje 12[V]	$220.00 MXN
1 USB2Dynamixel	$828.00 MXN
Subtotal	$6,945.00 MXN

Tabla 3.3 Materia prima del robot

Mano de obra.

Material	Costo aproximado
Corte láser	$200.00 MXN
Circuito impreso	$100.00 MXN
Subtotal	$300.00 MXN

Tabla 3.4 Mano de obra del robot

El Subtotal de producción es= $ 7,545.00 MXN y el IVA (16 %)= $ 1207.20 MXN
Total= $8,752.2 MXN

Los precios utilizados para los costos de producción pertenecen a la fecha del 20 de Octubre de 2015.

Capítulo 4

Implementación y resultados

Como se observó en el Capítulo 2 se realizó el análisis teórico del robot por el cual se desarrollaron simulaciones para verificar si el robot diseñado con las características deseadas se comporta de forma correcta en el espacio de trabajo, para evitar colisiones y daños en el equipo. Por lo que se decidio generar distintas pruebas para verificar que el robot se desplace de la manera correcta.

Para la implementación del robot se generaron 3 pruebas:

- Implementación del tiempo de muestreo en el microcontrolador.
- Cinemática directa en el robot.
- Cinemática inversa en el robot.
- Implementación del Control PID .
- Implementación del seguimiento de trayectoria circular.
- Implementación del seguimiento de trayectoria mediante parametrización.

4.1. Implementación del tiempo de muestreo

El tiempo de muestreo, es el tiempo que transcurres entre dos mediciones consecutivas, es fundamental para la adquisición de datos y se suele expresar en frecuencia. Siempre que se desea medir un sistema, se necesita que nuestra frecuencia de muestro sea superior a la frecuencia del sistema.

El tiempo de muestreo se calculo de forma experimental utilizando SIMULINK y el programa del microcontrolador. En el microcontrolador se realizo la siguiente operación:

$$Tiempodemuestreo = Tiempo_{actual} - Tiempo_{anterior} \tag{4.1}$$

Esta prueba soló se realizó para obtener una aproximación del tiempo de muestreo. En el programa del microcontrolador se definen las variables del Tiempo, el tiempo actual se adquiere al final del programa y el tiempo anterior al inicio del mismo, el tiempo de muestreo se calcula despues de haber obtenido el tiempo actual. El microcontrolador recibe la información necesaria envíada desde una computadora y al final regresa los datos obenidos de los sensores junto con el tiempo de muestreo.
El valor experimental obtenido de la ecuación (4.1) es:

$Tiempodemuestreo = 0.7$ [ms]

4.2. Cinemática directa en el robot

Para comprobar el análisis realizado respecto a la cinemática directa del capítulo 2, se desarrolló una interfaz en GUI MATLAB y el microcontrolador para poder desplazar el robot.

Para su comprobación se le asignaron a $\theta_1 = 0$, $\theta_2 = \frac{\pi}{3}$ y $\theta_3 = -\frac{\pi}{2}$, dando como resultante X_d= 20 [cm] Y_d= 0 [cm] Z_d= 12.6 [cm]. En la interfaz desarrollada se asignaron los valores de θ_1 para la cintura, θ_2 para el hombro y θ_3 para el codo, como se muestra en la Figura 4.1.

Figura 4.1: Interfaz para la cinemática directa

El resultado de la interfaz se implementa en el robot real, por lo que de la prueba se obtiene lo mostrado en la Figura 4.2.

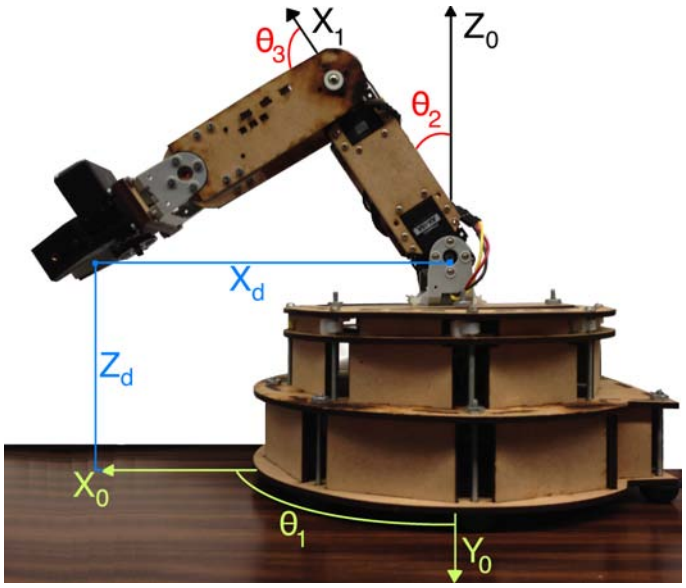


Figura 4.2: Desplazamiento del robot con cinemática directa

De la Figura 4.2 se realizaron las medidas correspondientes a X_d, Y_d y Z_d del robot y se verificó que las distancias obtenidas del efector final con respecto al sistema base eran las mismas, que se asignaron.

4.3. Cinemática inversa en el robot

Para comprobar el análisis realizado respecto a la cinemática inversa del capítulo 2, se desarrolló una interfaz en GUI MATLAB, donde a diferencia de la cinemática directa a este se le asignan las posiciónes deseadas del efector final. Para su comprobación se le asignaron a X_d= 12 [cm] Y_d= 12 [cm] y Z_d= 6.7 [cm], dando como resultante $\theta_1 = \frac{\pi}{4}$, $\theta_2 = \frac{5}{18}\ \pi$ y $\theta_3 = -\frac{17}{30}\ \pi$. En la interfaz desarrollada se asignaron los valores de X_d, Y_d y Z_d, como se muestra en la Figura 4.3.

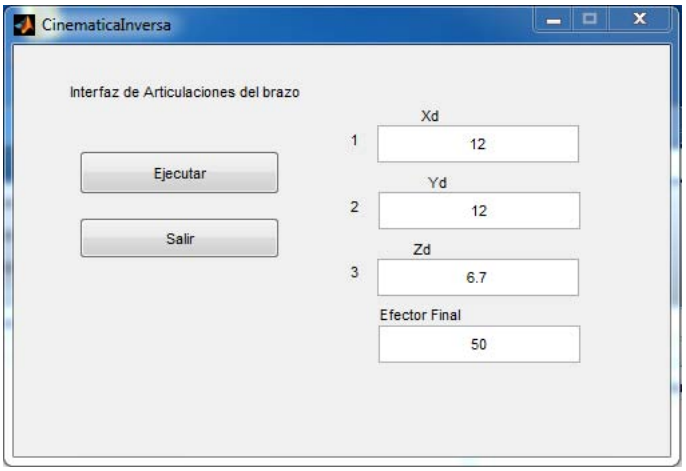


Figura 4.3: Desplazamiento del robot con cinemática inversa

El resultado de la interfaz se implementa en el robot real, por lo que de la prueba se obtiene lo mostrado en la Figura 4.4.

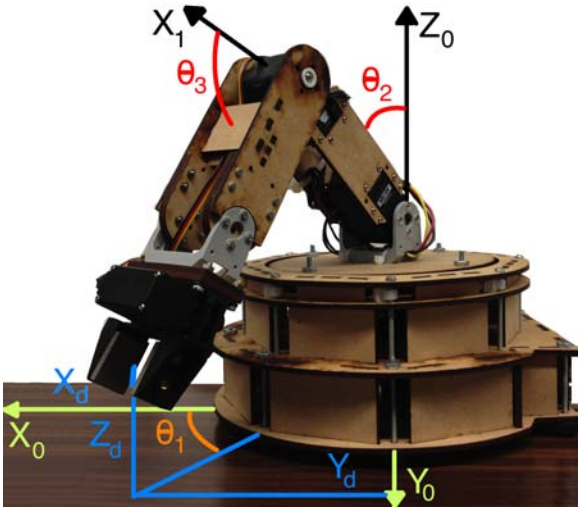


Figura 4.4: Desplazamiento del robot con cinemática directa

De la Figura 4.4 se realizaron las medidas correspondientes a θ_1 , θ_2 y θ_3 del robot y se verificaron los ángulos correspondientes de cada articulación.

4.4. Implementación del Control PID

Como se observa en el Capítulo 2, en la simulación se utilizo un controlador PID con compensación de gravedad en el sistema en lazo cerrado del Robot para observar su comportamiento.

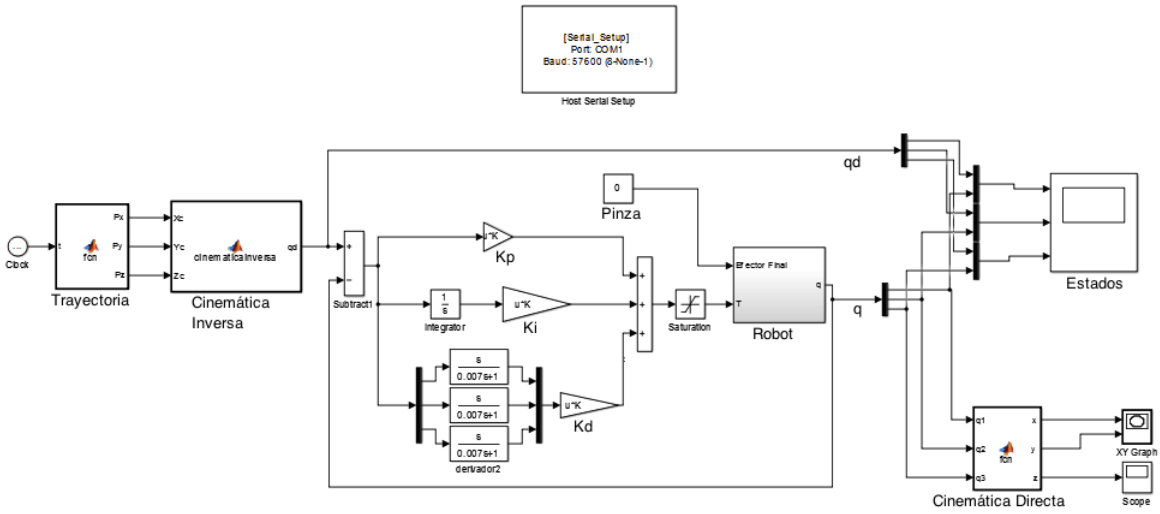


Figura 4.5: Diagrama de bloques para el control PID del Robot

En la Figura 4.5 se observa el diagrama de bloques para el control PID del Robot, este diagrama de bloque es similar al de la simulación, la diferencia es que en este no se toma el modelo dinámico del Robot a menos que se deseé implementar otro tipo de controlador.

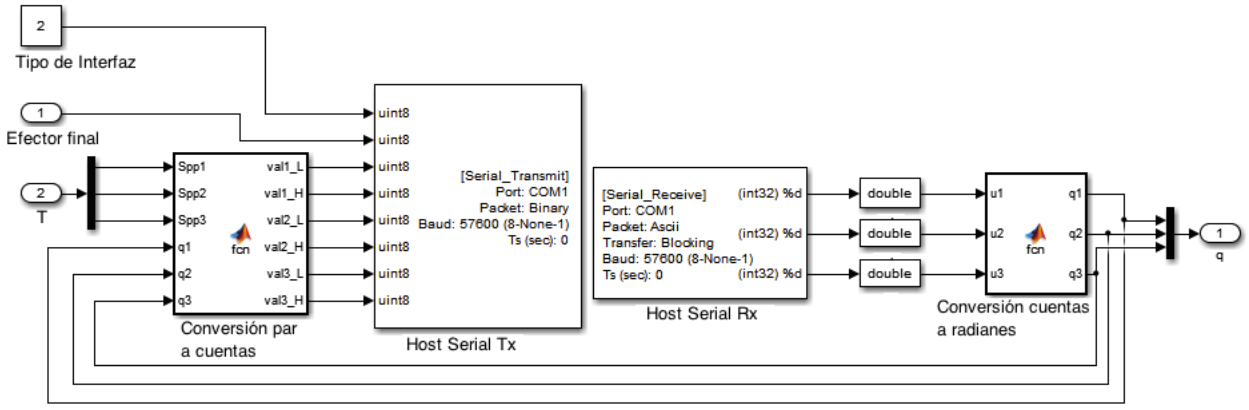


Figura 4.6: Bloque de comunicación del Robot

En la Figura 4.5 hay un bloque cuyo nombre es Robot, lo que tiene internamente este bloque se puede apreciar en la Figura 4.6, este bloque esta compuesto de lo siguiente:

- Conversión de par a cuentas, se encarga de adquirir los datos en Nm y los convierte a la resolución del servomotor. Se sabe que los motores tienen un para máximo de 1.5 [Nm], por lo que para ese par corresponde la cuenta 1023 en sentido anti-horario, y para el sentido horario corresponde 2047. Para cuando el par es 0 [Nm], corresponde la cuenta 0 en sentido anti-horario, y para el sentido horario corresponde 1024.
- Conversión de cuentas a radianes, la lectura de la posición articular va de 0[rad] a 5.236[rad](en grados es de 0^o a 300^o),para cuando esta ubicado en 0[rad] la cuenta es cero y para cuando se ubica en 5.236[rad] la cuenta es 1023 sin importar el sentido de giro.
- bloques de conversion de entero a double, los datos recibidos del microcontrolador se toman como entero, por lo que se adquiriran valores de 0 a 1023 enteros, se convierte a double porque se desea el valor en radianes.
- Bloque de Host Serial Tx, este bloque sirve para enviar todos los datos necesarios al microcontrolador.
- Bloque de Host Serial Rx este bloque sirve para recibir todos los datos de las posiciones articulares del microcontrolador
- Tipo de interfaz, este bloque se habilita en 2 para el caso de Simulink, debido a que la comunicación se realiza en lazo cerrado.
- Efector Final, se habilita con 0 para cuando esta abierta la pinza y 100 para cerrar completamente la pinza, este dato es en enteros.
- T, esta es la entrada del par de cada motor.

El controlador PID obtenido en la simulación se implementó en el Robot construido, pero el resultado no fue igual que el simulado. Para obtener los valores de las ganancias experimentalmente para el controlador PID, existen distintos métodos de sintonización. En el caso de este trabajo se utilizó el método de sintonización de Ziegler - Nichols de oscilaciones contiunas.

4.4.1. Método de sintonización de Ziegler - Nichols de oscilaciones continuas

El método tiene la ventaja de que no es necesario conocer la planta del sistema para poder sintonizarlo. Su sintonización se realiza en lazo cerrado, este procedimiento es válido solo

para plantas estables a lazo abierto. La sintonización de este controlador PID para cada articulación se hace independiente y es por regulación, es decir, en la sintonización a la articulación que se analizará se le asigna una posición articular deseada de 1 [rad].

Para la sintonización del control PID, en el caso de este proyecto primero se sintoniza la tercera articulación, ya que esta articulación no depende del funcionamiento de la segunda y la primera articulación. Después se sintoniza la segunda articulación, para este caso se habilita el control PID de la tercera articulación dandole la posición deseada con la que se sintonizó. Finalmente se sintoniza la primera articulación en este caso se habilita el control PID de la segunda y tercera articulación con la posición articular deseada que se sintonizó, ya que si no se le ejerce el control a las otras articulaciones no es posible sintonizar esta articulación, porque el efector final puede tener una fuerza de fricción con la base del espacio de trabajo.

Para la sintonización de cada articulación se realizaron los siguientes pasos:

1. Sólo se utilizó la acción proporcional, comenzando desde un valor de ganancia pequeño, se incremento la ganancia proporcional hasta que la salida del sistema presentara oscilaciones. La ganancia obtenida es conocida como K_c, representa la ganancia critica a la que puede llegar el sistema sin el comportamiento de la acción integral y derivativa.

2. Las oscilaciones tienen un período continuo que se le conoce como P_{cr}, como se observa en la Figura 4.7.

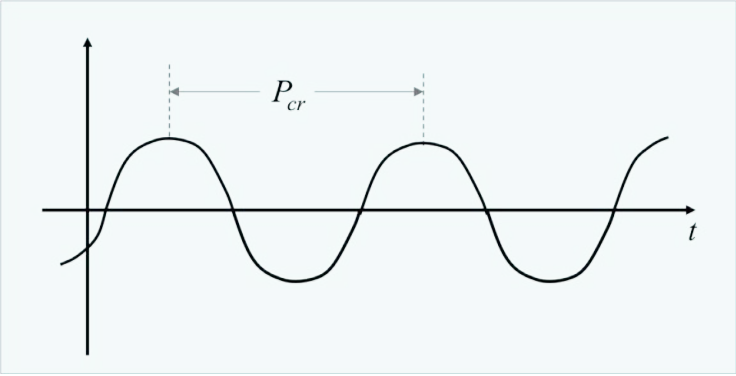


Figura 4.7: Oscilaciones continuas

3. Se calcularon las ganancias del controlador, respecto a la Tabla 4.1.

Controlador	K_p	τ_i	τ_d
PID	0.6 K_c	0.5 P_{cr}	0.125 P_{cr}

Tabla 4.1 Sintonización de PID

En la tabla 4.1 se obtiene la ganancia proporcional, para obtener la ganancia integral y derivativa se calculan de la siguiente forma:

$$K_i = \frac{K_p}{\tau_i} \tag{4.2}$$

$$K_d = K_p \, \tau_d \tag{4.3}$$

La ganancias K_i y K_d dependen de la ganancia proporcional como se observa en la ecuación (4.2) y (4.3). En este trabajo se obtuvieron los siguientes resultados tomando en cuenta las ecuaciones (4.2) , (4.3) y las ecuaciones de la Tabla 4.1.

Para la primera articulación se obtuvo $K_c = 3.4$ y $P_{cr} = 13.2$, en base a estos valores se calcularon las ganancias del control PID que dieron como resultado: $K_p = 2$, $K_i = 0.3$ y $K_d = 1.65$.

Para la segunda articulación se obtuvo $K_c = 8.81$ y $P_{cr} = 5.55$, en base a estos valores se calcularon las ganancias del control PID que dieron como resultado: $K_p = 5.286$, $K_i = 1.9$ y $K_d = 3.66$.

Para la tercera articulación se obtuvo $K_c = 8.3$ y $P_{cr} = 7.8$, en base a estos valores se calcularon las ganancias del control PID que dieron como resultado: $K_p = 4.98$, $K_i = 1.27$ y $K_d = 4.85$.

En base a las ganancias del controlador PID calculadas para cada articulación se probaron en el Robot y se observó su comportamiento. El comportamiento que presento cada articulación del Robot se observaron pequeñas oscilaciones, por lo que se ajustaron las ganancias K_i y K_d, tal que las oscilaciones disminuyeran lo mas posible o desaparecieran.

4.4.2. Método de diferenciación

En la implementación del microcontrolador con MATLAB, es necesario un método de derivación, debido a que no se tiene sensor de velocidad. Los servomotores AX-12 pueden entregar datos de velocidad ángular, pero tienen la desventaja que usan un método muy sencillo conocido como diferenciación númerica. La diferenciación númerica se calcula como:

$$m_s = \frac{f(t+h) - f(t)}{(t+h) - t} \tag{4.4}$$

En la ecuación (4.4) representa un cambio en la posición con respecto al tiempo y h representa un incremento. Para el caso del proyecto, el resultado obtenido es la velocidad promedio de la articulación analizada. Sin embargo, se observó que el cálculo de la velocidad por diferenciación númerica no presenta un buen desempeño. Para la resolución del potenciometro del motor, la velocidad calculada por este metodo es imprecisa para velocidades bajas y altas.

Se optó entonces por estimar la velocidad ángular por medio de un filtro estable paso altas de primer orden con grado relativo cero. Este método de diferenciación aproximada, comúnmente es conocido como derivada sucia. La derivada sucia tiende un mejor desempeño que con el enfoque tradiciondal de diferenciación númerica, por lo que es comunmente utilizada en aplicaciones que requieren regulación de velocidad. La derivada sucia se puede representar como la ecuación (4.6).

$$G(s) = \frac{a\ s}{b\ s + 1} \tag{4.5}$$

En la ecuación (4.5) se puede observar la interpretación de la derivada sucia. Para a = 1, y b= tiempo de muestreo. En el proyecto se utilizo la derivada sucia para poder realizar la acción derivativa del controlador PID. El resultado de la derivada sucia en el robot se observa en la ecuación (4.6).

$$G(s) = \frac{s}{0.007\ s + 1} \tag{4.6}$$

4.5. Implementación del seguimiento de trayectoria circular

En el Capítulo 3 se puede observar en la generación de trayectorias, el análisis teórico del seguimiento de trayectoria circular. El diagrama de bloques del Robot real en simulink cambia

con respecto al de la simulación, debido a que se incluye el tiempo de muestreo y el método de derivación, este se puede observar en la Figura 4.5.

Para el seguimiento de trayectoria, se tomaron en cuenta los valores calculados en el método de sintonización del conrtolador PID. Se ejecuto el programa para que el robot real realizara la trayectoria y en SIMULINK se adquirieron todos los datos mientras se estuviera ejecutando. De los datos obtenidos se graficaron los estados que representan cada una de las posiciones articulares con respecto al tiempo, como se observa en la Figura 4.8.

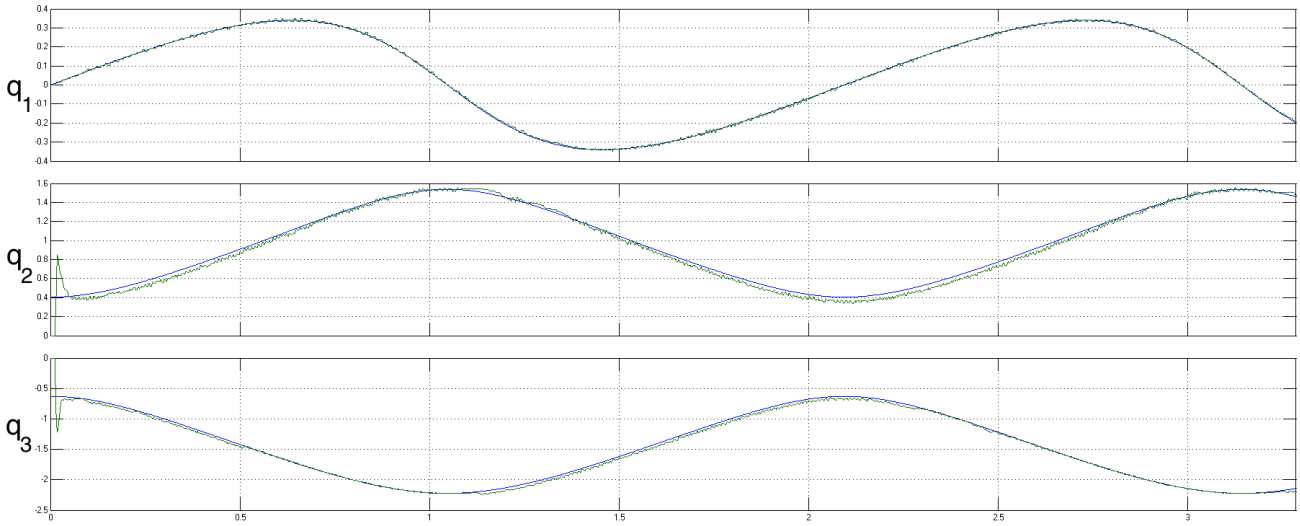


Figura 4.8: Comportamiento de los estados del Robot real sin movilidad en el efector final

En las Figuras 4.8 se puede observar el comportamiento de cada articulación, la articulación q_1 sigue la trayectoria relativamente bien ya que en esta articulación no hay efectos de la gravedad. Para el caso de las articulaciones q_2 y q_3, en estas si hay efecto de la gravedad por lo que se le dio una compensación de gravedad al robot para poder obtener un mejor comportamiento del mismo, esta compensación es la que se muestra en la ecuación (2.28).

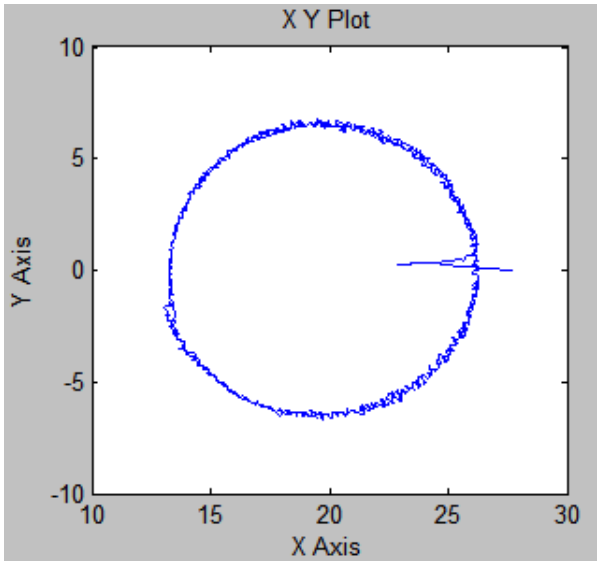


Figura 4.9: Grafica de la trayectoria en el Plano XY

En la Figura 4.9 se observa todo el desplazamiento generado en el Plano XY, por lo que se observan pequeños picos, y esto se debe a las ganancias proporcionadas en el robot y el efecto de la gravedad en las articulaciones q_2 y q_3.

4.6. Implementación del seguimiento de trayectoria mediante parametrización

El análisis teórico para la generación de una trayectoria mediante parametrización se puede observar en el Capítulo 3 en la generación de trayectorias. A diferencia del diagrama de bloques del seguimiento de trayectoria circular este cambia un poco, debido a que se desea que el brazo robótico sujete un objeto y lo desplace al lado contrario.

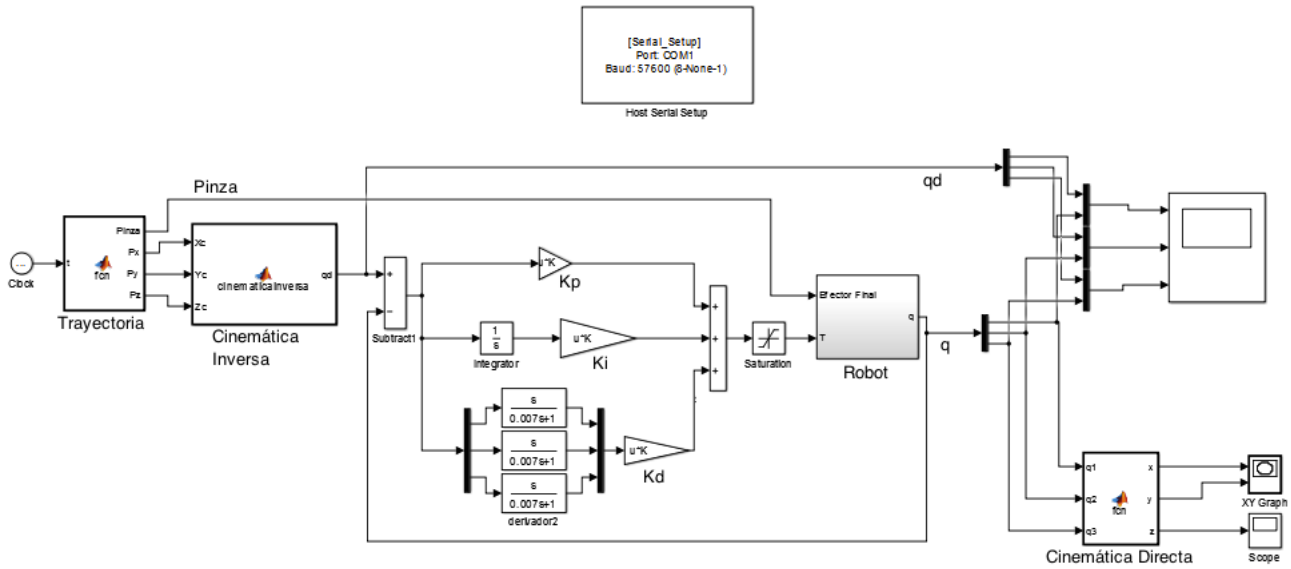


Figura 4.10: Diagrama de bloques para el control PID del Robot con movilidad en el efector final

En la Figura 4.10 se observa que el bloque de trayectoria tiene una entrada mas correspondiente al efector final. Para este caso se le da una condición inicial al efector final para poder sujetar la pieza y una condición final para soltar la pieza como se comento en el Capítulo 3. Su implementación fue similar a la del seguimiento de trayectoria circular, el problema fue presento un poco mas de oscilaciónes en sus posiciones articulares.

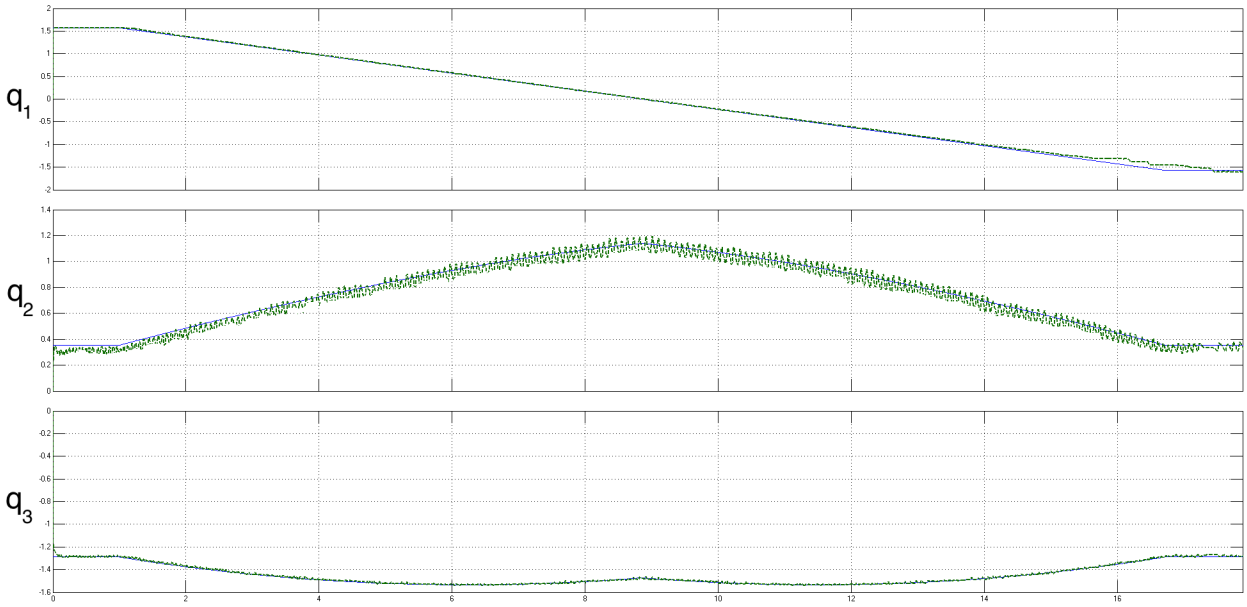


Figura 4.11: Comportamiento de los estados del Robot real con movilidad en el efector final

Como se puede observar en la Figura 4.11 las posiciones articulares de $q_1 y q_3$ siguen adecuadamente la trayectoria del Robot, en q_2 se tiene que ajustar mas las ganancias del controlador para que tenga un mejor comportamiento.

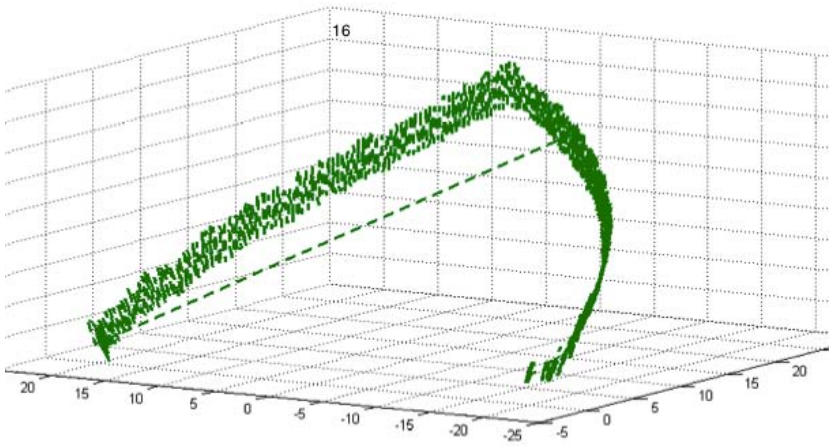


Figura 4.12: Grafica del seguimiento de trayectoria mediante parametrización del Robot real con movilidad en el efector final

En la Figura 4.12 se observa la trayectoria que generó el Robot en tiempo real, el problema que se obtuvo aquí fue las posiciones articulares que tomo la segunda articulación por lo que generó muchas oscilaciones en su movilidad.

Capítulo 5

Conclusiones

Durante el desarrollo de esta tesis se habilitó un prototipo experimental del Robot de 3 grados de libertad con arreglo cinemático antropomórfico, haciendo uso del equipo disponible desde su fabricación y adquisición de componentes nuevos, que se adecuaron a las necesidades de investigación para el Laboratorio de Control de Robots Industriales de la División de Ingeniería Eléctrica de la Facultad de Ingeniería de la UNAM. El proceso implicó una labor de búsqueda ya que no se contaba con algún tipo de manual o referencia de los motores a utilizar.

Para el caso de esta tesis no hubo selección del servomotor, debido que para inicios del proyecto ya se disponían con los servomotores Dynamixel AX-12. De estos motores se realizo una investigación detallada para obtener el número de articulaciones que el robot podría soportar y el espacio de trabajo que podría alcanzar. También fue necesario realizar la caracterización de los servomotores, afortunadamente el fabricante proporciona datos de la resolución que tiene cada parámetro y su rango de operación.

La parte que mas se complicó en el proyecto fue la implementación del controlador PID, ya que el microcontrolador funciona como un dispositivo de comunicación entre los motores y la computadora, realizando el envío de información bidireccionalmente pero no simultaneo para evitar colisiones(la comunicación la realiza por un mismo puerto). Esto evita que la información pueda fluir mas rápido, el proyecto funcionaría mejor si se utilizará un dispositivo que cuente mayor velocidad de procesamiento como es el caso de los DSP's o FPGA's o que todo el proceso lo realice en el microcontrolador.

Uno de los retos que se observaron durante el desarrollo del proyecto fue que existía una librería de los motores en MATLAB, lamentablemente no es compatible con SIMULINK que

es lo que se deseaba para la implementación de control en lazo cerrado. Para poder solucionar este problema se tuvo que comprender como funcionaban internamente los servomotores Dynamixel AX-12, existe muy poca información acerca de su programación, identificando toda la información necesaria para enviarles un paquete de datos y estos regresara un resultado, se puede observar su documentación en los Apendices.

En el Apendice F se incorporá un cronograma de las actividades realizadas durante el proyecto, por si se desea realizar un proyecto similar y observar el tiempo en el que se puede tardar en realizar dichas actividades, suele suceder que algunas actividades se terminan con un poco de anticipación o atraso.

Finalmente la experimentación se logró mediante el desarrollo de interfaces graficas que permiten obtener las posiciones articulares y la posición del efector final realizando una implementación en lazo abierto en el Robot y el desarrollo de trayectorias mediante la utilización de un controlador PID en lazo cerrado. En este caso se observaron algunos problemas, cuando se analiza su comportamiento en el Robot, el Robot presenta un pequeõ decaimiento.

Apéndice A

Configuración de Servomotores Dynamixel en RoboPlus

Por: Luis Román Hernández Torres.

El Software RoboPlus tiene la funcionalidad de manipular y configurar los motores Dynamixel mediante el uso de un dispositivo de Robotis.

Figura A.1: RoboPlus

En este caso se utilizara una Usb2Dynamixel para la configuración de los motores del tema de Tesis: Control y Diseño de un brazo Robótico de 3 GDL.

Figura A.2: USB2Dynamixel

Los pasos para realizar adecuadamente la configuración de los motores para no tener algún problema se explican a continuación:

Paso 1. Verificar que la Usb2Dynamixel se encuentre bien configurada

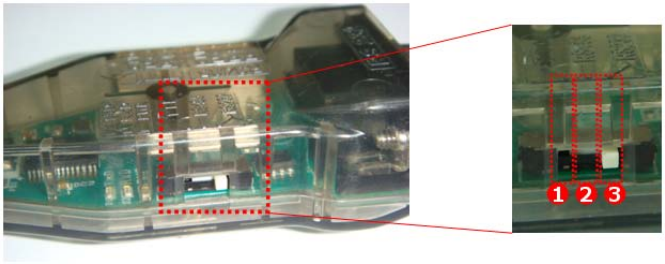


Figura A.3: Asignación de comunicación serial

En la Figura A.3 se observa la asignación del tipo de comunicación para el tipo de motor, en el caso de este trabajo de tesis el interruptor se debe seleccionar la comunicación TTL, en la Figura A.3 es la opción 1.

Paso 2. Verificar que los motores estén conectados adecuadamente

Figura A.4: Conexión de motores en Daisy Chain

Los motores Dynamixel AX tienen un conector con 3 puertos, este se conecta a la USB2Dynamixel, **cuando se configura por primera vez los motores es recomendable que se configure primero un motor sin conectar los demás en serie, ya que si se configuran todos a la vez pueden ocurrir problemas de comunicación debido a que todos tienen la misma dirección ID, los motores están predeterminados con un ID:1. Para mas información ver la pagina del Usb2Dynamixel: http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm.**

Paso 3. Abrir la aplicación RoboPlus

Figura A.5: Ventana Principal de RoboPlus

En la Ventana principal de RoboPlus se debe seleccionar la opción “Dynamixel Wizard“ como se ve en la Figura A.5, ya que es donde se va a realizar la comunicación con los motores.

Paso 4. Conectarse a la Usb2Dynamixel

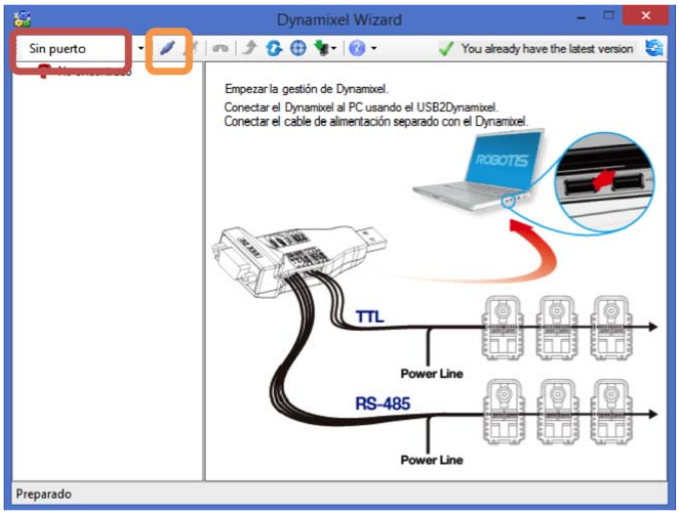


Figura A.6: Dynamixel Wizard

En este paso debe estar conectada la Usb2Dynamixel a la computadora y se debe saber cual es el puerto COM al que esta conectado el dispositivo.

En la ventana de la Figura A.6, ahí se va seleccionar el puerto en la opción que se enmarco de color rojo, se debe verificar que el puerto COM este asignado a la Usb2Dynamixel, de caso contrario no se enlazará a los motores.

Cuando se haya elegido el puerto COM correcto, seleccionar el botón que se en marco de color naranja en la Figura A.6.

Paso 5. Conectar motores para ser configurados

En este paso ya se debió de tener conectados los motores y deben de estar conectados a una fuente de alimentación de 12 V como máximo y mínimo 9 V.

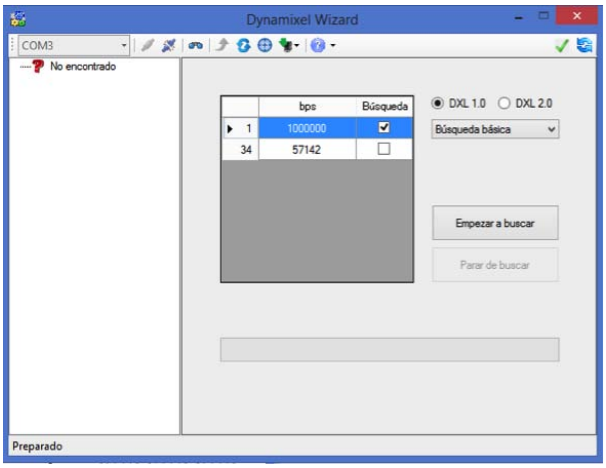


Figura A.7: Enlace de Configuración

En la Figura A.7 se selecciona la opción empezar a buscar, los bps asignados en este trabajo es el predeterminado que es 1000000 bps, por lo que al seleccionar el botón empezara a buscar los motores disponibles.

Paso 6. Selección de parámetros y motor a configurar

En este paso se explicará como asignar los IDs de los motores, para que no genere problemas de configuración cuando cada motor tenga asignado su propio ID como se explico en el **Paso 2**.

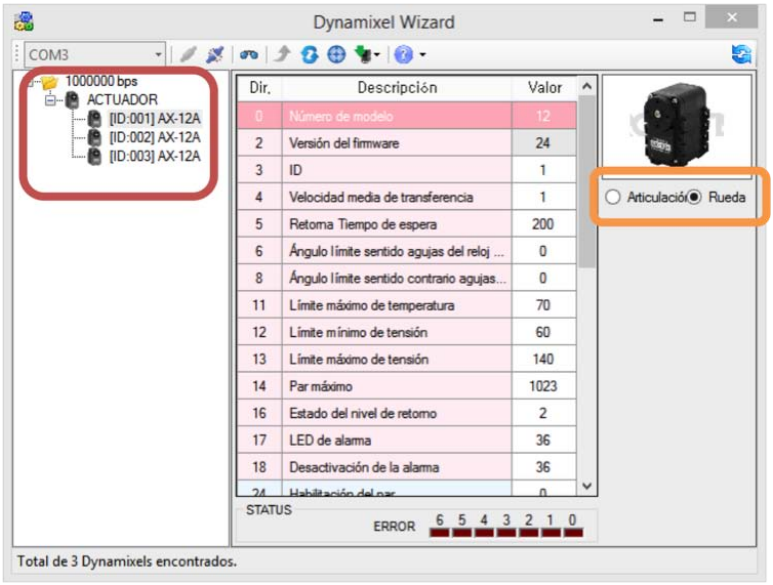


Figura A.8: Enlace de Configuración

En la Figura A.8 se enmarca de color rojo los motores con sus respectiva ID o nombre de identificación, esto permite que todos tengan nombres distintos para que no haya confusión en la comunicación. El ID se configura seleccionando la opción 3 o ID que se muestra en la Figura A.8, ahí va mostrar un botón y la opción de dirección para asignarle a los motores, si se configuran varios motores que tengan una identificación de números sucesivos empezando desde el No. 1 como se ve en el recuadro en rojo.

En la Figura A.8 se enmarca de naranja la modalidad de manejo del motor, ya que el modo articulación tiene la funcionalidad de manipular al motor como un servomotor, en el que se le asigna la posición y velocidad deseada para el motor. Estos servomotores no se comunican por señal PWM, lo hacen mediante comunicación UART. El modo Rueda permite manipular al motor como un motor de DC.

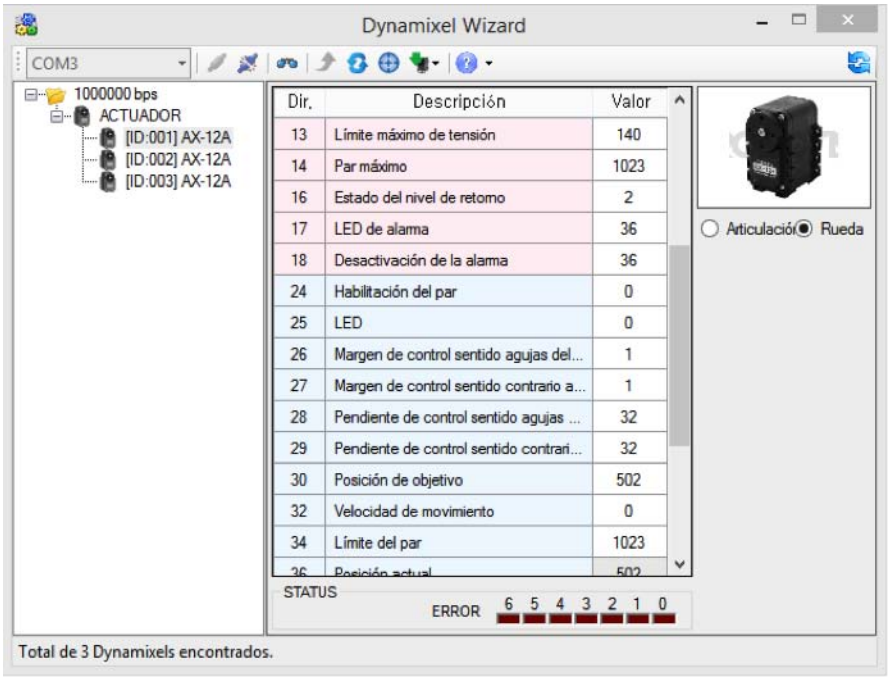


Figura A.9: Enlace de Configuración

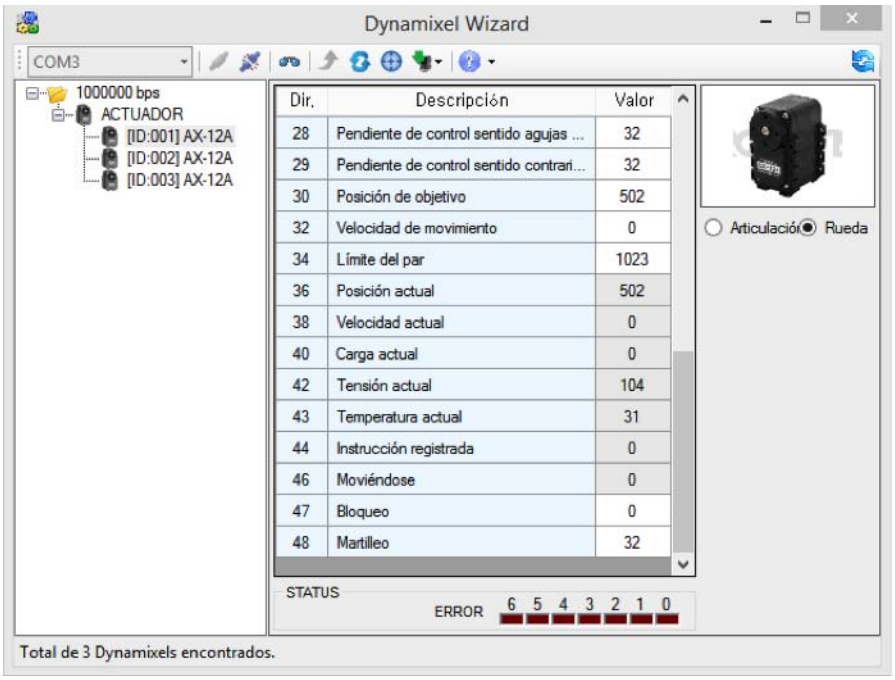


Figura A.10: Enlace de Configuración

En las Figuras A.8, A.9 y A.10 se muestran todos los parámetros que se deben de asignar para poder utilizar los motores. El único parámetro que cambia y no puede ser igual de cada motor es el 3 o ID. Para mayor información checar los manuales de Dynamixel. Las opciones 30 hasta la 46 no se toman en cuenta en la configuración, ya que con estos puedes mover el motor o tomar lectura de los datos que te puede entregar cada uno de ellos.

Apéndice B

Código de Instrucciones del microcontrolador

El microcontrolador Atmega 2560, se programó en el software de Arduino IDE. El código del microcontrolador y su breve explicación, se muestran a continuación.

```
//---------------------------LIBRERIAS------------------------------
            #include <DynamixelSerial1.h>
            #include <Servo.h>

            //----------------------Inicializacion de parametros----------
            Servo myservo;                  //servomotor de la pinza
            int opcion=4,i=0, inicio=0,c=0;  //SwitchCase, contadores
            int gripper=131,gripper_ant=131; //posición de la pinza
            char
            val1L=0,val1H=0,val2L=0,val2H=0,val3L=0,val3H=0;
            //valores que se envían a los 3 servomotores el valor HIGH y LOW
            int Position1,Position2,Position3;  //posiciones de potenciómetros de los motores
            int val11=0,val21=0,val31=0;   // estas variables ya no son necesarias
            char val1_L=0,val2_L=0,val3_L=0; // estas variables ya no son necesarias
            int opcion_ant=1;
            //opción anterior del switch case por si ocurre algún error
            int P1_ant=0,P2_ant=0,P3_ant=0;
            //guarda el dato de las posiciones por si ocurre un error

            //--------------------------- Inicializar puertos y motores---------------------------

            void setup(){
            Serial.begin(57600);   // Iniciar comunicación serial
            myservo.attach(10); //asignar puerto de comunicación de la pinza en el puerto 10
            delay(10);
            Dynamixel.begin(1000000,2);  // inicializar servomotores a 1Mbps en el puerto 2
            inicio=0;
            delay(10);

            Dynamixel.torqueStatus(1,OFF); //mantener detenido el motor 1
            delayMicroseconds(400);
            Dynamixel.torqueStatus(2,OFF); //mantener detenido el motor 2
            delayMicroseconds(400);
            Dynamixel.torqueStatus(3,OFF); //mantener detenido el motor 3
            delayMicroseconds(400);
            }

            // Funcion ciclica
            void loop(){

            if (Serial.available() > 0) {  //verifica si esta disponible el puerto
            //-----------------Lee datos para mover los motores----------
                opcion= Serial.read();
// si opción=1 se usa en modo servomotor y si es 2 como
// motor DC para obtener datos de las posiciones
                delayMicroseconds(80);
                gripper= Serial.read();
                delayMicroseconds(80);
                val1L= Serial.read();
                delayMicroseconds(80);
                val1H= Serial.read();
                delayMicroseconds(80);
                val2L= Serial.read();
                delayMicroseconds(80);
                val2H= Serial.read();
                delayMicroseconds(80);
                val3L= Serial.read();
                delayMicroseconds(80);
                val3H= Serial.read();
                delayMicroseconds(80);

              switch(opcion){
             //-------------Modo Servomotor-------------------
               case 1:
            if(opcion_ant !=2){
             /*------Lo ejecuta mientras no se haya implementado antes la opción 2, ya que al utilizar la opción 2 esta se ejecuta cíclicamente por el control en lazo cerrado y puede cometer un error, esto se hace para que no entre en este caso por si llega a ocurrir un problema de comunicación--------*/

            /*Activa modo servomotor o modo articulación como se conoce en ROBOPLUS  */
            Dynamixel.setEndless(1,OFF);
            delay(1);
            Dynamixel.setEndless(2,OFF);
            delay(1);
            Dynamixel.setEndless(3,OFF);
            delay(1);

               if(gripper_ant!=gripper){
                 myservo.write(gripper);          // envía datos a la pinza para abrirla o cerrarla
                  delay(10);
               }

            Dynamixel.syncPos3(val1L,val1H,val2L,val2H,val3L,val3H);
            /* se le asignan los datos a los 3 motores para moverlos a una velocidad asignada en la librería, y las posiciones de los servomotores que pide la función */
             delay(10);

                gripper_ant=gripper;

                opcion=0;
                opcion_ant=1;
            }   opcion=4;
                break;
```

```
/*  Función cíclica para el control en lazo cerrado de los motores  */
    case 2:
    opcion=3;
     if(gripper_ant!=gripper){
       myservo.write(gripper// envía datos a la pinza para abrirla o cerrarla
        delayMicroseconds(100);
     }
     /*  Asignar el modo rueda o motor de DC  para hacer girar el motor y leer los potenciómetros, este lo ejecutara  3 veces ya que tiende a ocurrir un error al mandarlo una sola vez la instrucción y no todos los motores la reciben de primera instancia*/
   if(inicio==0){
Dynamixel.setEndless(1,ON); //modo rueda para el motor 1
Dynamixel.setEndless(2,ON); //modo rueda para el motor 2
Dynamixel.setEndless(3,ON); //modo rueda para el motor 3
delayMicroseconds(5);

c++; /*incrementa contador c
inicio=0;
  val1L=0;
  val1H=0;
  val2L=0;
  val2H=0;
  val3L=0;
  val3H=0;
  Position1 = Dynamixel.readPosition(1); //Lee posición del motor 1
  Position2 = Dynamixel.readPosition(2); //Lee posición del motor 2
  Position3 = Dynamixel.readPosition(3); //Lee posición del motor 3

/* Manda a escribir los datos leidos via el Puerto com asignado del microcontrolador a la computadora, ya sea MATLAB o SIMULINK */

  Serial.print(Position1);
  Serial.print(",");
  Serial.print(Position2);
  Serial.print(",");
  Serial.println(Position3);
 // delay(10);

 if(c>=3){    //después de incrementarse 3 veces el contador activa inicio a 1
   inicio=1;
  }
 }
 else{   /* El activar a 1 "inicio" permite mandar a mover los motores a un cierto par dependiendo del que la computadora le envié a cada motor */
```



```
/*---------- si los datos no se encuentran en el Rango de utilidad, los vuelva a cero, para evitar alguna falla en el motor */
if(val1L > 255 || val1L < 0){
val1L=0;
}
if(val1H > 7 || val1H < 0){
/*para cuando val1H<3 el sentido es anti-horario y para cuando val1H >=4 cambia el sentido de giro horario  */
val1H=0;
}
if(val2L > 255 || val2L < 0){
val2L=0;
}
if(val2H > 7 || val2H < 0){
/*para cuando val2H<3 el sentido es anti-horario y para cuando val2H >=4 cambia el sentido de giro horario */
val2H=0;
}
if(val3L > 255 || val3L < 0){
val3L=0;
}
if(val3H > 7 || val3H < 0){
/*para cuando val3H<3 el sentido es anti-horario y para cuando val3H >=4 cambia el sentido de giro horario */
val3H=0;
}

Dynamixel.syncTorque3(val1L,val1H,val2L,val2H,val3L,val3H);
//Asigna el torque y sentido de giro del motor
     delayMicroseconds(30);

      Position1 = Dynamixel.readPosition(1); //Lee Posicion del motor 1
       delayMicroseconds(30);
      if(Position1>1023 || Position1<0){
/*Si llega a ser distinto valor al que puede dar el motor, devuelve la posición anterior*/
        Position1=P1_ant;
      }

       Position2 = Dynamixel.readPosition(2); //Lee Posicion del motor 2
        delayMicroseconds(30);
      if(Position2>1023 || Position2<0){
/*Si llega a ser distinto valor al que puede dar el motor, devuelve la posición anterior*/
        Position2=P2_ant;
      }

       Position3 = Dynamixel.readPosition(3); //Lee Posicion del motor 3
        delayMicroseconds(30);
      if(Position3>1023 || Position3<0){
/*Si llega a ser distinto valor al que puede dar el motor, devuelve la posición anterior*/
```

```
        Position3=P3_ant;
      }
     //Manda a escribir datos a la computadora
  Serial.print(Position1);
  Serial.print(",");
  Serial.print(Position2);
  Serial.print(",");
  Serial.println(Position3);

gripper_ant=gripper;
opcion=3;
opcion_ant=2;
P1_ant = Position1;
P2_ant = Position2;
P3_ant = Position3;
}
    break;
    //Datos predeterminados por si  no es ninguno de
los casos
    default:
    //opcion_ant=0;
    break;
 }
 }else{  // Si no esta disponible el puerto
if(opcion_ant==2){  /*si la ultima opción fue 2,
detiene los motores*/
Dynamixel.torqueStatus(1,OFF);
delayMicroseconds(100);
Dynamixel.torqueStatus(2,OFF);
delayMicroseconds(100);
Dynamixel.torqueStatus(3,OFF);
delayMicroseconds(100);
Dynamixel.torqueStatus(1,OFF);
Dynamixel.torqueStatus(2,OFF);
Dynamixel.torqueStatus(3,OFF);
delay(20);
//Dynamixel.syncTorque3(0,0,0,0,0,0);
//delayMicroseconds(200);
//Dynamixel.syncTorque3(0,0,0,0,0,0);

opcion=4;
}
if(opcion_ant==1){ /* Si la ultima opción fue 1,
genera un retardo para la espera de la siguiente
instrucción */
delay(500);
}

/* Envia los datos en cero de las posiciones
predeterminadamente por si los datos no los llega a
leer */
  Serial.print("0");
  Serial.print(",");
  Serial.print("0");
  Serial.print(",");
  Serial.println("0");
 }
 delayMicroseconds(400);
}
```

Apéndice C

Comandos utilizados en la librería de Arduino DynamixelSerial1

La librería para Arduino se utilizaron los siguientes comandos.

1. begin();

Descripción: Inicializa la comunicación serial en el Arduino.

Sintaxis: begin(Baudrate, DataControl);. Esta predeterminado como begin(1000000,2);.

Parámetros:

Baudrate esta predeterminado en el Arduino como 1000000 bps

DataControl esta predeterminado como 2, en donde este representa le puerto núm. 2 del Arduino.

$RxPin$ puerto de recepción de datos del Arduino.

$TxPin$ puerto de transmisión de datos del Arduino.

2. setEndless().

Descripción: Habilita o Deshabilita el modo continuo en la rotación del servomotor, o sea,

permite hacer girar el motor como un motor de DC donde solo necesitara el par de giro o como un servomotor en el que se le da la posición y velocidad a la que desea llegar.

Sintaxis: setEndless(ID,Status);

Parámetros:
ID Numero de identificación para cada servomotor.

Status el estado en el que se desea estar(ON o OFF). Para ON permite girar el eje del motor hasta que el usuario lo detenga. Para OFF permite girar el motor hasta la posición que se desea llegar con respecto al potenciómetro.

3. torqueStatus().

Descripción: Habilita o deshabilita el torque en el servomotor.

Sintaxis: torqueStatus(ID,Status);

Parámetros:
ID Número de Identificación del Servomotor.

Status ON o OFF para activar o desactivar el torque.

4. readPosition().

Descripción: Lee la posición de uno de los actuadores que se deseen conocer.

Sintaxis: readPosition(ID);

Parámetros:

ID Número de Identificación del Servomotor.

5. SyncPos3.

Descripción: Esta función permite desplazar 3 servomotores a posiciones deseadas. Este fun-

ciona para cuando el Endless esta desactivado.

Sintaxis:
syncPos3(Position1_L, Position1_H, Position2_L, Position2_H, Position3_L, Position3_H);

Parámetros:
ID Para este caso deben de estar predeterminadas los ID's como 1, 2 y 3 para cada servomotor.

PositionN_L (Position1_L, Position2_L y Position3_L) – Representa un valor de 0 – 255, este es una parte de la posición a la que se desea llegar.

PositionN_H(Position1_H, Position2_H y Position3_H) – Representa un valor de 0-3.

PosicionFinal El servomotor AX-12 puede llegar hasta una posición de 0300 grados o de 0 a 1023 cuentas. La posición es una suma entre el dato PositionN_L y PositionN_H.

PosicionFinal= 256*PosicionN_H + PosicionN_L

Por lo que si se desea llegar hasta 300 grados o 1023 cuentas.

PosicionFinal=256*3+255 = 1023

Por lo que si se desea llegar hasta 150 grados o 512 cuentas.

PosicionFinal=256*2+0 = 512

6. SyncTorque3.

Descripción: Esta función permite desplazar 3 servomotores al torque deseado . Este funciona para cuando el Endless esta activado.

Sintaxis:
syncTorque3(Speed1_L, Speed1_H, Speed2_L, Speed2_H, Speed3_L, Speed3_H);

Parámetros:

ID Para este caso deben de estar predeterminadas los ID's como 1, 2 y 3 para cada servomotor.

SpeedN_L (Position1_L, Position2_L y Position3_L) Representa un valor de 0 255, este es una parte de la posición a la que se desea llegar.

SpeedN_H(Position1_H, Position2_H y Position3_H) Representa un valor de 0 7.

ParFinal El servomotor AX12 puede llegar hasta un par de 0 a 1.5 Nm o de 0 a 1023 cuentas para el sentido anti-horario. Y de 1024 a 2047 en el sentido horario, donde 1024 representa un par de 0 y 2047 representa un par de 1.5 Nm aproximadamente. El ParFinal es una suma entre el dato SpeedN_L y SpeedN_H.

ParFinal=256*SpeedN_H+SpeedN_L.

Por lo que si se desea hacer girar el motor en sentido horario a la mitad del par, osea, 0 Nm o 0 cuentas.

ParFinal=256*0+0 = 0

Por lo que si se desea hacer girar el motor en sentido anti-horario a la mitad del par, osea, 0 Nm o 1024 cuentas aproximadamente.

ParFinal=256*4+0=1024

Por lo que si se desea hacer girar el motor en sentido horario a la mitad del par, osea, 0.75 Nm o 512 cuentas.

ParFinal=256*2+0 = 512

Por lo que si se desea hacer girar el motor en sentido anti-horario a la mitad del par, osea, 0.75 Nm o 1536 cuentas aproximadamente.

ParFinal=256*6+0=1536

Por lo que si se desea hacer girar el motor en sentido horario a la mitad del par, osea, 1.5 Nm o 1023 cuentas.

ParFinal=256*3+255 = 512

Por lo que si se desea hacer girar el motor en sentido anti-horario a la mitad del par, osea, 1.5 Nm o 2047 cuentas aproximadamente.

ParFinal=256*7+255=1536

Apéndice D

Ejemplos de Instrucciones del Servomotor Dynamixel AX-12 y AX-18

En este apartado se explicarán las instrucciones que se ocuparon en este trabajo mediante el uso de ejemplos en código hexadecimal. En codigo hexadecimal se toman 8 bits, por lo que en este caso se envian los datos en decimal tomando desde 0 a 255, para el caso de codigo hexadecimal es 0x00 hasta 0xFF.

Para el caso de begin como se observa en la librería para Arduino, este corresponde a generar un puerto de comunicación para mantener la comunicación entre el servomotor y el microcontrolador, por lo que esto no se explicará.

1. Lectura de datos

En el trabajo se utiliza la lectura de datos para adquirir los datos de posición del servomotor.

Por lo que si se desea obtener la posición a la que se encuentra el servomotor con ID=1, se debe envíar la siguiente Instrucción:

Paquete de Instrucción:

[255,255,ID,Length, Instruccion, PresentPosition_L, Byte_Read_Position,Checksum]

Parámetros:

Length= 4, esto corresponde a que hay 4 parámetros independientes en la cadena a enviar que son ID, Instrucción, PresentPosition_L y ByteReadPosition.

ID=1, corresponde al numero del servomotor que se desea conocer.

Instrucción= 2, corresponde al tipo de dato, el fabricante define un 2 si el dato que se ejecuta es de lectura.

PresentPosition_L= 36, Este corresponde al valor en estado bajo que se desea enviar contenido en la tabla de control.

Byte_Read_Position= 2, hace referencia al numero de datos en el que se desea recibir la posición, por lo que para el servomotor se necesita un estado de PosicionN_L y una PosicionN_H, un estado bajo y un alto como se observa en readPosition( ) de la librería de Arduino.

Checksum=(OR(ID+ Length+ Instruction+ PresentPosition_L+ ByteReadPosition)) AND (255)

La respuesta del servomotor es la siguiente:

Paquete de Status:

[255,255,ID,Length, Error, PositionN_L, PositionN_H,Checksum]

Del Paquete de Status los que son necesarios de adquirir es PosicionN_L y PosicionN_H, como ya se vio anteriormente, la posición Final del motor esta representada como:

PosicionFinal= 256*PosicionN_H+PosicionN_L

2. Escritura de datos

En el trabajo se utiliza la escritura de datos se utiliza para que el servomotor se configure o se desplace hasta algún valor deseado. Al final de cada instrucción el servomotor regresa un Status para verificar si la instrucción se realizo adecuadamente, esto no se mostrará en la explicación.

Por lo que si se desea configurarle el Endless en ON a un ID=2, se debe envíar la siguiente Instrucción:

Paquete de Instrucción:

[255,255,ID,Length, Instruccion, CCW_AngleLimit_L, CCW_AL_L, CCW_AL_H, Checksum]

Parámetros

Length= 5, esto corresponde a que hay 5 parámetros independientes en la cadena a enviar que son ID, Instrucción, CCW_AngleLimit_L, CCW_AL_L y CCW_AL_H.

ID=2, corresponde al numero del servomotor que se desea configurar

Instrucción= 3, corresponde al tipo de dato, el fabricante define un 3 si el dato que se ejecuta es de escritura.

CCW_AngleLimit_L= 8, Este corresponde al valor en estado bajo que se desea enviar contenido en la tabla de control.

CCW_AL_L=0 y CCW_AL_H=0

Para CCW_AL_L y CCW_AL_H el fabricante toma valores de 0 para un rotación completa, si se hubiera deseado que el Endless estuviera en OFF, los valores tendrían que ser CCW_AL_L=255 y CCW_AL_H= 3, debido a que CCW_Total = 256*CCW_AL_H+ CCW_AL_L,= 1023, corresponde a la máxima resolución asignada.

Checksum=(OR(ID+ Length+ Instruction+ CCW_AngleLimit_L+ CCW_AL_L+ CCW_AL_H)) AND (255)

Para el TorqueStatus, Si se desea deshabilitar el torque en el servomotor 3.

Paquete de Instrucción:

[255,255,ID,Length, Instruccion, TorqueEnable, Status, Checksum]

Parámetros:

Length= 4, esto corresponde a que hay 4 parámetros independientes en la cadena a enviar que son ID, Instrucción, TorqueEnable y Status.

ID=3, corresponde al numero del servomotor que se desea configurar.

Instrucción= 3, corresponde al tipo de dato, el fabricante define un 3 si el dato que se ejecuta es de escritura.

TorqueEnable=24, Este corresponde al valor que se desea enviar contenido en la tabla de control.

Status= 0, El valor del Status corresponde a 1 si se desea habilitar el torque y 0 si se desea deshabilitarlo.

Checksum=(OR(ID+ Length+ Instruction+ TorqueEnable+ Status)) AND (255)

3. SyncWrite

En la tesis se utiliza la función de SyncWrite para que todos los motores realicen la misma instrucción, pero se pueden tomar distintos parámetros para cada uno de ellos.

Por lo que si se desea mover los 3 servomotores a una Posicion y velocidad deseada, Si suponemos lo siguiente:
- Si queremos que el Servomotor 1, se mueva a una velocidad 200 cuentas, y llegue ha 512 cuentas o 150 grados.
- Si queremos que el Servomotor 2, se mueva a una velocidad 512 cuentas, y llegue ha 1023 cuentas o 300 grados.
- Si queremos que el Servomotor 3, se mueva a una velocidad 800 cuentas, y llegue ha 102 cuentas o 30 grados.

Nota: si el EndLess se encuentra en ON los valores de velocidad corresponderán al par de giro por lo que se deben de tomar de 0-1023 en sentido anti-horario y de 1024-2047. Si el

EndLess esta en OFF la velocidad toma de 0-1023.

Paquete de Instrucción:

[255,255,Broadcast_ID,Length, Instruction, StartingAddress, LengthData, ID_1, Position1_L, Position1_H, Speed1_L, Speed1_H, ID_2, Position2_L, Position2_H, Speed2_L, Speed2_H, ID_3, Position3_L, Position3_H, Speed3_L, Speed3_H, Checksum]

Parámetros:

Broadcast_ID=254, esta instrucción permite mandar la instrucción a todos los servomotores, y cada servomotor adquiere de la cadena el dato que le corresponde.

LengthData=4, es el numero de datos independientes a mandar a cada motor, son PositionN_L, PositionN_H, SpeedN_L y SpeedN_H.

Length=19, este dato se calcula como: Length=(LengthData+1)*NUMBEROFDYNAMIXEL + 4; , en donde NumberOfDynamixel =3, ya que corresponde al numero de servomotores al que se le mandara los datos.

StartingAddress=30, este dato se toma como predeterminado para el inicio en modo Syncwrite.

ID_N=N, este corresponde al numero de servomotor que se desea configurar el parámetro. Por lo que en el ejemplo ID_1=1, ID_2=2, y ID_3=3.

PositionN_L , corresponde a la posición en estado bajo de cada servomotor como se muestra en cada servomotor. En el Arduino se representa el comando como PositionN_L= PositionFinal (donde toma los 8 bits menos significativos).

Position1_L= 0, Position2_L=255 y Position3_L= 102

PositionN_H, corresponde a la posición en estado alto de cada servomotor como se muestra en cada servomotor. En el Arduino se representa el comando como PositionN_H= PositionFinal >> 8 (Toma en cuenta los 8 bits mas significativos).

Position1_H= 2, Position2_H=3 y Position3_H= 0

SpeedN_L , corresponde a la velocidad en estado bajo de cada servomotor como se muestra en cada servomotor. En el Arduino se representa el comando como SpeedN_L= SpeedFinal (donde toma los 8 bits menos significativos).

Speed1_L= 200, Speed2_L=0 y Speed3_L= 32

SpeedN_H, corresponde a la velocidad en estado alto de cada servomotor como se muestra en cada servomotor. En el Arduino se representa el comando como SpeedN_H=SpeedFinal >> 8 (Toma en cuenta los 8 bits mas significativos).

Speed1_H= 0, Speed2_H=2 y Speed3_H= 3

Checksum es la suma de todos los valores que se ingresan a la cadena con unas ciertas operaciones AND y OR.

Checksum=(OR(Broadcast_ID + Length + Instruction + StartingAddress + LengthData + ID_1 + Position1_L + Position1_H + Speed1_L + Speed1_H + ID_2 + Position2_L + Position2_H + Speed2_L + Speed2_H + ID_3 + Position3_L + Position3_H + Speed3_L + Speed3_H)) AND (255)

Nota: Para poder tener mas información acerca de donde checar mas ejemplos o la estructura de como realizar las instrucciones se puede checar en la Web como: http://hackerspace.cs.rutgers.edu/ library/Bioloid/doc/AX12.pdf(User's Manual 2006 06 14 dynamixel ax12) o en http:// support.robotis.com/ en/product/dynamixel/ax_series/dxl_ax_actuator.htm.

Apéndice E

Diseño Mecánico

En este apartado se ponen el tamaño de cada pieza, por si existe la posibilidad de hacer modificaciones o mejoras al proyecto en algun momento.

Los componentes mecánicos que ocupa el trabajo de tesis son los siguientes:

7 tornillos de 1/8 a 5cm de largo aproximadamente.
8 tornillos de 1/8 a 4cm de largo aproximadamente.
8 tornillos de 1/8 a 1.5cm de largo aproximadamente.
23 tuercas para tornillos de 1/8
1 base embrelada de 10x10cms
20 Tornillos M2 (2mm) de 1.2[cm] de largo
4 Tornillos M2 (2mm) de 1.6[cm] de largo
24 Tuercas para M2
7 gomas anti-derrapante para tornillos de 1/8

Los componentes se desarrollaron en AutoCad 2015 en la versión de Estudiante que es gratis la licencia por 3 años. Las medidas de las piezas estan dadas en centímetros, las piezas se hicieron en corte lasser en el material MDF de 3mm de grosor como se ven a continuación:

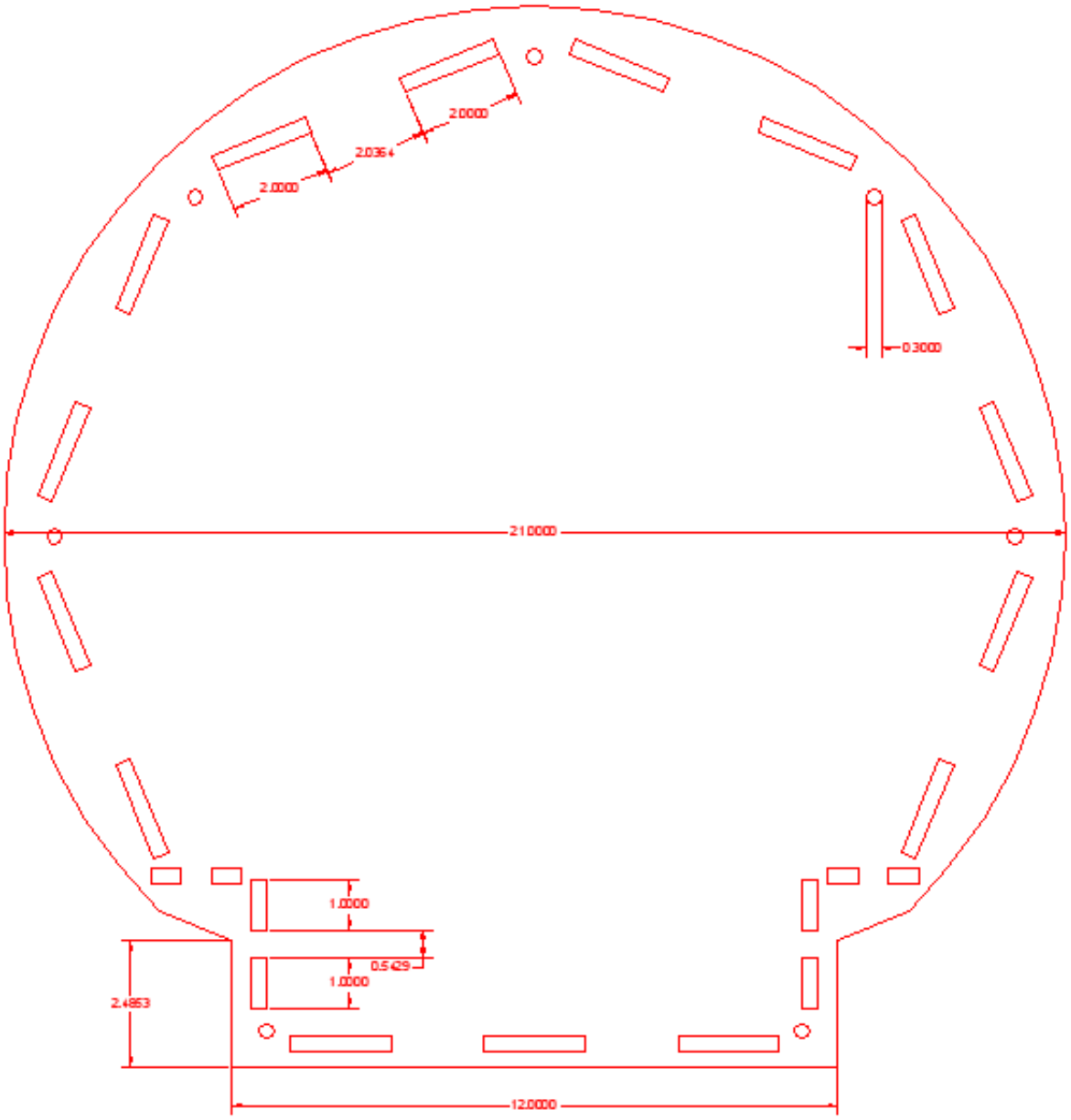


Figura E.1: Base para componentes

En la Figura F.1 se observa la placa donde se montan los componentes, en esta se sujetan mediante el uso de cinchos para poder sujertar el microcontrolador, en esta figura se realizaron unos orificios extras ya que no se tenian las medidas exactas del microcontrolador.

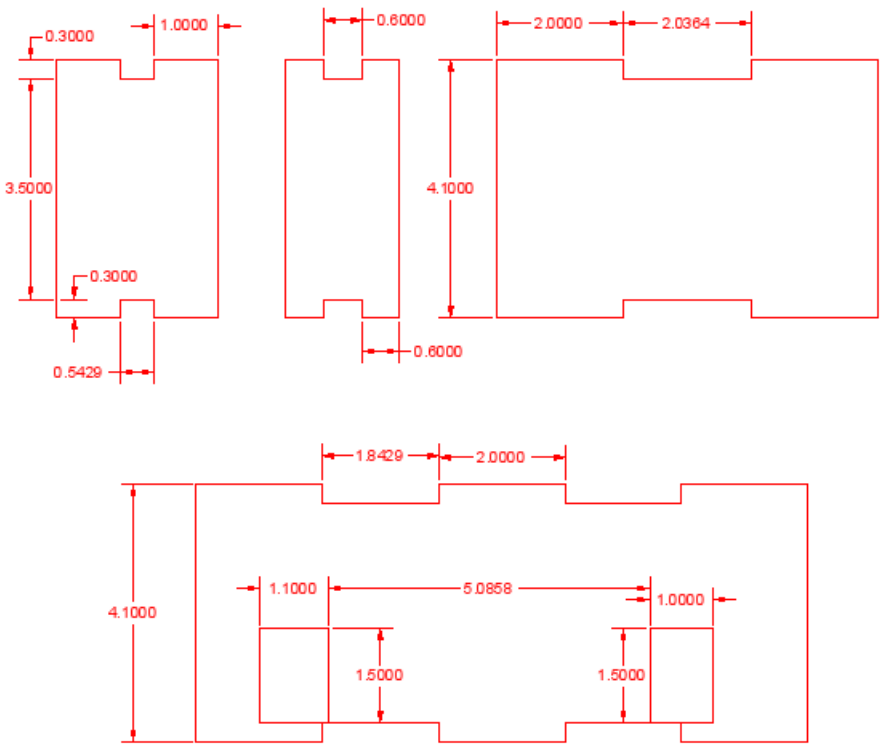


Figura E.2: Soportes para componentes

En la Figura F.2 se observa los soportes que se ocupan para que no caiga el peso sobre los componentes, la del lado superior derecho se repite 6 veces esa pieza, la de la parte superior izquierda y en medio 2 veces y la de abajo es solo una, en la ultima van las salidas de los conectores. En el soporte para embonar los conectores, hace falta el orificio del switch por lo que se debe realizar respecto a las medidas del Switch a utilizar.

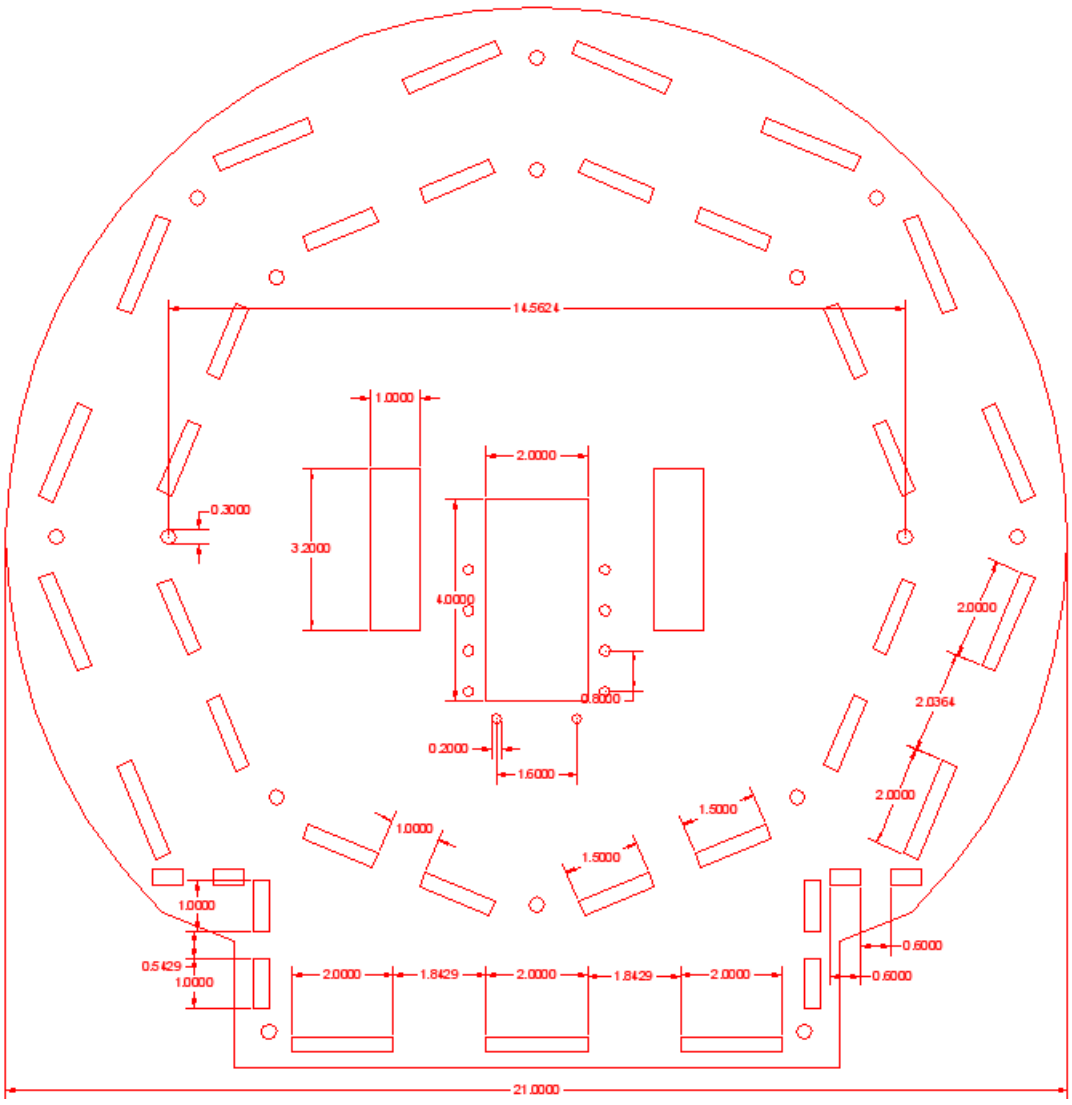


Figura E.3: Base para el servomotor de la cintura

En la Figura F.3 en la parte de enmedio de la figura va el primer servomotor, el cual se atornilla a los pequeños orificios que tiene a sus costados, esta placa esta ubicada arriba de la placa de los componentes tomando los soportes de la Figura F.2 en sus costados.

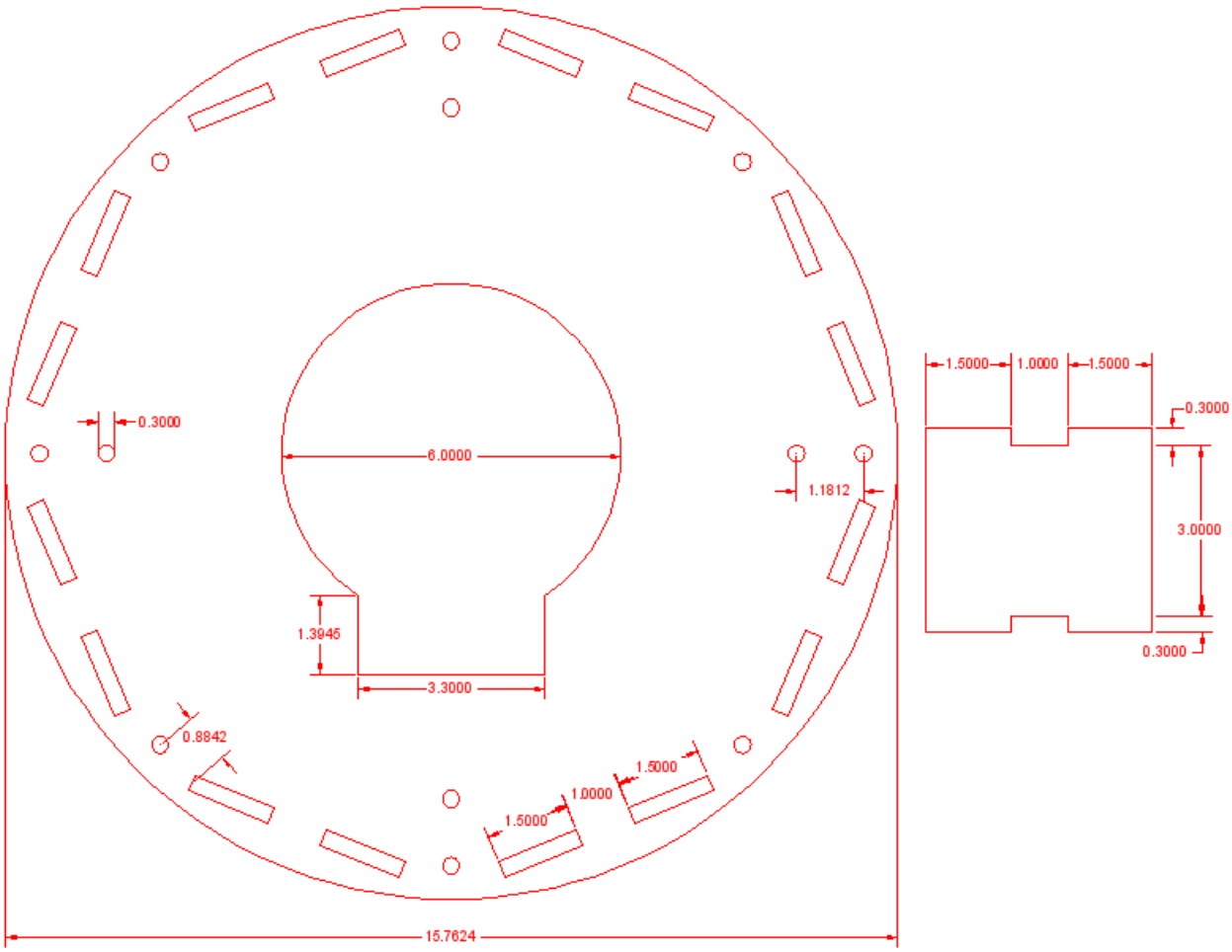


Figura E.4: Base para rodamiento

En la Figura F.4 es una base intermedia en la que permite que el servomotor pase por la parte de enmedio y en ella hay unos soportes como se pueden observa en el costado derecho del cual son 8 soportes iguales tomando la base de la Figura F.3 y F.4.

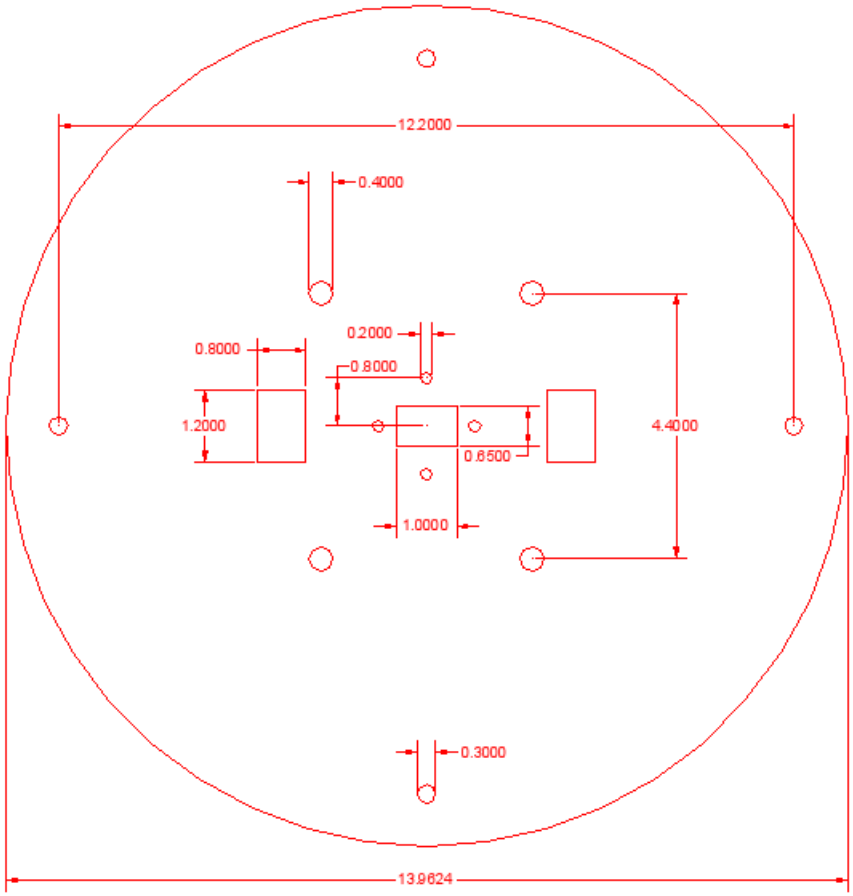


Figura E.5: Circuito de comunicación

En la Figura F.5 se encuentra una pieza circular en la que se atornilla el rodamiento o base embarelada como se ve en la Figura 3.12 en los orificios de los extremos de la Figura F.5 y en los que se acoplan a estos en la Figura F.4. En la Figura F.5 tambien se observan 3 orificios rectangulares, en los 2 de los extremos permite el paso de cableado y en el de enmedio no es necesario su uso esta de extra.

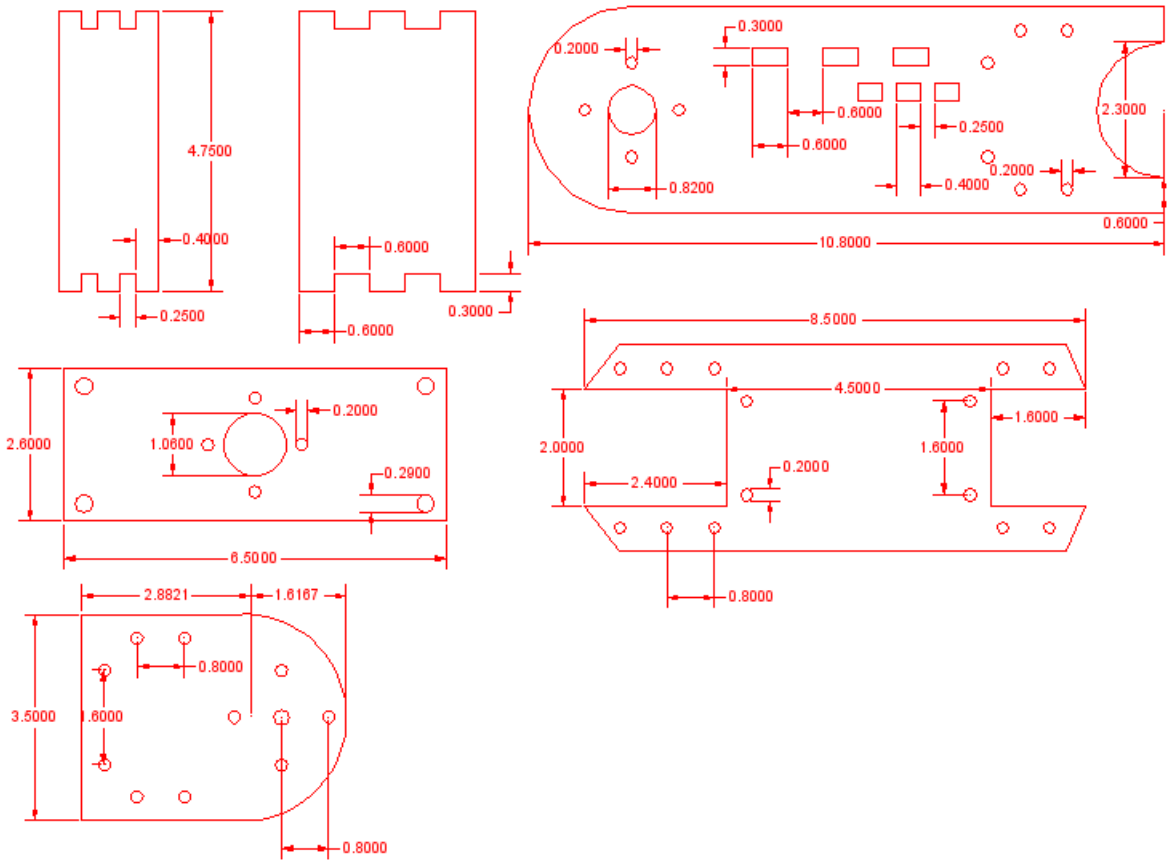


Figura E.6: Circuito de comunicación

En la Figura F.6 se encuentra las piezas que se ocuparon para armar el brazo. La pieza del lado derecho inferior se repite 2 veces, esta se atornilla en el segundo y tercer motor, se puede apreciar en la Figura 4.2 y la Figura 4.4, ya que se invierten las posiciones de los servomotores, esto se aprecia mejor cuando se ve el giro en sentido horario y anti-horario de cada motor, se hace para para que coincidan en la programación. La pieza superior derecha se repite 2 veces, esta solo se ajusta al tercer servomotor a estas piezas se le embonan las 2 piezas de la parte superior izquierda, estas van enmedio de las piezas para que den soporte al antebrazo, se utiliza tambien tornillos y piezas que se ocupan para el FP04F2 como se ve en la Figura 3.16.

Finalmente se atornilla el FP04F2 de la Figura 3.16 mediante la pieza en la parte inferior de la Figura F.6, esta se puede repetir 2 veces o 4 veces dependiendo del tamaño de los tornillos a utilizar, En el robot se utilizaron 4, 2 para cada pieza del antebrazo, ya que los tornillos

eran mas largos de lo esperado. En El FP04F2 se atornilla con tornillos de 2.8mm de ancho como se observa en la Tabla 3.3, la pieza ubicada en la parte media del lado izquierdo, esta se repite 2 veces, y esta se atornilla a la pinza como se ve en la Figura 3.6 y Figura 3.17.

Apéndice F

Cronograma

Cronograma																								
	Semana																							
Actividades	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Proponer un tipo de arreglo cinemático de un robot de 3 GDL																								
Analizar las cinemáticas y dinámicas del Robot a diseñar																								
Proponer alternativas a los servomotores AX-12																								
Analizar, investigar información acerca de los Servomotores AX-12																								
Diseñar circuitos eléctricos para los motores y el microcontrolador																								
Diseñar piezas para el robot en AutoCAD(Diseño Mecánico)																								
Establecer materiales para la construcción del Robot																								
Construir el Robot																								
Desarrollar interfaces en MATLAB & SIMULINK																								
Buscar alternativas al microcontrolador																								
Diseñar el código fuente para el microcontrolador																								
Diseñar un sistema en lazo cerrado del Robot en SIMULINK																								
Simular el comportamiento del Robot																								
Sintonizar el controlador PID del Robot																								
Implementar las interfaces desarrolladas en el Robot																								

Bibliografía

[1] Siciliano, B.(2010). *Robotics Modelling, Planning and Control*. En L. Sciavicco, L. Villani, G. Oriolo (Eds.). Londres: Springer- Verlag

[2] Lopez R., Mauro G.(2014). *Puesta en marcha del robot Omni Phantom de Sensable*. . México: Universidad Nacional Autónoma de México

[3] Jazar, Reza N.(2007). Theory of Applied Robotics Kinematics, Dynamics and Control. Second Edition, Denavit-Hartenberg Notation(pp. 233-242). Londres: Springer- Verlag

[4] Francisco(2013). *Grados de Libertad de un Robot*. Fecha de consulta:18 de Agosto de 2015. http://coparoman.blogspot.mx/2013/05/12-grados-de-libertad-de-un-robot.html

[5] Gonzalez,V.(2008). *Estructura de los Robots Industriales*. Fecha de consulta:18 de Agosto de 2015. http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/ Tema_5.4.htm

[6] ROBOTIS (2010). *AX-12/ AX-12+ /AX-12A*. Fecha de consulta: 18 de Agosto de 2015. http://support.robotis.com/en/product/dynamixel/ax_series/ dxl_ax_actuator.htm

[7] Castillo, Luis (2011). *Memoria EEPROM*. Fecha de consulta: 18 de Agosto de 2015. http://es.slideshare.net/be77ocas/eeprom-10170780

[8] *Memoria RAM*. Fecha de consulta: 18 de Agosto de 2015. http://www.significados.com/memoria-ram/

[9] *Microcontrolador*. Fecha de consulta: 18 de Agosto de 2015. http://microcontroladores-e.galeon.com

[10] *RoboPlus*. Fecha de consulta: 18 de Agosto de 2015. http://support.robotis.com/en/software/roboplus_main.htm

[11] *Arduino IDE*. Fecha de consulta: 18 de Agosto de 2015. https://www.arduino.cc

[12] *Manual Servomotor AX-12*. Fecha de consulta: 18 de Agosto de 2015. http://support.robotis.com/en/product/dynamixel/ax_series/ dxl_ax_actuator.htm