DISEÑO, ELABORACIÓN Y CONTROL DE UN BRAZO DE DOS GRADOS DE LIBERTAD

Reporte de proyecto

Dinámica y control de robots

Asesores:

- Dr. Martín Hernández Ordoñez
- Dr. Roger Miranda Colorado

Integrantes:

- Arias Lopez Fernando
- Banda Porras Marco Antonio
- Becerra Reyna Edgar Isauro
- Farías Chávez Gilberto
- Guzmán Garza Alan Emmanuel
- Ledezma Silva Aldo
- Martínez Sánchez Ruben
- Palacios Mendoza Gabriel
- Pérez Segura Eva Karina
- Ramírez suárez Ángel Arturo

Ingeniería Mecatrónica

Fecha: 10 de diciembre de 2013

Ciudad Victoria, Tamaulipas.

Contents

Abstract

In the following document we explain in detail the analysis and development of a multipurpose service robot starting with the mechanical aspects of the device, followed by the electronics and finally the programming and control algorithms employed.

Resumen

En este documento se presenta el análisis cinemático y dinámico de un robot de servicio multipropósito,los aspectos mecánicos del dispositivo, seguido de la electrónica empleada y finalmente los algoritmos de control y demás aplicaciones elaboradas.

1. PROBLEMÁTICA.

La robótica de servicio en México se encuentra en una etapa de desarrollo inicial, con únicamente algunos sistemas elaborados con propósito de investigación [1] y la mayor parte de las aplicaciones industriales importadas de otras empresas alemanas o estadounideneses en su mayoría.

El objetivo de la robótica es el sustituir la labor humana en tareas tediosas y peligrosas como la actividad industrial, el trabajo en situaciones extremas (desastres naturales, condiciones de alta o baja temperatura, ambientes tóxicos o radioactivos) o simplemente actividades tediosas (operaciones de ensamble, tareas sencillas o del hogar) en las cuales se preferiría el tener un sistema capaz de realizarlas de forma prolongada y con una alta precisión y exactitud. La tendencia actual es generar robots de asistencia que sean capaces de asistir en una variedad de tareas al usuario desde actividades del hogar a sociales y como acompañantes, siendo Japón y Alemania unos de los países líderes en la industria actualmente[3], [2].

1.1 Propuesta realizada.

Se propone la elaboración de un robot de servicio flexible que sea capaz de adaptar su estructura mecánica y algoritmos de control para realizar una amplia gama de actividades de asistencia al usuario como actividades de rescate, asistencia en el hogar, limpieza, manipulación y transporte de elementos y dispositivos varios, cuidado de menores de edad y personas con capacidades especiales o adultos mayores entre muchas otras.

Para ello se pretende desarrollar una base de control que permita el añadir y modificar los componentes estructurales y actuadores en base a esta tarea con un circuito electrónico que sea capaz de alimentar y controlar la diversa gama de actuadores deseados. Entre los elementos que se pretende desarrollar son distintos tipos de llantas y unidades de desplazamiento además de sistemas de actuadores multiherramienta (brazos, herramienta, etc.) que puedan ser implementados de manera fácil y rápida requiriendo una instalación mínima.

Se decidió realizar el análisis cinemático y dinámico de la estructura robótica movil y el manipulador de manera separada debido a la complejidad que implica el análisis de la estructura en su totalidad, de forma que es posible controlar la navegación y el manipulador de manera independiente, partiendo además de la premisa de realizar la manipulación cuando el navegador se encuentra en estado estático, lo cual implica que será posible considerarlo como una base fija.

1.2 Objetivo general.

Desarrollar una plataforma robótica de servicio de propósito general capaz de realizar una serie de actividades y permitir la reprogramación y reconfiguración de la misma.

1.2.1 Objetivos específicos.

- Elaborar una base robótica móvil estándar y adaptable a una gran variedad de actividades.
- Desarrollar la aplicación de control (framework) encargada de coordinar el resto de las aplicaciones robóticas a través de aplicaciones libres y abiertas. En este caso se ha seleccionado Python como el lenguaje de preferencia.
- Establecer un sistema de comunicación tipo Cliente/Servidor que permita comunicar y controlar la plataforma robótica a distancia empleando una gran variedad de dispositivos móviles.
- Desarrollar un manipulador y el torso del robot empleando análisis cinemático y dinámico para calcular el efecto de la carga estructural y efectos durante el movimiento del manipulador con y sin carga.
- Crear los algoritmos de reconocimiento en base a los sensores establecidos en el sistema robótico.
- Programar el sistema de control para el brazo del manipulador en base a los cálculos cinemáticos y dinámicos realizados.
- Diseñar y construir mecanismos y dispositivos aplicables en la plataforma móvil.

2. METODOLOGÍA.

Para el desarrollo del sistema aquí presentado se requirió hacer uso del método analítico en la etapa de diseño para generar una configuración óptima para la tarea deseada para el robot a través de los conocimientos adquiridos en las asignaturas de estática, dinámica, control clásico, cinemática y dinámica y control de robots en las cuales se requiere considerar diversas variables que impactan directamente en el desempeño de la estructura como es el caso de momentos y esfuerzos inherentes al movimiento de las cargas tanto de los eslabones y elementos que conforman el sistema robótico, como de fuerzas y pares generados por efecto de factores externos provenientes del entorno tales como la fuerza gravitatoria, fricción, efecto de impulso producto del entorno convectivo o de fuerzas que interactúan con el sistema, o producto de la interacción del sistema mismo con el entorno durante la modificación o manipulación del mismo.

Durante la etapa de análisis se generaron las ecuaciones necesarias para predecir el comportamiento estático y dinámico del robot, esto es, durante un estado estable y en transitorio al realizar operaciones además de un diseño utilizando una aplicación de diseño industrial, en este caso, SolidWorks2013 que permite facilitar y realizar una serie de pruebas sobre la estructura.

Finalizada la etapa de análisis y diseño se procedió a la elaboración del dispositivo, con lo cual se genera la estructura y se fabrican los circuitos de potencia encargados de impulsar los actuadores que permitirán el movimiento de todas las juntas y partes móviles del robot. Durante esta etapa se hizo uso de sistemas de potencia tales como transistores de tipo CMOS, los cuales permiten controlar grandes cargas de alta potencia a través de señales de voltaje de baja potencia.

Hecho ésto, se procedió a generar los algoritmos de control del sistema, tanto para el brazo manipulador como para el robot como una entidad integral. Para ello se ha hecho uso de diversos lenguajes especializados para cada uno de los módulos del mismo.

3. DESARROLLO.

3.1 Análisis mecánico.

Antes de la construcción del prototipo funcional del robot de servicio fue necesario realizar un análisis previo de los componentes mecánicos para seleccionar los componentes adecuados para la estructura y optimizar el diseño del robot. De esta forma se vuelve posible conseguir un diseño más eficaz para realizar las tareas para las cuales fué elaborado y más eficiente al disminuir la cantidad de recursos necesarios para su construcción.

Para realizar los análisis requeridos se hizo uso de dos métodos:

- Método analítico.
  Se realizaron las ecuaciones necesarias para la resolución de los problemas utilizando los principios de **diseño mecánico**, **análisis cinemático** y **análisis dinámico** de las estructuras haciendo uso de aplicaciones de computación científica como **Matlab** como apoyo para la resolución de las ecuaciones.

- Método de ingeniería asistida por computadora.
  Se elaboró un modelo del sistema en una aplicación de diseño industrial, siendo en este caso **SolidWorks 2013**, en la cual se puede visualizar el sistema y realizar simulaciones y análisis del comportamiento de éste ante diversas y complejas condiciones de forma rápida y sencilla.

SolidWorks 2013

.

SolidWorks 2013 es un conjunto de módulos integrados para ingeniería asistida por computadora desarrollados por Dassault Systemes para el diseño y análisis de esfuerzos de productos en el área mecánica, además de ser compatible con aplicaciones de manufactura industrial como tornos y fresadoras CNC [9].

Debido a la complejidad del brazo manipulador se realizó una sección separada dedicada exclusivamente a su diseño y elaboración además de una sección dedicada al análisis mecánico del sistema del navegador donde se analiza el robot como una entidad completa y en la cual se cuenta el brazo como un cuerpo móvil de éste.

Figura 3.1: Logotipo de SolidWorks 2013.

3.1.1 Brazo manipulador.

Un brazo manipulador consiste en una serie de vínculos casi rígidos conocidos como **eslabones** que se encuentran enlazados a través de articulaciones conocidas como **juntas** que permiten el movimiento relativo de los vínculos adyacentes. Estas articulaciones generalmente se instrumentan con sensores de posición que permiten medir la posición relativa de los vínculos adyacentes.

Existen diversos tipos de juntas, clasificándose principalmente en **prismáticas** y **rotacionales.** Las juntas prismáticas permiten un desplazamiento lineal con respecto a un eje de referencia mientras que las juntas rotacionales permiten obtener un desplazamiento angular. [20]

Para el robot elaborado se hizo uso de juntas rotacionales debido a que éstas permiten un mayor espacio de trabajo como se ha comprobado de forma teórica en [20] y [19].

Cinemática directa del manipulador.

Para obtener la cinemática directa del manipulador se hizo uso de la convención de Denavit-Hartenberg que permitió establecer los sistemas de referencia a través de los cuales se identifica el conjunto de ángulos y longitudes en el manipulador.

1. Se dibujó la estructura de acuerdo a la simbología convencional para identificar las juntas rotacionales y prismáticas en la estructura.

2. Posteriormente se establecieron los orígenes en cada una de las juntas iniciando desde la base del sistema manipulador, la cual se identificó como 0, hasta n siendo n-1 el número de grados de libertad que posee la estructura.

3. Se identificó el eje de acción sobre el cual giran las juntas de tipo rotacional que se encuentran en el manipulador y se estableció el **eje z** en éstas de acuerdo a la convención.

4. Siguiendo la regla de la mano derecha se estableció x partiendo perpendicular al eje z desde el eslabón n_i hasta n_{i-1} para cada uno de los eslabones del manipulador.

5. Se estableció y siguiendo la convención.

6. Se generó una tabla de parámetros de las juntas de acuerdo a lo establecido en la convención.

i	θ_i	α_{i-1}	d_i	a_{i-1}
1	θ_1	0	0	0
2	θ_2	-90°	0	0
e	0	0	L_e	0

7. Se generan las matrices de transformación homogénea de acuerdo a la fórmula general:

$$T_i^{i-1} = \begin{bmatrix} c_\theta & -s_\theta & 0 & a_{i-1} \\ c_{\theta i} s_{\alpha i} & c_\theta c_{\alpha i} & -s_{\alpha i-1} & -d_i s_{\alpha i-1} \\ s_\theta s_{\alpha i} & s_\theta c_{\alpha i-1} & c_{\alpha i-1} & d_i c_{\alpha i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

8. Las matrices de transformación homogéneas obtenidas son:

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_e^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

9. Se obtienen las matrices de transformación homogénea con respecto al orígen utilizando Matlab, las cuales pueden verse en las figuras 9 y 9.

$$T_2^0 = \begin{pmatrix} c_1 c_2 & -c_1 s_2 & -s_1 & 0 \\ c_2 s_1 & -s_1 s_2 & c_1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_e^0 = \begin{pmatrix} c_1c_2 & -c_1s_2 & -s_1 & -\boldsymbol{e}Ls_1 \\ c_2s_1 & -s_1s_2 & c_1 & \boldsymbol{e}Lc_1 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Análisis dinámico del brazo manipulador.

La dinámica es la ciencia encargada del análisis de los cuerpos mecánicos en movimiento y cómo las fuerzas interactúan con cada uno de los elementos de éste.

Es utilizado para el control acoplado en el cual se considera el efecto que tienen las fuerzas del resto de los eslabones desde el efector final y cómo se propagan desde éste hacia la base y viceversa. Una de las técnicas más utilizadas que hace uso de este tipo de modelo dinámico es el control por par calculado, mediante el cual es posible calcular la velocidad, par torsor y aceleración necesarios para que el robot sea capaz de seguir una trayectoria determinada.

Método de Newton-Euler. El método de Newton-Euler es un método desarrollado para calcular de manera iterativa las ecuaciones dinámicas que definen un sistema determinado. Para ello se calculan los efectos de las fuerzas desde la base hacia el efector final y posteriormente la retropropagación de dichas fuerzas a través de cada uno de los eslabones hacia la base.

Para ello se hace uso de una serie de ecuaciones bien definidas, comenzando con las iteraciones externas que pueden observarse a continuación para el robot especificado:

$$w_1^1 = R_0^1 w_0^0 + \dot{\theta}_1 z_1^1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}$$

$$\dot{w}_1^1 = R_0^1 \dot{w}_0^0 + \left[R_0^1 w_0^0 \; x \; \dot{\theta}_1 z_1^1\right] + \ddot{\theta}_1 z_1^1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix}$$

3.1.2 Navegador móvil.

Se realizó el diseño del navegador móvil haciendo uso de la aplicación para diseño industrial SolidWorks 2013 con el cual se obtuvo el modelo del sistema como se observa en la figura 3.2 y 3.3 respectivamente.

$$\dot{v}_1^1 = R_0^1[(\dot{w}_0^0 \; x \; P_1^0) + [w_0^0 \; x \; (w_0^0 \; x \; P_1^0)] + \dot{v}_0^0] = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

$$\dot{v}_{c1}^1 = (\dot{w}_1^1 \; x \; P_{c1}^1) + w_1^1 \; x \; (w_1^1 \; x \; P_{c1}^1) + \dot{v}_1^1 = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

$$F_1^1 = m_1 \dot{v}_{c1}^1 = m_1 \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

$$N_1^1 = I_1^{c1} w_1^1 + (w_1^1 \; x \; I_1^{c1} w_1^1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$w_2^2 = R_1^2 w_1^1 + \dot{\theta}_2 z_2^2 = \begin{bmatrix} -s_2 \dot{\theta}_1 \\ -c_2 \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$\dot{w}_2^2 = R_1^2 \dot{w}_1^1 + \left[R_1^2 w_1^1 \; x \; \dot{\theta}_2 z_2^2\right] + \ddot{\theta}_2 z_2^2 = \begin{bmatrix} -c_2 \dot{\theta}_1 \dot{\theta}_2 - s_2 \ddot{\theta}_1 \\ s_2 \dot{\theta}_1 \dot{\theta}_2 - c_2 \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

$$\dot{v}_2^2 = R_1^2[(\dot{w}_1^1 \; x \; P_2^1) + [w_1^1 \; x \; (w_1^1 \; x \; P_2^1)] + \dot{v}_1^1] = \begin{bmatrix} -s_2 g \\ -c_2 g \\ 0 \end{bmatrix}$$

$$\dot{v}_{c2}^2 = (\dot{w}_2^2 \; x \; P_{c2}^2) + w_2^2 \; x \; (w_2^2 \; x \; P_{c2}^2) + \dot{v}_2^2 = \begin{bmatrix} -s_2 \dot{\theta}_1 l_e z_2 \dot{\theta}_2 - s_2 g + (s_2 \dot{\theta}_1 \dot{\theta}_2 - c_2 \ddot{\theta}_1) l_e z_2 \\ -c_2 \dot{\theta}_1 l_e z_2 \dot{\theta}_2 - c_2 g + (c_2 \dot{\theta}_1 \dot{\theta}_2 + s_2 \ddot{\theta}_1) l_e z_2 \\ -s_1^2 \dot{\theta}_1^2 l_e z_2 + c_2 \dot{\theta}_1 l_e z_2 \dot{\theta}_2 \end{bmatrix}$$

$$F_2^2 = m_2\dot{v}_{c2}^2 = m_2\begin{bmatrix} -s_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 - s_2 g + (s_2\dot{\theta}_1\dot{\theta}_2 - c_2\ddot{\theta}_1)l_e z_2 \\ -c_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 - c_2 g + (c_2\dot{\theta}_1\dot{\theta}_2 + s_2\ddot{\theta}_1)l_e z_2 \\ -s_1^2\dot{\theta}_1^2 l_e z_2 + c_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 \end{bmatrix}$$

$$N_2^2 = I_2^{c2}\dot{w}_2^2 + (w_2^2 \; x \; I_2^{c2}w_2^2) = [0]$$

$$n_2^2 = N_2^2 + R_e^2 n_3^3 + (P_{c2}^2 \times F_2^2) + (P_3^2 \times R_3^2 f_3^3) = \begin{bmatrix} m_2\, l_e z_2(-c_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 - c_2 g + (c_2\dot{\theta}_1\dot{\theta}_2 + s_2\ddot{\theta}_1)l_e z_2) \\ -[l_e z_2 m_2(-s_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 - s_2 g + (s_2\dot{\theta}_1\dot{\theta}_2 - c_2\ddot{\theta}_1)l_e z_2)] \\ 0 \end{bmatrix}$$

$$T_2 = (n_2^2)^T z_2^2 = 0$$

$$F_2^2 = f_2^2$$

$$f_1^1 = R_2^1 f_2^2 + F_1^1$$

$$n_1^1 = N_1^1 + R_2^1 n_2^2 + (P_{c1}^1 \times F_1^1) + (P_2^1 \times R_2^1 f_2^2) =$$

$$\begin{bmatrix} c_2[m_2\, l_e z_2(-c_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 - c_2 g + (c_2\dot{\theta}_1\dot{\theta}_2 + s_2\ddot{\theta}_1)l_e z_2)] + s_2[l_e z_2 m_2(-s_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 - s_2 g + (s_2\dot{\theta}_1\dot{\theta}_2 - c_2\ddot{\theta}_1)l \\ 0 \\ -s_2[m_2\, l_e z_2(-c_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 - c_2 g + (c_2\dot{\theta}_1\dot{\theta}_2 + s_2\ddot{\theta}_1)l_e z_2)] - c_2[l_e z_2 m_2(-s_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 - s_2 g + (s_2\dot{\theta}_1\dot{\theta}_2 - c_2\ddot{\theta}_1) \end{bmatrix}$$

$$T_1 = (n_1^1)^T z_1^1 = \big[-s_2[m_2\, l_e z_2(-c_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 - c_2 g + (c_2\dot{\theta}_1\dot{\theta}_2 + s_2\ddot{\theta}_1)l_e z_2)] - c_2[l_e z_2 m_2(-s_2\dot{\theta}_1 l_e z_2\dot{\theta}_2 - s_2 g + (s_2\dot{\theta}_1\dot{\theta}_2 - c_2\ddot{\theta}_1)l_e z_2)]\big]$$

Se obtuvo la cinemática directa del manipulador [19] y se generaron las siguientes matrices de transformación homogéneas teniendo en cuenta la consideración actual de que el sistema del manipulador se encuentra fijo a una base que no cambia su posición en el espacio.

$$T_0^1 \tag{3.2}$$

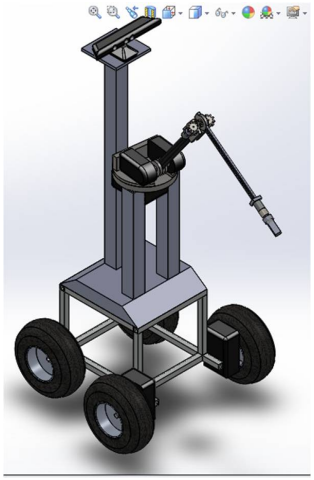


Figura 3.2: Diseño conceptual del robot de servicio utilizando SolidWorks 2013.

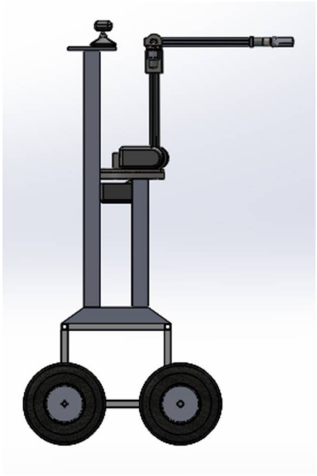


Figura 3.3: Diseño conceptual del robot de servicio utilizando SolidWorks 2013.

3.2 Programación y control.

3.2.1 Comunicación por red.

La selección de la arquitectura del sistema de información y control del robot representa un factor clave en las limitaciones de sus capacidades operativas y que resulta influenciado principalmente por los recursos disponibles para la elaboración del dispositivo.

Gran parte de los sistemas robóticos actuales hacen uso de una arquitectura integrada en embebidos, ya sea a través del uso de microprocesadores y microcontroladores integrados en un sistema de control unificado o a través de la inclusión de computadoras integradas en el robot que les permiten realizar tareas de procesamiento mucho más complejas. En general los robots suelen incorporar las computadoras en aplicaciones más complejas y para las cuales las capacidades de un circuito integrado resultarían insuficientes, lo cual sin embargo añade un mayor peso y dimensiones a la estructura.

En años recientes y con la mayor presencia de las redes de comunicaciones inalámbricas, parti-

cularmente de protocolos estandarizados como es el caso del protocolo WiFi o el protocolo TCP/IP que permiten a los sistemas el comunicar instrucciones complejas y que ha sido implementado de manera intensiva en aplicaciones de automatización industriales y de domótica.

Estas tecnologías pueden ser utilizadas en lo sistemas robóticos para permitir entre otras cosas, el enviar comandos a un sistema robótico con acceso a la red y que puede ser controlado sin importar su localización. Esto además puede ser aprovechado para lograr reducir las necesidades computacionales de los sistemas robóticos y mecatrónicos elaborados al convertir el sistema robótico en un cliente más de una red de computadoras y desplazando las necesidades de computación intensiva a un servidor de alta capacidad cuya localización es de poca importancia para el desarrollador.

Utilizando esta arquitectura se vuelve posible el contar con uno o más clientes que envíen órdenes y monitoreen el estado del sistema robótico a distancia haciendo las peticiones al servidor, el cual se encarga de realizar al computación y algoritmos de control necesarios y enviar únicamente el resultado ya procesado al sistema robótico para que realice la acción solicitada. De manera inversa es posible lograr que el robot envíe los datos de manera cruda al servidor, el cual se encarga de interpretar la información y convertirla en una forma útil al usuario sin requerir ciclos de procesamiento por parte del robot para obtener el resultado, lo cual libera al sistema de recursos que pueden ser destinados a otras tareas.

Python.

Figura 3.4: Logo del lenguaje interpretado Python.

Python es un lenguaje de programación interpretado de propósito general, lo cual significa que es ejecutado línea por línea al ser llamado a diferencia de lenguajes compilados como C y Java que generan un ejecutable, lo cual permite que Python pueda ser utilizado para realizar pruebas de prototipado rápidas [10].

El lenguaje es capaz de soportar programación estructurada, programación orientada a objetos, programación funcional, entre muchas otras y al ser un lenguaje libre y abierto con una comunidad de desarrolladores muy activa, constantemente se realizan mejoras y se liberan librerías que extienden la funcionalidad de este lenguaje.

En el proyecto del robot de servicio se buscó que Python fuera implementado como lenguaje principal de manejo de control del robot, además de implementarlo en conjunción de la librería Tor-

nado para la comunicación del sistema robótico con un servidor web a través del cual se coordinan las actividades del robot y con ello se permite funcionalidades de telepresencia.

Tornado.

Figura 3.5: Logo de la librería para operaciones servidor/cliente Tornado.

Tornado [11].

KineticJS.

Figura 3.6: Logo de la plataforma de desarrollo KineticJS.

KineticJS es un framework (TRADUCCION) elaborado en JavaScript para la estructura Canvas de HTML5 que permite la elaboración de animaciones de alto desempeño, transiciones, gestión de nodos, capas, filtrado, almacenamiento en caché, manejo de eventos para aplicaciones de escritorio y móviles entre muchas otras características [16].

De creación relativamente reciente al igual que el protocolo HTML5, permite la generación de contenido dinámico en las páginas Web de manera sencilla y rápida utilizando un paradigma orientado a objetos en el cual cada elemento de la aplicación representa una entidad con atributos y métodos propios.

3.2.2 Visión computacional.

Uno de los objetivos más importantes para los robots de servicio consiste en dotarlos de la capacidad de reconocer su entorno y ser capaces de realizar un "mapeo" del área de trabajo en la cual se desempeñan de manera que sean capaces de reconocer y evadir los obstáculos que se presenten durante la navegación, sean capaces de buscar y transportar objetos solicitados por el usuario, además de reconocer y diferenciar entre distintas personas con las cuales interactuará el dispositivo.

Para lograr todas estas tareas se requiere contar con sensores que permitan al robot el visualizar su entorno como suelen ser videocámaras y sensores de presencia o distancia, y en los diseños más avanzados, lásers que permitan diferenciar con precisión la profundidad y las texturas de objetos en el espacio.

Antiguamente el contar con este tipo de sistemas implicaba un alto costo, particularmente en aquellas aplicaciones que han hecho uso de sistemas de detección por láser de alta calidad, los cuales llegan a tener costos desde $\$90,000{,}00$ o más. En la actualidad, gracias a la investigación realizada en tecnología de visión y sensores para la industria de los videojuegos, se ha logrado generar sistemas integrales de sensores relativamente compactos y económicos, de los cuales se destaca el MS Kinect [12], creado por la empresa PrimeSense para la compañía Microsoft como una interfaz visual entre el usuario y la consola de videojuegos X-Box.

MS Kinect.

Figura 3.7: Dispositivo interfaz Microsoft Kinect.

El sistema MS Kinect es una interfaz de desarrollo visual cuyo propósito original consistió en un dispositivo capaz de optimizar la experiencia del usuario en videojuegos al permitir una interacción más natural con el ambiente mediante algoritmos capaces de reconocer al usuario y permitirle interactuar con la realidad virtual generada por los videojuegos.

Este dispositivo sin embargo, vió un uso más práctico en el área de robótica debido a la enorme cantidad de prestaciones que es capaz de proveer en un empaque compacto y económico, permi-

tiendo al robot obtener datos confiables del entorno con mayor facilidad.

Entre las prestaciones que provee el dispositivo MS Kinect se encuentran [13]:

- Cámara RGB con tres canales de datos en una resolución de 1280x960 pixeles.
- Emisor y receptor infrarrojo para la detección de profundidad.
- Arreglo de cuatro micrófonos para captura de audio y localización de la fuente de origen del sonido.
- Acelerómetro de tres ejes para el reconocimiento de la inclinación del dispositivo.
- Motor para control de un grado de libertad del dispositivo.

Libfreenect.

Figura 3.8: Logo de la comunidad de software libre OpenKinect.

La librería libfreenect comenzó como un esfuerzo conjunto por la comunidad hacker y la organización Adafruit para comunicar el dispositivo MS Kinect de la empresa Microsoft con la computadora debido a que éste fué diseñado con fines de entretenimiento, estando la empresa originalmente en contra de que su dispositivo fuera modificado. Haciendo caso omiso a ésto y a través de una competencia establecida por la empresa Adafruit se ofreció entregar $\$3,000{,}00$ dólares a aquél que lograse comunicar el MS Kinect con cualquier sistema operativo, generándose una carrera entre los mejores hackers por lograrlo mediante ingeniería inversa. El hacker Héctor Martín logró realizarlo y obtuvo el premio, liberando además su código para que fuera utilizado por toda la comunidad de software libre.

Así inició la comunidad OpenKinect, la cual desarrolló en base al código liberado por Héctor la librería *libfreenect* que permite reconocer y accesar cada uno de los elementos provistos por el MS Kinect a bajo nivel utilizando el lenguaje C. En la actualidad se han generado wrappers para otros lenguajes y ahora es posible utilizar la librería desde Python, Java, entre muchos otros [14].

3.2.3 Etapa de control.

Arduino 2560.

Figura 3.9: Placa del Arduino Mega 2560.

El Arduino Mega 2560 es una tarjeta microcontroladora basada en el ATMega 2560. Posee 54 pines de entrada/salida digitales con 15 que pueden ser utilizados como salida PWM, 16 entradas analógicas, 4 UARTs, un oscilador de cristal de 16MHz, conexión USB y botón de reset.

Características de operación[18]:

Voltaje de operación	5 volts
Voltaje de entrada	7 a 12 volts
Pines de entrada/salida digitales	54 con 15 de salida PWM
Pines de entrada analógica	16
Corriente directa por pin	40mA
Corriente directa para pin de 3.3 volts	50mA
Memoria flash	256kB (8 utilizados por el bootloader).
SRAM	8kB
EEPROM	4kB
Reloj	16MHz

La plataforma es ampliamente desarrollada y posee una comunidad de desarrolladores libres que permiten que la plataforma posea una amplia conectividad y acceso a librerías especializadas lo cual lo vuelven una solución de desarrollo ideal para el prototipado.

3.3 Electrónica.

3.3.1 Etapa de potencia.

L298N.

Figura 3.10: Integrado para control de motores L298N.

El integrado utilizado para el control del motor de corriente directa del navegador es el modelo L298N el cual es un integrado altamente utilizado para aplicaciones con requerimientos de corriente de hasta 2 Amperes en estado estable y 2.5 a 3 Amperes en etapa de transitorio.

Los pines del integrado poseen la siguiente configuración:

3.3.2 Sensores.

Codificador óptico.

Para realizar el control a través de odometría y la retroalimentación por medio de la cinemática y dinámica inversas del brazo manipulador se hace uso de dispositivos electrónicos conocidos como

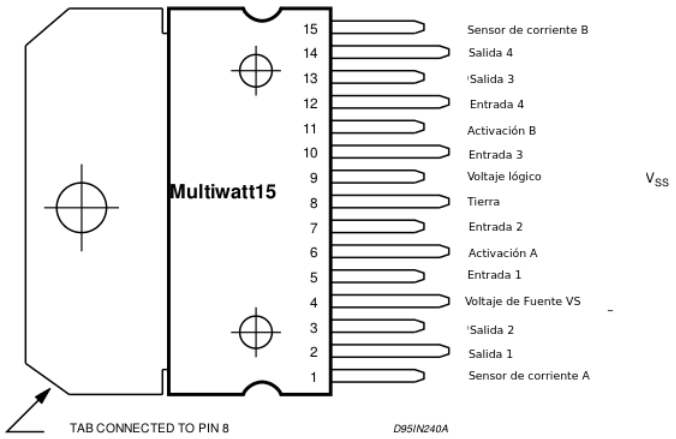


Figura 3.11: Pines del integrado L298N.

codificadores ópticos, los cuales consisten en una serie de discos acoplados a las flechas de giro del motor o a dispositivos o mecanismos encargados de transmitir el movimiento de éstas (cadenas, engranes, etc.) y entregar una medida del desplazamiento en un formato específico que pueda ser utilizado por el sistema para determinar su posición y otras variables de interés como velocidad y aceleración.

Para esta operación se ha seleccionado el uso de codificadores ópticos debido a su alta confiabilidad y facilidad de uso.

4. CONCLUSIONES.

A través del desarrollo de este sistema fue posible poner en práctica todo el conjunto de conocimientos adquiridos durante la carrera de Ingeniería Mecatrónica además de permitir observar la forma en que se vuelve posible integrar distintas tecnologías para generar un sistema integral reprogramable para realizar una serie de tareas específicas.

Se elaboró el análisis cinemático y dinámico de un sistema de dos grados de libertad con el propósito de utilizarlo como base para el control desacoplado y posterior avance hacia controles acoplados que consideren el efecto de los eslabones en la estructura y calcular las fuerzas y pares necesarios para desplazar el brazo manipulador hacia una trayectoria especificada.

A. Códigos del framework KineticJS.

A.1 Declaración de entidades utilizando KineticJS.

A continuación se presenta un ejemplo sencillo del uso de la plataforma KineticJS en el cual se declara una nueva capa de trabajo para KineticJS y se genera una entidad llamada círculo que permite la creación de un círculo rojo con contorno negro en el navegador web empleando únicamente KineticJS.

A.1.1 Instrucciones.

1. Generar un nuevo archivo en un procesador de textos, en este caso se hizo uso del editor libre **Sublime Text 2** en el sistema operativo Ubuntu Linux tal y como se observa en la figura A.1.

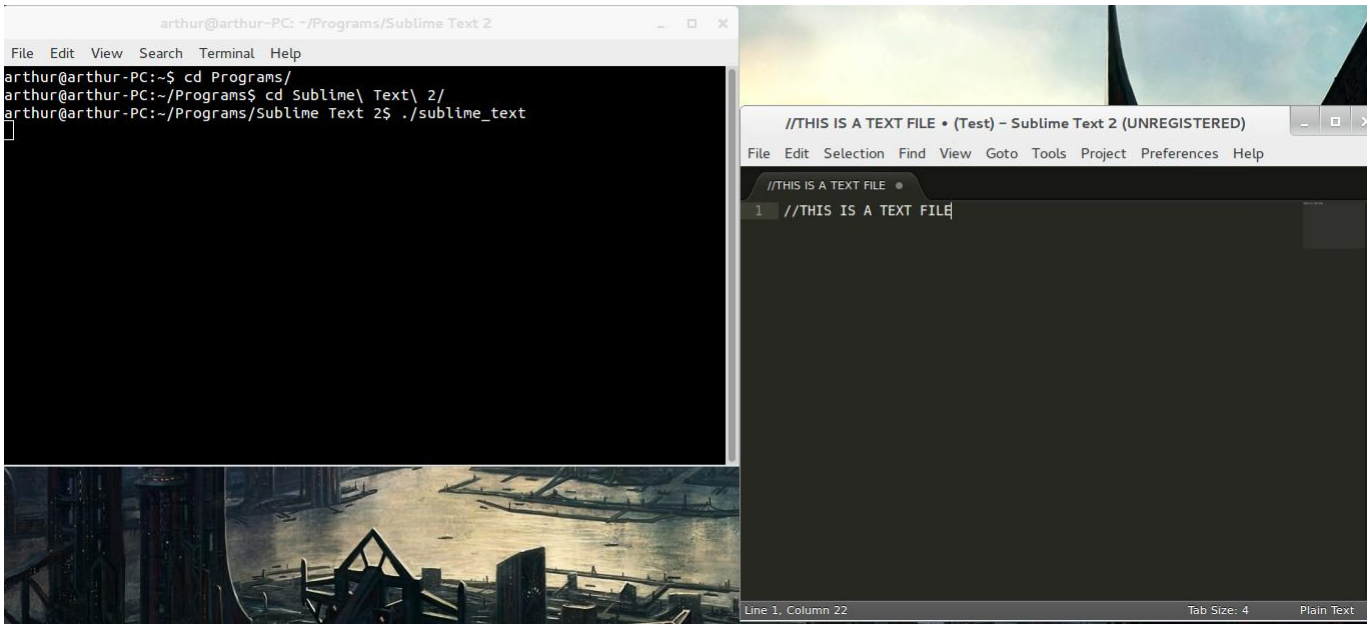


Figura A.1: Creación de un nuevo archivo de texto en Sublime Text 2.

2. Se introduce el código presentado en A.3 en el archivo y se guarda en formato .html.

3. Se abre el navegador y se observa el resultado. **NOTA:** Debido a la forma en que el código adquiere los archivos de KineticJS se vuelve absolutamente necesario que se cuente con una conexión a Internet. En caso contrario el documento aparecerá en blanco. En la figura A.2

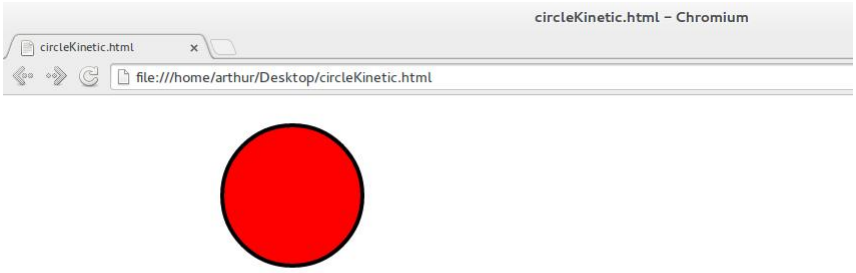


Figura A.2: Resultado obtenido de la implementación del código en el navegador web Chromium.

A.1.2 Código.

```
  <!DOCTYPE HTML>
<html>
  <head>
    <style>
      body {
        margin: 0px;
        padding: 0px;
      }
    </style>
  </head>
  <body>
    <div id="container"></div>
    <script src="http://d3lp1msu2r81bx.cloudfront.net/kjs/js/lib/
       kinetic-v4.7.0.min.js"></script>
    <script defer="defer">

    //*************************************************************************

    //****************************HERE BEGINS THE KINETICJS CODE
       .******************************
    //*************************************************************************

    //Defines a new KineticJS stage where the animation will be played.
      var stage = new Kinetic.Stage({
        container: 'container',       //It is of type 'container'
        width: 578,                   //Width of 578 pixels.
        height: 200                   //Height of 200 pixels.
      });

      //Defines a new layer that will contain the objets.
      var layer = new Kinetic.Layer();

```

Figura A.3: Código empleado para la generación de la figura en KineticJS.

```
29        //Defines a new circle entity.
30        var circle = new Kinetic.Circle({
31          x: stage.getWidth() / 2,      //Positioned horizontaly at the
              middle of the stage.
32          y: stage.getHeight() / 2,     //Positioned vertically at the
              middle of the stage.
33          radius: 70,                   //With a radius of 70.
34          fill: 'red',                  //Color red.
35          stroke: 'black',              //Black contour.
36          strokeWidth: 4                //The contour has a width of 4.
37        });
38
39        //Adds the shape to the layer
40        layer.add(circle);
41
42        //Adds the layer to the stage
43        stage.add(layer);
44      </script>
45    </body>
46  </html>
```

Ejemplo tomado de [17].

B. Códigos del framework Tornado.

B.1 Estableciendo conexión con un servidor web.

El siguiente código entrega una respuesta a una cadena introducida por el usuario en la consola de la terminal.

1. Se genera un nuevo archivo de texto. Para este ejemplo se hizo uso del procesador de textos libre **Sublime Text 2**, el cual puede ser inicializado desde terminal tal y como se observa en la figura B.1

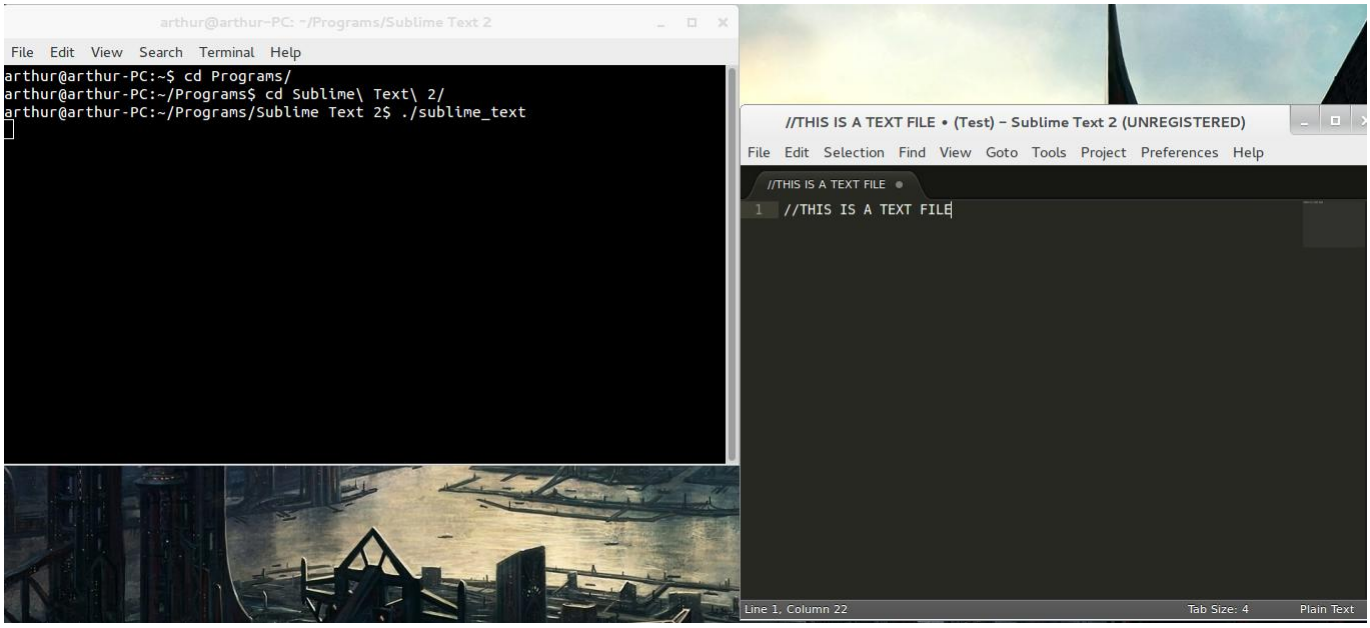


Figura B.1: Creación de un nuevo archivo de texto en Sublime Text 2.

2. Posteriormente se introduce el código que se observa en la figura B.4 y se guarda como Server.py.

3. Se inicializa en la terminal el servidor en Python utilizando el comando:

```
$ python Server.py --port=8080
```

   como se observa en la figura B.2.

4. Se abre una segunda terminal de consola y se teclea el comando:

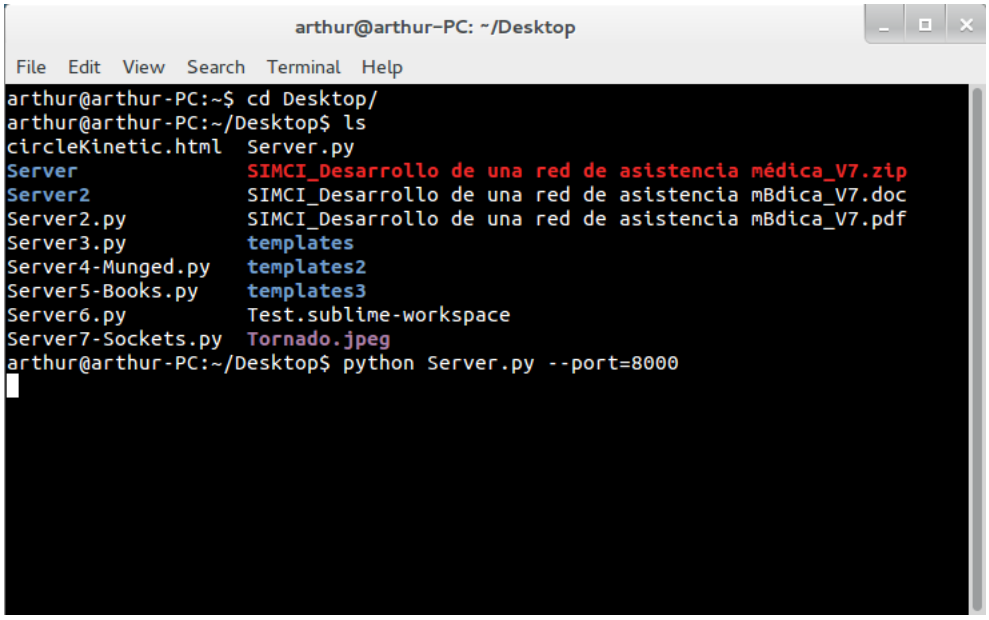


Figura B.2: Inicialización del servidor Tornado en consola.

```
1     $ curl http://localhost:8000/
```

y se generará una respuesta en la terminal por parte del servidor tal y como se observa en la figura B.3.

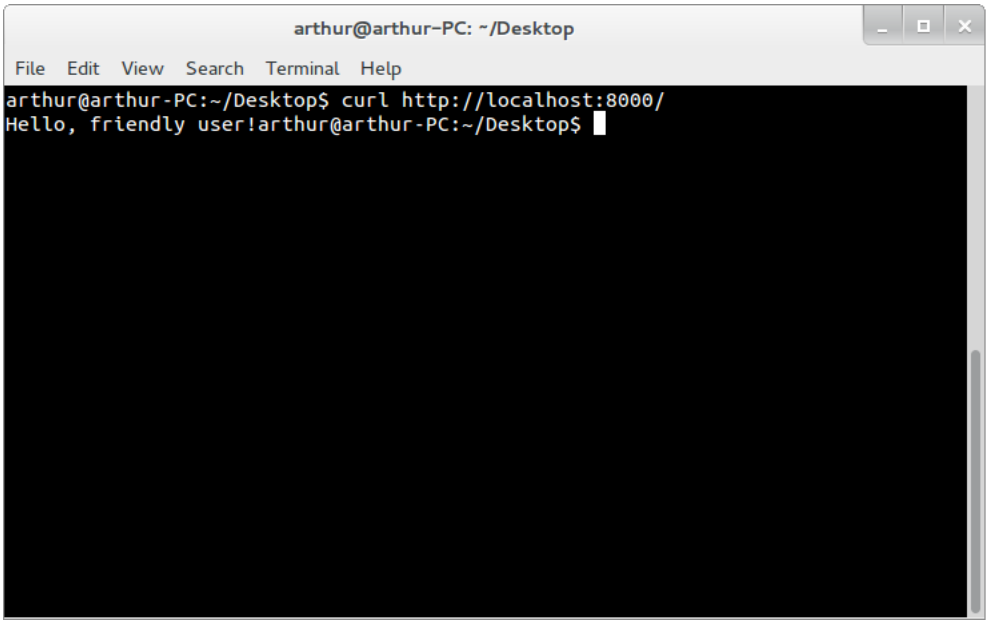
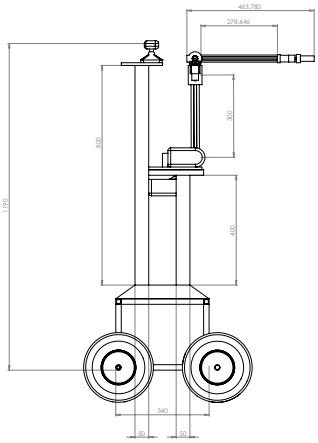


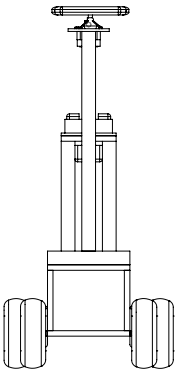
Figura B.3: Respuesta del servidor hacia el dispositivo cliente.

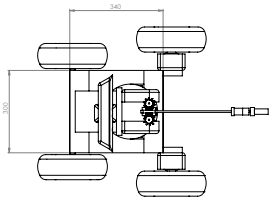
Figura B.4: Código empleado para realizar una comunicación cliente-servidor.

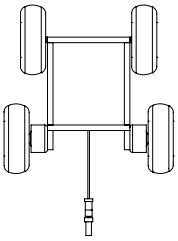
```
  import tornado.httpserver
import tornado.ioloop
import tornado.options
import tornado.web

from tornado.options import define, options
define("port", default=8000, help="run on the given port", type=int)
class IndexHandler(tornado.web.RequestHandler):
    def get(self):
        greeting = self.get_argument('greeting', 'Hello')
        self.write(greeting + ', friendly user!')

if __name__ == "__main__":
    tornado.options.parse_command_line()
    app = tornado.web.Application(handlers=[(r"/", IndexHandler)])
    http_server = tornado.httpserver.HTTPServer(app)
    http_server.listen(options.port)
    tornado.ioloop.IOLoop.instance().start()
```

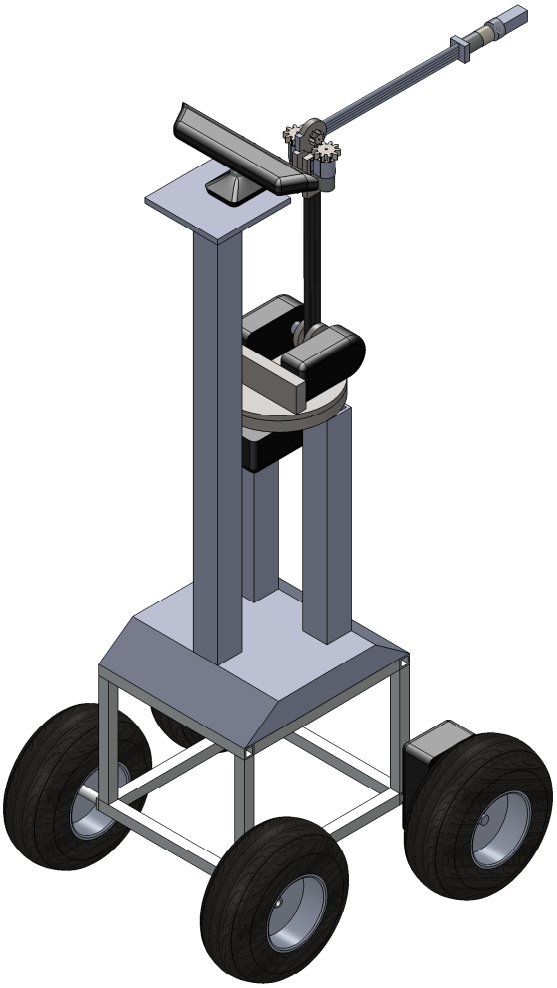
C. Elementos del sistema mecánico.











SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:

ACABADO:

REBARBAR Y ROMPER ARISTAS VIVAS

NO CAMBIE LA ESCALA

REVISIÓN

	NOMBRE	FIRMA	FECHA
DIBUJ.			
VERIF.			
APROB.			
FABR.			
CAL.			

TÍTULO:

MATERIAL:

N.º DE DIBUJO

SerbotIII-Concepto

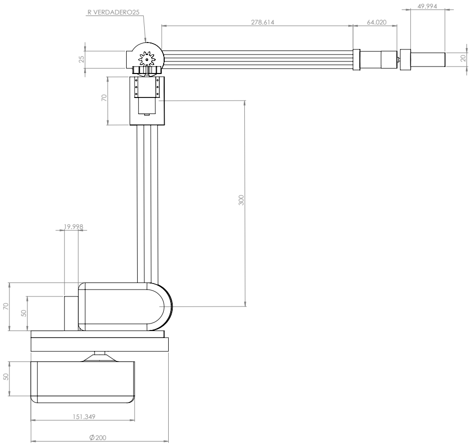
A0

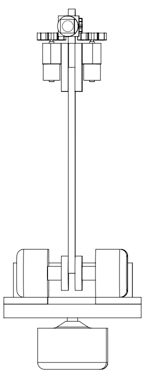
PESO:

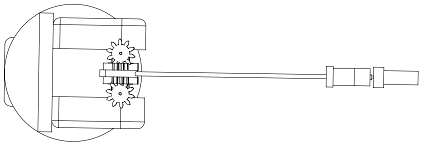
ESCALA:1:1

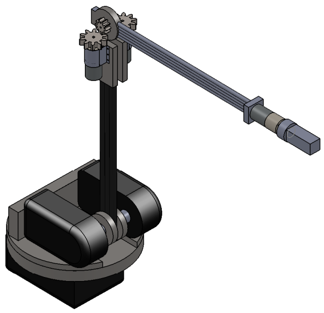
HOJA 1 DE 1

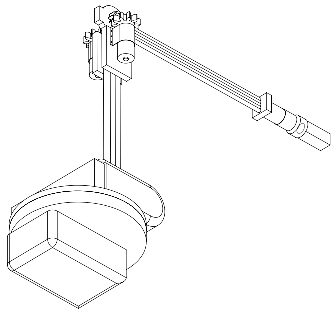
C.1 Brazo manipulador.

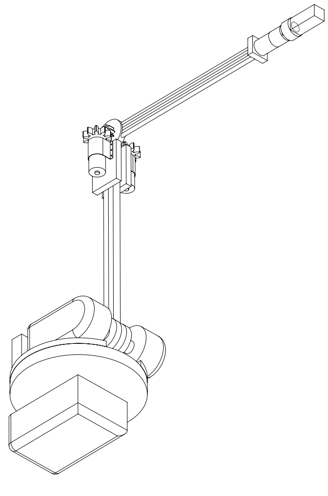












SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:

ACABADO:

REBARBAR Y ROMPER ARISTAS VIVAS

NO CAMBIE LA ESCALA

REVISIÓN

	NOMBRE	FIRMA	FECHA
DIBUJ.			
VERIF.			
APROB.			
FABR.			
CAL.			

TÍTULO:

MATERIAL:

N.º DE DIBUJO

DCR-Serbot_III-Brazo_manipulad

A0

PESO:

ESCALA:1:1

HOJA 1 DE 1

D. Instalación de la librería libfreenect.

Para la correcta instalación de la librería libfreenect se hizo uso de la compilación manual del código fuente a través de la página GitHub en la plataforma Ubuntu Linux versión 13.10 (Saucy Salamander).

D.1 Herramienta utilizada.

- Computadora personal Lenovo con procesador AMD E-300 APU 32 bits, 128 GB de disco duro y 1.6 GB de memoria RAM.
- Sistema operativo Ubuntu Linux 13.10 Saucy Salamander.
- Conexión a Internet.

D.2 Procedimiento [15].

- Instalar las librerías y componentes preliminares mediante el comando:

```
$ sudo apt-get install git-core cmake libglut3-dev pkg-config build-essential libxmu-dev
    libxi-dev libusb-1.0-0-dev
$sudo apt-get install git-core cmake freeglut3-dev pkg-config build-essential libxmu-dev
    libxi-dev libusb-1.0-0-dev
```

- Clonar la librería libfreenect en un directorio de instalación. En este caso se clonó en el directorio /Programs/

```
$ git clone git://github.com/OpenKinect/libfreenect.git
```

- Ingresar al directorio de la librería.

```
$ cd libfreenect
```

- Generar un directorio en el cual se configurará el código de la librería para el sistema e ingresar al directorio.

```
$ mkdir Build
$ cd Build
```

- Generar el código, configurar e instalar.

```
$ cmake ..
$ make
$ sudo make install
$ sudo ldconfig /usr/local/lib64/
```

- Para poder utilizar la librería sin tener que ingresar como administrador, añadir el usuario a los permisos de video y copiar el archivo de reglas de libfreeenct al directorio /etc/udev/rules.d.

```
$ sudo adduser $USUARIO video
$ sudo gedit /etc/udev/rules.d/51-kinect.rules
```

- Copiar el siguiente código y pegarlo en el archivo generado.

```
# ATTR{product}=="Xbox NUI Motor"
SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02b0
   ", MODE="0666"
# ATTR{product}=="Xbox NUI Audio"
SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02ad
   ", MODE="0666"
# ATTR{product}=="Xbox NUI Camera"
SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02ae
   ", MODE="0666"
```

- Reiniciar la sesión y ejecutar un ejemplo para ver si funciona correctamente.

```
bin/glview
```

E. Simulación del brazo robot.

Para el control del robot se hizo uso del método de control desacoplado en el cual cada una de las juntas del robot contiene un control independiente del resto y que se encarga de modular los movimientos de la junta sobre la cual actúa sin considerar las fuerzas y momentos del resto de los eslabones.

Entre las ventajas principales de este tipo de control se encuentra una mayor velocidad de procesamiento debido a que contiene un número menor de variables y puede considerarse un sistema lineal.

En la figura E.1 se presenta el controlador PD utilizado para regular los movimientos del robot en base a una entrada.

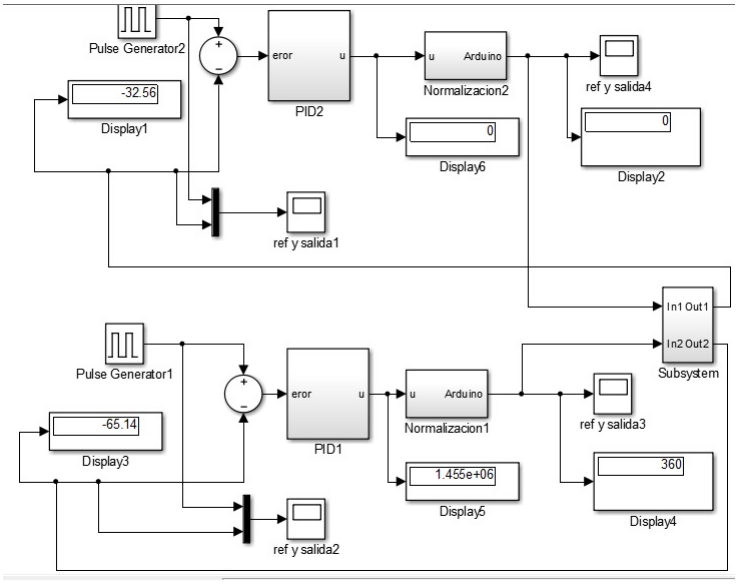


Figura E.1: Bloque de controlador PD.

Se elaboró el diseño del robot en SolidWorks y se exportó en formato .xml para poder acceder al ensamble en Matlab utilizando el complemento Simmechanics de Matlab. Mediante este complemento se simuló el comportamiento del robot utilizando el control PD que aparece en la figura E.1.

La estructura del robot puede observarse en la figura E.2.

Introduciendo una señal de referencia senoidal se obtiene la salida que puede observarse en la figura E.3.

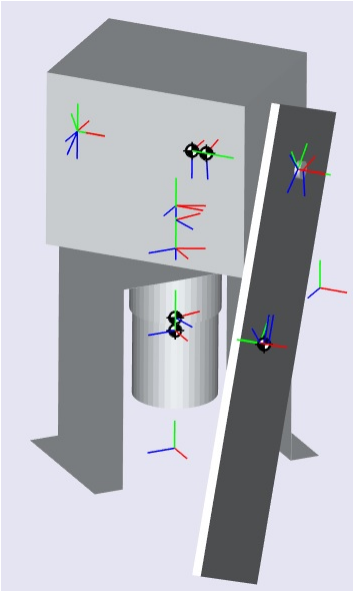


Figura E.2: Modelo del robot elaborado en SolidWorks.

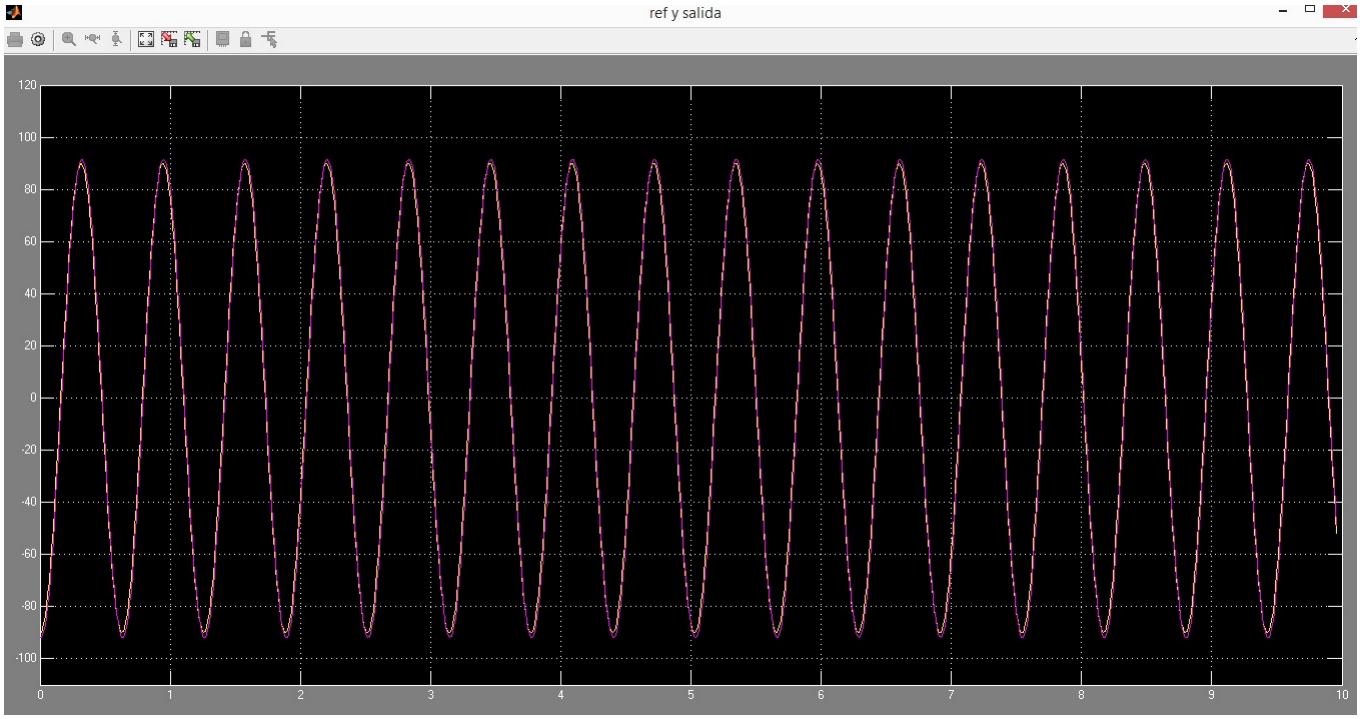


Figura E.3: Gráfica del comportamiento del sistema a una señal senoidal.

F. Control PID para robot físico.

Para el control físico del robot se utilizó de igual forma Simulink en conjunto con el wrapper para Arduino, a través del cual es posible obtener y emitir señales desde Matlab de acuerdo a los parámetros especificados.

Las constantes utilizadas fueron:

$$K_p = 3$$
$$K_i = 20$$
$$K_d = 0{,}09$$

Correspondientes a las constantes de proporcionalidad que se encarga de añadir más energía al sistema, la constante de integración y derivación encargadas de regular el sobretiro y tiempo de asentamiento del sistema respectivamente.

En la figura F.1 se puede observar el sistema encargado de controlar el manipulador haciendo uso de la librería Arduino para Matlab.

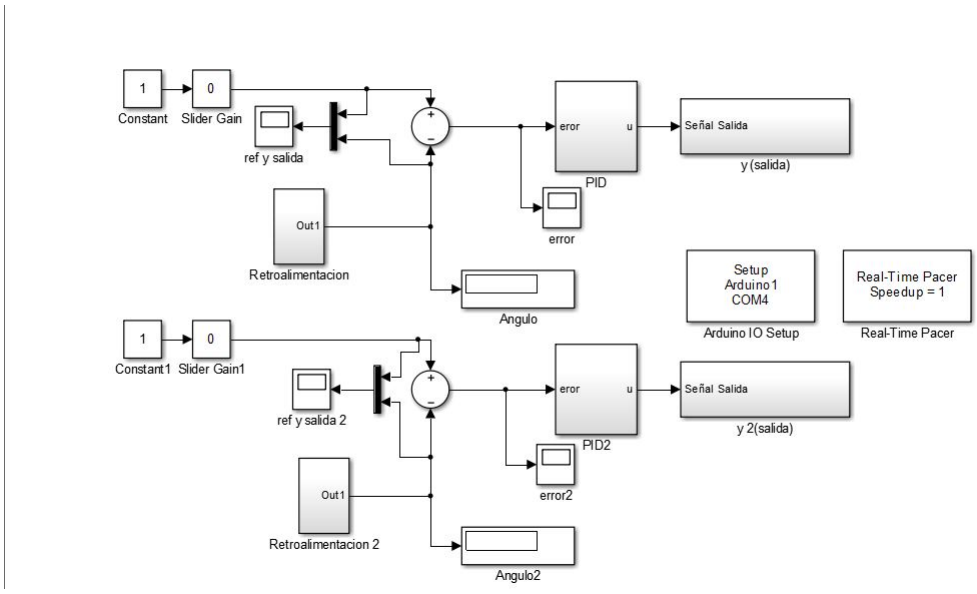


Figura F.1: Control de brazo manipulador en Matlab Simulink.

En la figura F.2 se muestra el control PID que se encuentra en el subsistema control PID.

En la figura F.3 se observa el acoplamiento del codificador óptico hacia el sistema mediante el cual es posible localizar la posición de la junta en cada momento determinado recibiendo pulsos.

Finalmente en la figura F.4 se ilustra la salida de Arduino.

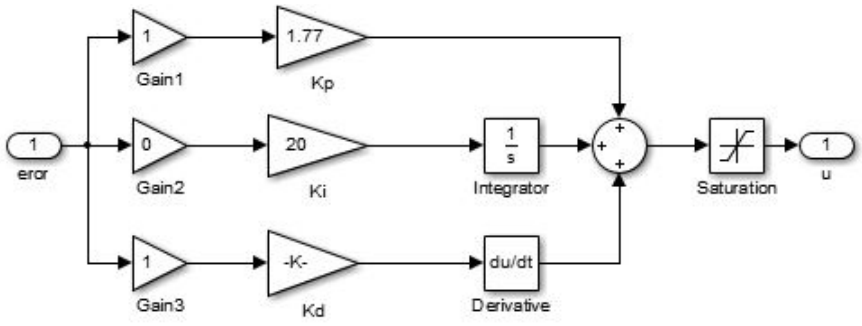


Figura F.2: Controlador PID.

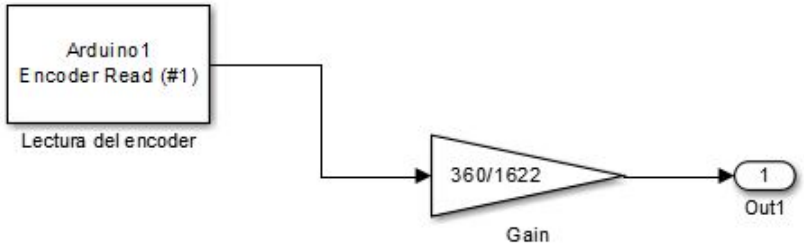


Figura F.3: Procesamiento de la señal del codificador óptico.

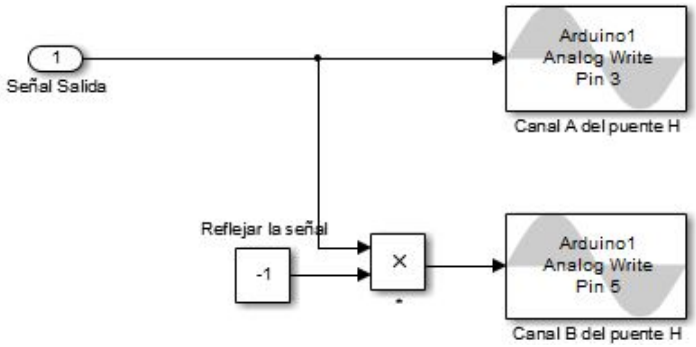


Figura F.4: Salida de Simulink hacia Arduino.

Bibliografía

[1] Markovito: Home Page. Visto por última vez: 25/octubre/2013.
http://ccc.inaoep.mx/~markovito/

[2] PAPERO.Visto por última vez: 25/octubre/2013.
http://www.ipapero.com/

[3] Sony AIBO. Visto por última vez: 25/octubre/2013.
http://www.sony-europe.com/support/aibo/4_0_support_breakdown.asp

[4] Seattle Robotics. Visto por última vez: 21/agosto/2013.
http://www.seattlerobotics.org/encoder/mar98/fuz/fl_part1.html#INTRODUCTION

[5] Society of Robots - Chasis construction. Visto por última vez: 20/septiembre/2013.
http://www.societyofrobots.com/mechanics_chassisconstruction.shtml

[6] Society of Robots - Statics. Visto por última vez: 20/septiembre/2013.
http://www.societyofrobots.com/mechanics_statics.shtml

[7] Society of Robots - Dynamics. Visto por última vez: 20/septiembre/2013.
http://www.societyofrobots.com/mechanics_dynamics.shtml

[8] Society of Robots - Arm design. Visto por última vez: 20/septiembre/2013.
http://www.societyofrobots.com/robot_arm_tutorial.shtml

[9] Shih, Randy H. *Learning SolidWorks 2013*. SDC Publications. 2012. Estados Unidos de Norteamérica. 486 páginas. ISBN: 978 - 1 - 58503 - 769 - 8.

[10] Python.org. Visto por última vez: 20/septiembre/2013.
http://www.python.org/

[11] Dory, Michael; Parrish, Adam; Berg, Brendan. *Introduction to Tornado.* Editorial O'Reilly. Primera edición. 2012. Estados Unidos de Norteamérica. 136 páginas. ISBN: 978 - 1 - 449 - 30907 - 7.

[12] Kinect for Windows. Visto por última vez: 2 de noviembre del 2013.
http://www.microsoft.com/en-us/kinectforwindows/

[13] Kinect for Windows sensor components and specifications. Visto por última vez: 2 de noviembre del 2013.
http://msdn.microsoft.com/en-us/library/jj131033.aspx

[14] OpenKinect. The history. Visto por última vez: 3 de noviembre del 2013.
http://openkinect.org/wiki/History

[15] OpenKinect. Getting Started. Visto por última vez: 3 de noviembre del 2013.
http://openkinect.org/wiki/Getting_Started

[16] KineticJS. Visto por última vez: 18/septiembre/2013.
http://kineticjs.com/

[17] HTML5 Canvas Tutorials. Visto por última vez: 18/septiembre/2013.
http://www.html5canvastutorials.com/kineticjs/html5-canvas-kineticjs-circle-tutorial/

[18] Arduino Mega - Schematics. Visto por última vez: 21/agosto/2013.
http://arduino.cc/en/Main/arduinoBoardMega2560

[19] W. Spong, Mark; Vidyasagar, M. *Robot Dynamics and Control.* Editorial John Wiley and Sons Inc. Primera edición. Estados Unidos de Norteamérica. ISBN: 0-471-61243-X.

[20] Craig, John J. *ROBÓTICA.* PEARSON EDUCACIÓN. México, 2006. ISBN: 970-26-0772-8. Páginas: 408.