UNIVERSIDAD POLITÉCNICA DE LA ZONA METROPOLITANA DE GUADALAJARA

Jesús Jail Avalos Lupercio

Raúl Israel García Barajas

Ricardo Martínez Jacinto

Josué Adrián Moreno Martínez

Héctor Alejandro Nolasco Casillas

Rodrigo Rubio García

Juan Pablo Salguero Hernández

Brazo Antropomórfico

Tlajomulco de Zúñiga, Jalisco

Agosto de 2019

Índice

Contenido

Índice

Figuras

Índice

Tablas

Introducción

El ser humano siempre ha buscado la forma de facilitar sus tareas, en un principio se comenzó transportando cargas fácilmente gracias a la invención de la rueda, momento en que se inició una imparable evolución de la técnica y la ingeniería.

A medida que la ingeniería se ha desarrollado, se ha ido pasando progresivamente de diseñar herramientas utilizadas por un humano, a buscar máquinas con cualidades antropomórficas que realicen tareas cada vez más complejas, y, a pesar de que ya existen robots diseñados para realizar una gran variedad de tareas, no debemos pensar que un robot flexible podrá llevar a cabo cualquier tarea.

Por este motivo deben reconocerse las tareas propias de cada aplicación y realizar el diseño del robot pensando en la configuración mecánica más apropiada para esas tareas concretas, y cuando se busca la implantación de un robot o manipulador para una aplicación concreta, es común buscar un robot que posea un área accesible que contenga la trayectoria deseada.

Además, posteriormente, se debe programar la trayectoria del actuador del robot de forma que se adecue a los requerimientos, ya que esto implica utilizar solo una pequeña parte del potencial del robot y además se estaría utilizando un robot que tendría un peso, tamaño, precio y coste de programación que no son necesarios para esa aplicación.

Planteamiento del problema

En una empresa normalmente, las líneas de producción son operadas por trabajadores humanos, lo que conlleva a que se existan perdidas tanto de tiempo como de producto, siendo operadores humanos, el riesgo de imprecisión y los posibles errores por distracciones son más comunes.

Un brazo robótico, puede ser una herramienta perfecta dado a que reduce tiempos, riesgos físicos en maniobras de gran esfuerzo o peligrosas y también elimina los accidentes por distracciones.

Justificación

En este proyecto se propone diseñar un mecanismo lo más simple posible para realizar una tarea específica. De esta forma su tamaño y peso serían contenidos, además tendría menor cantidad de sensores y actuadores.

Un diseño mecánico eficiente, desembocaría en un menor coste del producto y una menor necesidad de control y programación, se tendría una mayor robustez de los elementos, por lo que es previsible obtener una mayor fiabilidad y coste de mantenimiento, además de permitir mayores productividades debido a la capacidad de trabajar con mayores aceleraciones., y tendríamos como la última consecuencia de este tipo de diseño, sería un menor tiempo de fabricación, debido a su mayor simplicidad.

Meta

Diseñar un robot antropomórfico de 3GDL (Grados de libertad) que sea capaz de realizar movimientos en base a datos establecidos vía ROS en PC.

Objetivos

El presente proyecto tiene como finalidad realizar un diseño de un brazo mecánico dando una gran importancia a la mejora del diseño mecánico, y dicho brazo mecánico debe tener únicamente un grado de libertad y debe poder trasladar cargas de 300gr de peso.

Se requiere que el diseño sea lo más simple y barato posible.

Para este diseño debe realizarse una optimización de modo que se asegure que la configuración sea eficiente con respecto a varios criterios diferentes, este proyecto requerirá que se obtengan una variedad de configuraciones diferentes que posean un grado de libertad, y que una vez obtenidas dichas configuraciones se deberá realizar una comparación que conduzca a elegir una de ellas de cara al posterior proceso de optimización.

1. Brazo Antropomórfico

Un brazo antropomórfico tiene sus tres principales articulaciones de tipo R, (y también las restantes), con lo cual emplea las coordenadas angulares para determinar las posiciones de su elemento terminal.

Se llama antropomórfico por que simula los movimientos de un brazo humano, el primer eje se corresponde con el cuerpo, el segundo con el brazo, el tercero con el antebrazo y el resto de con la muñeca-mano; la primera articulación se corresponde con el giro de la cintura, la segunda con el del hombro, la tercera con el del codo y el resto están en la muñeca.

Este robot posee gran accesibilidad y maniobrabilidad, es rápido y ocupa poco espacio en relación al campo de trabajo que abarca.

Debido a sus características es el modelo más versátil en aplicaciones y se ha impuesto a los demás, sobre todo en Células de Fabricación Flexible.

Como inconvenientes se pueden citar que tiene menos precisión que otros modelos, que, si trabaja con carga y velocidades altas, se producen inercias de giro difíciles de compensar y que sus articulaciones deben tener juego casi nulo, pues un pequeño juego angular se amplifica en posición en función de la longitud del eje correspondiente, con lo cual puede dar errores considerables.

Figura 1.-Robot antropomórfico

2. Cinemática del robot

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia sin considerar las fuerzas que intervienen. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares.

Denavit y Hartenberg

Denavit y Hartenberg propusieron un método sistemático para describir y representar la geometría espacial de los elementos de una cadena cinemática, y en particular de un robot, con respecto a un sistema de referencia fijo.

Este método utiliza una matriz de transformación homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciéndose el problema cinemático directo a encontrar una matriz de transformación homogénea 4×4 que relacione la localización espacial del extremo del robot con respecto al sistema de coordenadas de su base

Los cuatro parámetros de D-H (θi , di , ai , $\propto i$) dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y siguiente.

Θi. Es el ángulo que forman los ejes $xi–1$ y xi medido en un plano perpendicular al eje $zi–1$, utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.

di. Es la distancia a lo largo del eje $zi–1$ desde el origen del sistema de coordenadas (i–1)-ésimo hasta la intersección del eje $zi–1$ con el eje xi. Se trata de un parámetro variable en articulaciones prismáticas.

ai. Es la distancia a lo largo del eje xi que va desde la intersección del eje $zi–1$ con el eje xi hasta el origen del sistema i-ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes $zi–1$ y zi.

∝i. Es el ángulo de separación del eje zi–1 y el eje zi, medido en un plano perpendicular al eje xi, utilizando la regla de la mano derecha.

Articulación	$\boldsymbol{a_{i-1}}$	$\boldsymbol{\propto_{i-1}}$	$\boldsymbol{di}$	$\boldsymbol{\theta i}$
1	0	0	0	$\theta 1$
2	L1	90	0	$\theta 2$
3	L2	0	0	$\theta 3$

Tabla 1.-Parametros Denavit Hartemberg

Cálculo de la matriz Homogénea

$$\boldsymbol{T_i^{i-1}}\begin{bmatrix} \boldsymbol{C}\theta i & -\boldsymbol{S}\theta i & \mathbf{0} & a_{i-1} \\ \boldsymbol{S}\theta i C \propto_{i-1} & \boldsymbol{C}\theta i C \propto_{i-1} & -\boldsymbol{S} \propto_{i-1} & -\boldsymbol{di S} \propto_{i-1} \\ \boldsymbol{S}\theta i S \propto_{i-1} & \boldsymbol{C}\theta i S \propto_{i-1} & C \propto_{i-1} & \boldsymbol{di} C \propto_{i-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \textbf{...(ec1)}$$

$$\boldsymbol{T_1^0}\begin{bmatrix} \boldsymbol{C}\theta 1 & -\boldsymbol{S}\theta 1 & \mathbf{0} & 0 \\ \boldsymbol{S}\theta 1 & \boldsymbol{C}\theta 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \textbf{...(eq2)}$$

$$\boldsymbol{T_2^1}\begin{bmatrix} \boldsymbol{C}\theta 2 & -\boldsymbol{S}\theta 2 & \mathbf{0} & L1 \\ \mathbf{0} & \mathbf{0} & -\mathbf{1} & \mathbf{0} \\ \boldsymbol{S}\theta 2 & \boldsymbol{C}\theta 2 & 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \textbf{...(eq3)}$$

$$\boldsymbol{T_3^2}\begin{bmatrix} \boldsymbol{C}\theta 3 & -\boldsymbol{S}\theta 3 & \mathbf{0} & L2 \\ \boldsymbol{S}\theta 3 & \boldsymbol{C}\theta 3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \textbf{...(eq4)}$$

$$\boldsymbol{T_3^0}\begin{bmatrix} \boldsymbol{C}\theta 1\boldsymbol{C}\theta 2\boldsymbol{C}\theta 3 - \boldsymbol{C}\theta 3\boldsymbol{S}\theta 1\boldsymbol{S}\theta 2 & -\boldsymbol{C}\theta 1\boldsymbol{C}\theta 3\boldsymbol{S}\theta 2 - \boldsymbol{C}\theta 2\boldsymbol{C}\theta 3\boldsymbol{S}\theta 1 & \boldsymbol{S}\theta 3 & L2 + \boldsymbol{C}\theta 3\boldsymbol{L}1 \\ \boldsymbol{C}\theta 1\boldsymbol{C}\theta 2\boldsymbol{S}\theta 3 - \boldsymbol{S}\theta 1\boldsymbol{S}\theta 2\boldsymbol{S}\theta 3 & -\boldsymbol{C}\theta 1\boldsymbol{S}\theta 2\boldsymbol{S}\theta 3 - \boldsymbol{C}\theta 2\boldsymbol{S}\theta 1\boldsymbol{S}\theta 3 & -\boldsymbol{C}\theta 3 & \boldsymbol{L1}\boldsymbol{S}\theta 3 \\ \boldsymbol{C}\theta 1\boldsymbol{S}\theta 2 + \boldsymbol{C}\theta 2\boldsymbol{S}\theta 1 & \boldsymbol{C}\theta 1\boldsymbol{C}\theta 2 - \boldsymbol{S}\theta 1\boldsymbol{S}\theta 2 & 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

...(eq5)

3.Modelo y simulación del brazo robótico

Después de las modificaciones realizadas en el diseño de SolidWorks, se exporta el archivo a ANSYS para posteriormente realizarle un análisis estructural estático.

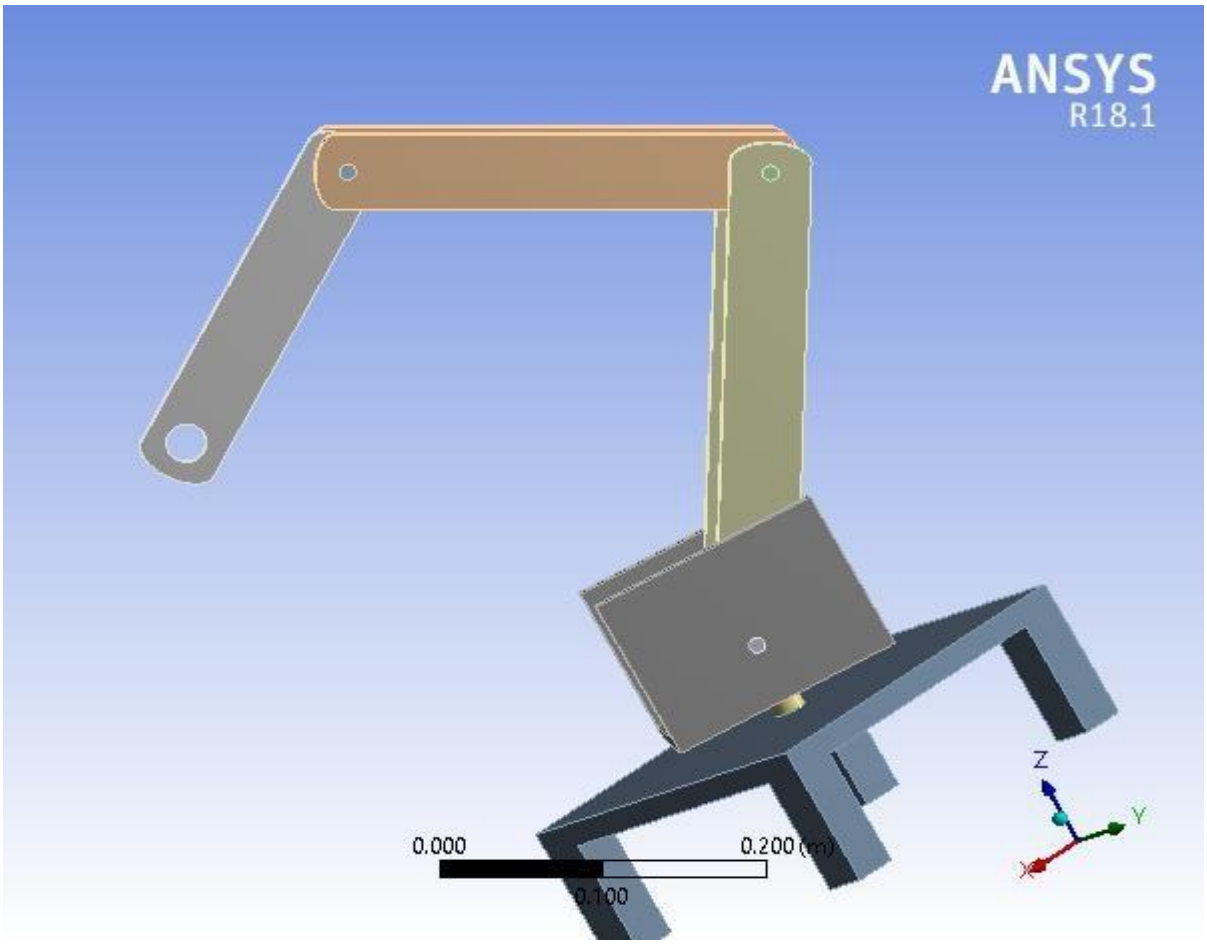
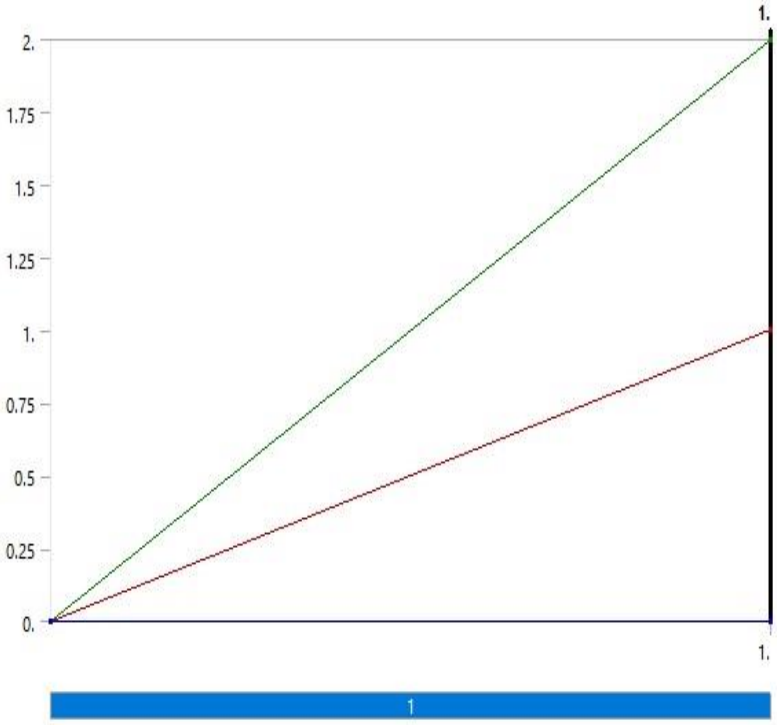


Figura 2-Estructura inicial del robot

En la gráfica 1 se aprecia la referencia a la cantidad de fuerza aplicada en cada componente.



Gráfica 1

En la tabla 4 se aprecian los resultados de la deformación, los cuales no son adecuados para ser utilizados con motor de corriente directa.

Mínimo [m]	Máximo [m]
0.	1.1029e-004

Tabla 2.-Resultados de la deformación total máxima (Máximos y mínimos)

En la figura 3 las áreas de color azul son las que no sufren una deformación y las rojas son las que tienen más posibilidad de sufrirla. En este caso la deformación total es baja, siendo aceptable el diseño.

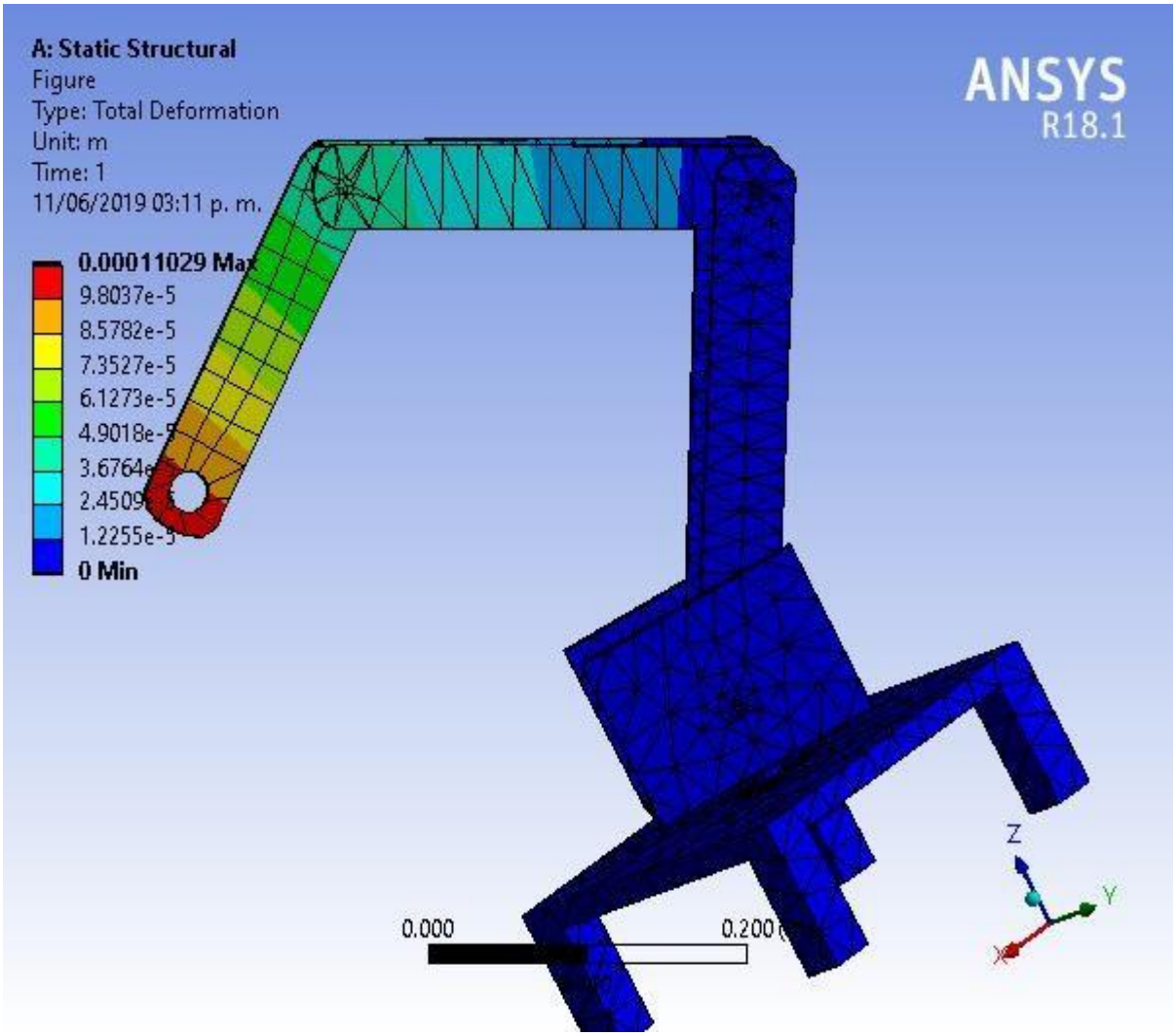


Figura 3- Resultado de Análisis de deformación

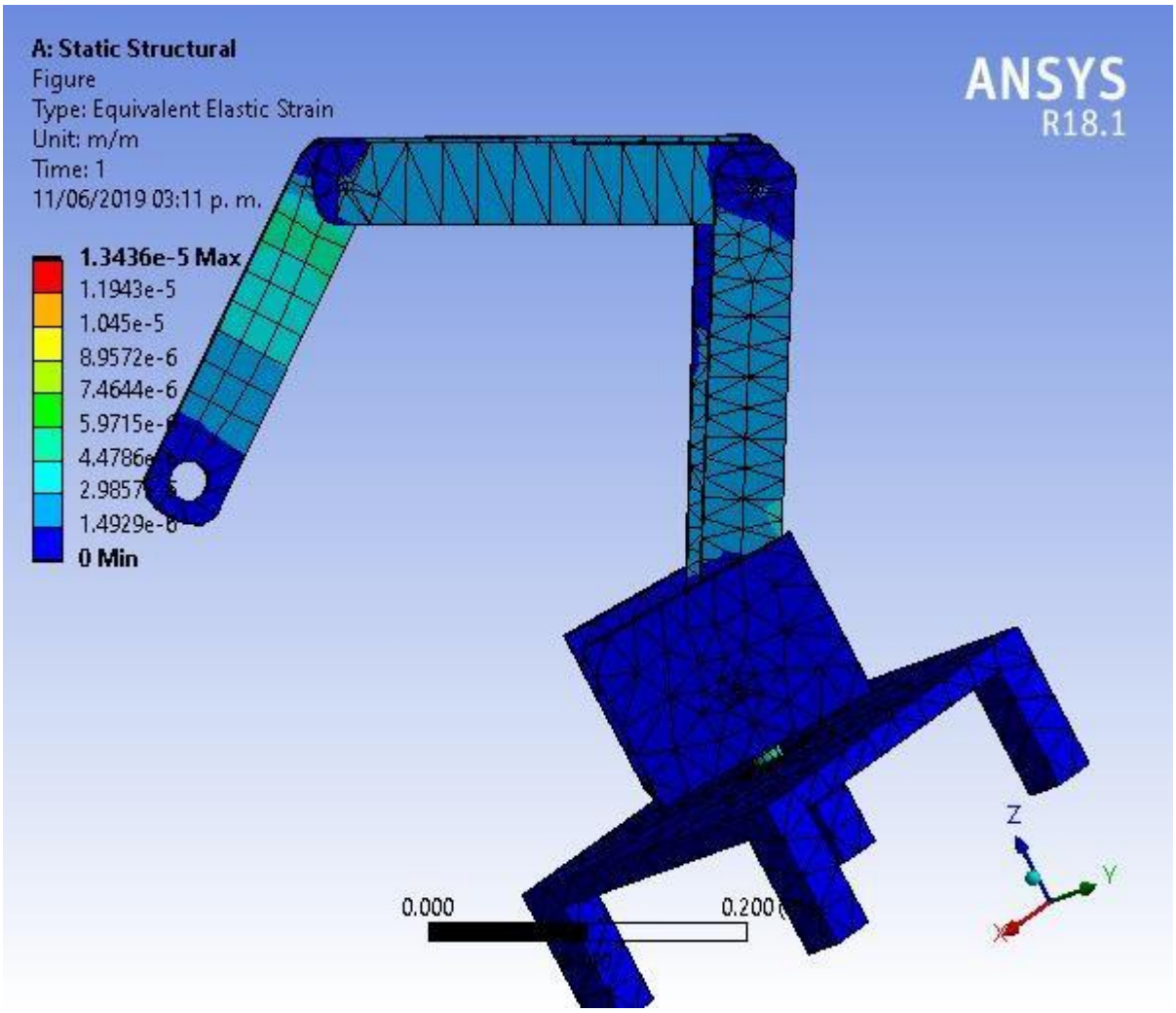


Figura 4-Resultado del Análisis de esfuerzos elásticos

4.Funcionamiento de Drivers A4988 y DRV8825 y motor a pasos

Drivers A4988 y DRV8825

El A4988 y el DRV8825 son controladores (drivers) que simplifican el manejo de motores paso a paso desde un autómata o procesador como Arduino.

Estos controladores nos permiten manejar los altos voltajes e intensidades que requieren estos motores, limitar la corriente que circula por el motor, y proporcionan las protecciones para evitar que la electrónica pueda resultar dañada.

Para su control únicamente requieren dos salidas digitales, una para indicar el sentido de giro y otra para comunicar que queremos que el motor avance un paso. Además, permiten realizar microstepping, una técnica para conseguir precisiones superiores al paso nominal del motor.

El A4988 ha alcanzado una gran popularidad en sus últimos tiempos debido a su uso en proyectos como, por ejemplo, en impresoras 3D caseras. Por su parte el DRV8825 es una versión mejorada del A4988 y, por tanto, tiene unas características ligeramente superiores.

En particular, el DRV8825 permite trabajar con tensiones superiores al A4988 (45V frente a 35V), e intensidades superiores (2.5A frente a 2A). Además, añade un nuevo modo de microstepping (1/32) que no está presente en el A4988.

Por lo demás, aparte de unas pequeñas diferencias, ambos dispositivos son similares en su montaje y uso. Incluso pueden ser compatibles entre sí, es decir, bajo ciertas condiciones podemos usar indistintamente uno u otro, e incluso una combinación de ambos de forma simultánea.

Modelo	A4988	DRV8825
Color	Verde o Rojo	Morado
Intensidad máxima	2A	2.5A
Tensión máxima	35V	45A
Microsteps	16	32
Rs típico	0.05, 0.1 o 0.2	0.1
Fórmulas	I_max = Vref / (8 * Rs)	I_max = Vref / (5 * Rs)
	Vref = I_max * 8 * Rs	Vref = I_max * 5 * Rs

Tabla 3.-Especificaciones de los drivers A4988 y DRV8825

Ambos controladores pueden alcanzar altas temperaturas durante su funcionamiento y es necesario disipar el calor para que el dispositivo no se dañe. Para intensidades superiores 1A en el A4988 y a 1.5A en el DRV8825 es necesario añadir un sistema de disipación de calor, e incluso ventilación forzada.

Disponen de protecciones contra sobreintensidad, cortocircuito, sobretensión y sobretemperatura. En general, son dispositivos robustos y fiables siempre que realicemos la conexión correctamente, e incorporemos disipación de calor si es necesario.

El A4988 y el DRV8825 son muy empleados en una gran variedad de proyectos que requieren el uso de motores paso a paso, como máquinas de CNC, plotters, robots que dibujan, impresoras 3D, y escáneres 3D.

También son un componente frecuente en proyectos para controlar robots y vehículos, especialmente en aquellos que requieren variar de forma individual la velocidad de cada rueda, como en vehículos con omniwheel o mecannum wheels.

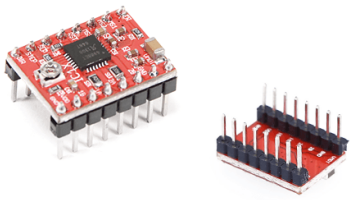


Figura 5. Driver A4988

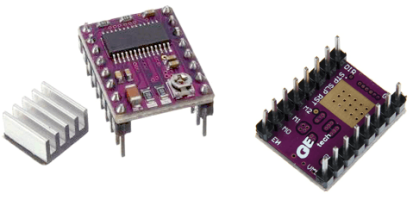


Figura 6 Driver DRV8825

Funcionamiento A4988 y DRV8825

Como en la mayoría de los controladores de motores el componente fundamental es un puente-H. En el caso del A4988 y DRV8825, destinados a controlar motores paso a paso, se dispone de dos puentes-H (uno por canal) constituidos por transistores MOSFET.

Sin embargo, a diferencia de controladores más simples como el L298N o el TB6612FNG, que presenta una electrónica relativamente simple, el A4988 y el DRV8825 tienen una electrónica considerablemente más compleja.

A modo de ejemplo, la siguiente imagen tenemos el esquema del DRV8255 y, cómo vemos, los puentes-H (remarcados en azul) representan una parte muy pequeña del conjunto.

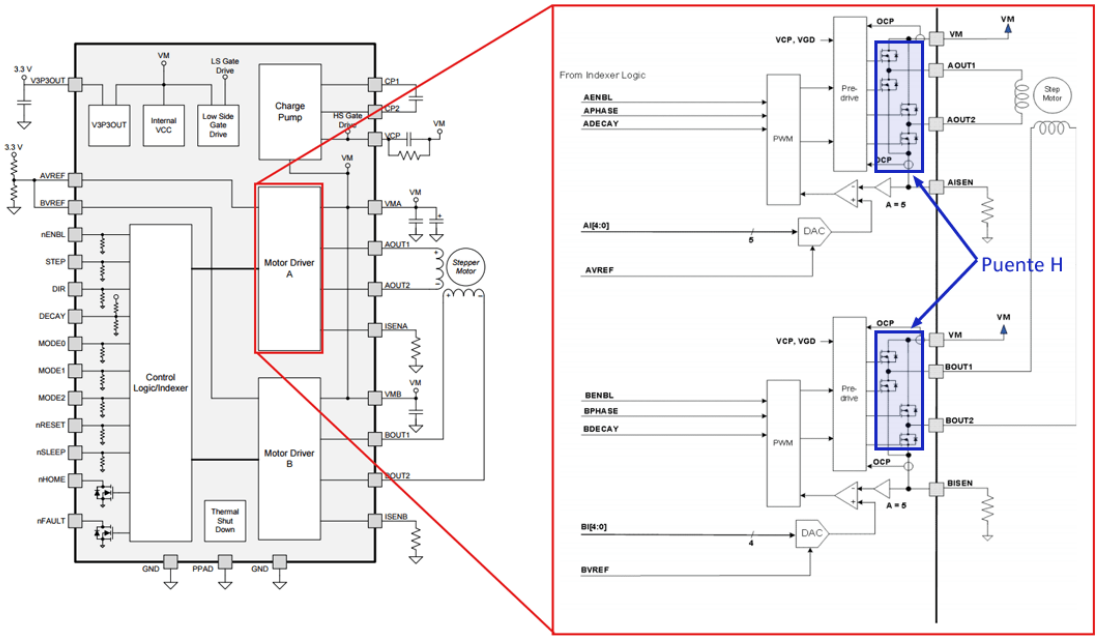


Figura 7. Esquemático del driver DRV8825

Uno de los motivos para esta complejidad es que únicamente requieren dos señales digitales de control para hacer funcionar el motor, y que incorporan las protecciones necesarias para su manejo.

El otro motivo de la complejidad de su electrónica es que incorporan funciones especialmente diseñadas para el control de motores paso a paso, como son el regulador de intensidad y el Microstepping. Veremos estas funcionalidades a continuación.

Regulación de intensidad (chopping)

Ambos controladores disponen de reguladores de intensidad incorporados. El motivo es que los motores paso a paso de cierto tamaño y potencia, como por ejemplo los NEMA 17 o NEMA 23, necesitan tensiones superiores a las que podrían soportar las bobinas por su corriente nominal.

Por ejemplo, supongamos que tenemos un motor NEMA 17 con 1.2A de intensidad nominal y 1.5 Ohm de resistencia por fase. Según la ley de Ohm deberíamos aplicar 1.8V a cada bobina para que circule la intensidad nominal de 1.2A. Sin embargo, con esa tensión el motor ni se movería.

Para el que el motor funcione correctamente necesitamos aplicar una tensión superior. En este ejemplo, la tensión nominal podría ser 12V. Pero si aplicáramos 12V directamente, nuevamente por la ley de Ohm, pasarían 8A por la bobina, lo cual destruiría el motor en poco tiempo.

Por este motivo, los controladores incorporan un limitador de intensidad, que permiten alimentar el motor a tensiones nominales superiores a las que es posible por su resistencia e intensidad máxima admisible.

Por supuesto, la ley de Ohmn debe cumplirse en todo momento por lo que, continuando con nuestro ejemplo, cuando alimentemos el motor a 12 V por la bobina pasará inevitablemente 8 A.

El limitador interrumpe la señal proporcionando una señal pulsada PWM de forma que el valor promedio de la intensidad que atraviesa la bobina es la intensidad nominal del motor. Terminando nuestro ejemplo, el limitador de tensión aplicaría el pulso durante el 15% del tiempo y mantendrá el motor apagado el 85% restante.

A este mecanismo de limitación de intensidad se le denomina Chopping.

Para regular la intensidad que proporcionara el limitador y ajustarlo al valor del motor que vayamos a emplear ambas placas disponen de un potenciómetro que regula la intensidad del limitador.

Una forma de estimar la intensidad del regulador es medir la tensión (Vref) entre el potenciómetro y GND y aplicar una fórmula que depende del modelo, que encontraréis en la tabla del principio de la entrada.

Estas fórmulas dependen el valor Rs de las resistencias ubicadas en la placa que pueden variar en función del fabricante. Los valores típicos también aparecen en la tabla, pero debéis comprobar el valor de las resistencias que monta vuestra placa.

Por ejemplo, con los valores de Rs habituales las fórmulas se reducen a:

Modelo	Rs	Fórmula reducida
A4988	50	I_max = 0,625 * V_ref
A4988	100	I_max = 1,25 * V_ref
A4988	200	I_max = 2 ,2* V_ref
DRV8825	100	I_max = 2 * V_ref

Tabla 4.-Corriente máximas y mínimas de los drivers A4988 y DRV8825

No obstante, el valor obtenido mediante esta medición es sólo una aproximación y puede ser inexacto, por lo que lo emplearemos sólo como una calibración inicial, y terminaremos el ajuste fino midiendo la corriente real que proporciona el controlador al motor mediante un amperímetro.

Microstepping

Como hemos dicho, el microstepping es una técnica que permite obtener pasos inferiores al paso nominal del motor paso a paso que vamos a controlar.

Cuando vimos las secuencias típicas de encendido de un paso a paso, vimos que aplicando un control todo a nada a las bobinas teníamos varias posibles combinaciones, de las cuales vimos las tres más habituales (1-fase, 2-fases, media-fase). Pero nadie ha dicho que tengamos que encender o apagar por completo las bobinas.

El microstepping hace variar la corriente aplicada a cada bobina emulando un valor analógico. Si pudiéramos a ambas bobinas dos señales eléctricas senoidal perfecta desfasadas 90º conseguiríamos un campo magnético rotatorio perfecto en el interior del motor.

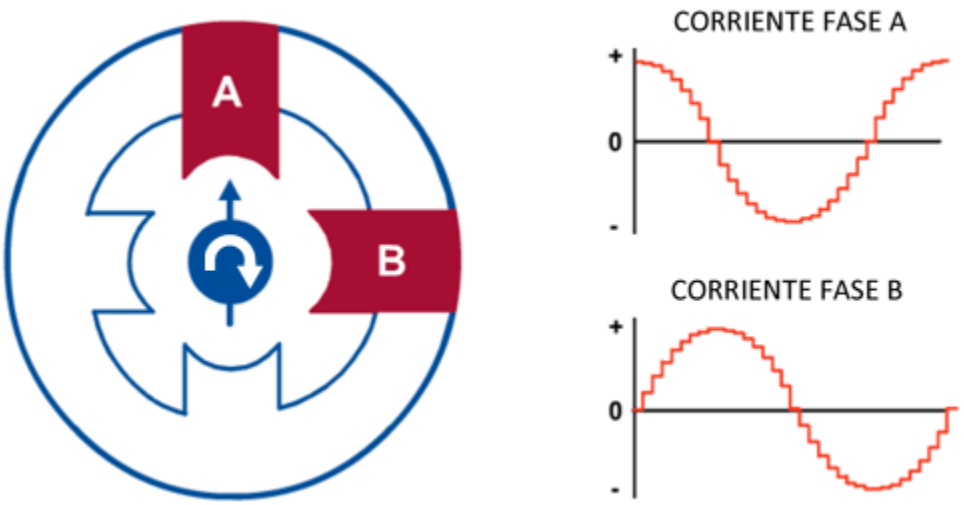


Figura 8-Funcionamiento de los pasos del motor a pasos tradicional

Por supuesto el controlador digital no genera valores analógicos perfectos, si no valores discretizados («a saltos»), por lo que la señal eléctrica que aplica es igualmente una función senoidal discretizada.

El resultado es un campo magnético un campo magnético rotativo con un paso inferior al paso nominal, que depende del número de niveles de discretos que podemos emplear en las señales de excitación de la bobina.

Cuando funcionamos sin microstepping (Modo full step), los controladores aplican una secuencia de 2-fases, por lo que aplican de forma permanente el 71% de la corriente del limitador a cada bobina. Únicamente varían el sentido en el que la corriente circula por la bobina.

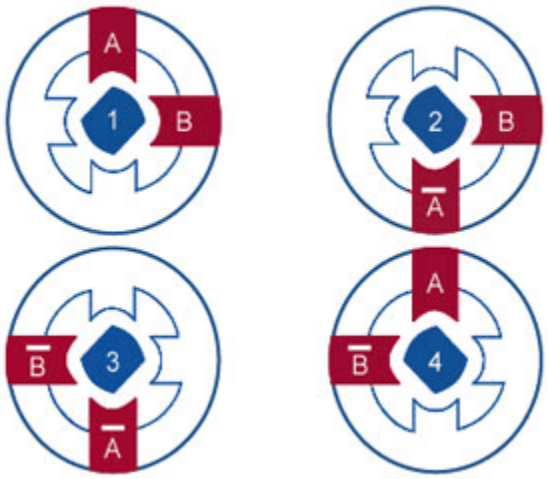


Figura 9-Funcionamiento de los pasos en un motor a pasos

Sin embargo, si aplicamos microstepping en cualquier de sus modos de funcionamiento, el controlador llega a aplicar el 100% de la corriente a una de las bobinas en un determinado paso. La cantidad de corriente concreta aplicada a cada bobina varía con cada paso.

Por ejemplo, con resolución 1/4 de paso, en el primer paso de la secuencia tendremos una bobina al 100% y la bobina B al 0%, en el segundo paso la bobina A al 92% y la bobina B al 38%, en el tercer paso la bobina A al 71%, y así sucesivamente.

El hecho de que sin microstepping la corriente aplicada es siempre el 71%, y si aplicamos microstepping (en cualquier resolución) la corriente aplicada llegará a ser del 100% en algún paso, es muy importante a la hora de calibrar la intensidad que circula por las bobinas.

Finalmente, la resolución con la que queremos que funcione el controlador se controla aplicando tensión a los Pines M0, M1 y M2. Estos pines están puestos a tierra mediante resistencias de Pull-Up, por lo que si no conectamos nada estarán a Low, y sólo deberemos forzar los pines en High.

Resolución		Pines M0, M1 y M2		
A4988	DRV8825	MODE0	MODE1	MODE2
Full step	Full step	Low	Low	Low
1/2 step	1/2 step	High	Low	Low
1/4 step	1/4 step	Low	High	Low
1/8 step	1/8 step	High	High	Low
–	1/16 step	Low	Low	High
–	1/32 step	High	Low	High
–	1/32 step	Low	High	High
1/16 step	1/32 step	High	High	High

Tabla 5.-Funcionamiento de los drivers A4988 y DRV8825

Dejar todos los pines desconectados dará lugar a usar modo Full Step

Esquema de montaje

El esquema de conexión de ambos controladores es muy similar. Incluso, como hemos dicho, bajo ciertas consideraciones ambos dispositivos son compatibles entre sí.

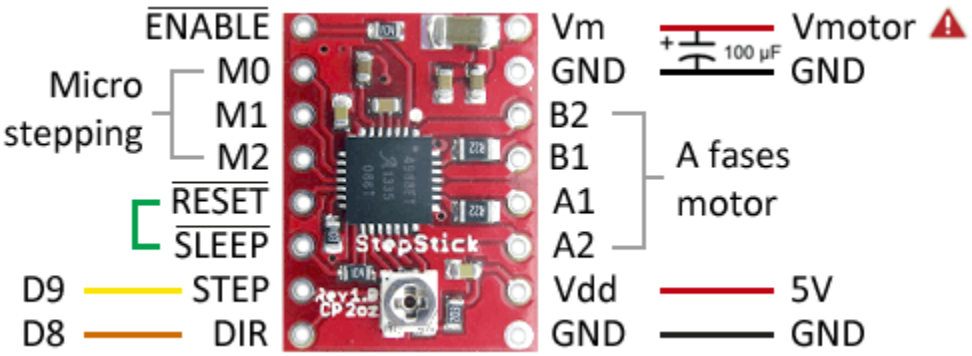


Figura 10-Esquema de controlador A4988

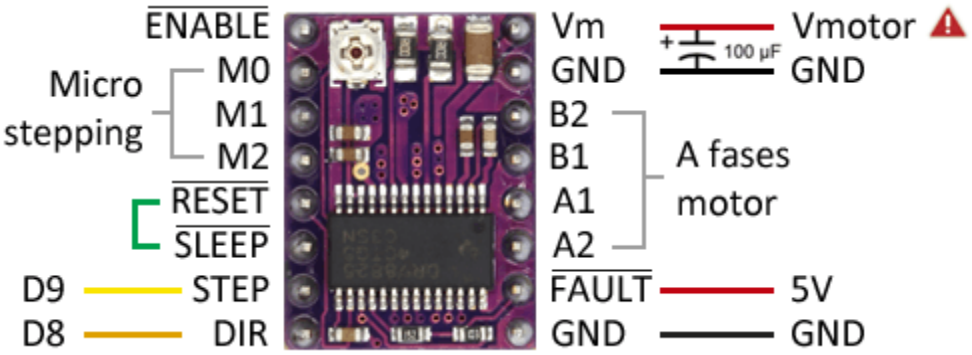


Figura 11-Esquema de controlador DRV8825

5.Componentes Mecánicos

Motor a pasos nema 17

Figura 12. Motor NEMA 17

Este motor paso a paso NEMA 17 es bipolar, tiene un ángulo de paso de 1.8° (200 pasos por vuelta) y cada bobinado es de 1.7A a 12V, capaz de cargar con 4Kg/cm.

Es un motor muy robusto ampliamente utilizando en Impresoras 3D o maquinas CNC de propósito general, aunque este es ligeramente más potente del recomendado para garantizar la mejor fiabilidad. El bastidor tiene una parte plana para asegurar un mejor ajuste.

- Tipo: Bipolar.
- Voltaje: 12V.
- Corriente: 1.7A

- Torque: 4000g/cm.
- Modelo: 17HS4401, 17HS series tamaño 42mm.
- Angulo de pasos de 1.8 Grados.
- No. Pasos: 200.
- No. de Cables: 4.
- Resistencia de la fase: 1.5 ohms.
- Inductancia de la bobina: 2.8mH.
- Eje de 5mm Diámetro / 20 mm de Largo.
- Conexiones: AZUL: B- / VERDE: A- ROJO: A+ / AMARILLO: B+

Motor NEMA 23

Figura 13. Motor NEMA 23

Los motores paso a paso son ideales en todas aquellas aplicaciones donde se necesita de fuerza al mismo tiempo que precisión. Este motor unipolar tiene una fuerza de 190 oz/in (14Kg/cm) con un diámetro de eje de 6.35mm (1/4") y en formato NEMA 23.

Es un motor unipolar por lo que, si lo quieres conectar a un controlador bipolar, puedes ignorar los cables amarillo y blanco para conectarlo.

Características:

- Ángulo de paso: 1,8 grados
- Pasos: 200 por vuelta

- Fases: 4
- Voltaje: 4.5V
- Corriente: 2A/fase
- Diámetro de eje: 6.35mm
- Torque: 14Kg/cm
- Formato: NEMA 23

6.Componentes Electrónicos

Shield CNC – GRBL

La Arduino CNC Shield es una pequeña placa que permite controlar hasta 4 motores paso a paso fácilmente con Arduino o Freescale gracias a su formato shield. Soporta 4 controladores de potencia Pololu A4988 o Pololu DRV8825 y dispone de todas las conexiones necesarias para conectar interruptores de final de carrera, salidas de relé y diversos sensores. Es totalmente compatible con el firmware de control GRBL y puede ser utilizada con cualquier modelo de Arduino.

Con esta placa podrás disponer de un sistema completo para montar tu propia máquina CNC, cortadora láser o cualquier otro sistema que necesite un control preciso con motores paso a paso.

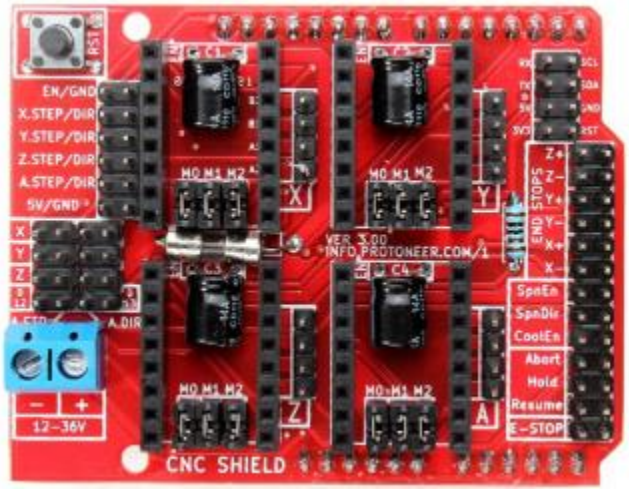


Figura 14. Shield CNC

Características:

- Voltaje de Potencia: 12- 36V DC
- Compatible con GRBL 0.9j (Firmware OpenSource para Arduino UNO que convierte código-G en comandos para motores Paso a Paso)
- Soporta 4 Ejes (X, Y, Z y duplicar uno de los anteriores o crear un eje a medida con los pines D12 y D13)
- 2 Fin de carrera por cada eje (6 en Total)
- Habilitador y dirección de Spindle
- Habilitador de refrigerante (coolant)
- Diseñador para drivers Pololu A4988 o DRV8825.
- Jumpers para elegir el micro-stepping de los drivers.
- Diseño Compacto
- Los motores se pueden conectar usando header o Molex hembra de 4 pines

Tarjeta de desarrollo Freescale FRDM-KL25Z

La tarjeta Freescale FRDM-KL25Z es una plataforma de desarrollo de bajo costo integrado por la familia de procesadores Kinetis L Series KL1x y KL2x basado en el procesador ARM® Cortex™-M0+.

Las características incluyen fácil acceso a los puertos I / O del procesador, el funcionamiento a baja energía permite el uso de baterías. Su construcción facilita el uso de interfaces para su expansión y posee además una interfaz integrada de depuración para la programación de la memoria flash y de control de gestión. La tarjeta Freescale FRDM-KL25Z es compatible con una amplia gama de software de desarrollo de Freescale y de terceros.

Los clientes también pueden utilizar mbed.org sin costo alguno, con pleno acceso al SDK en línea, herramientas, código reutilizable – lo que significa que no hay descargas, instalaciones o licencias – y una comunidad activa de desarrolladores.

Características:

- Procesador MKL25Z128VLK4 MCU – 48 MHz, 128 KB flash, 16 KB SRAM, USB OTG (FS), 80LQFP
- Conector mini USB tipo B con función de USB-host.
- Open SDA.
- Sensor capacitivo integrado.
- Acelerómetro MMA8451Q integrado.
- LED RGB integrado.
- Opciones de alimentación flexibles – USB, batería, fuente externa.
- Fácil acceso a los puertos I / O del procesador a través de los conectores compatibles con el Arduino UNO R3.
- Interfaz de depuración programable OpenSDA con múltiples aplicaciones disponibles, incluyendo:
- Interfaz de programación de la memoria flash del dispositivo de almacenamiento masivo.
- Interfaz de depuración P & E que provee control de ejecución de depuración y compatibilidad con herramientas IDE.
- Interfaz CMSIS-DAP.
- Aplicación de registro de datos.
- Mbed compatible.

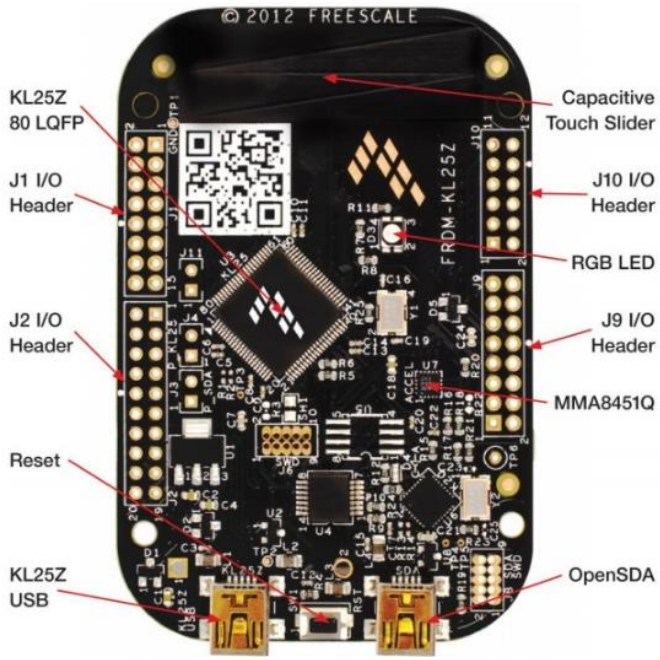


Figura15.- Puertos principales de la Freescale

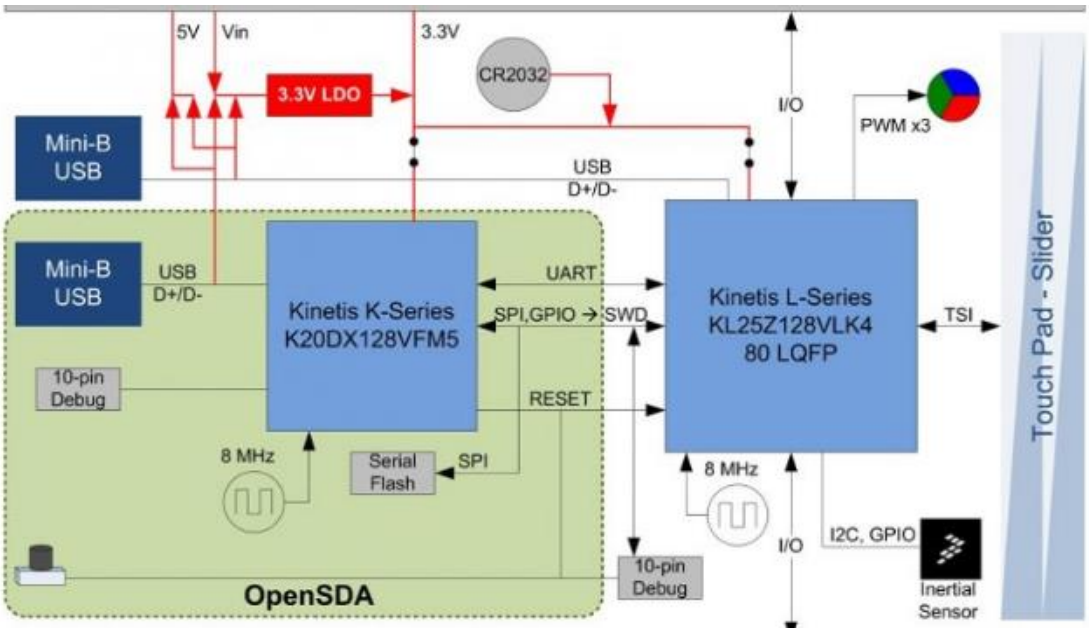


Figura 16. Esquemático de la Freescale

7.Modelo y simulación de brazo robótico (Diseño Modificado)

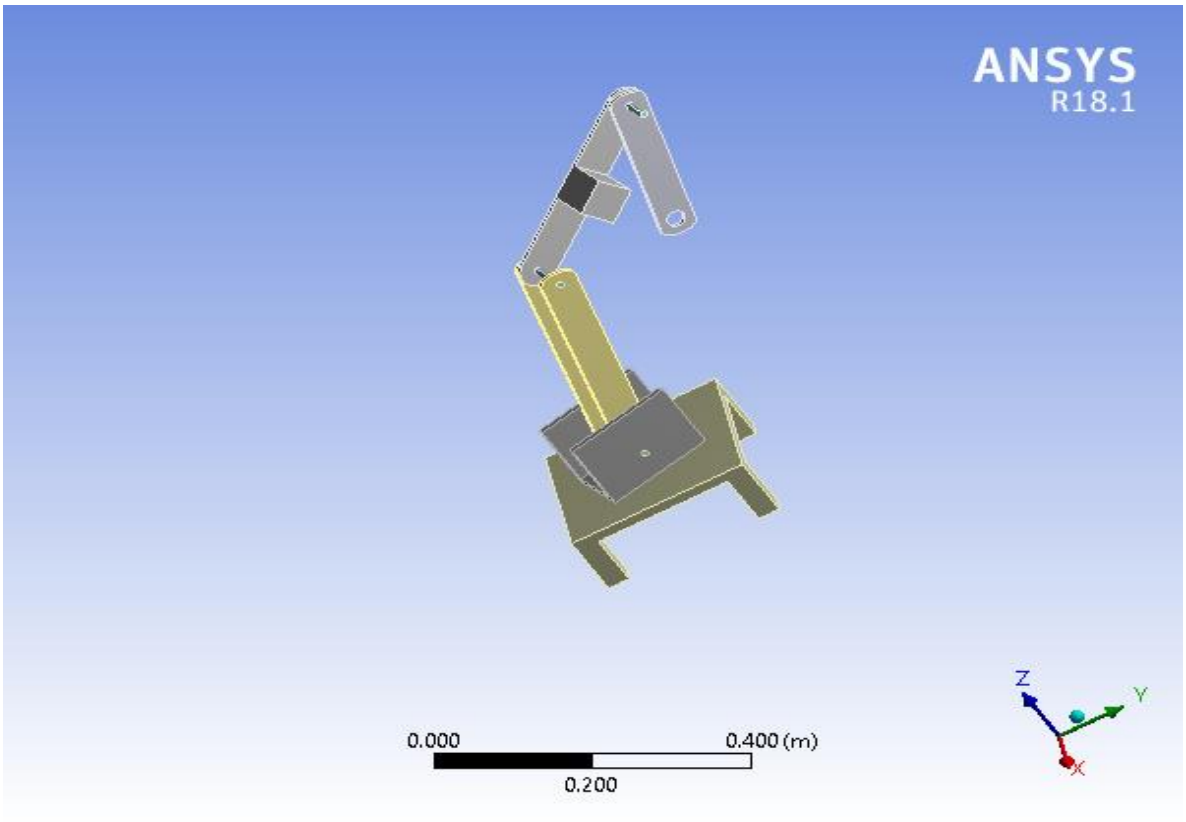


Figura 17. Estructura del robot modificada

En la figura 17 se muestra la estructura para su simulación de esfuerzos con el software de ANSYS.

Datos del Material: Madera

Material	Madera
Densidad	450 kg m^-3
Calor especifico	434 J kg^-1 C^-1

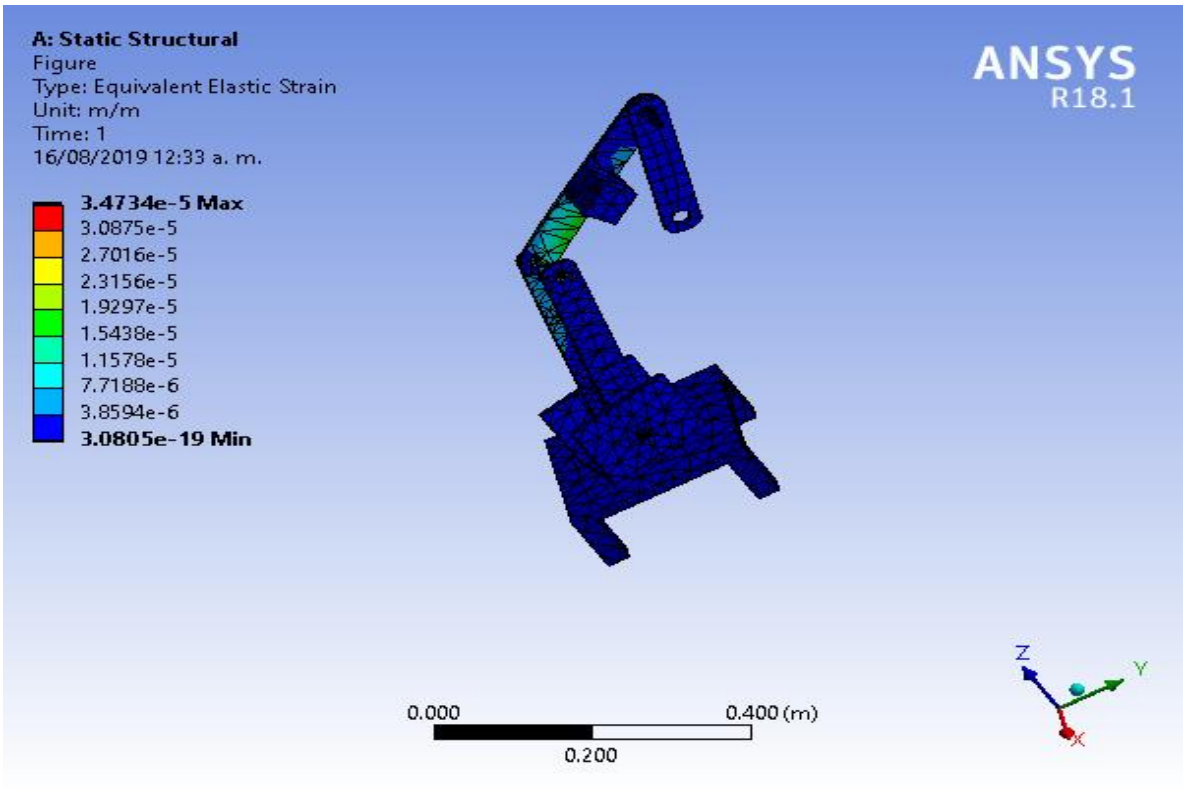


Figura 18. Resultado del análisis de deformación con el diseño modificado

En la figura 18 se muestra el resultado del análisis con esfuerzos de 10 N, los cuales indican que el material es el adecuado para resistir los esfuerzos

En el anexo A se puede apreciar mejor el análisis con madera y un nuevo material, aluminio.

8.Anexos

8.1 Anexo A: Análisis y modelado con material de Aluminio

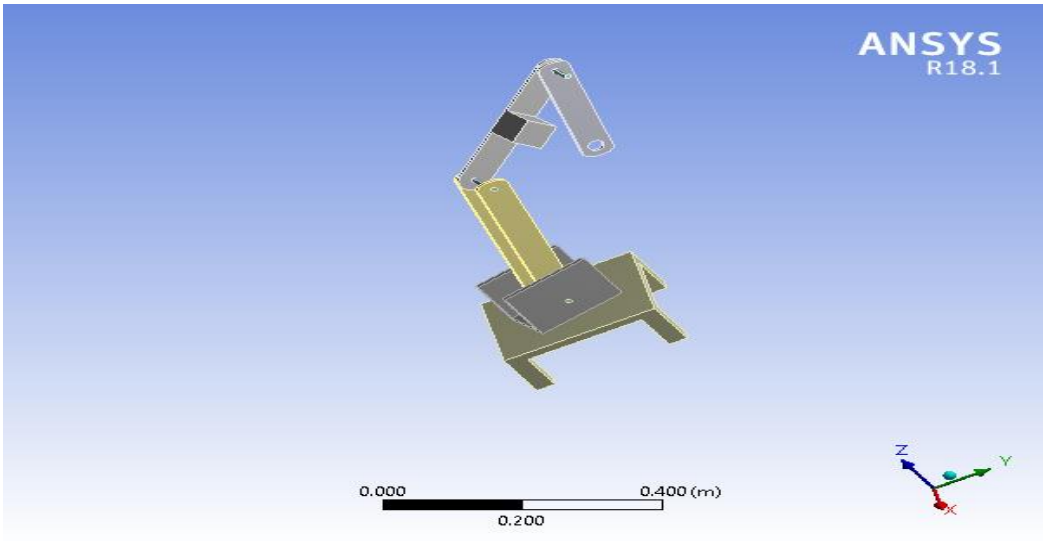


Figura 19. Estructura del robot con Aluminio como material

Constantes del aluminio

Density	2770 kg m^-3
Isotropic Secant Coefficient of Thermal Expansion	2.3e-005 C^-1
Specific Heat	875 J kg^-1 C^-1

Tabla 6. propiedades del aluminio

Coordenadas del sistema

Object Name	*Global Coordinate System*
State	Fully Defined
Definition	
Type	Cartesian
Coordinate System ID	0.
Origin	
Origin X	0. m
Origin Y	0. m
Origin Z	0. m
Directional Vectors	
X Axis Data	[1. 0. 0.]
Y Axis Data	[0. 1. 0.]
Z Axis Data	[0. 0. 1.]
Update Interval	2.5 s
Display Points	All
FE Connection Visibility	
Activate Visibility	Yes

Display	All FE Connectors
Draw Connections Attached To	All Nodes
Line Color	Connection Type
Visible on Results	No
Line Thickness	Single
Display Type	Lines

Table 7. nodos y coordenadas del brazo

Object Name	Equivalent Elastic Strain	Total Deformation
State	Solved	
Scope		
Scoping Method	Geometry Selection	
Geometry	All Bodies	
Definition		
Type	Equivalent Elastic Strain	Total Deformation
By	Time	
Display Time	Last	
Calculate Time History	Yes	
Identifier		
Suppressed	No	
Integration Point Results		
Display Option	Averaged	
Average Across Bodies	No	
Results		
Minimum	3.0805e-019 m/m	0. m
Maximum	3.4734e-005 m/m	1.4407e-004 m

Tabla 8.- Resultado del analisis

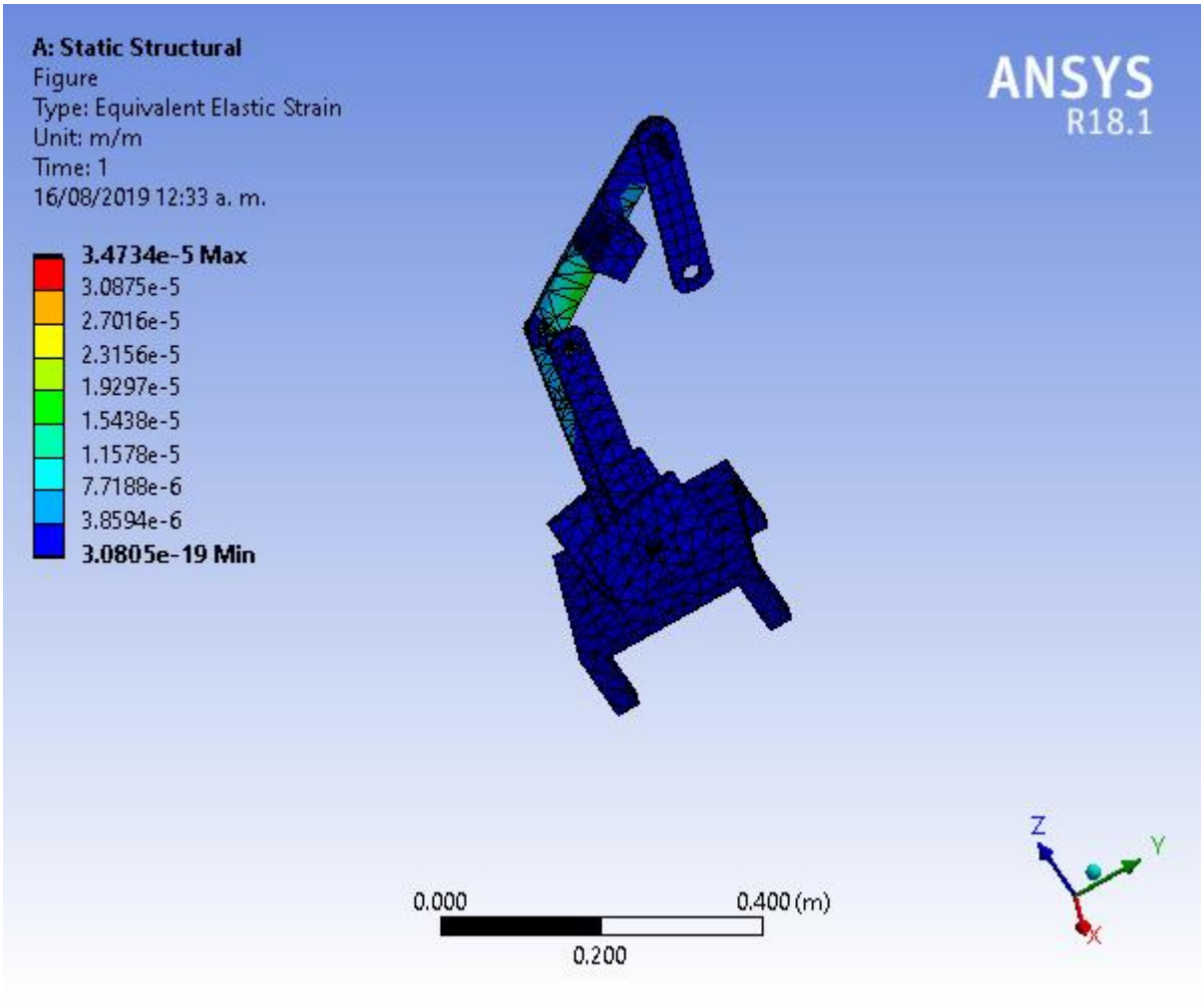


Figura 20. Grafica de esfuerzo de estrés en el segundo eslabón

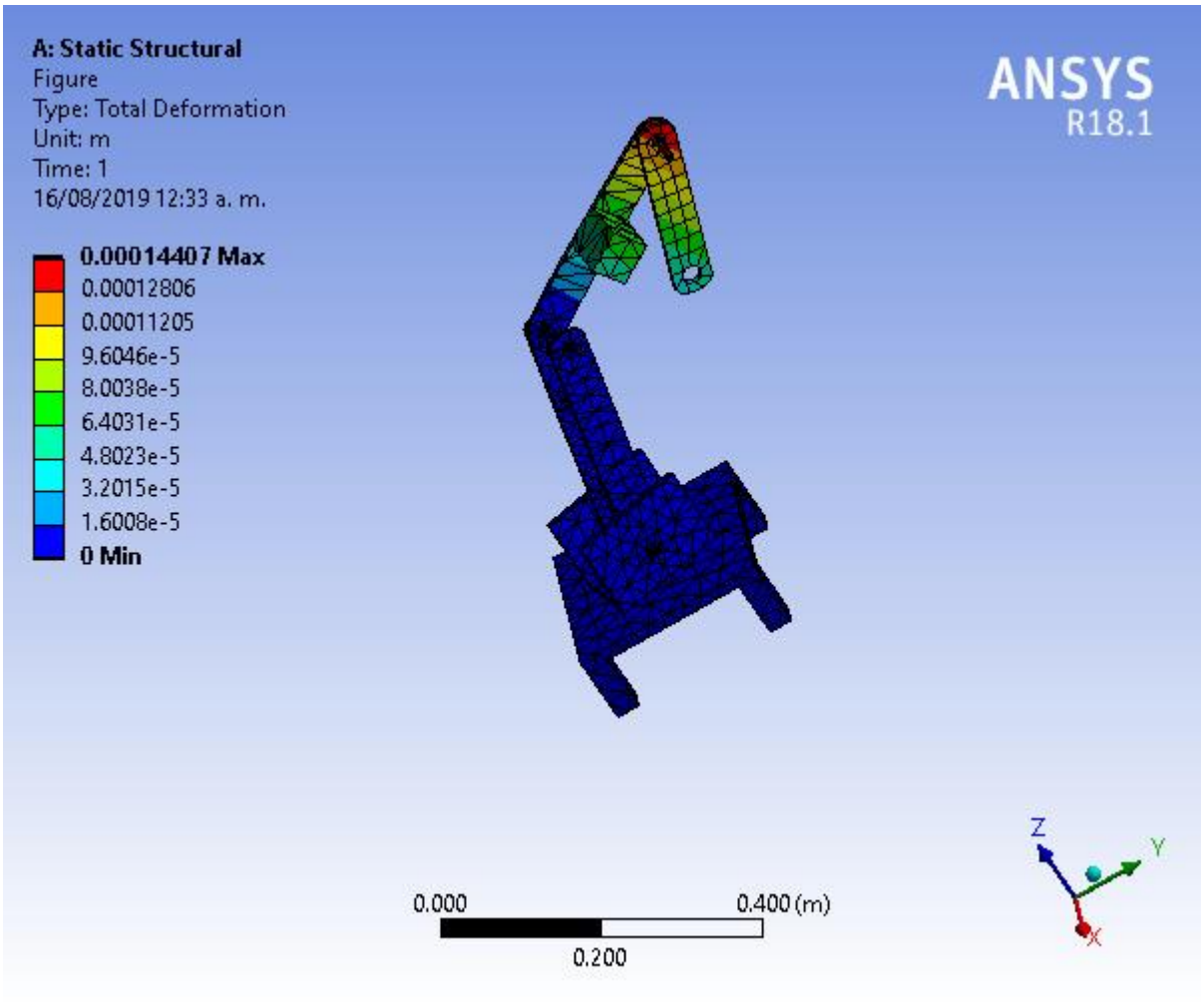


Figura 21. Grafica de esfuerzo de estrés en el tercer eslabón

8.2 Anexo B: Análisis con material madera

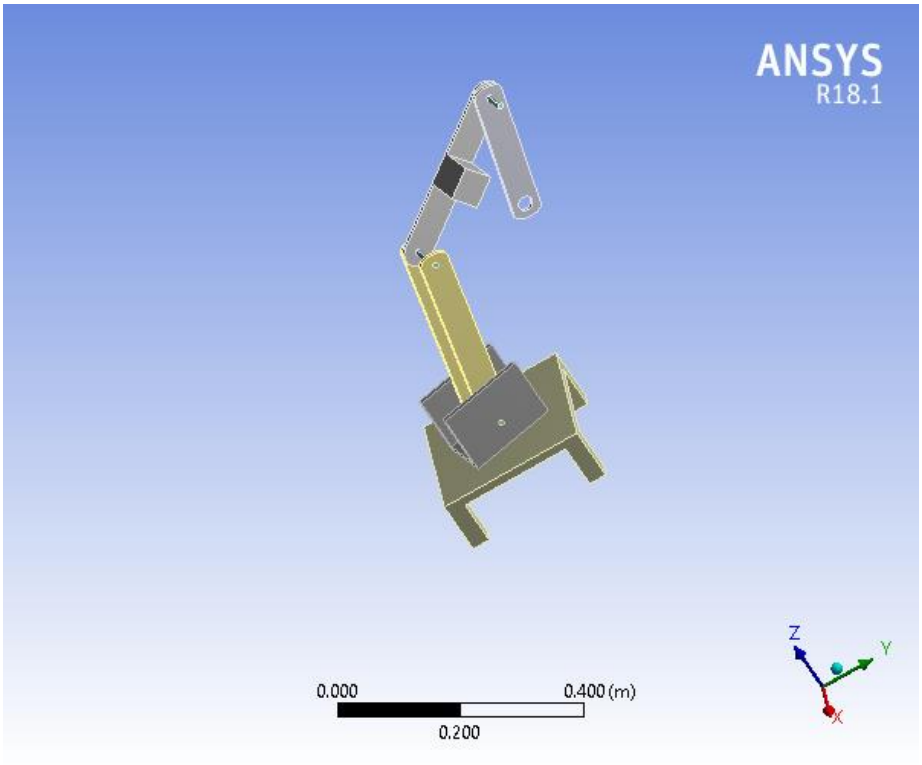


Figura 22. Estructura del robot con madera como material

Length Unit	Meters
Element Control	Program Controlled
Display Style	Body Color
Bounding Box	
Length X	0.27291 m
Length Y	0.45458 m
Length Z	0.61806 m
Properties	
Volume	2.3955e-003 m³
Mass	6.6355 kg
Scale Factor Value	1.
Statistics	
Bodies	9
Active Bodies	9
Nodes	9290
Elements	3426
Mesh Metric	None

Tabla 9. Datos de la estructura

Coordenadas del sistema

Object Name	Global Coordinate System
State	Fully Defined
Definition	
Type	Cartesian
Coordinate System ID	0.
Origin	
Origin X	0. m
Origin Y	0. m
Origin Z	0. m
Directional Vectors	
X Axis Data	[1. 0. 0.]
Y Axis Data	[0. 1. 0.]
Z Axis Data	[0. 0. 1.]
Group By	Bodies
Search Across	Bodies
Statistics	
Connections	12
Active Connections	12
Pinch Tolerance	Please Define
Generate Pinch on Refresh	No
Statistics	
Nodes	9290
Elements	3426

Tabla 10. Coordenadas de la estructura

Type	Equivalent Elastic Strain	Total Deformation
By	Time	
Display Time	Last	
Calculate Time History	Yes	
Identifier		
Suppressed	No	
Integration Point Results		
Display Option	Averaged	
Average Across Bodies	No	
Results		
Minimum	3.0805e-019 m/m	0. m
Maximum	3.4734e-005 m/m	1.4407e-004 m

Tabla 11. Resultado del análisis

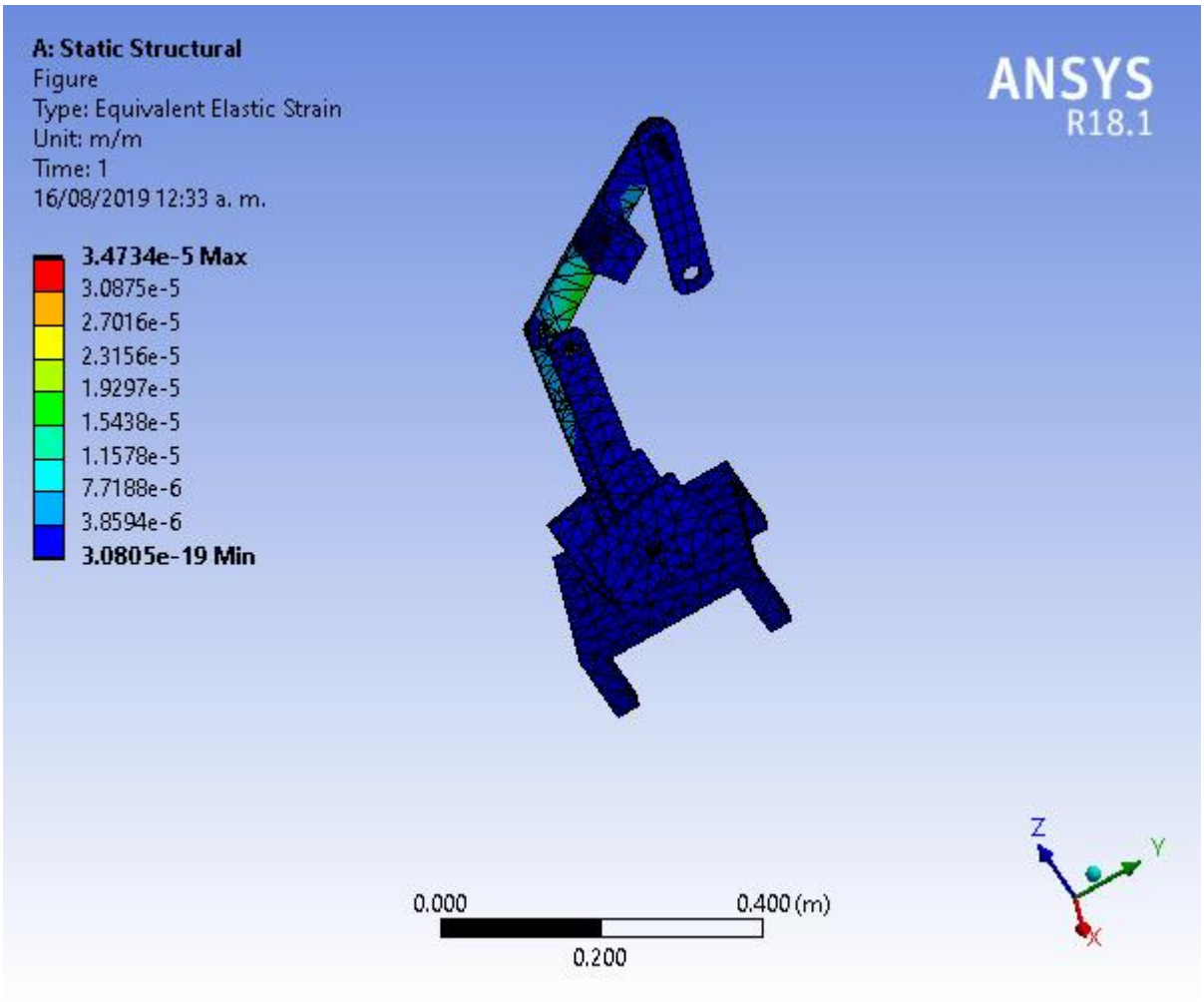


Figura 23. Grafica de esfuerzo de estrés en el segundo eslabón

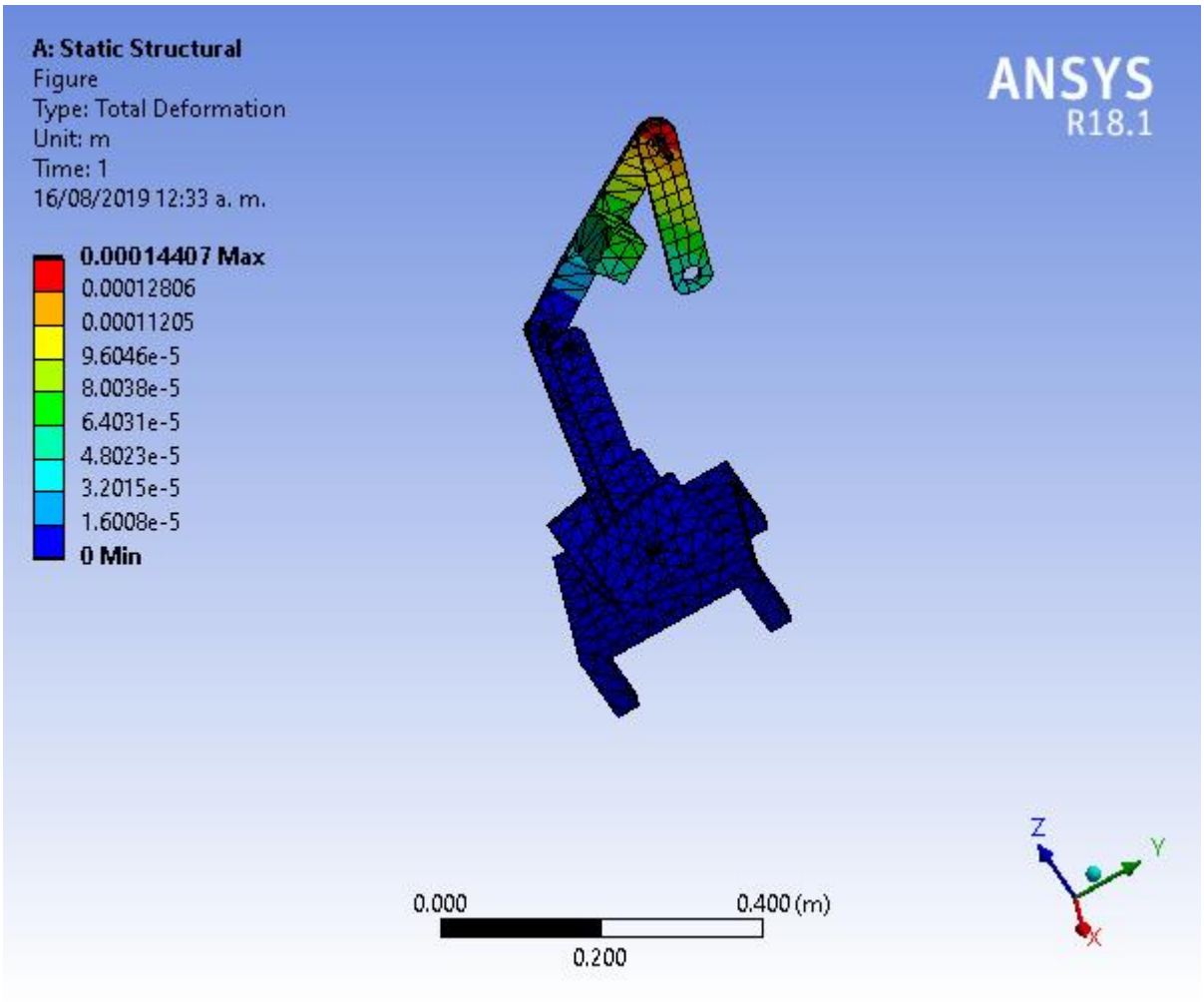


Figura 24. Grafica de esfuerzo de estrés en el segundo eslabón

8.3.- Anexo C: Programación

Programación de brazo Antropomórfico.

El uso de la Freescale como tarjeta de programación del brazo robótico permite al usuario escoger entre 2 entornos de programación usar un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) donde el usuario debe tener un nivel de conocimiento alto de la tarjeta y un alto nivel de conocimiento del IDE para programar y la otra opción es Mbed, que es una plataforma de prototipado rápido y experimentación con microcontroladores ARM Cortex-M3 y ARM Cortex-M0 de 32 bits.

Provee a los desarrolladores experimentados una plataforma productiva para realizar pruebas conceptuales y prototipos, mientras que a los principiantes sin experiencia previa les provee una forma accesible de realizar proyectos con microcontroladores de 32 bits mediante el acceso a las librerías, tutoriales y ejemplos, además de la comunidad online de mbed.

Mbed al ser una plataforma online de uso fácil, gratis y amigable para usuarios con conocimiento de programación en C fue la opción seleccionada pues el uso de un IDE también demanda un pago de licencia.

A continuación, se mostrarán segmentos del cogido usado para la programación del brazo robótico.

```
//libraries
#include "mbed.h"
#include <ros.h>
#include <std_msgs/Int16.h>
#include <std_msgs/UInt16.h>
#include <std_msgs/Int32.h>
#include <std_msgs/Float64.h>
#include <geometry_msgs/Twist.h>
```

Para empezar el desarrollo del código es necesario incluir las librerías. La librería ros.h permite la comunicación con ROS y las demás librerías son utilizadas

para incluir tipos de datos, en nuestro caso se incluyen enteros, enteros sin signo, flotantes y Twist.

```
ros::NodeHandle  nh;
DigitalOut step(D2);
DigitalOut dir(D5);
DigitalOut en(D8);
DigitalOut step1(D3);
DigitalOut dir1(D6);
DigitalOut step2(D4);
DigitalOut dir2(D6);
//AnalogIn Pot(PTB0);
int paso=0,paso1=0,paso2=0;
int servo_1,servo_2,servo_3;
```

En el segmento anterior se muestran los pines utilizados en la tarjeta Freescale es muy importante recalcar que esos pines deben tener exactamente esa dirección cuando se usa la Freescale y la Cnc Shield para el control de motores a pasos.

A continuación, se mostrarán los pines de la tarjeta Freescale y los pines en los cuales en un Arduino se mandan señales para hacer funcionar los motores a pasos (se muestra la imagen de Arduino porque la CNC Shield es compatible para la Freescale, pero fue diseñada para usarse con Arduino).

Los pines que más importan son los 3 pines de pasos y los tres pines de dirección los cuales establecen cuantos pasos dará el motor y hacía qué sentido girara el motor.

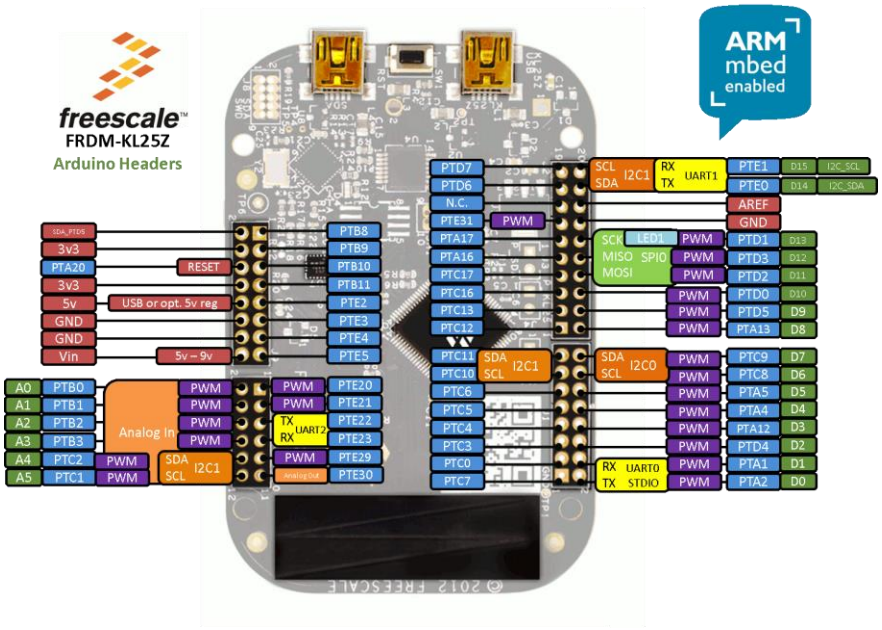


Figura 25. Pines de la tarjeta FreeScale

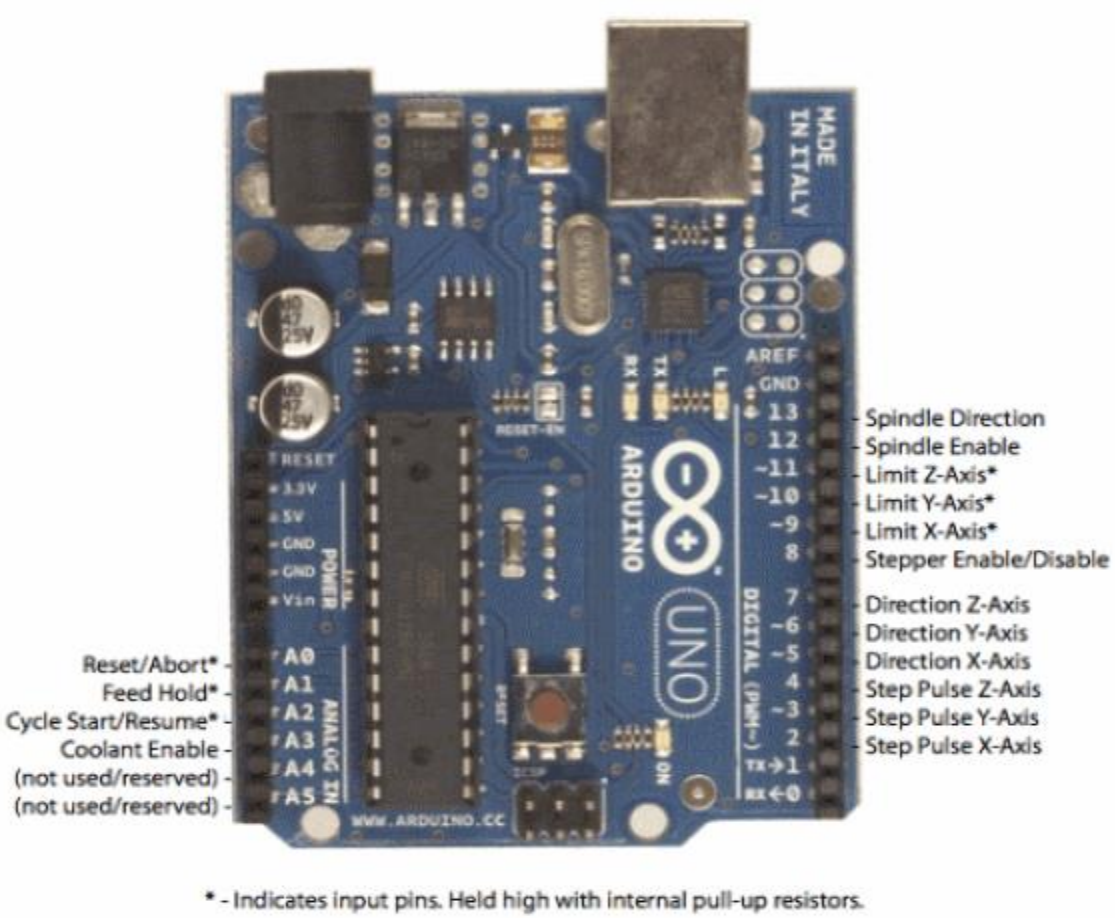


Figura 26. Pines de CNC Shield en Arduino

Después de nombrar los pines, se crean 6 variables de las cuales las variables con el nombre “paso” se encargan de llevar un contador cada una y en las variables “servo” se guardará el valor enviado por el usuario para saber cuántos pasos dará el motor.

Para que nuestro motor cambie de posición cada vez que publicamos un dato nuevo es necesario crear las funciones necesarias para que están sean llamadas cada vez que sean necesarios:

```
void servo_cb( const std_msgs::Int16& cmd_msg)
   servo_1 = cmd_msg.data;
}
void servo1_cb( const std_msgs::Int16& cmd_msg)
{
   servo_2 = cmd_msg.data;
}
void servo2_cb( const std_msgs::Int16& cmd_msg)
{
   servo_3 = cmd_msg.data;
}
void varios( const geometry_msgs::Twist& msg)
{
   servo_1 = msg.linear.x;
   servo_2 = msg.linear.y;
   servo_3 = msg.linear.z;
}
ros::Subscriber<geometry_msgs::Twist> sub3("varios", &varios);
ros::Subscriber<std_msgs::Int16> sub2("servo2", servo2_cb);
ros::Subscriber<std_msgs::Int16> sub1("servo1", servo1_cb);
ros::Subscriber<std_msgs::Int16> sub("servo", servo_cb);
```

Por último, se establecen el tipo de nodos que se utilizaran y la etapa de control de los motores hecha mediante arreglos de condiciones IF como se ve en el siguiente extracto del código.

```
int main()
{
   nh.initNode();
   nh.subscribe(sub);
   nh.subscribe(sub2);
   nh.subscribe(sub1);
   nh.subscribe(sub3);
   nh.advertise(pp);
   while (1) {
      nh.spinOnce();
      wait_ms(1);
   if(servo_1>paso){
      step=0;
      dir=0;
      en=0;
      wait(stepDelay);
      step=1;
      wait(stepDelay);
      paso=paso+1;
   }
   if(servo_1<paso){
      step=0;
      dir=1;
      en=0;
      wait(stepDelay);
      step=1;
      wait(stepDelay);
      paso=paso-1;
   }
   if(servo_2>paso1){
      step1=1;
      dir1=0;
      en=0;
      wait(stepDelay);
      step1=0;
      wait(stepDelay);
      paso1=paso1+1;
   }
```

```
    if(servo_2<paso1){
        step1=1;
        dir1=1;
        en=0;
        wait(stepDelay);
        step1=0;
        wait(stepDelay);
        paso1=paso1-1;
    }
    if(servo_3>paso2){
        step2=1;
        dir2=1;
        en=0;
        wait(stepDelay);
        step2=0;
        wait(stepDelay);
        paso2=paso2+1;
    }
    if(servo_3<paso2){
        step2=1;
        dir2=0;
        en=0;
        wait(stepDelay);
        step2=0;
        wait(stepDelay);
        paso2=paso2-1;
    }
}
}
```

Código Completo.

```
#include "mbed.h"
#include <ros.h>
#include <std_msgs/Int16.h>
#include <std_msgs/UInt16.h>
#include <std_msgs/Int32.h>
#include <std_msgs/Float64.h>
#include <geometry_msgs/Twist.h>
```

```
ros::NodeHandle  nh;

DigitalOut step(D2);
DigitalOut dir(D5);
DigitalOut en(D8);
DigitalOut step1(D3);
DigitalOut dir1(D6);
DigitalOut step2(D4);
DigitalOut dir2(D6);
//AnalogIn Pot(PTB0);
int paso=0,paso1=0,paso2=0;
int servo_1,servo_2,servo_3;

void servo_cb( const std_msgs::Int16& cmd_msg)
{
   servo_1 = cmd_msg.data;
}
void servo1_cb( const std_msgs::Int16& cmd_msg)
{
   servo_2 = cmd_msg.data;
}
void servo2_cb( const std_msgs::Int16& cmd_msg)
{
   servo_3 = cmd_msg.data;
}
void varios( const geometry_msgs::Twist& msg)
{
   servo_1 = msg.linear.x;
   servo_2 = msg.linear.y;
   servo_3 = msg.linear.z;

}

ros::Subscriber<geometry_msgs::Twist> sub3("varios", &varios);
ros::Subscriber<std_msgs::Int16> sub2("servo2", servo2_cb);
ros::Subscriber<std_msgs::Int16> sub1("servo1", servo1_cb);
ros::Subscriber<std_msgs::Int16> sub("servo", servo_cb);
```

```
int main()
{

    nh.initNode();
    nh.subscribe(sub);
    nh.subscribe(sub2);
    nh.subscribe(sub1);
    nh.subscribe(sub3);
    nh.advertise(pp);

    while (1) {
        nh.spinOnce();
        wait_ms(1);
    if(servo_1>paso){
        step=0;
        dir=0;
        en=0;
        wait(stepDelay);
        step=1;
        wait(stepDelay);
        paso=paso+1;
    }
    if(servo_1<paso){
        step=0;
        dir=1;
        en=0;
        wait(stepDelay);
        step=1;
        wait(stepDelay);
        paso=paso-1;
    }
    if(servo_2>paso1){
        step1=1;
        dir1=0;
        en=0;
        wait(stepDelay);
        step1=0;
```

```
    wait(stepDelay);
    paso1=paso1+1;
  }
  if(servo_2<paso1){
    step1=1;
    dir1=1;
    en=0;
    wait(stepDelay);
    step1=0;
    wait(stepDelay);
    paso1=paso1-1;
  }
  if(servo_3>paso2){
    step2=1;
    dir2=1;
    en=0;
    wait(stepDelay);
    step2=0;
    wait(stepDelay);
    paso2=paso2+1;

  }
  if(servo_3<paso2){
    step2=1;
    dir2=0;
    en=0;
    wait(stepDelay);
    step2=0;
    wait(stepDelay);
    paso2=paso2-1;
  }

}
}
```

Conclusión

Jesús Jail Avalos Lupercio: Utilizamos el microcontrolador FreeScale, la cual nos pareció muy intuitiva y fácil de programar ya que nunca la habíamos utilizado. Se utilizo la plataforma de internet Mbed como compilador. Pusimos a prueba distintos tipos de transmisión entre ellos poleas con bandas dentadas, piñón cadena y por último poleas con bandas.

Raúl Israel García Barajas: La realización de un brazo antropomórfico como proyecto final, trajo consigo complicaciones dignas de hacernos demostrar lo aprendido a través de materias cursadas anteriormente. La dinámica del robot, combinada con su diseño funcional y sus distintos controles fueron el resultado del trabajo que se realizó. Se hicieron leves cambios en el diseño como eliminar tornillería y madera innecesaria, necesarios para aligerar pesos y mejorar la funcionalidad del mismo.

Ricardo Martínez Jacinto: Principalmente se desarrolló un diseño que fuese ligero, con especificaciones resistentes, con una buena presentación y además que fuera fácil realizarle modificaciones. La parte del control, se realizó a través de ROS, un paquete de librerías en Linux, las cuales facilitan el trabajo a la hora de comunicarnos con nuestro proyecto, para darle indicaciones al robot. Se utilizaron coordenadas espaciales como instrucciones de movimiento en los distintos ejes con grado de libertad.

Josué Adrián Moreno Martínez:

Con la realización de este proyecto y la aplicación del conocimiento obtenido a lo largo de la carrera, puedo decir que fue el proyecto que más nos ha enseñado, desde el diseño mecánico hasta la programación tuvimos varios incidentes que reforzaron dicho conocimiento ayudándonos a escoger motores más fuertes como los NEMA 23 y agregando bandas y poleas para el sistema de movimiento mecánico, haciendo más efectivo su manejo y presentarlo sin ningún problema, por esto determino que en la realización de este proyecto se utilizaron todos los conocimientos obtenidos a lo largo de la carrera con un rotundo éxito.

Héctor Alejandro Nolasco Casillas:

Al realizar este proyecto pusimos a marcha nuestros conocimientos obtenidos de la carrera, aprendimos nuevos conocimientos y nuevas experiencias, como lo de los motores nemas son un poco más fáciles de utilizar, con esos se nos facilito un poco el trabajo al programarlos, le batallamos sobres los mecanismos, ya que no realizaban el funcionamiento adecuado y lo arreglamos poniéndoles bandas más ajustadas, acomodando el funcionamiento del brazo.

Rodrigo Rubio García:

En este proyecto tuvimos que experimentar una de las cosas nuevas, una de ellas los motores nema 23, los cuales nunca habíamos usado y es en esa la parte mas complicada del proyecto también el poder familiarizar con ROS fue uno de los retos mas importantes del proyecto dentro de lo que se refiere a programación, la parte del control, se llevó a cabo con ROS ya que se debió descarga e instalar un paquete de librerías en Linux, las cuales facilitan el trabajo a la hora de comunicarnos con nuestro proyecto para darle indicaciones al robot.

Juan Pablo Salguero Hernández:

La utilización de motores Nema, resulto bastante cómoda para este tipo de proyecto, principalmente se había decidido utilizar motor de corriente directa pero la implementación de los controles PID trajo bastantes inconvenientes con ello. En cambio, los motores NEMA, al ser motores a pasos, son más sencillos de controlar y solo basto la utilización de drivers adecuados, así como la SHIELD CNC