SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Ciudad Guzmán

INSTITUTO TECNOLÓGICO DE CD. GUZMÁN

Tesis

TEMA:

DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA BALL AND PLATE

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN ELECTRÓNICA

PRESENTA:

CARLOS AARÓN VELASCO LIÑAN

ASESOR(A):

M.I.E. CARLOS ENRIQUE MACIEL GARCÍA

CD. GUZMÁN JALISCO, MÉXICO, ENERO DE 2019

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Ciudad Guzmán

Cd. Guzmán, Municipio de Zapotlán el Grande, Jal, 18/Enero/2019

ASUNTO: Liberación de Proyecto para Titulación Integral.

M.C. FAVIO REY LUA MADRIGAL
JEFE DE LA DIVISION DE ESTUDIOS PROFESIONALES
P R E S E N T E

Por este medio le informo que ha sido liberado el siguiente proyecto para la Titulación Integral:

Nombre del Egresado:	CARLOS AARÓN VELASCO LIÑAN
Carrera:	INGENIERIA ELECTRONICA
No. De Control:	14290413
Nombre del Proyecto:	DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA BALL AND PLATE.
Producto:	TITULACIÓN INTEGRAL (Tesis)

Agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestros egresados.

A T E N T A M E N T E

M.E.H. MARCO ANTONIO SOSA LÓPEZ
JEFE DEL DEPTO. ELÉCTRICA Y ELECTRÓNICA

M.I.E. CARLOS ENRIQUE MACIEL GARCÍA ASESOR	M.I.E. JOSÉ MARÍA HERNÁNDEZ OCHOA REVISOR	M.I.E. LUIS ENRIQUE SALVADOR CANO REVISOR

C.p.expediente
DLAS/MASL/adc

S.E.P. TecNM
INSTITUTO TECNOLOGICO
DE CD. GUZMAN
DEPTO. ELECTRICA Y
ELECTRONICA

Análisis, Diseño e Implementación de un Modo de Control para un Sistema Ball and Plate II

Agradecimientos.

A Dios por permitirnos culminar la carrera profesional, a mis familiares: padres, hermana por su apoyo y motivación, a los profesores los cuales proporcionaron los cocimientos necesarios.

Resumen.

El presente proyecto se realizó con fines académicos, con el propósito de mostrar el modelado de un sistema ball and plate (bola y plato) utilizando algoritmos que involucren transformaciones lineales, como lo son traslación y rotación. Por medio de estos se realizó el cálculo de todos los puntos involucrados en nuestra plataforma de acuerdo con las características reales de nuestro sistema, para posteriormente aplicar modelos de cinemática inversa para obtener los ángulos que deben de mantener los servomotores para cierta posición de nuestro plato. Finalizando con la implementación en el lenguaje Python, el cual correrá dentro de una Raspberry Pi, donde se introducirán los valores de los ángulos y desplazamiento de nuestro sistema con respecto al origen establecido. Así pues, incluyendo la graficación en tiempo real de nuestra plataforma dentro de nuestro ambiente de trabajo con la ayuda de la librería Matplotlib.

Abstract.

The present project was carried out for academic purposes, showing the system model of ball and plate using algorithms that involve linear transformations, such as translation and rotation. By this method we got all the dots involved in or plat form following the trust features of our system, the next step was apply models of inverse kinematic to get all the angles that our servos must maintain. Finishing with the implementation in Python language, which will run inside our Raspberry Pi, where the values of the angles and the displacement of our system will be introduced with respect to the established origin. Also including the real-time graphing of our platform within our work environment with the help of the Matplotlib library.

índice

Índice de Figuras

Índice de Tablas

Introducción.

El ball and plate es un sistema el cual consta de una bola y un plato como su nombre lo dice, aunque el nombre se limite a esos dos términos el sistema es más complejo de lo que parece ya que el objetivo principal de este, es que se logre un equilibrio de la bola sobre el plato, girando el plato por medio de unos servomotores los cuales usualmente se sitúan en una base por debajo del plato, utilizando diversos métodos de censado para ubicar la posición de la bola sobre el plato, en nuestro caso se realizó mediante una pantalla táctil resistiva.

En el capítulo I se habla de los problemas planteados y el cómo es que este proyecto solucionara dichos problemas, se redactan los objetivos que se plantean en este proyecto, se justifica el por qué se eligió y se plasman las actividades desarrolladas.

En el capítulo II se muestra la descripción y características importantes de los diversos materiales, así como el principio de los conocimientos que se necesitaron en ciertos ámbitos tales como las matematizas, robótica y control.

En el capítulo III muestra los datos de la empresa tales como, donde se encuentra ubicada, política de calidad, valores, misión, visión entre otros.
En el capítulo IV muestra de manera detallada las actividades realizadas incluyendo imágenes, gráficos.

En el capítulo V se muestran los resultados obtenidos al realizar las actividades del capítulo anterior, describiendo el si se consiguió lo esperado o no y en tal caso que se realizó para conseguirlo.

En el capítulo VI se muestran los beneficios ante la empresa y cuáles fueron las limitaciones o dificultades que se obtuvieron al estar realizando dicho proyecto.

En el capítulo VI se muestran las conclusiones las cual redactan las experiencias y conocimientos adquiridos a lo largo del tiempo en el cual se realizó dicho proyecto,

Capítulo *I*: Generalidades del proyecto.

ANTECEDENTES DEL PROYECTO.

Dentro de la carrera de ingeniería en electrónica del Instituto Tecnológico de Ciudad Guzmán se presenta la propuesta para realizar un sistema real donde se puedan aplicar controladores PID con el fin de observar la reacción del sistema con dicho controlador.

PLANTEAMIENTO DEL PROBLEMA.

El ball and plate pretende satisfacer la necesidad de un sistema en donde se puedan aplicar controladores PID digitales diseñados por los estudiantes de la carrera de ingeniería en electrónica.

JUSTIFICACIÓN.

El ball and plate es un sistema que requiere un controlador donde influyen múltiples entradas y salidas, esto da la posibilidad de crear controladores MIMO (Multiple Inputs Multiple Outputs) y que los estudiantes puedan realizar prácticas ya que hasta el momento solo se realizaban de una salida.

OBJETIVOS.

General.

Diseñar, construir un sistema ball and plate

Especifico.

- Diseñar la plataforma
- Construir la plataforma
- Realizar el modelado
- Implementar el modelado en la plataforma real

HIPÓTESIS.

El sistema ball and plate diseñado a partir de una base con 6 servomotores, permitirá brindar un soporte mayor para el plato, y esto conlleva a una mayor estabilidad al ejercer movimiento de este.

Actividades desarrolladas.

1. Comenzamos estudiando el lenguaje Python.
2. Posteriormente se le instalo el sistema operativo a la Raspberry (en este caso fue el Raspbian).
3. Con la pantalla se realizaron mediciones para verificar el correcto funcionamiento.
4. Realizamos la comunicación de la pantalla con la Raspberry.
5. Instalamos librería de la controladora.
6. Realizamos pruebas de control de un servo con la Raspberry y con múltiples servos.
7. Diseño y construcción de la planta.
8. Modelado de la planta.
9. Implementación del modelo a planta real.

Capítulo *II*: Marco teórico.

RASPBERRY PI3:

¿Qué es?

Se trata de un ordenador o una computadora de pequeñas dimensiones, en la se podrá cargar un sistema operativo por medio de una memoria microSD. Sa podrá utilizar también como centro multimedia, centro de información o para fines recreativos. (Garzón Mancera & Garzón Melo, 2016)

En la Figura 2-1 se muestra la placa de una Raspberry.

Figura 2- 1: Raspberry Pi 3

Características:

Procesador de la tarjeta.

Es el elemento más importante, su ubicación está en el centro de la placa también se le denomina SOC es un chip PCM2835 Broadcom con un procesador ARM11 a 700Mhz, memoria de 512 Mb y una GPU videocore4. (Garzón Mancera & Garzón Melo, 2016)

Puerto Tarjeta MicroSD.

Una de las características de diseño de la tarjeta es que no tiene una unidad de disco duro, por eso utiliza una tarjeta de memoria SD como unidad de disco sólido, el tamaño mínimo es de 4GB si se necesita mayor capacidad de almacenamiento para instalación de software adicional soporta hasta 8GB. (Garzón Mancera & Garzón Melo, 2016)

Puerto Ethernet y USB.

Ambos puertos son alimentados por el chip BCM2835 para su correcto funcionamiento se sugiere una alimentación de la tarjeta de 2 amperios, para el funcionamiento de los 4 puertos USB cuenta con un hub USB que permite concentrar varios puertos, la velocidad de transferencia es de hasta 480Mbit/s; el puerto ethernet 10/100MB con un conector RJ45 tiene una tarjeta de red auto-sense que determina el tipo de conexión eliminando el tipo de cable y su configuración cruzado o punto a punto. (Garzón Mancera & Garzón Melo, 2016)

Pines GPIO.

Los pines de propósito general de entrada o salida GPIO ver Figura 2-2 permiten la comunicación de la tarjeta con el mundo real, aunque no son análogos tiene pines específicos para comunicación serial e I2C con resistencias en pull-up de 1.8k ohm lo que permite integrar una gran variedad de dispositivos como conversor adc y dac, sensores y controladores de dispositivos que utilizan este protocolo de comunicación, en total cuenta con 40 pines se debe tener cuidado ya que se pueden acceder de dos formas por el sistema de numeración de la tarjeta físico o por la numeración de canales del SOC. (Garzón Mancera & Garzón Melo, 2016)

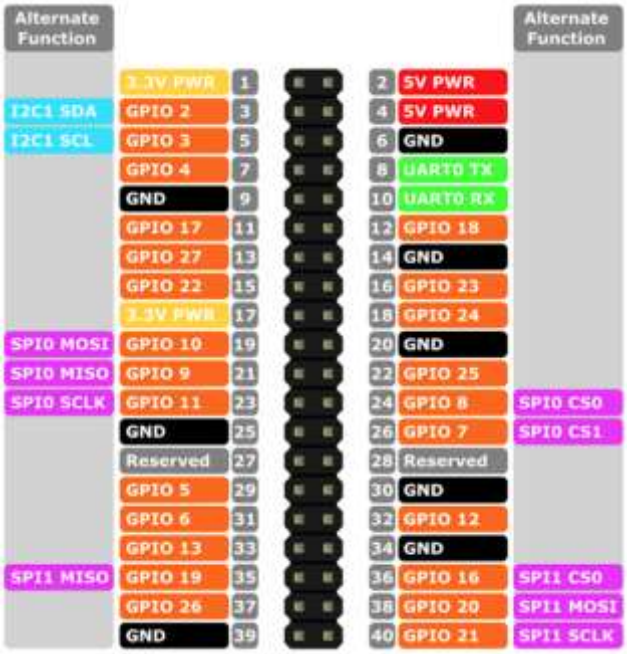

Figura 2- 2: Esquema de pines GPIO

Especificaciones:

- Procesador Quad Core 1.2GHz Broadcom BCM2837 de 64 bits
- 1 GB de RAM
- Bluetooth v4.1 de bajo consumo
- LAN inalámbrica BCM43438
- Puerto RJ45 Ethernet
- 4 puertos USB 2.0
- Conector GPIO de 40 pines
- Salida de 4 polos para audio estéreo y video compuesto
- 1 salida HDMI
- Puerto CSI para cámara
- Puerto DSI para pantalla táctil
- Puerto para tarjeta microSD
- Conector micro USB para alimentación

ARDUINO:

¿QUÉ ES ARDUINO?

Arduino es una plataforma de código abierto de prototipos electrónicos que se basa en hardware y software flexibles y fáciles de usar que ponen al alcance de cualquier persona la construcción de circuitos electrónicos/robots. En lo referente a hardware, se basa en placas que se pueden ensamblar a mano o que se pueden comprar directamente preensambladas. Cada una de las placas lleva un microcontrolador en el que se carga el programa software que es necesario desarrollar para "darle vida" a la placa. (Moreno Muño & Córcoles Córcoles, 2017)

En la siguiente imagen (Figura 2-3) puedes ver la Placa Arduino Uno R3 con sus partes más importantes señaladas que serán descritas en el apartado siguiente:

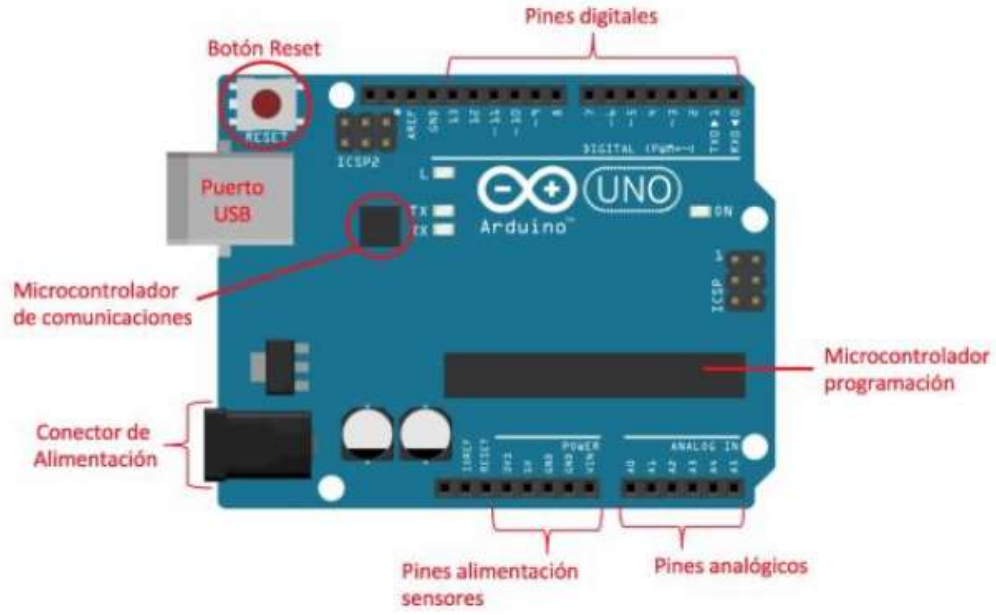

Figura 2- 3: Placa Arduino uno

Características:

Pines digitales.

Los pines digitales son las conexiones digitales de los dispositivos conectados en la placa. La placa de Arduino cuenta con 14 pines digitales, que van del 0 al 13.

Una señal digital solo puede tener dos estados:

- 0 (LOW, bajo, false): Indica 0V de tensión enviados desde la placa.
- 1 (HIGH, alto, true): Indica 5V de tensión enviados desde la placa.

(Moreno Muño & Córcoles Córcoles, 2017)

Por lo tanto, cuando ponemos un pin digital a valor HIGH, la placa suministra 5V de tensión por la salida que hayamos indicado, y si ponemos el valor a LOW suministrará 0V de tensión. (Ojo: Hay que tener en cuenta que los 5V no siempre son 5 ni los 0 siempre son 0)

Los pines digitales de Arduino pueden ser usados tanto de entrada como de salida.

(Moreno Muño & Córcoles Córcoles, 2017)

Pines analógicos.

Los pines analógicos pueden medir un rango de valores de voltaje, a diferencia de los digitales que solo entienden dos valores: 0-1, o lo que es lo mismo, 0V o 5V.
Con los pines analógicos vamos a poder **leer** valores intermedios entre 0V y 5V, representados con un valor entero comprendido entre **0 y 1023**, ya que la información se representa en números de 10 bits, y también vamos a poder **escribir** en los pines valores comprendidos entre **0 y 255**, ya que la información se representa en números de 8 bits.
En el punto anterior hemos hablado sobre pines digitales, si te fijas en ellos verás que aparecen algunos con el símbolo "~" en la placa, este símbolo indica que pueden ser utilizados también como pines analógicos. (Moreno Muño & Córcoles Córcoles, 2017)

Pines alimentación sensores.
Además de los pines de entrada y salida descritos anteriormente, Arduino dispone de pines que nos permiten alimentar componentes externos, concretamente uno con 5V y otro con 3,3V.
También dispone de pines de tierra (GND). (Moreno Muño & Córcoles Córcoles, 2017)

Microcontrolador de comunicaciones.
El microcontrolador de comunicaciones se encarga de gestionar las comunicaciones con todo lo que se conecta a la placa. (Moreno Muño & Córcoles Córcoles, 2017)

Microcontrolador de programación.
Este componente de la placa es el cerebro de la misma, es donde la placa almacena el programa que tiene que ejecutar y el que lo ejecuta.
El microcontrolador de la placa se programa utilizando el IDE (Entorno de Desarrollo Integrado) de programación gratuito de Arduino. En los apartados siguientes explicamos cómo instalarlo y como ponerlo a funcionar. (Moreno Muño & Córcoles Córcoles, 2017)

Botón reset.
El botón Reset permite reiniciar el programa que se ha cargado en el microcontrolador interrumpiendo la ejecución actual. Ten en cuenta que no borra el programa que se ha cargado, únicamente lo **reinicia**. (Moreno Muño & Córcoles Córcoles, 2017)

Puerto USB.

El puerto USB es el puerto mediante el cual nos comunicaremos con la placa de Arduino. Sus funciones principales son:

- Alimentación
- Cargar los programas en el microcontrolador.
- Envío de información desde la placa al ordenador.

Conector de alimentación.

Arduino dispone de un puerto de alimentación externo que nos permitirá hacer funcionar la placa sin utilizar un ordenador. Tienes que tener en cuenta el no alimentar la placa con más voltaje del que soporta, ya que podrías dañarla. Lo recomendado es alimentarla entre 7V y 12V. (Moreno Muño & Córcoles Córcoles, 2017)

SERVOMOTOR.

¿Qué es?

El servo es un potente dispositivo que dispone en su interior de un pequeño motor con un reductor de velocidad y multiplicador de fuerza, también dispone de un circuito que controla el sistema. El ángulo de giro del eje es de 180º en la mayoría de ellos, pero puede ser fácilmente modificado para tener un giro libre de 360º, como un motor standard.
(González, 2013)

Para controlar un servo se debe aplicar un pulso de duración y frecuencia específicas. Todos los servos disponen de tres cables, dos para alimentación Vcc y Gnd (4.8 a 6 [V]) y un tercero para aplicar el tren de pulsos de control, que hace que el circuito de control diferencial interno ponga el servo en la posición indicada, dependiendo del ancho del pulso.
(González, 2013)

En la Figura 2-4 y Figura2-5 se muestran de forma física los servomotores corona DS558HV y MG996 respectivamente, a continuación, se enuncian las especificaciones de cada uno de ellos.

Especificaciones servo corona DS558HV:

- Voltaje de funcionamiento: 6.0V / 7.4V
- Corriente de funcionamiento: 300mA / 400mA
- Velocidad de funcionamiento: 0.20sec.60º / 0.18sec.60º
- Puesto par: 12kg.cm / 14kg.cm
- Tamaño: 20X40X38.5mm
- Peso: 58g
- Banda muerta: ≤3useg
- Viaje de funcionamiento: 40º / un pulso lado 400US viajar
- Potenciómetro: 5 deslizador / Direct Drive
- Cojinete de bolas: MR106
- Engranaje: Metal
- Alambre de conector: 300mm
- Temperatura de funcionamiento: -20C + 60C

Figura 2- 4: Servo Corona DS558HV

Especificaciones servo MG996:

- Voltaje de operación: 4.8 V a 7.2 V

- Velocidad de operación: 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)
- Torque detenido: 9.4kgf·cm (4.8 V), 11kgf·cm (6 V)
- Con doble cojinete
- Ángulo de rotación: 120º aprox.
- Banda muerta: 5µs
- Peso: 55 g
- Dimensiones: Largo 40.7 mm, ancho 19.7 mm, altura 42.9 mm aprox.
- Largo del cable: 31 cm aprox.
- Con piñonera metálica
- Conector universal tipo "S" compatible con la mayoría de los receptores incluyendo Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum y Hitec, entre otros

Figura 2- 5: Servo MG996

Diferencias entre ambos:

Los servos corona DS558HV cuentan con las siguientes características que los hace resaltar en la comparativa con el servo MG996, lo que los convierte en la mejor opción a utilizar.

- Mayor resolución
- La banda muerta tightest
- El posicionamiento preciso
- Respuesta de control Faster
- Aceleración más rápida

- Torque constante durante todo el curso siervos

Un punto a favor del servo MG996 es el que resulta más económico en comparación con corona, aunque por dicho motivo cuenta con menores prestaciones tales como las mencionadas anteriormente.

LENGUAJE PYTHON

Historia.

Python fue creado por Guido van Rossum, un programador holandés a finales de los 80 y principio de los 90 cuando se encontraba trabajando en el sistema operativo Amoeba. Primariamente se concibe para manejar excepciones y tener interfaces con Amoeba como sucesor del lenguaje ABC. (Challenger Pérez, Díaz Ricardo , & Becerra García, 2014)

El 16 de octubre del 2000 se lanza Python 2.0 que contenía nuevas características como completa recolección de basura y completo soporte a Unicode. Pero el mayor avance lo constituye que este comenzó a ser verdaderamente desarrollado por la comunidad, bajo la dirección de Guido. (Challenger Pérez, Díaz Ricardo , & Becerra García, 2014)

El Python 3.0 es una versión mayor e incompatible con las anteriores en muchos aspectos, que llega después de un largo período de pruebas el 3 de diciembre del 2008. Muchas de las características introducidas en la versión 3 han sido compatibilizadas en la versión 2.6 para hacer de forma más sencilla la transición entre estas. (Challenger Pérez, Díaz Ricardo , & Becerra García, 2014)

Características.

- Python es un lenguaje muy expresivo, es decir, los programas Python son muy compactos: un programa Python suele ser bastante más corto que su equivalente en lenguajes como C. (Python llega a ser considerado por muchos un lenguaje de programación de muy alto nivel.)

- Python es muy legible. La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si utilizáramos otros lenguajes de programación.

- Python ofrece un entorno interactivo que facilita la realización de pruebas y ayuda a despejar dudas acerca de ciertas características del lenguaje.
- El entorno de ejecución de Python detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información muy rica para detectarlos y corregirlos.
- Python puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos.
- Posee un rico juego de estructuras de datos que se pueden manipular de modo sencillo.

(Marzal Varó & Gracia Luengo, 2013)

PANTALLA TÁCTIL RESISTIVA 4 HILOS

Modelo de la pantalla fk2-1645-000

Estos táctiles funcionan de forma matricial leyendo la posición donde se presionó comparando el valor de la coordenada X y coordenada Y de forma separada que luego el sistema micro controlado interpreta y procesa para saber en qué posición fue presionada. (Garzón Mancera & Garzón Melo, 2016)

Medición:

Al presionar el táctil (cualquier tipo de objeto dedo, puntero, esfera.), se unen dos capas metálicas resistivas lo cual permite la variación de la resistividad del componente ver Figura 2-5. La resistencia de estas capas no excede normalmente 1K ohm.

Los lados opuestos de las láminas disponen de contactos para acceder a un cable plano. El procedimiento para determinar las coordenadas de la posición del panel que ha sido presionado puede dividirse en dos pasos.

El primero es la determinación de la coordenada X y el segundo el de la coordenada Y del punto.

Para determinar la coordenada X, es preciso que el contacto izquierdo de la superficie X se conecte a tierra y el contacto derecho a la fuente de alimentación, esto permite obtener un divisor de tensión presionando el panel táctil, el valor de la tensión obtenida en el divisor se puede leer en el contacto inferior de la superficie Y. La tensión variará en el rango de 0 V a la tensión suministrada por la fuente de alimentación y depende de la coordenada X, si el punto está próximo al contacto izquierdo de la superficie X la tensión estará próxima a 0 V. (Challenger Pérez, Díaz Ricardo , & Becerra García, 2014)

Para la determinación de la coordenada Y, es preciso que el contacto inferior de la superficie Y se conecte a tierra, mientras que el contacto alto a la fuente de alimentación, en este caso la lectura de la tensión se hará en el contacto izquierdo de la superficie X. La tensión variará en el rango de 0 V a la tensión suministrada por la fuente de alimentación y depende de la coordenada Y, si el punto está próximo al contacto inferior de la superficie Y la tensión estará próxima a 0 las capas activas de una pantalla táctil a 4 hilos consisten en una capa parcialmente conductiva (resistiva) que se aplica de forma uniforme al panel. Las barras de bus conductivas se protegen con pintura de plata a través de los bordes opuestos del panel. Los paneles rígidos y flexibles se montan con las barras de bus perpendiculares entre sí. (Garzón Mancera & Garzón Melo, 2016)

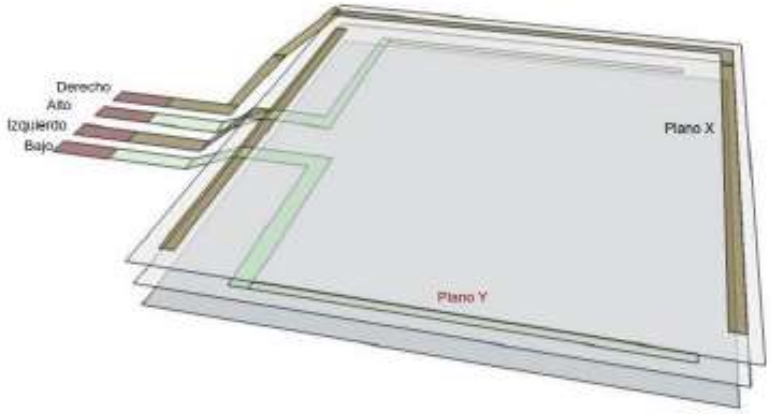

Figura 2- 6: Ejemplo del cómo se encuentran los contactos en una pantalla resistiva

CONTROLADORA PCA9685:

¿Qué es?

El Controlador de servos de 16 canales **(ver Figura 2-7)** está basado en el **chip PCA9685**, que es un generador PWM con reloj interno y resolución de 12 bits, el cual puede ser programado a través de una interfaz I2C. La gran versatilidad de este módulo es que solo requiere de 2 pines para controlar todas las señales PWM y no requiere señal alguna de oscilador o temporización, por lo que funciona de forma totalmente autónoma. (Factory, 2015)

Este módulo es una alternativa más económica al shield controlador de motores de Adafruit, sin embargo, conserva la compatibilidad de software con este, por lo que las librerías de Adafruit funcionan perfectamente con este módulo. (Factory, 2015)

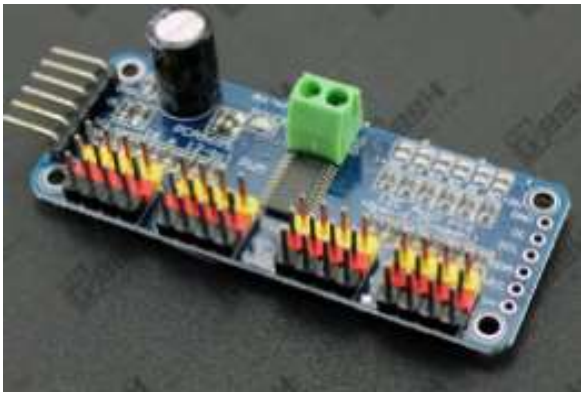

Figura 2- 7: Controladora PCA9685

Características:

- Controlador de PWM PCA9685 12 bits de resolución
- El controlador de servos es completamente independiente y no requiere señales de reloj externas
- Control de 16 señales PWM a través de 2 hilos I2C
- Salidas configurables “push-pull” o “drenador abierto”
- Habilitación / Des habilitación de salidas para anular rápidamente la señal en todas las salidas
- Bloque de terminales para la alimentación de los servos

- Resistencias de 220 Ohm en serie con cada salida PWM para protección y manejo directo de leds
- Diseño que permite colocar varias tarjetas en el mismo bus para un máximo de 992 salidas PWM independientes
- 6 jumpers "soldables" para configurar la dirección de I2C

(Factory, 2015)

TRANSFORMACIONES LINEALES.

Sean V y W espacios vectoriales reales. Una transformación lineal T de V en W es una función que asigna a cada vector **v** ∈ V un vector único Tv ∈ W y que satisface, para cada **u** y **v** en V y cada escalar α. (I. Grossman, 2015)

$$\boldsymbol{T}(u+v) = \boldsymbol{Tu} + \boldsymbol{Tv} \quad \textbf{(1)}$$

Y

$$\boldsymbol{T}(\boldsymbol{\alpha V}) = \boldsymbol{\alpha T v} \quad \textbf{(2)}$$

Transformación de rotación.

Suponga que el vector $v = \begin{pmatrix} x \\ y \end{pmatrix}$ en el plano xy se rota un ángulo θ (medido en grados o radianes) en sentido contrario al de las manecillas de reloj. Llame a este nuevo vector rotado $v' = \begin{pmatrix} x' \\ y' \end{pmatrix}$.

Entonces, como se ve en la Figura 2-x, si r denota la longitud de v (que no cambia por la rotación) (I. Grossman, 2015)

$x = r\cos\alpha \qquad y = r\,\text{sen}\,\alpha$

$x' = r\cos(\theta + \alpha) \qquad y' = r\,\text{sen}(\theta + \alpha)^{\dagger}$

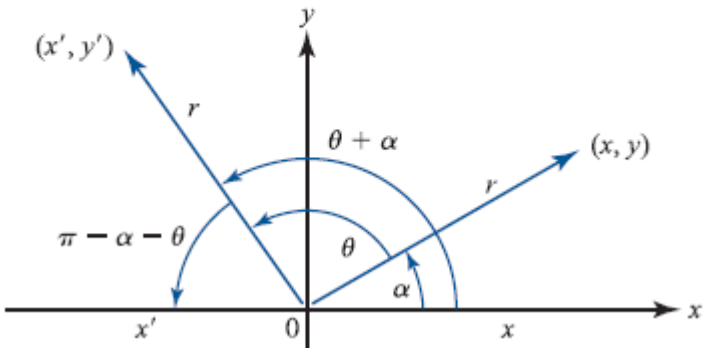

Figura 2- 8: Rotación de un vector

Pero $r\cos(\theta + \alpha) = r\cos\theta\cos\alpha - r\sin\theta\sin\alpha$, de manera que

$$x' = x\cos\theta - y\sin\theta \tag{3}$$

De manera similar $r\sin(\theta + \alpha) = r\sin\theta\cos\alpha + r\cos\theta\sin\alpha$, o sea

$$x' = x\sin\theta - y\cos\theta \tag{4}$$

Sea

$$A_0 = \begin{pmatrix} cos\theta & -sin\theta \\ sen\theta & cos\theta \end{pmatrix} \tag{5}$$

(I. Grossman, 2015)Entonces de (3) y (4), se ve que $A_0\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$. La transformación lineal T: $\mathbb{R}^2 \to \mathbb{R}^2$ definida por $T_v = A_0 v$, donde A_0 esta dado por (5), se llama transformación de rotación.

(I. Grossman, 2015)

CINEMÁTICA INVERSA.

Las ecuaciones cinemáticas directas, en la forma de la ecuación (6) o de las ecuaciones (7) y (8), establecieron la relación funcional entre las variables de articulaciones y la orientación y

posición del efector final. El problema de la cinemática inversa consiste en la determinación de las variables de articulaciones correspondientes a una orientación y posición específicas del efector final. La solución de este problema es de fundamental importancia con el fi n de transformar las especificaciones de movimiento asignadas al efector final en el espacio operacional en los correspondientes movimientos de espacio de las articulaciones. (Kumar Saha, 2010)

$$\mathbf{T} = \mathbf{T}1\ \mathbf{T}2\ \text{o}\ \mathbf{T}n \tag{6}$$

$$\mathbf{Q} = \mathbf{Q}1\ \mathbf{Q}2\ \text{o}\ \mathbf{Q}n \tag{7}$$

$$\mathbf{p} = \mathbf{a}1 + \mathbf{Q}1\ \mathbf{a}2 + \text{o} + \mathbf{Q}1\ \text{o}\ \mathbf{Q}n\text{–}1\ \mathbf{a}n \tag{8}$$

En la cinemática directa (ecuación 6), la matriz de posición y rotación del efector final se computa en forma única una vez que se conocen las variables de las articulaciones.

Por otro lado, el problema de la cinemática inversa es mucho más complejo debido a las siguientes razones:

- Por lo general, las ecuaciones por resolver son no lineales en las variables de articulaciones; de este modo, no siempre es posible encontrar una solución explicita.
- Pueden existir múltiples soluciones.
- También pueden existir soluciones infinitas, por ejemplo, en el caso de manipuladores robóticos cinemáticamente redundantes.
- Posiblemente no hay soluciones admisibles debido a la arquitectura del manipulador.

(Kumar Saha, 2010)

Un planteamiento posible frente al problema de la cinemática inversa es buscar una solución explicita usando algebra o geometría. Otra posibilidad es encontrar una solución numérica por medio de algún algoritmo de aproximación sucesiva. Aunque el primer planteamiento es generalmente más deseable para la aplicación de la solución al control de tiempo real de robots, no siempre es posible obtener las soluciones explicitas para manipuladores con arquitectura arbitraria. Mas bien, la clase de manipuladores para los cuales las soluciones explicitas quedan garantizadas es muy limitada. Observe, sin embargo, que la mayoría de los manipuladores que se utilizan en la industria pertenecen a esta clase. El enfoque algebraico a la solución explicita

significa la búsqueda de los ángulos de las de articulaciones por medio de la transformación algebraica de las ecuaciones (6) -(8). El enfoque geométrico significa la búsqueda de los ángulos de las articulaciones usando la heurística geométrica para aprovecharse de la estructura especial de los manipuladores. A veces es útil usar ambos enfoques juntos para resolver un problema. Puesto que es difícil encontrar una solución general para un manipulador con arquitectura arbitraria, las soluciones de cinemática inversa para la posición se presentan respecto a arquitecturas de robots específicas, como se explicará a continuación. (Kumar Saha, 2010)

Brazo planar de tres eslabones

Considere el brazo de la Figura 2-9. Lo que se busca aquí es encontrar los ángulos de las articulaciones θ1, θ 2 y θ 3 correspondientes a la posición y orientación determinadas para un efector final. Para un movimiento planar, la posición y orientación del efector final pueden especificarse por el origen del sistema coordenado 4, es decir, (*px py*), y por la orientación del sistema adjunto al efector final respecto al eje *X*1, es decir, el Angulo ϕ, como se muestra en la fi gura 6.3. Por lo tanto, se especifican como la entrada. Las soluciones se obtienen entonces mediante dos planteamientos distintos, como se explicará a continuación. (Kumar Saha, 2010)

- *Solución algebraica*

En primer lugar, se ilustra la técnica de la solución algebraica. Según el análisis de la cinemática de posición directa del brazo planar de tres eslabones, como se aprecia en la Figura 2-9

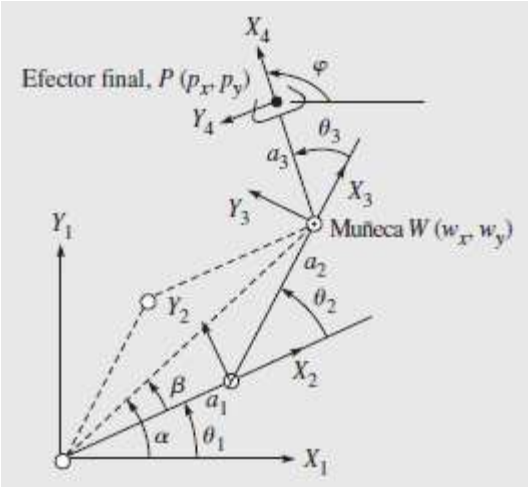

Figura 2- 9: Cinemática de un brazo planar de tres eslabones

$$\varphi = \theta 1 + \theta 2 + \theta 3 \quad (9)$$

$$px = a1c1 + a2c12 + a3c123 \quad (10)$$

$$py = a1s1 + a2s12 + a3s123 \quad (11)$$

Observe que estas ecuaciones son no lineales en los ángulos de las articulaciones θ 1, θ 2 y θ 3. Además, el problema de la cinemática inversa se simplifica mediante la subdivisión de la tarea, es decir, el efector final se orienta ya que se haya posicionado su muñeca *W* de la fi gura 6.3. Las coordenadas de *W* son *wx* y *wy*. Las relaciones de posición de las ecuaciones se reescriben entonces como: (Kumar Saha, 2010)

$$wx = px - a3\ cj = a1c1 + a2c12 \quad (12)$$

$$wy = py - a3\ sj = a1s1 + a2s12 \quad (13)$$

Elevando los dos lados de la ecuación al cuadrado, se encuentra

$$w2x + w2y = a12 + a2\text{^}2 + 2\ a1a2c2 \quad (14)$$

que proporciona

$$c_2 = \frac{w_2^2 + w_z^2 - a_1^2 - a_2^2}{2\,a_1\,a_2} \quad (15)$$

La existencia de una solución para la ecuación (15) impone la condición -1<=*c*2<=1. De lo contrario, el punto (*px*, *py*) estará fuera del espacio de trabajo alcanzable del brazo.
Entonces, (Kumar Saha, 2010)

$$s_2 = \pm\sqrt{1 - c_2^2} \quad (16)$$

donde el signo positivo es relativo a la postura del codo hacia arriba y el signo negativo se refiere a la postura del codo hacia abajo. Por lo tanto, el ángulo θ 2 se computa como

$$\theta 2 = \text{atan2}\,(s2, c2) \tag{16}$$

donde “atan2”, en contraste con “atan”, es una función en cualquier lenguaje de computación, como MATLAB, C, FORTAN, etc., que computa el valor de “tan^-1()” en el cuadrante apropiado. Si un punto se encuentra en el primer y tercer cuadrantes, la función “atan” da la respuesta en los dos casos que corresponden únicamente al primer cuadrante. Esto es así porque los argumentos *y/x* y *–y*(*–x*) dan como resultado el mismo valor numérico. Cuando se usa “atan2”, el argumento *y/x* proporcionará el resultado correspondiente al primer cuadrante, mientras que *–y/–x* dará resultados que se encuentran en el tercer cuadrante. Una vez que se determinó θ2, el ángulo θ1 se encuentra mediante la expansión de *c*12 y *s*12 de la ecuación (12) y su restructuración como: (Kumar Saha, 2010)

$$wx = (a1 + a2c2)c1 - a2s1s2 \tag{17}$$

$$wy = (a1 + a2c2)s1 + a2c1s2 \tag{18}$$

Con el fi n de evaluar **θ**1, la ecuación se multiplica por *a*2*s*2 y la ecuación (18) por (*a*1 + *a*2*c*2), seguido por la resta de la primera con respecto a la última. Esto da como resultado el valor de *s*1 como: (Kumar Saha, 2010)

$$s_1 = \frac{(a_1 + a_2 c_2) w_y - a_2 s_2 w_x}{\Delta} \tag{19}$$

En forma parecida, *c*1 se obtiene como

$$c_1 = \frac{(a_1 + a_2 c_2) w_x + a_2 s_2 w_y}{\Delta} \tag{20}$$

donde _ ≡ *a*1^2 + *a*2^2 + 2*a*1*a*2*c*2 = *wx*^2 + *wy*^2. Siguiendo la analogía a la ecuación (16), la solución

para _1 se obtiene de la siguiente manera:

$$\theta 1 = \text{atan2}\,(s1, c1) \tag{21}$$

Finalmente, se encuentra el ángulo θ 3, según la expresión de la ecuación, como

$$\boldsymbol{\theta}\ 3 = \varphi - \boldsymbol{\theta}\ 1 - \boldsymbol{\theta} 2 \tag{22}$$

(Kumar Saha, 2010)

CONTROLADOR PID.

Un controlador PID combina las tres acciones de control (ver Figura 2-10)

- Proporcional
- Integral
- Derivativa

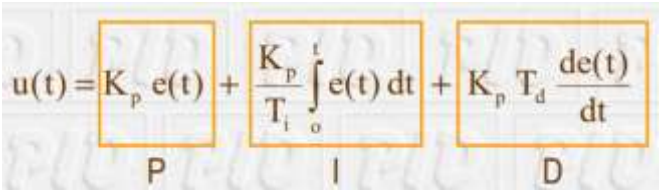

Figura 2- 10: Controlador PID

Control PID de plantas.

La Figura 2-11 muestra un control PID de una planta. Si se puede obtener un modelo matemático de la planta, es posible aplicar diversas técnicas de diseño con el fin de determinar los parámetros del controlador que cumpla las especificaciones del transitorio y del estado estacionario del sistema en lazo cerrado. Sin embargo, si la planta es tan complicada que no es fácil obtener su modelo matemático, tampoco es posible un método analítico para el diseño de un controlador PID. En este caso, se debe recurrir a procedimientos experimentales para la sintonía de los controladores PID. (Ogata, 2010)

El proceso de seleccionar los parámetros del controlador que cumplan con las especificaciones de comportamiento dadas se conoce como sintonía del controlador. Ziegler y Nichols sugirieron reglas para sintonizar los controladores PID (esto significa dar valores a *Kp*, *Ti* y *Td*) basándose

en las respuestas escalón experimentales o en el valor de *Kp* que produce estabilidad marginal cuando sólo se usa la acción de control proporcional. Las reglas de Ziegler-Nichols, que se presentan a continuación, son muy convenientes cuando no se conocen los modelos matemáticos de las plantas. (Por supuesto, estas reglas se pueden aplicar al diseño de sistemas con modelos matemáticos conocidos.) Tales reglas sugieren un conjunto de valores de *Kp*, *Ti* y *Td*.

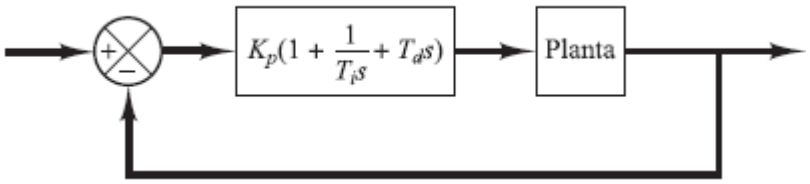

Figura 2- 11: Control PID de una planta

que darán una operación estable del sistema. No obstante, el sistema resultante puede presentar una gran sobre elongación en su respuesta escalón de forma que resulte no aceptable. En tales casos se necesitará una serie de ajustes finos hasta que se obtenga el resultado deseado. De hecho, las reglas de sintonía de Ziegler-Nichols dan una estimación razonable de los parámetros del controlador y proporcionan un punto de partida para una sintonía fina, en lugar de dar los parámetros *Kp*, *Ti* y *Td* en un único intento. (Ogata, 2010)

Reglas de Ziegler-Nichols para sintonizar controladores PID.

Ziegler y Nichols propusieron reglas para determinar los valores de la ganancia proporcional *Kp*, del tiempo integral *Ti* y del tiempo derivativo *Td*, basándose en las características de respuesta transitoria de una planta dada. Tal determinación de los parámetros de los controladores PID o sintonía de controladores PID la pueden realizar los ingenieros mediante experimentos sobre la planta. (Después de la propuesta inicial de Ziegler-Nichols han aparecido numerosas reglas de sintonía de controladores PID. Estas reglas están disponibles tanto en publicaciones técnicas como de los fabricantes de estos controladores).

Hay dos métodos denominados reglas de sintonía de Ziegler-Nichols: el primero y el segundo método. A continuación, se hace una breve presentación de estos dos métodos. (Ogata, 2010)

Primer método.

En el primer método, la respuesta de la planta a una entrada escalón unitario se obtiene de manera experimental, tal como se muestra en la Figura 2-12. Si la planta no contiene

integradores ni polos dominantes complejos conjugados, la curva de respuesta escalón unitario puede tener forma de S, como se observa en la Figura 2-13. Este método se puede aplicar si la respuesta muestra una curva con forma de S. Tales curvas de respuesta escalón se pueden generar experimentalmente o a partir de una simulación dinámica de la planta. (Ogata, 2010)

La curva con forma de S se caracteriza por dos parámetros: el tiempo de retardo L y la constante de tiempo T. El tiempo de retardo y la constante de tiempo se determinan dibujando una recta tangente en el punto de inflexión de la curva con forma de S y determinando las intersecciones de esta tangente con el eje del tiempo y con la línea $c(t)\%K$, tal como se muestra en la Figura 2-12. En este caso, la función de transferencia $C(s)/U(s)$ se aproxima mediante un sistema de primer orden con un retardo del modo siguiente: (Ogata, 2010)

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \tag{23}$$

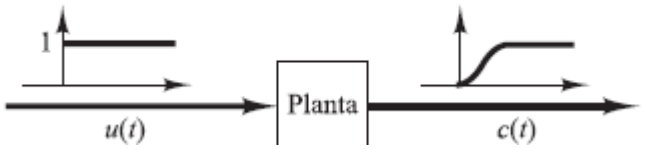

Figura 2- 12: Respuesta escalón unitario de una planta

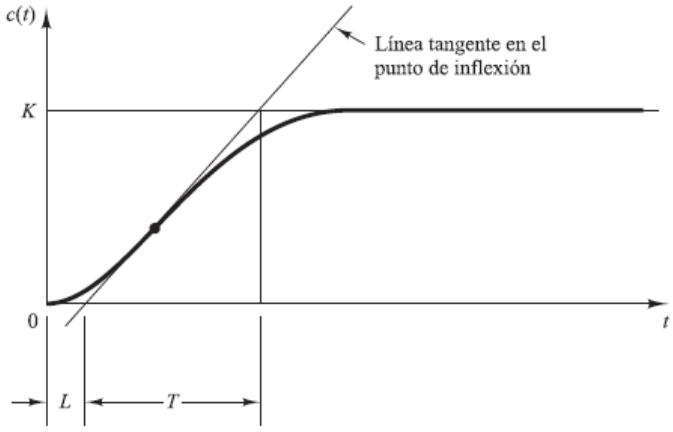

Figura 2- 13: Curva de respuesta en formato de S

Ziegler y Nichols sugirieron establecer los valores de *Kp*, *Ti* y *Td* de acuerdo con la fórmula que se muestra en la Tabla 8-1. Obsérvese que el controlador PID sintonizado mediante el primer método de las reglas de Ziegler-Nichols produce las ecuaciones de la Figura 2-14. (Ogata, 2010)

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

$$= 0.6K_{cr}\left(1 + \frac{1}{0.5P_{cr}s} + 0.125P_{cr}s\right)$$

$$= 0.075K_{cr}P_{cr}\frac{\left(s + \frac{4}{P_{cr}}\right)^2}{s}$$

Figura 2- 14: Ecuaciones Resultantes

Por tanto, el controlador PID tiene un polo en el origen y un cero doble en *s*%.1/*L*.

Tabla 2- 1: Regla de sintonía de Ziegler-Nichols basada en la respuesta escalón de la planta (primer metodo)

Tipo de controlador	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Segundo método.

En el segundo método, primero se fija *Ti*%ä y *Td*%0. Usando sólo la acción de control proporcional (véase la Figura 2-15), se incrementa *Kp* desde 0 hasta un valor crítico *K*cr, en donde la salida presente oscilaciones sostenidas. (Si la salida no presenta oscilaciones sostenidas

para cualquier valor que pueda tomar *Kp*, entonces este método no se puede aplicar.) Así, la ganancia crítica *K*cr y el periodo *P*cr correspondiente se determinan experimental- mente (véase la Figura 2-16). Ziegler-Nichols sugirieron que se establecieran los valores de los parámetros *Kp*, *Ti* y *Td* de acuerdo con la fórmula que se muestra en la Tabla 2-2. (Ogata, 2010)

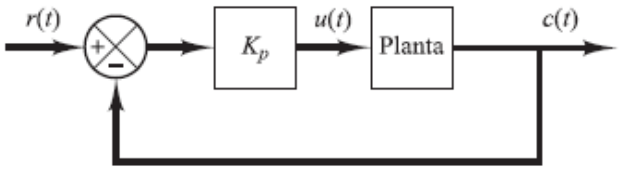

Figura 2- 15: Sistema en lazo cerrado con un controlador proporcional

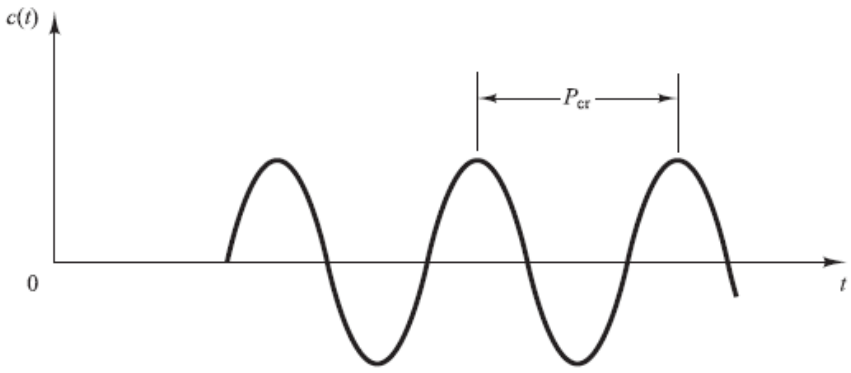

Figura 2- 16: Oscilación sostenida con periodo Pcr (Pcr se mide en seg.)

Tabla 2- 2: Regla de sintonía de Ziegler-Nichols basada en la ganancia crítica Kcr y periodo crítico Pcr (segundo metodo)

Tipo de controlador	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Obsérvese que el controlador PID sintonizado mediante el segundo método de las reglas de Ziegler-Nichols produce (véase Figura 2-17). (Ogata, 2010)

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

$$= 0.6K_{cr}\left(1 + \frac{1}{0.5P_{cr}s} + 0.125P_{cr}s\right)$$

$$= 0.075K_{cr}P_{cr}\frac{\left(s + \frac{4}{P_{cr}}\right)^2}{s}$$

Figura 2- 17: Ecuaciones resultantes del segundo método

Por tanto, el controlador PID tiene un polo en el origen y un cero doble en s%.4/*P*cr. Conviene darse cuenta de que, si el sistema tiene un modelo matemático conocido (como la función de transferencia), entonces se puede emplear el método del lugar de las raíces para encontrar la ganancia crítica *K*cr y las frecuencias de las oscilaciones sostenidas ucr, donde 2n/ucr%*P*cr. Estos valores se pueden determinar a partir de los puntos de cruce de las ramas del lugar de las raíces con el eje *j*u. (Obviamente, si las ramas del lugar de las raíces no cortan al eje *j*u este método no se puede aplicar). (Ogata, 2010)

Capítulo *III*. Caracterización del área de trabajo.

Instituto Tecnológico de Ciudad Guzmán

Avenida Tecnológico #100.
Ciudad Guzmán, Mpio. de Zapotlán el Grande, Jalisco, México
Tel. 01 (341) 575 20 50
Fax. 01 (341) 575 20 74
Web. www.itcg.edu.mx

ANTECEDENTES:

En Mayo de 1972 se hizo la petición al Sr. Presidente de la República, durante una de sus visitas a Cd. Guzmán, de la creación de una Institución Educativa de nivel superior en el Sur de Jalisco, que tuviera como finalidad propiciar el desarrollo cultural, técnico y económico de la región, así como reducir los flujos migratorios de los jóvenes estudiantes hacia las grandes ciudades en busca de su formación profesional. Así mismo, otro de los objetivos consistió en fomentar el arraigo de los egresados en sus lugares de origen.

El Instituto Tecnológico nace el 13 de Septiembre de 1972, sobre una extensión de terreno de 26 hectáreas con el nombre de Instituto Tecnológico Regional No.29 de Ciudad Guzmán, Jalisco.

El inicio de sus actividades se hace el 20 del mismo mes y año en las instalaciones del CERETI (hoy CETIS) en la ciudad de Guadalajara, contando con una población de 120 alumnos de nivel licenciatura, 15 docentes y 10 administrativos.

El 22 de Marzo de 1973, el CAPECE terminó la construcción de aulas y laboratorios, empezando en ese momento el nivel bachillerato ya en las instalaciones propias y en el mes de Junio del mismo año se integraron los estudiantes que se encontraban en el CERETI de Guadalajara, a partir de ese periodo el ITCG ha logrado un desarrollo en todo sentido apegado a los lineamientos emanados de la SEP. Las carreras ofrecidas por este Instituto actualmente se distribuyen de la siguiente manera: cuatro Licenciaturas y seis Ingenierías.

MISIÓN.

Somos una institución perteneciente al Tecnológico Nacional de México, que ofrece servicios de Educación Superior Tecnológica, formando profesionales íntegros, con capacidad científica y tecnológica para contribuir en la mejora de los sectores público, social y productivo, coadyuvando responsablemente en el desarrollo sustentable para favorecer la conformación de una sociedad más justa y humana.

VISIÓN.

Ser una Institución de Educación Superior Tecnológica con reconocimiento internacional y programas estratégicos de investigación e innovación tecnológica de vanguardia, acordes a las necesidades de los sectores público, social y productivo, ofreciendo servicios de calidad, producto del desarrollo, participación y superación permanente de nuestro talento humano, fomentando una relación responsable y comprometida con el medio ambiente.

VALORES.

- Interés público.
- Respeto.
- Respeto a los derechos humanos.
- Igualdad y no discriminación.
- Equidad de género.
- Entorno cultural y ecológico.
- Integridad.
- Cooperación.
- Liderazgo.
- Transparencia.
- Rendición de cuentas.

POLÍTICA DE CALIDAD.

El Instituto Tecnológico de CD. GUZMÁN establece el compromiso de implementar sus procesos, orientados a la atención de los requerimientos del contexto y de los riesgos identificados, para la satisfacción de estudiantes y partes interesadas, cumpliendo con sus

requisitos aplicables, mediante la eficacia y mejora de su SGC, sustentado en la Planeación Estratégica, Liderazgo y la Calidad del Proceso Educativo, conforme a la norma NMX-CC-9001-IMNC-2015/ISO 9001:2015.

DESCRIPCIÓN DEL ÁREA DE NEGOCIOS.

Este proyecto se realizó dentro del laboratorio de instrumentación de la carrera de electrónica donde se obtuvieron a préstamo los materiales necesarios incluyendo la impresora 3D, estos facilitados por el jefe de laboratorio y el casetero con el que se cuenta.

Uno de los elementos más importantes a considerar en la formación educativa es el aspecto práctico, que, manejado en forma integral, a la par con la teoría, nos proporciona resultados óptimos, lo que, a su vez, facilita que el alumno se integre a su área de trabajo en el momento de su egreso.

El objetivo primordial de los laboratorios es facilitar las herramientas necesarias, para la formación en el aspecto práctico del alumno, proporcionando espacios adecuados de trabajo, equipos, instrumentos, etc., para la elaboración de sus prácticas.

El Tecnológico de CD Guzmán anteriormente contaba con la carrera de Ingeniería Electrónica en Instrumentación por lo cual se tenía un laboratorio dedicado a esta área; a partir de la reestructuración de los planes de estudio como resultado de la Reforma Educativa en 1993 se cambió a la carrera de Ingeniería Electrónica con la especialidad en Instrumentación y Control de Procesos, debido a la demanda y crecimiento en el alumnado se consiguió la construcción del espacio físico que satisficiera dichas necesidades, por tal motivo se construyó un laboratorio de electrónica de dos niveles el cual fue entregado en 1993 aunque en ese momento sólo fueron las instalaciones más no así el equipamiento. La distribución planeada para este laboratorio contempla las áreas dedicadas a la electrónica analógica, electrónica digital, electrónica de potencia, cómputo, área de investigación, sección de cubículos, baños, sala audiovisual, oficina y dirección.

Capítulo *IV*: Descripción de las actividades desarrolladas.

ACTIVIDADES DEL PROYECTO.

En el laboratorio de electrónica no se cuenta con un sistema real en el cual se pueda aplicar un control PID y así poder observar las diferentes características de estos tales como la velocidad de respuesta entre otros parámetros. La idea de realizar un sistema ball and plate nace tras esta necesidad, y tomando en cuenta que los equipos que usualmente se necesitan o se cuenta con ellos resultan muy caros y de acceso restringido.

El sistema ball and plate al resultar relativamente pequeño y fácil de construir ya que consta de tres piezas las cuales se pueden realizar mediante la impresión en 3D con material PLA (Ácido Poliláctico), y el resto de las piezas se realizó por medio de acrílico el cual solo se recortaron a la medida deseada.

El modelado se del sistema se realizó con base en las transformaciones lineales teles como rotación y traslación.

El estudio y comprensión del lenguaje de programación Python fue algo estrictamente necesario ya que la Raspberry trabaja con dicho lenguaje por los que se tuvo que comprender la sintaxis para poder entender y redactar código.

La instalación se llevó a cabo en una serie de pasos a continuación se mencionan:

- Instalación de OS (Operative System)

1. Descargar el sistema operativo (ya sea en formato .zip o .img).
2. Descargar el programa Etcher el cual nos servirá para cargar la imagen en la tarjeta microSD.
3. Insertar la tarjeta microSD en un lector para posteriormente insertarla a un pc y ejecutar Etcher en la misma.
4. Seleccionar el archivo de donde se encuentra el OS, seleccionar la tarjeta SD, dar click en “Flash” para comenzar la escritura de los archivos en la tarjeta SD.
5. Una vez que termina inserta la tarjeta microSD a la Raspberry con los diferentes accesorios conectados (mouse, teclado, monitor y cable ethernet)

6. Conectar el cable de alimentación a la Raspberry para que encienda y se instale el sistema operativo.
7. Seguir la instalación como cualquier OS.

- Mediciones con la pantalla

Para identificar el funcionamiento de la pantalla se realizaron medicines tales que con un multímetro medir el valor de resistencia que arroja si se ejerce presión en cada una de las esquinas como se muestra en la Figura 4-1, e identificar cual es el comportamiento para tomar un punto de referencia y generar un sistema de coordenadas tomando un cero, cero como origen y siendo el valor mínimo de resistencia.

Figura 4- 1: Medición de la resistencia en la pantalla

Dado que la Raspberry no cuenta con entradas analógicas fue necesario implementar dispositivo intermedio, en este caso fue un Arduino, para esto fue necesario instalar dicho software dentro de la Raspberry, con el cual se leen los datos analógicos, y se mapean a una resolución establecida como se muestra en la Figura 4-2 y 4-3. (código completo ver anexo 1)

Instalación de Arduino.

- De la página web oficial de Arduino se descargó la ID. Esto realizado desde la Raspberry.

- Se extraen los archivos.
- Posteriormente se abre la ventana terminal y se posiciona en la carpeta de descargas, la carpeta que se creó al extraer los archivos (se escribe cd Downloads, cd “carpeta que se creó al extraer los archivos”).
- Posteriormente se escribe “./instal.sh” si esto no funciona se escribe “sudo ./instal.sh”
- Una vez que termine buscas en los programas de Raspberry lo ejecutas y si todo va bien ya está eso esto todo lo que se realiza.

```
int get_x_value(void){
  int x_pos;
  pinMode(Y1,INPUT);
  pinMode(Y2,INPUT);
  digitalWrite(Y2,LOW);

  pinMode(X1,OUTPUT);
  digitalWrite(X1,HIGH);
  pinMode(X2,OUTPUT);
  digitalWrite(X2,LOW);

    #ifdef SCREEN_WO_RESOLUTION
      x_pos=analogRead(Y1);
    #else
      x_pos=map(analogRead(Y1),MIN_X_VALUE,MAX_X_VALUE,0,X_RESOLUTION);
      x_pos = x_pos < 0?0:x_pos;
      x_pos = x_pos > X_RESOLUTION?X_RESOLUTION:x_pos;
    #endif
```

Figura 4- 2: Obtención del eje X de la pantalla

```
int get_y_value(void){
  int y_pos;

  pinMode(X1,INPUT);
  pinMode(X2,INPUT);
  digitalWrite(X2,LOW);
  pinMode(Y1,OUTPUT);
  digitalWrite(Y1,HIGH);
  pinMode(Y2,OUTPUT);
  digitalWrite(Y2,LOW);
     #ifdef SCREEN_WO_RESOLUTION
       y_pos = analogRead(X1);
     #else
       y_pos=map(analogRead(X1),MIN_Y_VALUE,MAX_Y_VALUE,0,Y_RESOLUTION);
     #endif

  #ifdef SCREEN_WO_RESOLUTION
    y_range[0] = y_pos < y_range[0]?y_pos:y_range[0];
    y_range[1] = y_pos > y_range[1]?y_pos:y_range[1];
    y_pos = y_pos < 0?0:y_pos;
    y_pos = y_pos > Y_RESOLUTION?Y_RESOLUTION:y_pos;
  #endif
  return y_pos;
```

Figura 4- 3: Obtención del eje Y de la pantalla

Una vez leídos estos valores son enviados a la Raspberry por medio del código que se muestra en la Figura 4-4.

```
  case 'w'://Raspberry request screen position
  reset_filter();
  counter_out=0;
  #ifdef SCREEN_WO_RESOLUTION
    reset_range_values();
  #endif

    if(bufferSize == 2){
      for(int i = 0;i<Numbers[0];i++){
        screen_pos[0]=get_x_value();
        screen_pos[1]=get_y_value();
        UART_PORT.println("p"+String(screen_pos[0])+","+String(screen_pos[1]));
        delay(Numbers[1]);
      }
      #ifdef SCREEN_WO_RESOLUTION
        UART_PORT.println("x_min = "+String(x_range[0])+" ,x_max = "+String(x_range[1]));
        UART_PORT.println("y_min = "+String(y_range[0])+" ,y_max = "+String(y_range[1]));
      #endif
    }
    else{
      UART_PORT.println("Buffer length doesn't match");
    }
  break;
```

Figura 4- 4: Envió de posición de la pantalla

Una vez mandado los datos es necesario decodificarlos en Python (código completo ver anexo 2), para eso se utiliza la librería serial, para estar seguros de que los datos llegaron de manera correcta se hizo uso de expresiones regulares como se muestra en la Figura 4-5.

```
if arduino.inWaiting() > 0:
    ##Read the serial port
    serial_input = arduino.readline()
    message = serial_input
    serial_input = serial_input[:-2].decode('utf-8') #Remove \n of expression

    matcher = re.compile(r'[A-Za-z][-]?[0-9]+([,][-]?[0-9]+)*$')
    #Verify the input
    if matcher.match(serial_input):
        char = serial_input[0]
        digits = serial_input[1:]
```

Figura 4- 5: Expresiones regulares

Para probar el correcto funcionamiento de la pantalla se creó una interfaz gráfica (Figura 4-6) en la cual se muestra los valores brutos que entrega la pantalla y los valores con el mapeo que se realizó en el código.

Figura 4- 6: Interfaz grafica

- Diseño de la plataforma

Para realizar el diseño se utilizó un software de modelado en 3D, el diseño se separó en tres piezas las cuales fueron fabricadas mediante una impresora 3d la cual usa PLA para llevar a cabo dicha impresión:

la primera pieza fue la base para la cual se tomó como referencia un hexágono irregular, la distribución fue de tal como se aprecia en la Figura 4-7.

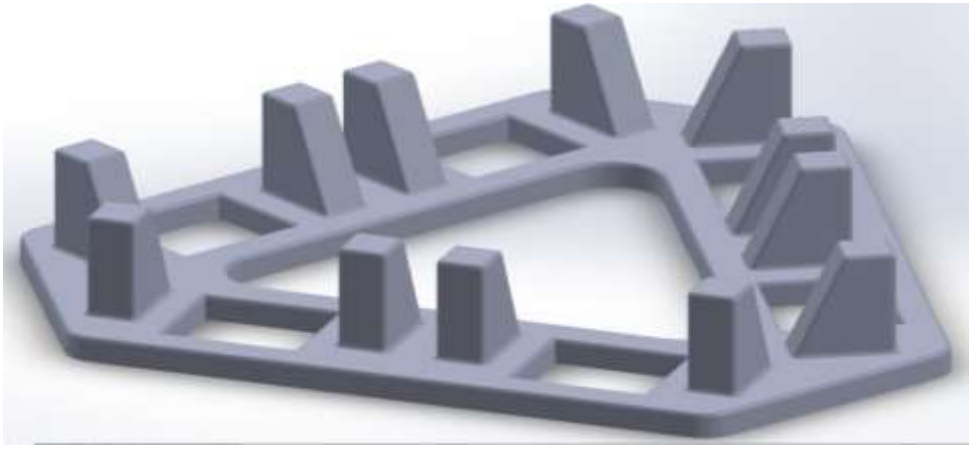

Figura 4- 7: Pieza 1

La segunda pieza, fue un soporte que conecta por medio de los tie rods (extensión con puntas móviles) con los servos, dicha pieza se puede observar en la Figura 4-8.

Figura 4- 8: Pieza 2

Tercera pieza: esta pieza fue una base en la cual está sentada un plato de acrílico, a su vez esta encaja con la pieza dos formando ya la estructura completa, esta pieza se puede observar en la Figura 4-9.

Figura 4- 9: Pieza 3

Una vez que contamos con todas las piezas se forma la estructura completa la cual se puede apreciar en la Figura 4-10.

Figura 4- 10: Estructura completa

- Controladora PCA9685

Para poder utilizar la controladora de servos fue necesario instalar una librería “Adafruit” esto lo realizamos tecleando en la ventana terminal los siguientes comandos:

- sudo pip3 install adafruit-circuitpython-motor
- sudo pip3 install adafruit-circuitpython-pca9685

Una vez terminado el proceso se realizaron pruebas para identificar como es el que trabaja, corroborar que funcione correctamente, estas pruebas fueron tanto con un servo como con múltiples. Solo que previamente se realizó una prueba con la pura Raspberry y el servo la cual

La cual consistió en mover un servo con l Raspberry utilizando la controladora, para esto se realizó un programa el cual se aprecia en la Figura 4-11, el cual consiste en importar las librerías, configurar los pines de la GPIO para que actúen como salida, para que mande los datos al servo, con datos me refiero al ancho del pulso el cual se especifica mediante del duty cicle (siclo de trabajo).

```
import RPi.GPIO as g
import time as t

g.setwarnings(False)
g.setmode(g.BOARD)
g.setup(7,g.OUT)
sv=g.PWM(7,50)

sv.start(7.5)
print("Posicion inicial")
t.sleep(2)

for i in range(5):
       sv.ChangeDutyCycle(2.5)
       print("Posicion -90")
       t.sleep(2)
       sv.ChangeDutyCycle(7.5)
       print("Posicion 0")
       t.sleep(2)
       sv.ChangeDutyCycle(12.5)
       print("Posicion 90")
       t.sleep(2)
       sv.ChangeDutyCycle(7.5)
       print("Posicion 0")
       t.sleep(2)
sv.ChangeDutyCycle(7.5)
print("Posicion final")
print("Programa finalizado")
sv.stop()
g.cleanup()
```

Figura 4- 11: Programa para controlar un servo

Las pruebas que se realizaron con múltiples servos fueron de tal manera que la controladora cuenta con 16 canales los cuales nos sirven para mover los servos ya que por ahí reciben la alimentación y la señal del PWM, en la Figura 4-12 se muestra el programa que se realizó, en este se aprecia el uso de los canales donde especifica por medio del cual se manda la información.

```
from board import SCL, SDA
import busio

from adafruit_pca9685 import PCA9685
from adafruit_motor import servo

import time as t

i2c = busio.I2C(SCL, SDA)
pca = PCA9685(i2c)

pca.frequency = 50

a=800          #Pulso minimo
b=2700         #Pulso maximo

ser0 = servo.Servo(pca.channels[ 0], min_pulse=a, max_pulse=b)
ser1 = servo.Servo(pca.channels[ 1], min_pulse=a, max_pulse=b)
ser2 = servo.Servo(pca.channels[ 4], min_pulse=a, max_pulse=b)
ser3 = servo.Servo(pca.channels[ 5], min_pulse=a, max_pulse=b)
ser4 = servo.Servo(pca.channels[ 6], min_pulse=a, max_pulse=b)
ser5 = servo.Servo(pca.channels[ 7], min_pulse=a, max_pulse=b)

m=180
p1=135
p2=170
p3=90
ser0.angle = p3
t.sleep(0.5)
ser1.angle = m - p3
t.sleep(0.5)
ser2.angle = p1
t.sleep(0.5)
ser3.angle = m - p3
t.sleep(0.5)
ser4.angle = p2
t.sleep(0.5)
ser5.angle = m - p1
t.sleep(0.5)
```

Figura 4- 12: Programa para controlar seis servos

- Modelado

Para comenzar la base de nuestra plataforma resultar ser un hexágono regular tal como se aprecia en la figura 4-13, donde los puntos que se aprecian son los puntos de rotación de los servos, al ser un hexágono la distancia de cada punto es de 60° por lo que se calcularon mediante la ecuación que se aprecia en la Figura 4-14.

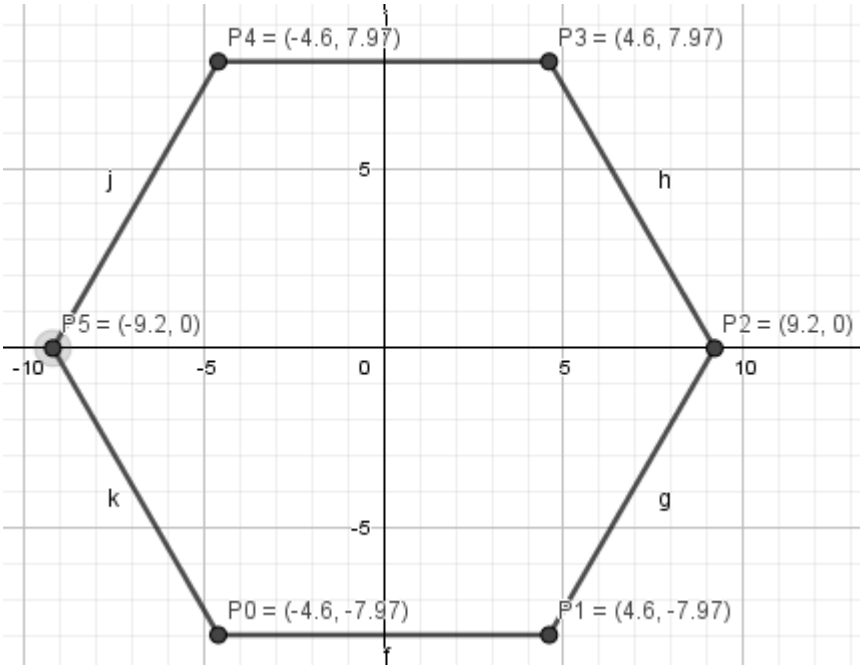

Figura 4- 13: Distribución de los ejes de rotación de cada servo

```
def base_points(radio):
    points =[]
    for angle in [240,300,0,60,120,180]:
        points.append([radio*math.cos(math.radians(angle)),radio*math.sin(math.radians(angle)),0])
    return points
```

Figura 4- 14: Función para calcular los puntos del hexágono mediante un radio especificado

Como se mencionó anteriormente, por medio de las transformaciones lineales podemos trasladar (Figura 4-15) o rotar (Figura 4-16) los punto cierto ángulo o desplazamiento requerido, para simplificar estos cálculos se realizó una función en Python la cual nos retorna los puntos modificados.

```python
def position_translate(position,delta):
    """
    ---Translate an specific point ---
    [x,y,z,1][1  0  0  1 = [x+dx,y+dy,z+dz,1]
             0  1  0  0
             0  0  1  0
             dx dy dz 0]
     position_translate(position,delta)
    """
    position.append(1) #Add to the enteredosition 1, we need it to multiply matrix
    pos_mat = np.matrix(position)


    translation_matrix = np.matrix([[1,0,0,1],[0,1,0,0],[0,0,1,0],[delta[0],delta[1],delta[2],0]])

    new_pos = position*translation_matrix
    new_pos = new_pos[0,:-1]

    position.pop()
    return new_pos
```

Figura 4- 15: Función de traslación

```python
def position_rotate(position,angles):
    """
    ---Rotate an specific point by the axis ---
    position_rotate(position,angles)
    position_rotate([x,y,z],[yaw,pitch,roll])
    Example:
       Rotate [0,0,1] by yaw 90 degrees
       position_rotate([0,0,1],[90,0,0])
    """
    mat_point = np.matrix(position)
    for i in range(len(angles)):
        angles[i] = math.radians(angles[i])

    rot_roll  = np.matrix([[1,0,0],[0,math.cos(angles[2]), -math.sin(angles[2])],[0,math.sin(angles[2]),math.cos(angles[2])]])
    rot_pitch = np.matrix([[math.cos(angles[1]),0,-math.sin(angles[1])],[0,1,0],[math.sin(angles[1]),0,math.cos(angles[1])]])
    rot_yaw   = np.matrix([[math.cos(angles[0]),-math.sin(angles[0]),0],[math.sin(angles[0]),math.cos(angles[0]),0],[0,0,1]])
    return mat_point*rot_roll*rot_pitch*rot_yaw
```

Figura 4- 16: Función de rotación

Como calcular los puntos del plato

Tomando en cuenta una cara de la plataforma donde se encuentran un par de puntos alineados con uno de los lados del plato posteriormente se calculan los puntos que van del centro del plato a la unión con la base, una vez que se trinen estos se aplica nuevamente una rotación y un recalculo para los siguientes dos pares de puntos, esta acción se aprecia en la Figura 4-17.

```
def plate_points(centroid_dist,scrap,euler_angles,translation):
    #Make the point for the translation
    point = [0,-centroid_dist,0]

    points=[]
    rot_angles = [0,-120,-240]
    for side_angle in rot_angles:
        anticlock = position_translate(point,[-(scrap/2),-scrap,0]).tolist()[0]
        anticlock = position_rotate(anticlock,[side_angle,0,0]).tolist()[0]
        anticlock = position_rotate(anticlock,[euler_angles[0],euler_angles[1],euler_angles[2]]).tolist()[0]
        anticlock = position_translate(anticlock,translation).tolist()[0]
        #print("anticlock = {}".format(anticlock))
        points.append(anticlock)
        clock = position_translate(point,[(scrap/2),-scrap,0]).tolist()[0]
        clock = position_rotate(clock,[side_angle,0,0]).tolist()[0]
        clock = position_rotate(clock,[euler_angles[0],euler_angles[1],euler_angles[2]]).tolist()[0]
        clock = position_translate(clock,translation).tolist()[0]
        #print("clock = {}".format(clock))
        points.append(clock)
    return points
```

Figura 4- 17: Calculo de los puntos del plato

Para lograr un mejor entendimiento de la plataforma se recurrió a realizar la gráfica de los puntos del plato, esto se aprecia en la Figura 4-18, también se incluyó la gráfica de los puntos de la base esto se muestra en la Figura 2-19. Una vez que se contó con esto puntos se usó la función de la Figura 2-20 para realizar el dibujo de los ejes, posterior mente en la Figura 2-21 se muestra dicho dibujo.

```
158
159 def points_to_xyz(points):
160     x = []
161     y = []
162     z = []
163     for i in points:
164         x.append(i[0])
165         y.append(i[1])
166         z.append(i[2])
167     return x,y,z
168
169 def draw_by_points(points,ax,fig,color):
170     x,y,z = points_to_xyz(points)
171
172     x.append(x[0])
173     y.append(y[0])
174     z.append(z[0])
175
176     ax.plot3D(x,y,z,c=color)
177     #fig.canvas.draw()
178
```

Figura 4- 18: Método para graficar de los puntos de plato

```
174                 end_servo = bf.set_servo_values(servos_value,min_signal_degree,max_sig
175
176             if plot:
177                 plt.cla()
178                 bf.draw_axis(110,110,220,ax,fig)
179
180                 bf.draw_by_points(base_points,ax,fig,'orangered')
181                 bf.draw_by_points(plate_points,ax,fig,'dodgerblue')
182
183                 bf.draw_servo(base_points,plate_points,servo_links[0],theta1,ax,fig)
184                 fig.canvas.draw()
185             sleep(delay)
```

Figura 4- 19: Método para graficar de los puntos de la base

```python
def draw_axis(axis_x,axis_y,axis_z,ax,fig):
    ax.set_xlim(-axis_x,axis_x)
    ax.set_ylim(-axis_y,axis_y)
    ax.set_zlim(0,axis_z)

    x = [-axis_x,0]
    y = [0,0]
    z = [0,0]
    ax.plot3D(x,y,z,'--r')
    #fig.canvas.draw()
    x = [0,axis_x]
    y = [0,0]
    z = [0,0]
    ax.plot3D(x,y,z,'r')
    #fig.canvas.draw()
    x = [0,0]
    y = [-axis_y,0]
    z = [0,0]
    ax.plot3D(x,y,z,'--b')
    #fig.canvas.draw()
    x = [0,0]
    y = [0,axis_y]
    z = [0,0]
    ax.plot3D(x,y,z,'b')
    #fig.canvas.draw()
    x = [0,0]
    y = [0,0]
    z = [0,axis_z]
    ax.plot3D(x,y,z,'g')
    #fig.canvas.draw()
```

Figura 4- 20: Función para graficar los ejes

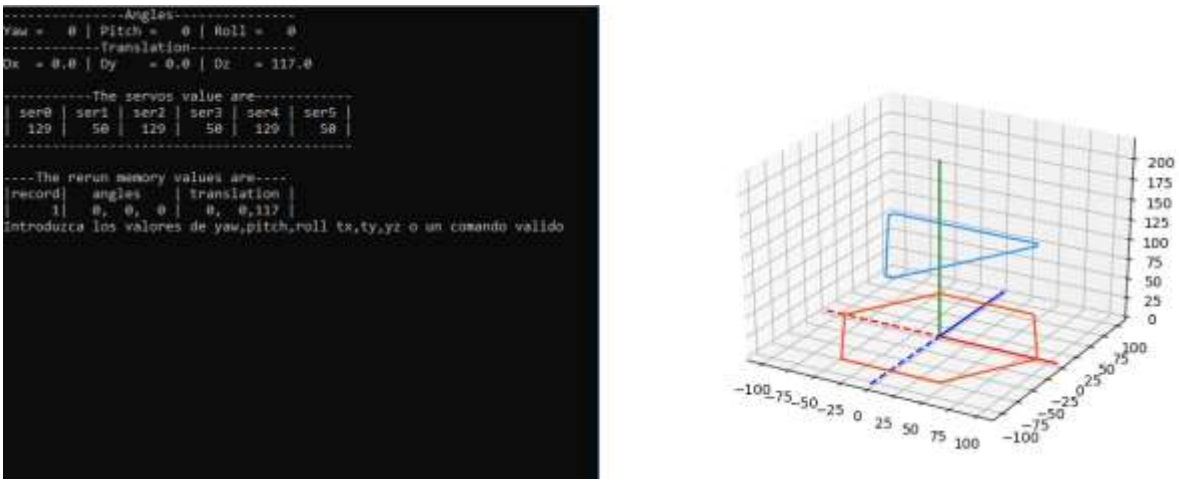

Figura 4- 21: Grafica de los puntos de la base y del plato, como de los ejes

Para calcular con ángulos de los servos fue necesario aplicar cinemática inversa, nuestro modelo pudo ser resuelto de la siguiente manera.

Considerando las ecuaciones siguientes:

$$Length2h = \sqrt{length2^2 - y2^2} \quad (1)$$

$$\cos\theta_2 = (x2^2 + z2^2 - length1^2 - length2h^2) / (2length1\ length2h) \quad (2)$$

$$\sin\theta_{21} = \sqrt{(1 - \cos\theta_2{}^2)} \quad (3)$$

$$\sin\theta_{22} = -\sqrt{(1 - cos\theta 2^2)} \quad (4)$$

$$\theta_{21} = \tan^{-1}\left(\frac{\sin\theta_{21}}{\cos\theta_2}\right) \quad (5)$$

$$\theta_{22} = \tan^{-1}\left(\frac{\sin\theta_{22}}{\cos\theta_2}\right) \quad (6)$$

Posibles soluciones para el ángulo de los servos:

$$\theta_{11} = \tan^{-1}\left(\frac{z_2}{x_2}\right) - \tan^{-1}\left(\frac{length2h\ \sin\theta_{21}}{length1+length2h\ \cos\theta_2}\right) \quad (7)$$

$$\theta_{12} = \tan^{-1}\left(\frac{z_2}{x_2}\right) - \tan^{-1}\left(\frac{length2h\ \sin\theta_{22}}{length1+length2h\ \cos\theta_2}\right) \quad (8)$$

$$x_{11} = length1\cos(\theta_{11}) \quad (9)$$

$$z_{11} = length1\sin(\theta_{11}) \quad (10)$$

$$x_{12} = length1\cos(\theta_{12}) \quad (11)$$

$$z_{12} = length1\sin(\theta_{12}) \quad (12)$$

En la Figura 4-22 se explica la relación que se tiene de un eje de rotación de un servo y su conexión con el soporte del plato, esto para encontrar el ángulo que debe de tener el eje del servo para llegar a un punto deseado en el soporte del plato.

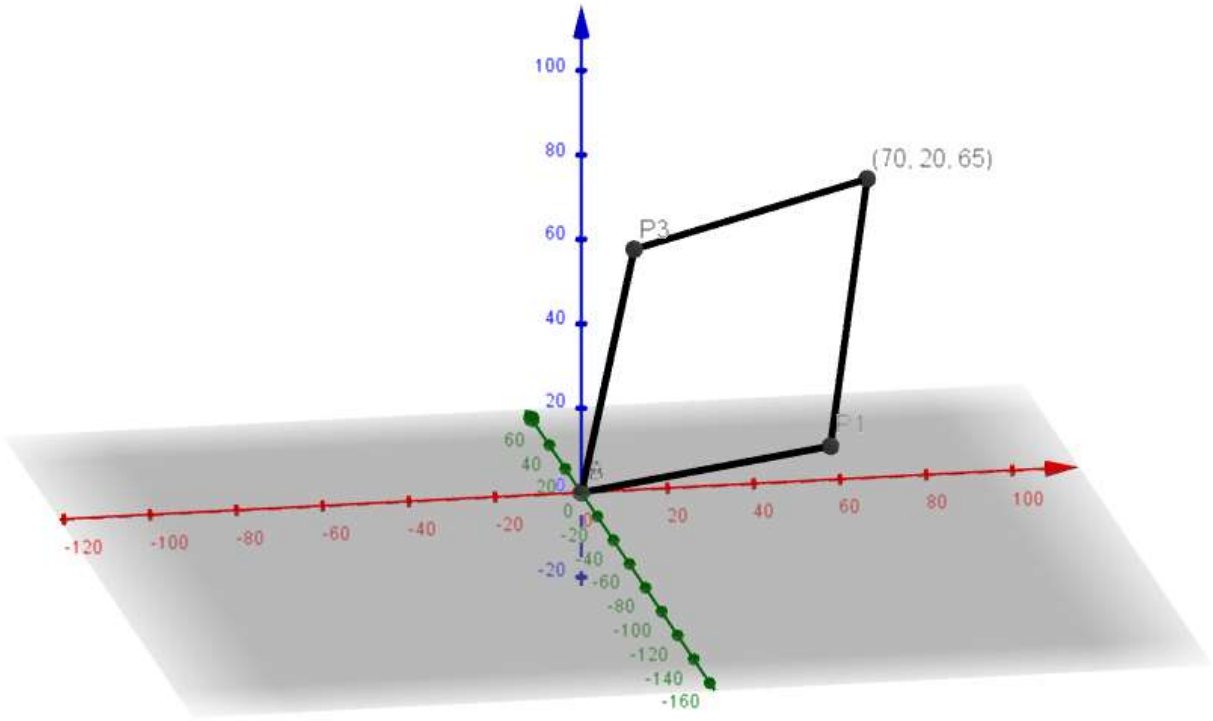

Figura 4- 22: Relación de un punto de la plataforma con el ángulo supuesto del servo

Posteriormente se aplicó el mismo principio para los seis servos, véanse de la Figura 4-23 a Figura 4-26. dicha acción realiza los cálculos necesarios. La parte del código que realiza la gráfica se muestra de la Figura 4-27 a la Figura 4-28

```
def get_servo_angle(position_input,links_length,base_input):
    if len(links_length) != 2:
        print("You should put only two links_length")
    if len(position_input[0]) != 3 or len(base_input[0]) != 3:
        print("This function is only available for 3 DOF with 2 links_length")

    theta1=[]
    theta2=[]

    current_point=0
    side_angles = [0,120,240]

    for current_angle in side_angles:
        #print("--- Point {}---".format(current_point))
        #print("Angle = {}".format(current_angle))
        position = position_rotate(position_input[current_point],[current_angle,0,0]).tolist()[0]
        base = position_rotate(base_input[current_point],[current_angle,0,0]).tolist()[0]

        #print("Plate {}".format(position))
        #print("Base {}".format(base))
        #print("Before translate {}".format(position))
        position = position_translate(position,[-base[0],-base[1],-base[2]]).tolist()[0]
```

Figura 4- 23: Calculo para 6 servos, parte 1

```
help_link = math.sqrt(math.pow(links_length[1],2)-math.pow(position[1],2))

costheta2 = (math.pow(position[0],2)+math.pow(position[2],2)-math.pow(links_length[0],2)-math.pow(help_link,2))/(2*links_length[0]*help_link)

sentheta2 = [-math.sqrt(1 - math.pow(costheta2,2)),math.sqrt(1 - math.pow(costheta2,2))]

theta2_c = []
theta1_c = []

theta2_c = [math.degrees(math.atan(sentheta2[0]/costheta2)),math.degrees(math.atan(sentheta2[1]/costheta2))]

theta1_c = [math.atan(float(position[2])/float(position[0]))-math.atan((help_link*sentheta2[0])/(links_length[0]+help_link*costheta2))]
theta1_c.append(math.atan(float(position[2])/float(position[0]))-math.atan((help_link*sentheta2[1])/(links_length[0]+help_link*costheta2)))
theta1_c[0] = math.degrees(theta1_c[0])
theta1_c[1] = math.degrees(theta1_c[1])

if position[0] < 0:
    theta1_c[0] = theta1_c[0]+180
    theta1_c[1] = theta1_c[1]+180
```

Figura 4- 24: Calculo para 6 servos, parte 2

```
theta1.append(theta1_c)
theta2.append(theta2_c)
current_point+=1
###antipoint
#print("--- Point {}---".format(current_point))
#print("Angle = {}".format(current_angle))
position = position_rotate(position_input[current_point],[current_angle,0,0]).tolist()[0]
base = position_rotate(base_input[current_point],[current_angle,0,0]).tolist()[0]

#print("Plate {}".format(position))
#print("Base {}".format(base))
#print("Before translate {}".format(position))
position = position_translate(position,[-base[0],-base[1],-base[2]]).tolist()[0]
#print("After translate {}".format(position))

help_link = math.sqrt(math.pow(links_length[1],2)-math.pow(position[1],2))

costheta2 = (math.pow(position[0],2)+math.pow(position[2],2)-math.pow(links_length[0],2)-math.pow(help_link,2))/(2*links_length[0]*help_link)

sentheta2 = [math.sqrt(1 - math.pow(costheta2,2)),-math.sqrt(1 - math.pow(costheta2,2))]

theta2_c = []
theta1_c = []
```

Figura 4- 25: Calculo para 6 servos, parte 3

```
    theta2_c = [math.degrees(math.atan(sentheta2[0]/costheta2)),math.degrees(math.atan(sentheta2[1]/costheta2))]

    theta1_c = [math.atan(float(position[2])/float(position[0]))-math.atan((help_link*sentheta2[0])/(links_length[0]+help_link*costheta2))]
    theta1_c.append(math.atan(float(position[2])/float(position[0]))-math.atan((help_link*sentheta2[1])/(links_length[0]+help_link*costheta2)))

    theta1_c[0] = math.degrees(theta1_c[0])
    theta1_c[1] = math.degrees(theta1_c[1])

    if position[0] < 0:
        theta1_c[0] = theta1_c[0]+180
        theta1_c[1] = theta1_c[1]+180
    theta1.append(theta1_c)
    theta2.append(theta2_c)
    current_point+=1

return theta1,theta2
```

Figura 4- 26: Calculo para 6 servos, parte 4

```
def draw_servo(base_points,plate_points,servos_length,angles,ax,fig):
    servo_points=[]
    rot_angle = [0,120,240]

    point = 0

    for side_angle in rot_angle:
        end_servo = []
        #print("Point {}".format(point))
        rotated_point = position_rotate(base_points[point],[side_angle,0,0]).tolist()[0]
        #print("Cross 1 - {}\nCross 2 - {}".format(angles[point][0],angles[point][1]))
        #Aca solucion de cos theta(2)
        end_servo.append([rotated_point[0]+servos_length*math.cos(math.radians(angles[point][0])),rotated_point[1],rotated_point[2]+servos_length*math.sin(math.radians(angles[point][0]))])
        end_servo.append([rotated_point[0]+servos_length*math.cos(math.radians(angles[point][1])),rotated_point[1],rotated_point[2]+servos_length*math.sin(math.radians(angles[point][1]))])
        end_servo[0] = position_rotate(end_servo[0],[-side_angle,0,0,]).tolist()[0]
        end_servo[1] = position_rotate(end_servo[1],[-side_angle,0,0,]).tolist()[0]
        servo_points.append(end_servo)
        #print("The length 1 is {}".format(two_points_length(servo_points[point][0],base_points[point])))
        #print("The length 2 is {}".format(two_points_length(servo_points[point][1],base_points[point])))
```

Figura 4- 27: Código para graficar, parte 1

```
point+=1
end_servo = []
rotated_point = position_rotate(base_points[point],[side_angle,0,0]).tolist()[0]
end_servo.append([rotated_point[0]+servos_length*math.cos(math.radians(angles[point][0])),rotated_point[1],rotated_point[2]+servos_length*math.sin(math.radians(angles[point][0]))])
end_servo.append([rotated_point[0]+servos_length*math.cos(math.radians(angles[point][1])),rotated_point[1],rotated_point[2]+servos_length*math.sin(math.radians(angles[point][1]))])
end_servo[0] = position_rotate(end_servo[0],[-side_angle,0,0,]).tolist()[0]
end_servo[1] = position_rotate(end_servo[1],[-side_angle,0,0,]).tolist()[0]
servo_points.append(end_servo)
point+=1
```

Figura 4- 28: Código para graficar, parte 2

```
for i in range(6):
    ax.plot3D([base_points[i][0],servo_points[i][0][0]],[base_points[i][1],servo_points[i][0][1]],[base_points[i][2],servo_points[i][0][2]],'k')
    ax.plot3D([plate_points[i][0],servo_points[i][0][0]],[plate_points[i][1],servo_points[i][0][1]],[plate_points[i][2],servo_points[i][0][2]],'k')
    ax.plot3D([base_points[i][0],servo_points[i][1][0]],[base_points[i][1],servo_points[i][1][1]],[base_points[i][2],servo_points[i][1][2]],':k')
    ax.plot3D([plate_points[i][0],servo_points[i][1][0]],[plate_points[i][1],servo_points[i][1][1]],[plate_points[i][2],servo_points[i][1][2]],':k')
```

Figura 4- 29: Código para graficar, parte 3

Capítulo *V*: Resultados obtenidos.

RESULTADOS.

Una vez que se realizó el diseño y las piezas de imprimieron se comenzó con la construcción montando en una primera instancia los servos en la base como se aprecia en la Figura 5-1.

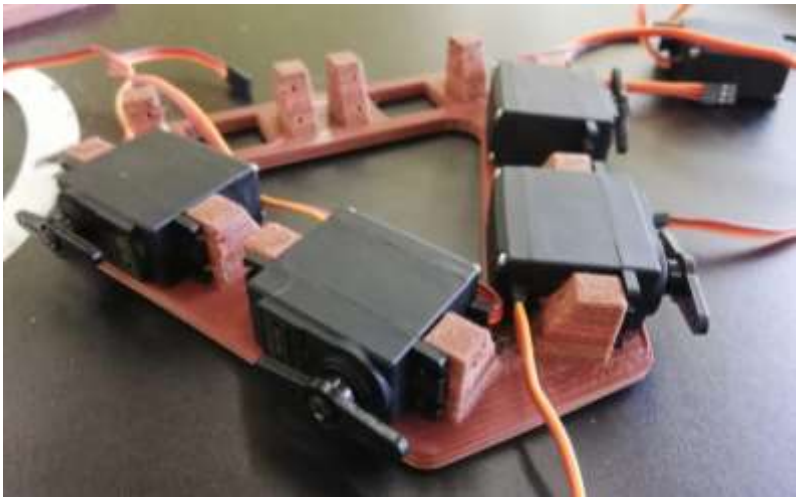

Figura 5- 1: Montura de servos en la base

Posteriormente se conectaron los tie ruds al eje de rotación de servo y se sujetaron al soporte para la base del plato como se aprecia en la Figura 5-2.

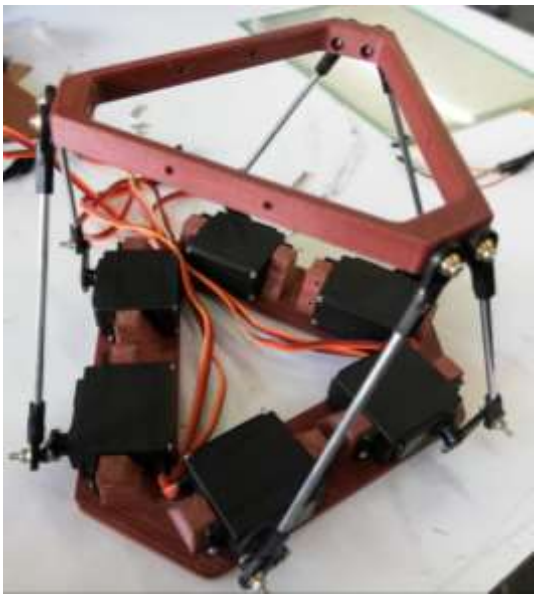

Figura 5- 2: Ensamblaje de tire ruds y soporte para la base del plato

En la Figura 5-3 se muestra la base para el plato y el plato ya montado y a su vez estos ensamblados al resto de la estructura. en la figura 5-4 se puede apreciar el resultado del cómo queda constituida la plataforma y el plato finalmente.

Figura 5- 3: Diseño y construcción final

Ya con la plataforma se comenzó con el modelado obteniendo lo siguiente.

En la Figura 5-4 se muestra el modelado tomado en cuenta los ejes de rotación de los servos, también se observa el soporte para el plato y la base, las líneas punteadas que se aprecian resultan de la cinemática inversa ya que para llegar a ese punto se puede tomar cualquiera de los dos caminos mientras en el resto sea igual para simplificar más el trabajo ya que de no ser iguales el resto de los cálculos se volverían más complicados.

Esta misma Figura muestra la base y el soporte del plato.

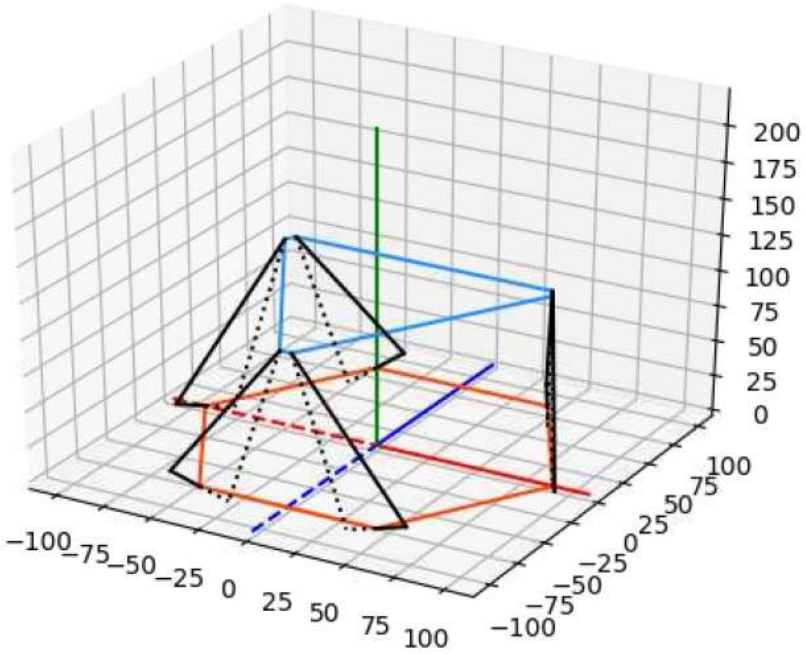

Figura 5- 4: Grafico representativo de la base y el plato

Una vez hecho esto se procedió a que a partir de la simulación obtener los ángulos que deben tener cada servo para un punto especifico del plato, lo cual se aprecia en la Figura 5-5 y 5-6 las cuales representan dos puntos distintos del plato, con la representación gráfica del mismo.

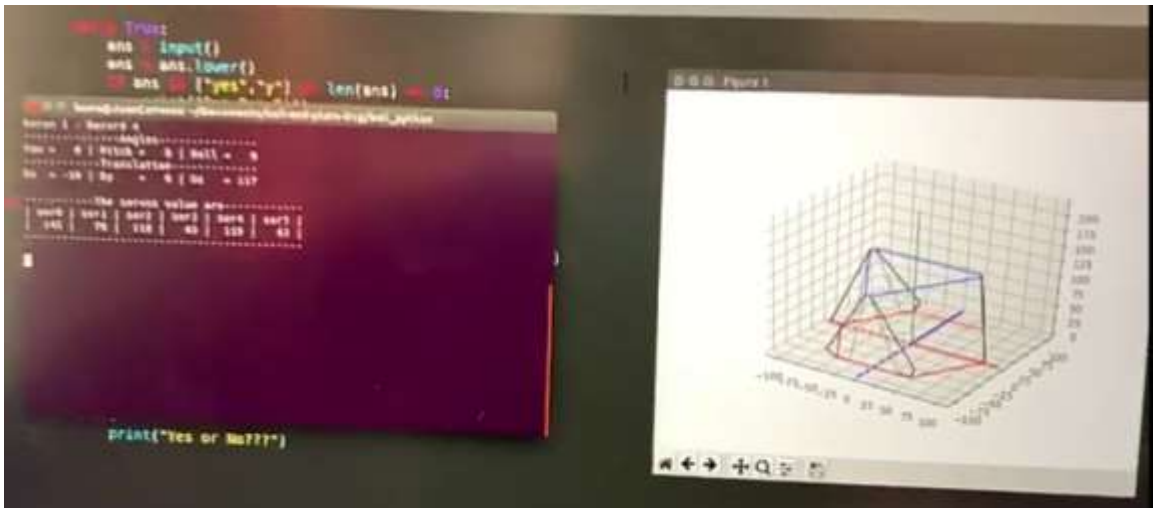

Figura 5- 5: Ángulos de los servos y representación gráfica (1)

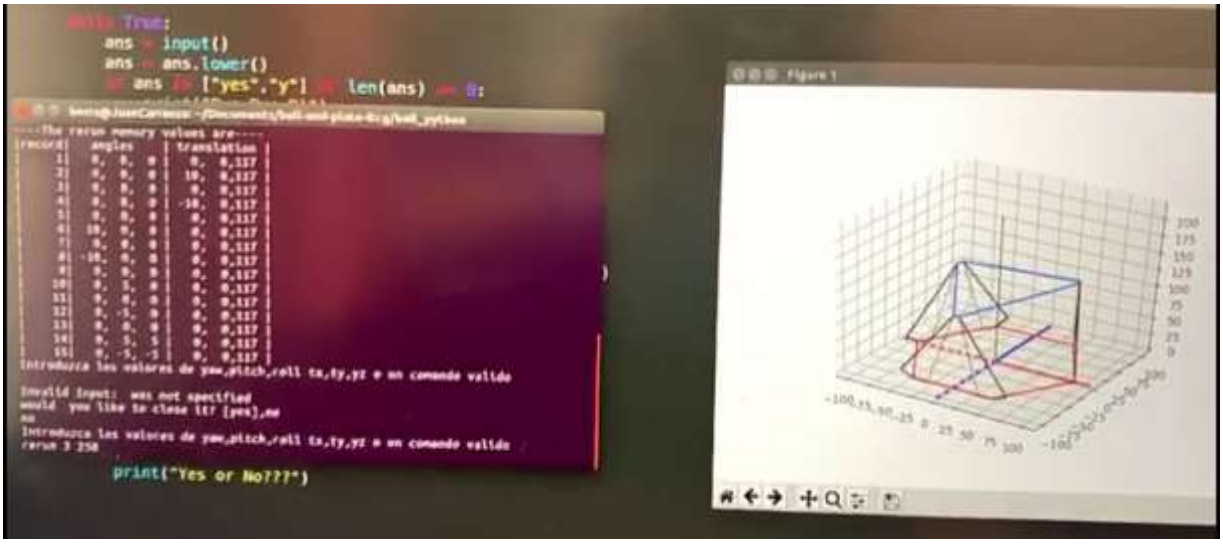

Figura 5- 6: Ángulos de los servos y representación gráfica (2)

Las Figuras 5-7 a 5-9 muestran los instantes en los cuales por medio de un telefono movil con el cual se aprecian los grados de inclinacion del plato durante las pruevas realizadas.

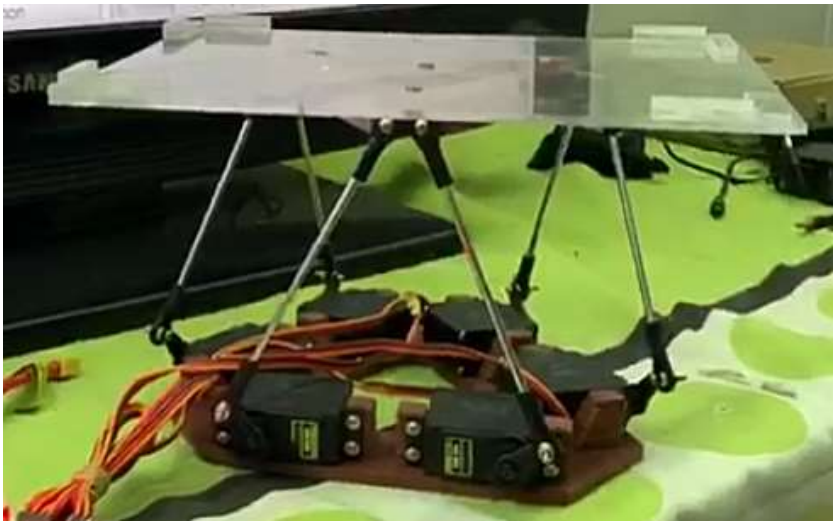

Figura 5- 7: Prueba 1

Figura 5- 8: Prueba 2 (a)

Figura 5- 9: Prueba 2 (b)

RESPUESTA A LA HIPÓTESIS

El diseño está basado en una plataforma Stewart con unas ligeras modificaciones, tanto en el acomodo de los ejes como variantes en la forma de la base.

En la Figura 5-10 se aprecia la superficie en la cual se apoya el plato y al igual los puntos dende se conecta a los servomotores, dada la distribución de los puntos dende se conecta a los servomotores generan una mayor cantidad de puntos en donde se transmite el movimiento.

Al haber una mayor cantidad superficie de contacto con el pato se genera una mayor estabilidad, esto se ve reflejado al momento de realizar movimiento puesto que resulta ser más suave, con esto se hace referencia al echo de que al moverse el plato se aprecia una menor turbulencia (en las Figuras 5-7 a 5-9 se muestran capturas de las pruebas de movimiento realizadas) sobre el mismo, en relación con un modelo convencional que solo cuenta con dos puntos móviles

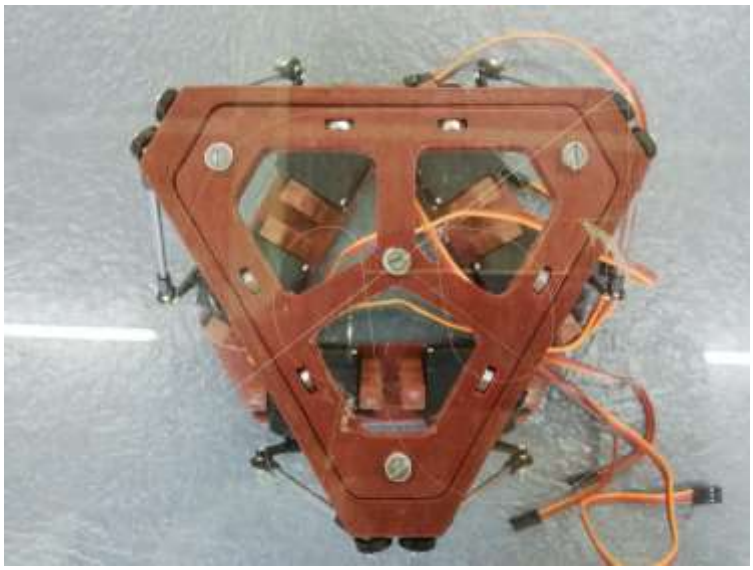

Figura 5- 10: superficie de apoyo del plato

Capítulo *VI*: Alcances y limitaciones.

ALCANCES.

La institución ahora cuenta con un sistema en el cual los alumnos podrán implementar sus controladores PID (MIMO) desarrollados en clase y observar las características de estos en el mundo real, tomando en cuenta que anteriormente solo se limitaba a conceptos teóricos. Quedando a su vez como base para poder realizar mejoras.

LIMITACIONES.

La entrega tardía de los materiales por parte de los proveedores.

El no encontrar los tie ruds, con esto me refiero a que los proveedores los incluían en su catálogo, pero no contaban con piezas disponibles

En nulo conocimiento en Python.

Capítulo *VII*: Conclusiones y recomendaciones.

CONCLUSIONES.

Con el término de este proyecto podemos rescatar los conocimientos adquiridos tanto en el manejo del lenguaje Python para crear código de programación. con el uso de una plataforma para el trabajo a distancia la cual se manejó para seguir trabajando fuera de la institución lo que resulto muy útil ya que no importaba el dónde se estuviera se podía seguir haciendo pruebas en cuanto a código para posteriormente solo aplicarlo físicamente.

Otro punto rescatable es el hecho de corroborar bien las versiones que van desde software y librerías utilizadas ya que de no ser iguales corría el riesgo de que no funcionase. Puesto que en trabajos previos solo solo me limitaba a trabajar con "x" software sin tomar en cuenta su versión

Mediante el diseño y construcción se utilizó software de diseño en 3D el cual aprendí a utilizar puesto que si bien sabia de dicho software no se me había presentado la necesidad de utilizarlo.

En cuanto al modelado se aplicaron diversas técnicas de transformaciones lineales y cinemática inversa que, si bien no desconocía el tema, jamás lo había trabajado y aplicado, esto conllevo a que tuviese un mayor conocimiento sobre el tema.

Un punto importante fue el hecho de trabajar en equipo, cosa que se aprendió durante todo el transcurso del proyecto, puesto que no sabía trabajar en equipo.

Bibliografía.

Challenger Pérez, I., Díaz Ricardo , Y., & Becerra García, R. A. (2014). El lenguaje de programacion Python. *Ciencias Holgín*.

Factory, G. (14 de Abril de 2015). *Geek Factory.* Obtenido de http://www.nxp.com/documents/data_sheet/PCA9685.pdf

Garzón Mancera, O. L., & Garzón Melo, Y. (Mayo de 2016). Diseño e implementación de una plataforma bola y plato, para la. Bogotá, Colombia.

González, A. (Diciembre de 2013). *Universidad Técnica Federico Santa.* Obtenido de http://www2.elo.utfsm.cl/~mineducagv/docs/ListaDetalladadeModulos/servos.pdf

I. Grossman, S. (2015). *Matematicas 4: Álgebra lineal.* McGraw-Hill.

Kumar Saha, S. (2010). *Itroducción a la robótica.* McGraw-Hill.

Marzal Varó, A., & Gracia Luengo, I. (2013). *Introducción a la programación con Python.*

Moreno Muño, A., & Córcoles Córcoles, S. (2017). En *Aprende Arduino en un fin de semana* (págs. 19-22).

Ogata, k. (2010). *Ingeniería de control moderna.* Pearson.

Anexos.

ANEXO 1.- SCRIPT DE ARDUINO

```
#include
"UART.h"
                #include "TOUCH_SCREEN.h"

                extern int x_range[2],y_range[2];
                extern int counter_out;
                void setup(){
                 uart_init();
                 reset_filter();
                }

                void loop(){
                 if(UART_PORT.available()){
                    char caracter;
                    int Numbers[20];
                    int bufferSize=0;
                    int screen_pos[2];

                     if(uart_get(&caracter,&bufferSize,Numbers)){
                       switch (caracter){
                         case 'f'://Raspberry request screen position
                           #ifdef SCREEN_WO_RESOLUTION
                             reset_range_values();
                           #endif

                             if(bufferSize == 2){
                               for(int i = 0;i<Numbers[0];i++){
                                 screen_pos[0]=get_x_value();
                                 screen_pos[1]=get_y_value();

                                 average_filter(screen_pos);
```

```
            if(screen_pos[0]                                    >=                                    0)
UART_PORT.println("p"+String(screen_pos[0])+","+String(screen_pos[1]));
            else                UART_PORT.println("p-1,-1");

            delay(Numbers[1]);
           }
           #ifdef SCREEN_WO_RESOLUTION
            UART_PORT.println("x_min = "+String(x_range[0])+" ,x_max = "+String(x_range[1]));
            UART_PORT.println("y_min = "+String(y_range[0])+" ,y_max = "+String(y_range[1]));
           #endif
          }
          else{
           UART_PORT.println("Buffer length doesn't match");
          }
         break;

         case 'w'://Raspberry request screen position
         reset_filter();
         counter_out=0;
         #ifdef SCREEN_WO_RESOLUTION
          reset_range_values();
         #endif

          if(bufferSize == 2){
           for(int i = 0;i<Numbers[0];i++){
            screen_pos[0]=get_x_value();
            screen_pos[1]=get_y_value();
            UART_PORT.println("p"+String(screen_pos[0])+","+String(screen_pos[1]));
            delay(Numbers[1]);
           }
           #ifdef SCREEN_WO_RESOLUTION
            UART_PORT.println("x_min = "+String(x_range[0])+" ,x_max = "+String(x_range[1]));
            UART_PORT.println("y_min = "+String(y_range[0])+" ,y_max = "+String(y_range[1]));
           #endif
          }
          else{
           UART_PORT.println("Buffer length doesn't match");
          }
         break;

        case 'a':
```

```
            if(bufferSize == 2){
              pinMode(A0,INPUT);
              pinMode(A1,INPUT);
                for(int i = 0;i<Numbers[0];i++){
                 UART_PORT.println("a"+String(analogRead(A0))+","+String(analogRead(A1)));
                 delay(Numbers[1]);
                }
            }
            else{
              UART_PORT.println("Buffer length doesn't match");
            }
            break;

           case 's':
                counter_out=0;
                reset_filter();

                break;
           default:
            UART_PORT.println("Action was not defined");
            uart_help();
            break;
         }
       }
       else{
        UART_PORT.println("Invalid Input");
        uart_help();
       }
  }
}

void uart_help(void){
  UART_PORT.println("fX,T - Get touch screen data X times each T (in milliseconds) with filter and counterout");
  UART_PORT.println("   -Example: f10,100 (Get 10 touch screen position each 100 milliseconds)");
  UART_PORT.println("wX,T - Get touch screen data X times each T (in milliseconds) without filter and counterout");
  UART_PORT.println("   -Example: s10,100 (Get 10 touch screen position each 100 milliseconds)");
  UART_PORT.println("aX,D - Get A0-A1 analogInput X times each T (in milliseconds)");
  UART_PORT.println("   -Example: a11,250 (Get 11 A0-A1 analog read each 250 milliseconds)");
  UART_PORT.println(" ");
```

```
}
```

ANEXO 2.- SCRIPT DE RASPBERRY PARA LA PANTALLA

```
import
numpy
as np
        from time import sleep
        from threading import Timer
        import serial,re,sys,cv2

        #if __name__ == '__main__':
        #    if len(sys.argv) == 2:
        #        port = sys.argv[1]
        #    else:
        #        if len(sys.argv) == 0:
        #            print("You should enter serial port")
        #        else:
        #            print("Only one parameter is allowed")
        #        exit(1)

        try:
            UPDATE_TIME = .025
            arduino = serial.Serial("/dev/ttyACM0",115200)
            state = "waiting"
        except:
            print("The entered serial port is not correctly or available")
            exit(1)

        def update_screen(pos_):
            wall_paper = cv2.imread('itcg_image.jpg')

            cv2.putText(wall_paper,"Posicion",(90,130),cv2.FONT_ITALIC,1,(255,0,0),1)
            cv2.putText(wall_paper,"x={:4}
        y={:4}".format(pos_[0],pos_[1]),(50,170),cv2.FONT_ITALIC,.65,(255,0,0),1)
```

```
    cv2.imshow('Ball and Plate ITCG',wall_paper)

def serial_irq():   #This function is request each .25 seconds
    if state != "break":
        if state == "filter_off":#Request screen information Without filter
            arduino.write(str.encode("w1,0\n"))
        elif state == "filter_on":#Request screen information Without filter
            arduino.write(str.encode("f1,0\n"))
        t = Timer(UPDATE_TIME, serial_irq)
        t.start()

t = Timer(UPDATE_TIME, serial_irq)
t.start()

update_screen(['----','----'])

while state != "break":
    try:
        if arduino.inWaiting() > 0:
            ##Read the serial port
            serial_input = arduino.readline()
            message = serial_input
            serial_input = serial_input[:-2].decode('utf-8') #Remove \n of expression

            matcher = re.compile(r'[A-Za-z][-]?[0-9]+([,][-]?[0-9]+)*$')
            #Verify the input
            if matcher.match(serial_input):
                char = serial_input[0]
                digits = serial_input[1:]

                if digits.find(','):
                    digits = digits.split(',')

                if len(digits) == 2 and char == 'p':   ##If enter 2 numbers
                    pos=[digits[0],digits[1]]
                    #Updating position values
                    if pos[0] == '1500':
                        wall_paper = cv2.imread('itcg_image.jpg')
                        cv2.putText(wall_paper,"Posicion",(90,130),cv2.FONT_ITALIC,1,(255,0,0),1)
                        cv2.putText(wall_paper,"NULL",(125,170),cv2.FONT_ITALIC,.6,(255,0,0),1)
```

```
                cv2.imshow('Ball and Plate ITCG',wall_paper)
            else:
                update_screen(pos)
        else:
            print("Expression doesn't match")
            print(message)
            update_screen(['xxxx','xxxx'])
    if state == "waiting":
        update_screen(['----','----'])
    key = cv2.waitKey(1) & 0xFF

    if key == ord('w'):
        state = "filter_off"
    elif key == ord('f'):
        state = "filter_on"
    elif key == ord('s'):
        update_screen(['----','----'])
        state = "waiting"
        arduino.write(str.encode("s0,0\n"))
    elif key == ord('b'):
        print("Closing... application")
        state = "break"
        arduino.close()
        cv2.destroyAllWindows()
        break;


except KeyboardInterrupt:
    print("Closing... application")
    state = "break"
    arduino.close()
    cv2.destroyAllWindows()
    break
```