



**UNIVERSIDAD POLITÉCNICA  
SALESIANA  
SEDE GUAYAQUIL**  
FACULTAD DE INGENIERÍAS  
CARRERA DE INGENIERÍA ELECTRÓNICA

PROYECTO DE GRADUACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO  
DE INGENIERO ELECTRÓNICO

TEMA:

**DISEÑO E IMPLEMENTACIÓN DE UN ROBOT HEXÁPODO  
CONTROLADO INALÁMBRICAMENTE MEDIANTE ARDUINO Y  
ANDROID EQUIPADO CON UN DIRECTOR ELECTRO ÓPTICO.**

Autor:  
**LUIS ENRIQUE ULLOA MEJÍA**

Director  
**ING. BYRON LIMA MSc.**

GUAYAQUIL – ECUADOR  
2017

## **CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA**

Yo, Luis Enrique Ulloa Mejia estudiante de Ingeniería Electrónica de la Universidad Politécnica Salesiana, certifico que los conceptos desarrollados, análisis realizados y las conclusiones del presente trabajo son de exclusiva responsabilidad del autor.

Guayaquil, Enero del 2017

---

Luis Enrique Ulloa  
C.I.: 0926176090

## **CERTIFICADO DE CESIÓN DE DERECHOS**

A través del presente certificado, se ceden los derechos de propiedad intelectual correspondiente a este trabajo, a la universidad Politécnica Salesiana, según lo establecido por la ley de propiedad intelectual y por su normativa institucional vigente.

Guayaquil, Enero del 2017

---

Luis Enrique Ulloa  
C.I.: 092617609

## **CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN**

Por medio de la presente doy constancia que el Sr. Luis Ulloa Mejía ha desarrollado y elaborado satisfactoriamente el proyecto final de titulación, que se ajusta a las normas establecidas por la Universidad Politécnica Salesiana, por tanto, autorizo su presentación para los fines legales pertinentes.

---

Ing. Byron Lima MSc.  
DIRECTOR DEL PROYECTO

## **DEDICATORIA**

A mis padres, que son la inspiración por la cual lucho día a día y por lo cual este logro alcanzado solo será un peldaño hacia muchas otras metas.

Luis Enrique Ulloa

## **AGRADECIMIENTO**

A Dios todo poderoso sean mis agradecimientos más sinceros por permitirme cumplir mis metas, por darme la sabiduría y una hermosa familia.

Agradezco también a mis padres ya que por el amor a sus hijos han sacrificado sus mejores días, dando todas sus fuerzas para poder sacarnos adelante, brindándome todo su apoyo, su comprensión y su amor incondicional.

El aprecio más grande y reconocimiento al apoyo de mi tutor el Ing. Byron Lima, el cual me apoyo con sus conocimientos para la culminación de este proyecto.

Luis Enrique Ulloa Mejía.

## RESUMEN

AÑO	ALUMNO	DIRECTOR DE PROYECTO TÉCNICO	TEMA DEL PROYECTO
2017	ULLOA MEJÍA LUIS ENRIQUE.	ING. BYRON LIMA MSc.	DISEÑO E IMPLEMENTACIÓN DE UN ROBOT HEXÁPODO CONTROLADO INALÁMBRICAMENTE MEDIANTE ARDUINO Y ANDROID EQUIPADO CON UN DIRECTOR ELECTRO ÓPTICO.

El presente proyecto técnico de titulación, “DISEÑO E IMPLEMENTACIÓN DE UN ROBOT HEXÁPODO CONTROLADO INALÁMBRICAMENTE MEDIANTE ARDUINO Y ANDROID EQUIPADO CON UN DIRECTOR ELECTRO ÓPTICO”, su principal objetivo fue implementar un Robot Hexápodo el cual fuera un medio didáctico para la interacción y desarrollo de prácticas de laboratorio. Se realizaron pruebas de programación de locomoción del robot junto con la comunicación con el dispositivo móvil y Arduino. Se estableció un control on/off con Histéresis para la manipulación de los servos motores teniendo como referencia las coordenadas de posición obtenidas en el procesamiento de la imagen obtenida con una cámara y Labview. Se investigó sobre los avances tecnológicos de los sistemas de visión artificial, locomoción de Robots, algoritmos de Control. Se programaron los algoritmos de control para el DEO (Director Electro Óptico) en Labview. El sistema DEO se encarga del procesamiento de la imagen a través de una cámara USB, la misma que se conecta a Labview en donde mediante herramientas de visión artificial, se realiza el control para el seguimiento de patrones, saber su posición, tamaño y aproximar la distancia con un medio físico hacia el objeto. El robot Hexápodo es controlado inalámbricamente por bluetooth y realiza movimientos de forma manual y automática con una aplicación en Android personalizada.

Al ejecutar el proyecto de titulación se concluye que el procesamiento de imagen mediante herramientas de visión artificial en Labview tiene un de retardo con relación a la imagen en tiempo real, la cual se está obteniendo mediante la cámara USB. Esta implementación ha otorgado un prototipo a los estudiantes de la Carrera de Ingeniería electrónica de la Universidad Politécnica Salesiana pudiendo así utilizarlo para futuros desarrollos en el campo de la visión artificial y la robótica.

**PALABRAS CLAVES:** Robot / Hexápodo / Arduino / Labview / Visión / Artificial / Android / Sistema Deo / Servo motor / Control.

## ABSTRACT

YEAR	STUDENT	TECHNICAL PROJECT MANAGER	PROJECT THEME
2017	ULLOA MEJÍA LUIS ENRIQUE.	ING. BYRON LIMA MSc.	DESIGN AND IMPLEMENTATION OF A HEXÁPOD ROBOT CONTROLLED WIRELESS THROUGH ARDUINO AND ANDROID EQUIPPED WITH AN ELECTRO OPTICAL DIRECTOR

The present technical project of titration, "DESIGN AND IMPLEMENTATION OF A HEXÁPOD ROBOT CONTROLLED WIRELESS THROUGH ARDUINO AND ANDROID EQUIPPED WITH AN ELECTRO OPTICAL DIRECTOR", its main objective was to implement a Robot Hexápodo which was a didactic means for the interaction and development of practices from laboratory. Robot locomotion programming tests were performed along with communication with the mobile device and Arduino. An on / off control was established with Hysteresis for the manipulation of the servo motors, having as reference the position coordinates obtained in the image processing obtained with a camera and Labview. We investigated the technological advances of artificial vision systems, locomotion of Robots, Control algorithms. The control algorithms for DEO (Electro Optical Director) were programmed in Labview. The DEO system is responsible for the image processing through a USB camera, the same one that connects to Labview where, using artificial vision tools, the control is monitored for pattern tracking, position, size and approximation Distance with a physical medium towards the object. The Hexapod robot is wirelessly controlled by bluetooth and performs manual and automatic movements with a custom Android application.

When executing the titling project, it is concluded that the image processing using artificial vision tools in Labview has a delay in relation to the real time image, which is obtained through the USB camera. This implementation has given a prototype to the students of the Electronic Engineering Career of the Salesian Polytechnic University, being able to use it for future developments in the field of artificial vision and robotics.

**KEYWORDS:** Robot / Hexápodo / Arduino / Labview / Vision / Artificial / Android / Deo System / Servo motor / Control.

## ÍNDICE GENERAL

1.	EL PROBLEMA.....	2
1.1	Descripción del Problema.....	2
1.2	Importancia y alcance.....	2
1.3	Delimitación del problema .....	2
1.3.1.	Delimitación Temporal: .....	2
1.3.2.	Delimitación Espacial:.....	2
1.3.3.	Delimitación Académica:.....	3
1.4	Objetivos.....	3
1.4.1	Objetivo General .....	3
1.4.2	Objetivos Específicos.....	3
2.	ESTADO DEL ARTE .....	4
2.1	Servo Motor .....	4
2.2	Cámara Web USB HD .....	4
2.3	Arduino Mega .....	5
2.4	Labview .....	5
2.5	Ni Vision labview .....	5
2.6	Visión Artificial .....	8
3.	MARCO METODOLÓGICO .....	10
3.1	Diseño Mecánico .....	10
3.2	Diseño Electrónico.....	13
3.3	Diseño de la aplicación en Android .....	19
3.4	Simulación del Hexápodo en Matlab .....	21
3.5	Diseño de Interfaz en Labview .....	25
4.	RESULTADOS.....	33
4.1	Planos .....	34
4.2	Diseño de Baquelitas de alimentación y control: .....	35
4.3	Diseño de Aplicación en Android.....	35
4.4	Programación y Simulación del Robot Hexápodo en Matlab.....	38
4.5	Programación del Sistema DEO en Labview.....	39
	CONCLUSIONES Y RECOMENDACIONES.....	50
	REFERENCIAS BIBLIOGRAFICAS.....	51
	ANEXOS.....	52

## ÍNDICE DE FIGURAS

<b>Figura 1.</b> Delimitación espacial del proyecto técnico.....	2
<b>Figura 2.</b> Servo Motor De engranajes metálicos 180. ....	4
<b>Figura 3.</b> Cámara HD 1600 x 1200, 720 p 2.0 MPX. ....	4
<b>Figura 4.</b> Arduino Mega .....	5
<b>Figura 5.</b> Menú NI-IMAQ. ....	6
<b>Figura 6.</b> Menú NI-IMAQ. ....	6
<b>Figura 7.</b> Menú Vision Utilities. ....	6
<b>Figura 8.</b> Menú Image Processing. ....	7
<b>Figura 9.</b> Menú Color Processing. ....	7
<b>Figura 10.</b> Menú Machine Vision. ....	7
<b>Figura 11.</b> NI – IMAQ dx.....	8
<b>Figura 12.</b> NI – Vision Express.....	8
<b>Figura 13.</b> Diseño en 3D del Robot Hexápodo - Extremidad .....	10
<b>Figura 14.</b> Diseño en 3D del Robot Hexápodo .....	10
<b>Figura 15.</b> Base soporte del DEO .....	11
<b>Figura 16.</b> Servo Motor MG6r .....	11
<b>Figura 17.</b> Micro Servo SD90 .....	12
<b>Figura 18.</b> Diseño de Circuito Impreso en Proteus 8.....	13
<b>Figura 19.</b> Placa Impresa .....	13
<b>Figura 20.</b> Batería Lipo 8000mAh 2S1P 30C.....	14
<b>Figura 21.</b> Tripies para la caminata hacia adelante.....	15
<b>Figura 22.</b> Tripies para giro hacia la derecha .....	15
<b>Figura 23.</b> Tripies para giro hacia la Izquierda .....	16
<b>Figura 24.</b> Tripies para la caminata hacia atrás .....	16
<b>Figura 25.</b> Sensor Ultrasónico HC-SR04.....	17
<b>Figura 26.</b> Vinculación de Caracteres RX-TX.....	18
<b>Figura 27.</b> Módulo HC06 .....	18
<b>Figura 28.</b> Pantalla Principal de aplicación en Android.....	19
<b>Figura 29.</b> Solicitud de permisos de Bluetooth en Aplicación .....	19
<b>Figura 30.</b> Conexión con dispositivos vinculados en Aplicación.....	20
<b>Figura 31.</b> Opción Manual en Aplicación.....	20
<b>Figura 32.</b> Opción Automática en Aplicación .....	21
<b>Figura 33.</b> Asignación de ejes en una Extremidad .....	21
<b>Figura 34.</b> Tabla de Parámetros DH .....	22
<b>Figura 35.</b> Diagrama de Flujo Simulación en Matlab.....	22
<b>Figura 36.</b> Programación de extremidad de robot en Matlab.....	23
<b>Figura 37.</b> Trayectoria de extremidad en Matlab.....	23
<b>Figura 38.</b> Tiempos en cada segmento del robot en Matlab.....	23
<b>Figura 39.</b> Extremidades totales del Robot en Matlab. ....	24
<b>Figura 40.</b> Graficas de Extremidades y Cuerpo del Robot en Matlab. ....	24
<b>Figura 41.</b> Simulación del Robot Hexápodo en Matlab.....	24
<b>Figura 42.</b> Configuración de bloques DQ en el VI.....	25
<b>Figura 43.</b> Estados del proyecto en Case Structure .....	25
<b>Figura 44.</b> Estado “Inicio” en Case Structure.....	26
<b>Figura 45.</b> Estado “Stand By” en Case Structure.....	26
<b>Figura 46.</b> Estado “Búsqueda” en Case Structure.....	27
<b>Figura 47.</b> Estado “Salir” en Case Structure.....	27

<b>Figura 48.</b> Evento 0 “Time Out” en Case Event Structure .....	27
<b>Figura 49.</b> Evento 1 “Rojo, Verde, Azul” en Case Event Structure .....	28
<b>Figura 50.</b> Evento 2 “Reset Rojo, Verde, Azul” en Case Event Structure .	28
<b>Figura 51.</b> Evento 3 “Aprender” en Case Event Structure .....	29
<b>Figura 52.</b> Evento 4 “Calibración” en Case Event Structure.....	29
<b>Figura 53.</b> Evento 5 “Salir” en Case Event Structure.....	30
<b>Figura 54.</b> Index Array de Variable Local “Posición” .....	30
<b>Figura 55.</b> Control ON/OFF con Histéresis del sistema DEO .....	31
<b>Figura 56.</b> Límites de protección para el torque en los servomotores.....	31
<b>Figura 57.</b> Control ON/OFF – LINX Labview.....	32
<b>Figura 58.</b> Corte y Ensamblaje de Piezas del Robot (1).....	34
<b>Figura 59.</b> Corte y Ensamblaje de Piezas del Robot (2).....	34
<b>Figura 60.</b> Tarjetas Impresas colocadas en el Robot Hexápodo .....	35
<b>Figura 61.</b> Boton Adelante Click Down - Click UP .....	36
<b>Figura 62.</b> Botón Atrás Click Down - Click UP .....	36
<b>Figura 63.</b> Botón Izquierda Click Down –Click UP .....	37
<b>Figura 64.</b> Botón Derecha Click Down –Click UP .....	37
<b>Figura 65.</b> Botón Arriba-Abajo Click Down–Click UP .....	38
<b>Figura 66.</b> Simulación de Robot Hexápodo en Matlab (1).....	38
<b>Figura 67.</b> Simulación de Robot Hexápodo en Matlab (2).....	39
<b>Figura 68.</b> Posición de Servos vista central de sistema DEO .....	39
<b>Figura 69.</b> Posición de Servos vista lateral de sistema DEO .....	40
<b>Figura 70.</b> Posición de Servos vista lateral de sistema DEO .....	41
<b>Figura 71.</b> Posición de Servos vista Inferior de sistema DEO .....	42
<b>Figura 72.</b> Posición de Servos vista Superior de sistema DEO.....	43
<b>Figura 73.</b> Posición de Servos Máximo X Positivo de sistema DEO .....	44
<b>Figura 74.</b> Posición de Servos Máximo X Negativo de sistema DEO.....	45
<b>Figura 75.</b> Posición de Servos Máximo Y de sistema DEO .....	46
<b>Figura 76.</b> Posición de Servos Máximo Y Negativo de sistema DEO.....	47

## INTRODUCCIÓN

En este proyecto se da a conocer un prototipo robótico el cual contribuirá académicamente a los estudiantes de la Universidad Politécnica Salesiana logrando así que puedan experimentar con un medio físico práctico el marco teórico obtenido en clases.

Una de las principales soluciones para que un sistema DEO (Director Electro Óptico) sea eficiente es necesario tener un método de control con el cual pueda manipular los valores de entrada (coordenadas de posición del objeto) manteniendo así de forma correcta los valores de salida (servo motor). En el diseño del robot se tiene en consideración que la estructura sea de un material liviano el cual no se haga problemas para los servos motores poder levantar toda la estructura y que la locomoción del robot sea eficiente.

Este proyecto puede ser replicado para futuros desarrollos en la Carrera Ingeniería electrónica de la Universidad Politécnica Salesiana e incluso en el ámbito investigativo de la Institución.

Con el proyecto a continuación descrito se permite dar paso al uso de herramientas de visión artificial dentro de la Universidad Politécnica Salesiana sede Guayaquil, para luego ascender e implementarlas en diversos campos y aplicaciones.

# 1. EL PROBLEMA

## 1.1 Descripción del Problema

Actualmente en la Universidad Politécnica Salesiana sede Guayaquil, se imparte como materia Electiva I la cual tiene como contenido la Robótica que incluye como tal la Robótica Industrial y la Robótica Móvil, en dicha materia se tiene la necesidad de que los estudiantes puedan Interactuar con un prototipo de un Robot móvil, donde puedan experimentar y relacionar los datos teóricos y matemáticos con los prácticos fortaleciendo de esta manera el aprendizaje y el contenido en esta área.

## 1.2 Importancia y alcance

El proyecto técnico a implementar es de gran importancia para la carrera Electrónica, dado que será de útil para la experimentación de movimiento cinemático mediante la locomoción de un, de igual manera se verifica la operatividad del sistema DEO (Director Electro Óptico) comprobando los resultados en las variables propuestas.

Además de lo descrito cabe recalcar que el proyecto se podrá implementar y usar para futuros precedentes.

## 1.3 Delimitación del problema

### 1.3.1. Delimitación Temporal:

El proyecto de graduación se lo culmino en el año 2017. Tuvo una duración en su etapa de ejecución de 1 año, con un mes adicional para ultimar detalles y realizar preparativos para su posterior sustentación.

### 1.3.2. Delimitación Espacial:

El proyecto se lo va a utilizar en el Laboratorio de Control del bloque B de la Universidad Politécnica Salesiana de la ciudad de Guayaquil.



**Figura 1.** Delimitación espacial del proyecto técnico.  
(Maps, 2017)

### 1.3.3. Delimitación Académica:

El alcance del proyecto, en el ámbito académico abarca los conocimientos adquiridos en la lectura de datos desde un computador, usando interfaces estudiadas a lo largo de la carrera ingeniería Electrónica que se ven en materias como Control I, II y III, Electiva I, e Instrumentación.

Además, comprende la elaboración de un algoritmo de control para el seguimiento de objetos mediante visión artificial.

**Innovación:** El proyecto presenta desarrollos en el área científica de la robótica y servirá para futuras aplicaciones relacionadas en el ámbito de la electrónica y Visión Artificial.

**Impacto:** El proyecto se podrá utilizar por los estudiantes y docentes de la carrera Ingeniería Electrónica y servirá de base y soporte para desarrollos que se quiera estudiar e implementar.

## 1.4 Objetivos

### 1.4.1 Objetivo General

Diseñar e Implementar un Robot Hexápodo controlado Inalámbricamente mediante un dispositivo Móvil equipado con un Director Electro Óptico con adquisición y procesamiento de imágenes en tiempo real para la realización de prácticas en el área de Robótica móvil.

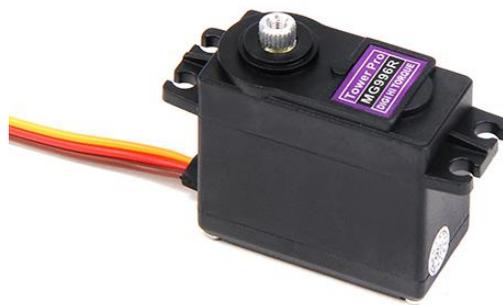
### 1.4.2 Objetivos Específicos

- Programar una tarjeta Arduino como controlador para la Locomoción del Robot Hexápodo Caminante.
- Implementar una comunicación inalámbrica entre el robot y un servidor externo (SO Android mediante un dispositivo móvil).
- Construir un sistema mecánico con un material resistente y liviano para el alargue del tiempo de vida del robot.
- Implementar un Director Electro Óptico mediante herramientas de visión artificial.
- Aplicar algoritmos de visión artificial para desarrollar tratamiento de imágenes mediante Labview.
- Diseñar un algoritmo de control, para manipular los servomotores que dan el movimiento de la cámara sobre el eje vertical u horizontal(X o Y).
- Diseñar una aplicación personalizada para el control del Robot Hexápodo Caminante.
- Elaborar 4 prácticas para desarrollarlas en la materia de Electiva I.
- Simular la locomoción del Robot Hexápodo utilizando el estudio de la cinemática de los Robots mediante Matlab.

## 2. ESTADO DEL ARTE

### 2.1 Servo Motor

Un Servo es un Motor DC pequeño que tiene una tarjeta electrónica para el control de movimiento del eje. Este puede ser controlado en grados al enviar un valor angular a la tarjeta electrónica. La posición del eje del servo es fija hasta que el valor angular cambie por algún otro valor codificado y este se encuentre dentro del tope del servo. En la práctica, son de mucha utilidad en el control de movimiento de superficies y la robótica. Existen Servo Motores que se ajustan a la necesidad y uso que se requiera tanto en fuerza, torque, tamaño y precio. (González, 2004)



**Figura 2.** Servo Motor De engranajes metálicos 180.  
(Automation Direct, 2015)

### 2.2 Cámara Web USB HD

Esta Cámara web Genius permite mantener una buena calidad de video, gracias a su cámara HD 1600 x 1200 ,720p con 2.0 megapíxeles. Imágenes nítidas y de color perfecto. Es utilizada en nuestro proyecto para la adquisición y procesamiento de imagen en Labview de nuestro sistema DEO.



**Figura 3.**Cámara HD 1600 x 1200, 720 p 2.0 MPX.

(Compras, 2016)

### 2.3 Arduino Mega

El Arduino Mega 2560 es un controlador que contiene una Atmega2560 contiene pines digitales de entrada y salida (de los cuales 15 salidas digitales son de PWM), entradas analógicas, (puertas seriales), un oscilador de 16MHz, conexión USB, conector de alimentación, una cabecera, y un botón de reinicio.

El controlador tiene una conexión USB para facilidad de programación y comunicación del mismo, es compatible con diversas plataformas de desarrollo.

La Arduino mega se energiza mediante una fuente de alimentación dc o mediante conexión USB, el voltaje recomendado para alimentarlo es de 7 hasta 12v. (ArduinoWebSite, 2017)



**Figura 4.**Arduino Mega  
(ArduinoWebSite, 2017)

### 2.4 Labview

National Instrument ha desarrollado una plataforma para el desarrollo de una variedad de aplicaciones enfocadas a la ingeniería y la ciencia. Su lenguaje de programación es basada en gráficos para la facilidad de creación y visualización de diferentes tipos de sistemas y algoritmos. Labview se lo utiliza para el análisis, adquisición de datos, supervisión, control de modelos, sistemas inteligentes y monitoreo de dispositivos conectados. Labview ha sido la solución predilecta para crear e implementar proyectos de desarrollo e innovación por décadas. (LABVIEW, 2016)

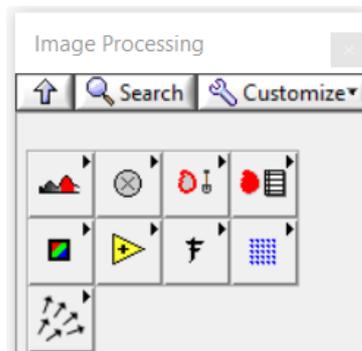
### 2.5 Ni Vision labview

NI Vision para LabVIEW, una parte del Desarrollo NI Vision Módulo - es una librería de VIs de LabVIEW que se puede utilizar para desarrollar visión artificial y aplicaciones de imágenes científicas. El NI Vision Módulo de Desarrollo también incluye las mismas funciones de imagen para otros entornos de desarrollo C LabWindows / CVI y, así como Controles ActiveX para Visual Basic. El asistente de visión, es otro Módulo de desarrollo de productos de software, que permite a un prototipo de su estrategia de aplicación de forma rápida y sin tener que hacer ningún tipo de programación.



## Image Processing

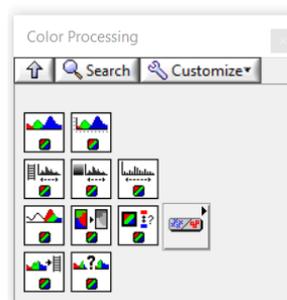
El menú Image Processing se lo utiliza para procesar la imagen, aplicar filtros analizar y adquirir valores mediante morfología.



**Figura 8.**Menú Image Processing.  
(National Instruments, 2005)

## Color Processing

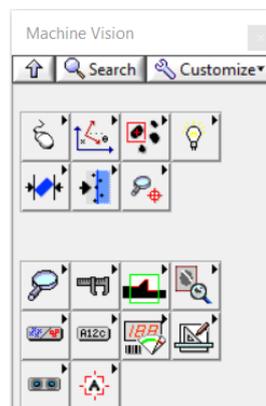
El menú Color Processing se lo utiliza para realizar el tratamiento en la imagen como el color, contraste, brillo, etc.



**Figura 9.**Menú Color Processing.  
(National Instruments, 2005)

## Machine Vision

El menú Machine visión tiene varias utilidades entre aquellas está la medición de distancia en puntos, buscar patrones, inspección, coordenadas, etc



**Figura 10.**Menú Machine Vision.  
(National Instruments, 2005)

## NI-IMAQdx

El menú NI-IMAQ dx tiene varias funciones para controlar las características de una cámara digitales externas.



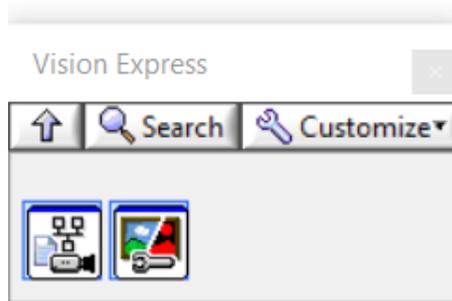
**Figura 11.**NI – IMAQ dx  
(National Instruments, 2005)

## Visión Express

El menú Vision Express permite interactuar rápidamente la adquisición de imagen y procesamiento de las mismas a través del uso de herramientas. (Véase Figura 12)

**Vision Acquisition:** Es una herramienta que ayuda a la adquisición de imágenes mediante cámaras análogas, digitales etc.

**Vision Assistant:** Es una herramienta que ayuda al procesamiento de imágenes y realizar varias aplicaciones. (National Instruments, 2005)



**Figura 12.**NI – Vision Express.  
(National Instruments, 2005)

## 2.6 Visión Artificial

Podríamos indicar que la Visión Artificial describe la estructura y las características de un mundo tridimensional, las imágenes bidimensionales del mundo pueden ser monocromáticas (de niveles de gris) o en colores, estas pueden resultar de una o varias cámaras e incluso pueden estas estacionarias o móviles.

Las características y propiedades del mundo tridimensional que se quiere deducir en visión artificial incluyen además de sus variables físicas, sus propiedades materiales. Como ejemplo de variables físicas son el tamaño, la posición, la forma. Ejemplo de propiedades materiales son la textura, iluminación, su color, y composición.

Si el ambiente cambia en el proceso de adquisición de la imagen, debemos inferir también el cambio e incluso adelantarnos en el futuro. Como entrada

de un sistema de visión artificial es una imagen en tiempo real obtenida por un medio de adquisición, y su salida es una representación de la escena, la cual es obtenida de la imagen principal. Esta descripción debe estar relacionada en la imagen en tiempo real y, por el otro, debe contener la característica requerida para la interacción que se desea realizar, por ejemplo la interacción con un robot que depende de la imagen principal en la entrada y la salida debe proporcionar una relación utilizable para el robot. (SOBRADO, 2003)

### 3. MARCO METODOLÓGICO

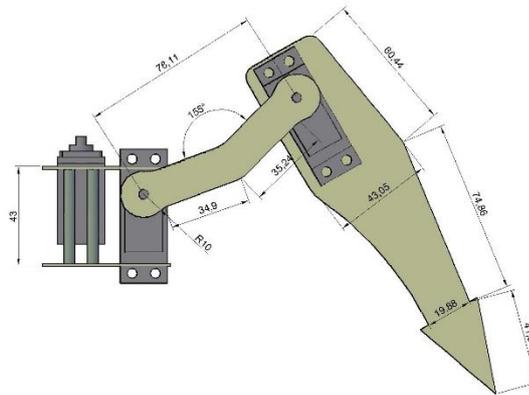
Para realizar un diseño de control para el sistema DEO (Director Electro Óptico) y para el control de locomoción del Robot, se necesita segmentarlos por diferentes etapas de trabajo.

#### 3.1 Diseño Mecánico

Lo primero que se realizó fue el diseño de la estructura del robot hexápodo considerando un diseño de características positivas tales como: buena flexibilidad, rápida adaptación.

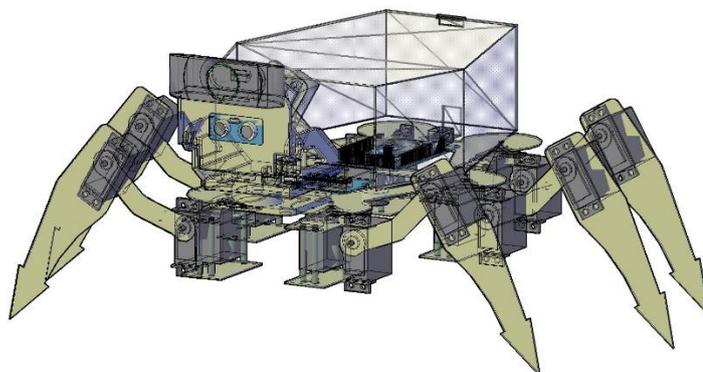
Para la construcción del Robot Hexápodo se realizó el diseño en CAD de los planos que consta el diseño de las piezas y las estructuras.

Las extremidades del Robot Hexápodo constan de 3 grados de libertad cada una obteniendo así un mejor soporte y flexibilidad en los movimientos del robot. Son 6 extremidades con las mismas características, teniendo un total de 18 grados de libertad en todas las extremidades.



**Figura 13.**Diseño en 3D del Robot Hexápodo - Extremidad

La estructura del robot Hexápodo se diseñó acorde a las extremidades obteniendo un modelo moderno y de visión agradable para el usuario.



**Figura 14.**Diseño en 3D del Robot Hexápodo

## Soporte de cámara con servos

El soporte de cámara con servos fue diseñado para ser colocado en la parte superior del Robot Hexápodo y que tenga dos grados de libertad tanto en el eje X como en el eje Y. El diseño se lo realizó en Autocad y se lo guardo en un formato. STL para poder imprimir las piezas del modelo en 3D.

La estructura base soporte para la cámara del sistema DEO se diseñó en CAD y se consideró que para esta base se utilizarían Micro Servos que son de menor tamaño por ende de menor torque, resistencia y menor fuerza. La base consta de dos grados de libertad una da movilidad en el eje horizontal (x) y la otra en el eje vertical (y).



**Figura 15.** Base soporte del DEO

## Servomotores

Luego de realizar el diseño del robot hexápodo se realizó un levantamiento de los actuadores y sensores que conforman este proyecto. Se pudo identificar 20 Servo Motores y 1 Sensor Ultrasónico.

Para el ensamblaje del hexápodo se utilizó 18 servomotores, se eligió servomotores de engranaje metálico por su alto torque, resistencia y fuerza.



**Figura 16.**Servo Motor MG6r

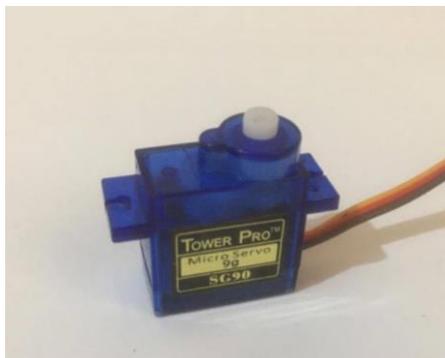
Este servo estándar de alto torque puede girar aproximadamente 180. Puede utilizar cualquier servo código, hardware o biblioteca para controlar estos servos. Las especificaciones se detallan a continuación:

### **Características:**

- Peso:55g
- Dimensión: 40.7 \* 19.7 \* 42.9mm aproximadamente.
- Torque máximo: 10 kg / cm
- Velocidad de funcionamiento: 0.14sec / 60 grados (6V)
- Voltaje de funcionamiento: 4.8-7.2V
- Corriente de funcionamiento: 500 mA – 900 mA(6V)
- Rango de temperatura: 0° a 55°

### **Micro Servo**

El servo SG90 Tower Pro un servo miniatura de gran calidad y diminutas dimensiones, Funciona con la mayoría de tarjetas electrónicas de control con micro controladores. Este tipo de servo es ideal para las primeras experiencias de aprendizaje y prácticas con servos, ya que sus requerimientos de energía son bastante bajos y se permite alimentarlo con la misma fuente de alimentación que el circuito de control. Por ejemplo, si se conecta a una tarjeta arduino, se puede alimentar durante las pruebas desde el puerto USB del PC sin mayor problema.



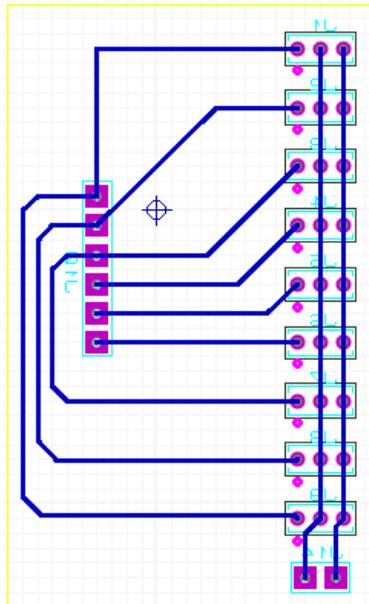
**Figura 17.**Micro Servo SD90

### **Características:**

- Micro Servo Tower-pro
- Velocidad: 0.10 sec/60° @ 4.8V
- Torque: 1.8 Kg-cm @ 4.8V
- Voltaje de funcionamiento: 3.0-7.2V
- Temperatura de funcionamiento: -30 °C ~ 60 °C
- Ángulo de rotación: 180°
- Ancho de pulso: 500-2400 µs
- Longitud de cable de conector: 24.5cm

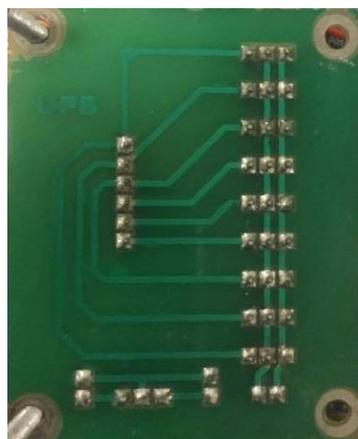
### 3.2 Diseño Electrónico

Lo que primero que se realizó fue el diseño de las tarjetas Impresas que sirven para alimentar todos los servomotores de nuestro prototipo. El software que se utilizó fue el Proteus 8, donde se diseñó las pistas en las dos tarjetas que se crearon. Una tarjeta se utilizó para alimentar el lado izquierdo del robot (3 extremidades) y la otra tarjeta se utilizó para alimentar el lado derecho del robot (3 extremidades).



**Figura 18.**Diseño de Circuito Impreso en Proteus 8

Luego de terminar con el diseño del circuito mediante un proveedor se imprimió en una placa de fibra de vidrio el cual es un material de mayor resistencia y duración.



**Figura 19.**Placa Impresa

## Batería de Alimentación

Para determinar la Batería Idónea se procedió a verificar el consumo de los 18 servomotores de nuestro robot Hexápodo, obteniendo un consumo de 5,5 A aproximadamente, por ende se utilizó una batería tipo Lipo de 8 A – 7.4 V para optimizar el trabajo y la eficiencia de los servomotores. La Batería que se utiliza tiene un tiempo de duración en funcionamiento de 45 minutos aproximadamente, luego se carga la batería en modo balanceado para que sus dos celdas se carguen equilibradamente.

### Especificaciones:

**Capacidad:** 8000mAh

**Voltaje:** 3S1P / 2 / celulares 11.1v

**Descarga:** 30 ° C Constante / 40C Burst

**Peso:** 644g (incluyendo cable, el enchufe y la caja)

**Dimensiones:** 169x69x27mm

**Balance de enchufe:** JST-XH

**Tapón de descarga:** 5,5 mm conector de balas (sin alojamiento).



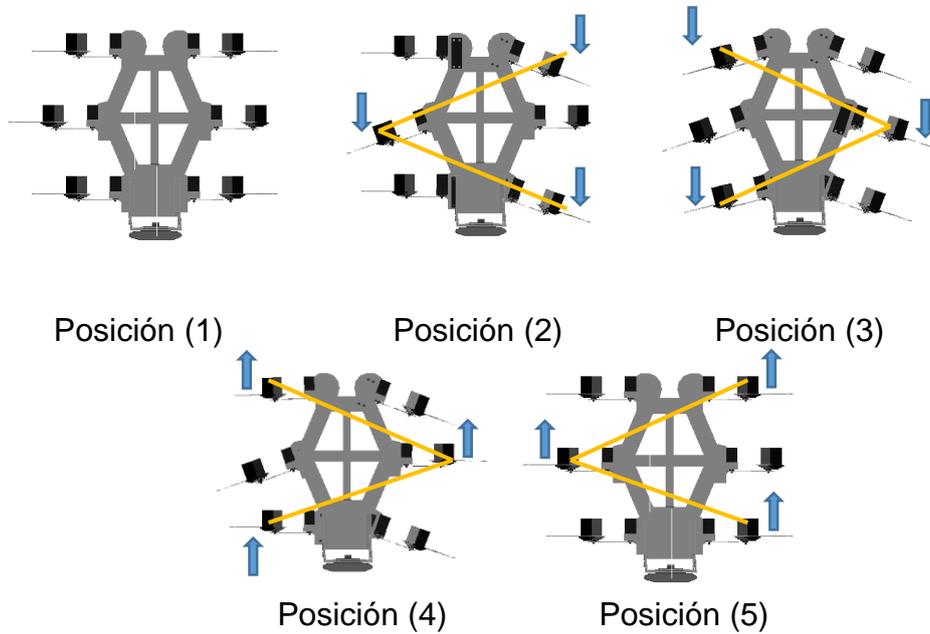
**Figura 20.**Batería Lipo 8000mAh 2S1P 30C

## Controlador para el Robot

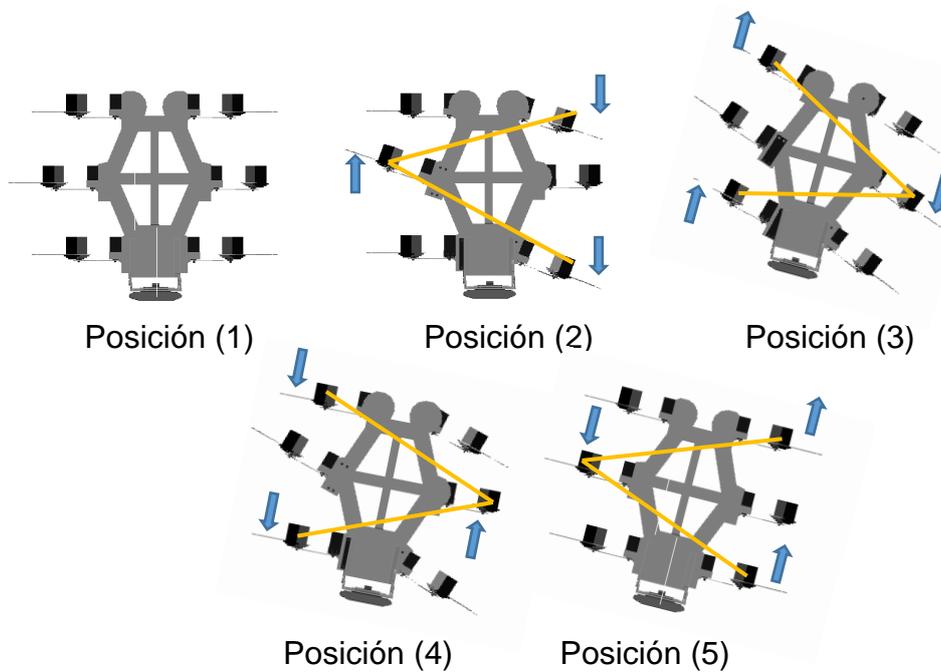
El controlador que se utilizó para la programación del robot Hexápodo es el Arduino Mega. El Arduino Mega 2560 a diferencia del Arduino Uno, tiene 12 salidas digitales PWM las cuales son utilizadas para controlar los servomotores del robot Hexápodo. Como solo se tiene 12 señales PWM y los servomotores son 18, se logra controlar 4 extremidades del robot, las dos extremidades restantes se conectan con otras dos diferentes para completar el control de todas las extremidades. En esta tarjeta se conecta el Modulo de comunicación Bluetooth utilizando los puertos seriales TX y RX para lograr así la transmisión y recepción de datos desde la aplicación en Android.

El robot puede avanzar de forma frontal y lateral, así como realizar movimientos de rotación sobre su centro geométrico. Para realizar los movimientos de locomoción, se aplica el concepto conocido como caminata con tripie o triangulo de equilibrio, donde el robot hexápodo conserva su

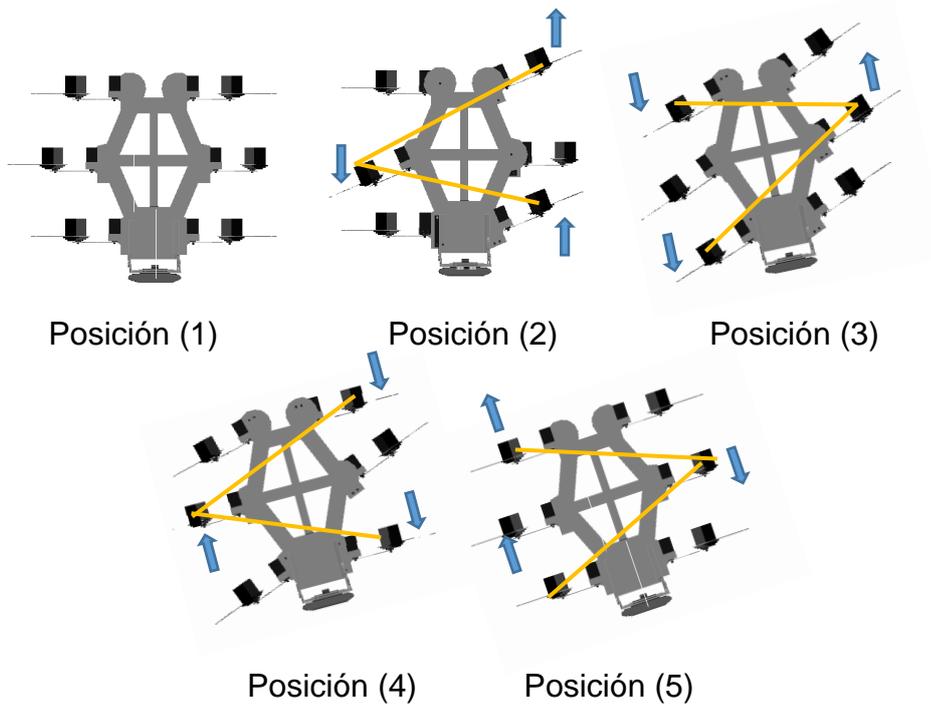
equilibrio, ya sea que este sostenido estáticamente o realizando un movimiento de avance o de rotación. El concepto consiste conservar el centro de gravedad del robot dentro del área formada por el tripie o triangulo de apoyo. En las Figura 21 hasta la 24 se muestra el tripie de movimiento directo y lateral del robot.



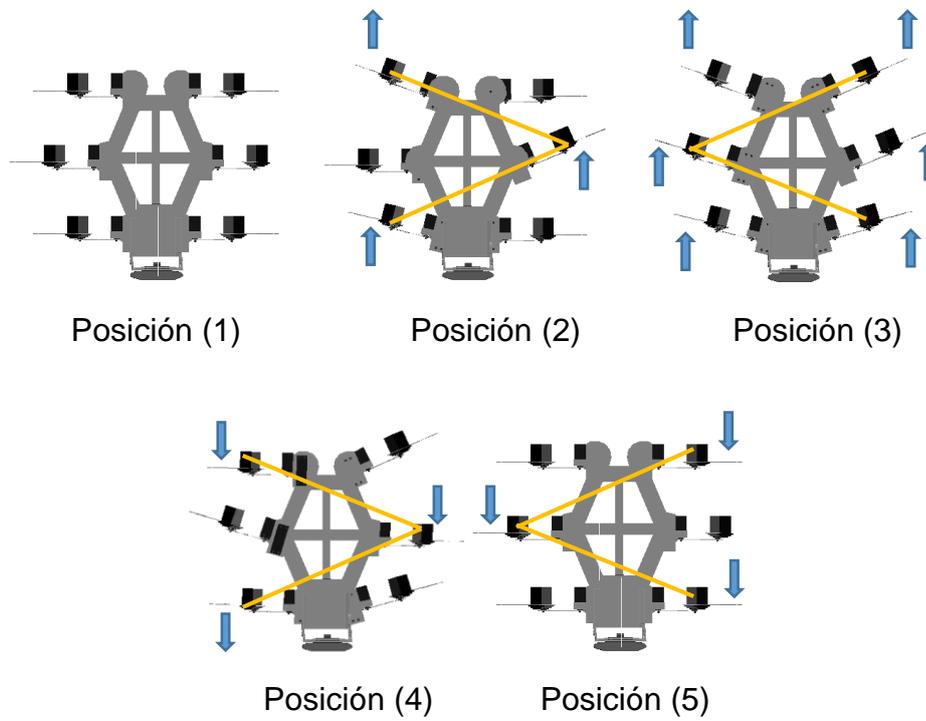
**Figura 21.** Tripies para la caminata hacia adelante



**Figura 22.** Tripies para giro hacia la derecha



**Figura 23.** Tripies para giro hacia la Izquierda



**Figura 24.** Tripies para la caminata hacia atrás

## Controlador para el Sistema DEO

El controlador que se utilizó para la programación en el sistema DEO es el Arduino UNO. El Arduino UNO tiene 6 salidas PWM de las cuales dos son utilizadas para los dos Micro Servos que se encuentran colocados en la base soporte. Se utiliza dos puertos de señales Digitales en donde se conecta el Sensor Ultra Sónico HC-SR04. El Controlador adquiere los datos de proximidad y distancia del sensor y los muestra en el Labview.

## Sensor Ultra Sónico.

El Sensor Ultrasónico se utilizó es el HC-SR04, es un sensor que mide distancia por medio de ultrasonidos es capaz de detectar objetos y calcular a que distancia se encuentra dentro de un rango de 2 a 450 cm, el dato que captura el sensor es recibido por la tarjeta Arduino UNO mediante las entradas digitales y lo muestra mediante la interfaz de Labview. Se utilizó la librería MakerHub Linx para realizar la lectura de datos del sensor entre el Arduino y el Labview, la lectura de datos en el sensor es constante y varia su valor dependiendo de la distancia del objeto a medir.

## Características

- Dimensiones: 43 x 20 x 17 mm
- Voltaje de alimentación: 5 Vcc
- Frecuencia de Operación: 40 KHz
- Rango mínimo: 1.7 cm
- Rango máximo: 4.5 m
- Duración del pulso eco de salida: 100-25000  $\mu$ S.
- Duración mínima del pulso de disparo 10  $\mu$ S.
- Tiempo mínimo de espera entre una medida y el inicio de otra: 20 mS.



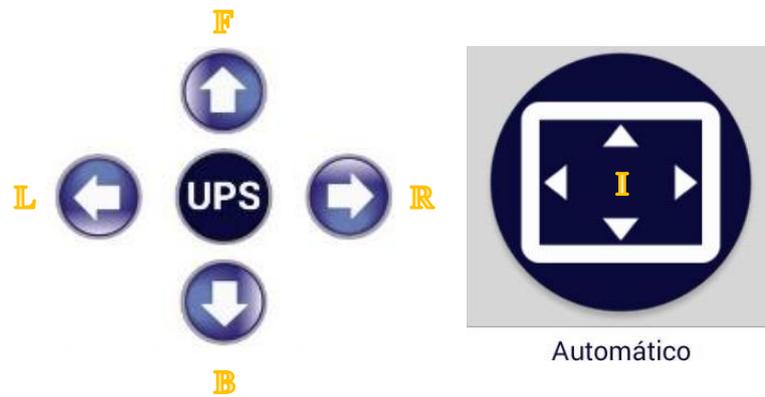
**Figura 25.**Sensor Ultrasónico HC-SR04  
(ElectronicLab, 2017)

## Módulo de comunicación bluetooth

El Modulo de comunicación Bluetooth que se utilizó es el HC06, este módulo de bluetooth es el que ofrece la comunicación con la tarjeta Arduino Mega, mediante este módulo es posible la comunicación entre el Arduino y el dispositivo móvil obteniendo un control inalámbrico del robot. El Modulo HC06 es conectado al Arduino mediante los puertos de comunicación RX-TX, estos puertos se encargan de la transmisión y recepción de datos generados por la

aplicación en Android. Los datos enviados fueron caracteres, las letras que se asignaron para cada secuencia de pasos en la programación son:

- Adelante : F
- Atrás: B
- Derecha: R
- Izquierda: L
- Arriba-Abajo: W
- Automático: I



**Figura 26.**Vinculación de Caracteres RX-TX

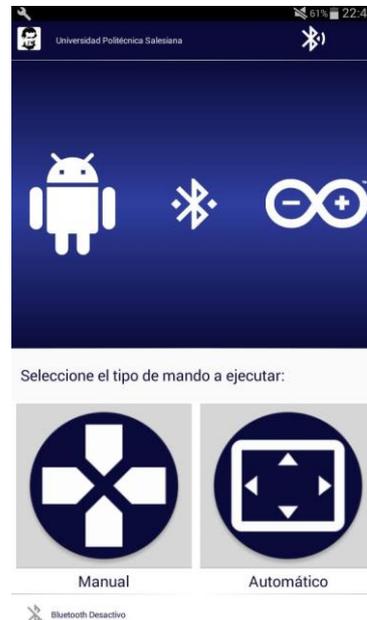
La transmisión de caracteres es constante y dependerá de la activación de la tecla direccional deseada en la Aplicación. Cada carácter accionado activa una secuencia de pasos programado en el robot Hexápodo.



**Figura 27.**Módulo HC06  
(Ruben, 2014)

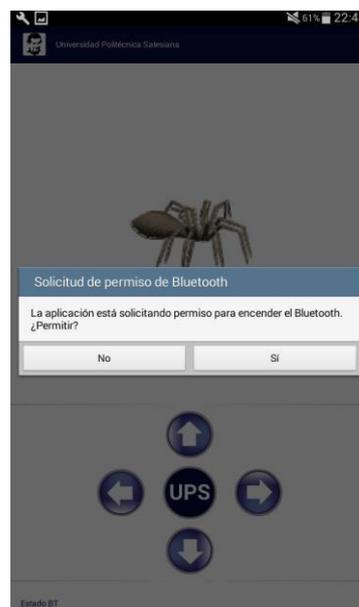
### 3.3 Diseño de la aplicación en Android

Lo primero que se realizó fue buscar la interfaz de usuario que se necesitaba para la aplicación, una interface que se ajuste a las necesidades de locomoción del Robot Hexápodo.



**Figura 28.** Interfaz Principal de aplicación en Android.

Al iniciar la aplicación se nos mostrará la interfaz principal en nuestro dispositivo móvil, si el bluetooth se encuentra desactivado preguntara si desea activarlo.



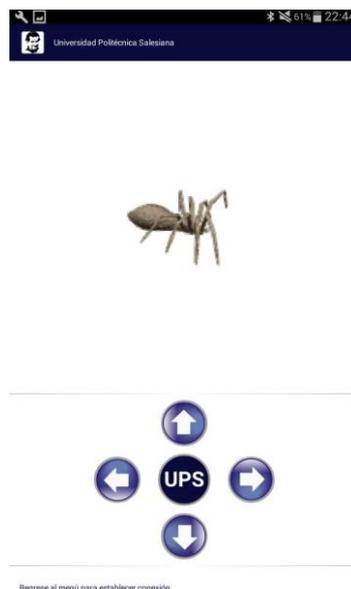
**Figura 29.** Solicitud de permisos de Bluetooth en Aplicación

La aplicación consta con una opción de comunicación en donde busca automáticamente dispositivos que se encuentren cercanos y activados, cuando se elige el dispositivo que se desea conectar la aplicación da un aviso que los dispositivos se encuentran conectados y activos.



**Figura 30.** Conexión con dispositivos vinculados en Aplicación

La aplicación consta con dos opciones de mando, la primera opción es de manual y la otra de automática. En la opción de mando manual existen botoneras direccionales que cuando se presione alguna tecla, se ejecutará la locomoción del robot en el sentido de la tecla activada.



**Figura 31.** Opción Manual en Aplicación.

Al activar la opción de mando automática se ejecutara la locomoción del robot en una secuencia de pasos determinados secuencialmente.



Regrese al menú para establecer conexión

**Figura 32.** Opción Automática en Aplicación

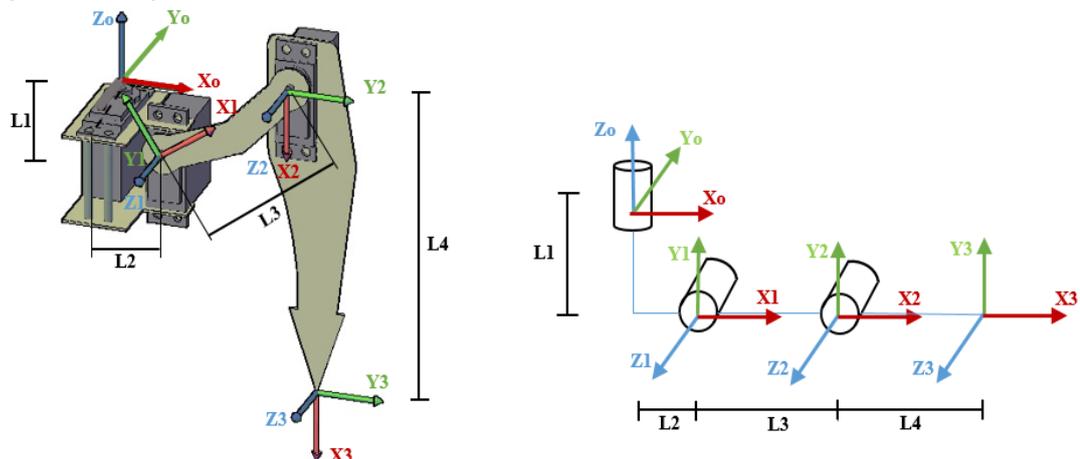
### 3.4 Simulación del Hexápodo en Matlab

La extremidad del robot hexápodo se conforma por 3 articulaciones en donde se produce un movimiento angular, lo que se desea es obtener la posición final del actuador de la extremidad ( $P(X, Y, Z)$ ) en función de los ángulos en cada Articulación (Cinemática Directa) y los grados que corresponden a cada articulación en función de la posición de la extremidad (Cinemática Inversa).

$$(X, Y, Z) = F(\theta_1, \theta_2, \theta_3)$$

$$(\theta_1, \theta_2, \theta_3) = F(X, Y, Z)$$

Para el análisis Cinemático encontramos los parámetros Denavit Hartenberg que corresponden a una extremidad.



**Figura 33.** Asignación de ejes en una Extremidad

## Tabla de Parámetros Denavit Hartenberg

	<b>e</b>	<b>d</b>	<b>a</b>	$\infty$
<b>1</b>	e1	L1	L2	$\pi/2$
<b>2</b>	e2	0	L3	0
<b>3</b>	e3	0	L4	0

Figura 34. Tabla de Parámetros DH

A partir de los parámetros obtenidos se ingresa los parámetros DH en Matlab para luego realizar la simulación del robot. Para tener claros y ordenados los pasos para la programación del robot se realizó un diagrama de flujo como se muestra en la Figura 35.

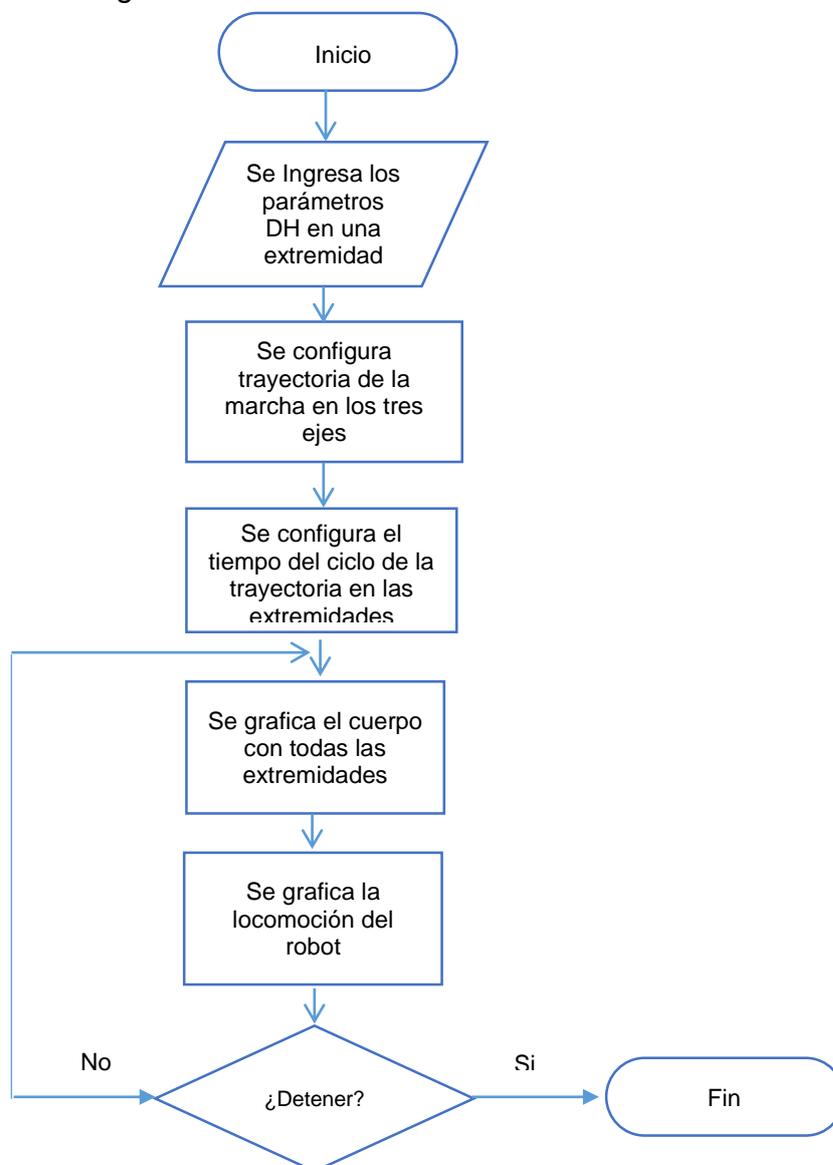


Figura 35. Diagrama de Flujo Simulación en Matlab

La librería rvcTools es la que se utiliza para trabajar con programación enfocado a la robótica. La programación está basado en la creación de eslabones mediante los parámetros Denavit Hartenberg.

Se crea las dimensiones en una extremidad de un robot, como se mencionó anteriormente en cada extremidad tenemos 3 grados de libertad.

```

4 %Locomoción de Robot Hexápodo en Matlab
5 % Se establece las dimensiones de los enlaces de la extremidad, las unidades son en metros.
6 L1 = 0.02;L2=0.02; L3 =0.07;L4= 0.12
7
8 % crear los vínculos de la pierna sobre la base de parámetros DH
9 % theta d a alpha
10 links(1) = Link([ 0 L1 L2 pi/2 ], 'standard');
11 links(2) = Link([ 0 0 L3 0 ], 'standard');
12 links(3) = Link([ 0 0 L4 0 ], 'standard');
13
14 % creamos un robot para representar una sola pierna
15 leg = SerialLink(links, 'name', 'leg', 'offset', [pi/2 0 -pi/2]);
16

```

**Figura 36.**Programación de extremidad de robot en Matlab.

Definimos parámetros de trayectoria para la pierna creada tales como los límites de avance y retroceso para la pata en la tierra, distancia de pie del cuerpo a lo largo del eje y, altura de pie cuando sube y baja.

```

17 % definir los parámetros clave de la trayectoria de la marcha , caminando en la dirección x
18 xf = 10; xb = -xf; % límites de avance y retroceso para los pies en la tierra
19 y = 10; % distancia del pie del cuerpo a lo largo del eje y
20 zu = 2; zd = 10; % altura de pie cuando sube y baja.
21 % definir la trayectoria rectangular tomada por el pie
22 segments = [xf y zd; xb y zd; xb y zu; xf y zu] * 0.01;
23

```

**Figura 37.**Trayectoria de extremidad en Matlab.

A la secuencia que se crea le colocamos los tiempos en los que dura cada segmento, para que se siga un ciclo reitirativo.Luego colocamos las dimensiones de la anchura y la altura del robot para crear el cuerpo rectangular.

```

37 %
38 % Utilizamos un tiempo de aceleración finita para obtener un buen camino %agradable, lo que significa
39 %que el pie nunca pasa realmente a través de cualquiera de estos puntos .
40 %Esto hace que la configuración del robot inicial plantear y velocidad difícil.
41 % creamos una trayectoria más larga ciclica : 1, 2, 3, 4, 1, 2, 3, 4.
42 %El primer segmento de 1 > 2 incluye la rampa inicial y la final 3-> 4 tiene la %desaceleración.
43 %Sin embargo, el medio 2-> 3-> 4-> 1 es ciclico suave.
44
45 segments = [segments; segments];
46 tseg = [3 0.25 0.5 0.25]';
47 tseg = [1; tseg; tseg];
48 x = mstraj(segments, [], tseg, segments(1,:), 0.01, 0.1);
49
50 % ciclo
51 xcycle = x(100:500,:);
52 qcycle = leg.ikine( transl(xcycle), [], [1 1 1 0 0 0], 'pinv');
53
54 %dimensiones del cuerpo rectangular, anchura y altura del robot, las piernas están en alrededor del cuerpo.
55 W = 0.21; L = 0.3;

```

**Figura 38.**Tiempos en cada segmento del robot en Matlab.

Luego creamos las 6 extremidades totales del robot,Cada uno es un clon de la extremidad que construimos anteriormente, tiene un nombre único , y una base para representar su posición en el cuerpo del robot andante .

```

62 - legs(1) = SerialLink(leg, 'name', 'leg1');
63 - legs(2) = SerialLink(leg, 'name', 'leg2', 'base', transl(-L, 0, 0));
64 - legs(3) = SerialLink(leg, 'name', 'leg3', 'base', transl(-L, -W, 0)*trotz(pi));
65 - legs(4) = SerialLink(leg, 'name', 'leg4', 'base', transl(0, -W, 0)*trotz(pi));
66 - legs(5) = SerialLink(leg, 'name', 'leg5', 'base', transl(-L/2, -W, 0)*trotz(pi));
67 - legs(6) = SerialLink(leg, 'name', 'leg6', 'base', transl(-L/2, 0, 0));

```

Figura 39. Extremidades totales del Robot en Matlab.

Se crea gráficamente las extremidades y el cuerpo del Robot Hexápodo considerando la instancia de cada eje de las extremidades en el cuerpo caminante.

```

72 - % dibujar el cuerpo del robot
73 - patch([0 -L -L 0], [0 0 -W -W], [0 0 0 0], ...
74 - 'FaceColor', 'b', 'FaceAlpha', 0.5)
75 - % Declaramos cada eje del robot.
76 - for i=1:6
77 -     legs(i).plot(qcycle(1,:), plotopt);
78 - end
79 - hold off
80 -
81 - % caminar
82 - k = 1;
83 - while 1
84 -     legs(1).plot(gait(qcycle, k, 500, 0), plotopt);
85 -     legs(2).plot(gait(qcycle, k, 700, 0), plotopt);
86 -     legs(3).plot(gait(qcycle, k, 700, 1), plotopt);
87 -     legs(4).plot(gait(qcycle, k, 500, 1), plotopt);
88 -     legs(5).plot(gait(qcycle, k, 500, 1), plotopt);
89 -     legs(6).plot(gait(qcycle, k, 500, 1), plotopt);
90 -     drawnow
91 -     k = k+1;
92 - end

```

Figura 40. Graficas de Extremidades y Cuerpo del Robot en Matlab.

Para poder compilar la programación se agrega la librería rvctools y luego se compila el código, se podrá observar cómo nos muestra las matrices de los parámetros DH y nos grafica en movimiento el Robot Hexápodo.

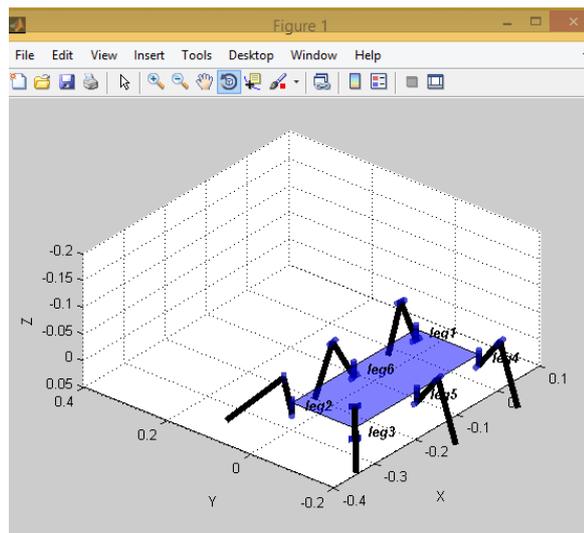


Figura 41. Simulación del Robot Hexápodo en Matlab.

### 3.5 Diseño de Interfaz en Labview

Lo que primero se realizo fue adherir los bloques para la configuración de la cámara, generación, adquisición y procesamiento de la imagen: IMAQdx Open Camera VI , IMAQdx Configure Grab VI , IMAQdx Grab VI .

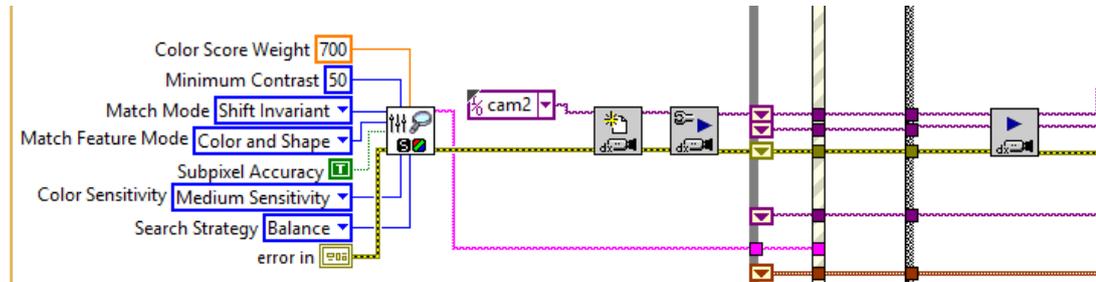


Figura 42. Configuración de bloques DQ en el VI.

En la programación se uso tres tipos de estructura: Se utilizo un “Case Structure” para poder ordenar los estados que tiene el proyecto, en esta estructura se tienen los estados, Inicio, Stand By, Busqueda y Salir.

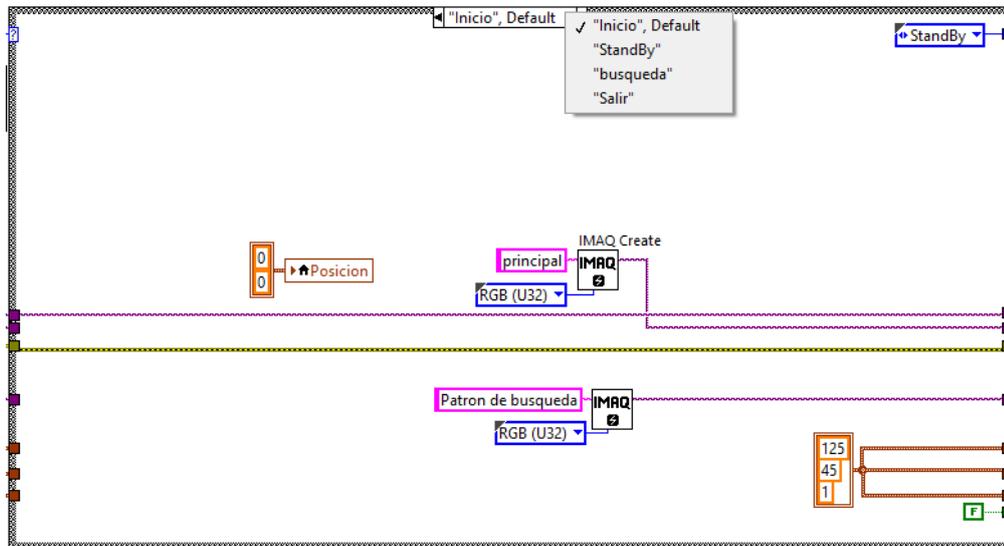


Figura 43. Estados del proyecto en Case Structure

En el primer estado “Inicio” se adquiere y se crea la imagen en tiempo real utilizando el bloque IMAQ Create , en uno crearemos nuestra imagen principal y en otro crearemos el patrón de búsqueda, el tipo de imagen que se utiliza es RGB U32(color). Al momento de dar Run continuo en nuestro programa se mostrara en la imagen principal la imagen adquirida en tiempo real por la cámara USB.

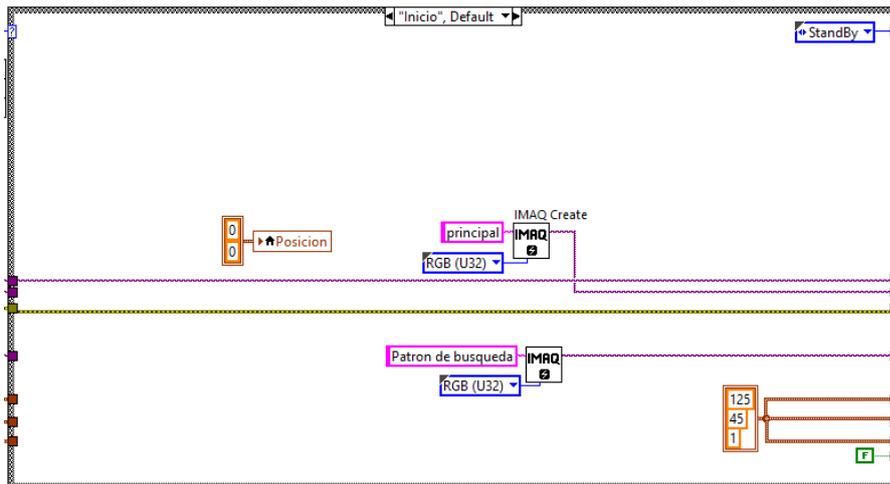


Figura 44. Estado "Inicio" en Case Structure

En el estado "Stand By" el bloque "IMAQdx Grab VI" realiza una grabación continua y procesa el color de la imagen utilizando el bloque "IMAQ ColorBCGLookup VI". Este estado se mantendrá en espera hasta que no se haya activado el botón de evento "Aprender".

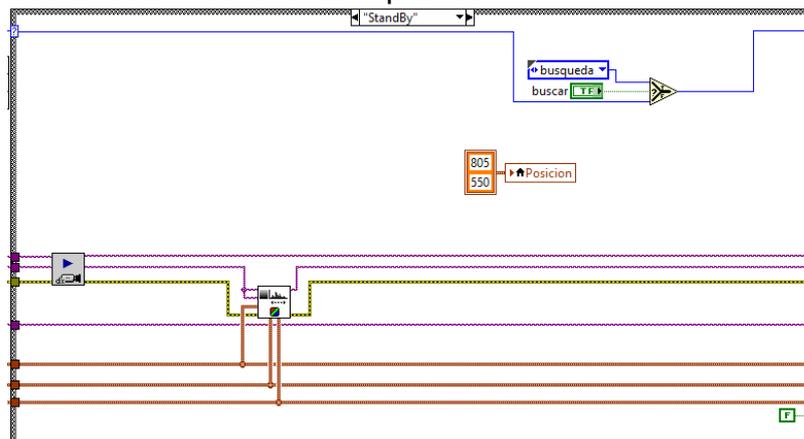


Figura 45. Estado "Stand By" en Case Structure.

En el estado "Busqueda se continua adquiriendo una imagen de grabación mediante el bloque "IMAQdx Grab VI" y se procesa el color de la imagen adquirida mediante el bloque "IMAQ ColorBCGLookup VI". El bloque "IMAQ Match Color Pattern VI" realiza una búsqueda de patrones del color asignado como entrada, cuando realiza la búsqueda se mostrara un rectángulo de color rojo en el patrón seleccionado, esto lo realiza el bloque "Pattern Matching". Los resultados de ambos bloques que es la posición del Match en coordenadas salen en un dato String por ende lo ingreso en un bloque de arreglo "Index Array Function", luego lo separo en dos valores mediante el bloque "Unbundle By Name Function" y muestro los valores de posición (X y Y) en un cuadro de texto de muestra. Estos valores varían dependiendo de la actualización del Match Pattern.

El botón Detener detiene la búsqueda y hace que regrese al estado “Stand By”, esperando nuevamente un nuevo patrón para buscar.

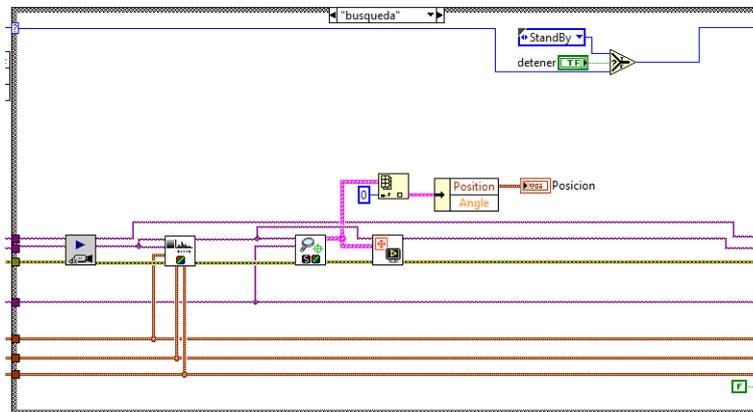


Figura 46.Estado “Búsqueda” en Case Structure.

En el evento salir, detiene todos los eventos y cierra el bloque de grabación de la cámara.

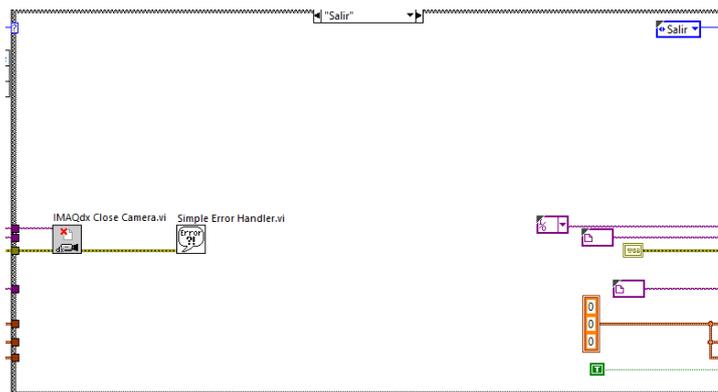


Figura 47.Estado “Salir” en Case Structure.

Se utilizó la estructura “Event Structure” para programar los eventos del proyecto, en el primer evento “time out” se incluyó la estructura “case Structure” en donde se encuentran todos los estados explicados anteriormente.

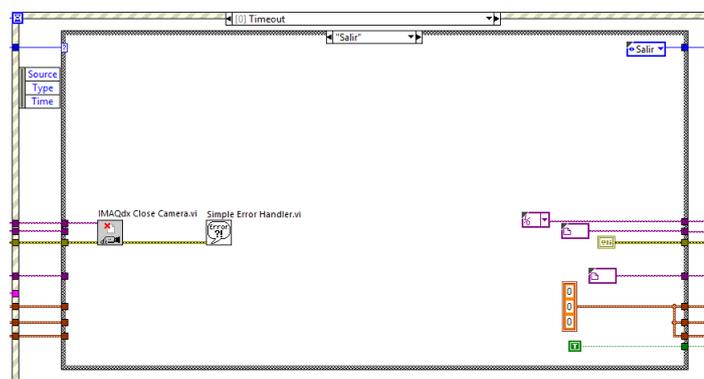
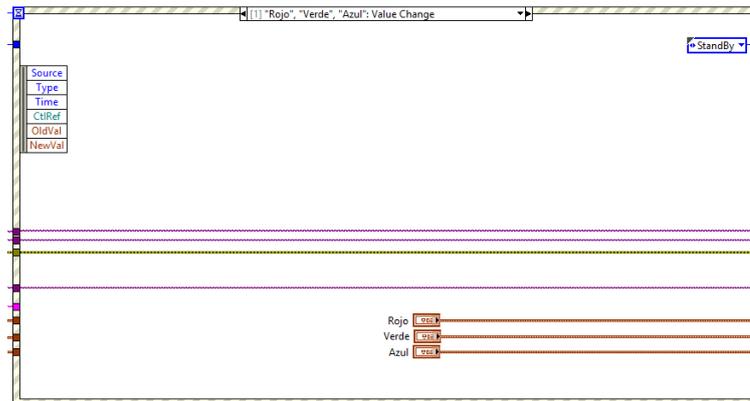


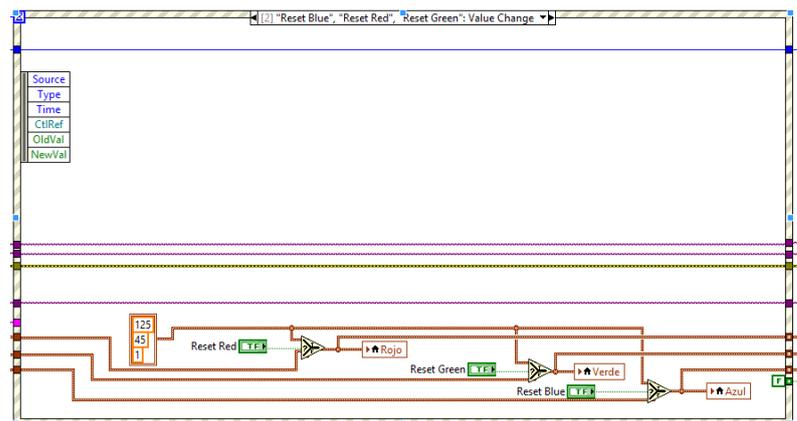
Figura 48.Evento 0 “Time Out” en Case Event Structure

En el evento 1 “Rojo, Verde, Azul”, se encuentran clusters de Brillo, Contraste y Gamma correspondientes al RGB para cuando se requiera poder variar las características de la imagen.



**Figura 49.**Evento 1 “Rojo, Verde, Azul” en Case Event Structure

En el evento 2 “Reset Rojo, Reset Verde, Reset Azul”, se encuentran botonerías de reset para restaurar los valores cambiados de Brillo, Contraste y Gamma correspondientes al RGB.



**Figura 50.**Evento 2 “Reset Rojo, Verde, Azul” en Case Event Structure

El evento 3 “Aprender” se ejecuta seleccionando una región de interés “ROI” , activando el botón Aprender se guarda el extracto de imagen seleccionada la cual sería usada para realizar la búsqueda por patrones, el bloque que extrae la imagen es “IMAQ Extract VI” .

El extracto de imagen guardado es enviado a los bloques “IMAQ Learn Color Pattern VI”  y “MAQ Setup Learn Color Pattern VI” , el cual permite aprender y modelar el color de la imagen que realizaría el seguimiento.

De igual forma al momento de aplastar la tecla aprender mediante un cálculo se relaciona la sección ROI de sus valores en pixeles en valores en centímetros, estos valores de referencia quedan ingresados en ese estado.

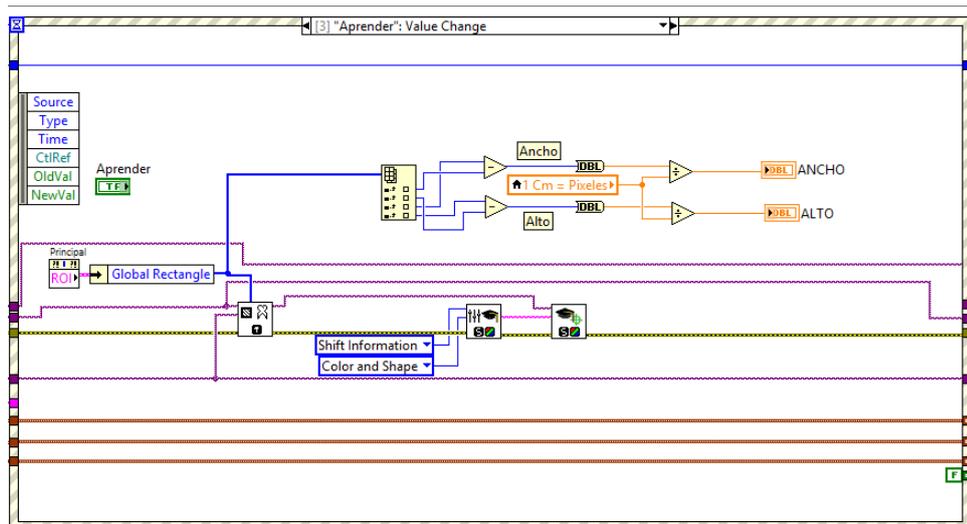


Figura 51.Evento 3 “Aprender” en Case Event Structure

El Evento 4 “Calibración” es activado al presionar el botón de Calibración, lo que realiza es obtener los datos guardados en el estado anterior y con una nueva selección ROI nos calcula como resultado el ancho y el alto del objeto seleccionado que se desea buscar.

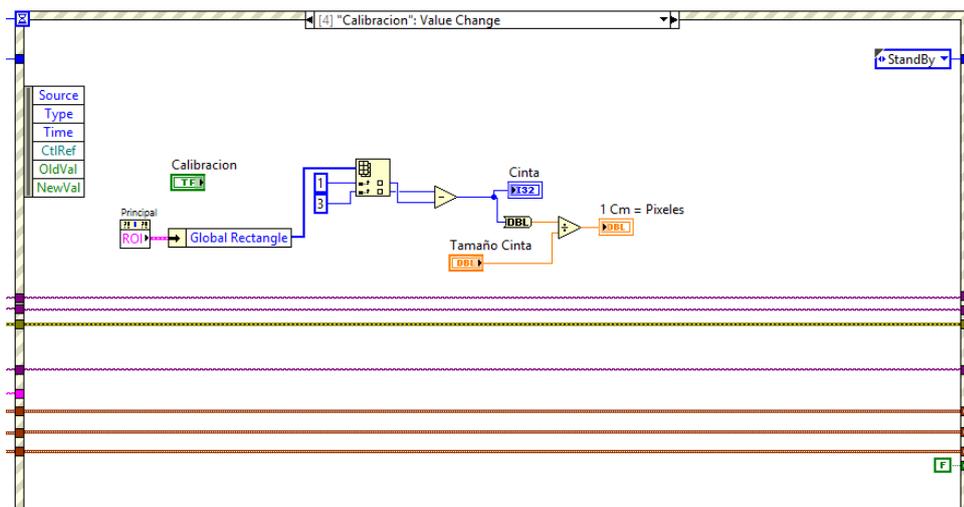
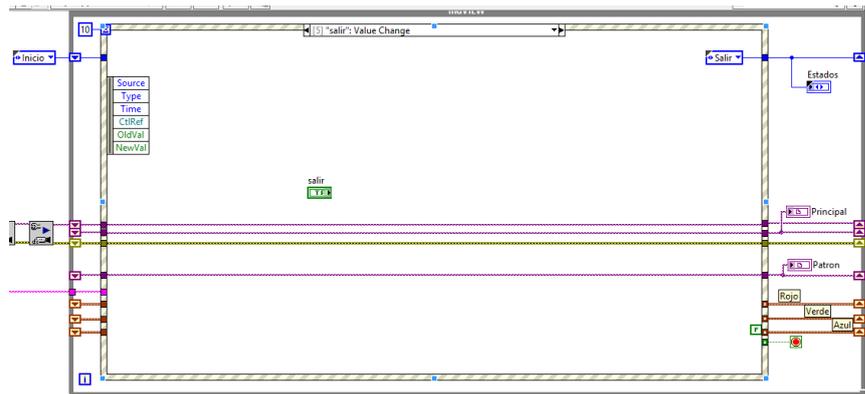


Figura 52.Evento 4 “Calibración” en Case Event Structure

El evento 5 “Salir”, se activa al presionar el botón se Salir. Lo que realiza este evento es activar el Loop condition de la Estructura While Loop que es aquel que engloba las demás estructuras añadidas.



**Figura 53.**Evento 5 “Salir” en Case Event Structure

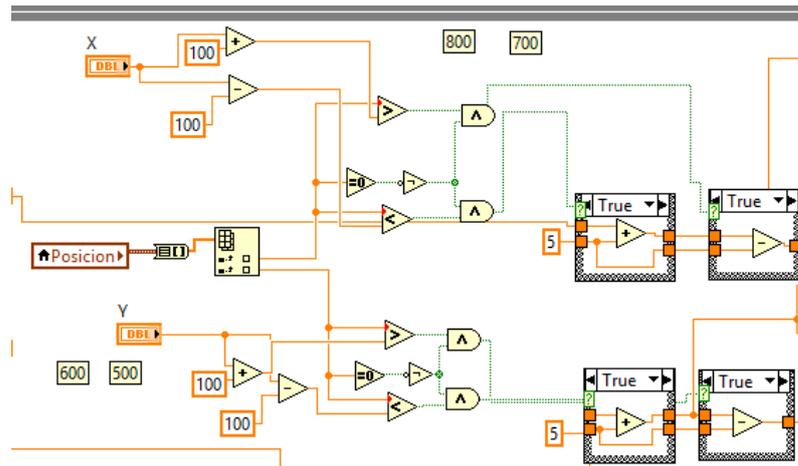
Se creó una nueva estructura While Loop en donde se programará la parte de control de los motores mediante la variable Posición generada por el Match obtenido por la parte de visión. Lo primero que se realiza es crear una variable Local “Posición”, de esta variable se obtiene dos valores de coordenadas correspondientes al eje X y al eje Y, mediante el arreglo “Index Array Function”  separamos los dos valores para manipularlos de mejor forma.



**Figura 54.**Index Array de Variable Local “Posición”

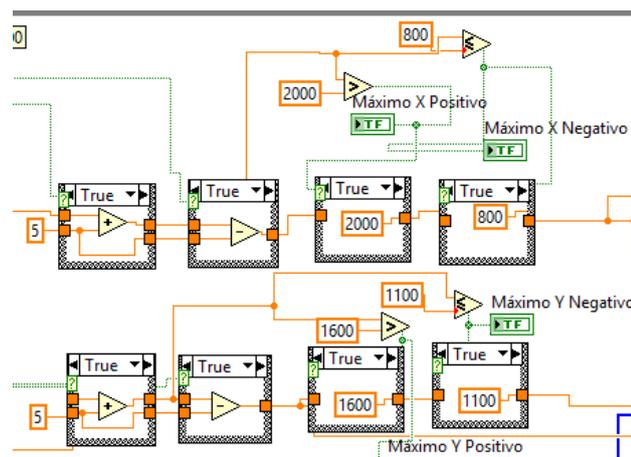
El control que se implementó es un control ON/OFF con Histéresis, tenemos dos Sliders con los que se puede variar el Set Point de la posición en un valor deseado, los valores de histéresis son  $\pm 100$  los cuales se realiza una comparación de “mayor que” y “menor que”, cuando el valor de posición sea mayor o menor al set Point se aumenta o disminuye el valor de ángulo en el servo motor.

De esta forma el servo motor actuara dependiendo de los cambios de valores en las variables de posición y mientras no exista cambios de posición el servomotor se mantendrá en su mismo valor.



**Figura 55.**Control ON/OFF con Histéresis del sistema DEO

Aunque los servomotores tienen un giro de 0 a 180° aproximadamente, se programaron límites en los servomotores para que no se fueren de alguna u otra manera al momento de llegar al tope de su torque. De esta manera cuando lleguen al valor máximo y mínimo colocado se encenderá un indicador y el servo se mantendrá en ese valor hasta que no se considere otro valor aceptado en el rango.



**Figura 56.**Límites de protección para el torque en los servomotores.

Para realizar la comunicación entre el Labview y el controlador Arduino se utilizó el Tool “LINX”, el Tool “LINX” para el servomotor trabaja con un rango de valores en PWM entre 500 a 2500 para ser mejor visible en nuestra interfaz gráfica se realizó una conversión a grados de 0 a 180°: Los bloques del Tool “LINX” “Initialize”  es aquel que comunica inicializando el Arduino con el Labview, “Servo Open”  indica cuantos servomotores se conectaran al controlador, “Servo Set Pulse Width”  lee y escribe los valores de ancho de pulso que se asigna a los servomotor, “HC-SR04”  recibe y muestra los valores en tiempo de real del sensor Ultra sónico(centímetros y pulgadas),

“Servo Close”  cierra los bloques de iniciación del servo, “Close”  cierra la comunicación con el controlador Arduino.

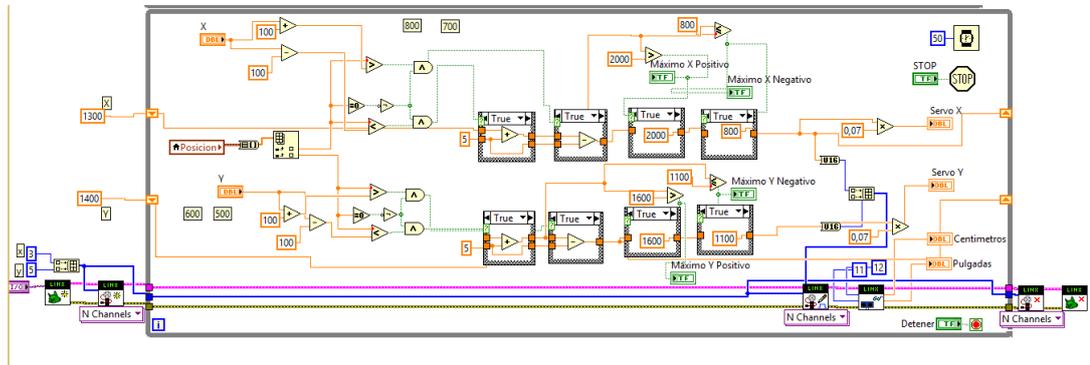


Figura 57. Control ON/OFF – LINX Labview

## 4. RESULTADOS

- Se realizó la programación de la locomoción del Robot Hexápodo utilizando el software y el controlador de Arduino: se utilizó la estrategia de triangulo para obtener mayor estabilidad en los movimientos del robot.
- Se implementó una comunicación inalámbrica bluetooth entre el controlador del robot y la aplicación instalada en el Dispositivo Móvil: mediante el modulo bluetooth y la aplicación en el dispositivo Móvil se envía y se recibe datos tipo carácter, estos activan una secuencia de movimiento asignada a una letra en particular.
- Se construyó la estructura mecánica del robot Hexápodo utilizando aluminio como material resistente y liviano para el alargue de vida de tiempo del robot: la estructura fue medida y cortada a mano utilizando las herramientas necesarias para realiza el acople necesario de las piezas.
- Se Implementó un Director Electro Óptico (DEO) utilizando herramientas de Visión Artificial: el sistema DEO consta de una estructura base soporte, una cámara USB HD, un Sensor Ultrasónico, un controlador Arduino conectado mediante USB al Software Labview.
- Se realizó algoritmos de visión artificial para la adquisición y tratamiento de imágenes mediante Labview: Mediante el software Labview se adquiere y procesa la imagen obtenida por la cámara, se adquiere variables físicas como tamaño, distancia y posición del objeto a seguir.
- Se utilizó un control on/off con histéresis para controlar los servomotores que permiten el movimiento de la cámara sobre el eje vertical (X) y el eje horizontal (Y).
- Se diseñó una aplicación personalizada para el Sistema operativo Android para el control inalámbrico del Robot Hexápodo: esta aplicación consta dos opciones una manual y otra automática.
- Se elaboró 4 prácticas de desarrolladas para que el estudiante pueda utilizarlas como guía de aprendizaje:
  - Análisis y Simulación de la locomoción del Robot Hexápodo mediante el uso de Matlab aplicando cinemática del robot y los parámetros Denavit Hartenberg
  - Movimiento de Servomotores utilizando el controlador Arduino y Labview.
  - Realizar una aplicación en Android para establecer comunicación entre bluetooth y Arduino.
  - Variables Físicas de un objeto en movimiento utilizando el Director Electro Óptico.
- Se realizó la programación de locomoción del robot Hexápodo utilizando el estudio de los Parámetros Denavit Hartenberg.

A continuación, se detallan cada uno de los objetivos planteados a inicio:

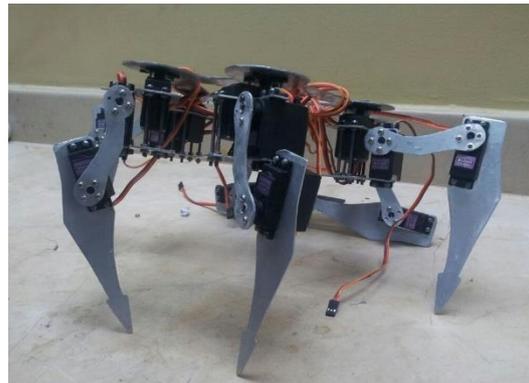
#### 4.1 Planos

Se realizó el diseño de los planos de la estructura del robot y de la base soporte del DEO. Posteriormente se implementaron los planos y se realizaron pruebas tanto de movimiento, peso, equilibrio y se realizó un acabado en la estructura.

Estos planos se los puede apreciar en el Anexo 1.



**Figura 58.** Corte y Ensamblaje de Piezas del Robot (1)

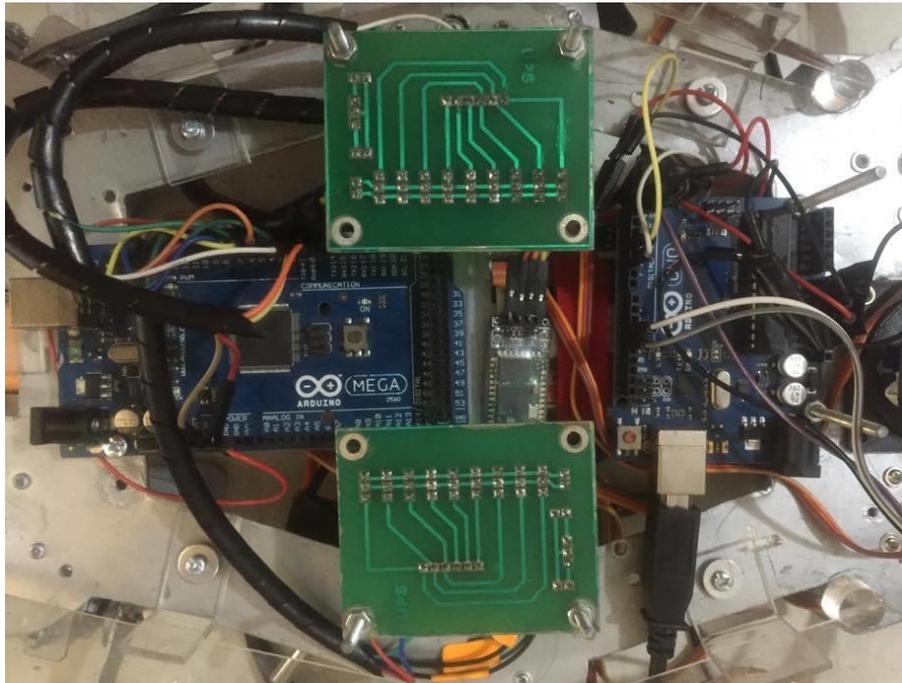


**Figura 59.** Corte y Ensamblaje de Piezas del Robot (2)

## 4.2 Diseño de Baquelitas de alimentación y control

Se realizó el diseño de las tarjetas del Hexápodo utilizando la plataforma PROTEUS (Ares), se imprimió las pistas de las tarjetas en un material bueno y resistente (fibra de vidrio). Se implementaron las tarjetas en el robot y se realizaron pruebas de alimentación y de control del Robot.

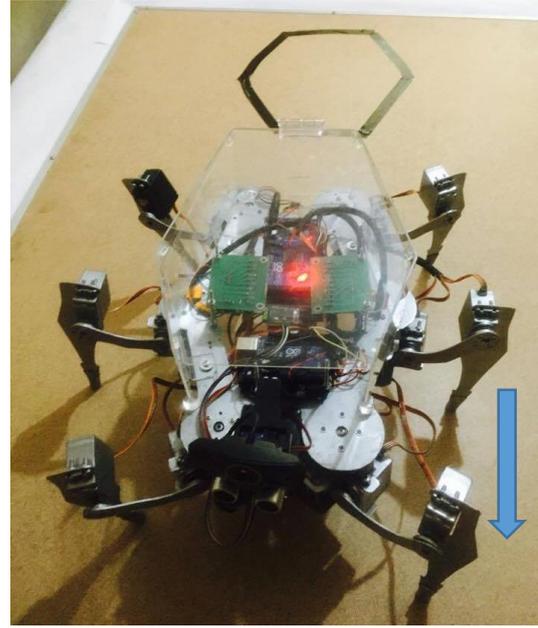
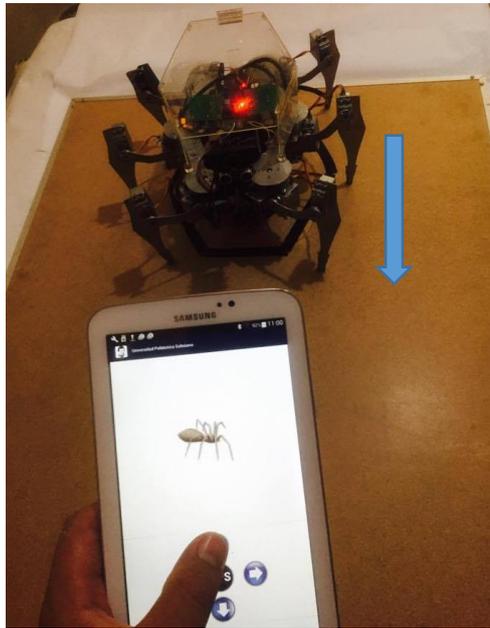
Diagrama de Conexiones se lo puede apreciar en el Anexo 2.



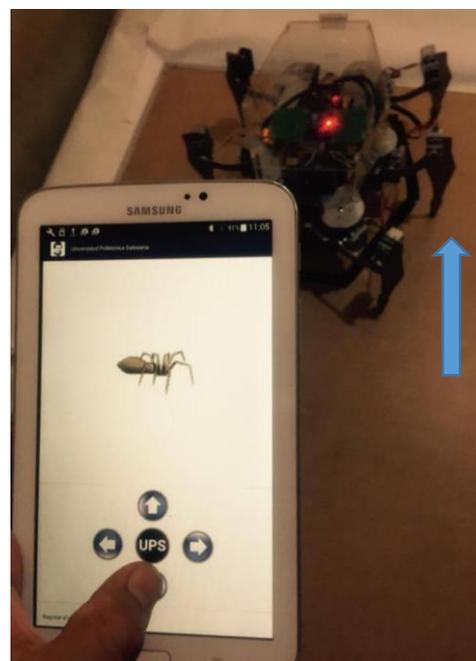
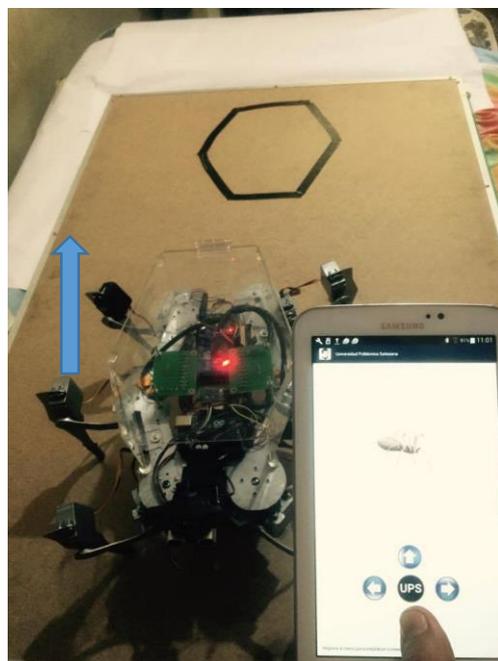
**Figura 60.** Tarjetas Impresas colocadas en el Robot Hexápodo

## 4.3 Diseño de Aplicación en Android

Se realizó el diseño de la aplicación en Android, basándose en que se tenga una buena interacción entre el robot y el usuario, llegando a obtener una aplicación con dos funcionalidades una manual y otra automática. En la forma manual, el usuario controla mediante teclas de posición los pasos del robot. La forma automática el robot sigue una secuencia definida de pasos automáticamente.



**Figura 61.** Boton Adelante Click Down - Click UP



**Figura 62.** Botón Atrás Click Down - Click UP

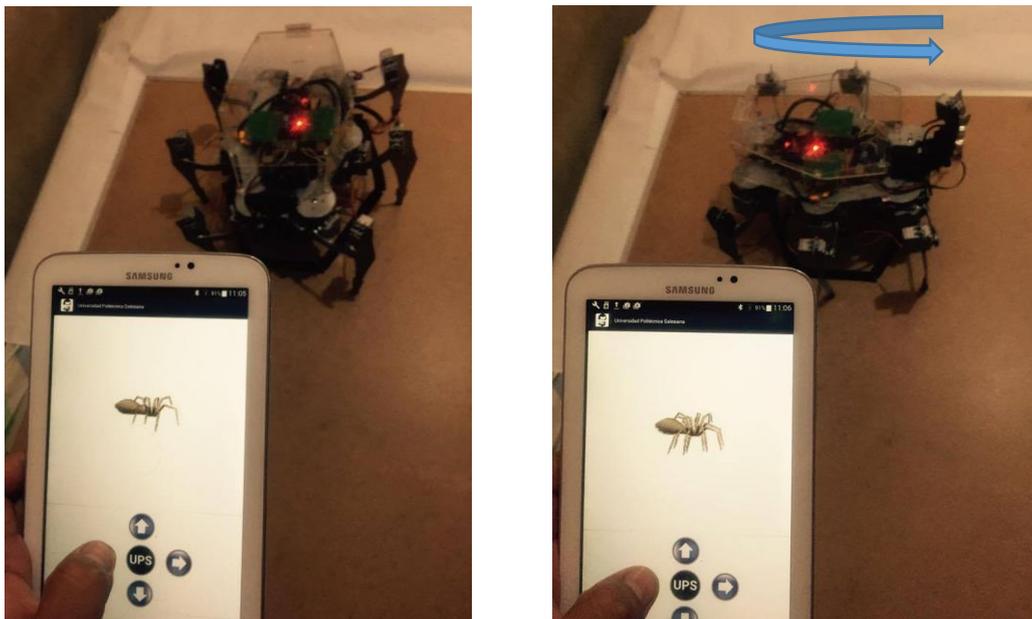


Figura 63. Botón Izquierda Click Down –Click UP

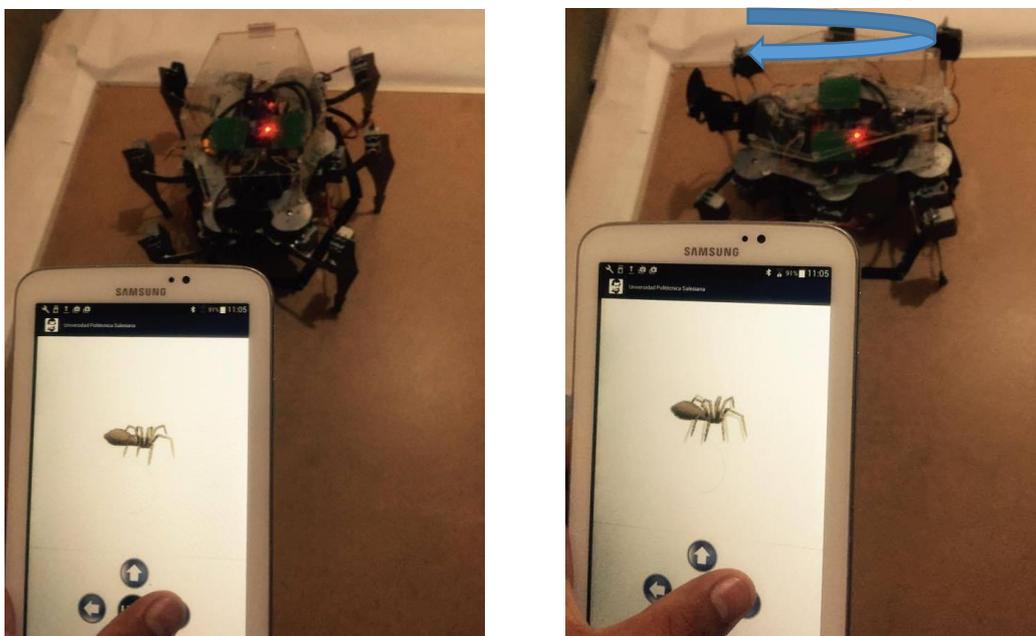


Figura 64. Botón Derecha Click Down –Click UP

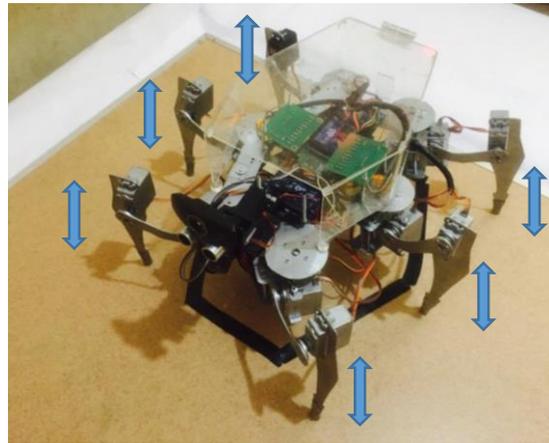
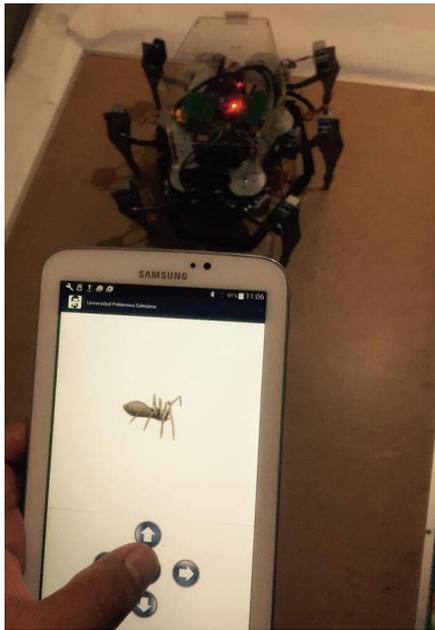


Figura 65. Botón Arriba-Abajo Click Down–Click UP

#### 4.4 Programación y Simulación del Robot Hexápodo en Matlab

Se realizó la Simulación del robot Hexápodo utilizando los parámetros Denavit Hartenberg de la librería rvcTools en Matlab. La programación simula la locomoción del robot en una vista 3D.

La programación en Matlab se lo puede apreciar en el anexo 3.

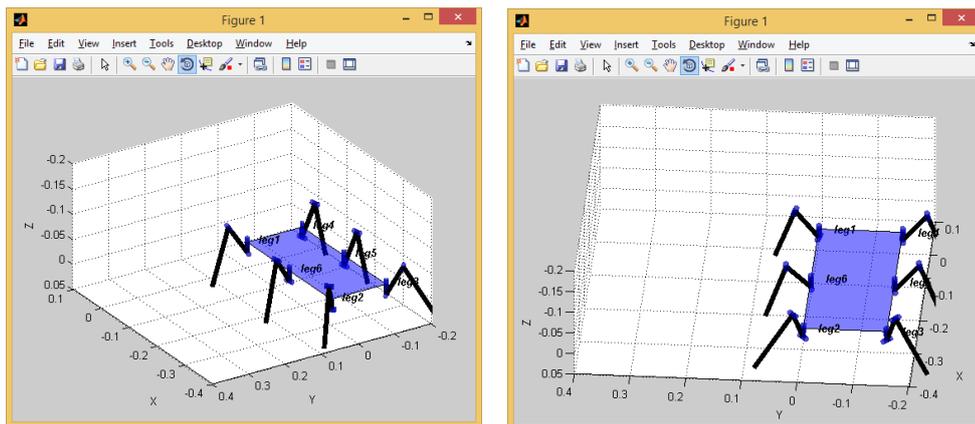


Figura 66. Simulación de Robot Hexápodo en Matlab (1)

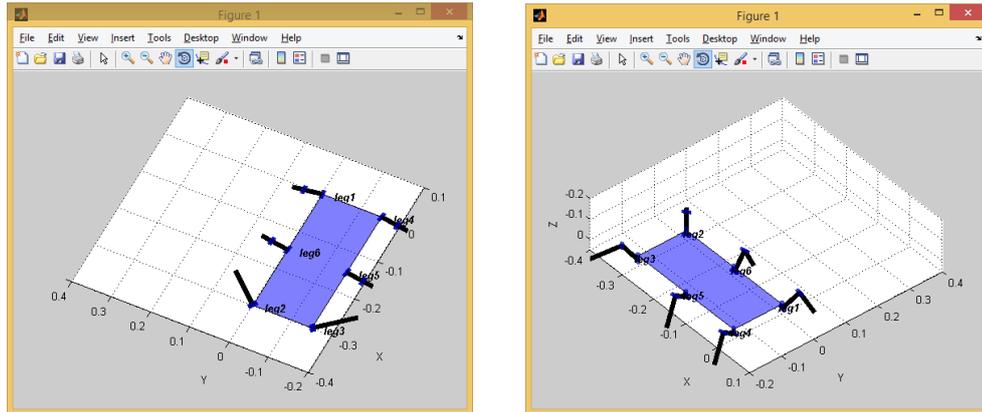


Figura 67. Simulación de Robot Hexápodo en Matlab (2)

#### 4.5 Programación del Sistema DEO en Labview.

La programación del sistema DEO se lo realizo en la plataforma Labview mediante librerías de visión artificial y una cámara USB para adquirir la imagen en tiempo real. Para la estructura de la programación se utilizó maquinas secuenciales, diagramas de estado en donde seguimos una secuencia lógica y funcional. Obteniendo así resultados positivos.

La programación en Labview se lo puede apreciar en el anexo 5.



Figura 68. Posición de Servos vista central de sistema DEO

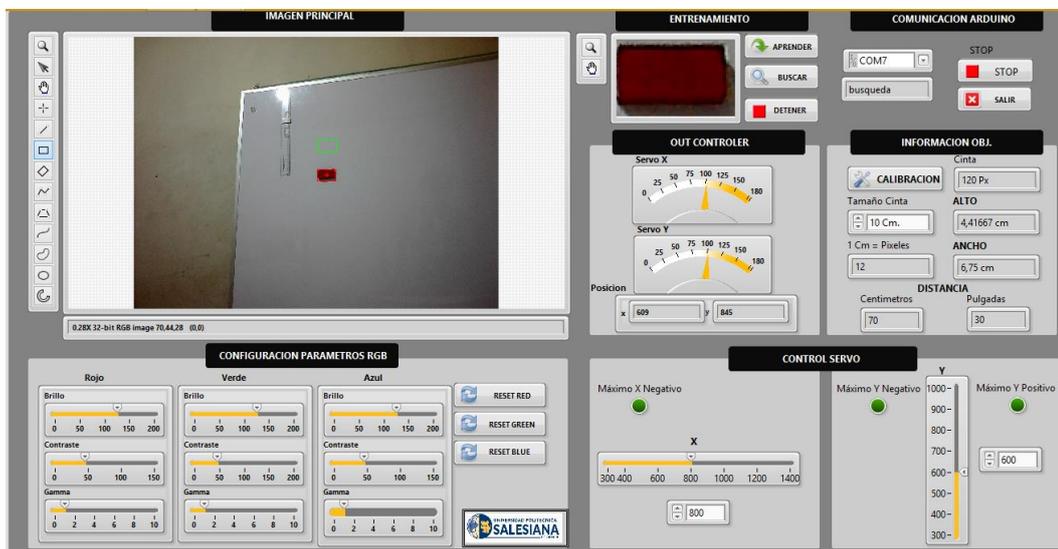


Figura 69. Posición de Servos vista lateral de sistema DEO



Figura 70. Posición de Servos vista lateral de sistema DEO

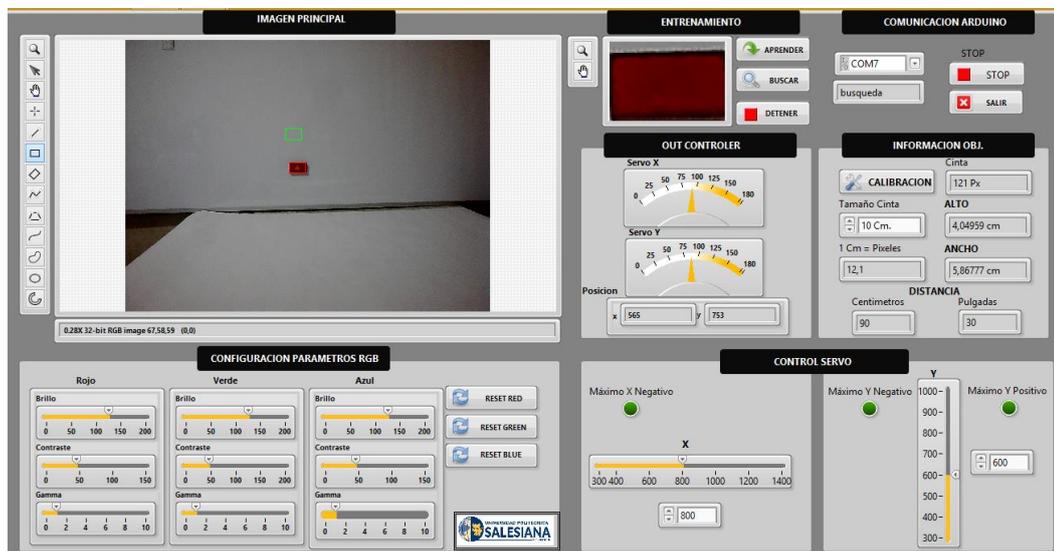


Figura 71. Posición de Servos vista Inferior de sistema DEO

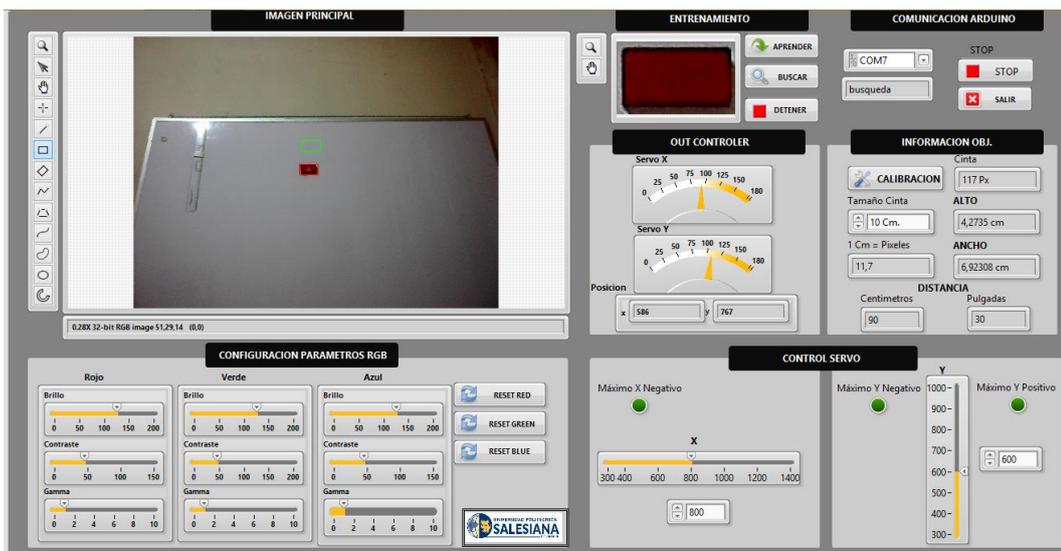
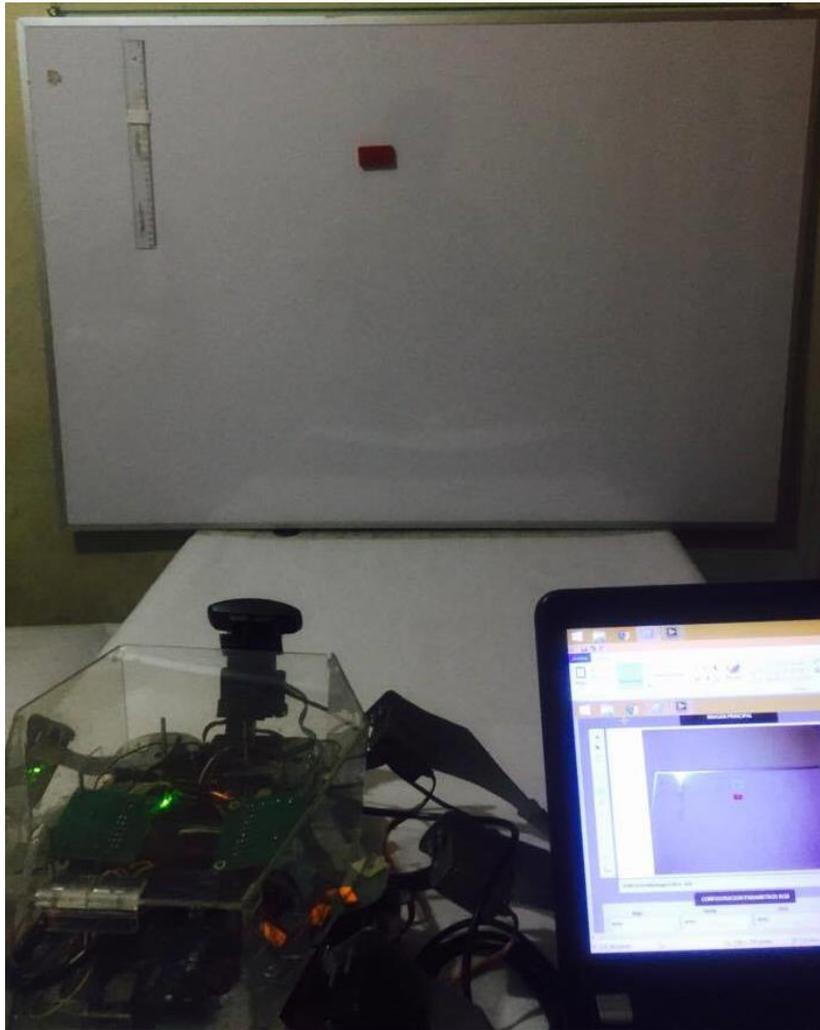


Figura 72. Posición de Servos vista Superior de sistema DEO



Figura 73. Posición de Servos Máximo X Positivo de sistema DEO



Figura 74. Posición de Servos Máximo X Negativo de sistema DEO.

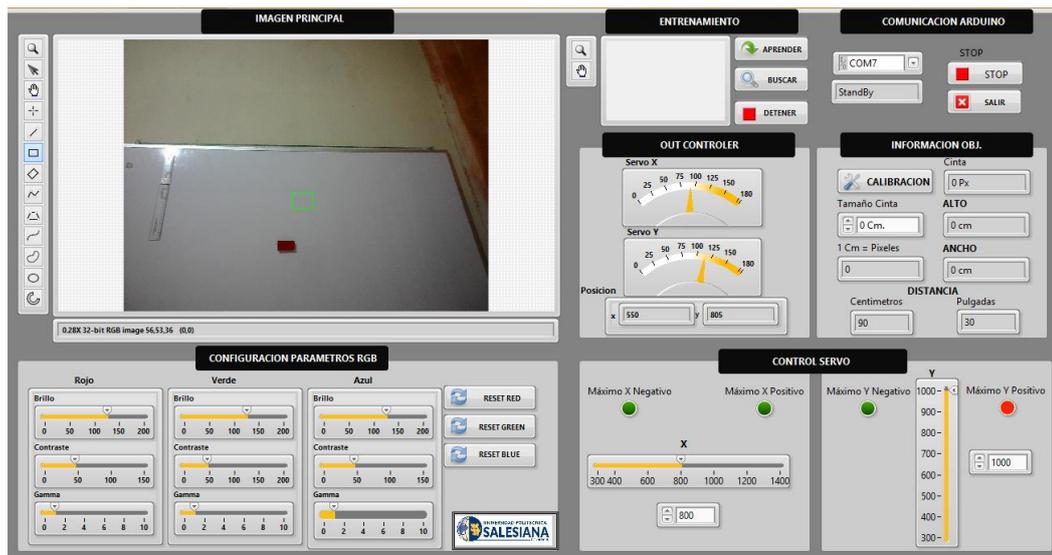
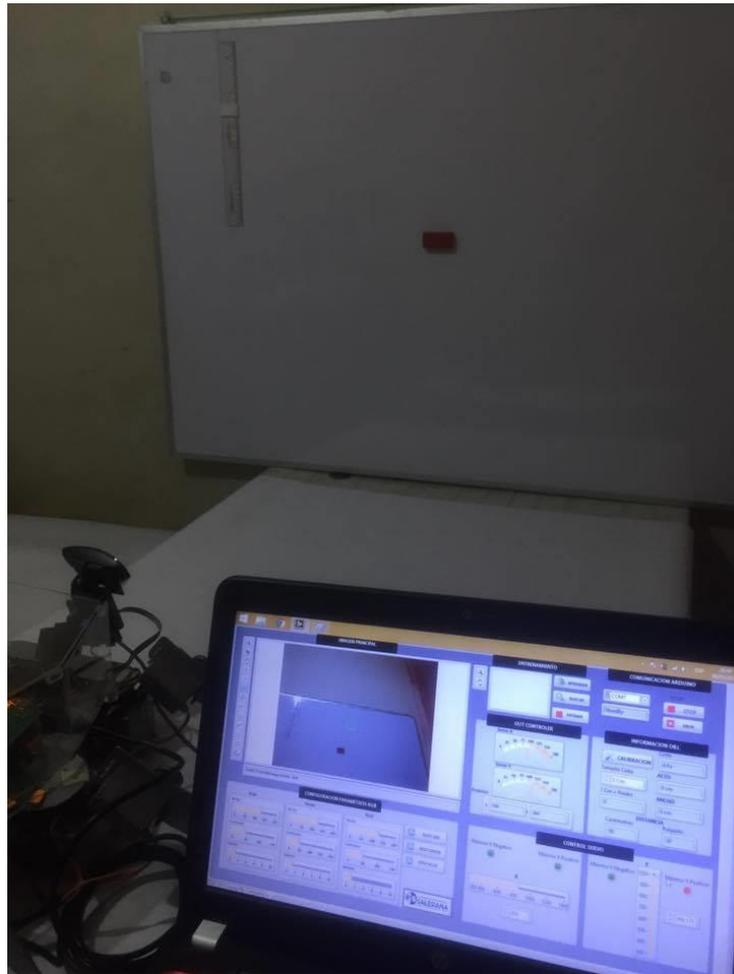


Figura 75. Posición de Servos Máximo Y de sistema DEO.

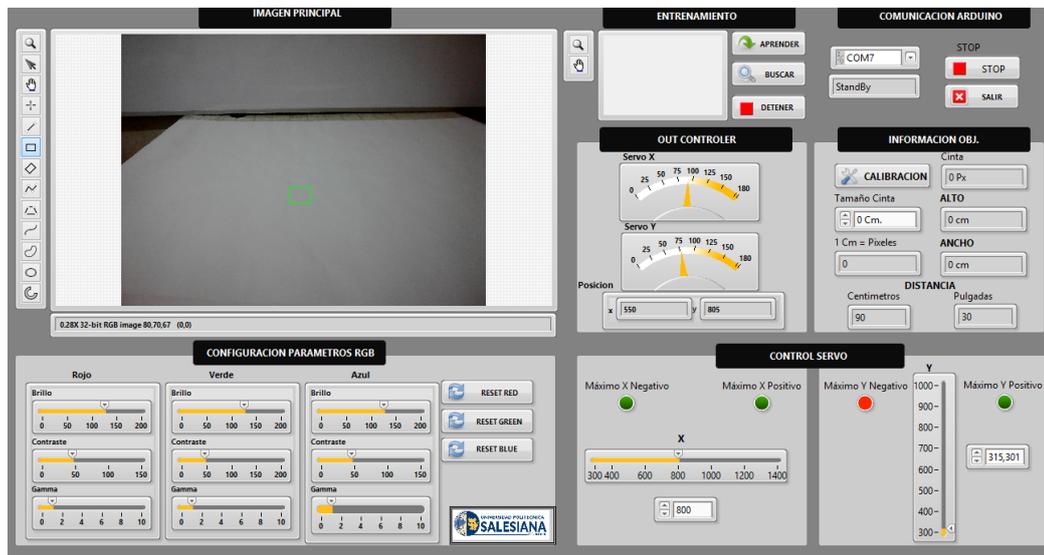
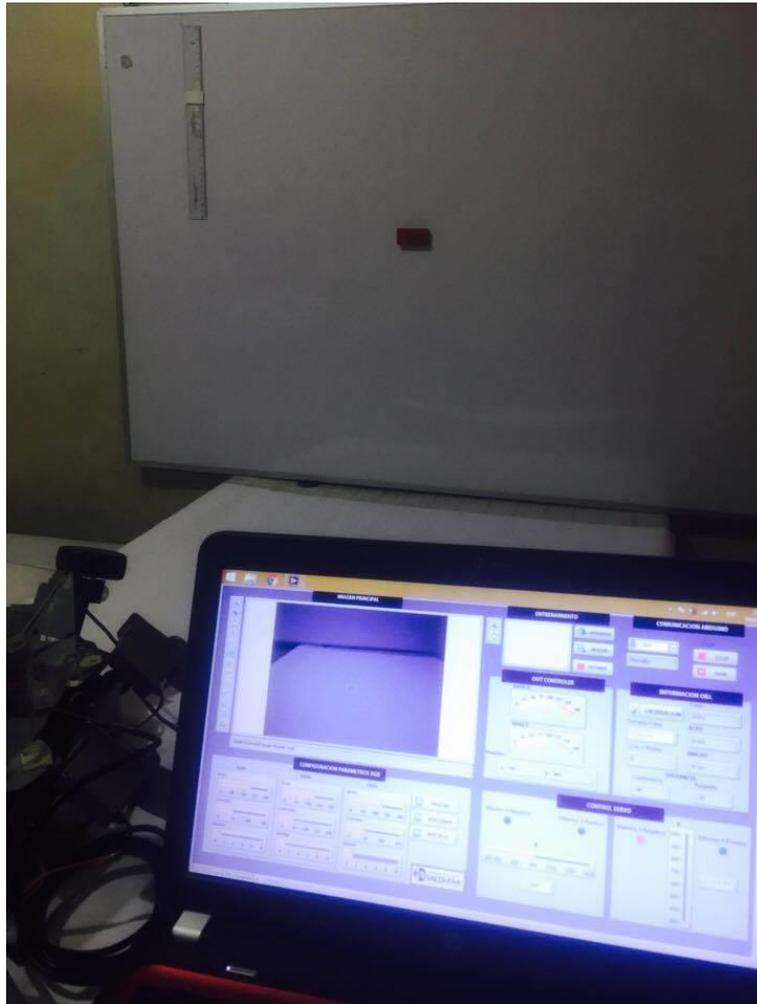


Figura 76. Posición de Servos Máximo Y Negativo de sistema DEO.

## **CONCLUSIONES Y RECOMENDACIONES**

### **Conclusiones**

El proyecto de titulación pudo realizarse con éxito, empleando los conocimientos adquiridos a lo largo de la carrera. Ha sido de vital importancia elaborar con anticipación los diseños, planos, así como realizar el diseño de la lógica secuencial en la visión artificial. Se pudo manipular los valores de coordenadas adquiridos de posición de seguimiento del objeto y se los asigno como valores controlados en nuestros servomotores del sistema DEO.

En la locomoción del Robot Hexápodo, se obtuvo resultados positivos mediante la interacción de la aplicación en Android y el controlador Arduino del Robot, influyo mucho la estructura de aluminio que se utilizó la cual es liviana y resistente y ayudo que la locomoción tuviera mejor rendimiento.

Se obtuvieron 4 prácticas experimentales para que los estudiantes de la universidad puedan experimentar los objetivos específicos del proyecto.

La implementación de esta plataforma de desarrollo representa una gran ayuda para el desarrollo de futuras líneas de investigación por poseer recursos de altas prestaciones, buena calidad y reducido costo.

### **Recomendaciones**

En el sistema DEO es importante considerar factores negativos que podrían influir en el rendimiento del sistema tales como: la iluminación del espacio, la resolución de la cámara, características del objeto.

Es importante considerar el material que se utilizó para la construcción del robot Hexápodo, ya que tiene que ser ligero para que los servomotores puedan levantar la estructura, las tarjetas y demás componentes, por ende por su alto torque se utilizó servo motores de engranaje metálico.

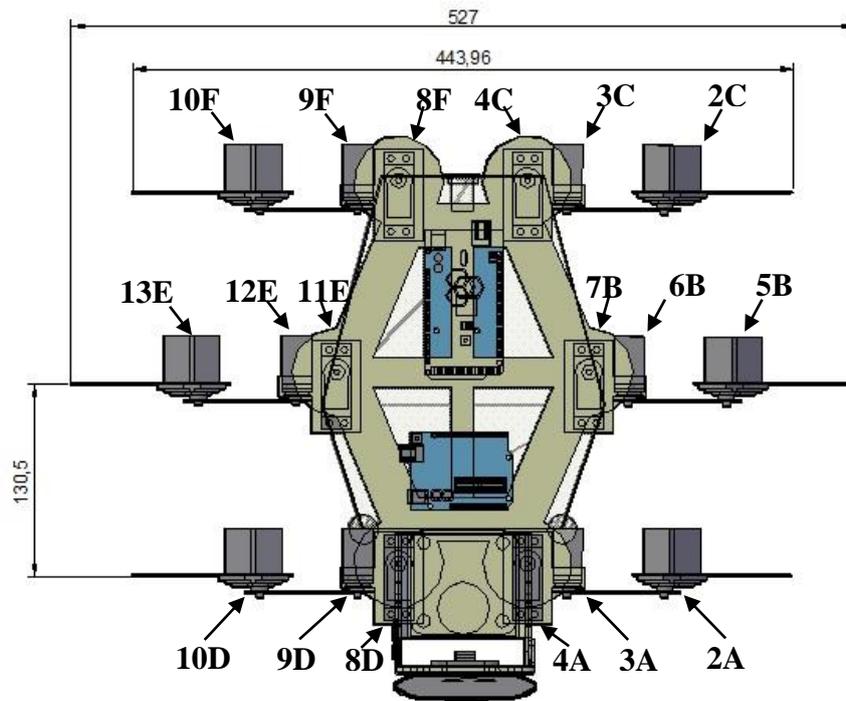
Se recomienda leer los informes y resolver las prácticas previo al uso del proyecto para que se obtenga un mejor manejo del mismo.

## REFERENCIAS BIBLIOGRAFICAS

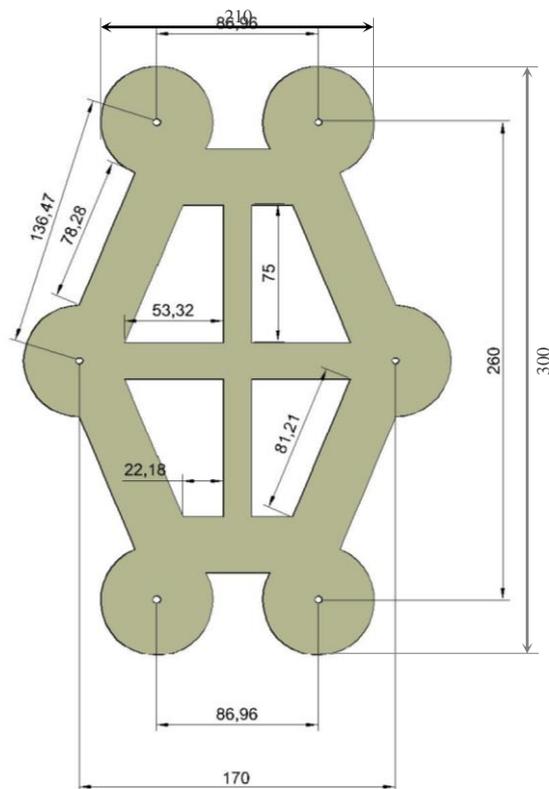
- Arduino. (23 de 01 de 2015). *Arduino mega 2560*. Obtenido de <http://arduino.cc/en/main/arduinoboardmega2560>
- ArduinoWebSite. (2017). *Arduino*. Obtenido de <https://www.arduino.cc/en/Main/arduinoBoardMega2560>
- Automation Direct. (14 de 02 de 2015). *Motor DC*. Obtenido de [http://www.automationdirect.com/adc/Overview/Catalog/Motors/DC\\_Motors\\_\\_General\\_Purpose\\_IronHorse\\_\(up\\_to\\_2HP\)/56C\\_Permanent\\_Magnet\\_DC\\_Motors\\_-z\\_90\\_VDC\\_Armature\\_\(0.33\\_-\\_1.5HP\)](http://www.automationdirect.com/adc/Overview/Catalog/Motors/DC_Motors__General_Purpose_IronHorse_(up_to_2HP)/56C_Permanent_Magnet_DC_Motors_-z_90_VDC_Armature_(0.33_-_1.5HP))
- Compras, C. (2016). *Compras Compras*. Obtenido de <http://www.compracompras.com/ar/producto/620748367/web-cam-genius-facecam-2020-hd-720p-2m-no-mic>
- Electronica, 5. H. (2017). *Electronica, 5 Hertz*. Obtenido de <http://5hertz.com/tutoriales/?p=571>
- ElectronicLab. (2017). *ElectronicLab*. Obtenido de <https://electronilab.co/tienda/sensor-de-distancia-de-ultrasonido-hc-sr04/>
- fbuenonet. (5 de marzo de 2015). *Thingiverse*. Obtenido de <http://www.thingiverse.com/thing:708819>
- González, V. R. (2004). *Control y Robotica*. Obtenido de Valladolid II: [http://platea.pntic.mec.es/vgonzale/cyr\\_0204/ctrl\\_rob/robotica/sistema/motores\\_servo.htm](http://platea.pntic.mec.es/vgonzale/cyr_0204/ctrl_rob/robotica/sistema/motores_servo.htm)
- Google. (18 de 01 de 2015). *Google maps*. Obtenido de <https://www.google.com/maps/d/viewer?mid=zDh2nynNpFjw.kTFO3nPvHmYo&ie=UTF8&oe=UTF8&msa=0>
- HobbyKing. (2016). *HobbyKing*. Obtenido de [https://hobbyking.com/en\\_us/zippy-flightmax-8000mah-2s1p-30c.html](https://hobbyking.com/en_us/zippy-flightmax-8000mah-2s1p-30c.html)
- LABVIEW. (29 de 05 de 2016). *National Instruments*. Obtenido de <http://www.ni.com/labview/esa/>
- National Instruments. (2005). *NI Vision for Labiew, user manual*. Obtenido de [www.ni.com: http://www.ni.com/pdf/manuals/371007b.pdf](http://www.ni.com: http://www.ni.com/pdf/manuals/371007b.pdf)
- Pomares, J. (03 de 12 de 2014). *Manual de arduino*. Obtenido de <http://rua.ua.es/dspace/bitstream/10045/11833/1/arduino.pdf>
- Pro, T. (22 de enero de 2017). *electronicoscaldas*. Obtenido de [http://www.electronicoscaldas.com/datasheet/MG996R\\_Tower-Pro.pdf](http://www.electronicoscaldas.com/datasheet/MG996R_Tower-Pro.pdf)
- Pro, T. (2017). *Hacker Development*. Obtenido de <http://hacker.instanet.net/Kits/MicroServo.shtml>
- Proto Pic. (16 de 02 de 2015). *Arduino Mega*. Obtenido de <http://proto-pic.co.uk/arduino-mega2560-rev-3/>
- Ruben, J. (21 de Febrero de 2014). *GeekFactory*. Obtenido de <http://www.geekfactory.mx/tutoriales/bluetooth-hc-05-y-hc-06-tutorial-de-configuracion/>
- SOBRADO, E. A. (2003). *SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO Y MANIPULACIÓN DE OBJETOS UTILIZANDO UN BRAZO ROBOT*. lima.

# **ANEXO 1**

**PLANOS DEL ROBOT HEXÁPODO**

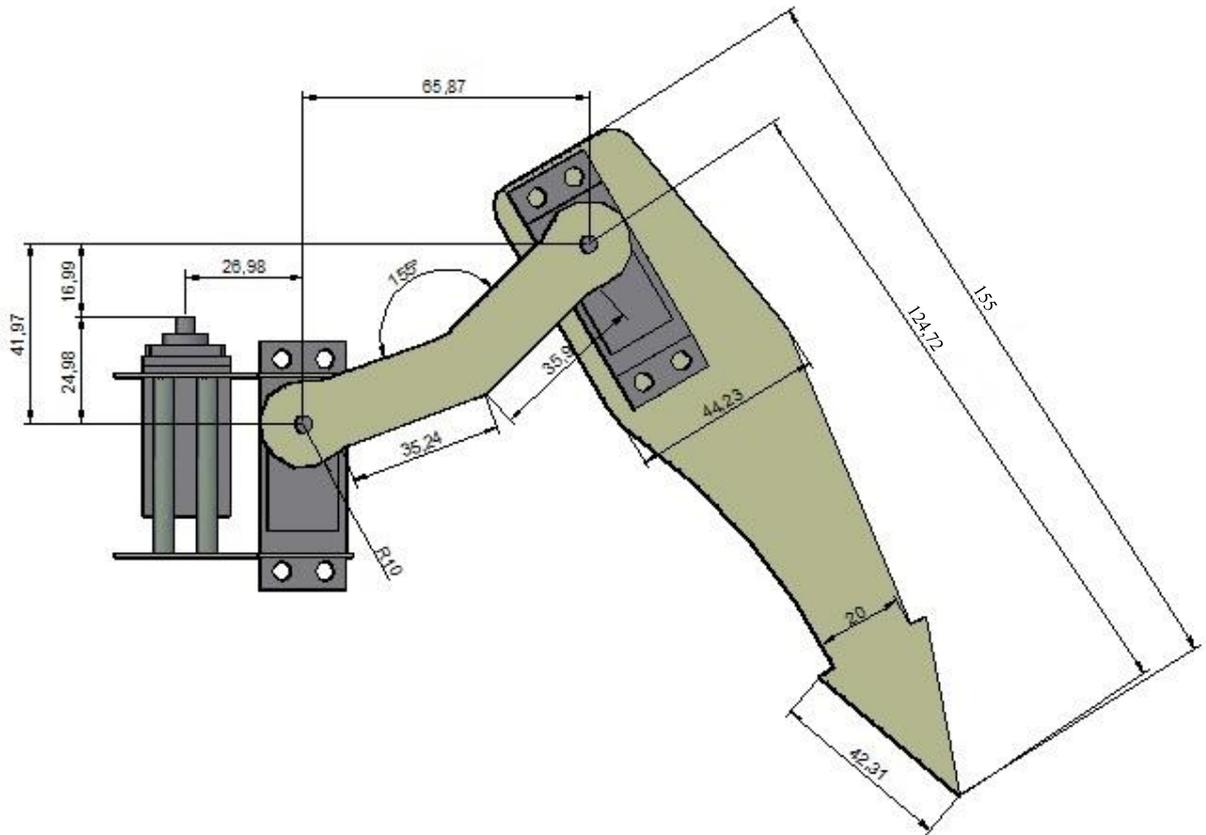


**Distribución de Servomotores del Hexápodo**

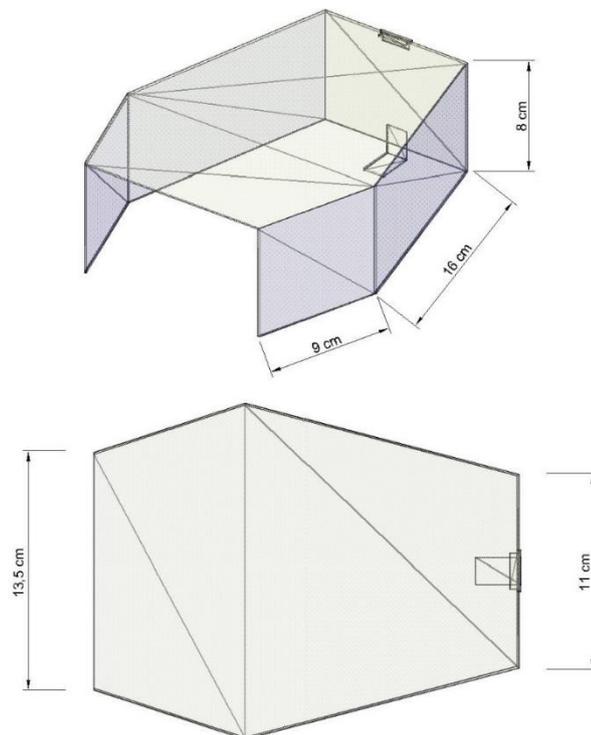


**Diseño del Soporte Principal**

Las unidades de las cotas estan en milímetros

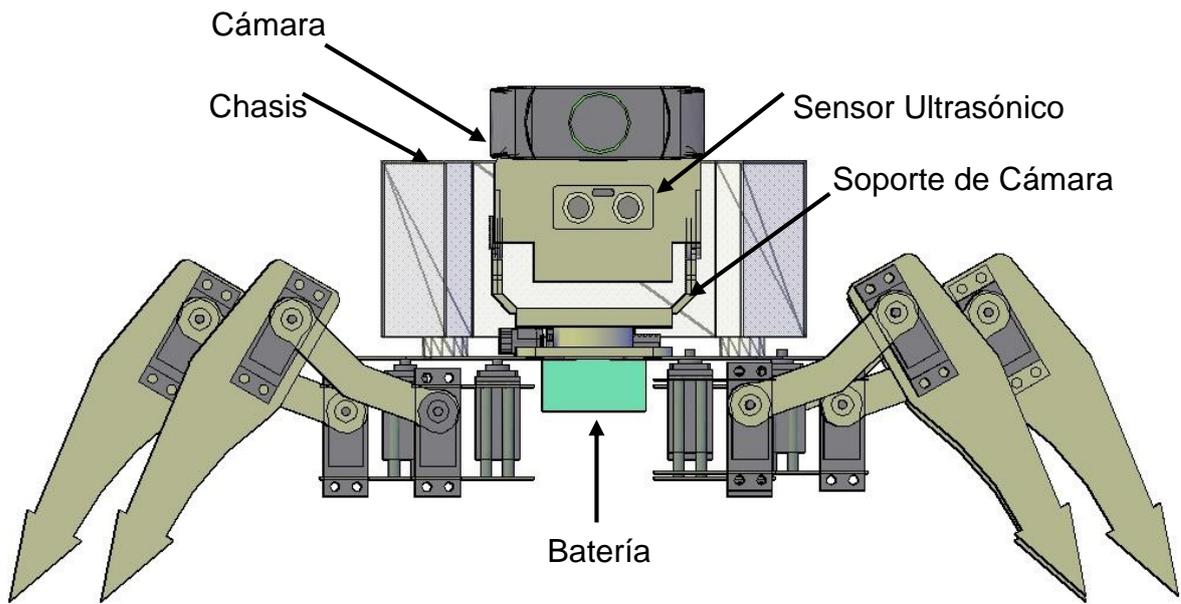


**Diseño estructural de una pata**

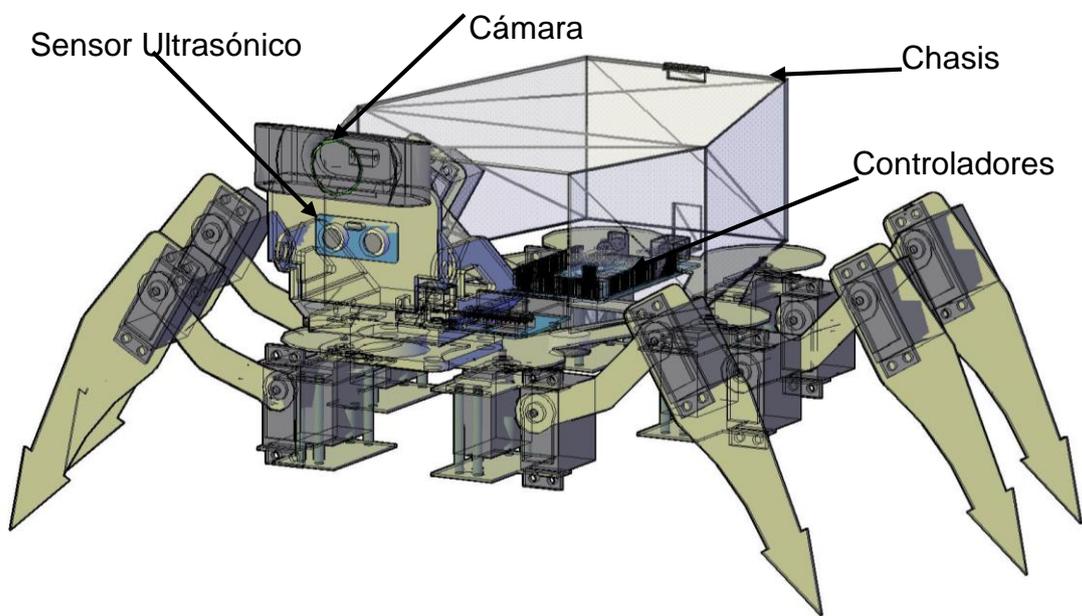


**Diseño de chasis para protección**

Las unidades de las cotas estan en milímetros



**Vista Frontal del Hexápodo**



**Vista Isométrica del Hexápodo**

# **ANEXO 2**

**DIAGRAMA DE CONEXIONES**

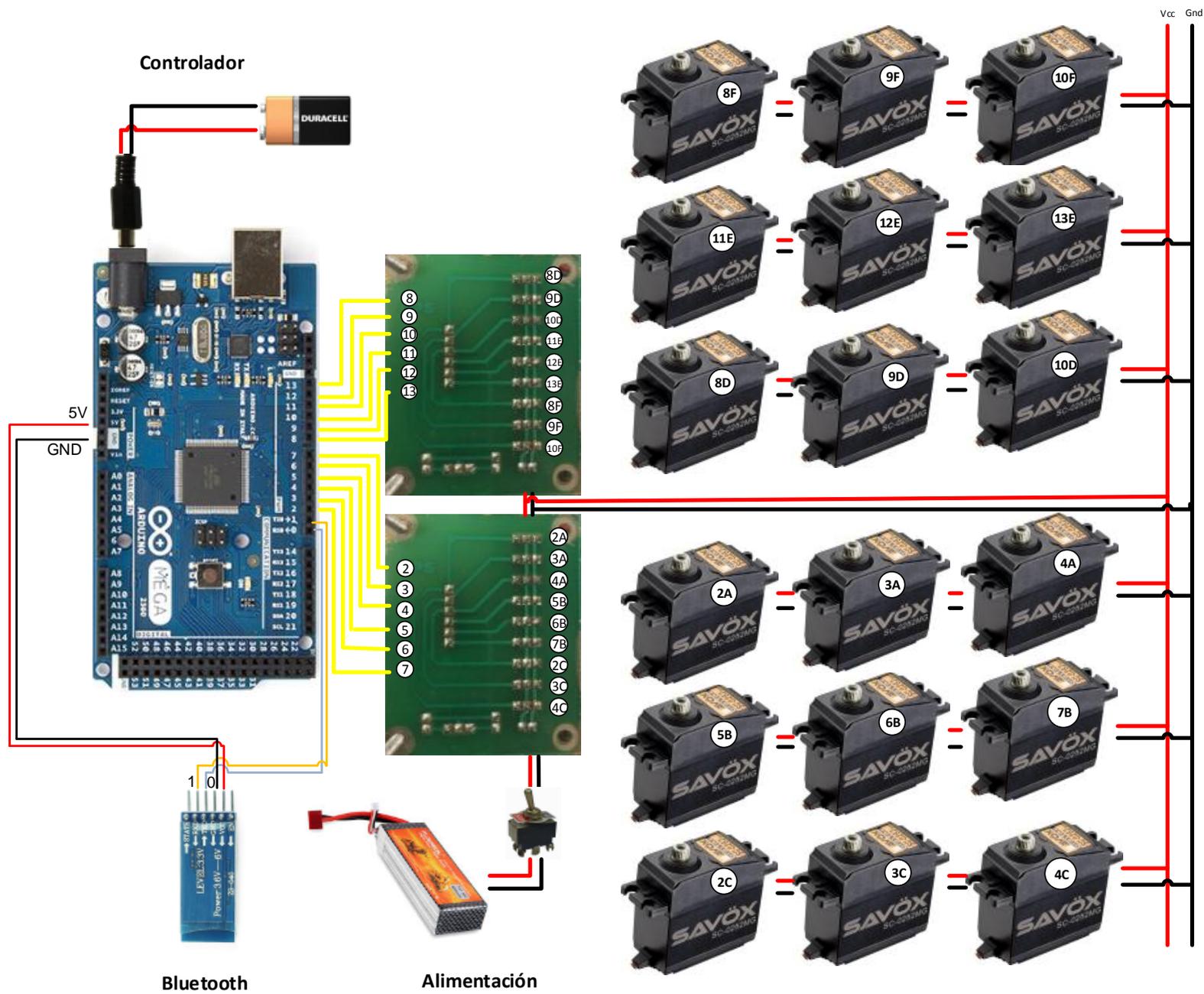


Diagrama de Conexiones para el Controlador Arduino Mega

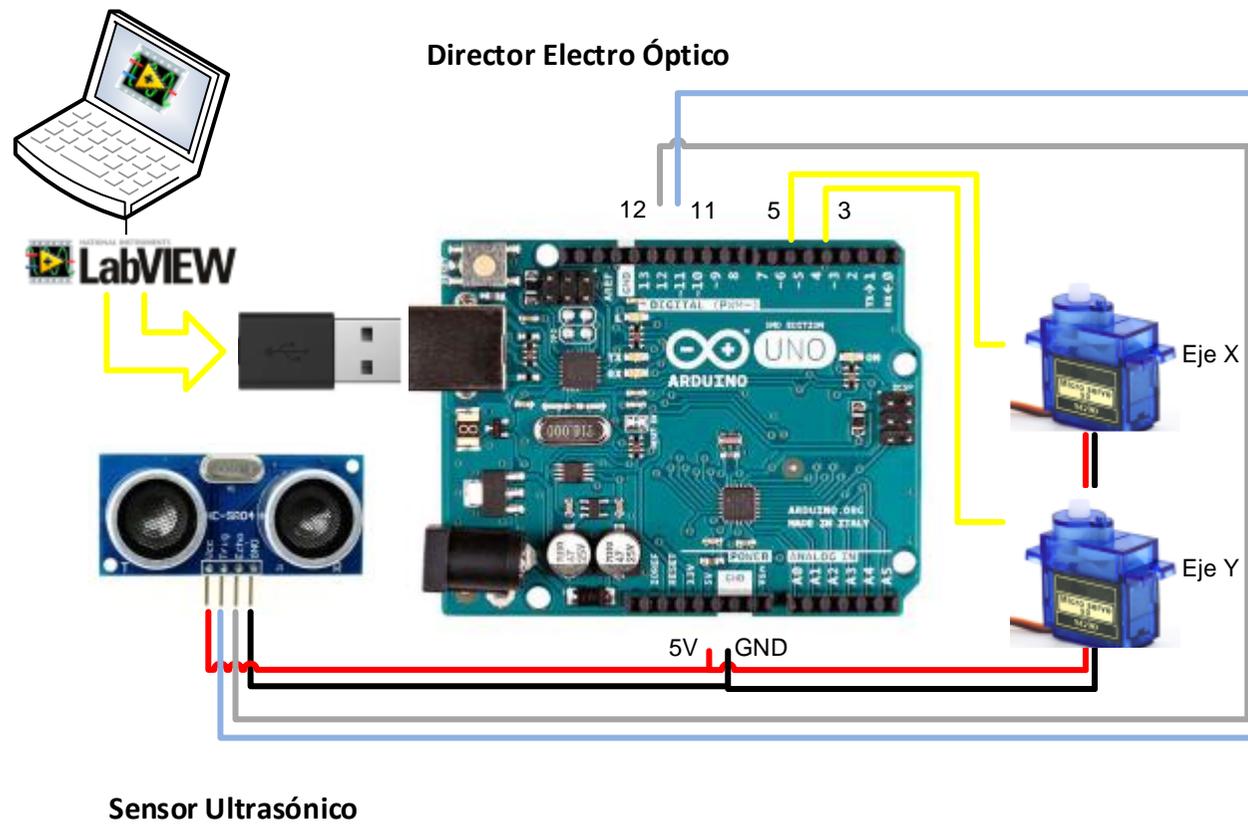


Diagrama de Conexiones para el Controlador Arduino UNO

# **ANEXO 3**

**PROGRAMACIÓN PARA  
SIMULACIÓN DEL ROBOT EN  
MATLAB**

```
%Locomoción de Robot Hexápodo en Matlab
% Se establece las dimensiones de los enlaces de la extremidad, las
unidades son en metros.
```

```
L1 = 0.02; L3 =0.07;L4= 0.12
```

```
% crear los vínculos de la pierna sobre la base de parámetros DH
%          theta d   a alpha
links(1) = Link([ 0  L1  0 pi/2 ], 'standard');
links(2) = Link([ 0  0  L3  0 ], 'standard');
links(3) = Link([ 0  0  L4  0 ], 'standard');
```

```
% creamos un robot para representar una sola pierna
leg = SerialLink(links, 'name', 'leg', 'offset', [pi/2 0 -pi/2]);
```

```
% definir los parámetros clave de la trayectoria de la marcha , caminando en
la dirección x
```

```
xf = 8; xb = -xf; % límites de avance y retroceso para los pies en la tierra
y = 8; % distancia del pie del cuerpo a lo largo del eje y
zu = 2; zd = 8; % altura de pie cuando sube y baja.
% definir la trayectoria rectangular tomada por el pie
segments = [xf y zd; xb y zd; xb y zu; xf y zu] * 0.01;
```

```
% construir el modo de andar, los puntos son :
```

```
% 1 inicio del movimiento del pie
% 2 final del movimiento del pie
% 3 final de aumento de pie
% 4 pie elevado hacia adelante
%
```

```
% los segmentos de tiempo son:
```

```
% 1->2 3s
% 2->3 0.5s
% 3->4 1s
% 4->1 0.5ss
```

```
% Un total de 4s , de los cuales 3s es caminar y 1s se pone a cero . En la
muestra % 0.01s
```

```
% el tiempo esto es exactamente 400 pasos de largo
```

```
%
```

```
% Utilizamos un tiempo de aceleración finita para obtener un buen camino
%agradable, lo que significa
```

```
%que el pie nunca pasa realmente a través de cualquiera de estos puntos .
```

```
%Esto hace que la configuración del robot inicial plantear y velocidad difícil.
```

```
% creamos una trayectoria más larga cíclica : 1, 2, 3, 4, 1, 2, 3, 4.
```

```
%El primer segmento de 1 > 2 incluye la rampa inicial y la final 3-> 4 tiene la
%desaceleración.
```

```
%Sin embargo, el medio 2- > 3-> 4- > 1 es cíclico suave.
```

```

segments = [segments; segments];
tseg = [3 0.25 0.5 0.25]';
tseg = [1; tseg; tseg];
x = mstraj(segments, [], tseg, segments(1,:), 0.01, 0.1);

% ciclo
xcycle = x(100:500,:);
qcycle = leg.ikine( transl(xcycle), [], [1 1 1 0 0 0], 'pinv' );

%dimensiones del cuerpo rectangular, anchura y altura del robot, las piernas
están en alrededor del cuerpo.
W = 0.21; L = 0.3;

%Dibujamos la extremidad con las características colocadas.
plotopt = leg.plot({'noraise', 'nobase', 'noshadow', ...
    'nowrist', 'nojaxes'});
% plotopt = leg.plot({'noraise', 'norender', 'nobase', 'noshadow', ...
%   'nowrist', 'nojaxes', 'ortho'});
% crear 6 robots de las piernas. Cada uno es un clon de la pierna robot que
construimos anteriormente,
%tiene un nombre único, y una base para representar su posición en el
cuerpo del robot andante .
legs(1) = SerialLink(leg, 'name', 'leg1');
legs(2) = SerialLink(leg, 'name', 'leg2', 'base', transl(-L, 0, 0));
legs(3) = SerialLink(leg, 'name', 'leg3', 'base', transl(-L, -W, 0)*trotz(pi));
legs(4) = SerialLink(leg, 'name', 'leg4', 'base', transl(0, -W, 0)*trotz(pi));
legs(5) = SerialLink(leg, 'name', 'leg5', 'base', transl(-L/2, -W, 0)*trotz(pi));
legs(6) = SerialLink(leg, 'name', 'leg6', 'base', transl(-L/2, 0, 0));

% crear un eje de tamaño fijo para el robot, y fijar z positiva hacia abajo.
clf; axis([-0.4 0.1 -0.2 0.4 -0.20 0.05]); set(gca,'Zdir', 'reverse')
hold on
% dibujar el cuerpo del robot
patch([0 -L -L 0], [0 0 -W -W], [0 0 0 0], ...
    'FaceColor', 'b', 'FaceAlpha', 0.5)
% Declaramos cada eje del robot.
for i=1:6
    legs(i).plot(qcycle(1,:), plotopt);
end
hold off

% caminar
k = 1;
while 1
    legs(1).plot( gait(qcycle, k, 500, 0), plotopt);
    legs(2).plot( gait(qcycle, k, 700, 0), plotopt);

```

```
legs(3).plot( gait(qcycle, k, 700, 1), plotopt);  
legs(4).plot( gait(qcycle, k, 500, 1), plotopt);  
legs(5).plot( gait(qcycle, k, 500, 1), plotopt);  
legs(6).plot( gait(qcycle, k, 500, 1), plotopt);  
drawnow  
k = k+1;  
end
```

# **ANEXO 4**

**PROGRAMACIÓN DEL ROBOT  
HEXÁPODO EN ARDUINO**

```

#include <Servo.h>
//PRIMERAS PATAS lado IZQUIERDO
Servo Ao1;
Servo Ao2;
Servo Ao3;
Servo Bo1;
Servo Bo2;
Servo Bo3;
// PRIMERAS PATAS lado DERECHO
Servo Do1;
Servo Do2;
Servo Do3;

Servo Eo1;
Servo Eo2;
Servo Eo3;

const int pos=80;

char dataIn = 'S'; //caracter del telefono
char determinant; //Used in the check function, stores the character received from the
phone.
char det;

void setup()
{
  Serial.begin(9600);
  char det;
  pinMode(13,OUTPUT);
  //PRIMERAS PATAS lado IZQUIERDO
  Ao1.attach(2);
  Ao2.attach(3);
  Ao3.attach(4);

  Bo1.attach(5);
  Bo2.attach(6);
  Bo3.attach(7);
  // PRIMERAS PATAS lado DERECHO
  Do1.attach(8);
  Do2.attach(9);
  Do3.attach(10);
  Eo1.attach(11);
  Eo2.attach(12);
  Eo3.attach(13);
}
void loop()
{
  posinicial();
  delay (500);
  while(1)
  {
    det = check();
    while (det == 'F') //if incoming data is a F, move forward
    {
      paso1();
      // delay(400);
    }
  }
}

```

```

    det = check();
}
while (det == 'B') //if incoming data is a B, move back
{
    paso2();
    //delay(400);
    det = check();
}
while (det == 'L') //if incoming data is a L, move wheels left
{
    paso4();
    //delay(400);
    det = check();
}
while (det == 'R') //if incoming data is a R, move wheels right
{
    paso5();
    //delay(400);
    det = check();
}

while (det == 'I') //if incoming data is a I, turn right forward
{
    paso6();
    //delay(400);
    det = check();
}
while (det == 'J') //if incoming data is a J, turn right back
{
    det = check();
}
while (det == 'G') //if incoming data is a G, turn left forward
{
    det = check();
}
while (det == 'H') //if incoming data is a H, turn left back
{
    det = check();
}
while (det == 'S') //if incoming data is a S, stop
{
    posinicial();
    det = check();
}
while (det == 'U') //if incoming data is a U, turn ON front lights
{
    // tell servo to go to position in variable 'pos'
    //digitalWrite(pinfrontLights, HIGH);
    det = check();
}
while (det == 'u') //if incoming data is a u, turn OFF front lights
{
    //digitalWrite(pinfrontLights, LOW);
}

```

```

        det = check();
    }
    while (det == 'W') //if incoming data is a W, turn ON back lights
    {
        // digitalWrite(pinbackLights, HIGH);
        paso3();
        //delay(400);
        det = check();
    }
    while (det == 'w') //if incoming data is a w, turn OFF back lights
    {
        //digitalWrite(pinbackLights, LOW);
        det = check();
    }
}

/*while(1)
{
delay(1000);
paso1();
delay (1000);
retizq();
delay(1000);
paso2();
delay(1000);
retdere();
delay(1000);
retorno();
}*/

}

}

void paso1()
{
//PASO 1
delay(200);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(70); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(90);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(50); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(90);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(50); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
}

```

```

delay(200);
//return
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(40); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(50); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
//return
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
}
void paso2()//atras
{
//PASO 2
delay(200);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(70); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(30);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(90); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
}

```

```

delay(200);
Bo1.write(30);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(90); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
//return
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(60);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(60);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
//return
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
}
void paso3();//SUBE Y BAJA
{
//SUBE
delay(200);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(70);
Ao3.write(100);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(90);
Bo3.write(100);

```

```
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(60);
Do3.write(20);
```

```
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(70);
Eo3.write(60);
```

```
//BAJA
delay(500);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
```

```
//SUBE
delay(200);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(70);
Ao3.write(100);
```

```
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(90);
Bo3.write(100);
```

```
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(60);
Do3.write(20);
```

```
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(70);
Eo3.write(60);
```

```
//BAJA
delay(500);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
```

```
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
```

```
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
```

```

Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
}
void paso4()//IZQUIERDA
{
//PASO 4
delay(200);
Do1.write(70); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
delay(200);
Do1.write(50); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
Bo1.write(30);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
delay(200);
Do1.write(50); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
Bo1.write(30);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);

delay(200);
//return
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);

delay(200);

Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(60);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(40); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);

```

```

Ao1.write(60);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(50); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
//return
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);

}

void paso5();//DERECHA
{
//PASO 5
delay(200);

Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
//return
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(70); //PRIMERAS PATAS lado IZQUIERDO

```

```

Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(90); // PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(90); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(90); // PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(90); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
//return
Bo1.write(60); // PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay (200);
}
void posinicial()//PARADA
{
Ao1.write(80); // PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);

Bo1.write(60); // PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);

Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);

Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
}
void paso6()// AUTOMÁTICO
{
delay(200);
Bo1.write(60); // PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(70); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(90); // PATA DEL MEDIO derecho
Bo2.write(120);

```

```
Bo3.write(110);
Do1.write(50); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(90);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(50); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
//return
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(40); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(50); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);

delay(200);

//return
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
//PASO 2 ATRAS
delay(200);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(70); //PRIMERAS PATAS lado IZQUIERDO
```

```
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(30); // PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(90); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(30); // PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(90); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
//return
Bo1.write(60); // PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
```

```
delay(200);
Ao1.write(80); // PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(60); // PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(60); // PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
```

```
delay(200);
//return
Ao1.write(80); // PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
//PASO 5 DERECHA
```

```

delay(200);

Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
//return
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);

delay(200);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(70); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(90);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(90); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(90);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(90); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
//return
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);

```

```
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
```

```
//SUBE
  delay(200);
Ao1.write(80); // PRIMERAS PATAS lado DERECHO
Ao2.write(70);
Ao3.write(100);
```

```
Bo1.write(60); // PATA DEL MEDIO derecho
Bo2.write(90);
Bo3.write(100);
```

```
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(60);
Do3.write(20);
```

```
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(70);
Eo3.write(60);
```

```
//BAJA
  delay(500);
Ao1.write(80); // PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Bo1.write(60); // PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
```

```
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
  delay(200);
//segunda repeticion
```

```
Bo1.write(60); // PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(70); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
  delay(200);
Bo1.write(90); // PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(50); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
  delay(200);
Bo1.write(90); // PATA DEL MEDIO derecho
Bo2.write(100);
```

```
Bo3.write(110);
Do1.write(50); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
//return
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(40); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(50); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
//return
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
//PASO 2 ATRAS
delay(200);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(70); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(30);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(90); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
```

```
delay(200);
Bo1.write(30);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(90); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
//return
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(60);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(60);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
//return
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
//PASO 5 DERECHA
delay(200);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(100);
Ao3.write(110);
```

```
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(40);
Eo3.write(40);
delay(200);
Ao1.write(100);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(100); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
//return
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(70); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(90);// PATA DEL MEDIO derecho
Bo2.write(120);
Bo3.write(110);
Do1.write(90); //PRIMERAS PATAS lado IZQUIERDO
Do2.write(30);
Do3.write(10);
delay(200);
Bo1.write(90);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(90); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
delay(200);
//return
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
//SUBE
delay(200);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(70);
Ao3.write(100);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(90);
Bo3.write(100);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
```

```

Do2.write(60);
Do3.write(20);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(70);
Eo3.write(60);
//BAJA
delay(500);
Ao1.write(80);// PRIMERAS PATAS lado DERECHO
Ao2.write(80);
Ao3.write(110);
Bo1.write(60);// PATA DEL MEDIO derecho
Bo2.write(100);
Bo3.write(110);
Do1.write(70); // PRIMERAS PATAS lado IZQUIERDO
Do2.write(50);
Do3.write(10);
Eo1.write(70); // PATA DEL MEDIO izquierdo
Eo2.write(60);
Eo3.write(40);
delay(200);
}
int check()
{
  if (Serial.available() > 0) //Check for data on the serial lines.
  {
    dataIn = Serial.read(); //Get the character sent by the phone and store it in 'dataIn'.
    if (dataIn == 'F')
    {
      determinant = 'F';
    }
    else if (dataIn == 'B')
    {
      determinant = 'B';
    }
    else if (dataIn == 'L')
    {
      determinant = 'L';
    }
    else if (dataIn == 'R')
    {
      determinant = 'R';
    }
    else if (dataIn == 'I')
    {
      determinant = 'I';
    }
    else if (dataIn == 'J')
    {
      determinant = 'J';
    }
    else if (dataIn == 'G')
    {
      determinant = 'G';
    }
    else if (dataIn == 'H')
    {

```

```
    determinant = 'H';  
  }  
  else if (dataIn == 'S')  
  {  
    determinant = 'S';  
  }  
  else if (dataIn == 'U')  
  {  
    determinant = 'U';  
  }  
  else if (dataIn == 'u')  
  {  
    determinant = 'u';  
  }  
  else if (dataIn == 'W')  
  {  
    determinant = 'W';  
  }  
  else if (dataIn == 'w')  
  {  
    determinant = 'w';  
  }  
}  
  
}
```

# **ANEXO 5**

**PROGRAMACIÓN DEL SISTEMA  
DEO EN LABVIEW**

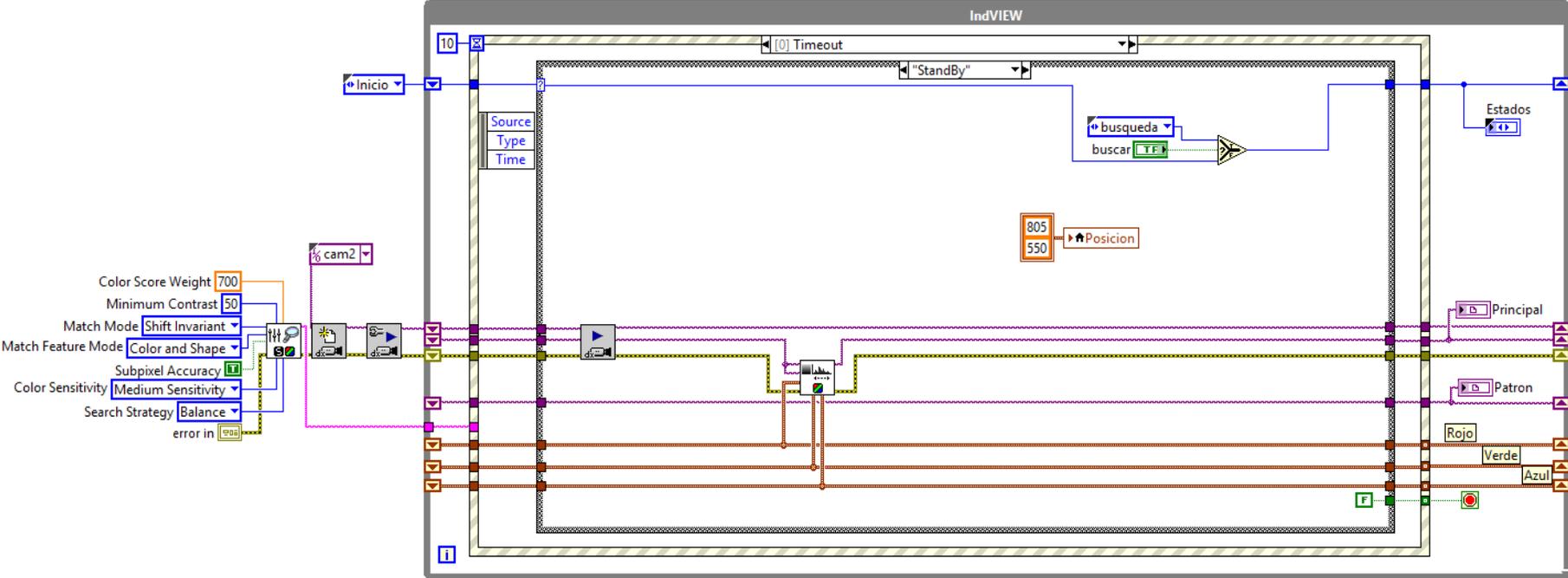
# Panel Frontal del Programa

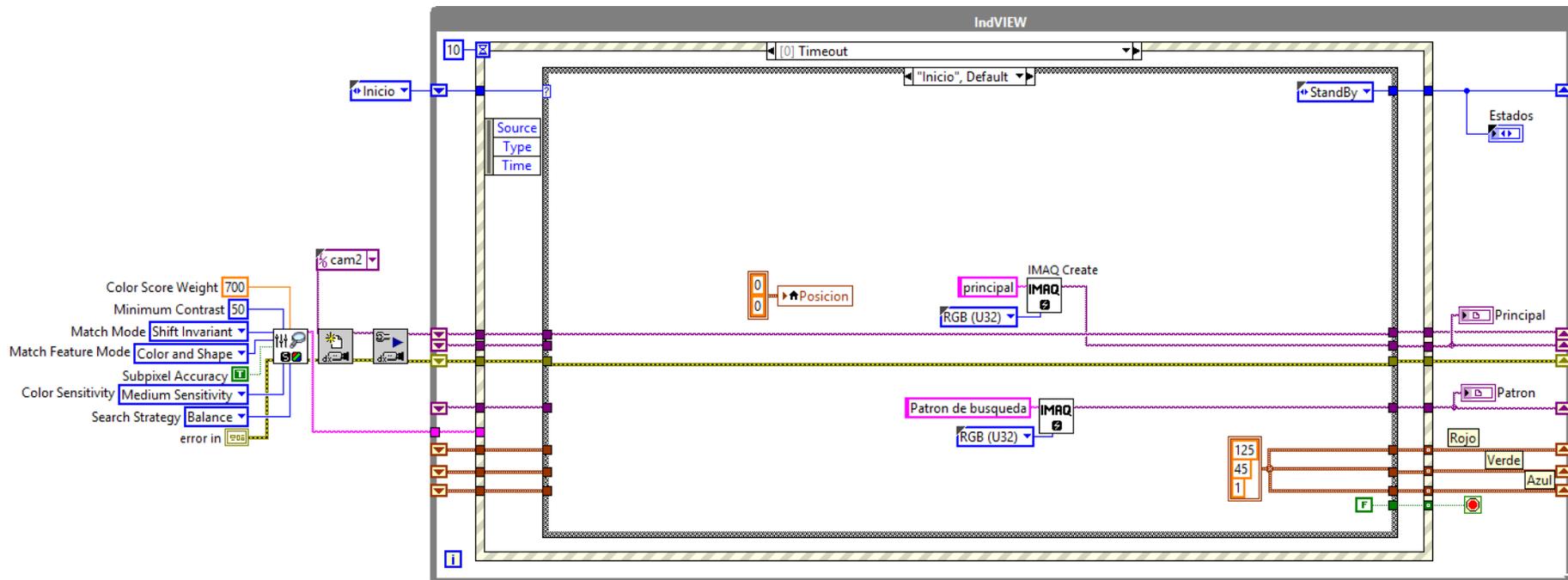
The interface is divided into several functional panels:

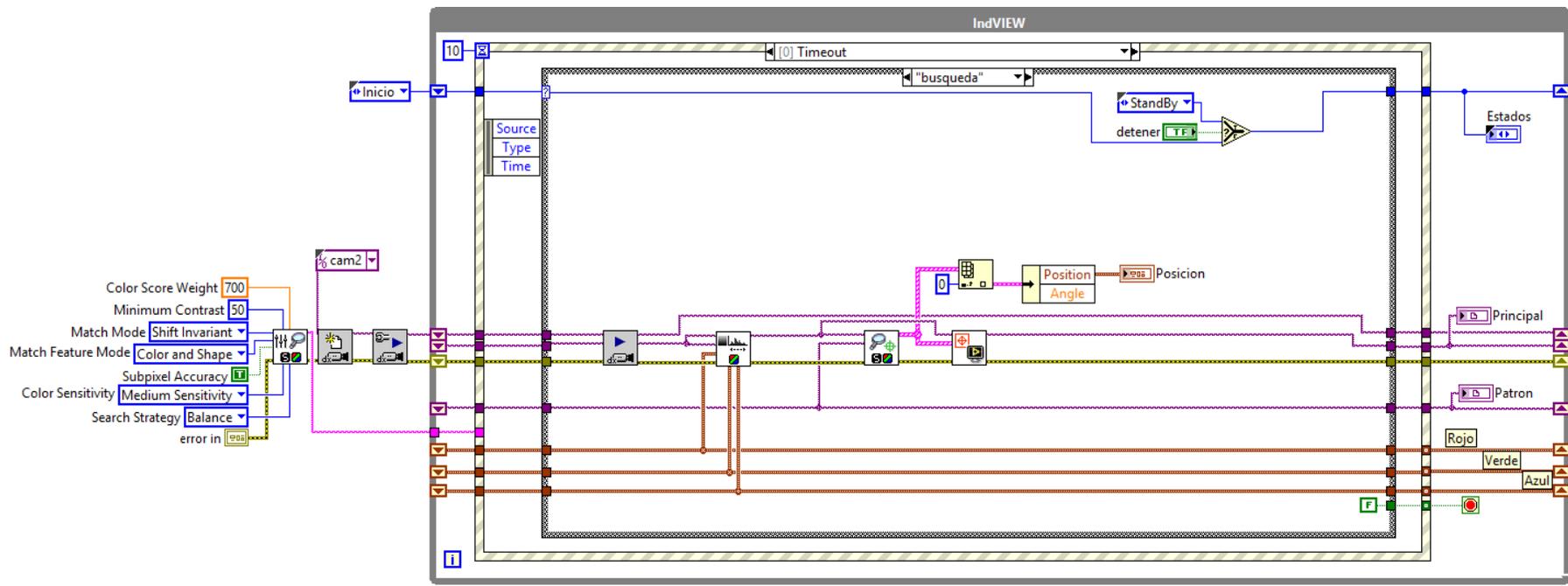
- IMAGEN PRINCIPAL:** A large central area for displaying the main image, with a toolbar on the left for zooming and navigation.
- ENTRENAMIENTO:** Contains buttons for 'APRENDER' (Learn), 'BUSCAR' (Search), and 'DETENER' (Stop).
- COMUNICACION ARDUINO:** Includes a dropdown menu (currently showing 'N/A'), a 'StandBy' field, and 'STOP' and 'SALIR' (Exit) buttons.
- OUT CONTROLLER:** Features two servo position gauges for 'Servo X' and 'Servo Y', and a 'Posicion' section with 'x' (550) and 'y' (805) input fields.
- INFORMACION OBJ.:** A calibration section with 'CALIBRACION' button, 'Cinta' (0 Px), 'Tamaño Cinta' (0 Cm), '1 Cm = Pixeles' (0), and 'DISTANCIA' (0) in both 'Centimetros' and 'Pulgadas'.
- CONFIGURACION PARAMETROS RGB:** Three columns for 'Rojo', 'Verde', and 'Azul', each with sliders for 'Brillo' (0-200), 'Contraste' (0-150), and 'Gamma' (0-10). Includes 'RESET RED', 'RESET GREEN', and 'RESET BLUE' buttons.
- CONTROL SERVO:** Controls for 'Máximo X Negativo', 'Máximo X Positivo', 'Máximo Y Negativo', and 'Máximo Y Positivo' with corresponding sliders and a 'Y' axis scale from 300 to 1000.

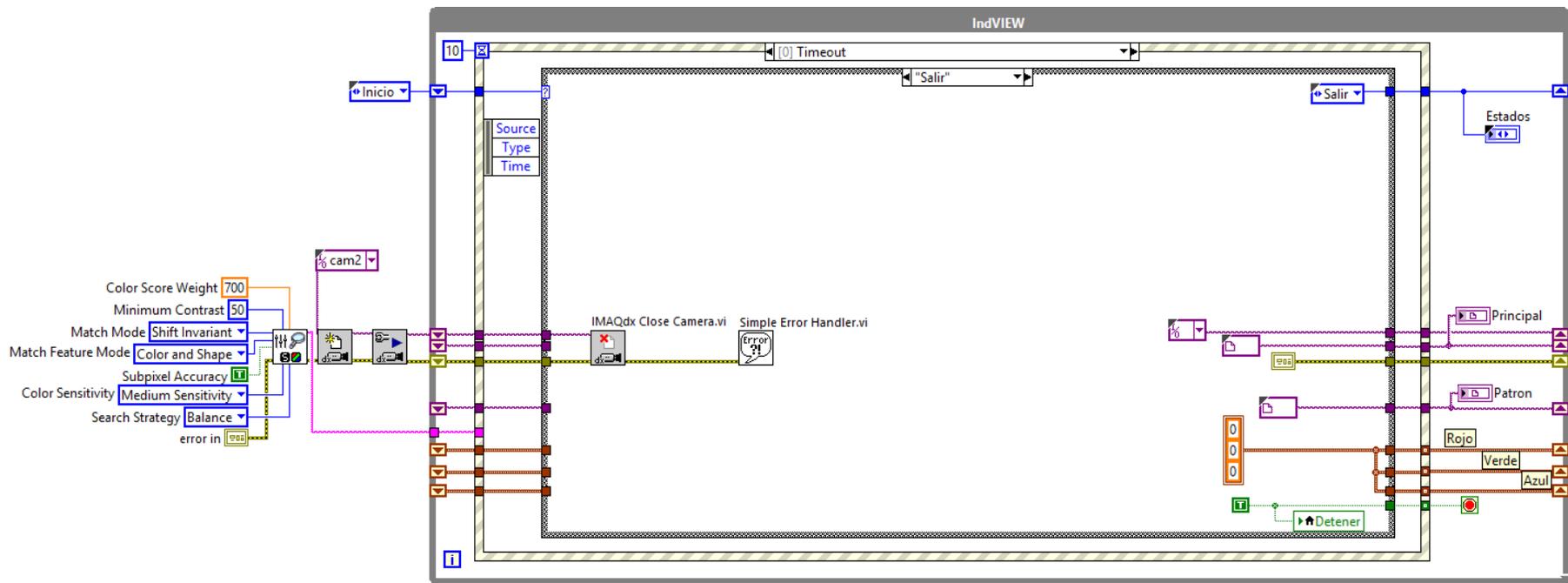
UNIVERSIDAD POLITÉCNICA SALESIANA

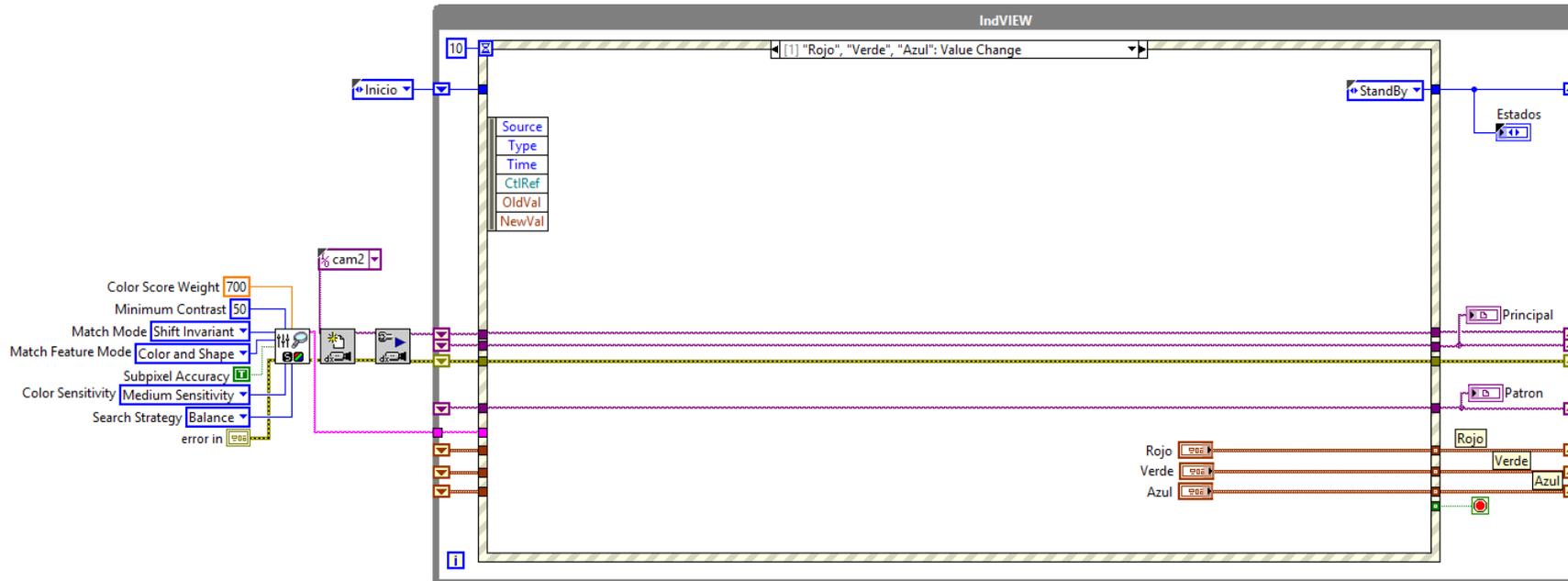
# Diagrama de Bloques

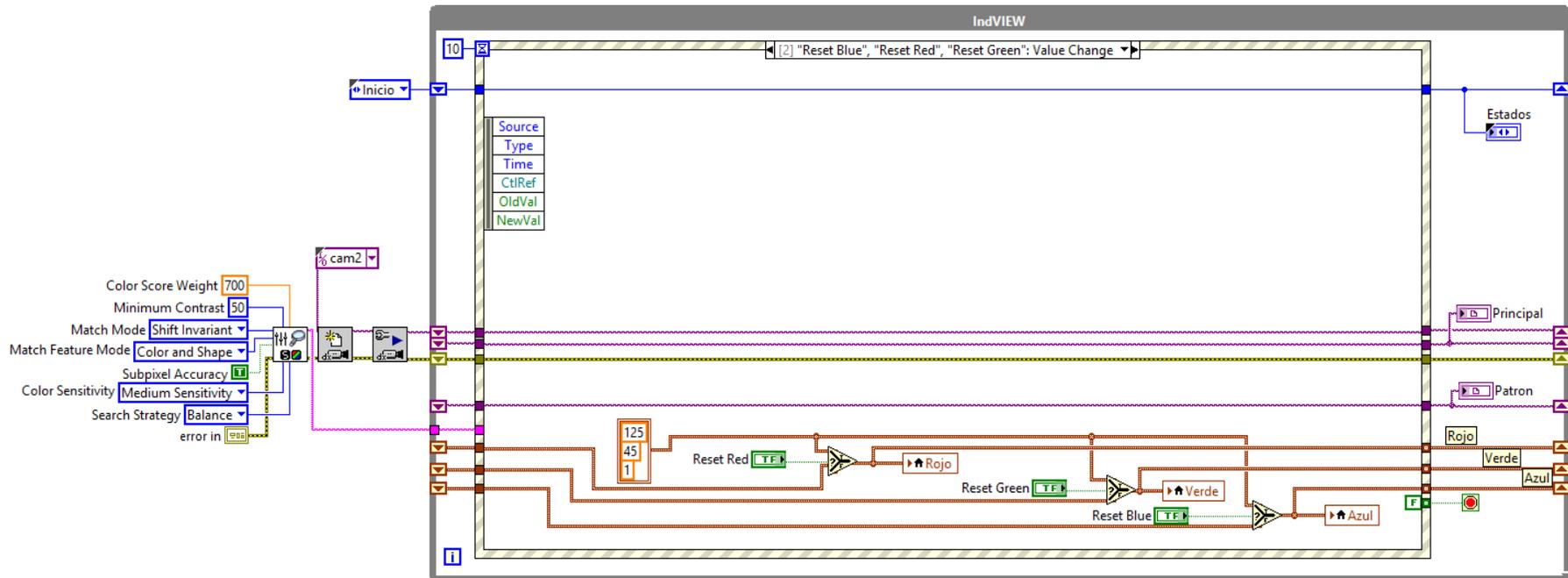




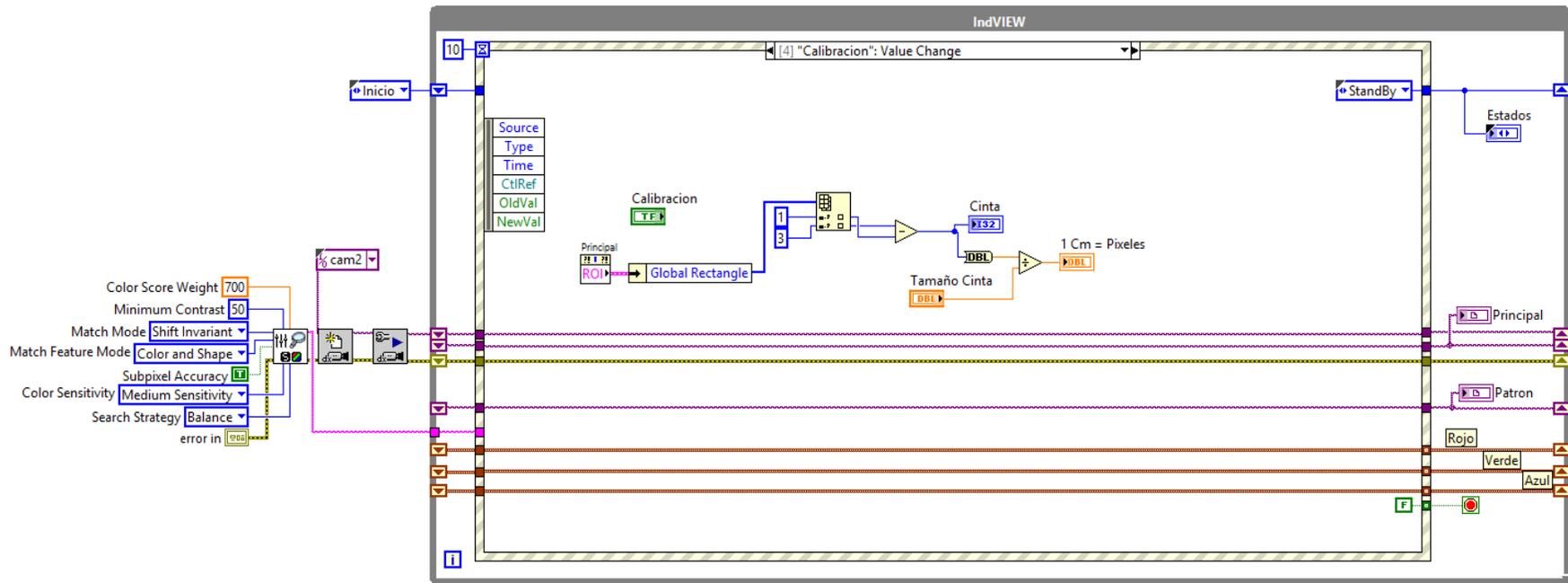


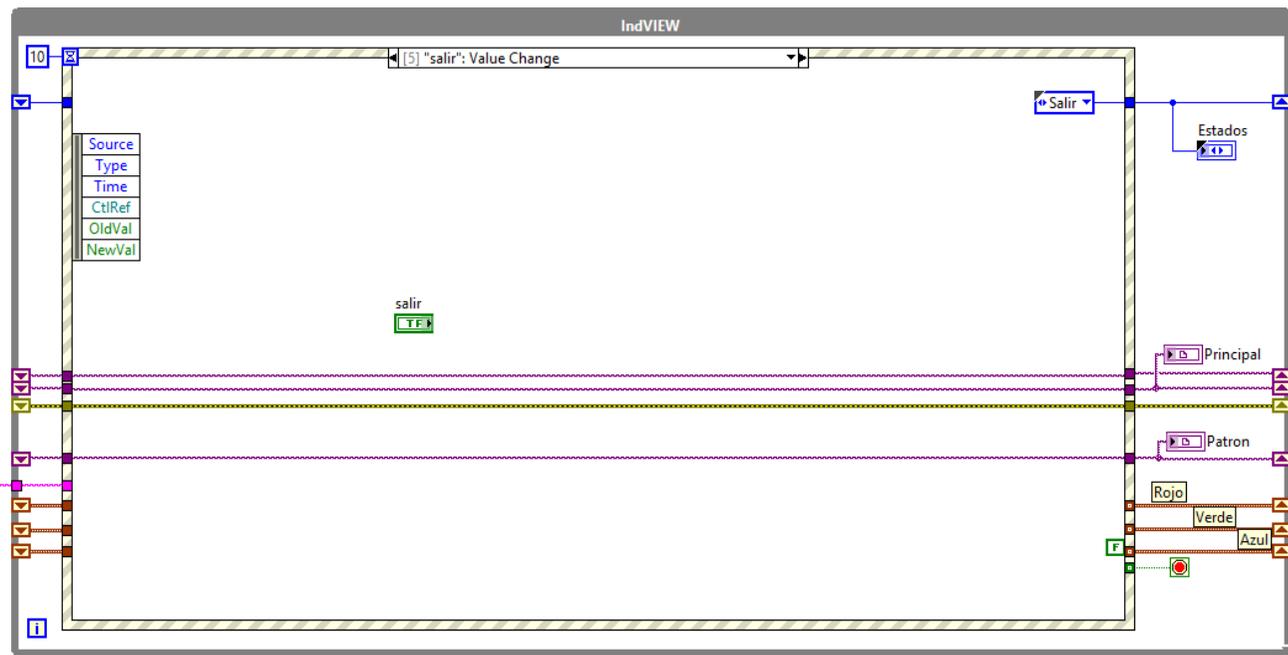
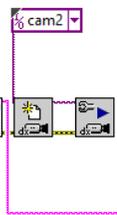
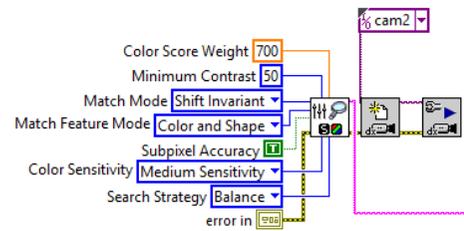


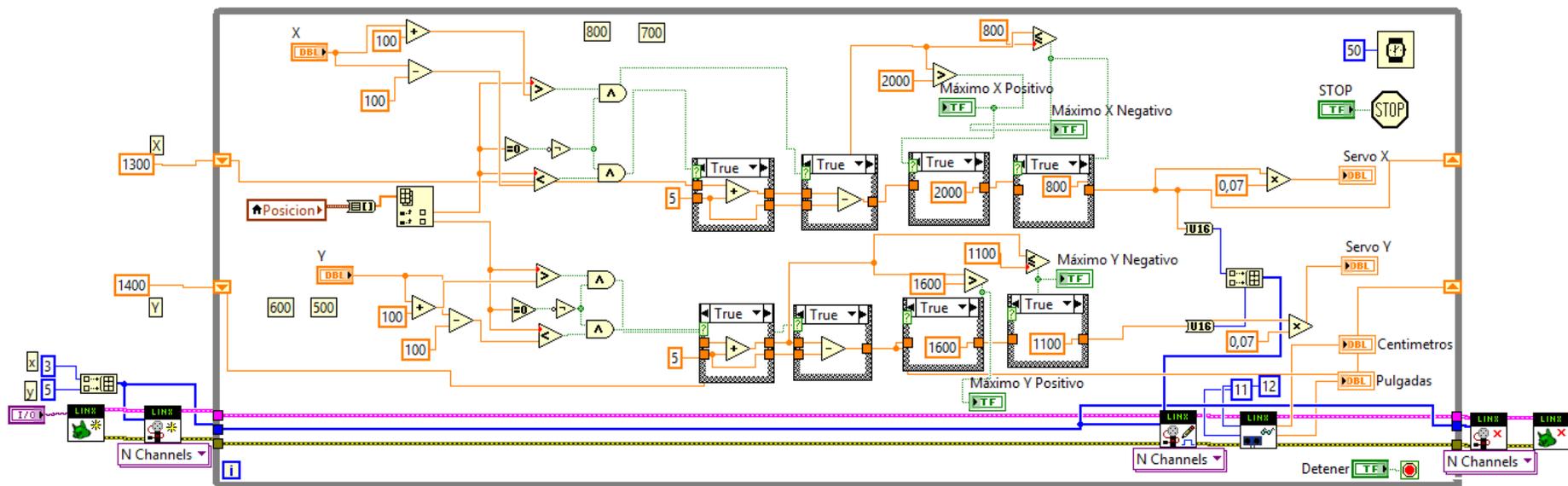












# PRÁCTICAS

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

## 1. DATOS INFORMATIVOS

a. MATERIA / CÁTEDRA RELACIONADA:

Electiva I / Robótica

b. No. DE PRÁCTICA: 1

c. NÚMERO DE ESTUDIANTES: 2

d. NOMBRE DOCENTE: Ing. Byron Lima.

e. TIEMPO ESTIMADO: 2 Horas

## 2. DATOS DE LA PRÁCTICA

a. TEMA:

**Análisis y Simulación de la locomoción del Robot Hexápodo mediante el uso de MATLAB aplicando la Cinemática del Robot y los parámetros Denavit-Hartenberg.**

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

**b. OBJETIVO GENERAL:**

- Diseñar un programa en Matlab para simular la locomoción de un Hexápodo.

**c. OBJETIVOS ESPECÍFICOS:**

- Familiarizar al alumno con el entorno del módulo Matlab.
- Diseñar y simular el movimiento de un hexápodo en Matlab.
- Aprender a razonar y a tomar decisiones ante los problemas que se le planteen, de tal forma que lo estimulen en la creatividad del alumno.
- Desarrollar el espíritu de equipo y de compañerismo.
- Integrar de una forma práctica la teoría cubierta en los cursos anteriores.

**d. MARCO TEÓRICO**

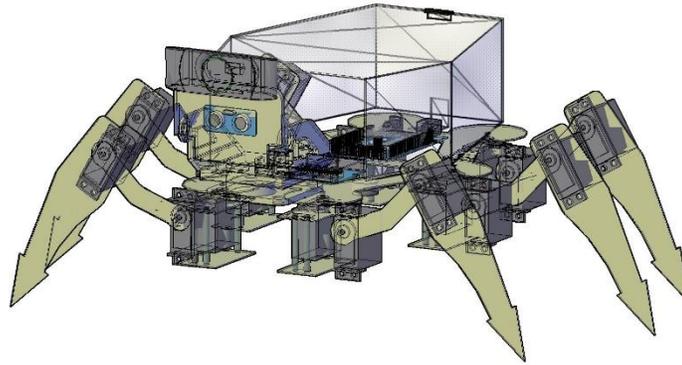
**1. Robot Hexápodo**

Dado que un robot puede ser estáticamente estable en tres o más patas, un robot hexápodo tiene una gran flexibilidad para moverse. Si las piernas se incapacitan, el robot puede aún ser capaz de caminar. Además, no se necesitan todas las piernas del robot para la estabilidad, las otras patas son libres de llegar a nuevas colocaciones de los pies o manipular una carga útil. Muchos robots hexápodos son biológicamente inspirados en la locomoción Hexápoda. Los hexápodos pueden ser utilizados para probar teorías biológicas sobre la locomoción de insectos, el control motor y la neurobiología. Los diseños hexápodos varían de acuerdo a la pierna.

Los insectos robots de inspiración suelen ser lateralmente simétricos, como el robot RISE en el Carnegie Mellon. Un hexápodo radialmente simétrico es el robot ATLETA (el todo terreno hexagonal-Legged Extra-Terrestre Explorer) en el JPL. Normalmente, las piernas tienen en cada rango, de dos a seis grados de libertad. Los pies del hexápodo suelen ser señalados, pero también pueden ser puntados con material adhesivo, para ayudar a escalar las paredes o las ruedas y el robot pueda conducirse rápidamente, cuando la tierra es plana. Los hexápodos son controlados por los aires, que permiten que el robot se mueva hacia adelante, a su vez y tal vez dar pasos al lado. Algunos de los aires más comunes son los siguientes: 1. Alternando trípode: 3 patas en el suelo a la vez. 2. Cuadrúpedo. 3. Rastreo: mover una sola pierna a la vez. (ARQHYS, 2017)

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



**Figura 1.**Robot Hexápodo

Para más información técnica vea la bibliografía.

**e. MARCO PROCEDIMENTAL**

Creación de la Comunicación

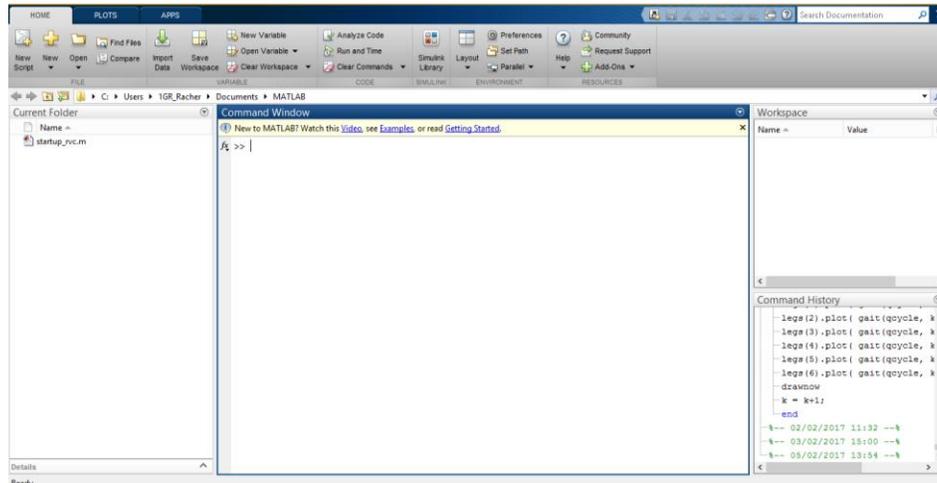
1. Tener Instalado Matlab
2. Instalar Driver de Matlab
3. Descargar y tener el Rvctools, que es el toolkit que se requiere.
4. Localizar la carpeta de nuestro toolkit rvctools y dar le run a starup\_rvc para que se inicie el tool y podamos empezar a elaborar la simulación del hexápodo.

Creación del Proyecto

1. Abrir el programa Matlab y se nos mostrara la ventana principal del programa.

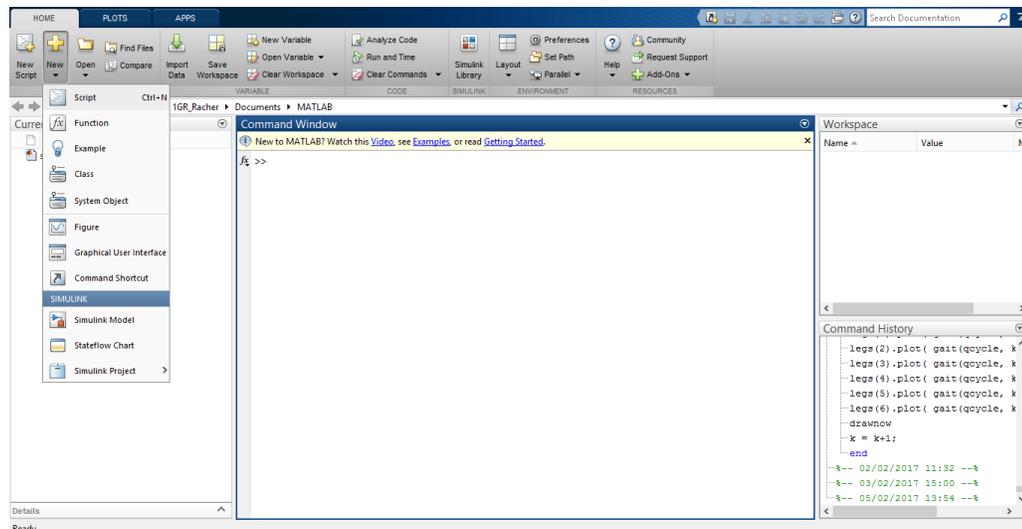
<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



**Figura 2.** Ventana Principal de Matlab

2. Escoger Home → New → Script, se nos abrirá una ventana de edición de programación.

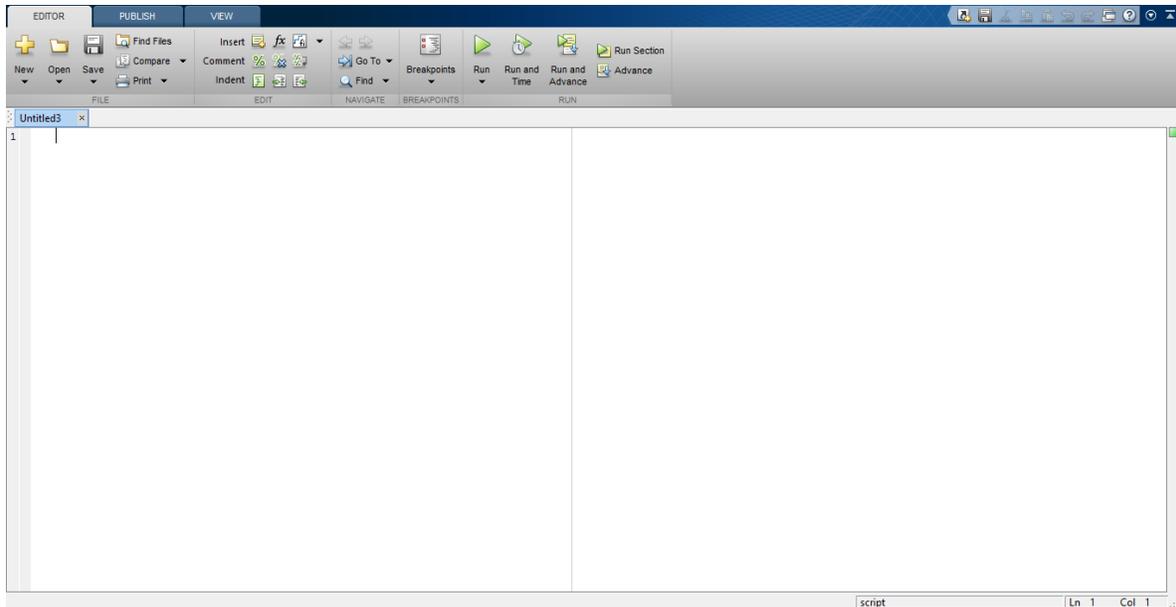


**Figura 3.** Creación de Nuevo Script

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

3. Se abrirá una ventana en donde podemos empezar nuestra programación.



**Figura 4.** Creación de Nuevo Script

### **Parámetros DH**

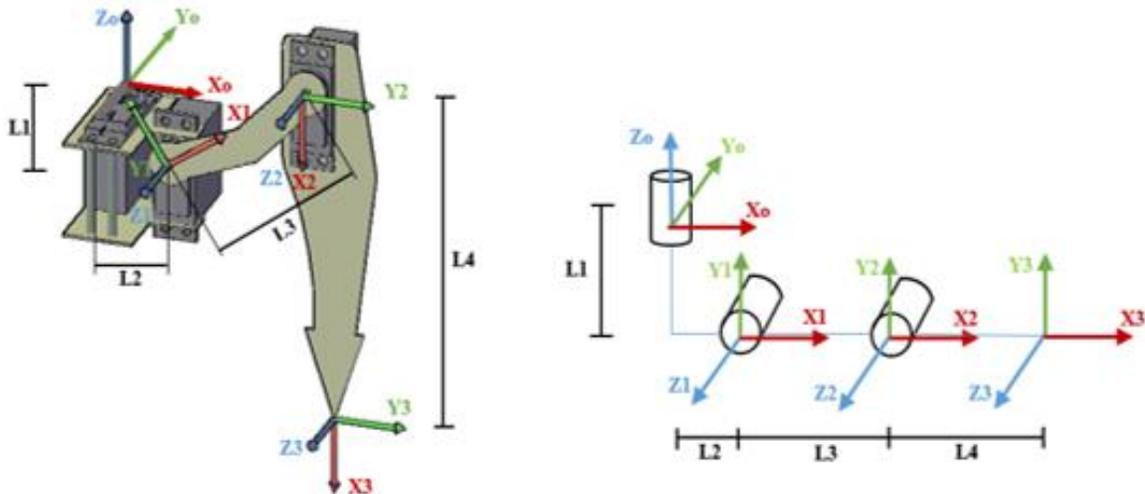
La extremidad del robot hexápodo se conforman por 3 articulaciones en donde se produce un movimiento angular, lo que se quiere obtener es la posición del extremo de la extremidad (P(X, Y, Z)) en función de los ángulos en cada Articulación (Cinemática Directa).

$$(X, Y, Z) = F(\theta_1, \theta_2, \theta_3)$$

Para el análisis Cinemático se encuentran los parámetros Denavit Hartenberg de una extremidad a partir de los parámetros que la conforman.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



**Figura 5.**Asignación de ejes en una Extremidad

**Tabla de Parámetros Denavit Hartenberg**

	<b>e</b>	<b>d</b>	<b>a</b>	$\infty$
<b>1</b>	$e_1$	L1	L2	$\pi/2$
<b>2</b>	$e_2$	0	L3	0
<b>3</b>	$e_3$	0	L4	0

**Figura 6.**Tabla de Parámetros DH

A partir de los parámetros obtenidos se ingresa los parámetros DH en Matlab para luego realizar la simulación del robot.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

### Edición del Programa

La librería rvcTools es la que se utiliza para trabajar con programación enfocada a la robótica. La programación está basado en la creación de eslabones mediante los parámetros Denavit Hartenberg.

Se crea las dimensiones en una extremidad de un robot, como se mencionó anteriormente en cada extremidad tenemos 3 grados de libertad.

```

4 %Locomoción de Robot Hexápodo en Matlab
5 % Se establece las dimensiones de los enlaces de la extremidad, las unidades son en metros.
6 - L1 = 0.02;L2=0.02; L3 =0.07;L4= 0.12
7
8 % crear los vínculos de la pierna sobre la base de parámetros DH
9 % theta d a alpha
10 - links(1) = Link([ 0 L1 L2 pi/2 ], 'standard');
11 - links(2) = Link([ 0 0 L3 0 ], 'standard');
12 - links(3) = Link([ 0 0 L4 0 ], 'standard');
13
14 % creamos un robot para representar una sola pierna
15 - leg = SerialLink(links, 'name', 'leg', 'offset', [pi/2 0 -pi/2]);
16

```

**Figura 7.**Programación de extremidad de robot en Matlab.

Definimos parámetros de trayectoria para la pierna creada tales como los límites de avance y retroceso para la pata en la tierra, distancia de pie del cuerpo a lo largo del eje y, altura de pie cuando sube y baja.

```

17 % definir los parámetros clave de la trayectoria de la marcha , caminando en la dirección x
18 - xf = 10; xb = -xf; % límites de avance y retroceso para los pies en la tierra
19 - y = 10; % distancia del pie del cuerpo a lo largo del eje y
20 - zu = 2; zd = 10; % altura de pie cuando sube y baja.
21 % definir la trayectoria rectangular tomada por el pie
22 - segments = [xf y zd; xb y zd; xb y zu; xf y zu] * 0.01;
23

```

**Figura 8.**Trayectoria de extremidad en Matlab.

A la secuencia que se crea le colocamos los tiempos en los que dura cada segmento, para que se siga un ciclo reitrativo.Luego colocamos las dimensiones de la anchura y la altura del robot para crear el cuerpo rectangular.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

```

37 %
38 % Utilizamos un tiempo de aceleración finita para obtener un buen camino %agradable, lo que significa
39 %que el pie nunca pasa realmente a través de cualquiera de estos puntos .
40 %Esto hace que la configuración del robot inicial plantear y velocidad difícil.
41 % creamos una trayectoria más larga ciclica :. 1, 2, 3, 4, 1, 2, 3, 4.
42 %El primer segmento de 1 > 2 incluye la rampa inicial y la final 3-> 4 tiene la %desaceleración.
43 %Sin embargo, el medio 2-> 3-> 4-> 1 es ciclico suave.
44
45 - segments = [segments; segments];
46 - tseg = [3 0.25 0.5 0.25]';
47 - tseg = [1; tseg; tseg];
48 - x = mstraj(segments, [], tseg, segments(1,:), 0.01, 0.1);
49
50 % ciclo
51 - xcycle = x(100:500,:);
52 - qcycle = leg.ikine( transl(xcycle), [], [1 1 1 0 0 0], 'pinv');
53
54 %dimensiones del cuerpo rectangular, anchura y altura del robot, las piernas están en alrededor del cuerpo.
55 - W = 0.21; L = 0.3;

```

**Figura 9.** Tiempos en cada segmento del robot en Matlab.

Luego creamos las 6 extremidades totales del robot, Cada uno es un clon de la extremidad que construimos anteriormente, tiene un nombre único , y una base para representar su posición en el cuerpo del robot andante .

```

62 - legs(1) = SerialLink(leg, 'name', 'leg1');
63 - legs(2) = SerialLink(leg, 'name', 'leg2', 'base', transl(-L, 0, 0));
64 - legs(3) = SerialLink(leg, 'name', 'leg3', 'base', transl(-L, -W, 0)*trotz(pi));
65 - legs(4) = SerialLink(leg, 'name', 'leg4', 'base', transl(0, -W, 0)*trotz(pi));
66 - legs(5) = SerialLink(leg, 'name', 'leg5', 'base', transl(-L/2, -W, 0)*trotz(pi));
67 - legs(6) = SerialLink(leg, 'name', 'leg6', 'base', transl(-L/2, 0, 0));
..

```

**Figura 10.** Extremidades totales del Robot en Matlab.

Se crea gráficamente las extremidades y el cuerpo del Robot Hexápodo considerando la instancia de cada eje de las extremidades en el cuerpo caminante.

```

72 % dibujar el cuerpo del robot
73 - patch([0 -L -L 0], [0 0 -W -W], [0 0 0 0], ...
74         'FaceColor', 'b', 'FaceAlpha', 0.5)
75 % Declaramos cada eje del robot.
76 - for i=1:6
77 -     legs(i).plot(qcycle(1,:), plotopt);
78 - end
79 - hold off
80
81 % caminar
82 - k = 1;
83 - while 1
84 -     legs(1).plot( gait(qcycle, k, 500, 0), plotopt);
85 -     legs(2).plot( gait(qcycle, k, 700, 0), plotopt);
86 -     legs(3).plot( gait(qcycle, k, 700, 1), plotopt);
87 -     legs(4).plot( gait(qcycle, k, 500, 1), plotopt);
88 -     legs(5).plot( gait(qcycle, k, 500, 1), plotopt);
89 -     legs(6).plot( gait(qcycle, k, 500, 1), plotopt);
90 -     drawnow
91 -     k = k+1;
92 - end

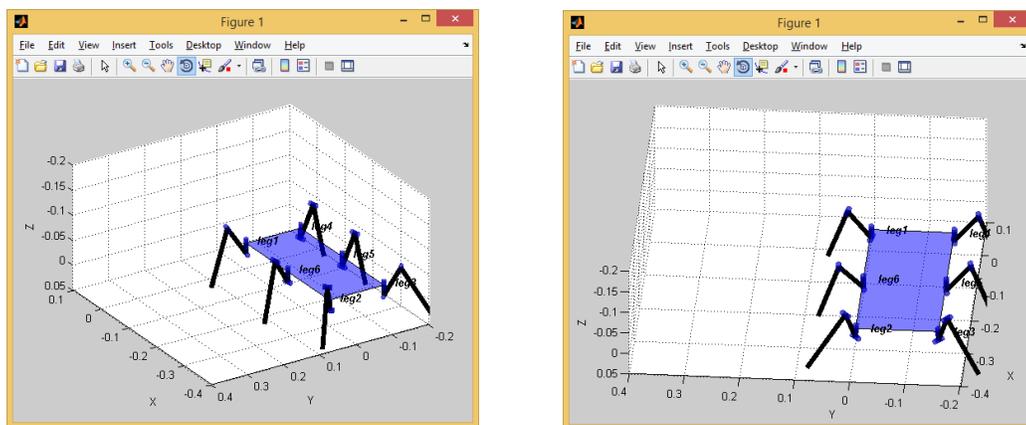
```

**Figura 11.** Graficas de Extremidades y Cuerpo del Robot en Matlab.

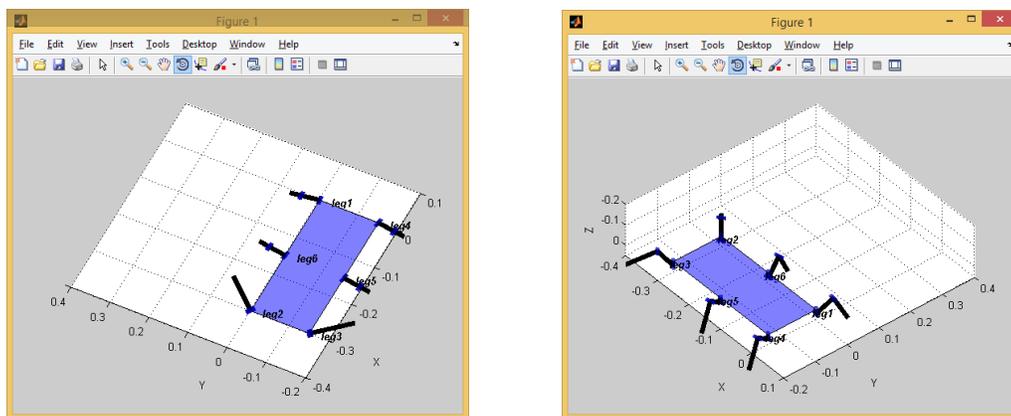
<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución</b> <b>Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

Para poder compilar la programación se agrega la librería rvctools y luego se compila el código, se podrá observar cómo nos muestra las matrices de los parámetros DH y nos grafica en movimiento el Robot Hexápodo.



**Figura 12.** Simulación de Robot Hexápodo en Matlab (1)



**Figura 13.** Simulación de Robot Hexápodo en Matlab (2)

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

### Questionario

**1) ¿Cómo se llama la plataforma y el toolk con el cual realiza la simulación de locomoción del robot Hexápodo?**

La plataforma es Matlab y el toolkit es rvctools.

**2) ¿Cuántos links ingresa en la matriz de parámetros DH y cuáles son las variables DH?**

Se ingresa 3 Links en la matriz DH y las variables son theta, a, d y alpha.

**3) ¿Cuántos grados de libertad tiene en total el robot Hexápodo?**

El robot hexápodo tiene en total 18 grados de libertad.

**4) ¿Se puede crear más piernas en el robot si se lo requiere?**

Si es posible, se tendría que dibujar las piernas adicionales en el cuerpo del robot y asignarle un movimiento.

**f. RECURSOS UTILIZADOS (EQUIPOS, ACCESORIOS Y MATERIAL CONSUMIBLE)**

**Materiales necesarios:**

- Teoría parámetros DH

**Instrumentos:**

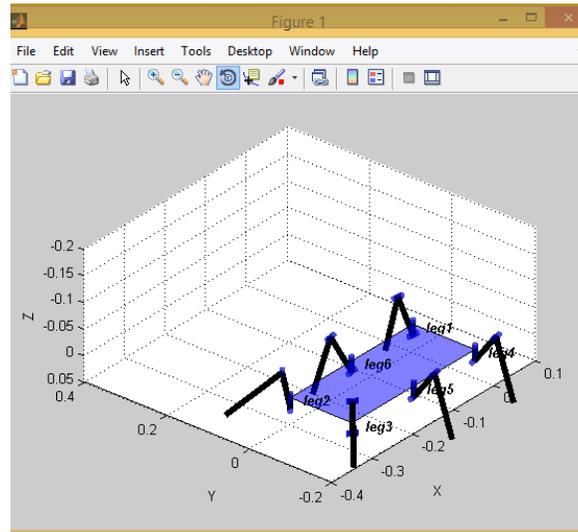
- Computador en el cual este instalado Matlab y toolkits necesarios.

**g. REGISTRO DE RESULTADOS**

Resultado de la simulación del robot.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>ANÁLISIS Y SIMULACIÓN DE LA LOCOMOCIÓN DEL ROBOT HEXÁPODO MEDIANTE EL USO DE MATLAB APLICANDO LA CINEMÁTICA DEL ROBOT Y LOS PARÁMETROS DENAVIT-HARTENBERG.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



**Figura 14.** Simulación del Robot Hexápodo en Matlab.

### **Conclusiones:**

Mediante los ejes de coordenadas correspondientes a cada Articulación de una extremidad, se pudo obtener la tabla con los parámetros DH. Estos parámetros fueron necesarios para ingresarlos a la programación de Matlab y su vez poder obtener la simulación de la locomoción del robot Hexápodo.

### **Recomendaciones**

Es importante ser cuidadoso al momento de trazar los ejes en las coordenadas de las articulaciones, ya que se tiene obtener correctamente los parámetros DH y así poder ingresarlos en la programación de Matlab.

## **h. BIBLIOGRAFÍA UTILIZADA**

ARQHYS. (05 de 02 de 2017). Revista ARQHYS.com.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

## 1. DATOS INFORMATIVOS

a. MATERIA / CÁTEDRA RELACIONADA:

Electiva I / Robótica

b. No. DE PRÁCTICA: 2

c. NÚMERO DE ESTUDIANTES: 2

d. NOMBRE DOCENTE: Ing. Byron Lima.

e. TIEMPO ESTIMADO: 2 Horas

## 2. DATOS DE LA PRÁCTICA

a. TEMA:

**Movimiento de Servomotores utilizando el Controlador ARDUINO y LABVIEW.**

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

**b. OBJETIVO GENERAL:**

- Diseñar un programa en Labview para controlar el movimiento de un servomotor mediante la comunicación de la tarjeta Arduino y el Labview.

**c. OBJETIVOS ESPECÍFICOS:**

- Familiarizar al alumno con el entorno de los módulos de Arduino y Labview.
- Diseñar y simular el movimiento de un servomotor mediante labview.
- Establecer y comprobar comunicación entre el software Labview y el controlador Arduino.
- Aprender a razonar y a tomar decisiones ante los problemas que se le planteen, de tal forma que lo estimulen en la creatividad del alumno.
- Desarrollar el espíritu de equipo y de compañerismo.
- Integrar de una forma práctica la teoría cubierta en los cursos anteriores.

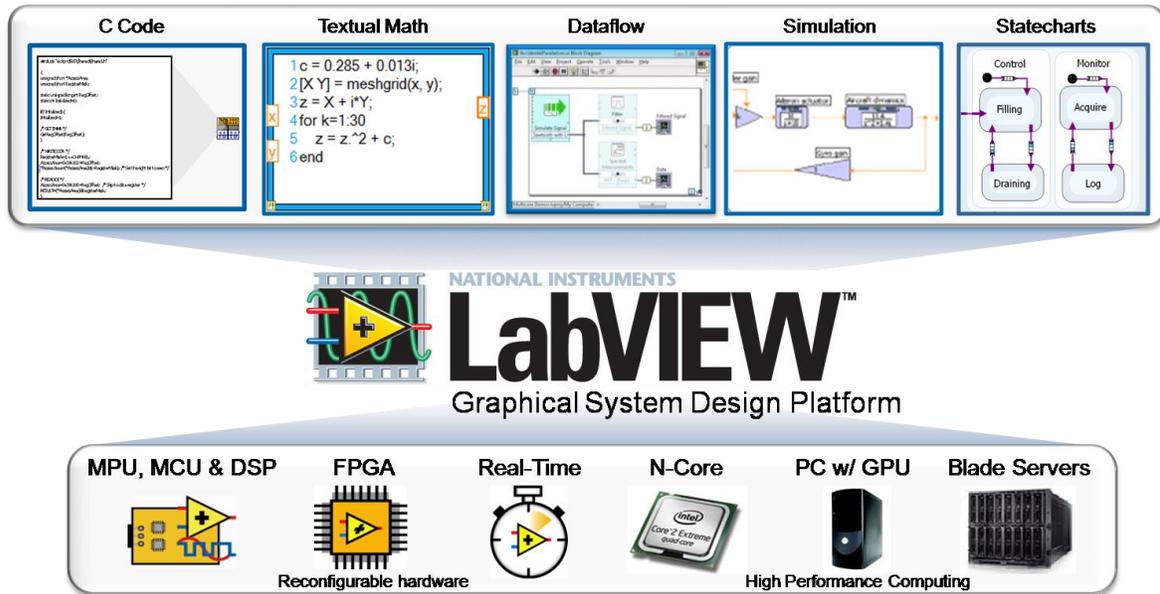
**d. MARCO TEÓRICO**

**LABVIEW**

LabVIEW es su herramienta para resolver más rápido y de manera más eficiente los problemas de hoy en día con la habilidad de evolucionar y resolver con sus retos futuros. LabVIEW ofrece integración sin precedentes con todo el hardware de medidas, software legado existente e IP al aprovechar las últimas tecnologías de cómputo. LabVIEW (acrónimo de Laboratory Virtual Instrumentation Engineering Workbench) es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico. Recomendado para sistemas hardware y software de pruebas, control y diseño, simulado o real y embebido, pues acelera la productividad. El lenguaje que usa se llama lenguaje G, donde la G simboliza que es lenguaje Gráfico. Este programa fue creado por National Instruments (1976) para funcionar sobre máquinas MAC, salió al mercado por primera vez en 1986. Ahora está disponible para las plataformas Windows, UNIX, MAC y GNU/Linux. La penúltima versión es la 2013, con la increíble demostración de poderse usar simultáneamente para el diseño del firmware de un instrumento RF de última generación, a la programación de alto nivel del mismo instrumento, todo ello con código abierto. Y posteriormente la versión 2014 disponible en versión demo para estudiantes y profesional, la versión demo se puede descargar directamente de la página National Instruments. (Instrument, 2017)

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



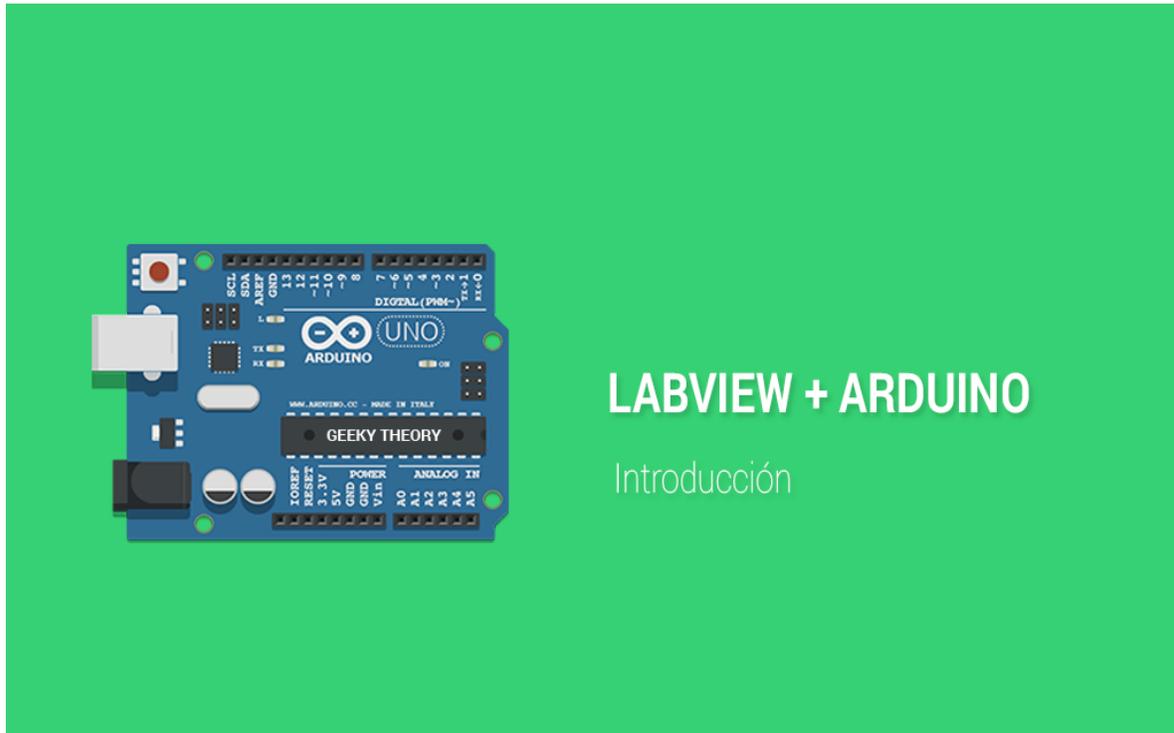
**Figura 1.** Plataforma Labview

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinarios.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida.<sup>4</sup> Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, y Atmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque que es ejecutado en la placa. Se programa en el ordenador para que la placa controle los componentes electrónicos.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
 <b>UNIVERSIDAD POLITÉCNICA SALESIANA</b> ECUADOR		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



**Figura 2.** Comunicación de Labview y Arduino

Para más información técnica vea la bibliografía.

**e. MARCO PROCEDIMENTAL**

Creación de la Comunicación

5. Tener Instalado Labview
6. Instalar Driver de Labview
7. Instalar el Labview Package Manager, por medio de este gestor se instala el VIPM Arduino toolkit.
8. Descargar el Arduino Toolkit del VIPM
9. Instalar el IDE Arduino
10. Verificar el reconocimiento del puerto USB parte del IDE Arduino, por medio del administrador de dispositivos de Windows.
11. Para que Labview pueda comunicarse con Arduino se enviara a la memoria de Arduino el programa LIFA Base, el programa se encuentra en la carpeta " **C:\Program Files (x86)\National Instruments\LabVIEW**

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing.Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

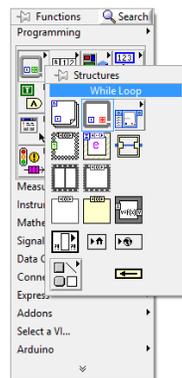
**2011\vi.lib\LabVIEW Interface for Arduino\Firmware\LIFA\_Base** “ luego que se compila el programa a la tarjeta este ya se encarga de comunicarse con el Arduino.

### Creación del Proyecto

4. Abrir el programa Labview
5. Escoger File→ New VI
6. Ir a la opción Windows, elegir Tile left and Right

### Edición del Programa

Dar clic derecho en el Diagrama de bloques y buscar las estructuras, elegiremos la estructura While Loop



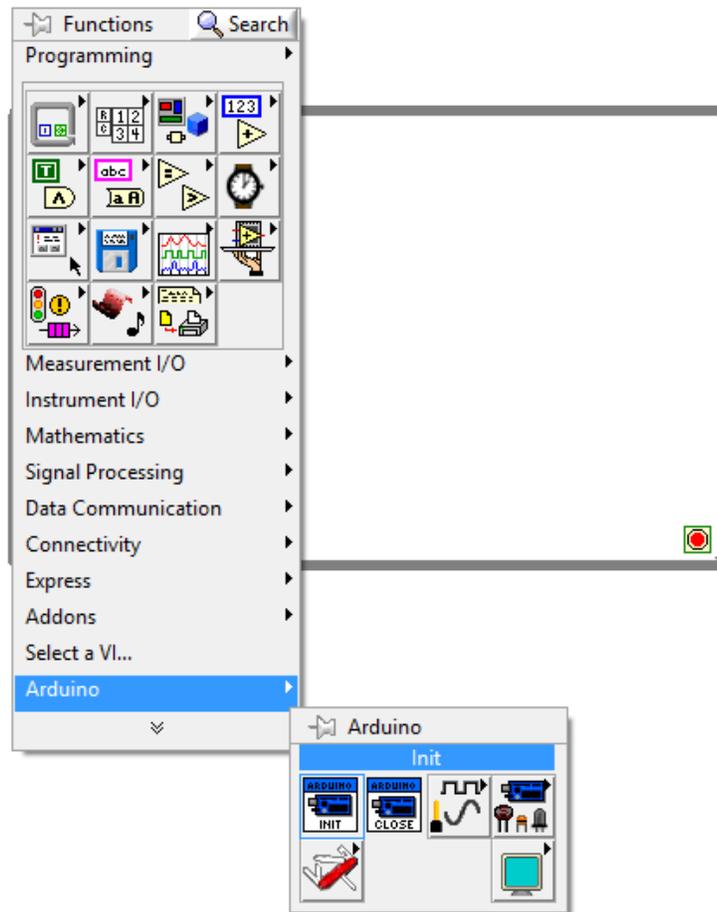
**Figura 3.** Selección de la estructura While Loop

7. Una vez colocada la estructura while Loop comenzaremos a colocar los bloques de Arduino dentro de la estructura. Damos clic derecho nuevamente dentro de la ventana de diagrama de bloques buscamos la pestaña de Arduino, ahí encontraremos todas las herramientas de Arduino.

8. Elegimos el bloque de Arduino INIT  , que es para iniciar el Arduino.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing.Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

<b>REVISION 1/1</b>		<i>Página 1 de -</i>
		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible	
<b>CARRERA</b>	Ingeniería Electrónica	
<b>SEDE</b>	Guayaquil	

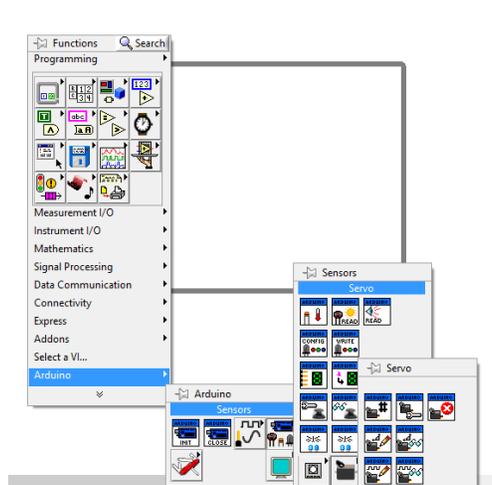


**Figura 4.** Selector de Objetos, Bloques de Arduino, INIT entrada de Arduino

9. Seleccionar otro bloque de Arduino y elegir el bloque CLOSE  , colocarlo en mi ventana de diagrama de bloques.
10. Buscamos dentro de los bloques de Arduino la opción Sensors, buscamos dentro los bloques que dicen Servo.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing.Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



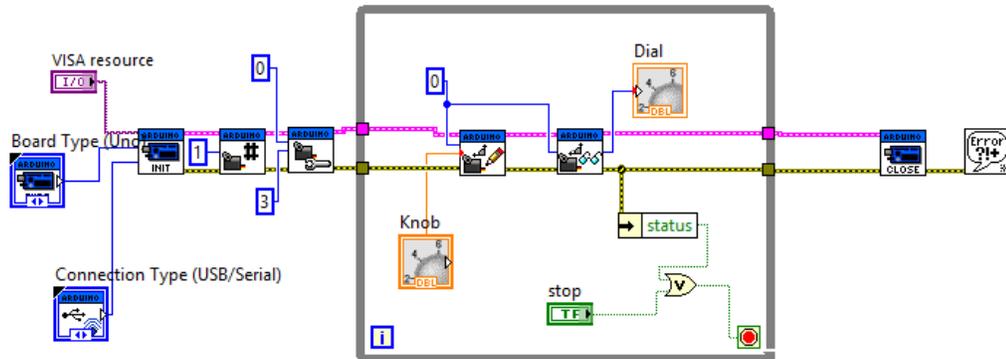
**Figura 5.** Bloques de servos en la librería de Arduino

11. En los bloques servos elegimos: set Number of servos , configure servo , servo write angle , servo read angle .
12. En el bloque Arduino INT  vamos a asignar el medio de comunicación VISA resource , el tipo de Arduino y el tipo de conexión (usb/serial).
13. En el bloque Arduino set number servos  indicamos cuantos servos se van a controlar.
14. En el bloque configure servo  se asigna el número de servos (0) y el pin en donde se conectara en la tarjeta Arduino (3), que es una salida PWM.
15. En el bloque servo write angle  se asigna en el número de servo (0) y en ángulo se pone el ángulo que se desea o a su vez se coloca un KNOB para poder variar el ángulo a nuestro consideración.
16. En el bloque Servo read angle  se asigna el número de servo (0) y en el angulo se coloca un indicador ya sea un METER o un DIAL para leer el ángulo que está variando en el servo.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing.Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

<b>REVISION 1/1</b>		<i>Página 1 de -</i>
		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible	
<b>CARRERA</b>	Ingeniería Electrónica	
<b>SEDE</b>	Guayaquil	

17. En el bloque Arduino CLOSE  se le coloca un bloque de error .



**Figura 6.** Diagrama de bloques para control de Servo Motor.

18. Conectamos nuestro servomotor en nuestra tarjeta Arduino, el cable de color amarillo va al PIN (3) que fue asignado, el cable rojo va al pin de 5v, y el cable negro va al PIN de GND.

19. Conectamos nuestro cable USB de nuestro Arduino a nuestro Computador.

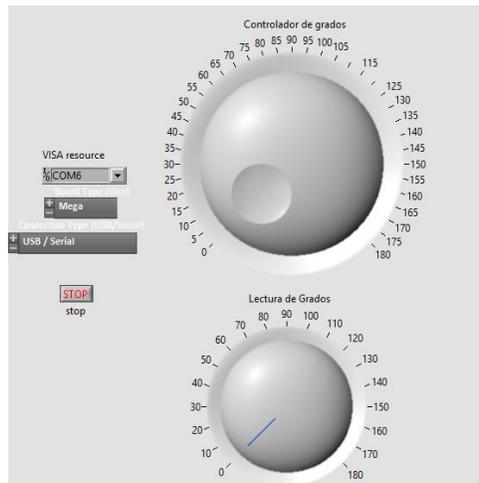
Simulación del Circuito

20. En el panel frontal de nuestro VI encontraremos una pestaña de VISA resource, elegimos el puerto COM que hará la conexión entre el Labview y el Arduino. En el tipo de Arduino elegimos la tarjeta con la que estamos trabajando ya sea un Arduino UNO o un Arduino MEGA.

21. En nuestro VI le damos RUN y comprobamos el funcionamiento de nuestro programa, visualizando tanto físicamente el movimiento del servo como virtualmente en Labview.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



**Figura 7.** Ejecutando Panel frontal de programa.

### Cuestionario

**5) ¿Cómo se llaman las dos plataformas que se utiliza para el control del servo motor en esta práctica?**

Se llaman Labview y Arduino.

**6) ¿Cuál es el intervalo de operación de un servo motor?**

El servo motor tiene un margen de posición de 0 a 180° aproximadamente.

**7) ¿Qué toolkits adicionales se necesita para la comunicación de Arduino y Labview?**

Se necesita tener Instalado el Labview, el toolkit de Arduino , el VISA y el VPM.

**8) ¿Qué programa se envía a la memoria del Arduino para que exista la comunicación?**

Se Compila al Arduino el “LIFA Base” se encuentra en : ” **C:\Program Files (x86)\National Instruments\LabVIEW 2011\vi.lib\LabVIEW Interface for Arduino\Firmware\LIFA\_Base “**

**9) ¿Es posible controlar más de un servo motor al mismo tiempo?**

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing.Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

Si es posible, se tiene que tener en consideración los puertos PWM de la tarjeta Arduino que serían la limitante.

**10) ¿Es posible visualizar la salida del Servo motor mediante un registrador grafico?**

Si es posible, con la ayuda de la herramienta Waveform Chart que se coloca a la salida del bloque de lectura de Angulo del servo de Arduino.

**f. RECURSOS UTILIZADOS (EQUIPOS, ACCESORIOS Y MATERIAL CONSUMIBLE)**

**Materiales necesarios:**

- Servo Motor
- Tarjeta Arduino
- Cables Multipar

**Instrumentos:**

- Computador en el cual este instalado Labview y los toolkits necesarios.

**g. REGISTRO DE RESULTADOS**

Esquema del panel frontal

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing.Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
 <b>UNIVERSIDAD POLITÉCNICA SALESIANA</b> ECUADOR		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

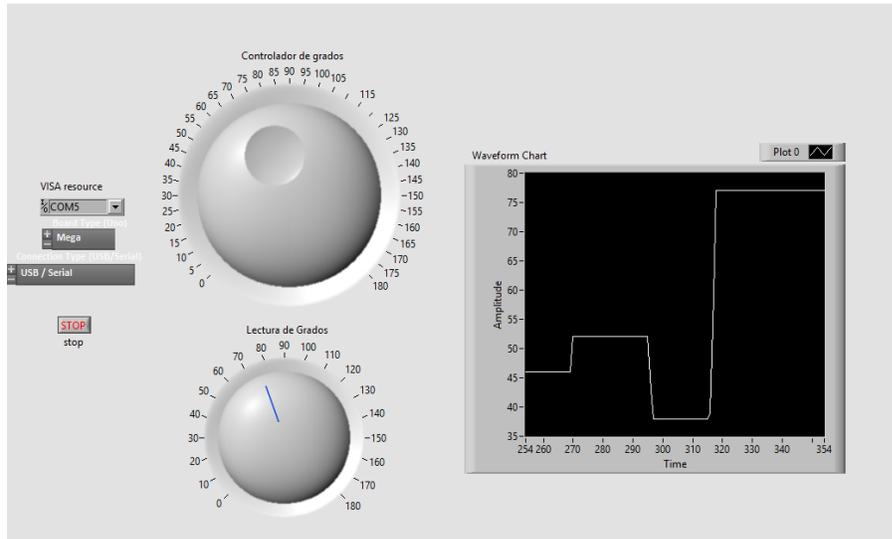
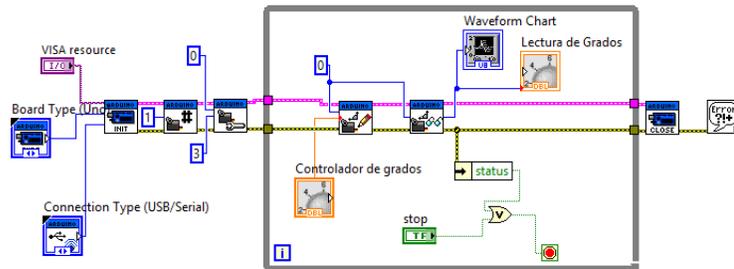


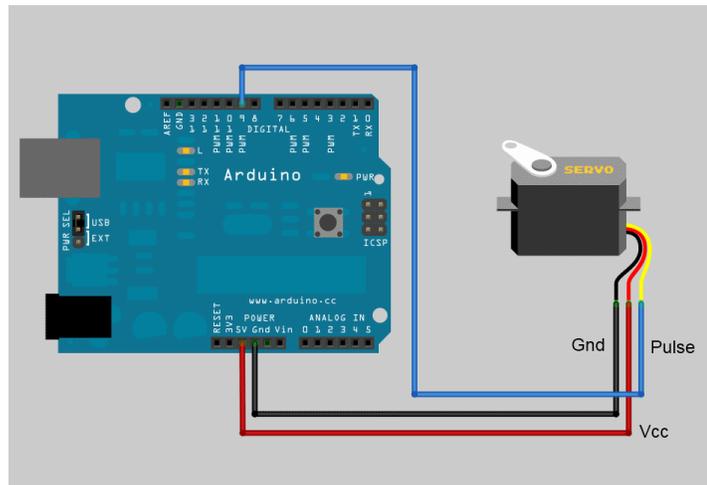
Diagrama de bloques



<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>MOVIMIENTO DE SERVOMOTORES UTILIZANDO EL CONTROLADOR ARDUINO Y LABVIEW.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

Vista en la tarjeta Arduino conexión con servomotor.



### Conclusiones:

Se logró comunicar el controlador Arduino con el software Labview y a su vez se realizaron pruebas de control de movimiento del Servo Motor mediante la librería de Arduino en Labview.

### Recomendaciones

Es necesario conectar correctamente el Pin de señal PWM de nuestro controlador en el cable de señal de nuestro servo Motor, y a su vez configurar el mismo Pin PWM en la programación de Labview.

### **h. BIBLIOGRAFÍA UTILIZADA**

Instrument, N. (Febrero de 2017). National Instrument. Obtenido de <http://www.ni.com/labview>

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing.Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>REALIZAR UNA APP EN ANDROID PARA ESTABLECER COMUNICACIÓN ENTRE BLUETOOTH Y ARDUINO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

## 1. DATOS INFORMATIVOS

a. MATERIA / CÁTEDRA RELACIONADA:

Electiva I / Robótica

b. No. DE PRÁCTICA: 3

c. NÚMERO DE ESTUDIANTES: 2

d. NOMBRE DOCENTE: Ing. Byron Lima.

e. TIEMPO ESTIMADO: 2 Horas

## 2. DATOS DE LA PRÁCTICA

a. **TEMA:**

**Realizar una app en Android para establecer comunicación entre bluetooth y Arduino.**

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>REALIZAR UNA APP EN ANDROID PARA ESTABLECER COMUNICACIÓN ENTRE BLUETOOTH Y ARDUINO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

**b. OBJETIVO GENERAL:**

- Desarrollar una aplicación en Android para poder interactuar entre un dispositivo móvil y Arduino mediante bluetooth.

**c. OBJETIVOS ESPECÍFICOS:**

- Familiarizar al alumno con el entorno de los módulos de Arduino y android app.
- Verificar y comprobar la comunicación entre un dispositivo móvil con android y la tarjeta Arduino.
- Aprender a razonar y a tomar decisiones ante los problemas que se le planteen, de tal forma que lo estimulen en la creatividad del alumno.
- Desarrollar el espíritu de equipo y de compañerismo.
- Integrar de una forma práctica la teoría cubierta en los cursos anteriores.

**d. MARCO TEÓRICO**

**MIT APP INVENTOR**

MIT App Inventor es un principiante introducción innovadora de la programación y la aplicación de creación que transforma el complejo lenguaje de codificación basada en texto en bloques de construcción visual, arrastrar y soltar. Las donaciones simples interfaces gráficas incluso un novato sin experiencia la posibilidad de crear una aplicación básica, completamente funcional dentro de una hora o menos.



<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>REALIZAR UNA APP EN ANDROID PARA ESTABLECER COMUNICACIÓN ENTRE BLUETOOTH Y ARDUINO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

**Figura 1.** Plataforma App Inventor.

En 2015, la comunidad del MIT App Inventor se compone de cerca de 3 millones de usuarios que representan a 195 países. Más de 100.000 usuarios activos semanales han construido más de 7 millones de android! Como una herramienta de código abierto que pretende realizar la programación y la creación de aplicaciones accesibles a una amplia gama de audiencias, el MIT App Inventor ha captado la atención de:

- Educadores formales e informales que han utilizado el MIT App Inventor para introducir la programación a sus estudiantes de informática, miembros del club de ciencias, después de la escuela asistentes programas, y los campistas de verano. Muchos educadores también han comenzado a usar el MIT App Inventor para desarrollar aplicaciones en apoyo de sus propios objetivos de instrucción.
- Los empleados del gobierno y civiles y voluntarios que han aprovechado el poder de MIT App Inventor para desarrollar aplicaciones a medida, a menudo hiper-locales en respuesta a los desastres naturales y las necesidades de la comunidad.
- Los diseñadores y gerentes de producto que han visto el potencial que tiene el MIT App Inventor para apoyar el proceso de diseño iterativo a través de creación rápida de prototipos, pruebas e iteración.
- Los investigadores que utilizan el MIT App Inventor para crear aplicaciones personalizadas que pueden procesar la recopilación de datos y los requisitos de análisis en una amplia variedad de campos de la medicina a social.
- Los aficionados y los empresarios que quieren convertir rápidamente una idea en una aplicación sin la curva de costos o el aprendizaje de los procesos de creación de aplicaciones más tradicionales. (Inventor, 2017)

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>REALIZAR UNA APP EN ANDROID PARA ESTABLECER COMUNICACIÓN ENTRE BLUETOOTH Y ARDUINO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



**Figura 2.** Estructura de App Inventor.

#### e. MARCO PROCEDIMENTAL

##### Iniciación de APP Inventor

12. Tener Internet en una PC
13. Tener Instalado un navegador web, colocar en la barra de navegación app inventor.
14. En la opción Get Started dar click en Start.
15. Se cargara otra ventana, buscaremos la opción [ai2.appinventor.mit.edu](http://ai2.appinventor.mit.edu) y le damos click.
16. Nos mostrara la interface beta del desarrollador app inventor.
17. Damos click en la opción Start New Proyecto, agregamos el nombre que queremos para nuestro proyecto.
18. Se nos abrirá una interface para programar nuestra app. En la parte izquierda se observara una paleta con herramientas, en la parte media estará la vista principal de nuestra app, en el lado derecho se mostraran los componentes que se utilizan las características de los mismos.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>REALIZAR UNA APP EN ANDROID PARA ESTABLECER COMUNICACIÓN ENTRE BLUETOOTH Y ARDUINO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

### Programación de la APP

22. En la paleta de herramientas comenzamos a ubicar las que necesitamos en nuestra interface principal.(Botoneras, labels de texto, imágenes)
23. Es importante no olvidar de colocar dentro de nuestra interface app la herramienta Bluetooth Client, es la que nos permite la conectividad con nuestro dispositivo móvil.



24. Luego de colocar los componentes que se necesitaran en nuestra app, vamos a la

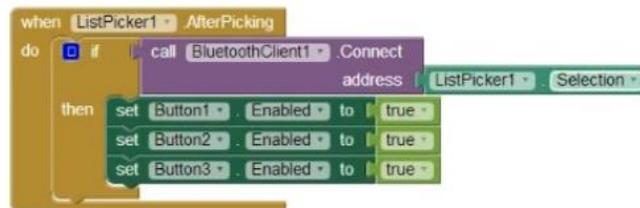
parte de superior derecha, damos click en la opción block.



25. Se nos abra una vista en donde en la parte izquierda aparezcan todos los componentes que seleccionamos en el diseño de nuestra app.
26. Comenzamos a programar la parte de comunicación con el bluetooth, utilizamos la opcion ListPicker en union con bluetooth client. Esta parte es antes que se conecte con el dispositivo bluetooth.



27. Luego programamos la seccion en donde el dispositivo se conecte mediante el bluetooth, cuando eso suceda los botones ya estarian habilitados.



28. Cuando el botón 1 se active se activara el bluetooth cliente y enviara un texto por bluetooth hacia el Arduino. Cabe indicar que el texto que se coloque en la app tiene que ser el mismo que se programe en el Arduino. Lo misma acción se realizara para los demás botones.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>REALIZAR UNA APP EN ANDROID PARA ESTABLECER COMUNICACIÓN ENTRE BLUETOOTH Y ARDUINO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



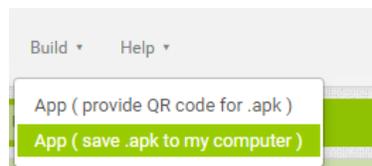
29. Cuando inicialice el screen, los arreglos horizontal tienen que estar visibles por lo siguiente se los asigna un valor true.



30. Cuando quiera salir de la aplicación presionamos el botón 3, que desconectara el bluetooth client y desactiva los demás botones para que permanezca inactivos.



31. Luego que se termina la programación podemos guardar la aplicación y podemos crear un apk para luego pasarlo a nuestro dispositivo e instalarlo. Damos click en la opción project y ahí seleccionamos la opción save apk to my computer, para guardar el apk en nuestro computador y luego instalarlo en nuestro dispositivo con android.



<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>REALIZAR UNA APP EN ANDROID PARA ESTABLECER COMUNICACIÓN ENTRE BLUETOOTH Y ARDUINO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

### Questionario

**11) ¿Cómo se llama la herramienta que utilizamos para establecer la conexión mediante bluetooth?**

Se llama bluetooth client.

**12) Explique brevemente sobre la comunicación entre el dispositivo con la app y la tarjeta arduino**

Se busca el modulo bluetooth a conectar, una vez establecida la conexión se procede a ejecutar o accionar botones que ellos a su vez envían caracteres o datos al Arduino para que el Arduino los reciba y accione lo que se tenga programado.

**13) ¿Cuántas opciones se tiene para instalar la aplicación en el dispositivo movil?**

Se tiene dos opciones: la primera es tener un código QR para así instalar la aplicación mediante la app store y la otra opción es descarga el apk en la computadora y de igual forma realizar la instalación en el dispositivo.

**14) ¿Qué tipo de carácter tiene que ser el que se envía mediante bluetooth al Arduino?**

Puede ser cualquier tipo de carácter, por ejemplo si se pone una letra “a” en minúscula de igual forma tiene que ser declarada en la programación del Arduino “a” en minúscula.

### f. RECURSOS UTILIZADOS (EQUIPOS, ACCESORIOS Y MATERIAL CONSUMIBLE)

**Materiales necesarios:**

- Tarjeta Arduino

**Instrumentos:**

- Computador con internet.

### g. REGISTRO DE RESULTADOS

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>REALIZAR UNA APP EN ANDROID PARA ESTABLECER COMUNICACIÓN ENTRE BLUETOOTH Y ARDUINO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



### **Conclusiones:**

Se logró realizar la comunicación Bluetooth y él envió de Datos tipo carácter mediante la conexión RX y TX del controlador Arduino. Se logró interactuar con la aplicación y verificar su funcionamiento con el controlador Arduino

### **Recomendaciones**

Es importante declarar correctamente el carácter que quiere asignar tanto en la programación en Arduino, como en la programación en la aplicación en Android para que luego haya un correcta comparación de datos.

### **h. BIBLIOGRAFÍA UTILIZADA**

Inventor, A. (Febrero de 2017). *APP Inventor*. Obtenido de <http://ai2.appinventor.mit.edu/>

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
 <b>UNIVERSIDAD POLITÉCNICA SALESIANA</b> <small>ECUADOR</small>		<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

## 1. DATOS INFORMATIVOS

a. MATERIA / CÁTEDRA RELACIONADA:

Electiva I / Robótica

b. No. DE PRÁCTICA: 4

c. NÚMERO DE ESTUDIANTES: 2

d. NOMBRE DOCENTE: Ing. Byron Lima.

e. TIEMPO ESTIMADO: 2 Horas

## 2. DATOS DE LA PRÁCTICA

a. **TEMA:**

**Variables físicas de un objeto en movimiento utilizando el Director Electro Óptico.**

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

**b. OBJETIVO GENERAL:**

- Encontrar variables físicas de un objeto en movimiento mediante seguimiento de patrones utilizando herramientas de Visión Artificial de Labview.

**c. OBJETIVOS ESPECÍFICOS:**

- Familiarizar al alumno con el entorno de los módulos de Arduino y Labview.
- Verificar y comprobar las variables físicas de un objeto a seguir utilizando el DEO.
- Aprender y comprobar el uso de las herramientas de visión de Labview.
- Aprender a razonar y a tomar decisiones ante los problemas que se le planteen, de tal forma que lo estimulen en la creatividad del alumno.
- Desarrollar el espíritu de equipo y de compañerismo.
- Integrar de una forma práctica la teoría cubierta en los cursos anteriores.

**d. MARCO TEÓRICO**

**MODULO DE VISIÓN**

**Visión Artificial**

Podríamos decir que la Visión Artificial (VA) describe la deducción automática de la estructura y propiedades de un mundo tridimensional posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales del mundo. Las imágenes pueden ser monocromáticas (de niveles de gris) o colores, pueden provenir de una o varias cámaras e incluso cada cámara puede estar estacionaria o móvil.

Las estructuras y propiedades del mundo tridimensional que queremos deducir en visión artificial incluyen no sólo sus propiedades geométricas, sino también sus propiedades materiales. Ejemplos de propiedades geométricas son la forma, tamaño y localización de los objetos. Ejemplos de propiedades de los materiales son su color, iluminación, textura y composición.

Si el mundo se modifica en el proceso de formación de la imagen, necesitaremos inferir también la naturaleza del cambio, e incluso predecir el futuro. La entrada a un sistema de VA es una imagen obtenida por un elemento de adquisición, mientras que su salida es una descripción de la escena, la cual ha sido obtenida a partir de la imagen. Por un lado, esta descripción debe estar relacionada de algún modo con aquella realidad que produce la imagen y, por el otro, debe contener toda la información requerida para la tarea de interacción con el medio ambiente que se desea llevar a cabo, por ejemplo mediante un

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

robot. Esto es, la descripción depende en alguna forma de la entrada visual y debe proporcionar información relevante y utilizable por el robot. (MALPARTIDA, 2003)

#### e. MARCO PROCEDIMENTAL

##### Ejecucion del sistema DEO

19. Tener Instalado Labview
20. Instalar Driver de Labview y los toolkits de Ni IMAQ y Ni Visión Develoment.
21. Instalar el Labview Package Manager, por medio de este gestor se instala el VIPM Arduino toolkit.
22. Descargar el Arduino Toolkit del VIPM
23. Instalar el IDE Arduino
24. Verificar el reconocimiento del puerto USB parte del IDE Arduino, por medio del administrador de dispositivos de Windows.
25. Para que Labview pueda comunicarse con Arduino se enviara a la memoria de Arduino el programa LIFA Base, el programa se encuentra en la carpeta " **C:\Program Files (x86)\National Instruments\LabVIEW 2011\vi.lib\LabVIEW Interface for Arduino\Firmware\LIFA\_Base** " luego que se compila el programa a la tarjeta este ya se encarga de comunicarse con el Arduino.

##### Crear Comunicación de nuestro Programa

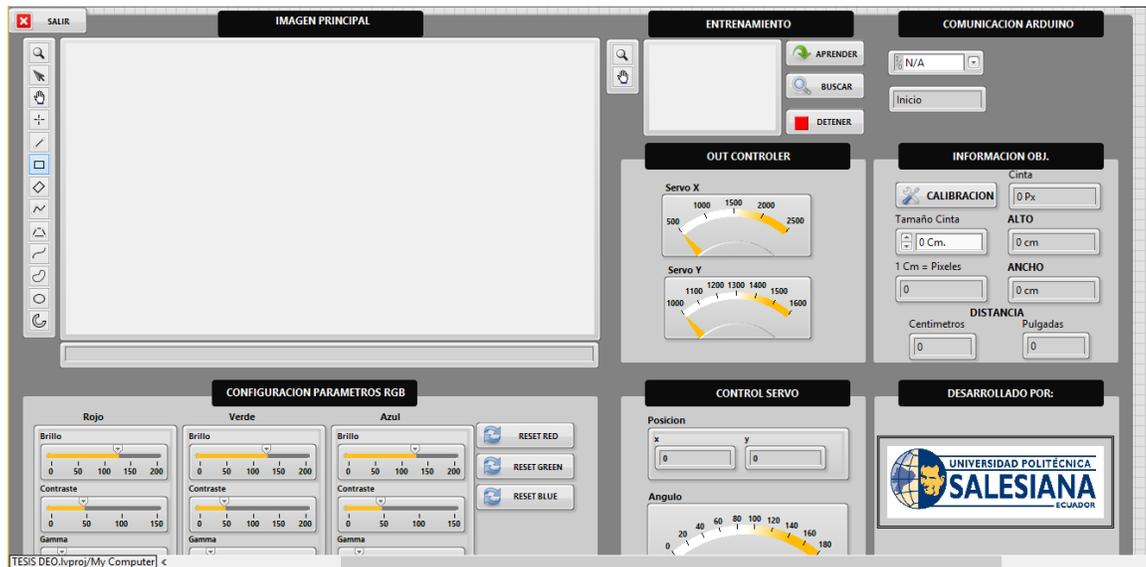
32. Abrir el programa Labview
33. Escoger Nuestro Programa DEO
34. Ir a la opción Windows, elegir Tile left and Right
35. En el panel de control de nuestro programa en la pestaña de Visa Resource elegimos el Puerto COM del Arduino para establecer comunicación, luego buscamos la pestaña Cam en donde elegiremos nuestra cámara a utilizar.

##### Simulación del Programa

36. Una vez que elegimos la cámara y el puerto com, damos Click en RUN continuo y vemos como se ejecuta nuestro programa.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

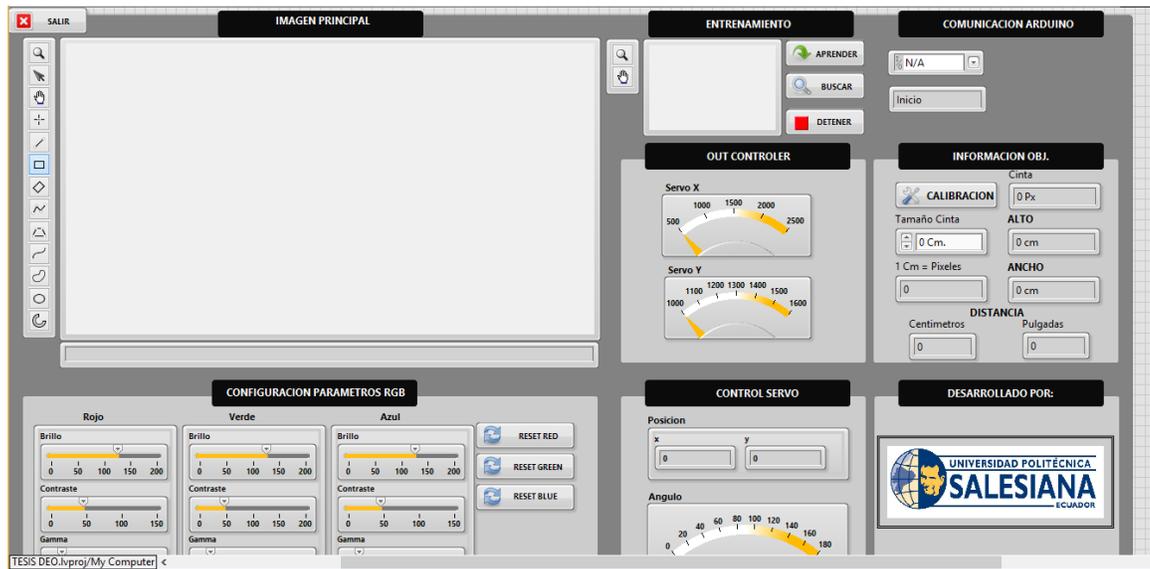


**Figura 1.** Vista del programa en ejecución en el panel Frontal.

37. Una vez dado Run podemos maniobrar con nuestro sistema DEO de Labview, observemos que tenemos tres botonerías: Aprender, buscar y Detener los tres botones son los que nos permitirán guardar un patrón a seguir, dar un seguimiento al patrón guardado y detener el evento en ejecución. También tenemos 2 displays: Entrenamiento que guarda la imagen del patrón a seguir e imagen Principal que sigue el patrón guardado en tiempo real.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

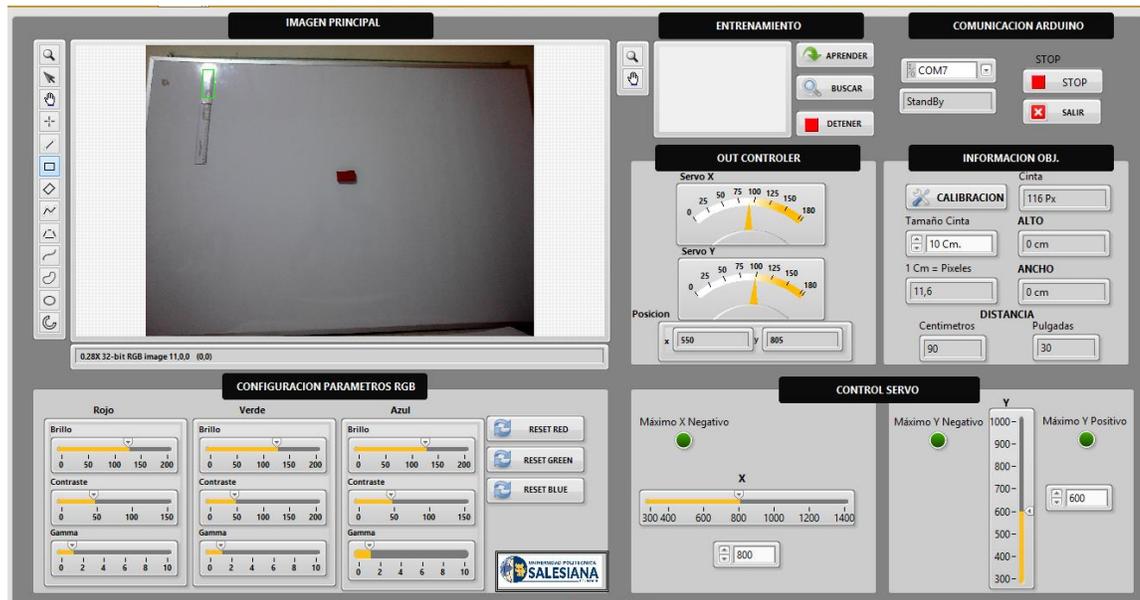


**Figura 2.** Panel frontal de nuestro Programa

38. Comenzamos seleccionando el objeto a seguir utilizando el cursor de nuestro mouse.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		



**Figura 3.** Ejecución del programa – Selección de patrón ROI.

39. Luego de seleccionar el patrón que queremos seguir, damos click en el botón aprender podemos observar como el patrón que seleccionamos se guarda en el display de Entrenamiento.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución</b> <b>Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

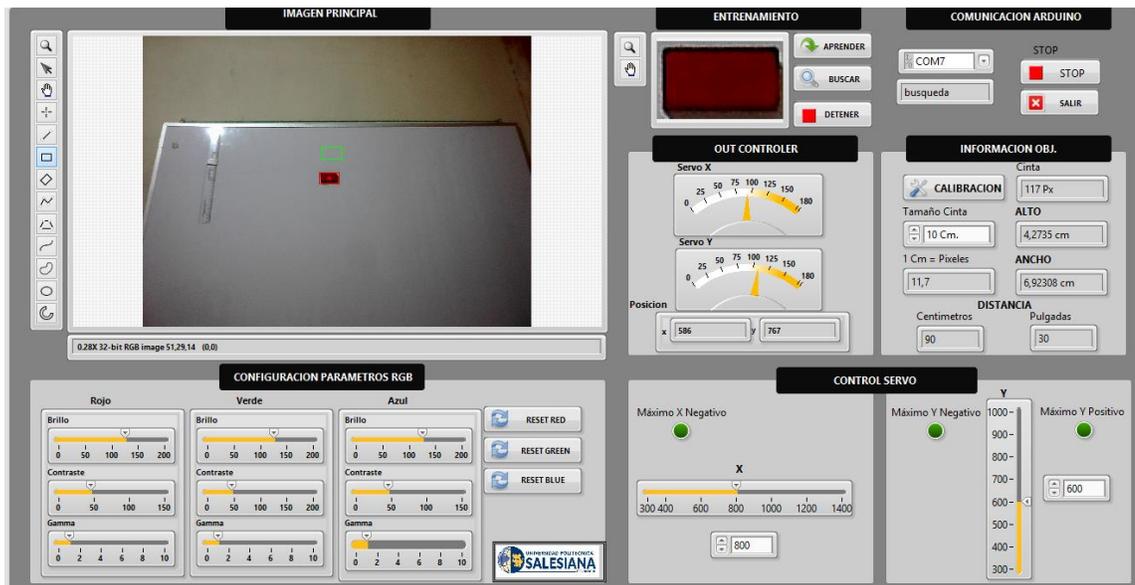


**Figura 4.** Ejecución del programa – Botón Aprender

40. Luego podemos dar click en el botón Buscar y nuestra imagen principal podemos observar como aparece un ROI de Color rojo que es el que realiza el seguimiento del patrón asignado.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución</b> <b>Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

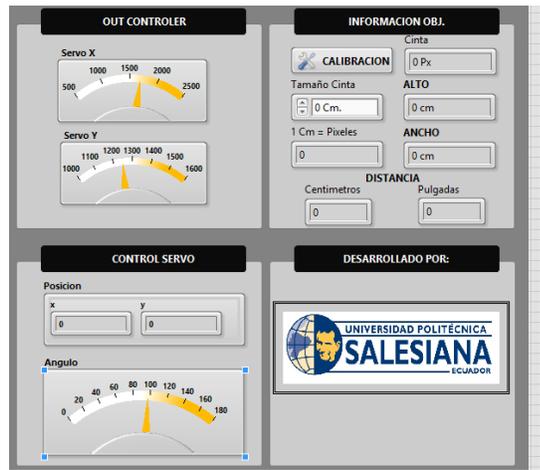


**Figura 5.** Ejecución del programa - Botón Buscar

41. Se puede mover el objeto a seguir y comprobar que al mismo tiempo el cuadro de matches sigue el objeto.
42. En la parte inferior y lateral se muestra donde tenemos controles para calibrar el funcionamiento de la medición del tamaño del objeto que estamos siguiendo. Para obtener el tamaño del objeto tenemos que seleccionar de referencia una sección de la cinta métrica y luego dar click en el botón de Calibración, donde se guardaran valores automáticamente para cuando luego nosotros seleccionemos el objeto a seguir y demos click en Aprender nos salgan los valores aproximados del tamaño del objeto.
43. Tenemos un indicador Meter en donde se visualiza el movimiento virtual de los servo motores.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

<b>REVISION 1/1</b>		<i>Página 1 de -</i>
		<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible	
<b>CARRERA</b>	Ingeniería Electrónica	
<b>SEDE</b>	Guayaquil	



**Figura 6.** Ejecución del programa – Tamaño del objeto- Meter del servo motor.

### Questionario

**15) ¿Cómo se llaman las plataformas que se utilizan para el funcionamiento del sistema DEO?**

Se llaman Labview, Arduino, NI IMAQ, NI Vision development y Fuzzy Logic Toolkit.

**16) ¿Indique sobre que ejes están trabajando los servos motores que tienen la cámara?**

Los servo motores que tienen la cámara, trabajan en el eje X y en el eje Y. En un Ángulo de 0 a 180 ° aproximadamente.

**17) ¿Qué Variables se puede obtener con el sistema DEO?**

El sistema DEO detecta objetivos, procesa la imagen, captura la imagen e indica el tamaño, la distancia y la posición del objeto detectado.

**18) ¿Qué controlador se utiliza para el posicionamiento del Servo?**

Nuestro controlador es un control ON // OFF con histéresis.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

		<b>REVISION 1/1</b>	<i>Página 1 de -</i>
		<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible		
<b>CARRERA</b>	Ingeniería Electrónica		
<b>SEDE</b>	Guayaquil		

**19) ¿Qué Bloque se utiliza para realizar la búsqueda de patrones en objetos?**

Para realizar la búsqueda de patrones utilizamos el bloque de visión Pattern Matching.

**20) ¿Es posible visualizar la salida del Servo motor mediante un registrador grafico?**

Si es posible, con la ayuda de la herramienta Waveform Chart que se coloca a la salida del bloque de lectura de Angulo del servo de Arduino.

**f. RECURSOS UTILIZADOS (EQUIPOS, ACCESORIOS Y MATERIAL CONSUMIBLE)**

**Materiales necesarios:**

- Servo Motor
- Tarjeta Arduino
- Cables Multipar

**Instrumentos:**

- Computador en el cual este instalado Labview y los toolkits necesarios.

**g. REGISTRO DE RESULTADOS**

Esquema del panel frontal

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

<b>REVISION 1/1</b>		<i>Página 1 de -</i>
 <b>UNIVERSIDAD POLITÉCNICA SALESIANA</b> ECUADOR	<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible	
<b>CARRERA</b>	Ingeniería Electrónica	
<b>SEDE</b>	Guayaquil	

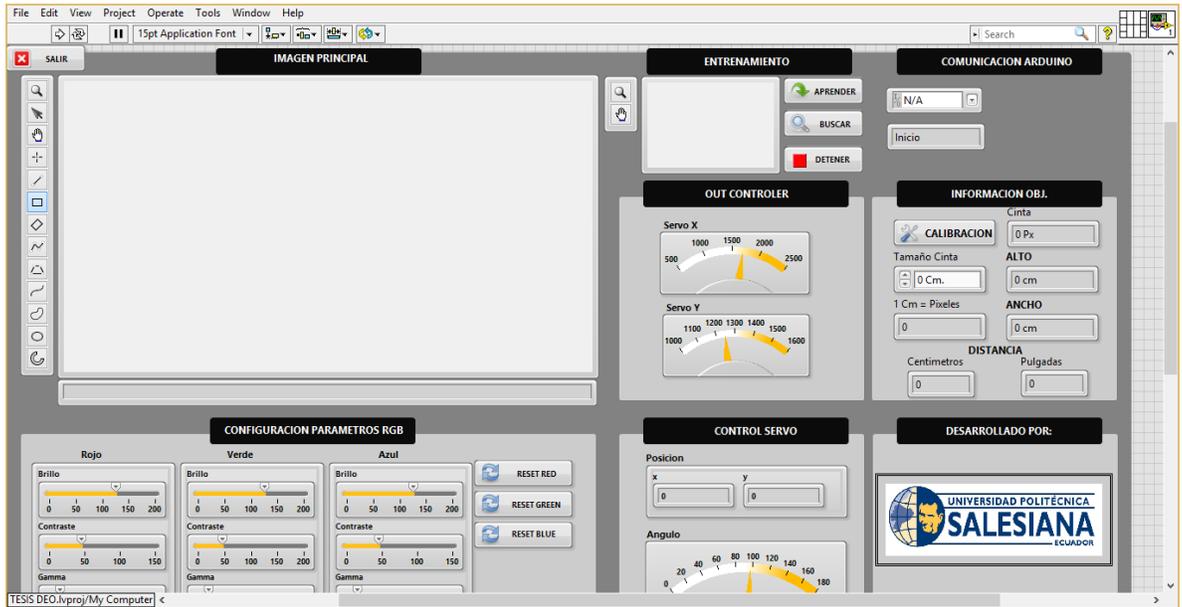
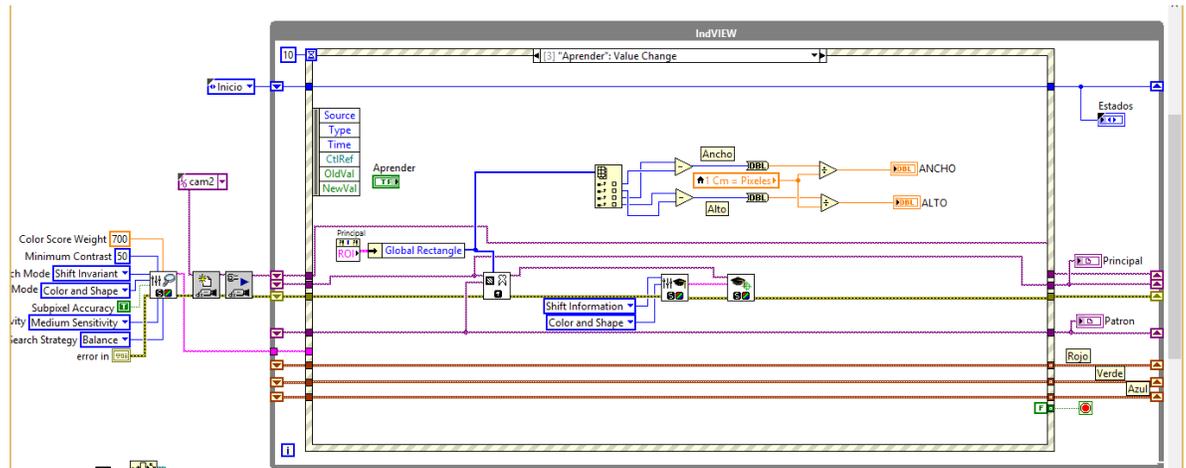
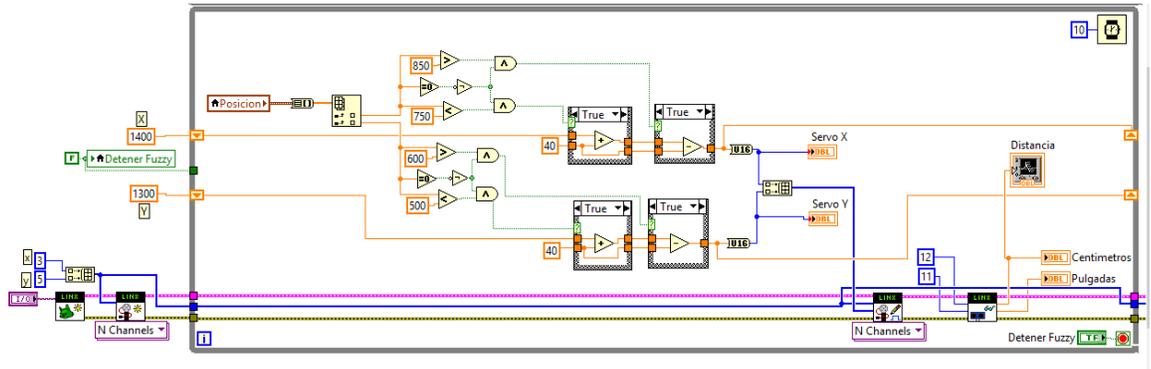


Diagrama de bloques



<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Victor Huilcapí MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>

<b>REVISION 1/1</b>		<i>Página 1 de -</i>
	<b>VARIABLES FÍSICAS DE UN OBJETO EN MOVIMIENTO UTILIZANDO EL DIRECTOR ELECTRO ÓPTICO.</b>	
<b>LABORATORIO</b>	Laboratorio de Fabricación Flexible	
<b>CARRERA</b>	Ingeniería Electrónica	
<b>SEDE</b>	Guayaquil	



### Conclusiones:

Se obtuvo el seguimiento del objeto y las variables deseadas de distancia, posición, y tamaño del objeto deseado. Se verificó las etapas del controlador de visión en conjunto con el control on/off del servo motor.

### Recomendaciones

Se recomienda considerar la intensidad de luz y la resolución de la cámara para obtener un mejor resultado en el procesamiento y adquisición de la imagen para consiguiente realizar las pruebas del sistema DEO.

### **BIBLIOGRAFÍA UTILIZADA**

MALPARTIDA, E. A. (2003). SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO Y MANIPULACIÓN DE OBJETOS UTILIZANDO UN BRAZO ROBOT. Lima.

<b>Elaborado por:</b> Luis Ulloa Mejia	<b>Revisado por:</b> Ing. Byron Lima MSc.	<b>Aprobado por:</b> Ing. Víctor Huilcapi MSc.
<b>Fecha de Elaboración</b> 14/02/2017	<b>Fecha de Revisión</b>	<b>Número de Resolución Consejo de Carrera:</b>