Universidad del Bío-Bío
Facultad de Ciencias Empresariales

Software reconocedor de patrones de gestos y voz a través de Kinect para el movimiento de un brazo robótico por conexión Arduino (KiBot).

Proyecto de título para optar al título de Ingeniero Civil en Informática.

24 de Junio de 2013
Concepción – Chile

Integrantes:
Braulio Cifuentes Cuadra.
Jaime Novoa Rivera.

Profesor guía:
Juan Carlos Parra Márquez.

Resumen

Este proyecto se presenta para dar conformidad a los requisitos exigidos por la Universidad del Bío–Bío para el proceso de titulación de la carrera de Ingeniería Civil en Informática.

Esta investigación presenta los resultados del reconocimiento de patrones a través de Kinect de Microsoft, dispositivo utilizado para la captura de imágenes y sonido. Este consta de una cámara, un emisor infrarrojo y un micrófono.

La cámara capta las articulaciones y extremidades de las personas que están frente a ella y dibuja un esqueleto en un espacio tridimensional. Además con el micrófono incorporado se pueden comparar palabras o frases estipuladas por el sistema, para ver si estas coinciden y realizar una acción determinada.

Lo anteriormente nombrado servirá de manera de entrada al sistema, el cual enviara señales al Arduino que permitirá el movimiento de un brazo robótico con motores DC.

Abstract

This project appears to give pursuant to the requirements of the Universidad del Bío-Bío for the degree process of the Civil Informatics Engineering .

This research presents the pattern results's recognition through Microsoft Kinect, a device used to capture images and sound. This consists of a camera, an infrared emitter and a microphone.

The camera captures the joints and limbs of people who are against it and draw a skeleton in a three dimensional space. In addition to the built-in microphone you can compare words or phrases set by the system, to see if they match and perform an action.

The above named will serve as an input to system, which send signals to the Arduino which allows the movement of a robotic arm with DC motors.

Índice General

Índice Figuras

1 INTRODUCCIÓN

El hombre en su afán de ampliar sus conocimientos es que ha incluido a la robótica en su diario vivir, pero el desarrollo de esta tecnología aun no ha sido totalmente investigado como se demuestra en películas y/o videos. Por este motivo, se ha decidido investigar cómo controlar un robot a través de movimientos del cuerpo humano reconocidos por Kinect, con el fin de poder obtener conociendo que pueda ser utilizado a futuro en empresas, educación e incluso el hogar.

La Kinect comenzó siendo un dispositivo usado en la consola Xbox 360, con el fin de promover el ejercicio a través de los videojuegos. Tal fue su impacto dentro de la sociedad que distintas áreas fueron las que intentaron incluir este artefacto para realización de distintas actividades.

Es por esto, que se ha decidido utilizar Kinect como reconocedor de patrones de movimiento, con lo que se desea enviar esta información a través de un dispositivo Arduino para que este pueda remitir estas señales en el brazo robótico.

2 DEFINICIÓN DE LA EMPRESA O INSTITUCIÓN

2.1 Descripción de la empresa

Creada en 1989, la Facultad de Ciencias Empresariales (FACE) ubicada en Av. Collao #1202 tiene como objetivo prioritario el cultivo de las disciplinas de Administración, Auditoría, Finanzas, Computación e Informática, enfatizando como áreas de estudio el Desarrollo Regional, Pequeña y Mediana Empresa, Gerencia Pública, Planificación y el Control de Gestión Estratégico, Política de Negocios, Tecnologías de Información y Gestión Informática.
Su permanente búsqueda de este objetivo se manifiesta no sólo en la docencia regular que imparte, sino también en la creación de programas de Diplomados, Postítulos y Magíster, en el desarrollo de proyectos de investigación y en la realización de actividades de extensión, asistencia técnica y capacitación en las áreas anteriormente mencionadas.

Misión:
Somos una facultad comprometida en la formación de profesionales de excelencia y en la creación y transmisión de conocimientos que sean un aporte de calidad para el desarrollo regional, basándonos en el cultivo de las áreas disciplinarias de gestión e informática.
Nuestra responsabilidad es ser un referente válido en la renovación permanente del conocimiento por medio de la excelencia académica, respondiendo apropiadamente a las inquietudes del medio empresarial y de la sociedad en general en las disciplinas que nos son propias.

Visión:
Ser una facultad destacada por su excelencia académica en el ámbito de las ciencias empresariales y ciencias de la computación.

La estructura organizacional es la siguiente:

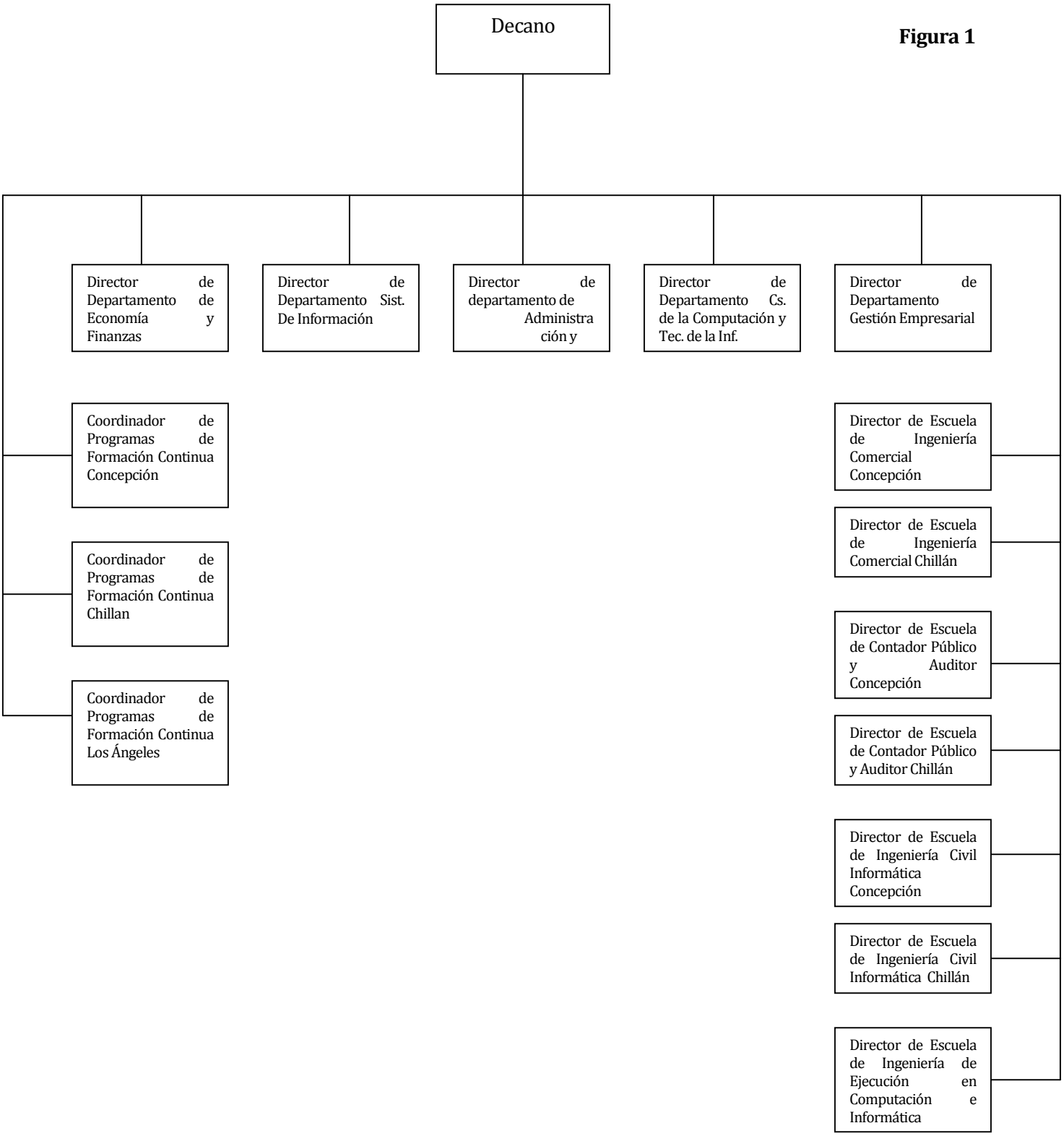


Figura 1

2.2 Descripción de la investigación

El proyecto contempla el estudio de la SDK Kinect for Windows y documentación sobre el uso de Arduino, con el fin de obtener el desarrollo de un software que permita mover un brazo robótico.
Este brazo robótico se moverá a través del uso de Kinect, la cual reconocerá patrones pre-establecidos del cuerpo humano como gestos y voz. La comunicación entre el brazo robótico y el computador se hará a través de una placa Arduino, la cual detectara las señales emitidas por computador que son redistribuidas a través de los distintos integrados conectados a la placa y serán interpretados con movimientos para el robot. La Kinect se conectara al computador a través de USB, en donde existirá un software (KiBot) que interprete dichos movimientos.

3 PLANIFICACIÓN INICIAL DE LA INVESTIGACIÓN

3.1 Introducción

En esta sección se especifican los objetivos de la investigación comprendido por sus siglas y abreviaciones además las principales funciones que debe cumplir el software.

3.1.1 Objetivos del proyecto

Objetivos generales:

A través de la investigación, diseñar e implementar un software que permita reconocer patrones de gestos y voz originados por el cuerpo humano, con el fin de replicar estos movimientos en acciones de un brazo robótico. Además del desarrollo del software se monta una placa Arduino, la cual será el interlocutor entre dispositivos (kinect y brazo robótico).

Objetivos específicos:

- Reconocer un cuerpo humano.
- Reconocer patrones de gestos.
- Reconocer patrones de voz.
- Calibración de cámara.
- Mover brazo robótico.
- Descifrar información y convertir en señales a través de Arduino.

3.1.2 Definiciones, Siglas y Abreviaciones

Las abreviaciones utilizadas en este documento serán las siguientes:

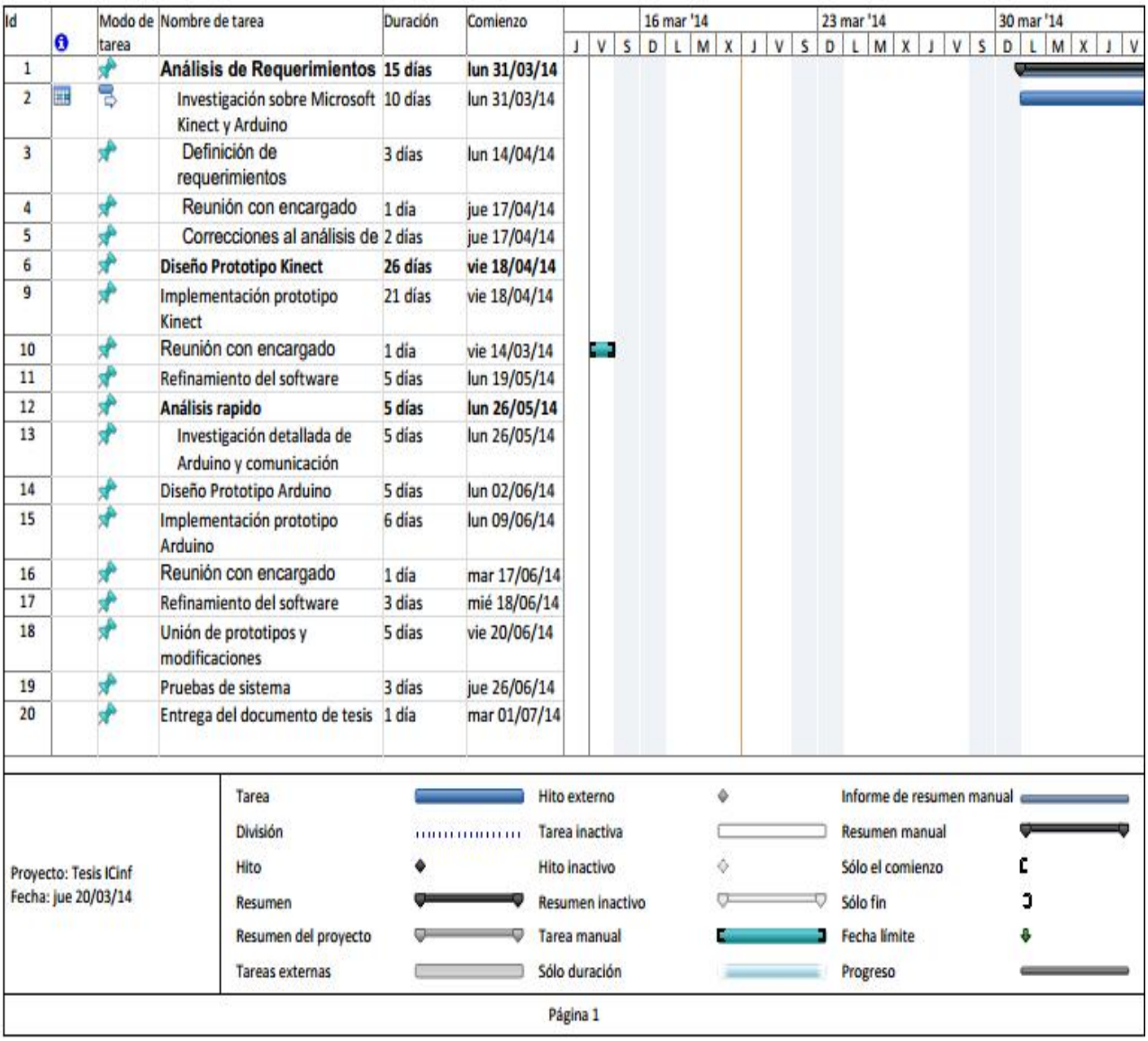
Figura 2

1. Kinect: Cámara utilizada para videojuegos (XBOX).
2. Arduino: Placa con microcontrolador, diseñada para facilitar la electrónica en proyectos.
3. SDK: Kit de desarrollo de Software
4. Hardware: Refiere a la parte física que compone a un computador, ya sea de escritorio, personal.
5. Open Source: Refiere a software que se distribuye bajo licencia igual-compartir, esto permite adquirirlos sin necesidad de pagar licencia alguna, sin costo.
6. FACE: Facultad de Ciencias Empresariales.
7. Windows: Windows es el sistema operativo más popular que existe hoy en día, desarrollado por Microsoft.
8. IEEE: Corresponden a Institute of Electrical and Electronics Engineers, tiene por función la estandarización.
9. Puente H: Circuito electrónico, que permite a un motor DC girar en ambos sentidos.
10. Integrado: Circuito electrónico.
11. Dupont: Nombre de cables utilizados para arduino.
12. VS: Software utilizado para desarrollo de Software
13. KWSRL: Kinect for window Speech Recongnition Language pack
14. DC: Corriente continua.

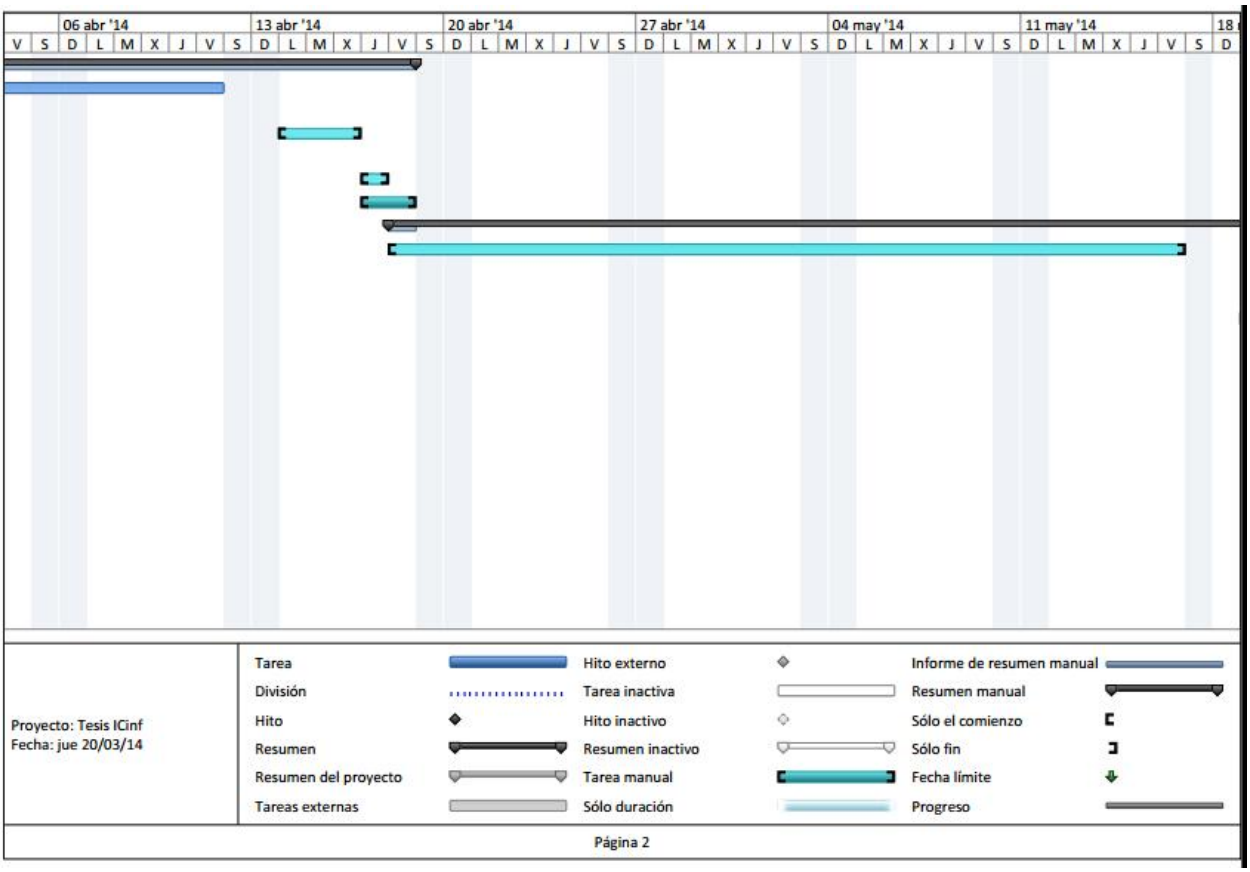
3.2 Planificación temporal

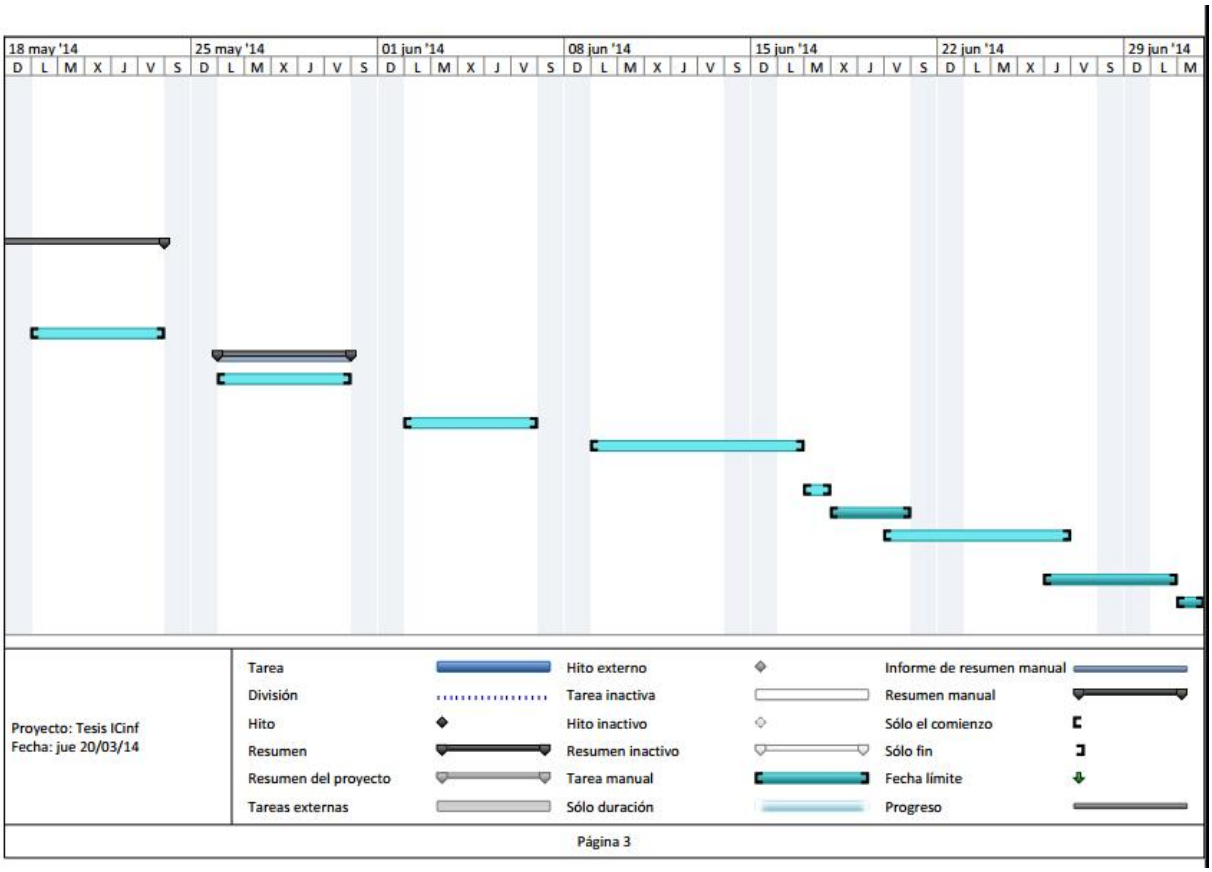
A continuación se presenta la planificación del proyecto.

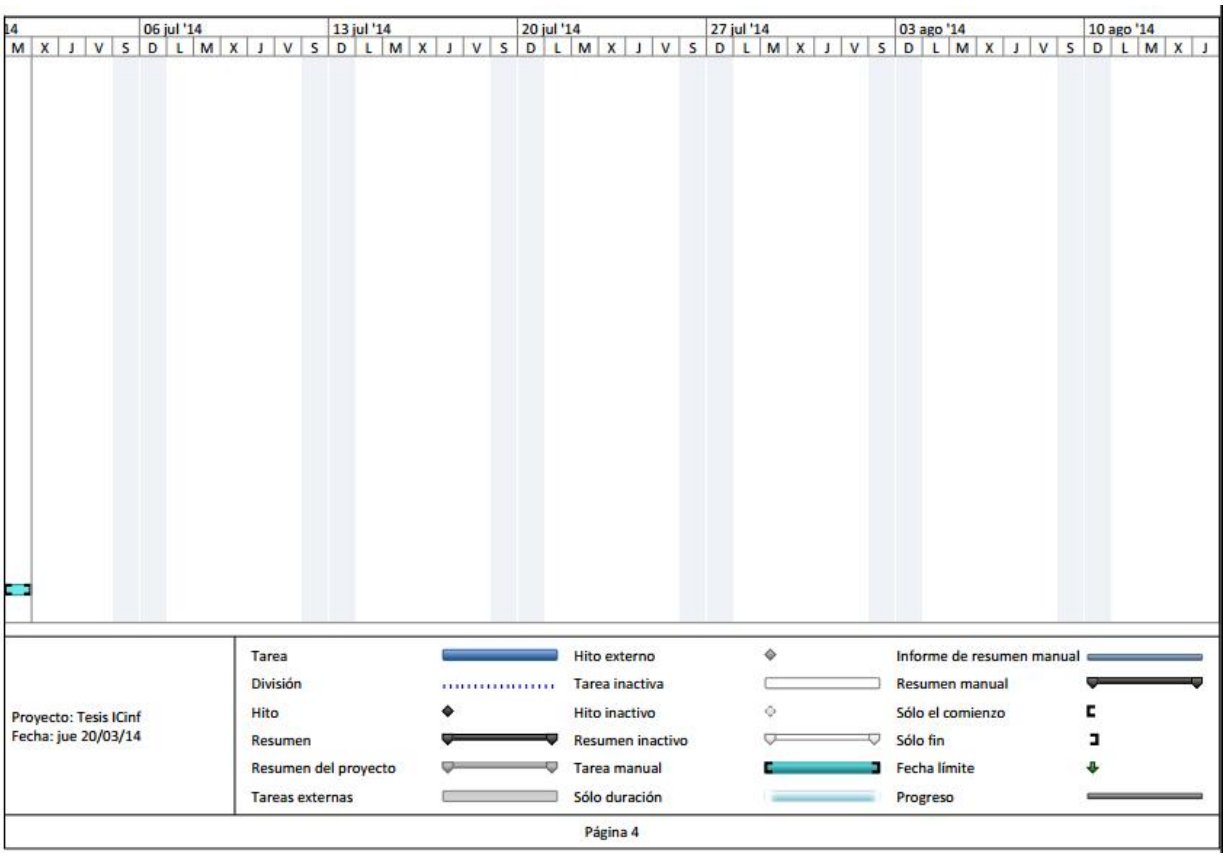
Figura 3



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo
1		**Análisis de Requerimientos**	**15 días**	**lun 31/03/14**
2		Investigación sobre Microsoft Kinect y Arduino	10 días	lun 31/03/14
3		Definición de requerimientos	3 días	lun 14/04/14
4		Reunión con encargado	1 día	jue 17/04/14
5		Correcciones al análisis de	2 días	jue 17/04/14
6		**Diseño Prototipo Kinect**	**26 días**	**vie 18/04/14**
9		Implementación prototipo Kinect	21 días	vie 18/04/14
10		Reunión con encargado	1 día	vie 14/03/14
11		Refinamiento del software	5 días	lun 19/05/14
12		**Análisis rapido**	**5 días**	**lun 26/05/14**
13		Investigación detallada de Arduino y comunicación	5 días	lun 26/05/14
14		Diseño Prototipo Arduino	5 días	lun 02/06/14
15		Implementación prototipo Arduino	6 días	lun 09/06/14
16		Reunión con encargado	1 día	mar 17/06/14
17		Refinamiento del software	3 días	mié 18/06/14
18		Unión de prototipos y modificaciones	5 días	vie 20/06/14
19		Pruebas de sistema	3 días	jue 26/06/14
20		Entrega del documento de tesis	1 día	mar 01/07/14







3.3 Ambiente de ingeniería de SW

Metodología

Para la construcción del proyecto, se evaluaron diversos métodos de desarrollo, dentro de los cuales, un método se adapta a nuestros requerimientos. La metodología escogida es prototipo, ya que nuestros conocimientos sobre el tema a investigar y desarrollar son sumamente básicos, de manera que podamos minimizar los riesgos durante la creación del software, además se irán desarrollando avances paulatinos lo cual nos permite verificar que lo desarrollado va de acorde a las necesidades del proyecto.
La imagen a continuación muestra la estructura de la metodología.

Figura 4

Estándares de documentación.

Se utiliza la Plantilla de Documentación del proyecto de desarrollo de software (versión 6 de Diciembre de 2011) en los puntos 1, 2 y 3.

Para el apoyo de desarrollo del proyecto las herramientas son las siguientes:

Nombre: Visual Studio
Número de versión: 2012
Descripción: IDE para desarrollo de software.
Tipo de licencia: Licencia estudiante.

Nombre: Arduino
Número de versión: 1.0.5
Descripción: Enterno de desarrollo para lenguaje de programación Arduino.
Tipo de licencia: Libre.

Nombre: Kinect for Windows Developer Toolkit
Número de versión: 1.8.0
Descripción: Herramientas de Kinect para desarrollo.
Tipo de licencia: Libre.

Nombre: Kinect for Windows Driver
Número de versión: 1.8
Descripción: Reconocimiento de Kinect.
Tipo de licencia: Libre.

Nombre: Kinect for Windows SDK
Número de versión: 1.8
Descripción: Librerías para el desarrollo con Kinect.
Tipo de licencia: Libre.

Nombre: Kinect for Windows Speech Recongnition Language Pack (es-MX)
Número de versión: 11.0
Descripción: Lenguaje español para el reconocimiento de palabras con Kinect.
Tipo de licencia: Libre.

Microsoft OFFICE 2007:
Software utilizado en le creación y edición de los informes.

Para el desarrollo se utilizarán notebooks con las siguientes características:

Figura 5

Marca: Lenovo
Modelo: Z400
Procesador: Intel Core I5-3520M
Memoria:6 GB DDR3
Disco Duro: 1TB RPM 7200 SATA
Pantalla: Ancha de 14" Resolución HD (1366x768). Iluminación Retro LED. Acabado Brillante
Video: GeForce GT 730M 1GB
Cámara Web: 2 MP integrada con micrófono integrado
Touch Pad: Integrado
Lan: 10/100 Ethernet
WiFi: 802.11b/g /n
BlueTooth: Dispositivo integrado
1 x VGA
1 x HDMI
3 x USB 2.0
1 x RJ-45
1 x Entrada Micrófono
1 x Salida audífono (estéreo)
1 x Puerto Express Card /34

Figura 6

Marca: Dell
Series: Inspiron
Model: 14-N4050
Color: Negro
Sistema operativo: Windows 7 Ultimate 32 bits
Especificaciones
Procesador: Intel Core i3-2330M 2.20GHz
Pantalla: 14"
Resolución: 1366 x 768 Píxeles
Memoria RAM: 2GB DDR3
Disco Duro: 500GB
Unidad Óptica: 8X CD/DVD
Video: Intel HD Graphics 3000
Wi-Fi: Standard Dell Wireless TM 802.11b/g/n
Webcam: 1.3MP HD (1280 x 720)
Puertos

HDMI: 1 x HDMI
Lector de tarjetas: 5-en-1
USB: 3 x USB 2.0
VGA: 1 x VGA
Audio: 1 x Micrófono / 1 x Audífono
Físicas
Dimensiones: 13.5" x 9.69" x 1.20" - 1.33"
Peso: 2,24 kg.

4 ALCANCES DEL PROYECTO

4.1 Limites

El presente proyecto se enmarca en el funcionamiento de Kinect, la cual para reconocimiento de voz, esta puede registrar cualquier tipo y tono de voz, mientras se hable el idioma español (latinoamericano), cualquier otro idioma no fue implementado para este proyecto, además, solo se reconoce las palabras claves designadas por los desarrolladores. En lo que respecta al movimiento, reconoce gestos previamente definidos por los desarrolladores. En todos los movimientos definidos se realizan con las 4 extremidades, pero para poder tener control del menú solo es necesaria la mano derecha.

En lo que respecta a Arduino, su limitación corresponde a que el circuito desarrollado solo moverá motores de corriente continua (Motores DC), ya que si fuera otro tipo de motor, debería existir otro tipo de conexionado.

5 DESARROLLO DE INVESTIGATIVO

5.1 Introducción

Comenzando con la investigación, es fundamental saber que componentes son los que se van a utilizar, tanto para el movimiento del robot y reconocimiento de patrones.

5.2 Tecnología Kinect

Para comenzar a investigar sobre una tecnología con Kinect, primero se debe saber que es, que es lo que hace y cuáles son sus restricciones, además si es posible conectar directamente al computador.

Para responder estas inquietudes, Kinect es un dispositivo que permite a los usuarios controlar e interactuar sin necesidad de tener contacto físico con un computador, mediante una interfaz de usuario que puede reconoce gestos, comandos de voz, y objetos e imágenes. Kinect dentro de sus restricciones, posee inconvenientes con la cantidad de personas a identificar, con un límite máximo de 6 personas (Kinect 1.0, para XBOX 360), pero esta cantidad es suficiente para lo que se necesita para el proyecto. Por otro lado, la cámara reconoce usuarios hasta un máximo aproximado de 6 metros y un mínimo de 1,5 metros. El dispositivo, no puede recibir directamente luz, ya que se pierde recepción y calidad de imagen, con lo que se dificulta el reconocimiento de un individuo.

Figura 7

Otro ítem importante a considerar, es que como se uso una Kinect para Xbox360, esta no posee una entrada USB, por lo que se considero la adquisición de un adaptador para poder realizar la conexión.

Figura 8

Ya teniendo en cuenta lo básico, se procede a indagar en lo más técnico. Algunos puntos principales son:

Sensores

- Lentes de color y sensación de profundidad
- Microfono multi-arreglo
- Ajuste de sensor con su motor de inclinación
- Totalmente compatible con las consolas existentes de Xbox 360

Campo de visión

- Campo de visión horizontal: 57 grados
- Campo de visión vertical: 43 grados
- rango de inclinación física: ± 27 grados
- Rango de profundidad del sensor: 1,2 - 6,5 metros

Data Streams (Flujo de datos)

- 320 × 240 a 16 bits de profundidad @ 30fps
- 640 × 480 32-bit de color @30fps
- Audio de 16-bit @ 16 kHz

Sistema de Seguimiento

- Rastrea hasta 6 personas, incluyendo 2 jugadores activos

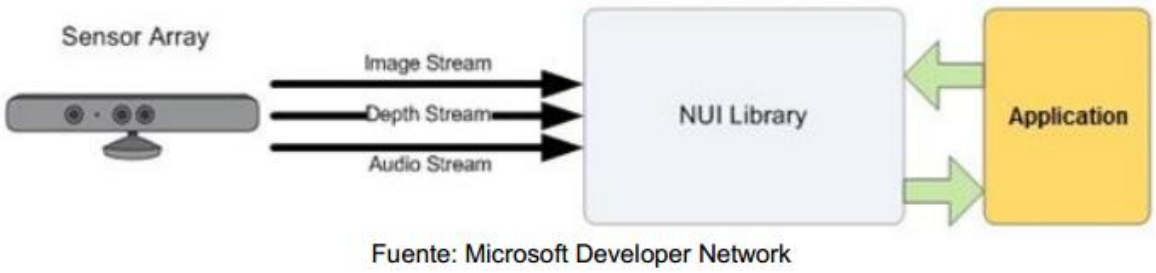
- Rastrea 20 articulaciones por jugador activo

Sistema de audio

- Sistema de cancelación de eco que aumenta la entrada de voz
- Reconocimiento de voz múltiple

Un punto importante para poder manipular Kinect, es Kinect SDK For Windows, la cual proporciona una biblioteca de software sofisticado y herramientas para ayudar a los desarrolladores a utilizar Kinect de una forma natural, detectando acontecimientos del mundo real. Kinect y la biblioteca de software interactuan con la aplicación, como se muestra en la siguiente figura.

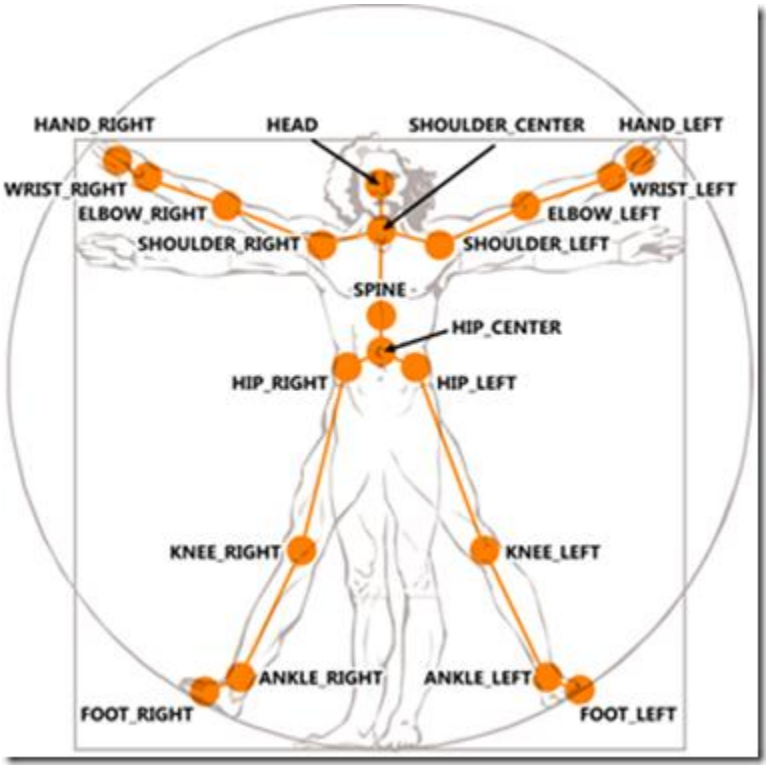
Figura 9



Fuente: Microsoft Developer Network

Para el proyecto, se necesita el reconocimiento del cuerpo humano. La Kinect nos facilita esta opción, ya que al reconocer un cuerpo, genera un esqueleto determinando por puntos, los cuales son articulaciones y extremidades del cuerpo, como se muestra en la siguiente figura.

Figura 10



Por otro lado, Kinect posee un micrófono integrado, el cual en unión con su SDK permite el reconocimiento de palabras. Para poder realizar esto es necesaria la instalación de un paquete de lenguajes, o sea, instalar el idioma que deseas que reconozca.

Para poder comenzar el desarrollo, es fundamental descargar e instalar los siguientes programas:

Figura 11

Kinect for Windows Developer Toolkit 1.8
Kinect for Windows Driver 1.8
Kinect for Windows SDK 1.8
Kinect for Windows Speech Recongnition Language Pack (es-MX) 11.0
Kinect for Windows Runtime 1.8

Que se pueden adquirir desde el siguiente sitio (Junio-2014).
http://www.microsoft.com/en-us/kinectforwindowsdev/Start.aspx

Se necesita descargar Coding4Fun Kinect Toolkit desde la URL

http://c4fkinect.codeplex.com/

Por último, para comenzar la programación, es necesario el IDE Visual Studio, que para este caso se utiliza la versión 2012. Se necesita este software, debido a que la tecnología SDK Kinect es propiedad de Microsoft, lo cual requiere Visual Studio 2010 o superior para su desarrollo.

5.3 Tecnología Arduino

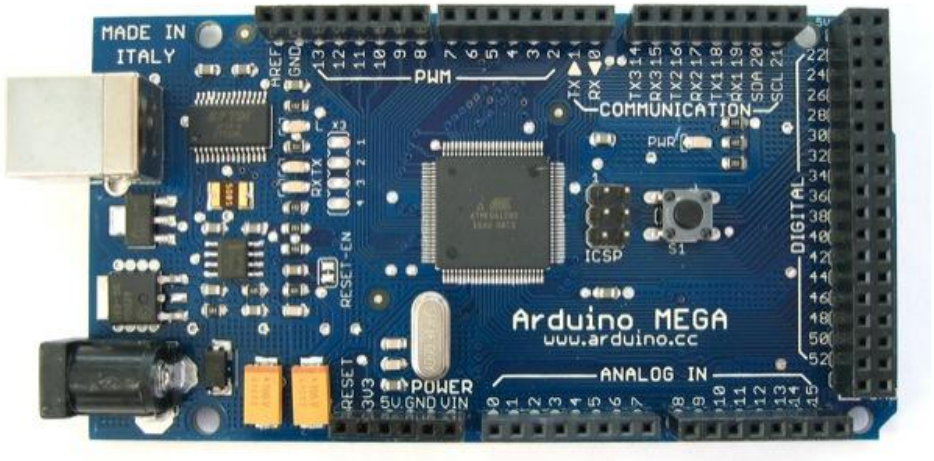
Al igual que el punto anterior, se debe investigar sobre una tecnología con Arduino, ya que se debe saber con que se está trabajando, para que sirve y como es su funcionamiento.
Arduino es una placa con un microcontrolador, la cual facilita el uso de la electrónica, además, una de sus principales cualidades es que es programable a través de un software propio.

Durante el proyecto se utiliza la plataforma Arduino como intermediario entre el computador y el brazo robótico, ya que el software envía señales al Arduino y este las interpreta de manera que pueda mover cada uno de los motores del brazo robótico.
La programación correspondiente a Arduino no posee una alta complejidad, pero para poder hacer uso del software Arduino LLC v1.0.5 el cual permite la programación en Arduino, se debe instalar el driver correspondiente a la placa Arduino MEGA, ya que este tipo de placa es la que se utiliza en el proyecto.
Para descargar el software Arduino LLC v.1.0.5 y el driver para el Arduino MEGA, debe visitar el siguiente sitio.

http://arduino.cc/ (Junio 2014)

Se presenta a continuación una imagen de Arduino MEGA.

Figura 12



5.4 Características de brazo robótico (OWI 535).

Con el afán de poder interpretar movimientos y transformarlos en acciones, es que se ha decido utilizar el brazo robótico modelo OWI 535.

Figura 13

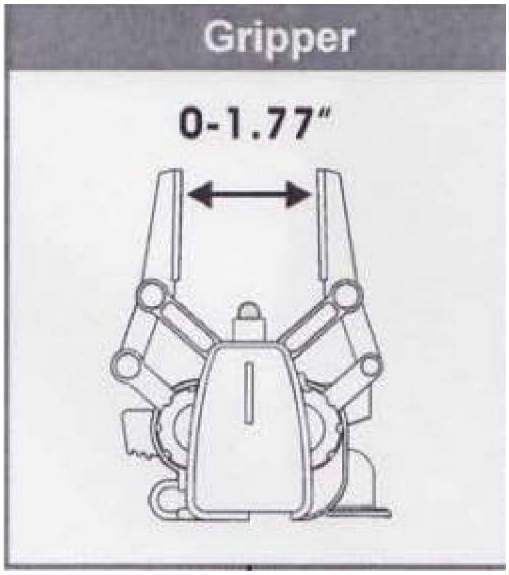
Sus características son:

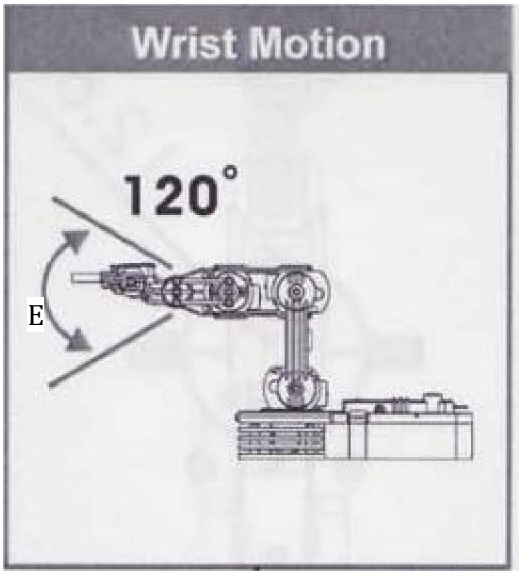
- Máximo levantamiento: 100 g.
- Alcance Horizontal: 12,6".
- Dimensiones: 9" de largo, 6" de ancho y 15" de alto.
- Peso: 658 g.
- Alimentación: 6 Vcc (4 baterías tipo "D").

Los motores y acciones del brazo robótico son las siguientes:

Figuras 14

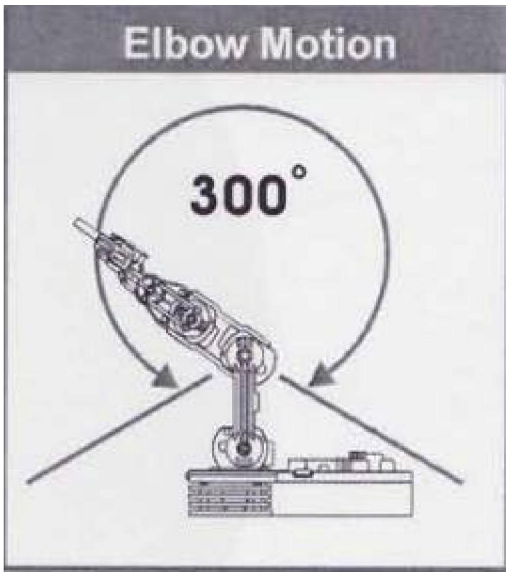
• Abertura máxima de la tenaza: 1,77 pulgadas. Las pinzas para el proyecto corresponden al motor 1 y su función determinada es abrir y cerrar.

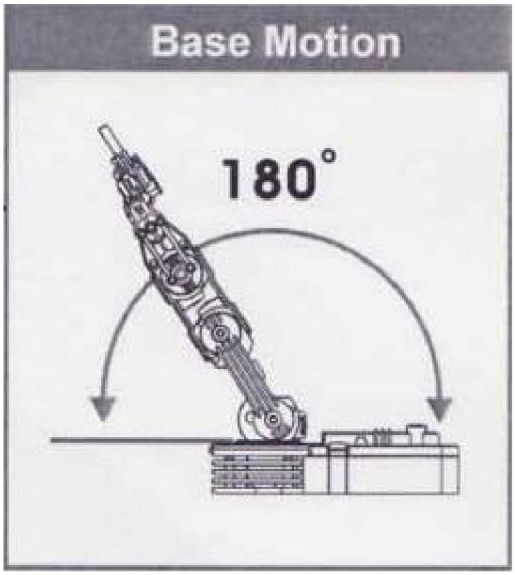




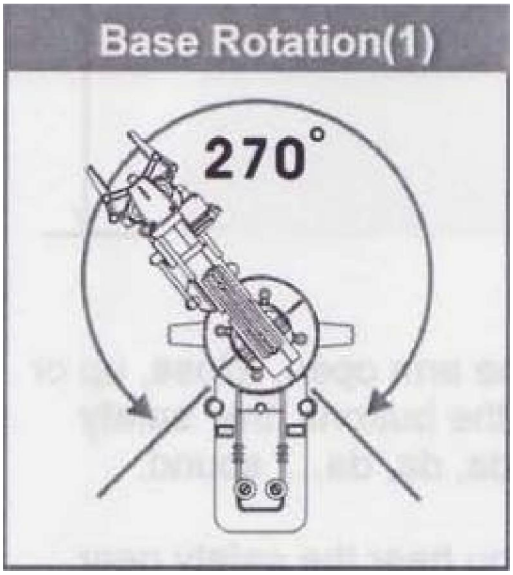
• Movimiento vertical de muñeca: 120°.
Este motor corresponde al número 2 y sus movimientos son arriba y abajo.

- Movimiento vertical del codo: 120°. Este motor corresponde al número 3 y sus movimientos son arriba y abajo.





• Movimiento vertical del hombro: 180°. Este motor corresponde al número 4 y sus movimientos son arriba y abajo.

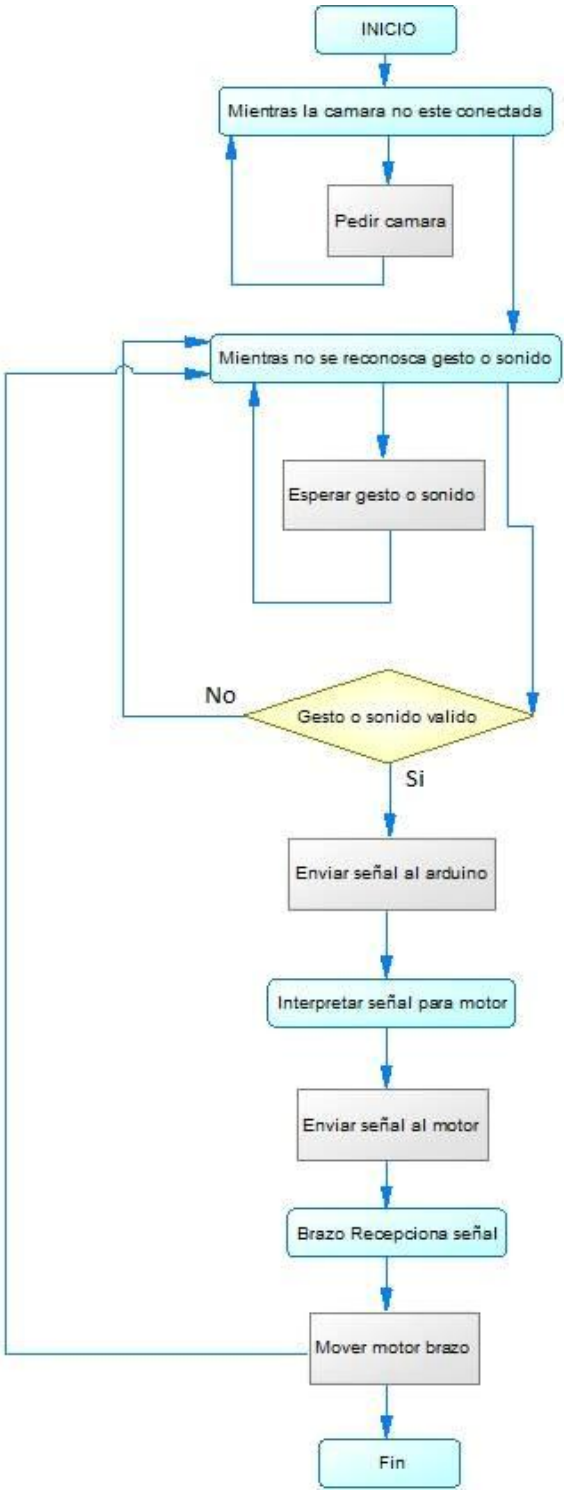


- Movimiento horizontal del hombro: 270°. Este motor corresponde al número 5 y sus movimientos son izquierda y derecha.

5.5 Algoritmo del software

Para el proyecto, se ha analizado distintas estrategias para poder cumplir el objetivo de mover el brazo robótico con movimientos y palabras captados por Kinect. El funcionamiento esta desarrollado de la siguiente manera.

Figura 15



En resumen, el algoritmo consta en detectar la Kinect, si esta está conectada, se hablita la recepción de sonidos o gestos, si el sonido o el gesto es válido, el software envía una señal a Arduino indicado al motor que corresponde y la acción a realizar, cuando el Arduino procesa esa acción, envía una señal al brazo robótico y se realiza el movimiento.

6 DESCRIPCIÓN DE GESTOS, SONIDOS Y SEÑALES

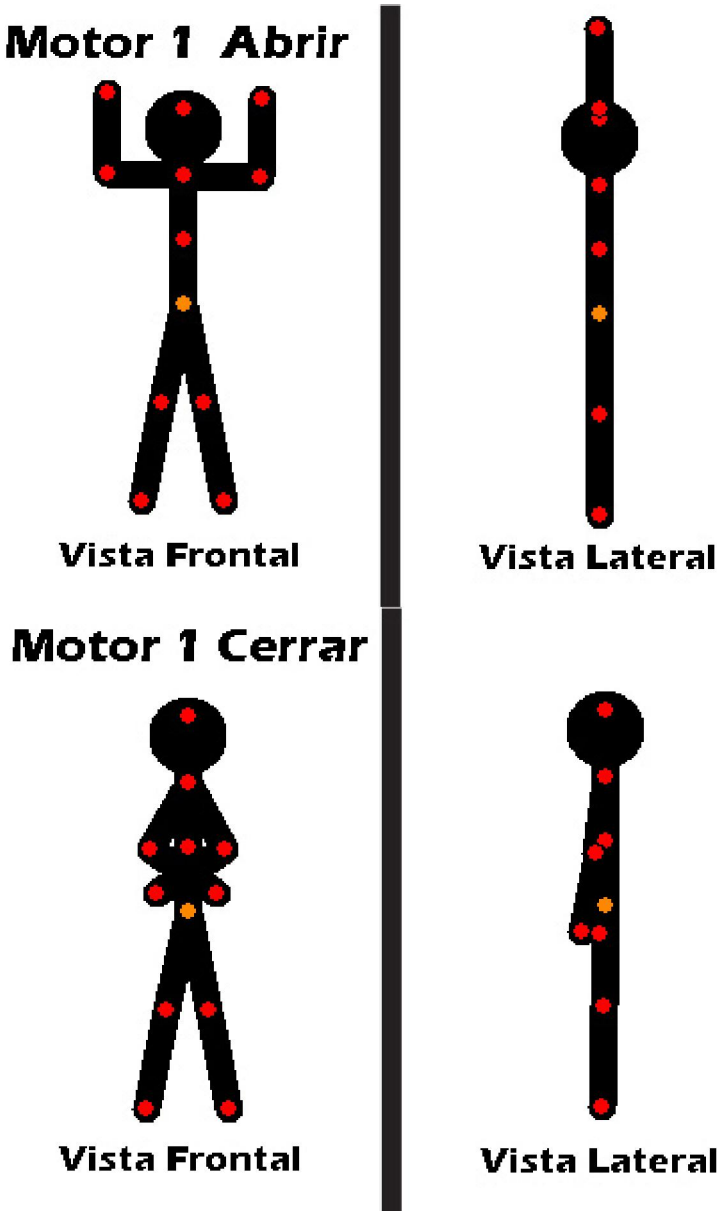
6.1 Descripción de gestos

El software es capaz de detectar patrones, obteniendo datos determinados desde la Kinect. Los gestos están determinados por el grupo desarrollador y el profesor guía.

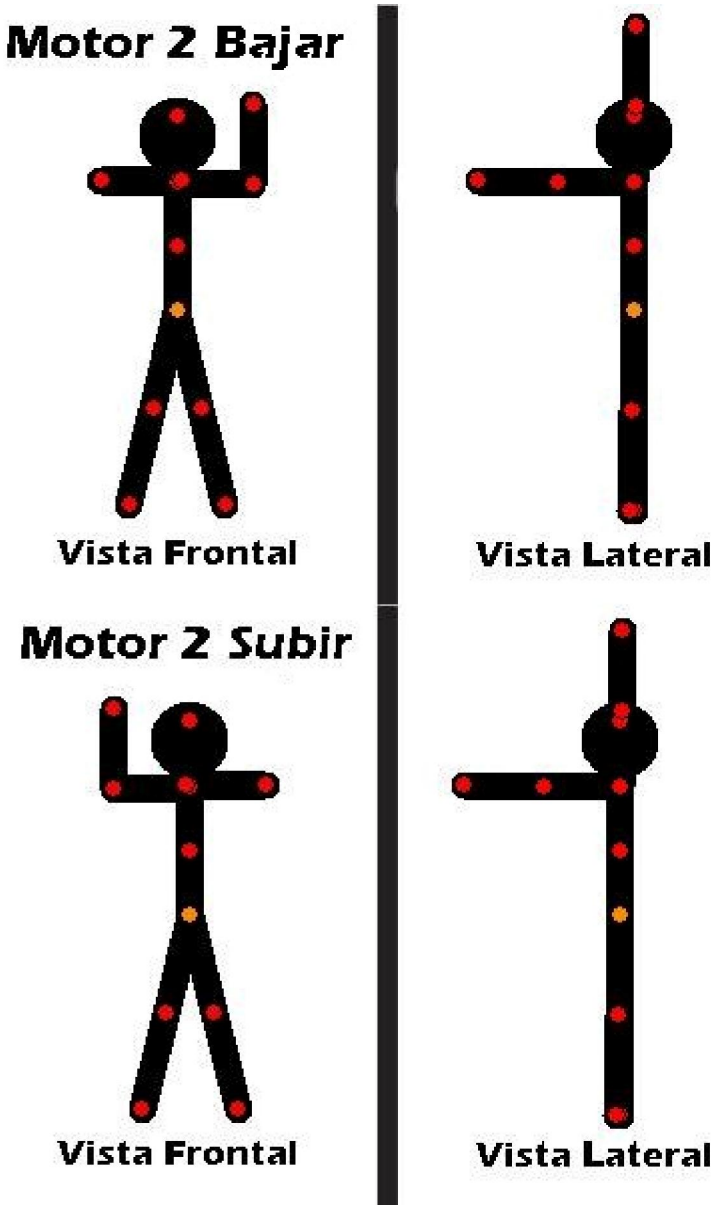
Para el movimiento de cada motor se asigno un gesto específico, los cuales se describen a continuación.

Para el motor 1, los movimientos son los siguientes:

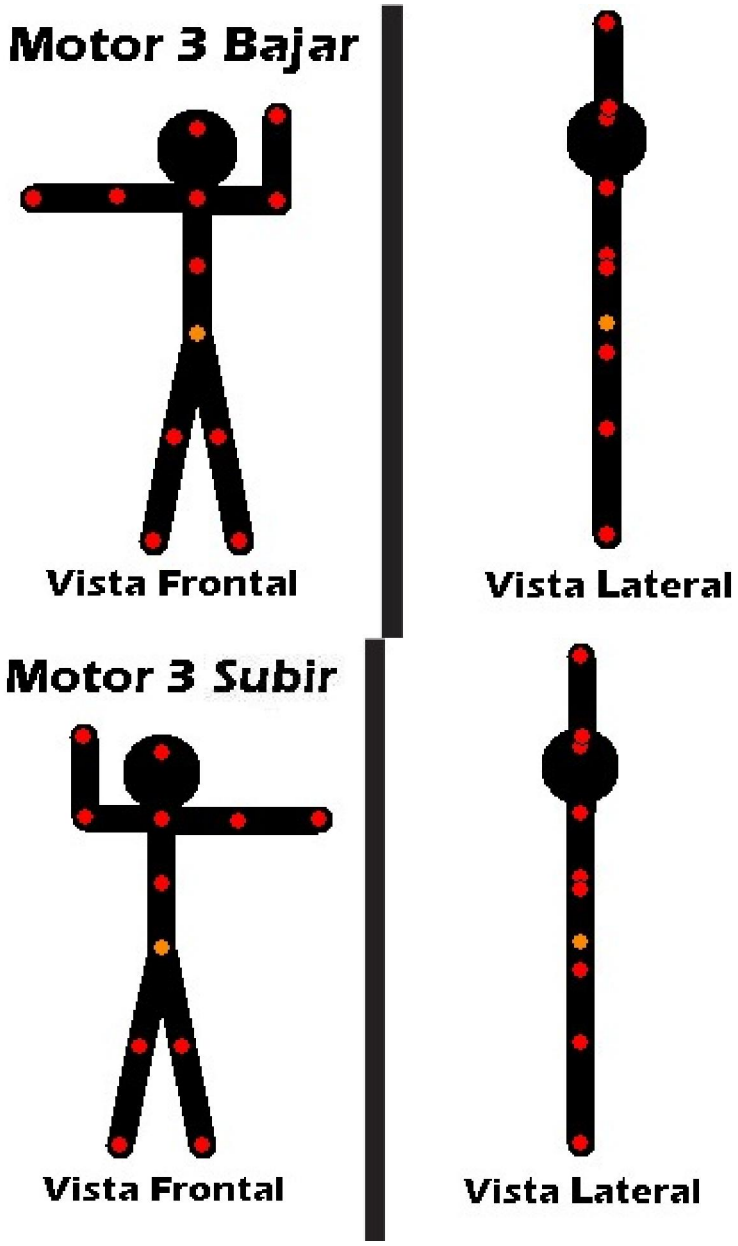
Figura 16



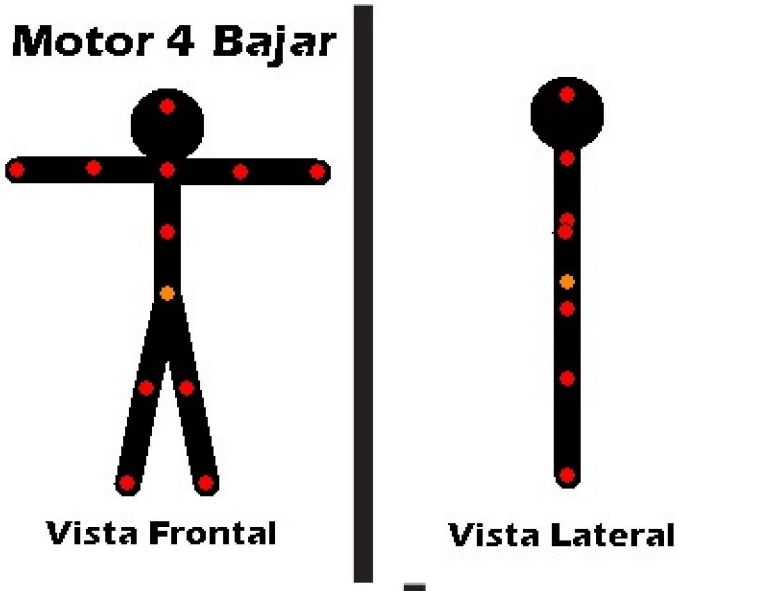
Para el motor 2, los movimientos son los siguientes:

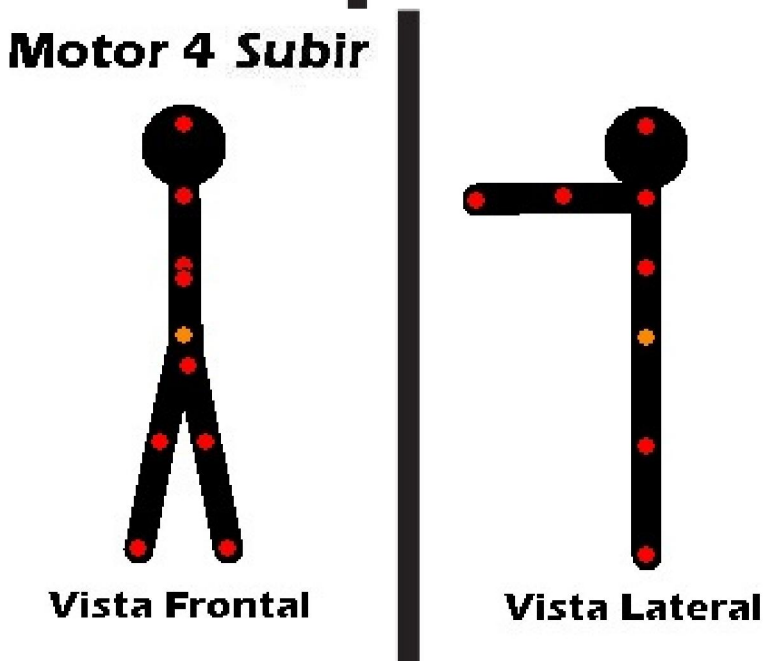


Para el motor 3, los movimientos son los siguientes:

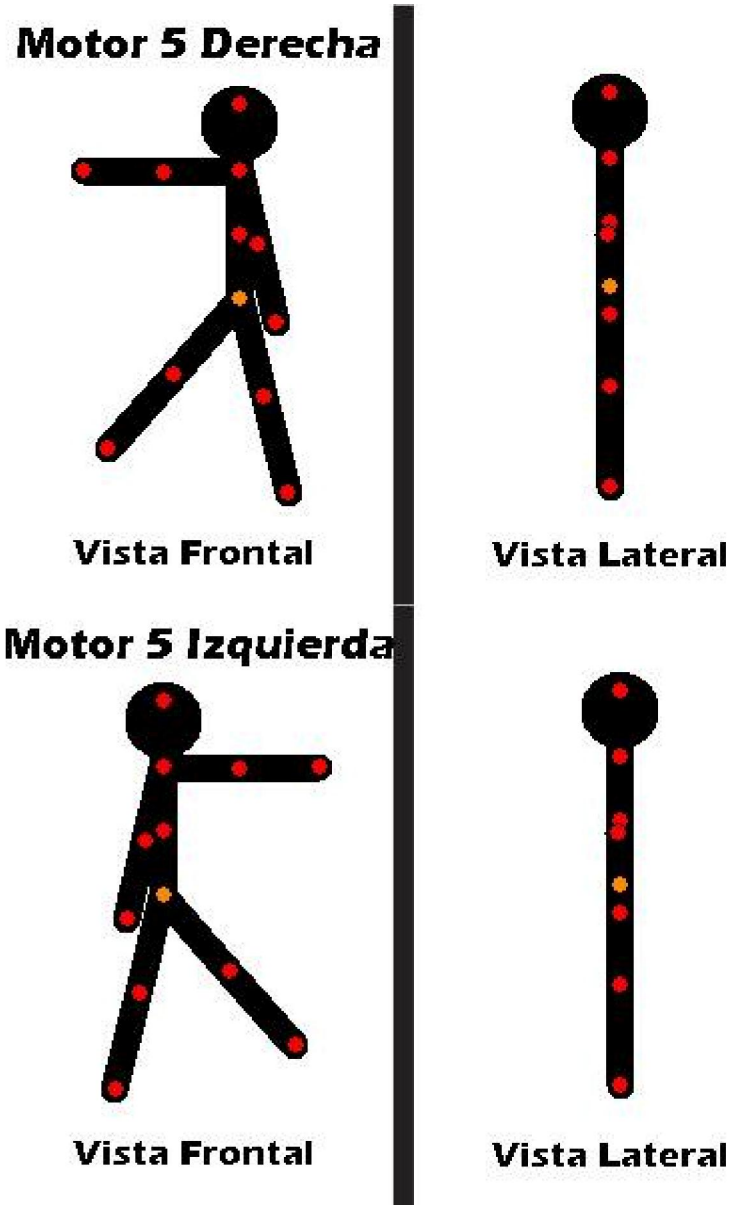


Para el motor 4, los movimientos son los siguientes:





Para el motor 5, los movimientos son los siguientes:



6.2 Descripción de sonidos

A continuación se presentan palabras claves que reconoce el sistema, cada una con una acción predeterminada y algunas necesitan una palabra como precondición para su utilización.

Figura 17

Palabra	Precondición	Acción
Uno	-	Selecciona motor 1
Dos	-	Selecciona motor 2
Tres	-	Selecciona motor 3
Cuatro	-	Selecciona motor 4
Cinco	-	Selecciona motor 5
Abrir	Debe estar seleccionado motor 1	Abre pinzas, motor 1
Cerrar	Debe estar seleccionado motor 1	Cierra pinzas, motor 1
Subir	Debe estar seleccionado motor 2, 3 o 4	Sube brazo, motor 2, 3 o 4 respectivamente
Bajar	Debe estar seleccionado motor 2, 3 o 4	Baja brazo, motor 2, 3 o 4 respectivamente
Derecha	Debe estar seleccionado motor 5	Mueve derecha brazo, motor 5
Izquierda	Debe estar seleccionado motor 5	Mueve izquierda brazo, motor 5
Adiós	-	Vuelve al menú principal

Si se realiza una combinación no valida, el software envía la notificación a través de audio de comando inválido.

6.3 Interpretación de señales

El software envía al Arduino distintas señales, las cuales son reconocidas por él y genera el movimiento de cada uno de los motores. Las señales corresponden a un valor numérico dentro del software, el cual es asociado a cada acción percibida.

A continuación la lista de señales son las siguientes.

Figura 18

Señal	Acción de motor
0	Derecha motor 5
1	Abrir pinzas, motor 1
2	Cerrar pinzas, motor 1
3	Bajar motor 2
4	Subir motor 2
5	Bajar motor 3
6	Subir motor 3
7	Bajar motor 4
8	Subir motor 4
9	Izquierda motor 5

7 DISEÑO PROTOTIPO KINECT

7.1 Definición de movimientos

Para el trabajo que respecta a la detección de gestos, se identificaron 11 puntos del cuerpo humano (extremidades y articulaciones), que permiten la realización de los 10 movimientos permitidos por el sistema.

Los puntos del cuerpo humano son los que muestra la siguiente imagen.

Figura 19

Para cada movimiento se utilizaron los siguientes puntos:

Para motor 1, abrir pinzas: Mano Izquierda, mano derecha y cabeza.

Para motor 1, cerrar pinzas: Mano Izquierda, mano derecha, codo izquierdo y codo derecho.

Para motor 2, subir: Mano Izquierda, mano derecha, codo izquierdo, hombro izquierdo y hombro derecho.
Para motor 2, bajar: Mano Izquierda, mano derecha, codo derecho, hombro izquierdo y hombro derecho.

Para motor 3, subir: Mano Izquierda, mano derecha, codo izquierdo, hombro izquierdo y hombro derecho.

Para motor 3, bajar: Mano Izquierda, mano derecha, codo derecho, hombro izquierdo y hombro derecho.

Para motor 4, subir: Mano Izquierda, mano derecha, codo izquierdo, codo derecho, hombro izquierdo y hombro derecho.

Para motor 4, bajar: Mano Izquierda, mano derecha, codo izquierdo, codo derecho, hombro izquierdo y hombro derecho.

Para motor 5, izquierda: Mano izquierda, codo izquierda, hombro izquierda, rodilla derecha y pie izquierda.

Para motor 5, derecha: Mano derecha, codo derecho, hombro derecho, rodilla derecha y pie derecho.

Los valores entregados por la SDK de Kinect referidos al cuerpo humano son en las coordenadas de los ejes X,Y,Z, por lo tanto, los movimientos fueron definidos según se acomodan a estos ejes. Para que los movimientos no fueran tan estrictos es que se dio un margen de holgura para mejorar su uso.

7.2 Diseño de prototipo de gestos

Teniendo clara la idea, es que se comienza a desarrollar código.
A continuación se presenta la declaración y asignación de movimientos

```
Joint hombroI;
Joint hombroD;
Joint manoI;
Joint manoD;
Joint cabeza;
Joint pieI;
Joint pieD;
Joint rodillaD;
Joint rodillaI;
Joint codoD;
Joint codoI;
```

Cada Joint es punto que se utiliza para determinar un gesto.

```
hombroI = skeleton.Joints[JointType.ShoulderLeft];
hombroD = skeleton.Joints[JointType.ShoulderRight];
manoI = skeleton.Joints[JointType.HandLeft];
manoD = skeleton.Joints[JointType.HandRight];
cabeza = skeleton.Joints[JointType.Head];
pieI = skeleton.Joints[JointType.FootLeft];
pieD = skeleton.Joints[JointType.FootRight];
rodillaD = skeleton.Joints[JointType.KneeRight];
rodillaI = skeleton.Joints[JointType.KneeLeft];
codoD = skeleton.Joints[JointType.ElbowRight];
codoI = skeleton.Joints[JointType.ElbowLeft];
```

Kinect detecta personas que interactúan con ella, en este caso se detecta la persona más cerca a la Kinect y a ella se le obtienen los puntos anteriormente determinados. La variable "Skeleton" es la variable que corresponde a la persona que la Kinect reconoce.

Lo más importante es invocar la referencia a Kinect, ya que sin esto, no se podría utilizar funciones o variables propias de Kinect.

```
using System.IO.Ports;
using Microsoft.Kinect;
```

Además, se está incluyendo la referencia al uso de puertos, por donde será enviada la señal al Arduino para que pueda gestionar el movimiento del brazo robótico.

```
if (bandera == true)
{
    //m1 abrir
    if (manoI.Position.Y > cabeza.Position.Y && manoD.Position.Y > cabeza.Position.Y &&
        manoI.Position.X < cabeza.Position.X && manoD.Position.X > cabeza.Position.X)
    {
        mov.Content = "m1 abrir";
        _port.Write("1");
        bandera = false;
    }
    else
    {
        if (manoD.Position.X < manoI.Position.X && codoD.Position.Y > manoD.Position.Y && codoI.Position.Y > manoI.Position.Y)
        {
            mov.Content = "m1 cerrar";
            _port.Write("2");
            bandera = false;
        }
        else
        {
            if (manoI.Position.Y > hombroI.Position.Y && Math.Abs(manoI.Position.Z - hombroI.Position.Z) < 0.2 &&
                Math.Abs(codoD.Position.Z - manoD.Position.Z) > 0.25 && Math.Abs(hombroD.Position.X - manoD.Position.X) < 0.1)
            {
                mov.Content = "m2 bajar";
                _port.Write("3");
                bandera = false;
            }

else
{
    if (manoD.Position.Y > hombroD.Position.Y && Math.Abs(manoD.Position.Z - hombroD.Position.Z) < 0.2 &&
    Math.Abs(codoI.Position.Z - manoI.Position.Z) > 0.25 && Math.Abs(hombroI.Position.X - manoI.Position.X) < 0.1)
    {
        mov.Content = "m2 subir";
        _port.Write("4");
        bandera = false;
    }
    else
    {
        if (Math.Abs(manoI.Position.X - codoI.Position.X) > 0.25 && Math.Abs(manoI.Position.Z - hombroI.Position.Z) < 0.1
            && Math.Abs(manoI.Position.Y - hombroI.Position.Y) < 0.1 &&
            manoD.Position.Y > hombroD.Position.Y+0.1 && Math.Abs(manoD.Position.Z - hombroD.Position.Z) < 0.2)
        {
            mov.Content = "m3 subir ";
            _port.Write("6");
            bandera = false;
        }
        else
        {
            if (Math.Abs(manoD.Position.X - codoD.Position.X) > 0.25 && Math.Abs(manoD.Position.Z - hombroD.Position.Z) <
            && Math.Abs(manoD.Position.Y - hombroD.Position.Y) < 0.1 &&
            manoI.Position.Y > hombroI.Position.Y + 0.1 && Math.Abs(manoI.Position.Z - hombroI.Position.Z) < 0.2)
            {
                mov.Content = "m3 bajar";
                _port.Write("5");
                bandera = false;
            }
            else
```

```
{
    if (Math.Abs(manoD.Position.X - codoD.Position.X) > 0.2 && Math.Abs(manoD.Position.Z - hombroD.Position.Z) < 0.1
        && Math.Abs(hombroD.Position.Y - manoD.Position.Y) < 0.1
        && Math.Abs(manoI.Position.X - codoI.Position.X) > 0.2 && Math.Abs(manoI.Position.Z - hombroI.Position.Z) < 0.1
        && Math.Abs(hombroI.Position.Y - manoI.Position.Y) < 0.1)
    {
        mov.Content = "m4 bajar";
        _port.Write("7");
        bandera = false;
    }
    else
    {
        if (Math.Abs(codoI.Position.Z - manoI.Position.Z) > 0.2 && Math.Abs(hombroI.Position.X - manoI.Position.X) < 0.1
            && Math.Abs(hombroI.Position.Y - manoI.Position.Y) < 0.1 &&
            Math.Abs(codoD.Position.Z - manoD.Position.Z) > 0.2 && Math.Abs(hombroD.Position.X - manoD.Position.X) < 0.1
            && Math.Abs(hombroD.Position.Y - manoD.Position.Y) < 0.1)
        {
            mov.Content = "m4 subir";
            _port.Write("8");
            bandera = false;
        }
        else
        {
            if (Math.Abs(manoD.Position.X - codoD.Position.X) > 0.25 && Math.Abs(manoD.Position.Z - hombroD.Position.Z) < 0.1
                && Math.Abs(manoD.Position.Y - hombroD.Position.Y) < 0.1
                && Math.Abs(pieD.Position.X - rodillaD.Position.X) > 0.1 )
             {
              mov.Content = "m5 derecha";
              _port.Write("0");
                bandera = false;
             }
                        if (Math.Abs(manoI.Position.X - codoI.Position.X) > 0.25 && Math.Abs(manoI.Position.Z - hombroI.Position.Z) < 0.1
                            && Math.Abs(manoI.Position.Y - hombroI.Position.Y) < 0.1
                            && pieI.Position.X - rodillaI.Position.X < -0.1)
                        {
                        mov.Content = "m5 izquierda";
                        _port.Write("9");
                        bandera = false;
                        }
                    }
                }
            }
        }
    }

}
```

Esto corresponde a la definición de cada gesto, con sus restricciones anteriormente mencionadas y con sus holguras respectivas, determinadas por los desarrolladores.
Su usó una variable llamada “bandera”, la cual permite ingresar cada 0.5 segundos a la función que detecta gestos, ya que no se contaba con computadores con la suficiente potencia para ejecutarlo en tiempo continuo.
Además, por cada gesto detectado correctamente, el software envía dos señales, una a la pantalla indicando el movimiento y el motor que lo realiza, la otra es enviada por puerto serial al Arduino.

NOTA: Cabe recalcar que el código para conectar la Kinect fue obtenido de los ejemplos de la SDK de Kinect.

7.3 Diseño de prototipo de voz

Para el prototipo de voz, se fueron probando distintas palabras, primero en ingles, luego en español. Ya que el programa permite este idioma (español latinoamericano) se decidió utilizar frases. Las frases en primera instancia eran una buena idea, pero estas al contener demasiadas palabras se confundían con otras no logrando lo que se esperaba. Por lo tanto solo se cambio a palabras simples, en donde se representan el motor a mover y la dirección de cada uno de estos.

Además para cada opción se designa un sonido predeterminado que se escuchara cuando se pronuncie.

```
void speechengine_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
    System.Media.SoundPlayer m1, m2, m3, m4, m5, abrir, cerrar, subm2, bajm2, subm3, bajm3,
    subm4, bajm4, derecha, izquierda, error, adios;
    m1 = new System.Media.SoundPlayer("../../Sounds/motor1.wav");
    m2 = new System.Media.SoundPlayer("../../Sounds/motor2.wav");
    m3 = new System.Media.SoundPlayer("../../Sounds/motor3.wav");
    m4 = new System.Media.SoundPlayer("../../Sounds/motor4.wav");
    m5 = new System.Media.SoundPlayer("../../Sounds/motor5.wav");
    abrir = new System.Media.SoundPlayer("../../Sounds/abrir pinzas.wav");
    cerrar = new System.Media.SoundPlayer("../../Sounds/cerrar pinzas.wav");
    subm2 = new System.Media.SoundPlayer("../../Sounds/subm2.wav");
    bajm2 = new System.Media.SoundPlayer("../../Sounds/bajm2.wav");
    subm3 = new System.Media.SoundPlayer("../../Sounds/subm3.wav");
    bajm3 = new System.Media.SoundPlayer("../../Sounds/bajm3.wav");
    subm4 = new System.Media.SoundPlayer("../../Sounds/subm4.wav");
    bajm4 = new System.Media.SoundPlayer("../../Sounds/bajm4.wav");
    izquierda = new System.Media.SoundPlayer("../../Sounds/izqm5.wav");
    derecha = new System.Media.SoundPlayer("../../Sounds/derm5.wav");
    error = new System.Media.SoundPlayer("../../Sounds/error.wav");
    adios = new System.Media.SoundPlayer("../../Sounds/adios.wav");
```

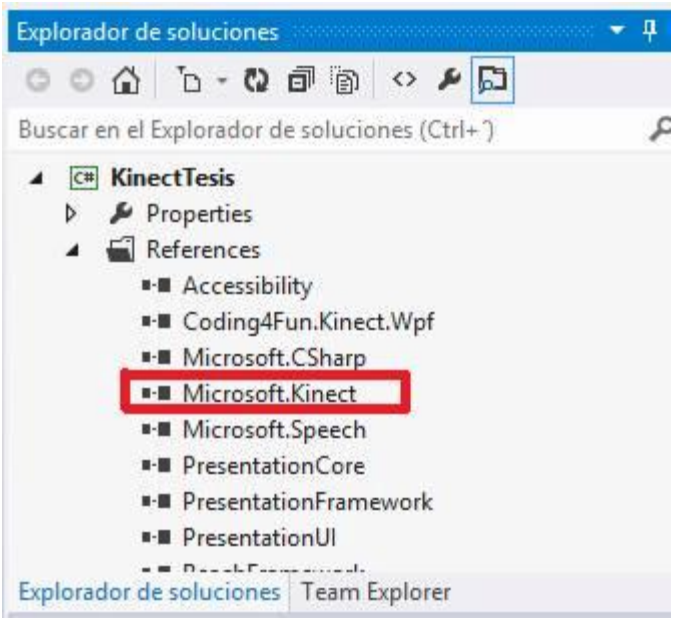
Para el reconocimiento de voz, se designaron las siguientes palabras:

```
opciones.Add("uno", "UNO");
opciones.Add("dos", "DOS");
opciones.Add("tres", "TRES");
opciones.Add("cuatro", "CUATRO");
opciones.Add("cinco", "CINCO");
opciones.Add("derecha", "DERECHA");
opciones.Add("izquierda", "IZQUIERDA");
opciones.Add("subir", "SUBIR");
opciones.Add("bajar", "BAJAR");
opciones.Add("abrir", "ABRIR");
opciones.Add("cerrar", "CERRAR");
opciones.Add("adios", "ADIOS");
```

Cada opción representa lo que se escucha (en minúsculas), con la opción en el switch (que es lo que realiza).

Para comenzar a utilizar la función de audio de Kinect, es necesario primero cargar una referencia llamada "Microsoft.Speech" la cual se encuentra en el directorio C:\Program Files\Microsoft SDKs\Speech\v11.0\Assembly (una vez se haya instalado el SDK de Kinect)

Figura 20



Una vez se haya cargado la referencia es necesario llamar a la librería de Kinect y a 2 librerías propias para el reconocimiento de voz.

```
using Microsoft.Kinect;
using Microsoft.Speech.AudioFormat;
using Microsoft.Speech.Recognition;
```

La librería "Microsoft.Speech.AudioFormat" permite al software conocer la información acerca del sonido o palabra que está entrando por micrófono, mientras "Microsoft.Speech.Recognition" crea gramáticas las cuales serán comparadas para el reconocimiento de voz.

A continuación se muestra lo que hace cada palabra definida:

```
switch (e.Result.Words[0].Text)
{
    //En caso de que digamos "uno" la llave "UNO" se abrira y se realizara lo siguiente
    case "UNO":
        //Se mandara un mensaje alusivo a la imagen
        if (flag != 1)
        {
            mensaje.Text = "Motor 1 Listo";
            mensaje.Background = new SolidColorBrush(Color.FromRgb(247, 126, 5));



            src = new Uri(@"/Images/motor1.png", UriKind.Relative);
            img = new BitmapImage(src);
            flag = 1;
            imagen.Source = img;
            m1.Play();
            Thread.Sleep(1800);
        }

        break;

 case "DOS":
     if (flag != 2)
     {
         mensaje.Text = "Motor 2 Listo";
         src = new Uri(@"/Images/motor2.png", UriKind.Relative);
         img = new BitmapImage(src);
         imagen.Source = img;
         mensaje.Background = new SolidColorBrush(Color.FromRgb(255, 0, 0));
         flag = 2;
         m2.Play();
         Thread.Sleep(1400);
     }
     break;

 case "TRES":
     if (flag != 3)
     {
         mensaje.Text = "Motor 3 Listo";
         src = new Uri(@"/Images/motor3.png", UriKind.Relative);
         img = new BitmapImage(src);
         imagen.Source = img;
         mensaje.Background = new SolidColorBrush(Color.FromRgb(5, 134, 247));
         flag = 3;
         m3.Play();
         Thread.Sleep(1400);
     }
     break;
```

```
case "CUATRO":
    if (flag != 4)
    {
        mensaje.Text = "Motor 4 Listo";
        src = new Uri(@"/Images/motor4.png", UriKind.Relative);
        img = new BitmapImage(src);
        imagen.Source = img;
        mensaje.Background = new SolidColorBrush(Color.FromRgb(12, 247, 73));
        flag = 4;
        m4.Play();
        Thread.Sleep(2000);
    }
    break;
case "CINCO":
    if (flag != 5)
    {
        mensaje.Text = "Motor 5 Listo";
        src = new Uri(@"/Images/motor5.png", UriKind.Relative);
        img = new BitmapImage(src);
        imagen.Source = img;
        mensaje.Background = new SolidColorBrush(Color.FromRgb(217, 12, 247));
        flag = 5;
        m5.Play();
        Thread.Sleep(2000);
    }
    break;
case "SUBIR":
    if (flag == 2 || flag == 3 || flag == 4)
    {
        if (flag == 2)
        {
            mensaje.Text = "Subiendo motor 2";
            _port.Write("3"); _port.Write("3");
            subm2.Play();
        }
        if (flag == 3)
        {
            mensaje.Text = "Subiendo motor 3";
            _port.Write("5"); _port.Write("5");
            subm3.Play();
        }
        if (flag == 4)
        {
            mensaje.Text = "Subiendo motor 4";
            _port.Write("7"); _port.Write("7");
            subm4.Play();
        }

        Thread.Sleep(2000);
    }
```

```
        else
        {
            mensaje.Text = "Combinación no válida";
            error.Play();
        }
        break;
    case "BAJAR":
        if (flag == 2 || flag == 3 || flag == 4)
        {
            if (flag == 2)
            {
                mensaje.Text = "Bajando motor 2";
                _port.Write("4"); _port.Write("4");
                bajm2.Play();
            }
            if (flag == 3)
            {
                mensaje.Text = "Bajando motor 3";
                _port.Write("6"); _port.Write("6");
                bajm3.Play();
            }
            if (flag == 4)
            {
                mensaje.Text = "Bajando motor 4";
                _port.Write("8"); _port.Write("8");
                bajm4.Play();
            }

            Thread.Sleep(2000);
        }
        else
        {
            mensaje.Text = "Combinación no válida";
            error.Play();
        }
        break;
    case "IZQUIERDA":
        if (flag == 5)
        {
            mensaje.Text = "Motor 5 hacia la izquierda";
            _port.Write("0"); _port.Write("0");
            izquierda.Play();
            Thread.Sleep(2000);
        }
        else
        {
            mensaje.Text = "Combinación no válida";
            error.Play();
        }
        break;
    case "DERECHA":
        if (flag == 5)
        {
```

```
            mensaje.Text = "Motor 5 hacia la izquierda";
            _port.Write("0"); _port.Write("0");
            izquierda.Play();
            Thread.Sleep(2000);
        }
        else
        {
            mensaje.Text = "Combinación no válida";
            error.Play();
        }
        break;
    case "DERECHA":
        if (flag == 5)
        {
            mensaje.Text = "Motor 5 hacia la derecha";
            _port.Write("9"); _port.Write("9");
            derecha.Play();
            Thread.Sleep(2000);
        }
        else
        {
            mensaje.Text = "Combinación no válida";
            error.Play();
        }
        break;
    case "ABRIR":
      if (flag == 1)
      {
          mensaje.Text = "Abriendo pinzas";
          _port.Write("1");
          _port.Write("1");
          abrir.Play();
          Thread.Sleep(2000);
      }
      else
      {
          mensaje.Text = "Combinación no válida";
          error.Play();
      }
      break;
    se "CERRAR":
      if (flag == 1)
      {
          mensaje.Text = "Cerrando pinzas";
          _port.Write("2");
          _port.Write("2");
          cerrar.Play();
          Thread.Sleep(2000);
      }
      else
      {
```

```
                    mensaje.Text = "Combinación no válida";
                    error.Play();
                }
                break;
            case "ADIOS":

                mensaje.Text = "Cerrando modulo";
                adios.Play();
                Thread.Sleep(520);
                (Application.Current.MainWindow.FindName("_mainFrame") as Frame).
                    Source = new Uri("Inicio.xaml", UriKind.Relative);

                break;


            default:
                //En caso de que no solo contenga una palabra tambien realizaremos
                //un switch para ver si la frase corresponde a alguna de nuestros valores de opcion

                mensaje.Text = "No se reconocio el comando";

                break;

        }
    }
}
}
}
```

Cada opción de selección de motor(desde uno a cinco), realiza un cambio de imagen de fondo, se envía un mensaje por pantalla , además de un mensaje por voz, también se levanta una "bandera" la que permite según su valor realizar el movimiento esperado por cada motor(subir, bajar, abrir, cerrar, izquierda y derecha).
Para las opciones de movimiento se envía un mensaje por pantalla, también una señal al Arduino y luego un mensaje por voz.

7.4 Diseño de modulo de calibración

Para la calibración de cámara, ya que esta consta con un motor integrado, se decidió dejar definido ciertos ángulos para dejar una mejor vista del usuario. Estos cambios de ángulos son activados por botones.

```
private void InitializeButtons()
{
    buttons = new List<Button> { BACKHOME, grado1, grado2, grado3, grado4, grado5, grado6, grado7 };
}
```

Al presionar un botón se registran dos actividades, la primera corresponde en mostrar por pantalla el ángulo que se ha presionado, el segundo corresponde a cambiar el ángulo físico de la Kinect.

```
#region eventos cambio de grado
private void RadioButton_Checked(object sender, RoutedEventArgs e)
{
    try
    {
        ValorAngulo.Content = 25 + "°";
        _Kinect.ElevationAngle = 25;

    }
    catch (Exception)
    {

    }
}

private void RadioButton_Checked_1(object sender, RoutedEventArgs e)
{
    try
    {
        ValorAngulo.Content = 20 + "°";
        _Kinect.ElevationAngle = 20;
    }
    catch (Exception)
    {

    }
}
```

```
private void RadioButton_Checked_2(object sender, RoutedEventArgs e)
{
    try
    {
        ValorAngulo.Content = 10 + "°";
        _Kinect.ElevationAngle = 10;

    }
    catch (Exception)
    {

    }
}

private void RadioButton_Checked_3(object sender, RoutedEventArgs e)
{
    try
    {
        ValorAngulo.Content = 0 + "°";
        _Kinect.ElevationAngle = 0;

    }
    catch (Exception)
    {

    }
}
```

```
private void RadioButton_Checked_4(object sender, RoutedEventArgs e)
{
    try
    {
        ValorAngulo.Content = -10 + "°";
        _Kinect.ElevationAngle = -10;

    }
    catch (Exception)
    {

    }
}

private void RadioButton_Checked_5(object sender, RoutedEventArgs e)
{
    try
    {
        ValorAngulo.Content = -20 + "°";
        _Kinect.ElevationAngle = -20;

    }
    catch (Exception)
    {

    }
}
 private void RadioButton_Checked_6(object sender, RoutedEventArgs e)
 {
     try
     {
         ValorAngulo.Content = -25 + "°";
         _Kinect.ElevationAngle = -25;

     }
     catch (Exception)
     {

     }
 }
```

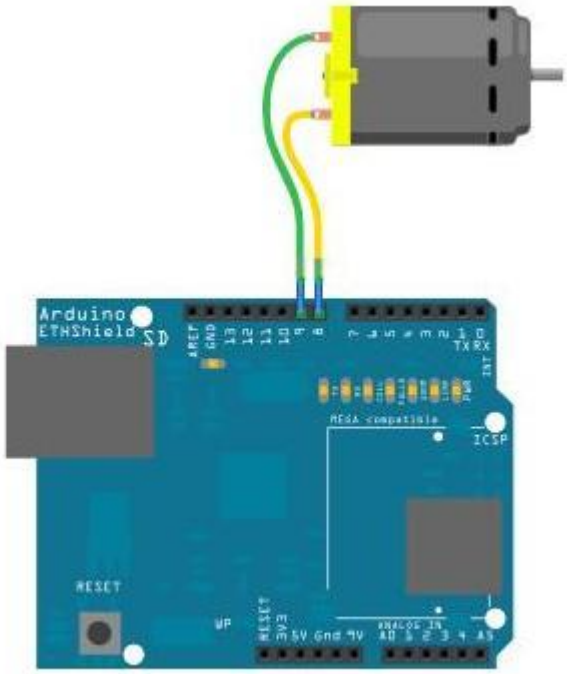
8 DISEÑO DE PROTOTIPO ARDUINO

8.1 Primer prototipo

Como idea general, ya que se conocía el funcionamiento y programación básica de Arduino, se pensó en conectar directamente el motor a una salida, enviando una señal alta (5 Volt) y conectando el otro borne a tierra (GND). Tras este proceso, el motor no realizo ningún movimiento, para lo cual se debió investigar de manera más profunda de cómo realizar el conexionado entre Arduino y un motor DC.

A continuación se presenta una imagen de lo que se realizo.

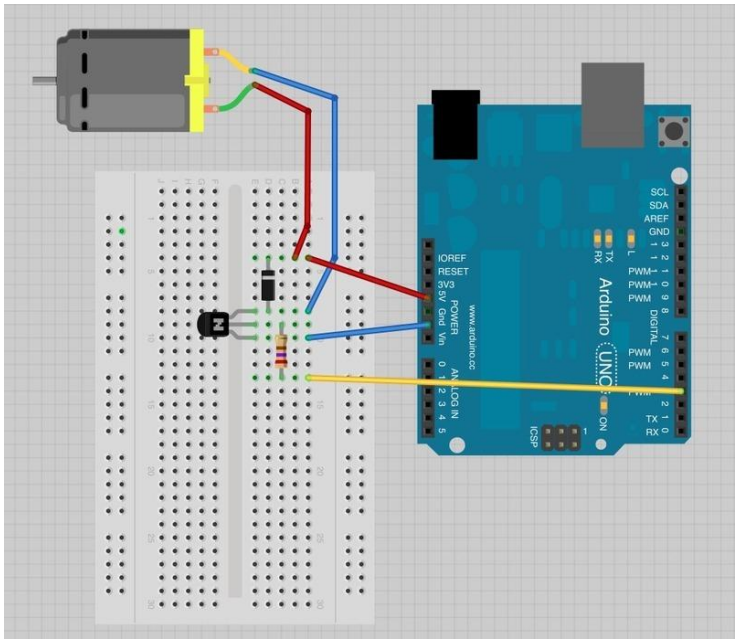
Figura 21



8.2 Segundo prototipo

Estudiando un poco sobre los motores DC y Arduino, es que se pudo dar cuenta de que el modelo anterior no funcionaba debido a que el motor necesitaba mayor corriente. Es por esto que se busco una forma de canalizar toda esa corriente proporcionada por el Arduino directamente al motor, entonces en la guía básica de Arduino se encontró un circuito el cual constaba de un transistor, una resistencia y un diodo, además del motor. Al realizar las pruebas se pudo mover el motor como se estimaba, pero surgió la siguiente incertidumbre. ¿Cómo podemos mover el mismo motor en ambos sentidos?
El segundo prototipo quedo realizado como se muestra en la imagen.

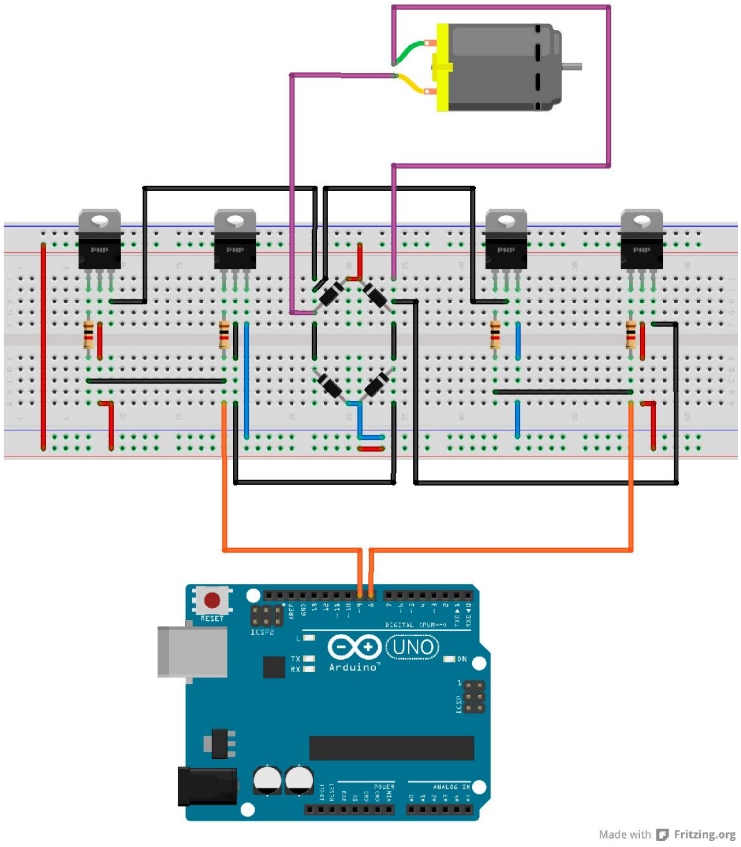
Figura 22



8.3 Tercer prototipo

Luego de investigar, se encontró una solución al problema que nos afectaba, es realizar un "puente H" para el motor, el cual permite cambiar las polaridades que llegan a los bornes del motor, pudiendo así, que el motor gire en ambos sentidos.

Figura 23

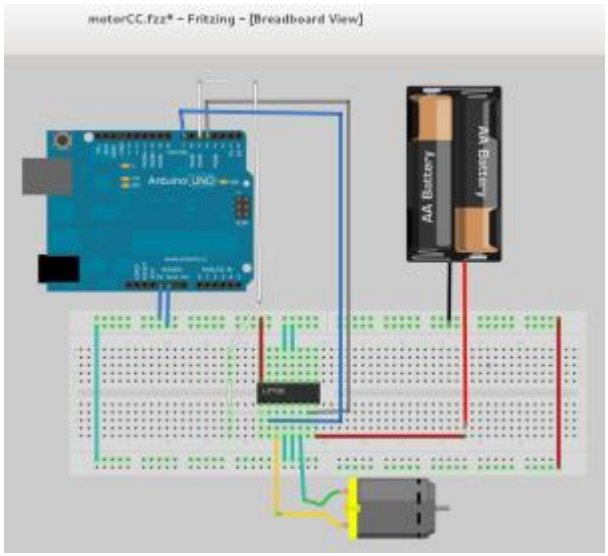


Para realización del circuito de "Puente H", se necesitan mucho materiales y espacio en la protoboard para la conexión de los 5 motores, por lo tanto esta opción quedo desechada.

Siguiendo con la investigación, se descubrió la existencia de un circuito integrado que cumple la misma función de un "puente H", pero en menor tamaño. Es por esto que se eligió esta opción para llevar a cabo el proyecto.

A continuación se presenta el circuito integrado L293D, funcionando en una placa Arduino.

Figura 24

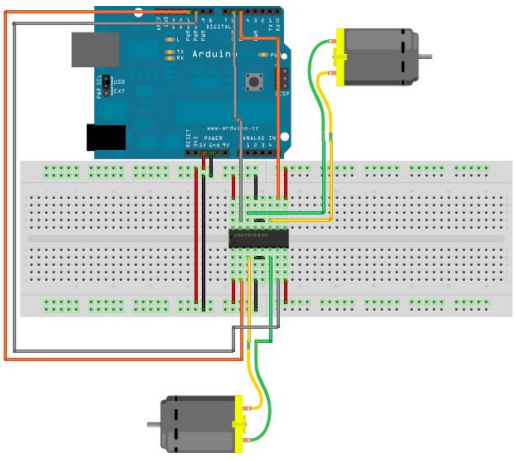


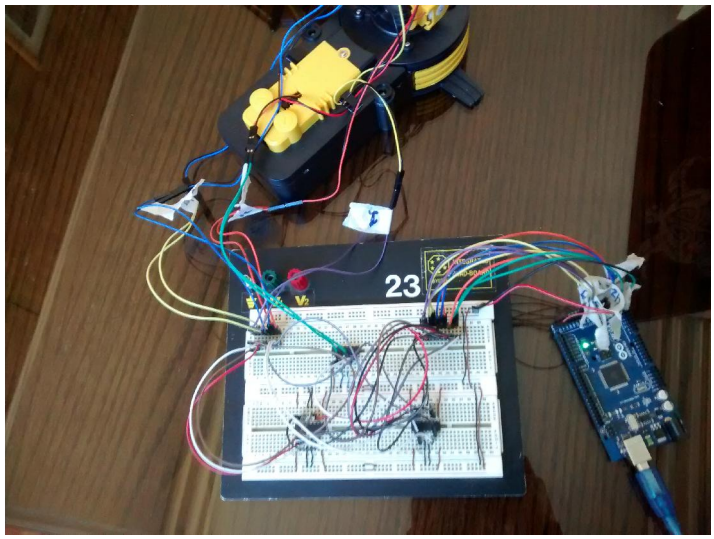
8.4 Cuarto prototipo

Como el tercer prototipo fue exitoso, es que se comenzó a implementar la totalidad del circuito a base del integrado L293D, cambiando incluso su alimentación externa a una interna (pilas a 5 Volt del Arduino).

La siguiente imagen muestra como queda conectado completamente un integrado con dos motores DC.

Figura 25





Una vez el circuito terminado y comprobando su funcionalidad es que se programa la totalidad de las funciones de la placa.

Primero se definen los pines de salida que se ocupan para enviar las señales al brazo robótico (son señales digitales).

Luego, el Arduino queda en espera que el software envíe una señal, en este caso, hace la comparación del valor y activa el motor correspondiente, manteniendo esta señal con un tiempo de duración de medio segundo (0,5 segundos).

```
int salida[10]={34,35,38,39,42,43,46,47,50,51};
int i;
int valor;

void setup()
{
  for(i=0;i<10;i++)
  {
    pinMode(salida[i],OUTPUT);
  }
  Serial.begin(9600);
}


void loop()
{
while(Serial.available()==0);

valor=Serial.read();
  switch(valor)
  {
    case '1':
      digitalWrite(salida[0],HIGH);
      delay(500);
      digitalWrite(salida[0],LOW);
      break;
```

```
case '2':
  digitalWrite(salida[1],HIGH);
  delay(500);
  digitalWrite(salida[1],LOW);
  break;


case '3':
  digitalWrite(salida[2],HIGH);
  delay(500);
  digitalWrite(salida[2],LOW);
  break;


case '4':
  digitalWrite(salida[3],HIGH);
  delay(500);
  digitalWrite(salida[3],LOW);
  break;


case '5':
  digitalWrite(salida[4],HIGH);
  delay(500);
  digitalWrite(salida[4],LOW);
  break;
```

```
case '6':
  digitalWrite(salida[5],HIGH);
  delay(500);
  digitalWrite(salida[5],LOW);
  break;


case '7':
  digitalWrite(salida[6],HIGH);
  delay(500);
  digitalWrite(salida[6],LOW);
  break;


case '8':
  digitalWrite(salida[7],HIGH);
  delay(500);
  digitalWrite(salida[7],LOW);
  break;


case '9':
  digitalWrite(salida[8],HIGH);
  delay(500);
  digitalWrite(salida[8],LOW);
  break;


case '0':
  digitalWrite(salida[9],HIGH);
  delay(500);
  digitalWrite(salida[9],LOW);
  break;
```

8.5 Pruebas de aplicación con Arduino

Como se encontró un circuito adecuado para poder mover el brazo robótico, es que se opto por la creación de un programa sencillo, con el fin de enviar señales desde dicho programa y lograr el moviendo del brazo.
El programa consta de 10 botones, los cuales corresponden a las 10 salidas que necesita el brazo robótico para moverse completamente.

Figura 26

Al presionar el botón se ejecuta el evento correspondiente. Por ejemplo, si presionamos el botón "1", se realiza el siguiente código.

```
private void btn1_Click(object sender, RoutedEventArgs e)
{
    _port.Write("1");
}
```

"_port" corresponde a una variable de puerto serial, en este caso, el método que utiliza es el "write" el cual permite enviar una señal por puerto serial. El valor dentro de los paréntesis corresponde a la señal que llega a Arduino, con lo que en este caso, le está diciendo que mueva el motor 1, por lo tanto que abra las pinzas.

Para usar la comunicación serial (USB), es importante agregar la referencia al código.

```
using System.IO.Ports;
```

Además, es clave crear una variable que maneje la comunicación serial, como es la variable "_port". También es importante inicializar esta variable "_port", en este caso se utiliza "COM12", esto debido que el Arduino en el computador de prueba lo dejo en ese puerto, y dejar claro que el BaudRate = 9600, ya que es la velocidad con la cual se hará el envío de señales.

```
private readonly SerialPort _port;

public MainWindow()
{
    InitializeComponent();
    _port = new SerialPort("COM12") { BaudRate = 9600 };
    _port.Open();
}
```

Teniendo esto entendido, se puede realizar la comunicación PC- Arduino sin mayores problemas.

9 UNIÓN DE PROTOTIPOS

9.1 Conexión Kinect-PC-Arduino

Una vez creado los prototipos de audio, gestos, calibración y diseño de circuito en Arduino, es que se proceden a juntar en un solo software.
Para esto, se creó un proyecto en Visual Studio, con lenguaje C#, en donde el tipo de proyecto es WPF (Windows Presentation Foundation), la cual presenta una creación de interfaz de forma menos compleja, ya que cuenta con características de aplicaciones de Windows y aplicaciones web.
Como existe el modulo de calibración, audio y gestos, es que se decidió crear un menú, el cual posee un botón por cada opción. Además se creó el botón para salir de la aplicación.

Figura 27

Para aprovechar el uso de Kinect, es que el equipo desarrollador decidió crear la navegación con una mano, así dejando de lado el teclado y el mouse (pero no deshabilitado).

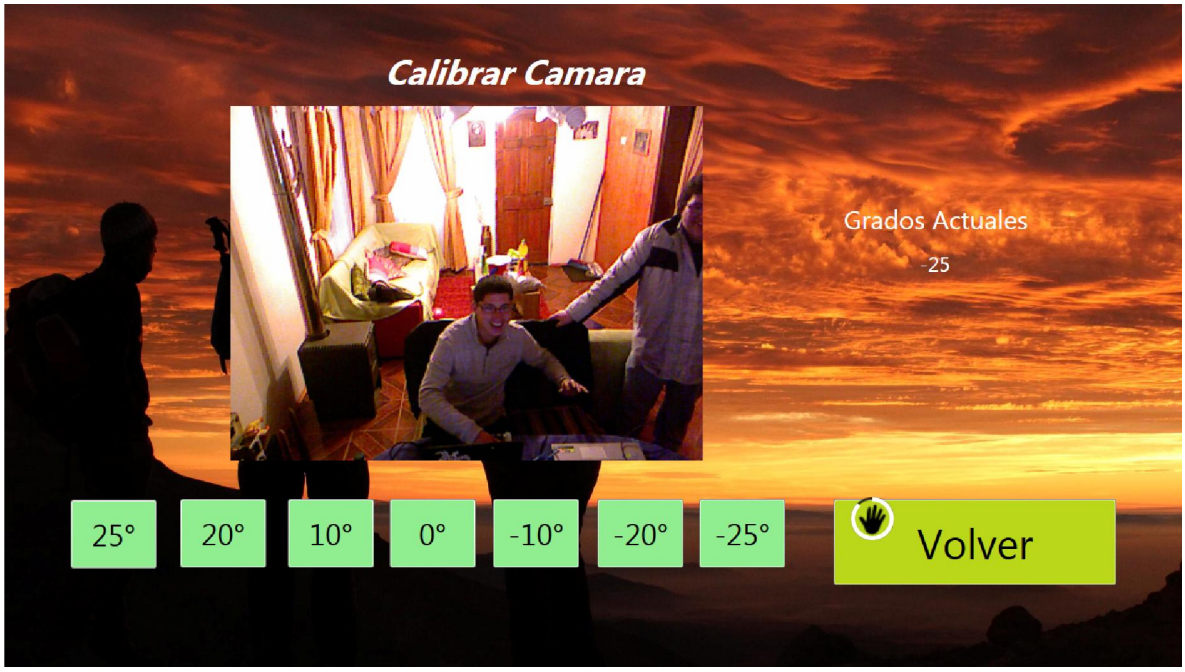
Para esto se creó un método que sigue los movimientos de la mano derecha del usuario, de tal manera, que cada vez que se posiciona sobre un botón, luego de 4 segundos se ingresara a la opción seleccionada.

Al ingresar a la opción Audio, el software envia un mensaje de bienvenida por el auricular. Además, se activa el micrófono interno de la Kinect, la cual gracias a la SDK, permite realizar el reconocimiento de voces. Cuando una palabra es válida, el sistema envía un mensaje por voz y por pantalla, además de reflejar esto en el brazo robótico.
Para salir de esta opción y retornar al menú principal, se debe decir la palabra “adiós”.

Para la opción de gestos, se activa una pantalla, en donde se puede observar el campo visual de la Kinect. También, al ingresar se activa la detección de gestos, los cuales fueron anteriormente mencionados y descritos.
Además, cada vez que un movimiento es aceptado, marca el numero del motor y la su movimiento correspondiente, finalizando con una acción en el brazo robótico.

La opción de calibrar cámara, contiene los grados que se escogieron para cambiar el ángulo de la Kinect, estos se seleccionan mediante el uso de la mano, posando esta más de 4

segundos sobre la opción. Para retornar al menú principal existe un botón volver que funciona de la misma forma que los anteriores.
Además, hay una pantalla que muestra el campo visual de Kinect y un mensaje al lado que da a conocer la cantidad de grados actuales de inclinación.



Para salir del software solo es necesario marcar con la mano sobre el botón salir durante unos segundos.

Al reunir todos los prototipos en uno solo, es que se tuvo problemas en la superposición de frames, de tal forma que cada vez que se abría una nueva opción, las opciones y botones del menú quedaban activos, por lo que a veces mientras se hacían pruebas de audio y sin querer nuestra mano estaba a la altura del botón "Salir" del menú principal, el software se tendía a cerrar. La solución a esto, fue crear un método, el cual bloquea los eventos propios del frame actual y pasa al nuevo frame con sus métodos.

En los casos que se muestre en algún frame el campo visual de Kinect, se debe hacer lo siguiente.

En lo que corresponde al XAML, se debe agregar lo siguiente.

```
<Image x:Name="Camara" Height="440" Width="838" Canvas.Left="417" Canvas.Top="141"/>
<Viewbox Grid.Row="1" Stretch="Uniform" HorizontalAlignment="Center" Canvas.Left="492" Canvas.Top="141" Height="440" Width="763"/>
```

Esto corresponde a una sección que ubica la imagen en un contenedor, la cual se actualiza. Pero para que esto funcione se debe agregar en el código CS.

```
    #region mostrar imagen en cam

    private void SensorColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
    {
        using (ColorImageFrame colorFrame = e.OpenColorImageFrame())
        {
            if (colorFrame != null)
            {
                // Copy the pixel data from the image to a temporary array
                colorFrame.CopyPixelDataTo(this.colorPixels);

                // Write the pixel data into our bitmap
                this.colorBitmap.WritePixels(
                    new Int32Rect(0, 0, this.colorBitmap.PixelWidth, this.colorBitmap.PixelHeight),
                    this.colorPixels,
                    this.colorBitmap.PixelWidth * sizeof(int),
                    0);
            }
        }
    }

    #endregion
}
```

Además, se debe agregar que cuando se inicialice la Kinect, el código para anclar este método.

```
kinectSensor.ColorFrameReady += this.SensorColorFrameReady;
```

Siendo KinectSensor la variable que corresponde a la Kinect reconocida.

Para el uso y reconocimiento de la mano, se debió agregar la imagen de la mano al frame, además del siguiente código en XAML.

Pero para que la mano funcione, se necesita el siguiente código.

Primero es necesario detectar el cuerpo, luego de eso definir que para el menú se utilizará la mano derecha como indicador.

```
//DEteccion del cuerpo y la mano a ocupar para manejar el menu
private void Kinect_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
{
    using (SkeletonFrame frame = e.OpenSkeletonFrame())
    {
        if (frame != null)
        {
            frame.CopySkeletonDataTo(this.FrameSkeletons);
            Skeleton skeleton = GetPrimarySkeleton(this.FrameSkeletons);

            if (skeleton == null)
            {
                kinectButton.Visibility = Visibility.Collapsed;
            }
            else
            {
                //MANEJO SOLO CON MANO DERECHA
                Joint primaryHand = skeleton.Joints[JointType.HandRight];
                TrackHand(primaryHand);

                //Tiempo para buscar gestos
                DateTime tiempoFin = DateTime.Now;
                TimeSpan transcurrido = tiempoFin.Subtract(tiempoInicio);
                if (transcurrido.Milliseconds > 450)
                {
                    bandera = true;
                    tiempoInicio = DateTime.Now;
                    Gestos(skeleton);
                }

            }
        }
```

10 PRUEBAS DE MOVIMIENTO

A continuación se hicieron una serie de pruebas, en las cuales se probaron distintos gestos, como lo muestran las siguientes figuras.

Figura 28

Para concluir con las pruebas de movimiento, todos los motores realizaban la acción esperada en ambos sentidos se giro, además, que cada vez que se terminaba un movimiento, este dejaba de replicarse en el brazo robótico.

11 PRUEBAS DE AUDIO

A continuación se hicieron una serie de pruebas de audio, en las cuales se probaron diferentes palabras claves que movían el motor, recibiendo una respuesta por voz y pantalla.

Figura 29

Para concluir con las pruebas de sonido, todos los motores realizaban la acción esperada al momento de decir la palabra clave, además, cada vez que se ingresaba una combinación no valida, el software enviaba un mensaje por pantalla y audio.

12 COMO ARMAR PASO A PASO

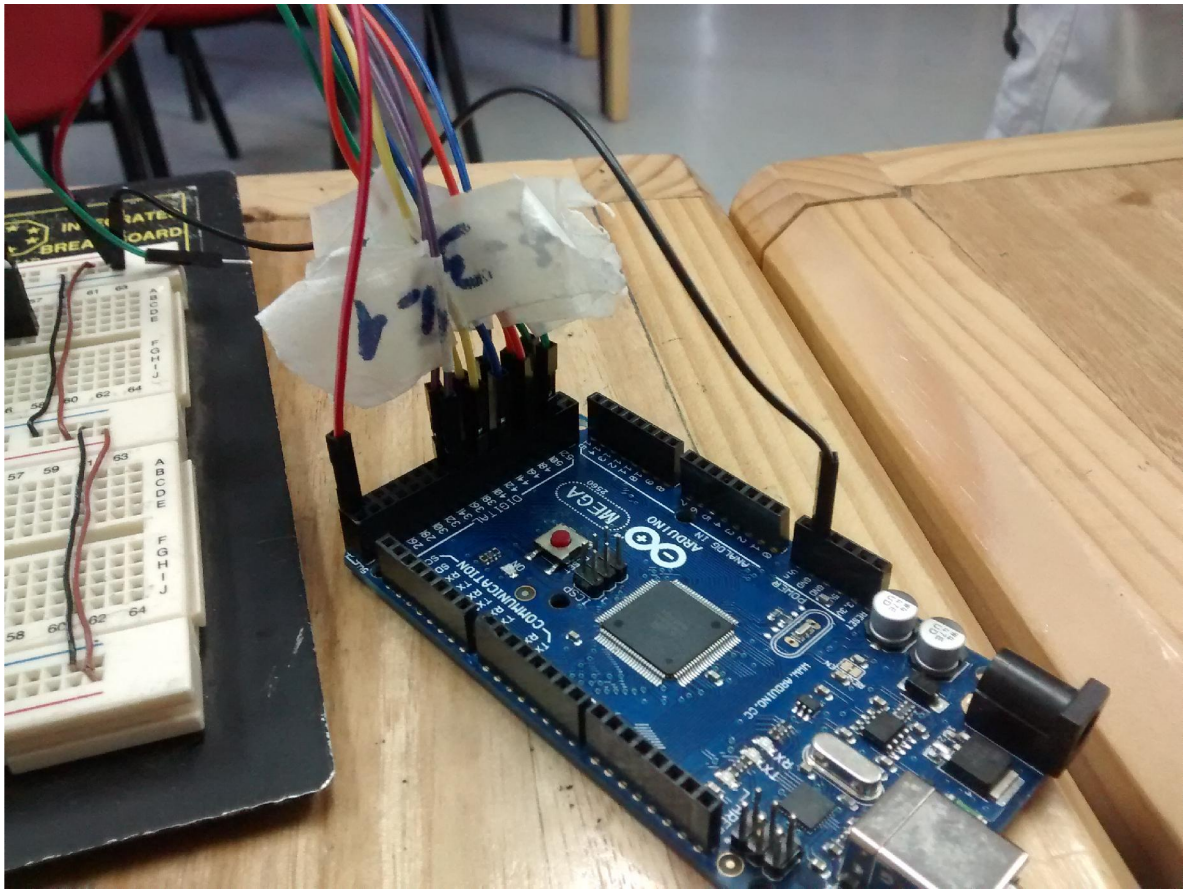
A continuación se presenta como armar el conexionado del brazo robótico con la Kinect, para lo cual se necesitan los siguientes materiales:

- Cámara Kinect (Precio $149.990).
- Arduino MEGA o algún otro equivalente con un mínimo de 10 salidas digitales (Precio $21.990).
- Brazo robótico OWI 535 (Precio $69.990).
- 40 Cables dupont (Precio $4.990)
- Protoboard (Precio $4.990)
- 3 integrados L293D (Precio $2.990)
- Cable USB arduino.
- Atornillador cruz y paleta.
- Computador que debe contar con los software mencionados anteriormente en el ítem 3.3.

Paso 1.

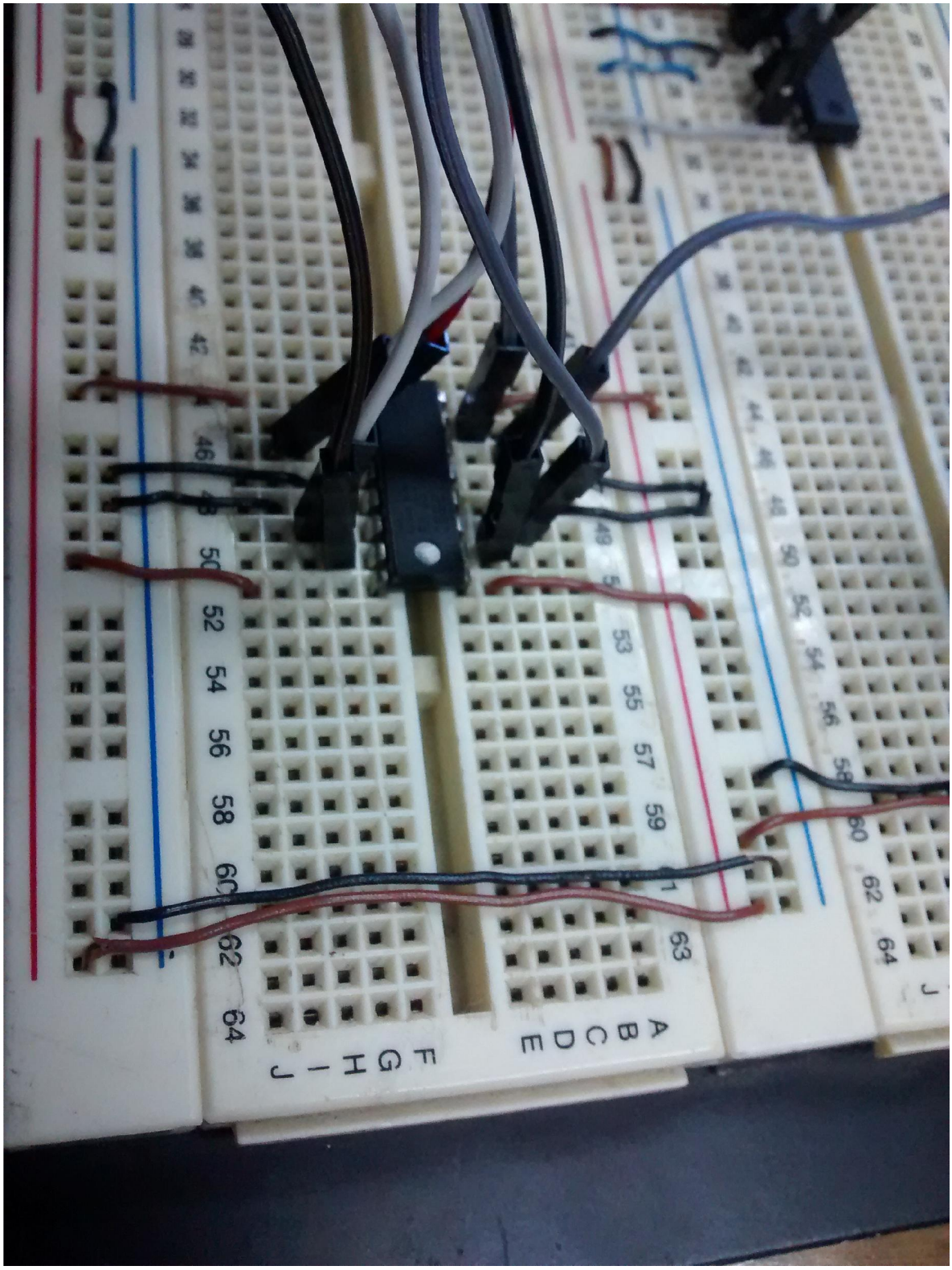
En el dispositivo Arduino conecte cables dupont a las salidas digitales 34, 35, 38, 39, 42, 43, 46, 47, 50, 51 (si utiliza otro Arduino verifique bien los puntos de salida).

Figura 30



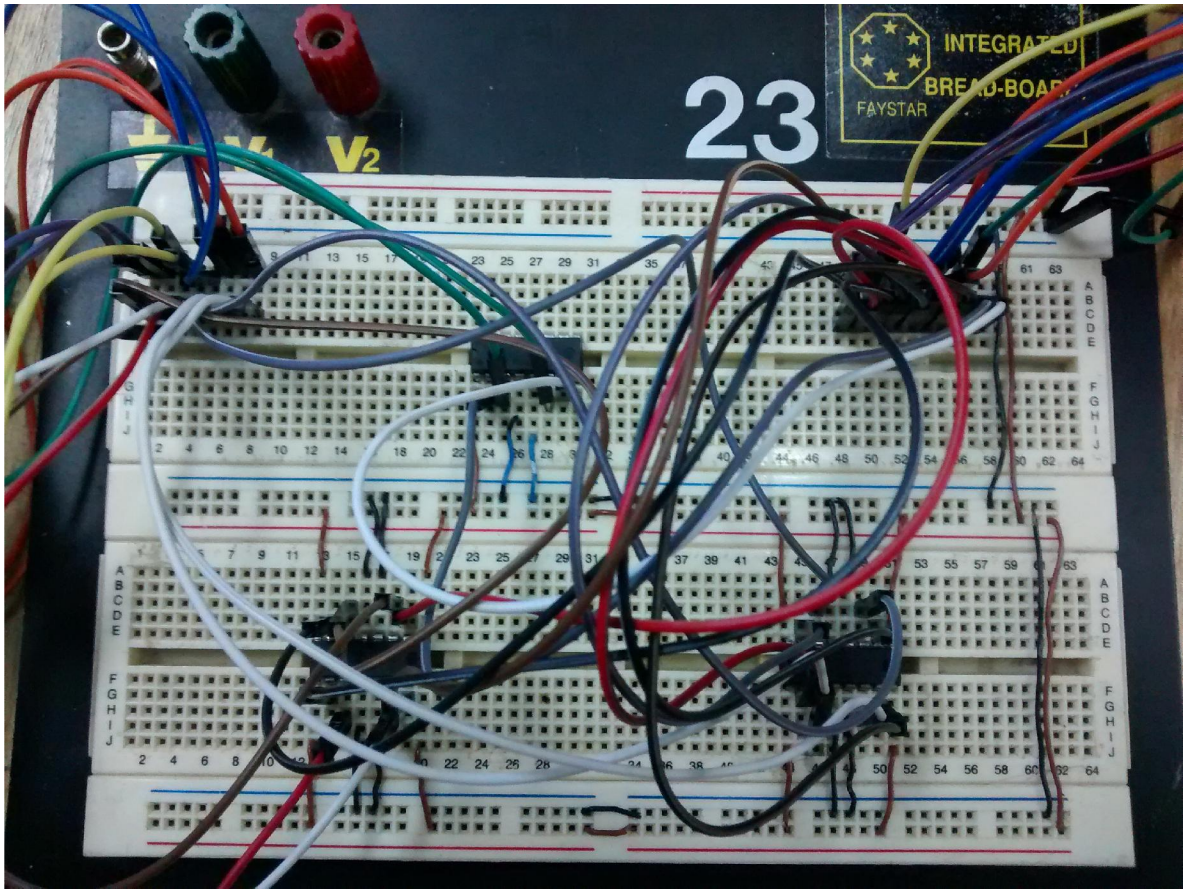
Luego, se procede a conectar los integrados a la protoboard, además de energizar y cablear a tierra como lo muestra a la siguiente figura.

Figura 31



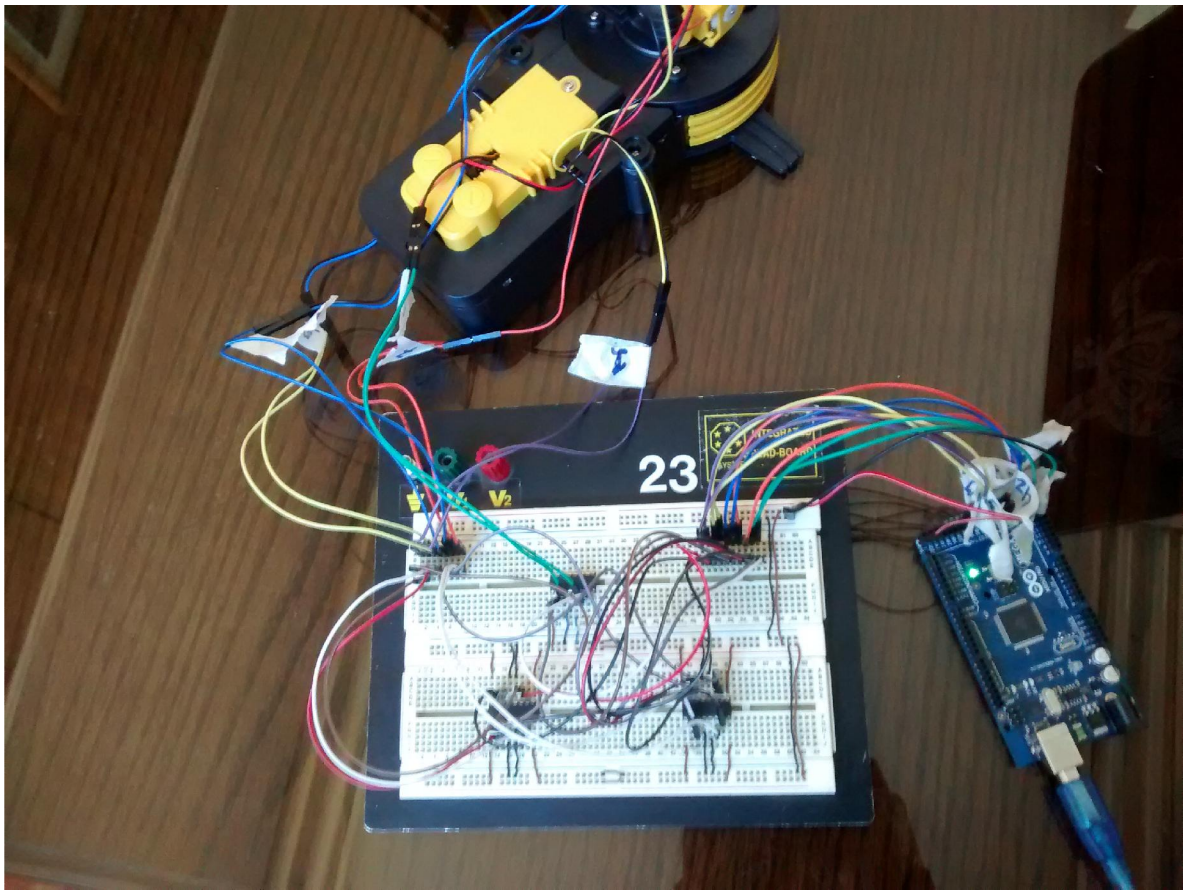
Después, se procede a conectar las salidas de Arduino a las entradas del integrado y conectar las salidas del integrado a los cables del brazo robótico.

Figura 32



Quedando de la siguiente manera.

Figura 33



Una vez finalizado el conexionado del hardware, se procede a programar el dispositivo Arduino. Para lo cual se necesita que el dispositivo esté conectado un computador por cable USB y que haya sido reconocido por este.

Se dispone a ejecutar el programa Arduino para realizar su programación. Para esto, se debe copiar el código puesto en el ítem 8.4 del actual documento.

Con esto, se da por terminado la programación Arduino y conexionado de brazo robótico.

Paso 2.
Para empezar, se necesita conectar la Kinect al computador y tener los respectivos software instalados.

Luego se inicia Visual Studio 2012, se debe ir a la pestaña "Archivos->Nuevo->proyecto", se procede a escoger la opción "Visual C#" y se selecciona "Aplicación WPF", se completa el nombre con "KinectTesis" y se presiona aceptar.

Se debe agregar la clase generics.cs al proyecto la cual contiene el funcionamiento base de la mano para el menú.

Ver anexo 6.1

Paso 2.1 Archivo MainWindow.xaml.cs

Este archivo maximiza la pantalla y permite la navegación a través de frame dentro de todo el programa. Luego deriva a archivo inicio.xaml.cs

Ver anexo 1.1

Además, se debe agregar el siguiente código en el archivo .xaml

Ver anexo 1.2

Paso 2.2 inicio.xaml.cs

En este archivo se declara todo el contenido del menú principal, como los botones y utilización de mano con la Kinect.
Se debe incluir la librería "using Microsoft.Kinect;" para utilizar Kinect.
Se utiliza código base para reconocer kinect obtenido del SDK de Kinect

Ver anexo 2.1

Además, en el archivo Inicio.xaml, se debe colocar el siguiente código que corresponde a la creación de los botones y la mano de la mano por Kinect.

Ver anexo 2.2

Paso 2.3 Audio.xaml.cs

En este archivo se define todo lo que corresponde a la detección de palabras por parte de Kinect, además del reconocimiento de la Kinect y la definición de palabras.
No olvidar instalar los programas bases y cargar las referencias mencionadas en el ítem 7.3.

También es necesario incluir en el proyecto una carpeta de sonidos e imágenes, las cuales contienen los audios a reproducir y las imágenes a mostrar respectivamente.

Ver anexo 3.1

Además, en el archivo Audio.xaml, se debe colocar el siguiente código que corresponde al llamado de las imágenes y etiquetas.

Ver anexo 3.2

Paso 2.4 Movimiento.xaml.cs

En este archivo se define todo lo que corresponde a la detección de movimientos por parte de Kinect, además del reconocimiento de la Kinect y la definición de gestos a detectar.
Para ver los gestos definidos se puede revisar el ítem 6.1

Ver anexo 4.1

Además, en el archivo Movimiento.xaml, se debe colocar el siguiente código que corresponde a la integración una pantalla con el campo visual de la Kinect, una etiqueta que muestra el gesto a realizado y un botón para regresar al menú principal.

Ver anexo 4.2

Paso 2.5 Calibrar.xaml.cs

En este archivo se define lo que corresponde a la manipulación de la inclinación de la Kinect, con el fin de ajustar para que el usuario se vea complemente en el área que se realiza la detección.

Ver anexo 5.1

Además, en el archivo Calibrar.xaml, se debe colocar el siguiente código que corresponde a la integración una pantalla con el campo visual de la Kinect y se definen botones para cambiar el ángulo de la cámara.

Ver anexo 5.2
Con todo esto, se puede conectar la Kinect y arduino, con lo que se puede comenzar a utilizar el software y entretenerse.

13 RESUMEN ESFUERZO REQUERIDO

Los tiempos ocupados por el equipo desarrollador son los siguientes:

Figura 34

Actividades/fases	**N° Horas Braulio CIfuentes**	**N° Horas Jaime Novoa**
Análisis de Requerimientos	**216**	**216**
Investigación sobre Microsoft Kinect y Arduino	200	200
Definición de requerimientos	10	10
Reunión con encargado	5	5
Correcciones al análisis de requerimientos	1	1
Diseño Prototipo Kinect	**279**	**379**
Implementación prototipo Kinect	250	350
Reunión con encargado	4	4
Refinamiento del software	25	25
Análisis rapido	**120**	**120**
Investigación detallada de Arduino y comunicación	50	50
Diseño Prototipo Arduino	20	20
Implementación prototipo Arduino	50	50
Reunión con encargado	**4**	**4**
Refinamiento del software	**30**	**30**
Unión de prototipos y modificaciones	**10**	**10**
Pruebas de sistema	**40**	**40**
Entrega del documento de tesis	**30**	**30**
TOTAL	729	829

Considerando la estimación inicial, los tiempos de desarrollo del sistema se ajustan parcialmente a los tiempos definidos al comienzo del proyecto.

14 CONCLUSIONES

Al finalizar el proyecto podemos concluir que los objetivos planteados fueros cumplidos a cabalidad, con lo cual se logro un éxito en el desarrollo del software y lo que respecta a este.

El cumplimiento de los requisitos para el desarrollo del software se debe al gran desempeño en el análisis y diseño de los objetivos, esto se debió principalmente al buen uso de la metodología implementada, ya que en su comienzo, el equipo desarrollador poseía conocimientos vagos sobre el tema, que durante los primeros meses se fueron atenuando debido a motivación de los desarrolladores.

Además, el equipo desarrollador se puso una nueva meta, la cual consistió en el desarrollo de un modulo de reconocimiento de voz, con el fin de agregar mayor funcionalidad al sistema, lo cual los llevo a investigar y aprender el uso de tecnologías de vanguardia.

Para concluir, del punto de vista académico, los conocimientos otorgados por la universidad no fueron lo suficiente para realizar el proyecto, pero si fue una buena base para poder investigar y aprender por métodos propios. Del punto de vista personal, la realización de este proyecto fue un buen reto, ya que se pusieron en juego habilidades adquiridas dentro de la Universidad.

15 BIBLIOGRAFÍA

- Ejemplos para desarollo [en línea]
  <http://stackoverflow.com>
  [Consulta: 17 Abril 2014]

- Google, búsqueda de información en general. [en línea]
  <http://www.google.cl>
  [Consulta: 10 Abril 2014]

- Repositorio de software Microsoft [en línea]
  <http://www.micrisoft.com/en-us/download>
  [Consulta: 17 Abril 2014]

- Librería Kinect. [en línea]
  < http://msdn.microsoft.com/en-us/library/hh855347.aspx >
  [Consulta: 7 Abril 2014]

- Videos tutoriales [en línea]
  <http://www.microsoft.com/enus/kinectforwindowsdev/Videos.aspx>
  [Consulta: 7 Abril 2014]

- Arduino con integrado. [en línea]
  < http://alonsodub.wordpress.com/2012/06/08/control-de-motor-cc-velocidad-y-direccion/comment-page-1/>
  [Consulta: 10 mayo 2014]

- Arduino oficial [en línea]
  <http://www.arduino.cc>
  [Consulta: 1 Abril 2014]

- Coding4Fun Kinect Toolkit [en línea]
  < http://c4fkinect.codeplex.com/>
  [Consulta: 30 Abril 2014]

16 ANEXOS

16.1 Anexo 1.1 Código MainWindows.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            this.WindowState = System.Windows.WindowState.Maximized;
            this.WindowStyle                                                    =
System.Windows.WindowStyle.None;————————————————————————

            if (Generics.LoadingStatus == 0)
            {
                _mainFrame.Source = new Uri("Inicio.xaml", UriKind.Relative);
                Generics.LoadingStatus = 1;
            }
        }

    }
}
```

16.2 Anexo 1.2 Código MainWindows.xaml

```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      Title="Inicio"  Height="480" Width="640" >

    <Grid x:Name="LayoutRoot">
        <Image  Source="Images/atardecer.jpg" Stretch="UniformToFill"/>
        <DockPanel>
            <Frame x:Name="_mainFrame"    NavigationUIVisibility="Hidden" />
        </DockPanel>
```

```
        </Grid>
    </Window>
```

16.3 Anexo 2.1 Código Inicio.xaml.cs

```
using Microsoft.Kinect;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfApplication1
{
    public partial class Inicio : Page
    {
        #region "Variables Kinect"
        private KinectSensor _Kinect;
        private Skeleton[] FrameSkeletons;

        List<Button> buttons;
        static Button selected;

        float handX;
        float handY;
        #endregion

        public Inicio()
        {
            InitializeComponent();
            InitializeButtons();
            Generics.ResetHandPosition(kinectButton);
            kinectButton.Click += new RoutedEventHandler(kinectButton_Click);
            DiscoverKinectSensor();
        }


        #region "Movimiento de mano por pantalla"
        //Listado de botones a manipular con la mano
        private void InitializeButtons()
        {
            buttons = new List<Button> { Audio, Movimiento, Salir, Calibrar};
        }

        //Evento para cambio del estado de kinect (saber si hay kinect)
        private void DiscoverKinectSensor()
        {
            KinectSensor.KinectSensors.StatusChanged += KinectSensors_StatusChanged;
```

```
            this.Kinect = KinectSensor.KinectSensors.FirstOrDefault(x => x.Status ==
KinectStatus.Connected);
        }

        //Analiza estado de kinect (hay o no kinect)
        private void KinectSensors_StatusChanged(object sender,
StatusChangedEventArgs e)
        {
            switch (e.Status)
            {
                case KinectStatus.Connected:
                    if (this.Kinect == null)
                    {
                        this.Kinect = e.Sensor;
                    }
                    break;
                case KinectStatus.Disconnected:
                    if (this.Kinect == e.Sensor)
                    {
                        this.Kinect = null;
                        this.Kinect = KinectSensor.KinectSensors.FirstOrDefault(x =>
x.Status == KinectStatus.Connected);
                        if (this.Kinect == null)
                        {
                            MessageBox.Show("Kinect Desconectada.");
                        }
                    }
                    break;
            }
        }

        //Clase Kinect con su set y get, para que si hay kinect la asigne
        public KinectSensor Kinect
        {
            get { return this._Kinect; }
            set
            {
                if (this._Kinect != value)
                {
                    if (this._Kinect != null)
                    {
                        this._Kinect = null;
                    }
                    if (value != null && value.Status == KinectStatus.Connected)
                    {
                        this._Kinect = value;
                        InitializeKinectSensor(this._Kinect);
                    }
                }
            }
        }


        //Cuando hay Kinect la inicia
        private void InitializeKinectSensor(KinectSensor kinectSensor)
        {
            if (kinectSensor != null)
            {
```

```
                //Obtiene el esqueleto y corrige su ubicacion
                kinectSensor.SkeletonStream.Enable(new TransformSmoothParameters()
                {
                    Correction = 0.5f,
                    JitterRadius = 0.05f,
                    MaxDeviationRadius = 0.04f,
                    Smoothing = 0.5f
                });

                //Nuevo esqueletos disponibles
                kinectSensor.SkeletonFrameReady += Kinect_SkeletonFrameReady;

                if (!kinectSensor.IsRunning)
                {
                    kinectSensor.Start();
                }

                //Guarda todos los esqueletos en pantalla
                this.FrameSkeletons = new
Skeleton[this.Kinect.SkeletonStream.FrameSkeletonArrayLength];

            }
        }



        //DEteccion del cuerpo y la mano a ocupar para manejar el menu
        private void Kinect_SkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
        {
            using (SkeletonFrame frame = e.OpenSkeletonFrame())
            {
                if (frame != null)
                {
                    //Busca en todos los esqueletos y escoge el mas cercano
                    frame.CopySkeletonDataTo(this.FrameSkeletons);
                    Skeleton skeleton = GetPrimarySkeleton(this.FrameSkeletons);

                    if (skeleton == null)
                    {
                        kinectButton.Visibility = Visibility.Collapsed;
                    }
                    else
                    {
                        kinectButton.Visibility = Visibility.Visible;
                        //MANEJO SOLO CON MANO DERECHA
                        Joint primaryHand = skeleton.Joints[JointType.HandRight];
                        TrackHand(primaryHand);

                    }
                }
            }
        }

        //Manejo de mano sobre el canvas y botones
        private void TrackHand(Joint hand)
        {
```

```
            if (hand.TrackingState == JointTrackingState.NotTracked)
            {
                kinectButton.Visibility = System.Windows.Visibility.Collapsed;
            }
            else
            {
                kinectButton.Visibility = System.Windows.Visibility.Visible;

                DepthImagePoint point =
this.Kinect.CoordinateMapper.MapSkeletonPointToDepthPoint(hand.Position,
DepthImageFormat.Resolution640x480Fps30);
                handX = (int)((point.X * LayoutRoot.ActualWidth /
this.Kinect.DepthStream.FrameWidth) -
                    (kinectButton.ActualWidth / 2.0));
                handY = (int)((point.Y * LayoutRoot.ActualHeight /
this.Kinect.DepthStream.FrameHeight) -
                    (kinectButton.ActualHeight / 2.0));
                Canvas.SetLeft(kinectButton, handX);
                Canvas.SetTop(kinectButton, handY);

                if (isHandOver(kinectButton, buttons)) kinectButton.Hovering();
                else kinectButton.Release();


                if (hand.JointType == JointType.HandRight)
                {
                    kinectButton.ImageSource =
"/KinectTesis;component/Images/myhand.png";
                    kinectButton.ActiveImageSource =
"/RVI_Education;component/Images/myhand.png";
                }
                else
                {
                    kinectButton.ImageSource =
"/KinectTesis;component/Images/myhand.png";
                    kinectButton.ActiveImageSource =
"/KinectTesis;component/Images/myhand.png";
                }
            }
        }

        //Detecta la posicion de la mano y carga el boton
        private bool isHandOver(FrameworkElement hand, List<Button> buttonslist)
        {
            var handTopLeft = new Point(Canvas.GetLeft(hand), Canvas.GetTop(hand));
            var handX = handTopLeft.X + hand.ActualWidth / 2;
            var handY = handTopLeft.Y + hand.ActualHeight / 2;

            foreach (Button target in buttonslist)
            {

                if (target != null)
                {
                    Point targetTopLeft = new Point(Canvas.GetLeft(target),
Canvas.GetTop(target));
                    if (handX > targetTopLeft.X &&
                        handX < targetTopLeft.X + target.Width &&
                        handY > targetTopLeft.Y &&
```

```
                        handY < targetTopLeft.Y + target.Height)
                    {
                        selected = target;
                        return true;
                    }
                }
            }
            return false;
        }


        //Obtiene el cuerpo mas cerca de la camara
        private static Skeleton GetPrimarySkeleton(Skeleton[] skeletons)
        {
            Skeleton skeleton = null;
            if (skeletons != null)
            {
                for (int i = 0; i < skeletons.Length; i++)
                {
                    if (skeletons[i].TrackingState == SkeletonTrackingState.Tracked)
                    {
                        if (skeleton == null)
                        {
                            skeleton = skeletons[i];
                        }
                        else
                        {
                            if (skeleton.Position.Z > skeletons[i].Position.Z)
                            {
                                skeleton = skeletons[i];
                            }
                        }
                    }
                }
            }
            return skeleton;
        }



        #endregion

        #region evento botones
        void kinectButton_Click(object sender, RoutedEventArgs e)
        {
            selected.RaiseEvent(new RoutedEventArgs(Button.ClickEvent, selected));
        }


        private void Audio_Click(object sender, RoutedEventArgs e)
        {
            bloquearpantalla();
            (Application.Current.MainWindow.FindName("_mainFrame") as Frame).Source
= new Uri("Audio.xaml", UriKind.Relative);
        }

        private void Movimiento_Click(object sender, RoutedEventArgs e)
        {
```

```
            bloquearpantalla();
            (Application.Current.MainWindow.FindName("_mainFrame") as Frame).Source
= new Uri("Movimiento.xaml", UriKind.Relative);
        }

        private void Salir_Click(object sender, RoutedEventArgs e)
        {
            App.Current.Shutdown();
        }

        private void Calibrar_Click(object sender, RoutedEventArgs e)
        {
            bloquearpantalla();
            (Application.Current.MainWindow.FindName("_mainFrame") as Frame).Source
= new Uri("Calibrar.xaml", UriKind.Relative);

        }

        private void bloquearpantalla()
        {
            try
            {
                KinectSensor.KinectSensors.StatusChanged -=
KinectSensors_StatusChanged;
                this.Kinect.SkeletonFrameReady -= Kinect_SkeletonFrameReady;
                kinectButton.Click -= new RoutedEventHandler(kinectButton_Click);

            }
            catch (Exception)
            {
            }
        }

        #endregion
    }
}
```

16.4 Anexo 2.2 Código Inicio.xaml

```
<Page x:Class="WpfApplication1.Inicio"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
      d:DesignHeight="737" d:DesignWidth="961"
     Title="Page">

    <Grid Canvas.Top="-16" x:Name="LayoutRoot">
        <Grid.RowDefinitions>
            <RowDefinition Height="197*"/>
            <RowDefinition Height="165*"/>
            <RowDefinition Height="375*"/>
        </Grid.RowDefinitions>
```

```
        <Canvas Margin="0,19,-410,-56" Grid.RowSpan="3">

            <Button  x:Name="Audio"  Content="Audio"  Height="366"  Canvas.Left="81"
    Canvas.Top="150"     Width="356"     Click="Audio_Click"     Background="#FFF5C522"
    FontSize="48"/>
            <Button     x:Name="Movimiento"     Content="Gestos"         Height="366"
    Canvas.Left="534"     Canvas.Top="150"     Width="332"     Click="Movimiento_Click"
    Background="#FF17CB6A" FontSize="48"/>
            <Button  x:Name="Salir"  Content="Salir"  Height="366"  Canvas.Left="952"
    Canvas.Top="150"     Width="349"     Click="Salir_Click"     Background="#FFB93C53"
    FontSize="48"/>
            <Button     x:Name="Calibrar"     Content="Calibrar"     Canvas.Left="443"
    Canvas.Top="590"   Height="66"   Width="493"   FontSize="36"   Background="#FFC9683B"
    Click="Calibrar_Click"/>
            <Controls:HoverButton   Margin="0"   Padding="0"   x:Name="kinectButton"
    ImageSize="50"

    ImageSource="/KinectTesis;component/Images/myhand.png"

    ActiveImageSource="/KinectTesis;component/Images/myhand.png"
                                TimeInterval="4000"             Panel.ZIndex="1200"
    Canvas.Left="0" Canvas.Top="0" />
        </Canvas>
    </Grid>
```

16.5 Anexo 3.1 Código Audio.xaml.cs

```
using Microsoft.Kinect;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Speech.AudioFormat;
using Microsoft.Speech.Recognition;
using System.Threading;
using System.IO.Ports;

namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for Page.xaml
    /// </summary>
    public partial class Audio : Page
    {
```

```
        private readonly SerialPort _port;
        #region "Variables Kinect"
        private KinectSensor _Kinect;
        int flag = 0;
        SpeechRecognitionEngine speechengine;
        #endregion


        public Audio()
        {
            InitializeComponent();
            this.Loaded += (s, e) => { DiscoverKinectSensor(); };
            _port = new SerialPort("COM12") { BaudRate = 9600 };
            try
            {
                _port.Open();
            }
            catch (Exception)
            {
            }
            conectaActiva();
        }


        //Evento para cambio del estado de kinect (saber si hay kinect)
        private void DiscoverKinectSensor()
        {
            KinectSensor.KinectSensors.StatusChanged += KinectSensors_StatusChanged;
            this.Kinect = KinectSensor.KinectSensors.FirstOrDefault(x => x.Status ==
KinectStatus.Connected);
        }


        //Analiza estado de kinect (hay o no kinect)
        private void KinectSensors_StatusChanged(object sender,
StatusChangedEventArgs e)
        {
            switch (e.Status)
            {
                case KinectStatus.Connected:
                    if (this.Kinect == null)
                    {
                        this.Kinect = e.Sensor;
                    }
                    break;
                case KinectStatus.Disconnected:
                    if (this.Kinect == e.Sensor)
                    {
                        this.Kinect = null;
                        this.Kinect = KinectSensor.KinectSensors.FirstOrDefault(x =>
x.Status == KinectStatus.Connected);
                        if (this.Kinect == null)
                        {
                            MessageBox.Show("Kinect Desconectada.");
                        }
                    }
                    break;
            }
```

```
        }

        //Selecciona kinect
        public KinectSensor Kinect
        {
            get { return this._Kinect; }
            set
            {
                if (this._Kinect != value)
                {
                    if (this._Kinect != null)
                    {
                        this._Kinect = null;
                    }
                    if (value != null && value.Status == KinectStatus.Connected)
                    {
                        this._Kinect = value;
                        InitializeKinectSensor(this._Kinect);
                    }
                }
            }
        }


        //Inicio de kinect
        //Muestra imagen producida por kinect
        //Detecta el cuerpo
        private void InitializeKinectSensor(KinectSensor kinectSensor)
        {
            if (kinectSensor != null)
            {

                if (!kinectSensor.IsRunning)
                {
                    kinectSensor.Start();
                }

            }
        }

        private void bloquearpantalla()
        {
            try
            {
                KinectSensor.KinectSensors.StatusChanged -=
KinectSensors_StatusChanged;
            }
            catch (Exception)
            {

            }
        }
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            //En esta linea se indica que cada vez que el estado del dispositivo
cambie se mandara llamar al evento KinectSensors_StatusChanged
            KinectSensor.KinectSensors.StatusChanged += new
EventHandler<StatusChangedEventArgs>(KinectSensors_StatusChanged1);
```

```
            //Este metodo conecta activa lo que hara es asignar el primer
dispositivo encontrado a nuestra variable _Kinect,ademas de inicializarlo e iniciar
el reconocimiento de voz
            conectaActiva();
        }

        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {

            if (this._Kinect != null)
            {
                this._Kinect.AudioSource.Stop();
                this._Kinect.Stop();
                this._Kinect = null;
            }
        }
        void KinectSensors_StatusChanged1(object sender, StatusChangedEventArgs e)
        {
            //Hacemos un switch para ver cual es el estado del dispositivo
            switch (e.Status)
            {
                //En caso de que el status sea Connected quiere decir que hay una
conexion correcta entre la PC y el Kinect
                case (KinectStatus.Connected):
                    //De la misma forma mandamos llamar al metodo conectaActiva() el
cual inicializara el dispositivo Kinect
                    conectaActiva();
                    break;
                //En caso de que el status sea Disconnected se la variable _Kinect
se volvera nula e intentaremos buscar otro dispositivo Kinect cuyo estado sea
Connected si no se encuentra mandaremos un mensaje indicando que No hay ningun
Kinect conectado
                case (KinectStatus.Disconnected):
                    if (this._Kinect == e.Sensor)
                    {
                        this._Kinect = null;
                        this._Kinect = KinectSensor.KinectSensors.FirstOrDefault(x
=> x.Status == KinectStatus.Connected);
                        if (this._Kinect == null)
                        {
                            statusK.Text = "No hay ningun Kinect conectado";
                        }
                    }
                    break;
            }
        }
        void conectaActiva()
        {
            //Nos aseguramos que la cuenta de sensores conectados sea de al menos 1
            if (KinectSensor.KinectSensors.Count > 0)
            {
                //Checamos que la variable _Kinect sea nula
                if (this._Kinect == null)
                {
                    //Asignamos el primer sensor Kinect a nuestra variable
                    this._Kinect = KinectSensor.KinectSensors[0];
                    if (this._Kinect != null)
```

```
                {
                    try
                    {
                        //Iniciamos el dispositivo Kinect
                        this._Kinect.Start();
                        //Esto es opcional pero ayuda a colocar el dispositivo
Kinect a un cierto angulo de inclinacion, desde -27 a 27
                        //_Kinect.ElevationAngle = 0;
                        //Informamos que se ha conectado e inicializado
correctamente el dispositivo Kinect
                        statusK.Text = "Haz conectado el Kinect";
                    }
                    catch (Exception ex)
                    {
                        //Si hay algun error lo mandamos en el TextBlock statusK
                        statusK.Text = ex.Message.ToString();
                    }
                    //Creamos esta variable ri que tratara de encontrar un
language pack valido haciendo uso del metodo obtenerLP()
                    RecognizerInfo ri = obtenerLP();

                    //Si se encontro el language pack requerido lo asignaremos a
nuestra variable speechengine
                    if (ri != null)
                    {
                        this.speechengine = new SpeechRecognitionEngine(ri.Id);

                        //Creamos esta variable opciones la cual almacenara las
opciones de palabras o frases que podran ser reconocidas por el dispositivo
                        var opciones = new Choices();
                        opciones.Add("accion", "ACCION");
                        opciones.Add("uno", "UNO");
                        opciones.Add("dos", "DOS");
                        opciones.Add("tres", "TRES");
                        opciones.Add("cuatro", "CUATRO");
                        opciones.Add("cinco", "CINCO");
                        opciones.Add("derecha", "DERECHA");
                        opciones.Add("izquierda", "IZQUIERDA");
                        opciones.Add("subir", "SUBIR");
                        opciones.Add("bajar", "BAJAR");
                        opciones.Add("abrir", "ABRIR");
                        opciones.Add("cerrar", "CERRAR");
                        opciones.Add("adios", "ADIOS");

                        //En esta linea "windows ocho" es el valor de opcion y
"TRES" es la llave y asi sucesivamente
                        //opciones.Add(new SemanticResultValue("windows ocho",
"TRES"));
                        //opciones.Add(new SemanticResultValue("nuevo windows",
"TRES"));


                        //Esta variable creará todo el conjunto de frases y
palabras en base a nuestro lenguaje elegido en la variable ri
                        var grammarb = new GrammarBuilder { Culture = ri.Culture
};

                        grammarb.Append(opciones);
```

```
                            //Creamos una variable de tipo Grammar utilizando como
parametro a grammarb
                            var grammar = new Grammar(grammarb);

                            //Le decimos a nuestra variable speechengine que cargue
a grammar
                            this.speechengine.LoadGrammar(grammar);

                            //mandamos llamar al evento SpeechRecognized el cual se
ejecutara cada vez que una palabra sea detectada
                            speechengine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(speechengine_SpeechRecognized);

speechengine.SetInputToAudioStream(_Kinect.AudioSource.Start(), new
SpeechAudioFormatInfo(EncodingFormat.Pcm, 16000, 16, 1, 32000, 2, null));
                            speechengine.RecognizeAsync(RecognizeMode.Multiple);
                        }

                    }
                }
            }
        }
        private RecognizerInfo obtenerLP()
        {
            System.Media.SoundPlayer sound;
            sound = new System.Media.SoundPlayer("../../Sounds/bienvenidos a
kibot.wav");
            sound.Play();


            foreach (RecognizerInfo recognizer in
SpeechRecognitionEngine.InstalledRecognizers())
            {
                string value;
                recognizer.AdditionalInfo.TryGetValue("Kinect", out value);
                //Aqui es donde elegimos el lenguaje, si se dan cuenta hay una parte
donde dice "es-MX" para cambiar el lenguaje a ingles de EU basta con cambiar el
valor a "en-US"
                if ("True".Equals(value, StringComparison.OrdinalIgnoreCase) && "es-
MX".Equals(recognizer.Culture.Name, StringComparison.OrdinalIgnoreCase))
                {
                    //Si se encontro el language pack solicitado se retorna a
recognizer
                    return recognizer;
                }
            }
            //En caso de que no se encuentre ningun languaje pack se retorna un
valor nulo
            return null;
        }
        void speechengine_SpeechRecognized(object sender, SpeechRecognizedEventArgs
e)
        {
            System.Media.SoundPlayer m1, m2, m3, m4, m5, abrir, cerrar, subm2,
bajm2, subm3, bajm3, subm4, bajm4, derecha, izquierda, error, adios;
            m1 = new System.Media.SoundPlayer("../../Sounds/motor1.wav");
            m2 = new System.Media.SoundPlayer("../../Sounds/motor2.wav");
```

```
            m3 = new System.Media.SoundPlayer("../../Sounds/motor3.wav");
            m4 = new System.Media.SoundPlayer("../../Sounds/motor4.wav");
            m5 = new System.Media.SoundPlayer("../../Sounds/motor5.wav");
            /*            abrir = new System.Media.SoundPlayer("../../Sounds/abrir
pinzas.wav");
                        cerrar = new System.Media.SoundPlayer("../../Sounds/cerrar
pinzas.wav");
                        subm2 = new
System.Media.SoundPlayer("../../Sounds/subm2.wav");
                        bajm2 = new
System.Media.SoundPlayer("../../Sounds/bajm2.wav");
                        subm3 = new
System.Media.SoundPlayer("../../Sounds/subm3.wav");
                        bajm3 = new
System.Media.SoundPlayer("../../Sounds/bajm3.wav");
                        subm4 = new
System.Media.SoundPlayer("../../Sounds/subm4.wav");
                        bajm4 = new
System.Media.SoundPlayer("../../Sounds/bajm4.wav");
                        izquierda = new
System.Media.SoundPlayer("../../Sounds/izqm5.wav");
                        derecha = new
System.Media.SoundPlayer("../../Sounds/derm5.wav");
                        error = new
System.Media.SoundPlayer("../../Sounds/error.wav");
                        adios = new
System.Media.SoundPlayer("../../Sounds/adios.wav");
                        */
            abrir = new System.Media.SoundPlayer("../../Sounds/accion.wav");
            cerrar = new System.Media.SoundPlayer("../../Sounds/accion.wav");
            subm2 = new System.Media.SoundPlayer("../../Sounds/accion.wav");
            bajm2 = new System.Media.SoundPlayer("../../Sounds/accion.wav");
            subm3 = new System.Media.SoundPlayer("../../Sounds/accion.wav");
            bajm3 = new System.Media.SoundPlayer("../../Sounds/accion.wav");
            subm4 = new System.Media.SoundPlayer("../../Sounds/accion.wav");
            bajm4 = new System.Media.SoundPlayer("../../Sounds/accion.wav");
            izquierda = new System.Media.SoundPlayer("../../Sounds/accion.wav");
            derecha = new System.Media.SoundPlayer("../../Sounds/accion.wav");
            error = new System.Media.SoundPlayer("../../Sounds/error.wav");
            adios = new System.Media.SoundPlayer("../../Sounds/adios.wav");

            //la variable igualdad sera el porcentaje de igualdad entre la palabra
reconocida y el valor de opcion
            //es decir si yo digo "uno" y el valor de opcion es "uno" la igualdad
sera mayor al 50 %
            //Si yo digo "jugo" y el valor de opcion es "uno" notaras que el sonido
es muy similar pero quizas no mayor al 50 %
            //El valor de porcentaje va de 0.0  a 1.0, ademas notaras que le di un
valos de .5 lo cual representa el 50% de igualdad

            const double igualdad = 0.4;

            //Si hay mas del 50% de igualdad con alguna de nuestras opciones
            if (e.Result.Confidence > igualdad)
            {
                Uri src;
                BitmapImage img;
```

```
                //haremos un switch para aquellos valores que se componen de
unicamente una palabra
                switch (e.Result.Words[0].Text)
                {
                    //En caso de que digamos "uno" la llave "UNO" se abrira y se
realizara lo siguiente
                    case "UNO":
                        //Se mandara un mensaje alusivo a la imagen
                        if (flag != 1)
                        {
                            mensaje.Text = "Motor 1 Listo";
                            mensaje.Background = new
SolidColorBrush(Color.FromRgb(247, 126, 5));



                            src = new Uri(@"/Images/motor1.png", UriKind.Relative);
                            img = new BitmapImage(src);
                            flag = 1;
                            imagen.Source = img;
                            m1.Play();
                            Thread.Sleep(1800);
                        }

                        break;

                    case "DOS":
                        if (flag != 2)
                        {
                            mensaje.Text = "Motor 2 Listo";
                            src = new Uri(@"/Images/motor2.png", UriKind.Relative);
                            img = new BitmapImage(src);
                            imagen.Source = img;
                            mensaje.Background = new
SolidColorBrush(Color.FromRgb(255, 0, 0));
                            flag = 2;
                            m2.Play();
                            Thread.Sleep(1400);
                        }
                        break;

                    case "TRES":
                        if (flag != 3)
                        {
                            mensaje.Text = "Motor 3 Listo";
                            src = new Uri(@"/Images/motor3.png", UriKind.Relative);
                            img = new BitmapImage(src);
                            imagen.Source = img;
                            mensaje.Background = new
SolidColorBrush(Color.FromRgb(5, 134, 247));
                            flag = 3;
                            m3.Play();
                            Thread.Sleep(1400);
                        }
                        break;
                    case "CUATRO":
                        if (flag != 4)
                        {
```

```
                            mensaje.Text = "Motor 4 Listo";
                            src = new Uri(@"/Images/motor4.png", UriKind.Relative);
                            img = new BitmapImage(src);
                            imagen.Source = img;
                            mensaje.Background = new
SolidColorBrush(Color.FromRgb(12, 247, 73));
                            flag = 4;
                            m4.Play();
                            Thread.Sleep(2000);
                        }
                        break;
                    case "CINCO":
                        if (flag != 5)
                        {
                            mensaje.Text = "Motor 5 Listo";
                            src = new Uri(@"/Images/motor5.png", UriKind.Relative);
                            img = new BitmapImage(src);
                            imagen.Source = img;
                            mensaje.Background = new
SolidColorBrush(Color.FromRgb(217, 12, 247));
                            flag = 5;
                            m5.Play();
                            Thread.Sleep(2000);
                        }
                        break;
                    case "SUBIR":
                        if (flag == 2 || flag == 3 || flag == 4)
                        {
                            if (flag == 2)
                            {
                                mensaje.Text = "Subiendo motor 2";
                                try
                                {
                                    _port.Write("4"); _port.Write("4");
                                }
                                catch (Exception)
                                {
                                }

                                subm2.Play();
                            }
                            if (flag == 3)
                            {
                                mensaje.Text = "Subiendo motor 3";
                                try
                                {
                                    _port.Write("6"); _port.Write("6");
                                }
                                catch (Exception)
                                {
                                }

                                subm3.Play();
                            }
                            if (flag == 4)
                            {
                                mensaje.Text = "Subiendo motor 4";
                                try
```

```
                {
                    _port.Write("8"); _port.Write("8");
                }
                catch (Exception)
                {
                }

                subm4.Play();
            }

            Thread.Sleep(2000);
        }
        else
        {
            mensaje.Text = "Combinación no válida";
            error.Play();
        }
        break;

    case "ACCION":

        break;

    case "BAJAR":
        if (flag == 2 || flag == 3 || flag == 4)
        {
            if (flag == 2)
            {
                mensaje.Text = "Bajando motor 2";
                try
                {
                    _port.Write("3"); _port.Write("3");
                }
                catch (Exception)
                {
                }

                bajm2.Play();
            }
            if (flag == 3)
            {
                mensaje.Text = "Bajando motor 3";
                try
                {
                    _port.Write("5"); _port.Write("5");
                }
                catch (Exception)
                {
                }

                bajm3.Play();
            }
            if (flag == 4)
            {
                mensaje.Text = "Bajando motor 4";
                try
                {
                    _port.Write("7"); _port.Write("7");
```

```
                }
                catch (Exception)
                {
                }

                bajm4.Play();
            }

            Thread.Sleep(2000);
        }
        else
        {
            mensaje.Text = "Combinación no válida";
            error.Play();
        }
        break;
    case "IZQUIERDA":
        if (flag == 5)
        {
            mensaje.Text = "Motor 5 hacia la izquierda";
            try
            {
                _port.Write("9"); _port.Write("9");
            }
            catch (Exception)
            {
            }

            izquierda.Play();
            Thread.Sleep(2000);
        }
        else
        {
            mensaje.Text = "Combinación no válida";
            error.Play();
        }
        break;
    case "DERECHA":
        if (flag == 5)
        {
            mensaje.Text = "Motor 5 hacia la derecha";
            try
            {
                _port.Write("0"); _port.Write("0");
            }
            catch (Exception)
            {
            }

            derecha.Play();
            Thread.Sleep(2000);
        }
        else
        {
            mensaje.Text = "Combinación no válida";
            error.Play();
        }
        break;
```

```
                    case "ABRIR":
                        if (flag == 1)
                        {
                            mensaje.Text = "Abriendo pinzas";
                            try
                            {
                                _port.Write("1");
                                _port.Write("1");
                            }
                            catch (Exception)
                            {
                            }

                            abrir.Play();
                            Thread.Sleep(2000);
                        }
                        else
                        {
                            mensaje.Text = "Combinación no válida";
                            error.Play();
                        }
                        break;
                    case "CERRAR":
                        if (flag == 1)
                        {
                            mensaje.Text = "Cerrando pinzas";
                            try
                            {
                                _port.Write("2");
                                _port.Write("2");
                            }
                            catch (Exception)
                            {
                            }

                            cerrar.Play();
                            Thread.Sleep(2000);
                        }
                        else
                        {
                            mensaje.Text = "Combinación no válida";
                            error.Play();
                        }
                        break;
                    case "ADIOS":

                        mensaje.Text = "Cerrando modulo";
                        adios.Play();
                        Thread.Sleep(520);
                        this._Kinect.AudioSource.Stop();
                        this._Kinect.Stop();
                        bloquearpantalla();
                        (Application.Current.MainWindow.FindName("_mainFrame") as
Frame).Source = new Uri("Inicio.xaml", UriKind.Relative);

                        break;
```

```
                    default:
                        //En caso de que no solo contenga una palabra tambien
realizaremos un switch para ver si la frase corresponde a alguna de nuestros valores
de opcion

                        mensaje.Text = "No se reconocio el comando";

                        break;

                }
            }
        }
    }
}
```

16.6 Anexo 3.2 Código Audio.xaml

```
<Page x:Class="WpfApplication1.Audio"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"


              xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
      d:DesignHeight="673.2" d:DesignWidth="1304.4"
  Title="Page">

    <Grid Canvas.Top="-16" x:Name="LayoutRoot">

        <Canvas Margin="0,0,10,0">
            <Label Content="Mover Brazo Robótico con comandos de voz"
Canvas.Left="204" Canvas.Top="36" Width="927" FontSize="40" FontStyle="Oblique"
FontWeight="ExtraBlack" Foreground="White"/>
            <Controls:HoverButton Margin="0" Padding="0" x:Name="kinectButton"
ImageSize="50"

ImageSource="/KinectTesis;component/Images/myhand.png"

ActiveImageSource="/KinectTesis;component/Images/myhand.png"
                                  TimeInterval="4000"            Panel.ZIndex="1000"
Canvas.Left="0" Canvas.Top="0" Visibility="Hidden" />
            <Image    Height="403"    HorizontalAlignment="Left"    Name="imagen"
Stretch="Fill"     VerticalAlignment="Top"     Width="677"     Canvas.Left="363"
Canvas.Top="119" Source="/KinectTesis;component/Images/robot.png"  />
            <TextBlock Height="100" Name="mensaje" Text="Reconocimiento de Voz
KiBot"     VerticalAlignment="Center"     Width="824"     Background="#FF3484FF"
Foreground="White"     FontWeight="Bold"     FontSize="30"     TextWrapping="Wrap"
TextTrimming="None" FontStretch="UltraExpanded" Canvas.Left="307" Canvas.Top="544"
TextAlignment="Center" />
            <TextBlock   Height="30"   HorizontalAlignment="Left"   Name="statusK"
Text="Kinect  Status"  VerticalAlignment="Top"  Width="100"  FontWeight="Light"
FontSize="16" Canvas.Left="1184" Canvas.Top="119" Foreground="White"/>
        </Canvas>
    </Grid>
```

```
    </Page>
```

16.7 Anexo 4.1 Código Movimiento.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.IO.Ports;
using Microsoft.Kinect;

namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for Page.xaml
    /// </summary>
    public partial class Movimiento : Page
    {

        private readonly SerialPort _port;

        #region "Variables Kinect"
        private KinectSensor _Kinect;
        private Skeleton[] FrameSkeletons;

        List<Button> buttons;
        static Button selected;
        private WriteableBitmap _ColorImageBitmap;
        private Int32Rect _ColorImageBitmapRect;
        private int _ColorImageStride;

        float handX;
        float handY;
        private WriteableBitmap colorBitmap;


        private byte[] colorPixels;
        private Boolean bandera = true;
        DateTime tiempoInicio = DateTime.Now;

        Joint hombroI;
        Joint hombroD;
        Joint manoI;
        Joint manoD;
        Joint cabeza;
        Joint pieI;
```

```
        Joint pieD;
        Joint rodillaD;
        Joint rodillaI;
        Joint codoD;
        Joint codoI;

        #endregion

        public Movimiento()
        {
            InitializeComponent();
            InitializeButtons();
            mov.Content = "Ninguno";
            Generics.ResetHandPosition(kinectButton);
            kinectButton.Click += new RoutedEventHandler(kinectButton_Click);
            DiscoverKinectSensor();
            _port = new SerialPort("COM12") { BaudRate = 9600 };
            try
            {
                _port.Open();
            }
            catch (Exception)
            {
            }
        }

        void Gestos(Skeleton skeleton)
        {
            hombroI = skeleton.Joints[JointType.ShoulderLeft];
            hombroD = skeleton.Joints[JointType.ShoulderRight];
            manoI = skeleton.Joints[JointType.HandLeft];
            manoD = skeleton.Joints[JointType.HandRight];
            cabeza = skeleton.Joints[JointType.Head];
            pieI = skeleton.Joints[JointType.FootLeft];
            pieD = skeleton.Joints[JointType.FootRight];
            rodillaD = skeleton.Joints[JointType.KneeRight];
            rodillaI = skeleton.Joints[JointType.KneeLeft];
            codoD = skeleton.Joints[JointType.ElbowRight];
            codoI = skeleton.Joints[JointType.ElbowLeft];


            //mov.Content="H: "+hombroD.Position.Y+" C: "+codoD.Position.Y+" M:
"+manoD.Position.Y;
            //movI.Content = "H: " + hombroI.Position.Y + " C: " + codoI.Position.Y
+ " M: " + manoI.Position.Y;

            if (bandera == true)
            {
                //m1 abrir
                if (manoI.Position.Y > cabeza.Position.Y && manoD.Position.Y >
cabeza.Position.Y &&
                    manoI.Position.X < cabeza.Position.X && manoD.Position.X >
cabeza.Position.X)
                {
                    mov.Content = "m1 abrir";
                    try
                    {
                        _port.Write("1");
```

```
                    }
                    catch (Exception)
                    {
                    }

                    bandera = false;
                }
                else
                {
                    if (manoD.Position.X < manoI.Position.X && codoD.Position.Y >
manoD.Position.Y && codoI.Position.Y > manoI.Position.Y)
                    {
                        mov.Content = "m1 cerrar";
                        try
                        {
                            _port.Write("2");
                        }
                        catch (Exception)
                        {
                        }

                        bandera = false;
                    }
                    else
                    {
                        if (manoI.Position.Y > hombroI.Position.Y &&
Math.Abs(manoI.Position.Z - hombroI.Position.Z) < 0.2 &&
                            Math.Abs(codoD.Position.Z - manoD.Position.Z) > 0.25 &&
Math.Abs(hombroD.Position.X - manoD.Position.X) < 0.1)
                        {
                            mov.Content = "m2 bajar";
                            try
                            {
                                _port.Write("3");
                            }
                            catch (Exception)
                            {
                            }

                            bandera = false;
                        }

                        else
                        {
                            if (manoD.Position.Y > hombroD.Position.Y &&
Math.Abs(manoD.Position.Z - hombroD.Position.Z) < 0.2 &&
                                Math.Abs(codoI.Position.Z - manoI.Position.Z) > 0.25 &&
Math.Abs(hombroI.Position.X - manoI.Position.X) < 0.1)
                            {
                                mov.Content = "m2 subir";
                                try
                                {
                                    _port.Write("4");
                                }
                                catch (Exception)
                                {
                                }
```

```
                                bandera = false;
                            }
                            else
                            {
                                if (Math.Abs(manoI.Position.X - codoI.Position.X) >
0.25 && Math.Abs(manoI.Position.Z - hombroI.Position.Z) < 0.1
                                    && Math.Abs(manoI.Position.Y -
hombroI.Position.Y) < 0.1 &&
                                    manoD.Position.Y > hombroD.Position.Y+0.1 &&
Math.Abs(manoD.Position.Z - hombroD.Position.Z) < 0.2)
                                {
                                    mov.Content = "m3 subir ";
                                    try
                                    {
                                        _port.Write("6");
                                    }
                                    catch (Exception)
                                    {
                                    }

                                    bandera = false;
                                }
                                else
                                {
                                    if (Math.Abs(manoD.Position.X -
codoD.Position.X) > 0.25 && Math.Abs(manoD.Position.Z - hombroD.Position.Z) < 0.1
                                    && Math.Abs(manoD.Position.Y -
hombroD.Position.Y) < 0.1 &&
                                    manoI.Position.Y > hombroI.Position.Y + 0.1 &&
Math.Abs(manoI.Position.Z - hombroI.Position.Z) < 0.2)
                                    {
                                        mov.Content = "m3 bajar";
                                        try
                                        {
                                            _port.Write("5");
                                        }
                                        catch (Exception)
                                        {
                                        }

                                        bandera = false;
                                    }
                                    else
                                    {
                                        if (Math.Abs(manoD.Position.X -
codoD.Position.X) > 0.2 && Math.Abs(manoD.Position.Z - hombroD.Position.Z) < 0.1
                                            && Math.Abs(hombroD.Position.Y -
manoD.Position.Y) < 0.1
                                            && Math.Abs(manoI.Position.X -
codoI.Position.X) > 0.2 && Math.Abs(manoI.Position.Z - hombroI.Position.Z) < 0.1
                                            && Math.Abs(hombroI.Position.Y -
manoI.Position.Y) < 0.1)
                                        {
                                            mov.Content = "m4 bajar";
                                            try
                                            {
                                                _port.Write("7");
                                            }
```

```
                                        catch (Exception)
                                        {
                                        }

                                        bandera = false;
                                    }
                                    else
                                    {
                                        if (Math.Abs(codoI.Position.Z -
manoI.Position.Z) > 0.2 && Math.Abs(hombroI.Position.X - manoI.Position.X) < 0.1
                                            && Math.Abs(hombroI.Position.Y -
manoI.Position.Y) < 0.1 &&
                                            Math.Abs(codoD.Position.Z -
manoD.Position.Z) > 0.2 && Math.Abs(hombroD.Position.X - manoD.Position.X) < 0.1
                                            && Math.Abs(hombroD.Position.Y -
manoD.Position.Y) < 0.1)
                                        {
                                            mov.Content = "m4 subir";
                                            try
                                            {
                                                _port.Write("8");
                                            }
                                            catch (Exception)
                                            {
                                            }

                                            bandera = false;
                                        }
                                        else
                                        {
                                            if (Math.Abs(manoD.Position.X -
codoD.Position.X) > 0.25 && Math.Abs(manoD.Position.Z - hombroD.Position.Z) < 0.1
                                                && Math.Abs(manoD.Position.Y -
hombroD.Position.Y) < 0.1
                                                && Math.Abs(pieD.Position.X -
rodillaD.Position.X) > 0.1 )
                                             {
                                              mov.Content = "m5 derecha";
                                              try
                                              {
                                                  _port.Write("0");
                                              }
                                              catch (Exception)
                                              {
                                              }

                                                 bandera = false;
                                             }
                                             else
                                             {
                                                 if (Math.Abs(manoI.Position.X -
codoI.Position.X) > 0.25 && Math.Abs(manoI.Position.Z - hombroI.Position.Z) < 0.1
                                                     && Math.Abs(manoI.Position.Y -
hombroI.Position.Y) < 0.1
                                                     && pieI.Position.X -
rodillaI.Position.X < -0.1)
                                                {
                                                mov.Content = "m5 izquierda";
```

```
                                        try
                                        {
                                            _port.Write("9");
                                        }
                                        catch (Exception)
                                        {
                                        }

                                        bandera = false;
                                        }
                                    }
                                }
                            }
                        }

                    }
                }
            }
        }

    }


        #region "Movimiento de mano por pantalla"

        //Listado de botones
        private void InitializeButtons()
        {
            buttons = new List<Button> { BACKHOME};
        }

        //Evento para cambio del estado de kinect (saber si hay kinect)
        private void DiscoverKinectSensor()
        {
            KinectSensor.KinectSensors.StatusChanged += KinectSensors_StatusChanged;
            this.Kinect = KinectSensor.KinectSensors.FirstOrDefault(x => x.Status ==
KinectStatus.Connected);
        }

        //Asociación de eventos producidos por la kinect
        private void bloquearpantalla()
        {
            try
            {
                KinectSensor.KinectSensors.StatusChanged -=
KinectSensors_StatusChanged;
                this.Kinect.SkeletonFrameReady -= Kinect_SkeletonFrameReady;
                kinectButton.Click -= new RoutedEventHandler(kinectButton_Click);
                this.Kinect.ColorFrameReady -= Kinect_ColorFrameReady;
            }
            catch (Exception)
            {

            }
        }
```

```
        //Analiza estado de kinect (hay o no kinect)
        private void KinectSensors_StatusChanged(object sender,
StatusChangedEventArgs e)
        {
            switch (e.Status)
            {
                case KinectStatus.Connected:
                    if (this.Kinect == null)
                    {
                        this.Kinect = e.Sensor;
                    }
                    break;
                case KinectStatus.Disconnected:
                    if (this.Kinect == e.Sensor)
                    {
                        this.Kinect = null;
                        this.Kinect = KinectSensor.KinectSensors.FirstOrDefault(x =>
x.Status == KinectStatus.Connected);
                        if (this.Kinect == null)
                        {
                            MessageBox.Show("Kinect Desconectada.");
                        }
                    }
                    break;
            }
        }

        //Selecciona kinect
        public KinectSensor Kinect
        {
            get { return this._Kinect; }
            set
            {
                if (this._Kinect != value)
                {
                    if (this._Kinect != null)
                    {
                        this._Kinect = null;
                    }
                    if (value != null && value.Status == KinectStatus.Connected)
                    {
                        this._Kinect = value;
                        InitializeKinectSensor(this._Kinect);
                    }
                }
            }
        }


        //Inicio de kinect
        //Muestra imagen producida por kinect
        //Detecta el cuerpo
        private void InitializeKinectSensor(KinectSensor kinectSensor)
        {
            if (kinectSensor != null)
            {
                ColorImageStream colorStream = kinectSensor.ColorStream;
```

```
                colorStream.Enable();
                this._ColorImageBitmap = new WriteableBitmap(colorStream.FrameWidth,
colorStream.FrameHeight,
                    96, 96, PixelFormats.Bgr32, null);
                this._ColorImageBitmapRect = new Int32Rect(0, 0,
colorStream.FrameWidth, colorStream.FrameHeight);
                this._ColorImageStride = colorStream.FrameWidth *
colorStream.FrameBytesPerPixel;

                //Mostrar imagen por pantalla
                this.colorPixels = new
byte[kinectSensor.ColorStream.FramePixelDataLength];
                this.colorBitmap = new
WriteableBitmap(kinectSensor.ColorStream.FrameWidth,
kinectSensor.ColorStream.FrameHeight, 96.0, 96.0, PixelFormats.Bgr32, null);
                this.Camara.Source = this.colorBitmap;

                kinectSensor.ColorFrameReady += SensorColorFrameReady;

                kinectSensor.SkeletonStream.Enable(new TransformSmoothParameters()
                {
                    Correction = 0.5f,
                    JitterRadius = 0.05f,
                    MaxDeviationRadius = 0.04f,
                    Smoothing = 0.5f
                });


                kinectSensor.SkeletonFrameReady += Kinect_SkeletonFrameReady;
                kinectSensor.ColorFrameReady += Kinect_ColorFrameReady;
                if (!kinectSensor.IsRunning)
                {
                    kinectSensor.Start();
                }

                this.FrameSkeletons = new
Skeleton[this.Kinect.SkeletonStream.FrameSkeletonArrayLength];

            }
        }

        private void Kinect_ColorFrameReady(object sender,
ColorImageFrameReadyEventArgs e)
        {
            using (ColorImageFrame frame = e.OpenColorImageFrame())
            {
                if (frame != null)
                {
                    byte[] pixelData = new byte[frame.PixelDataLength];
                    frame.CopyPixelDataTo(pixelData);
                    this._ColorImageBitmap.WritePixels(this._ColorImageBitmapRect,
pixelData,
                    this._ColorImageStride, 0);
                }
            }
        }

        //DEteccion del cuerpo y la mano a ocupar para manejar el menu
```

```
        private void Kinect_SkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
        {
            using (SkeletonFrame frame = e.OpenSkeletonFrame())
            {
                if (frame != null)
                {
                    frame.CopySkeletonDataTo(this.FrameSkeletons);
                    Skeleton skeleton = GetPrimarySkeleton(this.FrameSkeletons);

                    if (skeleton == null)
                    {
                        kinectButton.Visibility = Visibility.Collapsed;
                    }
                    else
                    {
                        //MANEJO SOLO CON MANO DERECHA
                        Joint primaryHand = skeleton.Joints[JointType.HandRight];
                        TrackHand(primaryHand);

                        //Tiempo para buscar gestos
                        DateTime tiempoFin = DateTime.Now;
                        TimeSpan transcurrido = tiempoFin.Subtract(tiempoInicio);
                        if (transcurrido.Milliseconds > 450)
                       {
                            bandera = true;
                            tiempoInicio = DateTime.Now;
                            Gestos(skeleton);
                       }

                    }
                }
            }
        }

        //Manejo de mano
        private void TrackHand(Joint hand)
        {
            if (hand.TrackingState == JointTrackingState.NotTracked)
            {
                kinectButton.Visibility = System.Windows.Visibility.Collapsed;
            }
            else
            {
                kinectButton.Visibility = System.Windows.Visibility.Visible;

                DepthImagePoint point =
this.Kinect.CoordinateMapper.MapSkeletonPointToDepthPoint(hand.Position,
DepthImageFormat.Resolution640x480Fps30);
                handX = (int)((point.X * LayoutRoot.ActualWidth /
this.Kinect.DepthStream.FrameWidth) -
                    (kinectButton.ActualWidth / 2.0));
                handY = (int)((point.Y * LayoutRoot.ActualHeight /
this.Kinect.DepthStream.FrameHeight) -
                    (kinectButton.ActualHeight / 2.0));
                Canvas.SetLeft(kinectButton, handX);
                Canvas.SetTop(kinectButton, handY);
```

```
                if (isHandOver(kinectButton, buttons)) kinectButton.Hovering();
                else kinectButton.Release();
                if (hand.JointType == JointType.HandRight)
                {
                    kinectButton.ImageSource =
"/KinectTesis;component/Images/myhand.png";
                    kinectButton.ActiveImageSource =
"/KinectTesis;component/Images/myhand.png";
                }
                else
                {
                    kinectButton.ImageSource =
"/KinectTesis;component/Images/myhand.png";
                    kinectButton.ActiveImageSource =
"/KinectTesis;component/Images/myhand.png";
                }
            }
        }

        //Detecta la posicion de la mano y carga el boton
        private bool isHandOver(FrameworkElement hand, List<Button> buttonslist)
        {
            var handTopLeft = new Point(Canvas.GetLeft(hand), Canvas.GetTop(hand));
            var handX = handTopLeft.X + hand.ActualWidth / 2;
            var handY = handTopLeft.Y + hand.ActualHeight / 2;

            foreach (Button target in buttonslist)
            {

                if (target != null)
                {
                    Point targetTopLeft = new Point(Canvas.GetLeft(target),
Canvas.GetTop(target));
                    if (handX > targetTopLeft.X &&
                        handX < targetTopLeft.X + target.Width &&
                        handY > targetTopLeft.Y &&
                        handY < targetTopLeft.Y + target.Height)
                    {
                        selected = target;
                        return true;
                    }
                }
            }
            return false;
        }


        //Obtiene el cuerpo
        private static Skeleton GetPrimarySkeleton(Skeleton[] skeletons)
        {
            Skeleton skeleton = null;
            if (skeletons != null)
            {
                for (int i = 0; i < skeletons.Length; i++)
                {
                    if (skeletons[i].TrackingState == SkeletonTrackingState.Tracked)
                    {
                        if (skeleton == null)
```

```
                        {
                            skeleton = skeletons[i];
                        }
                        else
                        {
                            if (skeleton.Position.Z > skeletons[i].Position.Z)
                            {
                                skeleton = skeletons[i];
                            }
                        }
                    }
                }
            }
            return skeleton;
        }



        #endregion

        private void BACKHOME_Click(object sender, RoutedEventArgs e)
        {
            _port.Close();
            bloquearpantalla();
            (Application.Current.MainWindow.FindName("_mainFrame") as Frame).Source
= new Uri("Inicio.xaml", UriKind.Relative);
        }

        void kinectButton_Click(object sender, RoutedEventArgs e)
        {
            selected.RaiseEvent(new RoutedEventArgs(Button.ClickEvent, selected));
        }


        #region mostrar imagen en cam

        private void SensorColorFrameReady(object sender,
ColorImageFrameReadyEventArgs e)
        {
            using (ColorImageFrame colorFrame = e.OpenColorImageFrame())
            {
                if (colorFrame != null)
                {
                    // Copy the pixel data from the image to a temporary array
                    colorFrame.CopyPixelDataTo(this.colorPixels);

                    // Write the pixel data into our bitmap

                    this.colorBitmap.WritePixels(
                        new Int32Rect(0, 0, this.colorBitmap.PixelWidth,
this.colorBitmap.PixelHeight),
                        this.colorPixels,
                        this.colorBitmap.PixelWidth * sizeof(int),
                        0);
                }
            }
        }
```

```
    #endregion
}
 }
```

16.8 Anexo 4.2 Código Movimiento.xaml

```
<Page x:Class="WpfApplication1.Movimiento"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"


      xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
 d:DesignHeight="673.2" d:DesignWidth="1304.4"
  Title="Page"
>
    <Grid Canvas.Top="-16" x:Name="LayoutRoot">

        <Canvas Margin="-6,0,10,0">
            <Viewbox  Grid.Row="1"  Stretch="Uniform"  HorizontalAlignment="Center"
Canvas.Left="492" Canvas.Top="141" Height="440" Width="763"/>
            <Button      x:Name="BACKHOME"      Content="Volver"      Height="100"
Canvas.Left="84"      Canvas.Top="527"      Width="328"      Click="BACKHOME_Click"
Background="#FFBBD71A" FontSize="48"  />
            <Controls:HoverButton  Margin="0"  Padding="0"  x:Name="kinectButton"
ImageSize="50"

ImageSource="/KinectTesis;component/Images/myhand.png"

ActiveImageSource="/KinectTesis;component/Images/myhand.png"
                                TimeInterval="4000"            Panel.ZIndex="1000"
Canvas.Left="0" Canvas.Top="0" />
            <Label  Content="Mover  Brazo  Robótico  con  gestos"  Canvas.Left="320"
Canvas.Top="34"        Width="783"        FontSize="40"        FontStyle="Oblique"
FontWeight="ExtraBlack" Foreground="White"/>
            <Label  Content="Ultimo  movimiento:"  Canvas.Left="67"  Canvas.Top="181"
Height="55" Width="345" FontSize="30" Foreground="White"/>
            <Label  x:Name="mov"  Content=""  Canvas.Left="130"  Canvas.Top="260"
Width="319" Height="48" FontSize="30" Foreground="White"/>
            <Label  x:Name="movI"  Content=""  Canvas.Left="78"  Canvas.Top="355"
Width="362" Height="48" FontSize="30" Foreground="White"/>
            <Image  x:Name="Camara"  Height="440"  Width="838"  Canvas.Left="417"
Canvas.Top="141"/>

        </Canvas>
    </Grid>
</Page>
```

16.9 Anexo 5.1 Código Calibrar.xaml.cs

```
using Microsoft.Kinect;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfApplication1
{

    public partial class Calibrar : Page
    {

        #region "Variables Kinect"
        private KinectSensor _Kinect;
        private Skeleton[] FrameSkeletons;
        private WriteableBitmap _ColorImageBitmap;
        private Int32Rect _ColorImageBitmapRect;
        private int _ColorImageStride;

        List<Button> buttons;
        static Button selected;

        float handX;
        float handY;
        private WriteableBitmap colorBitmap;

        private byte[] colorPixels;
        private int bandera = 0;
        #endregion



        public Calibrar()
        {
            InitializeComponent();
            InitializeButtons();
            Generics.ResetHandPosition(kinectButton);
            kinectButton.Click += new RoutedEventHandler(kinectButton_Click);
            DiscoverKinectSensor();

        }



        #region "Movimiento de mano por pantalla"
```

```
        private void InitializeButtons()
        {
            buttons = new List<Button> { BACKHOME, grado1, grado2, grado3, grado4,
grado5, grado6, grado7 };
        }

        //Evento para cambio del estado de kinect (saber si hay kinect)
        private void DiscoverKinectSensor()
        {
            KinectSensor.KinectSensors.StatusChanged += KinectSensors_StatusChanged;
            this.Kinect = KinectSensor.KinectSensors.FirstOrDefault(x => x.Status ==
KinectStatus.Connected);

        }


        //Analiza estado de kinect (hay o no kinect)
        private void KinectSensors_StatusChanged(object sender,
StatusChangedEventArgs e)
        {
            switch (e.Status)
            {
                case KinectStatus.Connected:
                    if (this.Kinect == null)
                    {
                        this.Kinect = e.Sensor;
                    }
                    break;
                case KinectStatus.Disconnected:
                    if (this.Kinect == e.Sensor)
                    {
                        this.Kinect = null;
                        this.Kinect = KinectSensor.KinectSensors.FirstOrDefault(x =>
x.Status == KinectStatus.Connected);
                        if (this.Kinect == null)
                        {
                            MessageBox.Show("Kinect Desconectada.");
                        }
                    }
                    break;
            }
        }

        //Selecciona kinect
        public KinectSensor Kinect
        {
            get { return this._Kinect; }
            set
            {
                if (this._Kinect != value)
                {
                    if (this._Kinect != null)
                    {
                        this._Kinect = null;
                    }
                    if (value != null && value.Status == KinectStatus.Connected)
                    {
```

```
                    this._Kinect = value;
                    InitializeKinectSensor(this._Kinect);
                }
            }
        }
    }


    //Inicio de kinect
    //Muestra imagen producida por kinect
    //Detecta el cuerpo
    private void InitializeKinectSensor(KinectSensor kinectSensor)
    {
        try
        {
            ValorAngulo.Content = this.Kinect.ElevationAngle;
        }
        catch (Exception)
        {
        }
        if (kinectSensor != null)
        {
            ColorImageStream colorStream = kinectSensor.ColorStream;
            colorStream.Enable();
            this._ColorImageBitmap = new WriteableBitmap(colorStream.FrameWidth,
colorStream.FrameHeight,
                96, 96, PixelFormats.Bgr32, null);
            this._ColorImageBitmapRect = new Int32Rect(0, 0,
colorStream.FrameWidth, colorStream.FrameHeight);
            this._ColorImageStride = colorStream.FrameWidth *
colorStream.FrameBytesPerPixel;

            //Para camara de campo visual
            this.colorPixels = new
byte[kinectSensor.ColorStream.FramePixelDataLength];
            this.colorBitmap = new
WriteableBitmap(kinectSensor.ColorStream.FrameWidth,
kinectSensor.ColorStream.FrameHeight, 96.0, 96.0, PixelFormats.Bgr32, null);
            this.Camara.Source = this.colorBitmap;

            kinectSensor.ColorFrameReady += SensorColorFrameReady;

            kinectSensor.SkeletonStream.Enable(new TransformSmoothParameters()
            {
                Correction = 0.5f,
                JitterRadius = 0.05f,
                MaxDeviationRadius = 0.04f,
                Smoothing = 0.5f
            });

            kinectSensor.SkeletonFrameReady += Kinect_SkeletonFrameReady;
            kinectSensor.ColorFrameReady += Kinect_ColorFrameReady;
            if (!kinectSensor.IsRunning)
            {
                kinectSensor.Start();
            }
```

```
                this.FrameSkeletons = new
Skeleton[this.Kinect.SkeletonStream.FrameSkeletonArrayLength];

            }
        }

        private void Kinect_ColorFrameReady(object sender,
ColorImageFrameReadyEventArgs e)
        {
            using (ColorImageFrame frame = e.OpenColorImageFrame())
            {
                if (frame != null)
                {
                    byte[] pixelData = new byte[frame.PixelDataLength];
                    frame.CopyPixelDataTo(pixelData);
                    this._ColorImageBitmap.WritePixels(this._ColorImageBitmapRect,
pixelData,
                    this._ColorImageStride, 0);
                }
            }
        }

        //DEteccion del cuerpo y la mano a ocupar para manejar el menu
        private void Kinect_SkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
        {
            using (SkeletonFrame frame = e.OpenSkeletonFrame())
            {
                if (frame != null)
                {
                    frame.CopySkeletonDataTo(this.FrameSkeletons);
                    Skeleton skeleton = GetPrimarySkeleton(this.FrameSkeletons);

                    if (skeleton == null)
                    {
                        kinectButton.Visibility = Visibility.Collapsed;
                    }
                    else
                    {
                        kinectButton.Visibility = Visibility.Visible;
                        //MANEJO SOLO CON MANO DERECHA
                        Joint primaryHand = skeleton.Joints[JointType.HandRight];
                        TrackHand(primaryHand);
                        if (bandera == 0)
                        {
                            ValorAngulo.Content = _Kinect.ElevationAngle;
                            bandera = 1;
                        }
                    }
                }
            }
        }

        //Manejo de mano
        private void TrackHand(Joint hand)
        {
            if (hand.TrackingState == JointTrackingState.NotTracked)
            {
```

```
                kinectButton.Visibility = System.Windows.Visibility.Collapsed;
            }
            else
            {
                kinectButton.Visibility = System.Windows.Visibility.Visible;

                DepthImagePoint point =
this.Kinect.CoordinateMapper.MapSkeletonPointToDepthPoint(hand.Position,
DepthImageFormat.Resolution640x480Fps30);
                handX = (int)((point.X * LayoutRoot.ActualWidth /
this.Kinect.DepthStream.FrameWidth) -
                    (kinectButton.ActualWidth / 2.0));
                handY = (int)((point.Y * LayoutRoot.ActualHeight /
this.Kinect.DepthStream.FrameHeight) -
                    (kinectButton.ActualHeight / 2.0));
                Canvas.SetLeft(kinectButton, handX);
                Canvas.SetTop(kinectButton, handY);

                if (isHandOver(kinectButton, buttons)) kinectButton.Hovering();
                else kinectButton.Release();
                if (hand.JointType == JointType.HandRight)
                {
                    kinectButton.ImageSource =
"/KinectTesis;component/Images/myhand.png";
                    kinectButton.ActiveImageSource =
"/KinectTesis;component/Images/myhand.png";
                }
                else
                {
                    kinectButton.ImageSource =
"/KinectTesis;component/Images/myhand.png";
                    kinectButton.ActiveImageSource =
"/KinectTesis;component/Images/myhand.png";
                }
            }
        }

        //Detecta la posicion de la mano y carga el boton
        private bool isHandOver(FrameworkElement hand, List<Button> buttonslist)
        {
            var handTopLeft = new Point(Canvas.GetLeft(hand), Canvas.GetTop(hand));
            var handX = handTopLeft.X + hand.ActualWidth / 2;
            var handY = handTopLeft.Y + hand.ActualHeight / 2;

            foreach (Button target in buttonslist)
            {

                if (target != null)
                {
                    Point targetTopLeft = new Point(Canvas.GetLeft(target),
Canvas.GetTop(target));
                    if (handX > targetTopLeft.X &&
                        handX < targetTopLeft.X + target.Width &&
                        handY > targetTopLeft.Y &&
                        handY < targetTopLeft.Y + target.Height)
                    {
                        selected = target;
                        return true;
```

```
                }
            }
        }
        return false;
    }


    //Obtiene el cuerpo
    private static Skeleton GetPrimarySkeleton(Skeleton[] skeletons)
    {
        Skeleton skeleton = null;
        if (skeletons != null)
        {
            for (int i = 0; i < skeletons.Length; i++)
            {
                if (skeletons[i].TrackingState == SkeletonTrackingState.Tracked)
                {
                    if (skeleton == null)
                    {
                        skeleton = skeletons[i];
                    }
                    else
                    {
                        if (skeleton.Position.Z > skeletons[i].Position.Z)
                        {
                            skeleton = skeletons[i];
                        }
                    }
                }
            }
        }
        return skeleton;
    }



    #endregion


    private void BACKHOME_Click(object sender, RoutedEventArgs e)
    {
        bloquearpantalla();
        (Application.Current.MainWindow.FindName("_mainFrame") as Frame).Source
= new Uri("Inicio.xaml", UriKind.Relative);
    }

    void kinectButton_Click(object sender, RoutedEventArgs e)
    {
        selected.RaiseEvent(new RoutedEventArgs(Button.ClickEvent, selected));
    }

    private void bloquearpantalla()
    {
        try
        {
            KinectSensor.KinectSensors.StatusChanged -=
KinectSensors_StatusChanged;
            this.Kinect.SkeletonFrameReady -= Kinect_SkeletonFrameReady;
```

```
                kinectButton.Click -= new RoutedEventHandler(kinectButton_Click);
                this.Kinect.ColorFrameReady -= Kinect_ColorFrameReady;
            }
            catch (Exception)
            {
            }
        }

        //PANTALLA
        private void SensorColorFrameReady(object sender,
ColorImageFrameReadyEventArgs e)
        {
            using (ColorImageFrame colorFrame = e.OpenColorImageFrame())
            {
                if (colorFrame != null)
                {
                    colorFrame.CopyPixelDataTo(this.colorPixels);

                    this.colorBitmap.WritePixels(
                        new Int32Rect(0, 0, this.colorBitmap.PixelWidth,
this.colorBitmap.PixelHeight),
                        this.colorPixels,
                        this.colorBitmap.PixelWidth * sizeof(int),
                        0);
                }
            }
      }

        #region eventos cambio de grado
        private void RadioButton_Checked(object sender, RoutedEventArgs e)
        {
            try
            {
                ValorAngulo.Content = 25 + "°";
                _Kinect.ElevationAngle = 25;

            }
            catch (Exception)
            {

            }
        }

        private void RadioButton_Checked_1(object sender, RoutedEventArgs e)
        {
            try
            {
                ValorAngulo.Content = 20 + "°";
                _Kinect.ElevationAngle = 20;
            }
            catch (Exception)
            {

            }
        }

        private void RadioButton_Checked_2(object sender, RoutedEventArgs e)
        {
```

```
        try
        {
            ValorAngulo.Content = 10 + "°";
            _Kinect.ElevationAngle = 10;

        }
        catch (Exception)
        {

        }
    }

    private void RadioButton_Checked_3(object sender, RoutedEventArgs e)
    {
        try
        {
            ValorAngulo.Content = 0 + "°";
            _Kinect.ElevationAngle = 0;

        }
        catch (Exception)
        {

        }
    }

    private void RadioButton_Checked_4(object sender, RoutedEventArgs e)
    {
        try
        {
            ValorAngulo.Content = -10 + "°";
            _Kinect.ElevationAngle = -10;

        }
        catch (Exception)
        {

        }
    }

    private void RadioButton_Checked_5(object sender, RoutedEventArgs e)
    {
        try
        {
            ValorAngulo.Content = -20 + "°";
            _Kinect.ElevationAngle = -20;

        }
        catch (Exception)
        {

        }
    }

    private void RadioButton_Checked_6(object sender, RoutedEventArgs e)
    {
        try
        {
```

```
            ValorAngulo.Content = -25 + "°";
            _Kinect.ElevationAngle = -25;

        }
        catch (Exception)
        {

        }
    }

#endregion
}
 }
```

16.10 Anexo 5.2 Código Calibrar.xaml

```
<Page x:Class="WpfApplication1.Calibrar"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:Controls="clr-
namespace:Coding4Fun.Kinect.Wpf.Controls;assembly=Coding4Fun.Kinect.Wpf"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
 d:DesignHeight="673.2" d:DesignWidth="1304.4"
  Title="Page">
    <Grid Canvas.Top="-16" x:Name="LayoutRoot">

        <Canvas Margin="0,0,3.6,0" >
            <Button      x:Name="BACKHOME"      Content="Volver"      Height="100"
Canvas.Left="962"      Canvas.Top="573"      Width="328"      Click="BACKHOME_Click"
Background="#FFBBD71A" FontSize="48"  />
            <Label  Content="Calibrar  Camara"  Canvas.Left="436"  Canvas.Top="46"
Width="783"      FontSize="40"      FontStyle="Oblique"      FontWeight="ExtraBlack"
Foreground="White"/>
            <Viewbox  Grid.Row="1"  Stretch="Uniform"  HorizontalAlignment="Center"
Canvas.Left="77" Canvas.Top="109" Height="429" Width="919">
                <Image      Name="Camara"      Height="325"      Canvas.Left="364"
Canvas.Top="270" Width="727"/>
            </Viewbox>
            <Label      Name="ValorAngulo"      Content=""      Canvas.Left="1057"
Canvas.Top="279" FontSize="24" Width="79" Foreground="White"/>
            <Label     x:Name="ValorAngulo_Copy"     Content="Grados     Actuales"
Canvas.Left="969" Canvas.Top="224" FontSize="30" Width="286" Foreground="White"/>
            <Controls:HoverButton  Margin="0"  Padding="0"  x:Name="kinectButton"
ImageSize="50"

ImageSource="/KinectTesis;component/Images/myhand.png"

ActiveImageSource="/KinectTesis;component/Images/myhand.png"
                                TimeInterval="4000"             Panel.ZIndex="1000"
Canvas.Left="0" Canvas.Top="0" />
            <Button x:Name="grado1" Content="25°" Canvas.Left="77" Canvas.Top="574"
Width="100"       Height="80"       FontSize="35"       Click="RadioButton_Checked"
Background="LightGreen"/>
```

```
            <Button       x:Name="grado2"       Content="20°"       Canvas.Left="204"
Canvas.Top="573"           Width="100"           Height="80"           FontSize="35"
Click="RadioButton_Checked_1" Background="LightGreen"/>
            <Button       x:Name="grado3"       Content="10°"       Canvas.Left="329"
Canvas.Top="573"           Width="100"           Height="80"           FontSize="35"
Click="RadioButton_Checked_2" Background="LightGreen"/>
            <Button x:Name="grado4" Content="0°" Canvas.Left="447" Canvas.Top="573"
Width="100"       Height="80"       FontSize="35"       Click="RadioButton_Checked_3"
Background="LightGreen"/>
            <Button       x:Name="grado5"       Content="-10°"       Canvas.Left="567"
Canvas.Top="573"           Width="100"           Height="80"           FontSize="35"
Click="RadioButton_Checked_4" Background="LightGreen"/>
            <Button       x:Name="grado6"       Content="-20°"       Canvas.Left="688"
Canvas.Top="573"           Width="100"           Height="80"           FontSize="35"
Click="RadioButton_Checked_5" Background="LightGreen"/>
            <Button       x:Name="grado7"       Content="-25°"       Canvas.Left="806"
Canvas.Top="573"           Width="100"           Height="80"           FontSize="35"
Click="RadioButton_Checked_6" Background="LightGreen"/>

        </Canvas>
    </Grid>
</Page>
```

16.11 Anexo 6.1 Código Generics.xaml

```
using Coding4Fun.Kinect.Wpf.Controls;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Kinect;

namespace WpfApplication1
{
   public static class Generics
    {


       public static int LoadingStatus = 0;

       public static void ResetHandPosition(HoverButton btnHoverButton)
       {
           btnHoverButton.HorizontalAlignment                                    =
System.Windows.HorizontalAlignment.Center;

           btnHoverButton.VerticalAlignment                                      =
System.Windows.VerticalAlignment.Bottom;

       }



    }
}
```