"Responsabilidad con pensamiento positivo"

UNIVERSIDAD TECNOLÓGICA ISRAEL

TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE:
INGENIERO EN ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES

TEMA:

ANÁLISIS Y CORRECCIÓN DE POSTURA EN EL PERSONAL ADMINISTRATIVO DE LA UNIVERSIDAD ISRAEL, BASADO EN KINECT.

AUTOR:
CHUQUITARCO GUDIÑO LUIS CARLOS

TUTOR TÉCNICO — ING. WILMER ALBARRACIN. MG

TUTOR METODOLÓGICO — ING. MAURO BOLAGAY. MG

QUITO – ECUADOR
2019

DECLARACIÓN

Yo Luis Carlos Chuquitarco Gudiño, estudiante de la Universidad Tecnológica Israel de la carrera de Electrónica y Telecomunicaciones, declaro que el contenido en este Proyecto de Titulación requisito previo a la obtención del Grado de Ingeniería en Electrónica Digital y Telecomunicaciones, son originales, y de exclusiva responsabilidad legal y académica del autor.

CHUQUITARCO GUDIÑO LUIS CARLOS
C.I.: 050294773-2

UNIVERSIDAD TECNOLÓGICA ISRAEL

CERTIFICACIÓN

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación certifico:

Que el trabajo de titulación" ANÁLISIS Y CORRECCIÓN DE POSTURA EN EL PERSONAL ADMINISTRATIVO DE LA UNIVERSIDAD ISRAEL, BASADO EN KINECT", presentado por el Sr. Luis Carlos Chuquitarco Gudiño, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se despliegue para su correspondiente estudio y calificación.

Quito D.M. septiembre 2019

TUTOR METODOLÓGICO

……..

Ing. Mauro Bolagay Egas, Mg.

UNIVERSIDAD TECNOLÓGICA ISRAEL

CERTIFICACIÓN

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación certifico:

Que el trabajo de titulación" ANÁLISIS Y CORRECCIÓN DE POSTURA EN EL PERSONAL ADMINISTRATIVO DE LA UNIVERSIDAD ISRAEL, BASADO EN KINECT", presentado por el Sr. Luis Carlos Chuquitarco Gudiño, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se despliegue para su correspondiente estudio y calificación.

Quito D.M. septiembre 2019

TUTOR TÉCNICO

……..

Ing. Wilmer Albarracin, Mg.

AGRADECIMIENTOS

Me permito agradecer a Dios por brindarme cada momento de mi vida, una oportunidad para alcanzar las metas que me he propuesto en el transcurso de mi existencia logrando cumplir los objetivos.

A mis padres por ser y por seguir siendo un apoyo incansable con sus concejos y virtudes a mis hermanos por estar presente en mis momentos felices y tristes, a mi esposa y mis hijos por ayudarme a culminar mi meta profesional.

A mis tutores, Ing. Wilmer Albarracin e Ing. Mauro Bolagay, quienes fueron la guía para que este proyecto sea una realidad

DEDICATORIA

Quiero dedicar este proyecto de grado a las personas que me acompañaron en el transcurso de mi vida, a mi amada esposa Alexandra Pruna por darme fuerza de voluntad y no decaer en el proceso de mi carrera estudiantil, a mis amados hijos Carlitos y Arlette por entenderme y comprender el poco tiempo que les brindaba para poder estar con ellos, a mis padres Carlos Chuquitarco y Margarita Gudiño que estuvieron en toda mi etapa estudiantil y siempre me brindaron un concejo, mis hermanos por siempre brindarme un incondicional ayudo a fortalecer mi espíritu, a mis queridos suegros por brindarme su apoyo su cariño y su ayuda cuando más lo necesitaba.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto describe el desarrollo de una aplicación para identificar una postura correcta del personal administrativo de la Universidad Israel con Kinect, que se desarrolla en Visual Studio 2019 con el lenguaje de programación C # WPF. El sistema se encarga de comprobar en tiempo real si el usuario se encuentra en una postura correcta o incorrecta, esto se lo realiza a través de una representación gráfica de las articulaciones, las cuales son diseñadas como vectores, el punto de referencia para los brazos va ser los codos (ELBOW), para la piernas va ser la rodilla (KNEE), para la espalda se tomara el cuello (SHOULDER) esto permitira realizar un algoritmo y poder calcular el ángulo que forman entre ellas, mediante este resultado se podrá definir si se encuentra en el rango para considerar que la postura esta correcta, la aplicación consta de un timer que hace referencia que si la posición se encuentra fuera de rango mayor a 1 min esta será considerada como mala postura, en este rango de tiempo el usuario podrá tener pausas activas, se creara un registro en Excel por hora e indicando en que minuto y que articulación se encuentra en una postura incorrecta y el ángulo medido.

Para el desarrollo del proyecto se utilizó un sensor Kinect Xbox360 con su adaptador para poder tener comunicación con el computador, el software SDK es oficial de Microsoft el cual tiene los drivers necesarios para poder reconocer la Kinect al computador. A final del desarrollo de la aplicación demostró ser una alternativa eficiente a la hora de realizar un análisis ergonómico de la postura de trabajo, para el personal que se encuentra trabajando por durante 8 horas diarias de tras de un escritorio.

Palabras claves: Kinect, programación, SDK, WPF, Excel.

ABSTRACT

This project describes the development of an application to identify a correct posture of Israel University's administrative staff with Kinect, which is developed in Visual Studio 2019 with the C# WPF programming language. The system takes care of checking in real-time if the user is in a correct or incorrect posture, this is done through a graphical representation of the joints, which are designed as vectors, the reference point for the arms it will be the elbows (ELBOW), for the legs will be the knee (KNEE), for the back will be taken the neck (SHOULDER) this will help to be able to realize an algorithm and be able to calculate the angle that they form between them, by this calculated result can be defined if it is found in the range to consider that the posture is correct, the application consists of a timer that refers that if the position is out of range greater than 1 min it will be considered as bad posture, in this time range the user may have pauses ac will create a record in Excel by hour and indicating at what minute and in which articulation you are in the wrong posture and the angle you have.

For the development of the project, an Xbox360 Kinect sensor was used with its adapter to be able to communicate with the computer, the SDK software that is official from Microsoft which has the necessary drivers to be able to recognize the Kinect to the computer. At the end of the development of the application proved to be an efficient alternative when performing an ergonomic analysis of the working posture, for staff who are working for 8 hours a day behind a desk.

Keywords: Kinect, programming, SDK, WPF, Excel.

INTRODUCCIÓN

Con estas múltiples características que cuenta la Kinect despertó la atención de los hackers, surgiendo hipótesis sobre nuevas aplicaciones que se podrían desarrollan si la Kinect podría conectarse al computador por USB, Hector Martin tuvo excito al poder hacker al dispositivo con la primera aplicación que permitía la utilización de la cámara RGB como las funciones de sensibilidad a la profundidad del dispositivo. Posteriormente del lanzamiento del código abierto junto con el middleware "intercambio de información entre aplicaciones".

Fuera de duda el dispositivo Kinect ha despertado la atención de investigadores, y educadores, en el transcurso de tiempo se creado diversas aplicaciones en la educación, en la salud y en entretenimiento, en la salud se realizó la Captura de movimiento en rehabilitación aplicada ayudar a la gente con lesiones y de este modo realizar una rehabilitación. Con esta idea se realizó el proyecto (Li et al., 2014) que elabora un entrenador virtual para el uso de fisioterapeutas y pacientes en fisioterapia con ejercicios físicos.

(Baroja Payan, Juarez Rivera, Rojas Duran, & Velasquez Calderon, 2014), desarrollaron un aplicación, el medir el grado de fatiga y los trastornos ocasionados al sistema musculo-esquelético, producidos por la carga postural a la que son expuestos los trabajadores, la cual permite visualizar a detalles la repetividad de movimientos inadecuados en el trabajo.

(Bernal Iñiguez, 2014) Desarrollaron un sistema de captura de movimiento para realizar un análisis ergonómico mediante sensores capaces es de representar el movimiento de una persona en tiempo real con la utilización de algoritmos correctos.

Otro proyecto fue la captura de movimiento en la danza. Este artículo se describe una forma visualizar y evaluar en tiempo real de los movimientos de la danza de ballet (Kyan et al., 2015).

El proyecto pretende analizar y corregir la postura de una persona que durante toda una vida laborar permanece en una postura sedentaria, con la ayuda de la Kinect el usuario será censado en todo tiempo e indicando si se encuentra en una postura correcta e incorrecta llevando un registro para que un evaluador de su criterio.

PLANTEAMIENTO JUSTIFICACION DEL PROBLEMA

En la actualidad el ser humano y su lugar de trabajo se encuentran conjuntamente en una interacción diaria, cuando esta interacción es conforme, resulta beneficioso para el ser humano alcanzando las metas propuestas en su Institución.

Cuando un trabajador está laborando en un área de oficina frente a un computador, por mucho tiempo con una postura no adecuada esto puede producir dolor de espalda, dolores de cabeza, estrés, fatiga muscular y otras patologías pudiendo llegar a disminuir su eficiencia en el trabajo.

El estudio ergonómico para el personal administrativo de una empresa tiene valores muy elevados y el tiempo de respuesta es muy demoroso, por lo cual se busca realizar dichos estudios de manera económica y su tiempo de respuesta sea más rápida tratando de evitar lesiones que perjudiquen al personal.

Con el diseño del prototipo desarollando la técnica de visión artificial, para la adquisición de datos y el procesamiento se busca disminuir lesiones debido a riesgos ergonómicos reconocido como uno de los mayores problemas que se tiene actualmente en muchas empresas a nivel administrativo por cuanto ellos pasan el mayor tiempo sentados frente a un computador, adquiriendo posturas inadecuadas y con estas enfermedades de columna, estrés, dolores de cabeza. Adisionalmente con el desarrollo de la aplicación se trata de disminuir costo de la Universidad para que puedan realizar estudios ergonómicos en su personal administrativo, evitando enfermedades antes mencionadas.

OBJETIVOS

Objetivo General

Desarrollar una aplicación que ayude al personal administrativo en la corrección de sus posturas de trabajo, mediante la adquisición y procesamiento de imágenes a través de una cámara Kinect.

Objetivos Específicos

1. Identificar los aspectos notables a tener en cuenta para realizar un estudio ergonómico basado en imágenes.

2. Desarrollar el algoritmo que permita comprobar la postura del personal administrativo de Universidad Israel.

3. Verificar el funcionamiento completo del sistema con el personal administrativo.

4. Prevenir enfermedades en el personal, mediante el acondicionamiento de una postura correcta en el entorno ergonómico

ALCANCE

En el presente proyecto se centra a un prototipo final que permita hacer un análisis ergonómico a través de la captura y procesamiento de imágenes, esta aplicación medira el angulo que forman las articulaciones para detectar si la persona se encuentra en una postura correcta o incorecta, el sistema dara un registro de datos con el angulo medido y que articulación se encuentra en una postura incorrecta permitiendo que el evaluador pueda dar soluciones a corto plazo tratando de evitar enfermedades.

DESCRIPCIÓN DE LOS CAPÍTULOS

El proyecto presentado se divide en cuatro capítulos:

En el capítulo I, se da a conocer lo que es la postura de trabajo de manera correcta e incorrecta y las consecuencias que ocasiona el tener una mala postura, así como los elementos que se utilizan para el desarrollo de la investigación.

En el capítulo II, se describe la metodología de investigación utilizada para el desarrollo de la aplicación, los métodos que se utilizó para determinar los objetivos del proyecto

En el capítulo III, se describe la propuesta que se plantea en la investigación, las herramientas necesarias tanto como hardware y software para el desarrollo de la aplicación sus diagramas de bloques, sus características, costos de la investigación, y cronograma de actividades para el desarrollo del proyecto.

En el capítulo IV, se describe la implementación del prototipo, la programación efectuada en la cámara Kinect para el censado de la postura del personal administrativo, constara las pruebas que se efectuó para dar constancia de la efectividad de la aplicación

Capítulo 1

FUNDAMENTACIÓN TEORICA

1.1 ERGONOMÍA.

La ergonomía es la ciencia que estudia el diseño del ambiente laboral, es la aplicación vinculada con la ingeniería y ciencias biológicas, para contribuir un desempeño entre hombre y el trabajo a una adaptación optima, con el propósito de mejor su rendimiento contribuyendo en el bienestar de la persona.

1.2 ERGONOMÍA EN LA OFICINA

Investiga el corregir y diseñar un ambiente profesional con la meta de acortar enfermedades asociados al tipo de actividad: movilidad limitada, posturas incorrectas, poca luminosidad y sus efectos perjudiciales sobre la salud y la comodidad de los usuarios, convirtiéndose en lesiones músculo-esqueléticas en manos, hombros, cuello, los problemas circulatorios, fatigas visuales. Resultado a esto, las industrias que construyen muebles y equipos de oficina se apuntan a mejorar, en ofrecer un diseño que refleje apropiadamente los avances y cuidados ergonómicas de progreso (Martinez Castro, 2012)

La precaución de accidentes profesionales en la oficina en mayor orden depende del adecuamiento de la oficina que se utilice, efectúe con las propiedades mínimas de eficacia ergonómica para reducir las molestias que presenta una postura frecuente en su labor como se indica en la figura 1.1.

Figura 1.1. Gráfico de postura en la oficina
Fuente: (Martinez Castro, 2012)

1.3 ANÁLISIS ERGONÓMICO

El estudio ergonómico del espacio de trabajo correcto es integral, se considera acciones, movimientos y desplazamientos, advirtiendo y planteando condiciones ergonómicamente adecuadas de acuerdo a normas y a la antropometría de cada personal que se encuentre en su puesto de trabajo, “de modo que el trabajo se adapte a la persona y no la persona al trabajo” como se indica en la figura 1.2.

Un estudio ergonómico del lugar en el cual se labora consiste en una realización cuidadosa de la ocupación para lo cual que se utilizan entrevistas, observaciones del evaluado al empleado con el fin de obtener la información necesaria para determinar una postura correcta.

Figura 1.2. Aspectos importantes ergonómicos
Fuente: (Escobar, 2014)

1.4 MÉTODOS DE EVALUACIÓN ERGONÓMICOS

Tanto las empresas, universidades, municipios los principales inconvenientes encontrados en las personas que trabajan al frente de un computador sufren de sequedad y ardor ocular, síndrome del túnel carpiano, lumbalgia, dolor de hombros, cansancio pesadez e hinchazón de las piernas, con tendencias repetitivas; la necesidad de establecer instrumentos que permita evaluar el ambiente de trabajo que puedan adquirir estas enfermedades para su debida corrección tratando de evitar o prevenir los efectos que se generen. A continuación, se presentan algunos métodos ergonómicos:

1.4.1 Apli Rula (Rapid Upper Limb Assessment)

Es fomentado para conceder una valoración rápida de los esfuerzos por lo que es expuesto los músculos esqueléticos del personal debido a una postura, función muscular y las fuerzas que ellos ejercen.

Permite la demostración de los trabajadores a factores de riesgo que pueden producir trastornos en las extremidades superiores de cuerpo: posturas, repetividad de movimientos, fuerzas aplicadas y actividad estática del sistema musculo esquelético. (Escobar, 2014)

1.4.2 JSI (Job Strain Index).

Ayuda a evaluar si el personal que los ocupan está inseguros a desarrollar trastornos traumáticos en la parte de las extremidades superiores debido a movimientos monótonos. Se evalúa el codo, la mano, la muñeca, el antebrazo. La técnica se basa en la medición de seis variables que una vez calculadas, se obtiene operaciones multiplicadores de una ecuación. El último número obtenido indica el factor de riesgo en la aparición de enfermedades en las extremidades superiores. “Las variables a medir por el evaluador son: la intensidad del esfuerzo, la duración del esfuerzo por ciclo de trabajo, el número de esfuerzos realizados en un minuto de trabajo, la desviación de la muñeca respecto a la posición neutra, la velocidad con la que se realiza la tarea y la duración de la misma por jornada de trabajo”. (Mas & Jose, 2015)

1.4.3 Owas

Permite una valoración de carga física de las posturas durante su labor de trabajo, se determina por la capacidad de valorar en conjunto todas las posturas durante su desempeño de trabajo con la comparación de posturas establecidas de espalda, brazos y piernas. Este método aunque es antiguo es el más empleado en la evaluación postural. (Mas & Jose, 2015)

1.4.4 Rosa (Rapid Office Strain Assessment)

El objetivo es evaluar el nivel de riesgos comúnmente asociados a los puestos de trabajo en oficinas. La técnica es ajustable al lugar de trabajo en los que el trabajador permanece sentado en una silla, frente a una mesa y manejando un equipo informático con pantalla de visualización de datos. Se consideran en la evaluación los elementos más comunes de estas estaciones de trabajo (silla, superficie de trabajo, pantalla, teclado, mouse y otros periféricos). Como resultado de su aplicación se obtiene una valoración del riesgo medido y una estimación de la necesidad de actuar sobre el puesto para disminuir el nivel de riesgo.

Para desarrollar el método ROSA los autores describieron las características de un puesto de trabajo en oficina de diseño óptimo, así como las posturas ideales (o neutrales) que debería adoptar el trabajador para minimizar el riesgo ergonómico. Estas características ideales se obtuvieron analizando las recomendaciones de la guía CSA Z412 canadiense, basada en la norma ISO 9241 (Ergonomic requirement for office work with visual display terminales). Para comprobar el riesgo de un lugar de trabajo el método ROSA analiza el grado de desviación existente entre el puesto evaluado y dichas características ideales. (Mas & Jose, 2015)

1.5 KINECT V1

Es un mecanismo, inicialmente considerado como un simple controlador de juego, que debido a sus elementos que lo forman: sensor de profundidad, cámara RGB, array de micrófonos y sensor de infrarrojos, es capaz de capturar el esqueleto humano reconociéndolo y colocándolo en el plano. (Murillo, Kinect for developers, s.f.) figura 1.3.

Figura 1.3. Kinect
Fuente: («Kinect», 2019)

1.5.1 Hardware Kinect

El dispositivo Kinect dispone de cuatro partes elementales que se explica a continuación figura 1.4.

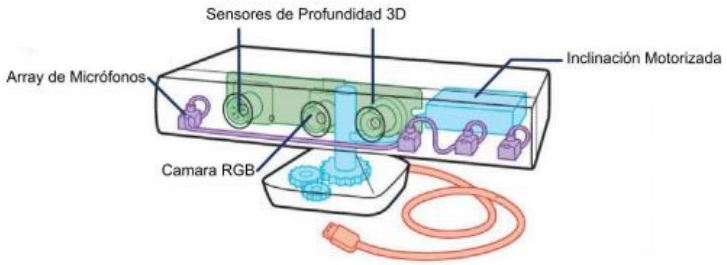


Figura 1.4. Hardware Kinect
Fuente: (Windows, 2013)

1.5.1.1 Sensor de profundidad.

Trabaja con dos elementos que se encuentra comprendidos en la Kinect; un reflector de luz y un sensor de tipo de luz. El reflector proyecta una matriz de rayos de luz infrarroja sobre el espacio, estos rebotan en los objetos y son recibidos por el sensor. La resolución es de 640x480 pixeles, cada codificación de pixeles se ocupan 11 bits, teniendo un resultado de 2048 niveles de profundidad. También posee de un campo de visión angular de 60° en posición horizontal y en la posición vertical es de 43°, su capacidad de detectar a una distancia del sensor 0,7m hasta los 6m. (Martino & Lema, 2011) Como se indica en figura.1.5.

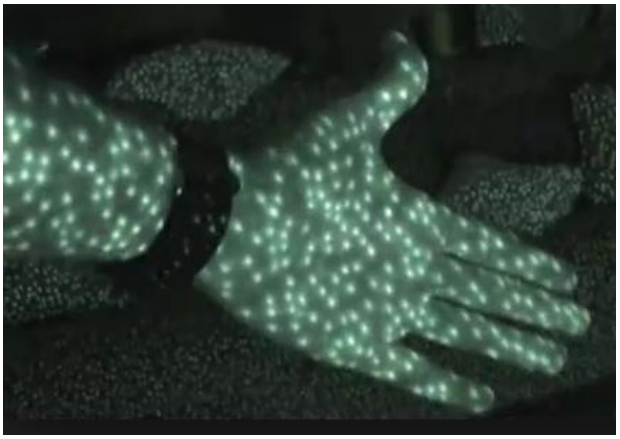


Figura 1.5. Sensor de profundidad
Fuente: (Martino & Lema, 2011)

1.5.1.2 Cámara RGB

Es una cámara de video que nos proporciona el reconocimiento facial y otras funciones para la detección, detectando tres componentes de color: rojo, verde y azul, logra captura de 30 cuadros por segundo con una resolución de 640x480 pixeles de color que se encuentran codificados con 8 bits como se indica en la figura 1.6.

Figura 1.6. Cámara RGB
Fuente: (Martino & Lema, 2011)

1.5.1.3 Micrófono Multi-Array

Se encuentra incorporado por cuatro micrófonos, son utilizados a para la localización de sonidos. Este arreglo de micrófonos se ha utilizado más para teleconferencias y consolas

de juegos debido a una fácil implementación y por su tamaño como se indica en la figura 1.7.

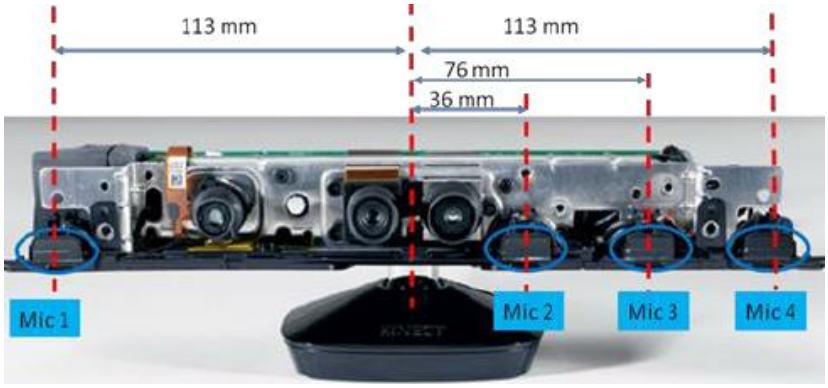


Figura 1.7. Micrófono multy-array
Fuente: (Pei et al., 2013)

1.5.1.4 Inclinación motorizada

Un motor mecánico en la base del sensor inclina de forma automática para arriba o abajo según sean necesario sus aplicaciones como se indica en la figura 1.8.

Figura 1.8. Motor mecánico
Fuente: (Ruiz, 2012)

1.5.2 SDK Kinect

El SDK de Kinect entrega muchas posibilidades a los usuarios, programadores, estudiantes con el fin para poder desarrollar y crear aplicaciones innovadoras en todas las áreas de desarrollo.

SDK proporciona los drivers para la cámara Kinect que son capaces de controlar, con una amplia gama de librerías con algoritmos.

SDK incluye:

- **Controladores Kinect:** para utilizar en una computadora con Windows 7, Windows 10 el cual controla:
    - ✓ Conjunto de micrófonos multi-array
    - ✓ Cámara de video y de profundidad
- **Skeletal Tracking:** utilizado para representar el cuerpo humano, el número de articulaciones que se representan partes del cuerpo como es la cabeza, hombros, cuello y brazos

1.5.3 Requerimientos de Hardware

Para comenzar a utilizar el SDK de Kinect se necesita las siguientes mínimas características según Microsoft.

- Computador con procesador dual-dual core, 2.66GHZ o superior
- Windows 7, en adelante que sea compatible con tarjetas graficas
- 4Gb de RAM
- Adaptador USB (este será utilizado si la Kinect es para XBOX 360)

Para el desarrollo de diferentes aplicaciones se puede utilizar algunos software que son compatibles con la Kinect o tienen desarrollada las librerías como son, visual studio 2010 en adelante, Processing versión 2.2.1, Scratch, Matlab, etc.

1.6 KINECT V2

Nace como mejora de la primera versión de Kinect de Microsoft y en conjunto con la nueva consola de videojuegos de Xbox One en el año 2013, incorpora una mejora en ellos sensores que permiten la interacción con la persona, y al igual que con la versión anterior permite su desarrollo a través de código abierto disponible en varias plataformas y con distintas herramientas.

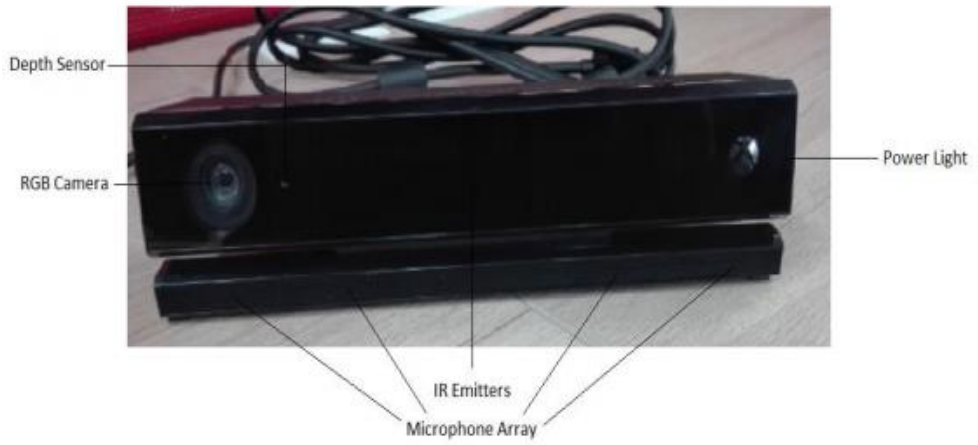


Figura 1.9. Componentes del dispositivo Kinect V2
Fuente: (Larumbe Bergera & Villanueva Larre, 2015)

En la tabla 1.1 se presentan las características técnicas entre el dispositivo Kinect V1 y Kinect V2.

Tabla 1.1:

Caracteristicas de los dispositivos Kinects

Capacidad	Kinect V1	Kinect V2
Profundidad	320×240, 640×480	512×424
	Distancias 0.8 a 4 metros en modo default.	Distancias 0.5 a 4.5 metros
Rastreo del cuerpo	Capacidad para detectar 6 personas, pero solo dos se pueden rastrear completamente	Capacidad para detectar 6 cuerpos, todos completamente rastreados.
	Por cuerpo rastreado, capaz de identificar 20 articulaciones	Por cuerpo rastreado, capaz de identificar 25 articulaciones
Video	640×480 @30 fps	1920×1080 @30 fps
	1280×960 @12 fps	High Definition
	Se denomina ColorStream	Se denomina ColorSource
Motor de Inclinación	Si.	No.
	Motor puede graduarse entre +27 grados y –27 grados.	Tiene mayor ángulo de visión. Sin embargo, en caso de que se requiera, se puede graduar manualmente su inclinación.
Puerto	USB 2.0	USB 3.0

Fuente: Elaborado por el autor

Capítulo 2

MARCO METODOLÓGICO

2.1 METODOLOGÍA DE INVESTIGACIÓN

En este capítulo se plantea la metodología, técnica y fases utilizadas para el desarrollo de una aplicación ergonómica para detectar una postura correcta e incorrecta del personal administrativo. El investigador no debe estar ligado a utilizar un único método fijo para el desarrollo de la investigación. Por tal motivo se presentan las metodologías que más se adaptó para el desarrollo del proyecto.

2.2 INVESTIGACIÓN EXPLORATORIA

En esta primera etapa se necesita conocer que trabajos se han realizado sobre la utilización de la cámara Kinect en el Ecuador en diferentes universidades específicamente en el desarrollo del bienestar para el cuidado de la salud. Para ello se definieron las siguientes fases para la generación de información relevante:

- Búsqueda de la información
- Análisis de la información

2.2.1 Busqueda de la información

En la búsqueda de la informacion, se utilizó la investigación bibliográfica de documentos, en el que tenían como referencia la cámara Kinect, recopilando información en pappers, libros, tesis publicadas. El resultado de la búsqueda de información se detalla en la tabla 2.1

Tabla 2.1:

Tesis de análisis para el proyecto Kinect

UNIVERSIDAD	TEMAS ANALIZADOS
Universidad Politécnica Salesiana	Diseño, desarrollo e implementación de un sistema que permita controlar un brazo robótico Mitsubishi RV-2AJ a través del periférico Kinect de Microsoft como interfaz de usuario
	Diseño e implementación de un sistema para el análisis del movimiento humano usando sensores Kinect
Universidad de las Fuerzas Armadas Espe	Diseño e implementación de un prototipo para fisioterapia con KINECT
	Diseño y construcción de una plataforma interactiva para fisioterapia continua pasiva para lesiones del hombro con el uso del dispositivo Kinect
Escuela Politécnica Nacional	Diseño e implementación de un sistema de navegación para asistencia de personas no videntes
	Control del movimiento de un Quadrotor mediante un sensor de profundidad Kinect

Fuente: Elaborado por el autor

2.2.2 Análisis de la información

Mediante la información obtenida se analiza los diferentes temas que realizaron aplicaciones con la cámara Kinect, en este punto se aplicó el método analítico que extrae las partes relevantes y optimizar el proyecto.

2.3 DESARROLLO DEL SOFTWARE

Para el desarrollo de la aplicación es necesario cumplir con las siguientes etapas:

- Diseño y codificación del algoritmo
- Validación del algoritmo
- Pruebas de funcionamiento

2.3.1 Diseño y codificación del algoritmo

Mediante los conocimientos adquiridos se procedió a realizar un diagrama de flujo, representando el proceso de planificación del desarrollo de la aplicación. En el capítulo 3 será respaldado dicho diagrama.

2.3.2 Validación del algoritmo

Para validar el algoritmo desarrollado fue necesario ejecutar la aplicación, verificando el funcionamiento y adquisición de datos la medición que forman las articulaciones del usuario a ser censado verificando si se encuentra en una postura correcta e incorrecta. En esta etapa se puede realizar ajustes necesarios para cumplir con el objetivo deseado.

2.3.3 Pruebas de Funcionamiento

En esta etapa, se concluirá con las pruebas de verificación del funcionamiento de la aplicación, determinando si la postura del personal administrativo de la Universidad Israel esto será argumentada en el capítulo 4.

Capítulo 3

PROPUESTA

3.1 DESCRIPCIÓN GENERAL DEL PROYECTO.

En el desarrollo del proyecto se realizará una aplicación encargada de ejecutar el seguimiento del personal administrativo en su puesto de trabajo, el seguimiento consiste en comprobar en todo momento si el personal se encuentra en una postura correcta o incorrecta, de acuerdo con las especificaciones que los libros proporcionan con la medición de los ángulos que forman las diferentes articulaciones.

El proyecto se encarga específicamente de identificar a un usuario en su puesto de trabajo, el cual deberá colocarse a una medida específica que permita el seguimiento adecuado a la cámara Kinect Xbox 360, posee distintas propiedades, tiene una cámara convencional RGB y un sensor de distancia, una serie de micrófonos, el sistema de percepción de profundidad consta de tres partes básicas, el proyector laser de infrarrojos, sensor CMOS y un microchip que procesa la información, su funcionamiento se basa en la proyección de un patrón de puntos pseudo-aleatorio y su lectura y triangulación mediante el sensor CMOS (ver anexo 4 datasheet Kinect).

Una vez identificado y analizado al usuario la observación será constante, se fijará en ciertos puntos específicos del cuerpo, si estos puntos clave no correspondieran a las especificaciones que se colocara en el código dará como resultado una postura incorrecta.

Al momento que la aplicación dé una alarma de posición incorrecto, el usuario deberá modificar la articulación que se encuentre fuera del rango de medición, una vez corregida la postura el mensaje de aviso incorrecto cambiara su estado a correcto, de esta forma se ayudara al personal administrativo de la Universidad Israel a tener una adecuada postura de trabajo y mejorar su calidad de vida.

3.2 DISEÑO INTEGRAL DE LA PROPUESTA

Lo integran los siguientes elementos electrónicos en la figura 3.1, cámara Kinect, portátil y la persona a realizar el análisis.

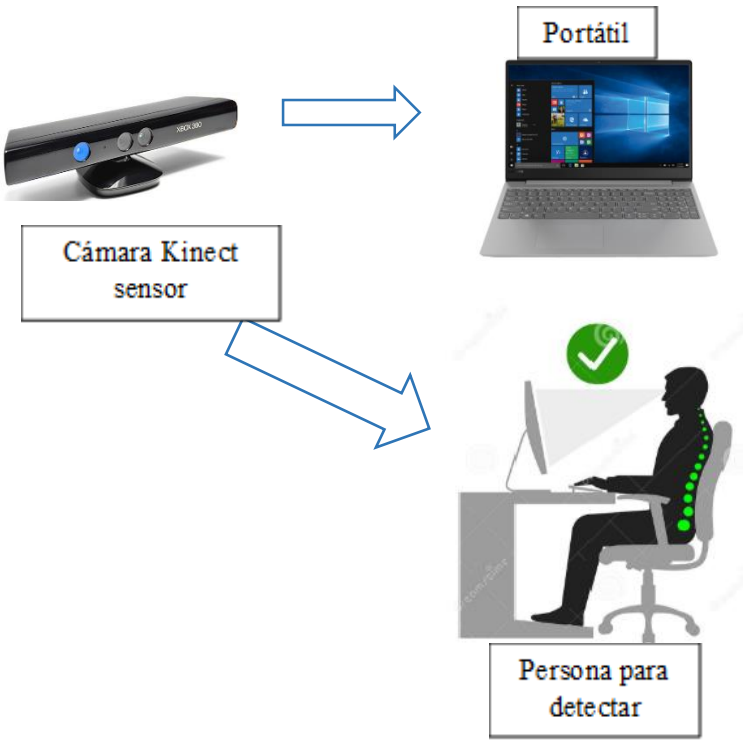


Figura 3.1. Elementos que conforman la propuesta
Fuente: Elaborado por el autor

Con la ayuda del SDK, el cual permite utilizar el sensor de profundidad y la cámara RGB con una resolución de 640x480, donde se le asigna una información de color y profundidad a cada pixel. La Kinect, captura un flujo de datos con la resolución a una velocidad de 30 cuadros por segundos.

Para un funcionamiento correcto debe cumplir con algunas condiciones teóricas:

- Los objetos tienen que estar dentro de un ángulo de visión $\pm 43°$ verticalmente y de $\pm 57°$ horizontalmente.
- Rango de profundidad del sensor 1,2 – 3,5 m
- Rango de inclinación física de $\pm 27°$

3.3 DIAGRAMA DE FLUJO

Para realizar la aplicación se debe seguir los procesos determinados, por lo que se empezará a plantearse una estructura. Ya con un diagrama de flujo presentado en la figura 3.2, 3.3, se tendrá más claro a donde se desea llegar, con la información extraída de la cámara se plantea como calcular los ángulos de medición de las articulaciones que serán objetos de medición.

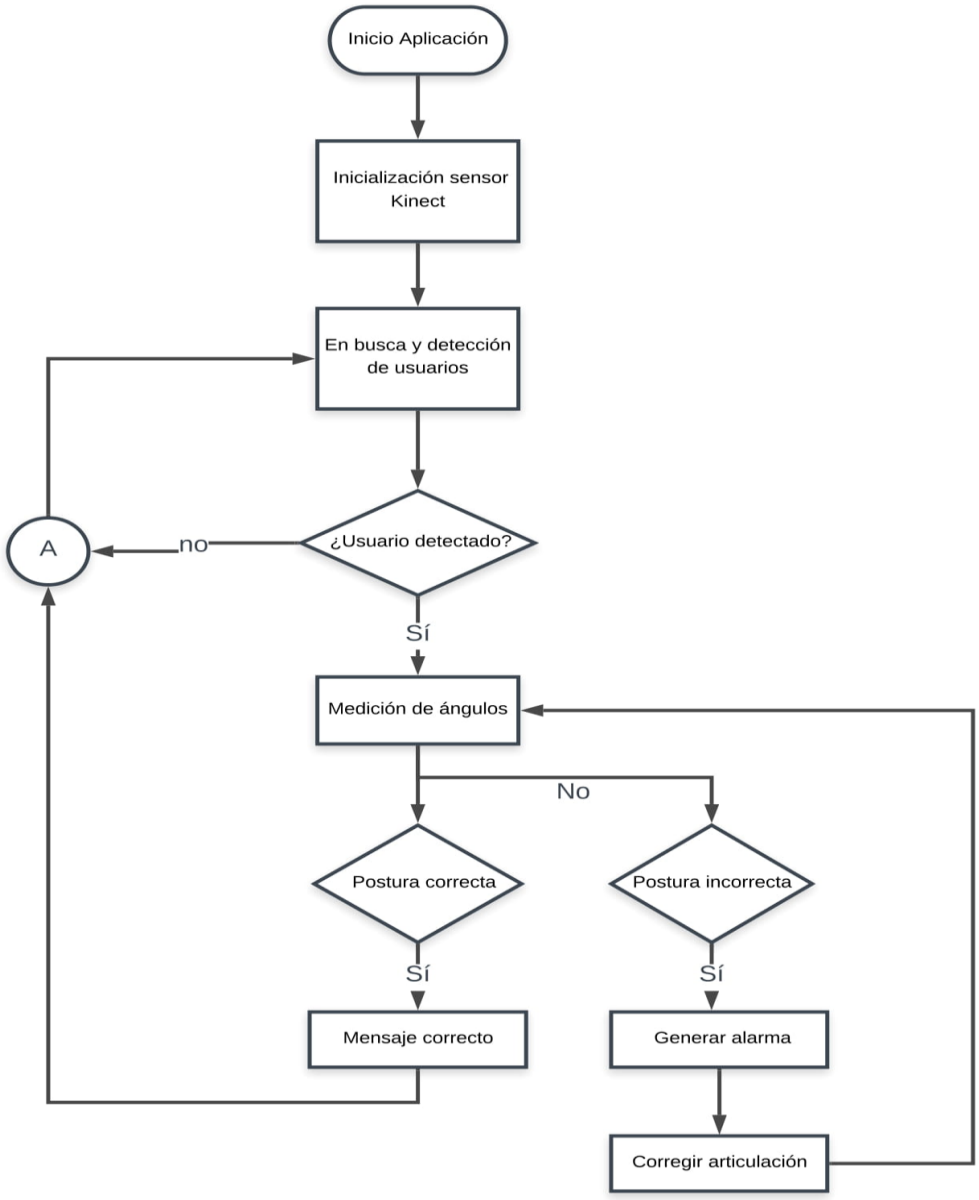


Figura 3.2. Diagrama de Flujo
Fuente: Elaborado por el autor

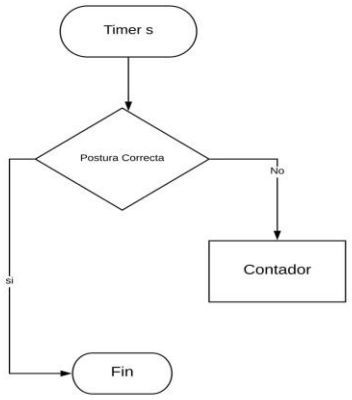


Figura 3.3. Diagrama de Flujo Timer seg
Fuente: Elaborado por el autor

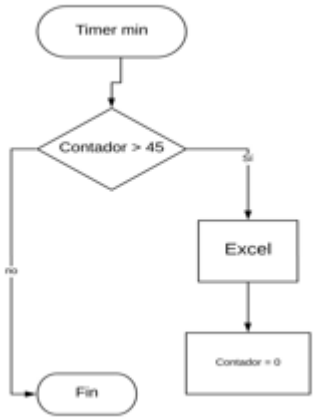


Figura 3.4. Diagrama de Flujo Timer min
Fuente: Elaborado por el autor

3.4 HERRAMIENTAS PARA EL DESARROLLO DE LA APLICACIÓN PROPUESTA

3.4.1 Kinect

Kinect es un dispositivo integrado por un sensor de profundidad, cámara RGB, array de micrófonos, sensor (emisor y receptor), capaza de reconocer y capturar el esqueleto humano posicionándolo en un plano 3D, toda la información que captura el dispositivo ayuda a la ejecución de la aplicación.

La SDK de Kinect puede detectar 20 puntos de las articulaciones del cuerpo humano presentado en la figura 3.5

Figura 3.5. Diagrama de Flujo
Fuente: Elaborado por el autor

Para el desarrollo de la aplicación se va a ocupar 15 puntos, eliminando ciertos puntos para optimizar el proceso de visualizacio estos puntos a eliminar van en el siguiente listado:

- Hand_Right – (mano derecha)
- Hand_Left – (mano izquierda)
- Hip_Center – (centro de la cadera)
- Foot_Right – (pie derecho)
- Foot_Left – (pie izquierdo)

3.4.2 SDK Kinect Microsoft V1.8

Es un instrumento que permite usar los datos obtenidos por la Kinect y utilizarlos en el lenguaje de programación para la creación de aplicaciones.

Es una herramienta que se encuentra disponible para su descarga gratuitamente.

3.4.3 Adaptador para Kinect Xbox 360

Este adaptador servirá para poder conectar la Kinect a la fuente de alimentación y a la computadora mediante USB como se muestra en la figura 3.6, cumplirá con las siguientes características figura 3.7:

- Voltaje de entrada de 110- 240V
- Voltaje de salida 12V
- Corriente de salida de 1.08 A

Figura 3.6. Adaptador de Kinect
Fuente: Elaborado por el autor

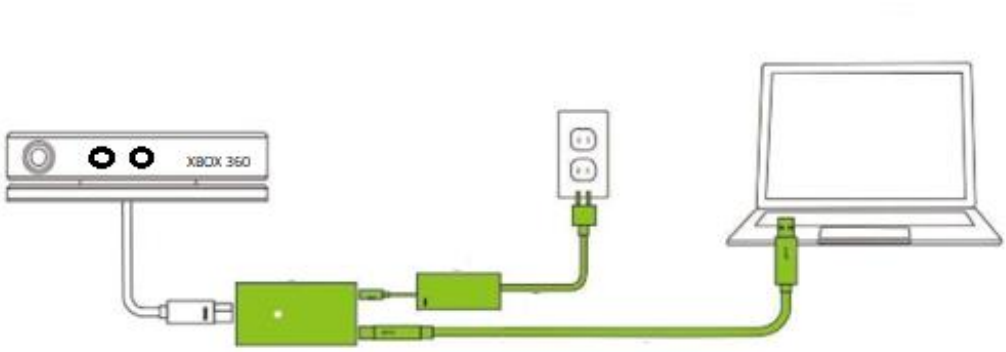


Figura 3.7. Conexión de USB
Fuente: Elaborado por el autor

3.4.4 Microsoft Visual Studio

El algoritmo a desarrollar se realiza en Visual Studio siendo este un programa que permite tener un entorno de programación amigable que utiliza un medio gráfico, soporta varios tipos de lenguaje de programación como son, C#, C++, Visual Basic.

Visual es compatible con todas las versiones de la Kinect en este caso para la aplicación se utilizará la Kinect Xbox 360 V1, la misma que va a desarrollarse es en WPF que permite el desarrollo basado en vectores, es un componente que su diseño se los puede realizar en 2D y 3D, animaciones, multimedia; esto nos permite compilar aplicaciones que incorporen estos tipos de elementos.

Visual contiene las librerías necesarias para poder desarrollar la aplicación como se lo indica en la siguiente figura 3.8.

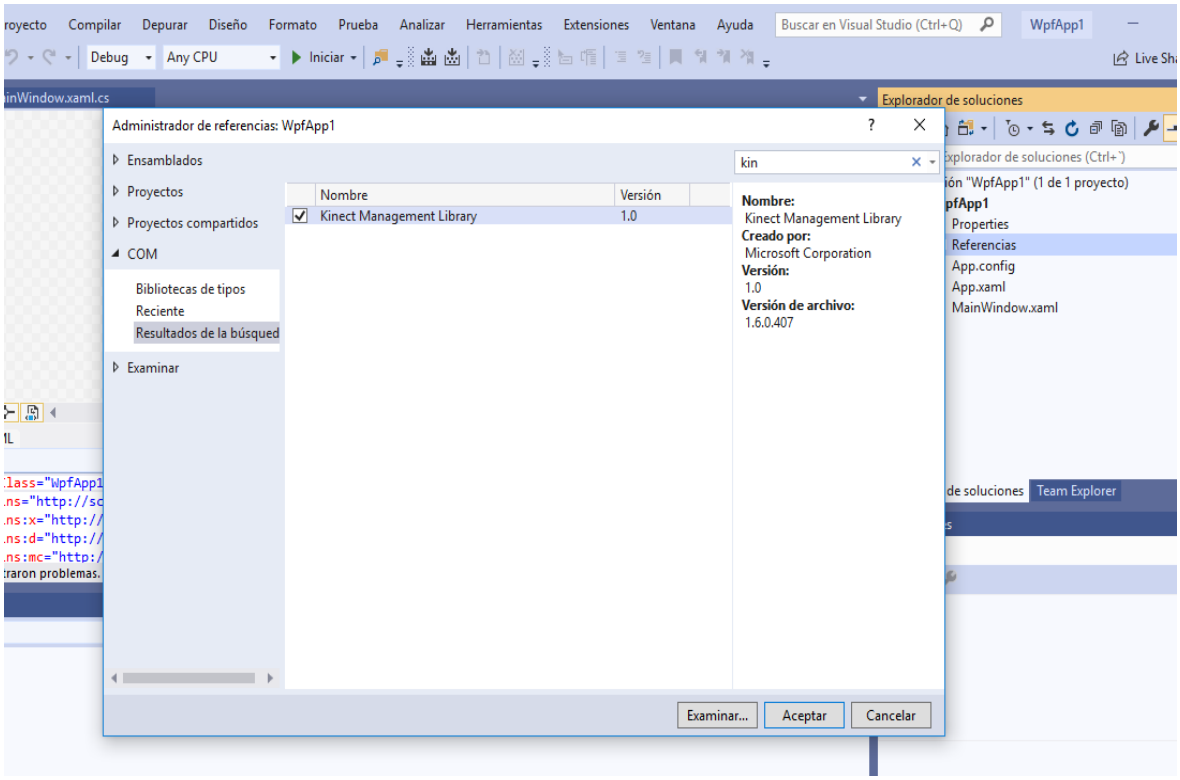


Figura 3.8. Librería de Microsoft Kinect
Fuente: Elaborado por el autor

3.5 ANÁLISIS DE COSTO Y TIEMPO

3.5.1 Costo

El costo para la realización de la aplicación no solo influye en la compra del equipo, también se adquirió un curso de programación de Visual Studio para reforzar los conocimientos adquiridos en la Universidad, se detalla en el cuadro de costos del equipo y materiales utilizados tabla 3.1.

Tabla 3.1:
Costo de equipos y materiales adquiridos para la aplicación

Equipos	Cantidad	Valor Unitario	Total
Cámara Kinect	1	$100	$100
Adaptador para Kinect Xbox 360	2	$25	$50
Pedestal para sujetar Kinect	1	$80	$80
cable alimentación	4m	$0,50	$2
Hombre/horas	20h	$250	$250
Movilización			
Transporte	1	$60	$60
Impresiones, Anillados			
Impresiones	200	$0,25	$50
Anillado/empastado tesis	1	$25	$25
		total	$617

Fuente: Elaborado por el autor

3.5.2 Tiempo

En el análisis de tiempo se realiza una descripción del cronograma (Anexo3) de actividades, realizando durante etapas que se distribuye el desarrollo del proyecto, en este proceso se realiza el algoritmo matemático, la programación fue desarrollada en 20 horas, estas fueron dividas 2 horas por día en su totalidad duraron 10 días para la realización, durante estos días se fue verificando el funcionamiento y comprobación al obtener la medición que forman las articulaciones e indicando si esta el usuario en una postura correcta o incorrecta.

3.6 VENTAJAS DE LA APLICACIÓN PROPUESTA

Para un estudio ergonómico en una empresa o institución, se debe contar con profesionales en salud ocupacional o instrumentos de medición. Estos costos son muy elevados para poder hacer un análisis a cada trabajador que se encuentran laborando detrás de un escritorio, posee un valor estimado de $400 a $600, por lo tanto no se puede realizar estos procedimientos, provocando que el personal continúe con problemas de salud por no tener una postura correcta. La universidad de Israel cuenta con un profesional para realizar evaluaciones al personal administrativo realizando un check list, medición de articulaciones y el ángulo que forman entre ellas durante las ocho horas de su labor diario, mediante la realización de este prototipo una de las ventajas que se tendrá es la reducción de costos, la aplicación podrá evaluar a todo el personal que tenga molestias musculares, estrés o enfermedades que provoquen tener una postura incorrecta, otra ventaja es que la Kinect remplazara el tiempo y el trabajo que hace el evaluador midiendo y verificando si el usuario posee la postura correcta en su área de trabajo esta a su vez dará un registro diario creando pestañas por hora de la evaluación que realiza la Kinect a la persona a ser censada.

Capítulo 4

IMPLEMENTACIÓN

4.1 DESARROLLO

En este capítulo se explicará el desarrollo del prototipo de la aplicación para realizar un análisis y corrección de postura, la secuencia de la implementación será la siguiente:

- Software
- Hadware

4.2 SOFTWARE DE PROGRAMACIÒN

Para el desarrollo del algoritmo se utilizó el software Visual Studio como se lo explico en el Capítulo 3, primer paso la creación de un nuevo proyecto como se indica en la siguiente figura 4.1.

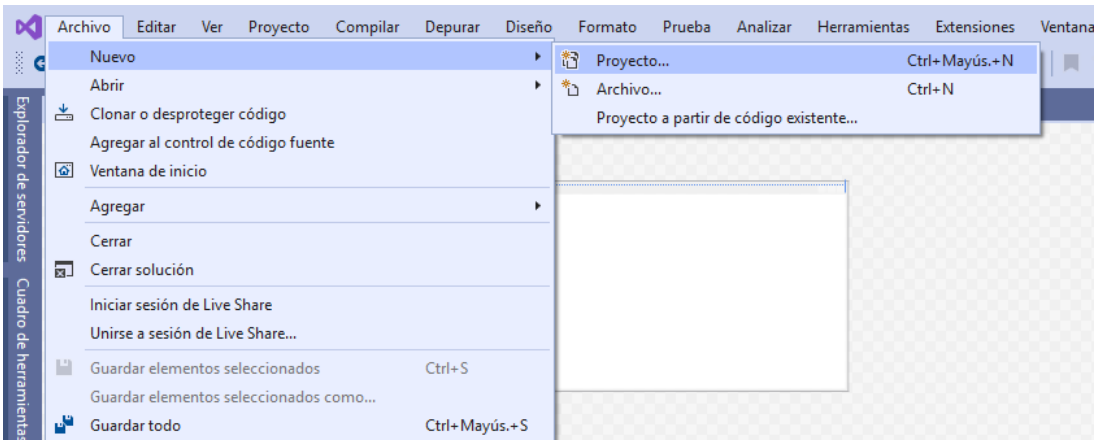


Figura 4.1. Creación de nuevo proyecto Visual.
Fuente: Elaborado por el autor

Posteriormente se creara el proyecto WPF App, se escribira un nombre al proyecto en nuestro caso se digitara SkeletonBasics representada en la figura 4.2.

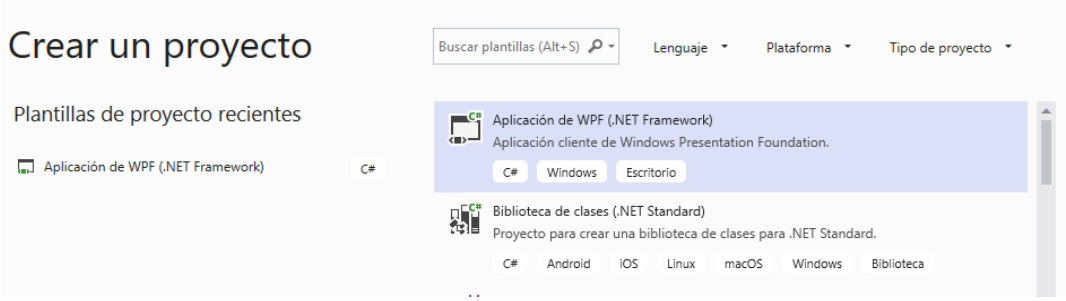


Figura 4.2. Nuevo proyecto WPF.
Fuente: Elaborado por el autor

Una vez creado el proyecto se debe agregar la referencia de Kinect, ubicarse en Solution Explore, en esta ventana seleccionar reference y con clic derecho seleccionar Add reference, en búsqueda se digitara Kinect como resultado se tendrá la referencia de la Kinect en la figura 4.3.

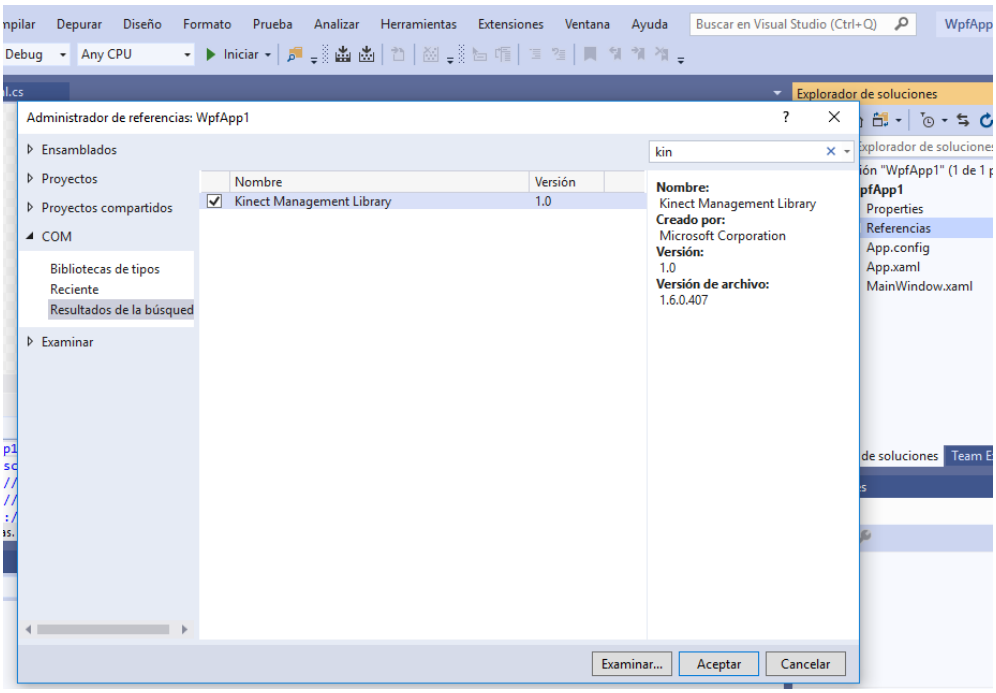


Figura 4.3. Seleccionar referencia Kinect.
Fuente: Elaborado por el autor

Colocada la referencia se dirige a la creación de una página de inicio en la cual se pedirá los datos del usuario como se muestra en la figura 4.4.

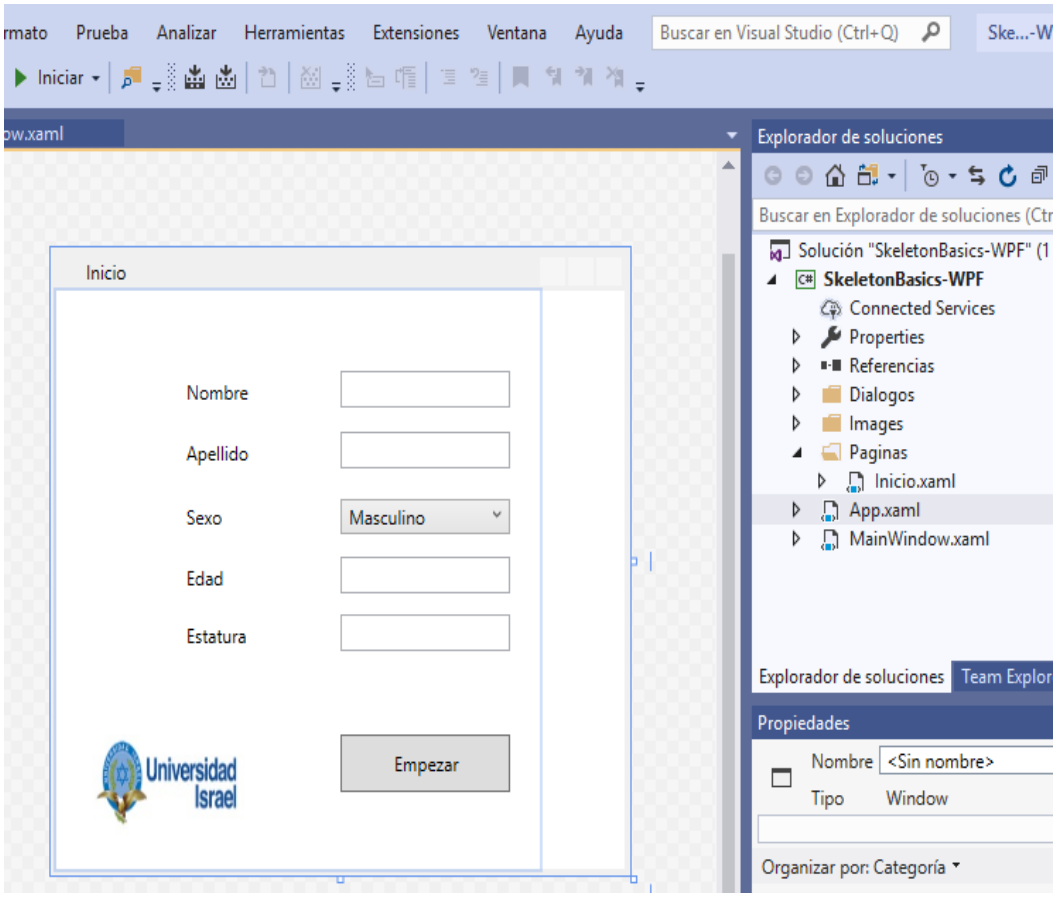


Figura 4.4. Creación de página de Inicio.
Fuente: Elaborado por el autor

En la ventana MainWindow.xaml, se va agregar diferentes diseños de logo de Imagen, block de texto, etiqueta, según sea nuestro diseño para la aplicación, en la figura 4.5 se podrá visualizar los diferentes diseños que se ocupó en la aplicación.

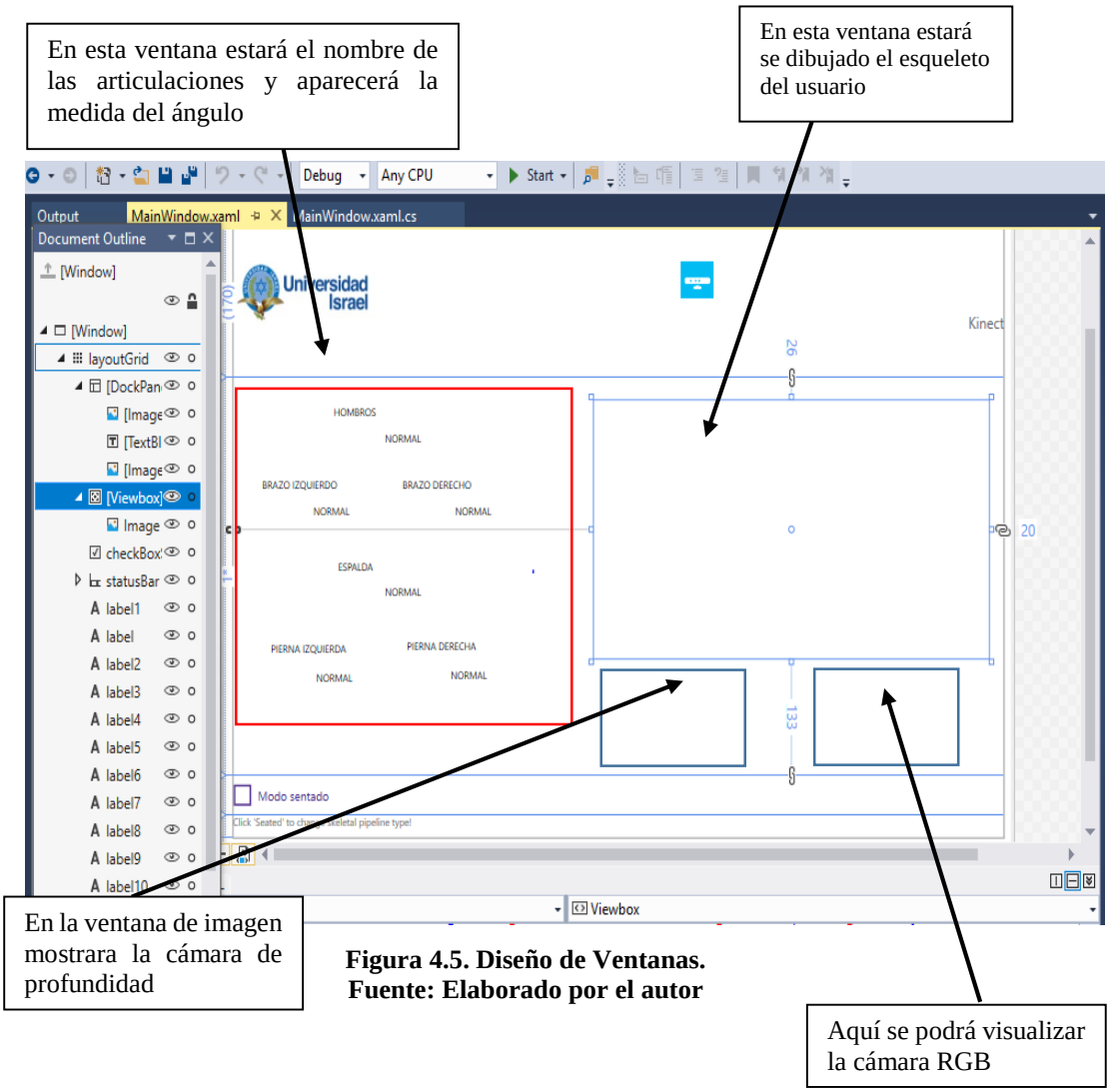


Figura 4.5. Diseño de Ventanas.
Fuente: Elaborado por el autor

Realizado el diseño de las diferentes ventanas de la aplicación, dirigirse a MainWindow.xaml.cs, aquí se comenzará con la programación, como esta es muy extensa se pondrá el lenguaje como anexo, aquí se explicará los eventos esenciales para el proyecto.

4.3 LIBRERIAS DE KINECT UTILIZADAS PARA LA APLICACIÓN

En esta parte se ira exponiendo las diferentes librerías de programación que se utilizara para el desarrollo de la aplicación.

4.3.1 Inicialización de la Kinect

Para comenzar la programación en el código fuente se debe agregar el DEIVIS que contiene todas las clases, métodos, propiedades de la Kinect figura 4.6.

```
using Microsoft.Kinect;
```

Figura 4.6. DEIVIS Kinect.
Fuente: Elaborado por el autor

4.3.2 Detección de esqueleto

Para el siguiente método se va trabajar con una interfaz natural de usuario, para que se pueda interactuar con la aplicación a través de movimientos del cuerpo como lo es cabeza, pies, manos. Para lo cual se va ocupar el flujo de datos de SkeletonStream, el cual envía flujo de datos de Frame constantes, el mismo se posesionará en tres dimensiones (x, y, z) una figura idéntica a un esqueleto humano como se muestra en la figura 4.7.

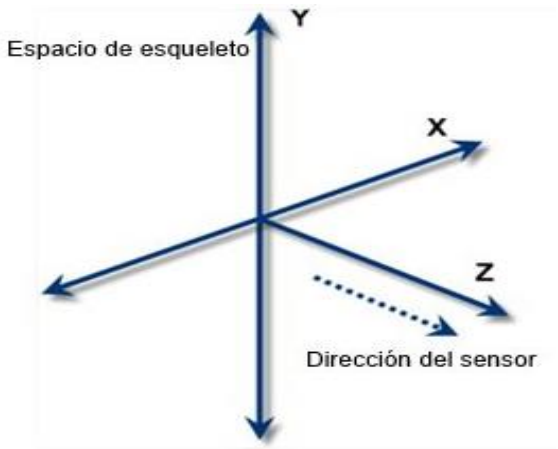


Figura 4.7. Plano dimensional Kinect.
Fuente: Elaborado por el autor

El siguiente paso es unirlas con una línea y tome la forma de un hueso entre las articulaciones, recordar que al iniciar el SkeletonStream este, está enviando posiciones de las articulaciones que el detecte cada segundo en diferentes coordenadas entonces dicho elemento debería permitir crear la línea con las coordenadas de las articulaciones y cuando

se reciba los datos borrar la línea y volverla a crear con nuevas coordenadas, para esto se va a ocupar el método JoinType (Este contiene todos los tipos de articulaciones posibles).

Para diseñar el torso se ocupara los siguientes puntos figura 4.8, y el código que se ocupa en Visual Studio figura 4.9:

1. HEAD (cabeza)
2. SHOULDER CENTER (centro del hombro)
3. SHOULDER LEFT (hombro izquierdo)
4. SHOULDER RIGHT (hombro derecho)
5. SPINE (spina)
6. HIP CENTER (centro de la cadera)
7. HIP RIHGT (cadera derecha)
8. HIP LEFT (cadera izquierda)

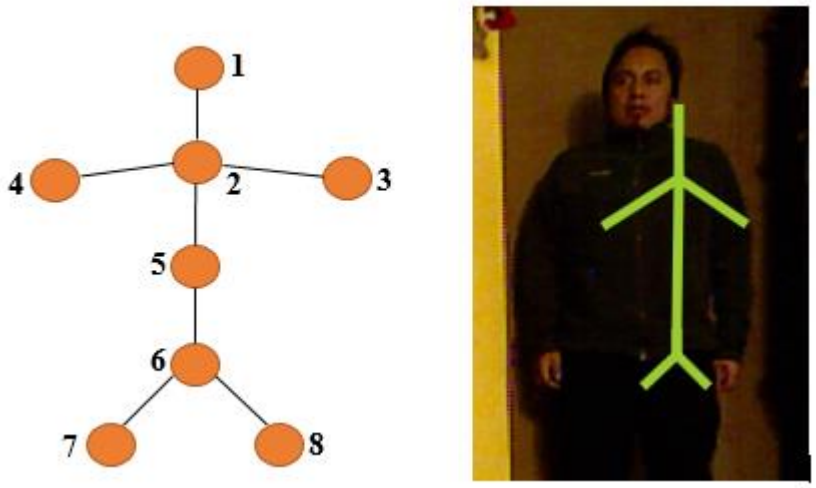


Figura 4.8. Diseño de torso
Fuente: Elaborado por el autor

```
// Render Torso
this.DrawBone(skeleton, drawingContext, JointType.Head, JointType.ShoulderCenter);
this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter, JointType.ShoulderLeft);
this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter, JointType.ShoulderRight);
this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter, JointType.Spine);
this.DrawBone(skeleton, drawingContext, JointType.Spine, JointType.HipCenter);
this.DrawBone(skeleton, drawingContext, JointType.HipCenter, JointType.HipLeft);
this.DrawBone(skeleton, drawingContext, JointType.HipCenter, JointType.HipRight);
```

Figura 4.9. Diseño de Torso en Visual Studio
Fuente: Elaborado por el autor

Los siguientes puntos serán utilizados para el diseño del brazo derecho figura 4.10, y el código que se ocupa en Visual Studio figura 4.11:

1. SHOULDER RIGHT (hombro derecho)
2. ELBOW_ RIGHT (codo derecho)
3. WRIST_ RIGHT (muñeca derecha)
4. HAND_RIGHT (mano derecha)

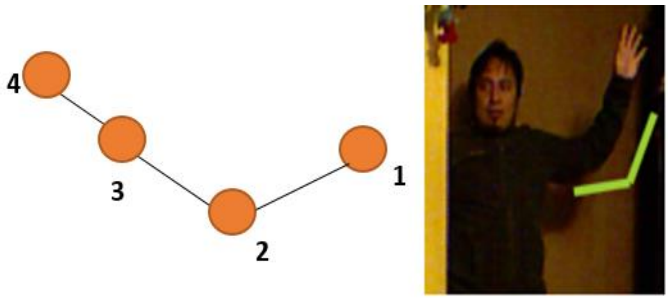


Figura 4.10. Diseño de Brazo Derecho
Fuente: Elaborado por el autor

```
// Right Arm
this.DrawBone(skeleton, drawingContext, JointType.ShoulderRight, JointType.ElbowRight);
this.DrawBone(skeleton, drawingContext, JointType.ElbowRight, JointType.WristRight);
this.DrawBone(skeleton, drawingContext, JointType.WristRight, JointType.HandRight);
```

Figura 4.11. Diseño de Brazo Derecho en Visual Studio
Fuente: Elaborado por el autor

En la realización del diseño del brazo izquierdo se ocupa los siguientes puntos figura 4.12, y el código a ocupar en Visual Studio figura 4.13:

1. SHOULDER LEFT (hombro izquierdo)
2. ELBOW_LEFT (codo izquierdo)
3. WRIST_LEFT (muñeca izquierda)
4. HAND_LEFT (mano izquierda)

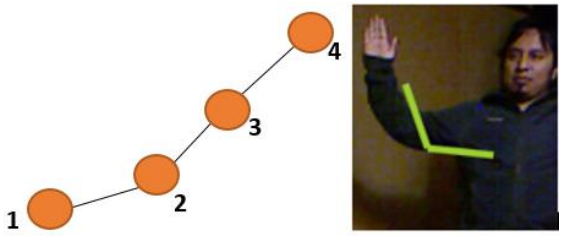


Figura 4.12. Diseño de Brazo Izquierdo
Fuente: Elaborado por el autor

```
// Left Arm
this.DrawBone(skeleton, drawingContext, JointType.ShoulderLeft, JointType.ElbowLeft);
this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft, JointType.WristLeft);
this.DrawBone(skeleton, drawingContext, JointType.WristLeft, JointType.HandLeft);
```

Figura 4.13. Diseño de Brazo Izquierdo en Visual Studio
Fuente: Elaborado por el autor

Para el diseño de la pierna izquierda se ocupara los siguientes puntos figura 4.14, y el código para su realizacion en Visual Studio figura 4.15:

1. HIP RIGHT (cadera derecha)
2. KNEE_ RIGHT (rodilla derecha)
3. ANKLE_ RIGHT (tobillo derecho)
4. FOOT_ RIGHT (pie derecho)

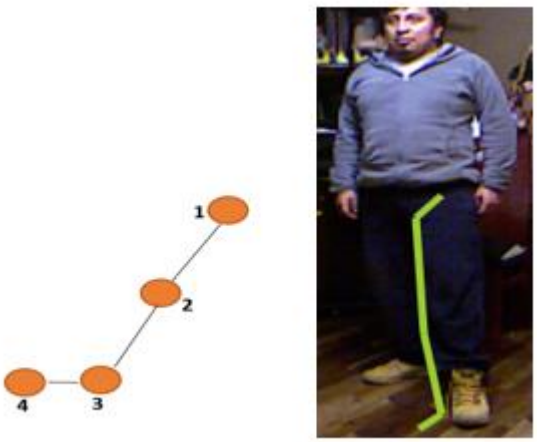


Figura 4.14. Diseño de Pierna Izquierda
Fuente: Elaborado por el autor

```
// Right Leg
this.DrawBone(skeleton, drawingContext, JointType.HipRight, JointType.KneeRight);
this.DrawBone(skeleton, drawingContext, JointType.KneeRight, JointType.AnkleRight);
this.DrawBone(skeleton, drawingContext, JointType.AnkleRight, JointType.FootRight);
```

Figura 4.15. Diseño de la Pierna Izquierda en Visual Studio
Fuente: Elaborado por el autor

El diseño de la pierna derecha se va utilizar los siguientes puntos figura 4.16, y el código a utilizar en Visual Studio figura 4.17:

1. HIP LEFT (cadera izquierda)
2. KNEE_ LEFT (rodilla izquierda)
3. ANKLE_ LEFT (tobillo izquierdo)
4. FOOT_ LEFT (pie izquierdo)

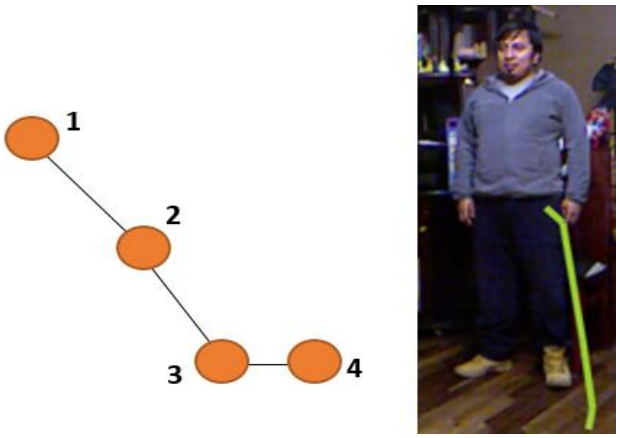


Figura 4.16. Diseño de Pierna Derecha
Fuente: Elaborado por el autor

```
// Left Leg
this.DrawBone(skeleton, drawingContext, JointType.HipLeft, JointType.KneeLeft);
this.DrawBone(skeleton, drawingContext, JointType.KneeLeft, JointType.AnkleLeft);
this.DrawBone(skeleton, drawingContext, JointType.AnkleLeft, JointType.FootLeft);
```

Figura 4.17. Diseño de la Pierna Derecha en Visual Studio
Fuente: Elaborado por el autor

4.3.3 Cálculo de ángulos entre articulaciones

Para la realización de esta aplicación de un estudio ergonómico para personal administrativo, se va a tratar de buscar que es lo que cada usuario está haciendo mal y posteriormente corregirlo, en este caso se debe observar que ciertas partes se mantengan en

una postura relativa, fijarse en los ángulos y la orientación de las articulaciones. En este punto se tiene que tener una idea de cómo calcular los ángulos, que rango de valores deben tener entre cada articulación, mediante la ergonomía nos expresa el ángulo que debe formarse en las articulaciones para tener una postura correcta como se indica en la tabla 4.1.

Tabla 4.1:
Ángulos que forman las articulaciones

Extremidad	Observaciones
Espalda	Se debe mantener la espalda apoyada sobre el respaldo del asiento formando un ángulo de 180
Brazos	Los brazos deben estar a la altura de la mesa, con ángulo de 90
Piernas	se debe mantener las piernas paralelas, sin cruzar y los pies apoyados al suelo , con un ángulo de 90
hombros	se debe mantener relajados con un ángulo de 120

Fuente: Elaborado por el autor

Mediante la cámara de profundidad de Kinect que captura información en torno en 3D, este sensor nos da las coordenadas (X, Y, Z) de los vectores formado entres dos puntos, que se representan en un plano 3D. Se tiene en cuenta que un vector tiene un punto inicial y un final también una magnitud y una dirección. Para la realización de los cálculos se debe definir los puntos de articulaciones que formarían entre ellos como se indica en la tabla 4.2.

Tabla 4.2:
Extremidades definidas en la aplicación.

Extremidad	Puntos de articulaciones que la componen		
	Punto Centro	**Vector (1)**	**Vector (2)**
Brazo izquierdo	Codo	Codo - Hombro Izquierdo	Codo - Muñeca Izquierda
Brazo derecho	Codo	Codo - Hombro Derecho	Codo - Muñeca Derecha
Pierna izquierda	Rodilla	Rodilla - Cadera Izquierda	Rodilla - Tobillo Izquierda
Pierna derecha	Rodilla	Rodilla - Cadera Derecha	Rodilla - Tobillo Derecho
Hombros	Espalda	Espalda-Hombro Izquierdo	Espalda - Hombro Derecho
Espalda	Espina	Espina - Espalda	Espina - Cabeza
Tronco	Cadera	Cadera - Espalda	Cadera - Rodilla

Fuente: Elaborado por el autor

En la tabla 4.2 se indica el proceso para la realización de vectores con las articulaciones y poder realizar el cálculo de los ángulos que se forman entre ellas expuesto en la figura 4.18 utilizabdo la siguiente formula.

$$\cos^{-1}\frac{A.B}{|A||B|}$$

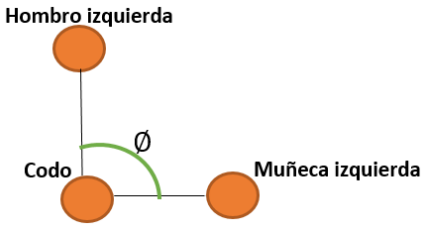


Figura 4.18. Medición de ángulo Brazo izquierdo
Fuente: Elaborado por el autor

En la tabla 4.1 según la especificaiones ergonómica, las articulaciones deben forman un ángulo ideal para ser considera una postura correcta, en la aplicación se planteó tener un rango de un mínimo y un máximo para determinar una postura correcta o incorrecta, en Ecuador la estatura estándar de hombres es de 1.65 cm y de mujeres 1.50 cm por lo que se debe sumar y restar 15° a los ángulos como se muestra en la tabla 4.3.

Tabla 4.3:
Ángulo de las extremidades definidas en la aplicación.

Extremidad	Ángulo postura correcta	Rango Aceptable	
		Mínimo	Máximo
Espalda	180	165	195
Brazos	90	75	105
Piernas	90	75	105
Hombros	120	105	135
Tronco	120	105	135

Fuente: Elaborado por el autor

4.3.4 Análisis y Resultados

Para comprobar los resultados de la aplicación se coloca a una persona a una distancia de 2.6 m del sensor Kinect el cual estará a una altura de 0.86cm sobre el nivel del suelo. Esta configuración se indica en la figura 4.19.

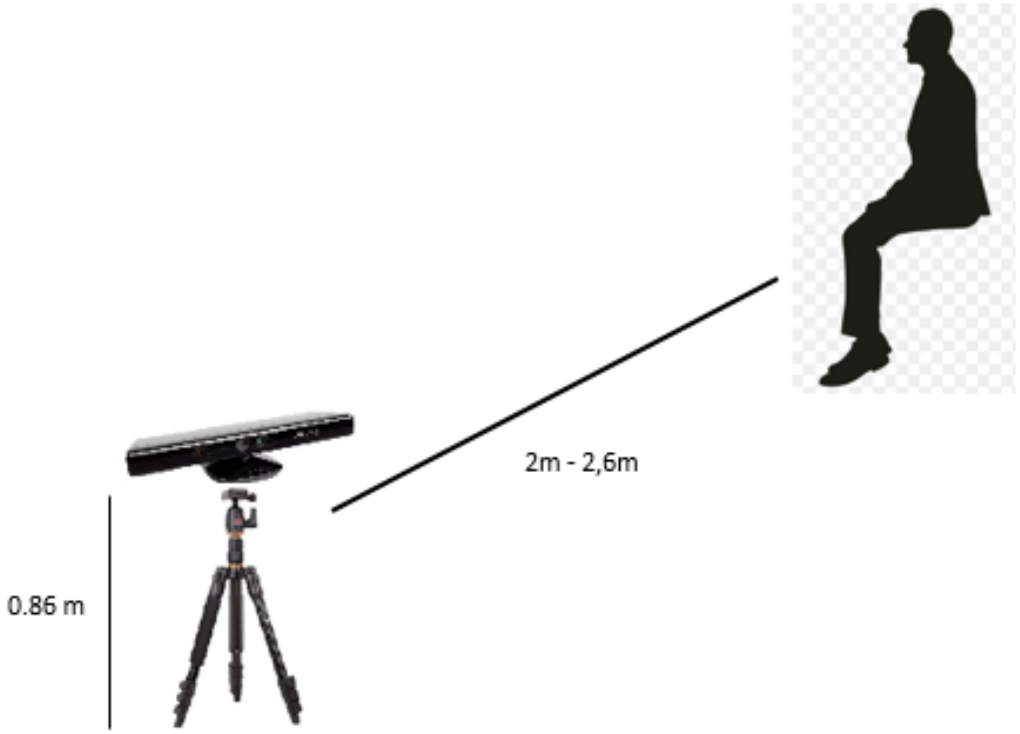


Figura 4.19. Definición de vectores Brazo izquierdo
Fuente: Elaborado por el autor

Se solicita al usuario la respectiva información para crearse un registro figura 4.20, se le pidió a la persona que tomara la posición de trabajo cotidiano en un escritorio, a continuación se va almacenando toda la información que adquiría la cámara Kinect indicando si se encontraba en una postura correcta o incorrecta en la tabla 4.4, figura 4.21 y figura 4.22 se puede visualizar como la aplicación indica el ángulo y la postura que se encuentra el usuario, este será censado cada minuto si la articulación esta fuera de rango mayor de 1min será considerado postura incorrecta y se guardara en el registro, durante este tiempo el usuario puede tener pausas activas siempre y cuando regrese a la postura correcta durante el minuto.

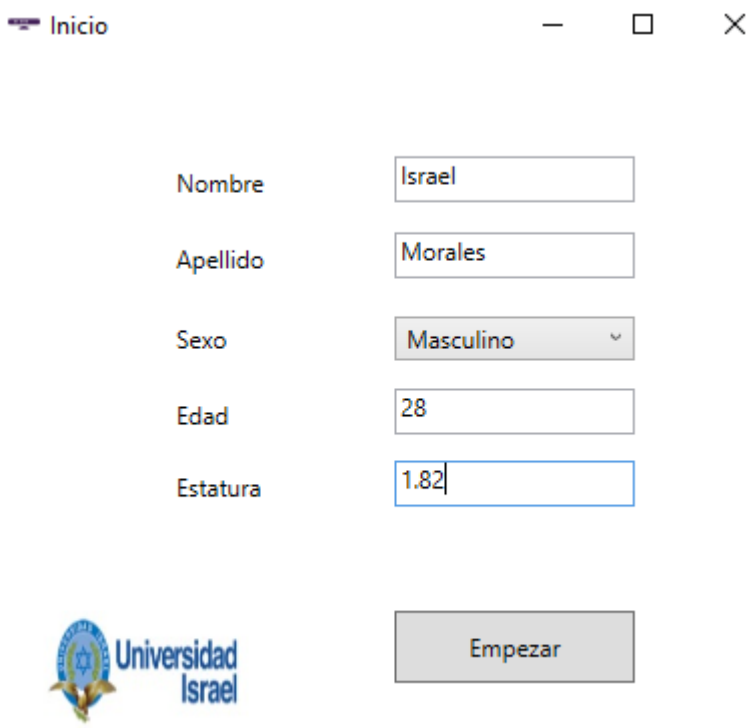


Figura 4.20. Ingreso de datos del usuario
Fuente: Elaborado por el autor

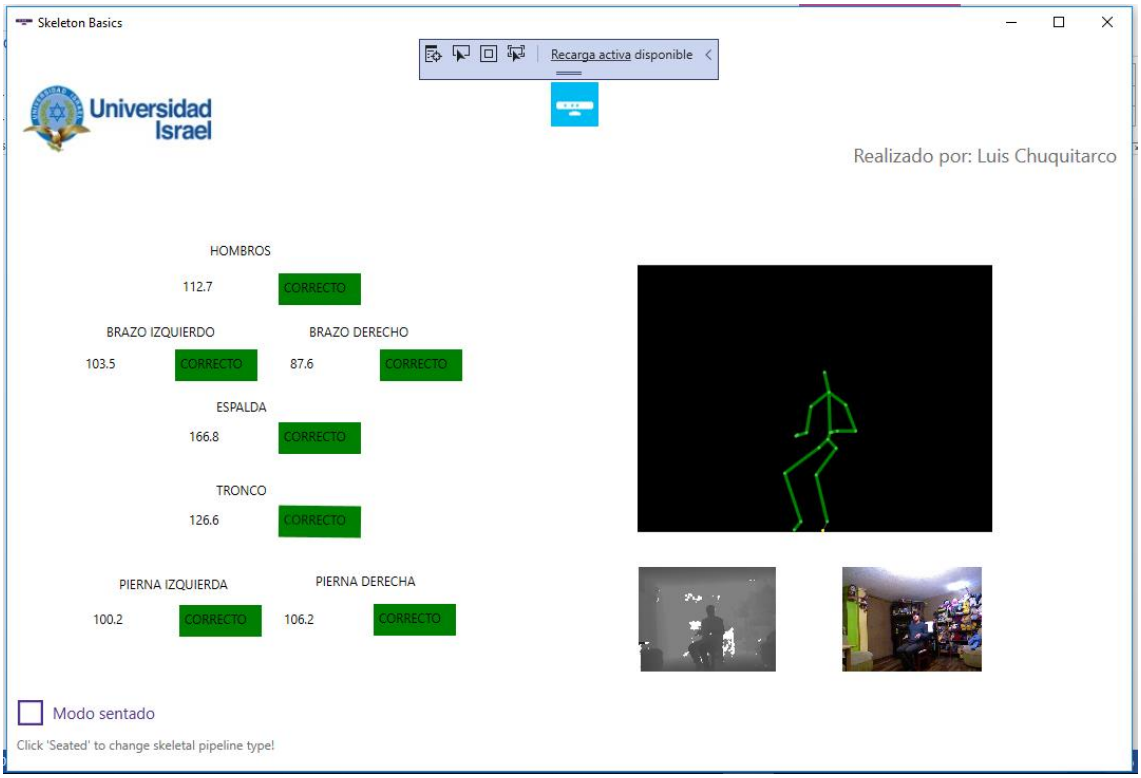


Figura 4.21. Postura Correcta
Fuente: Elaborado por el autor

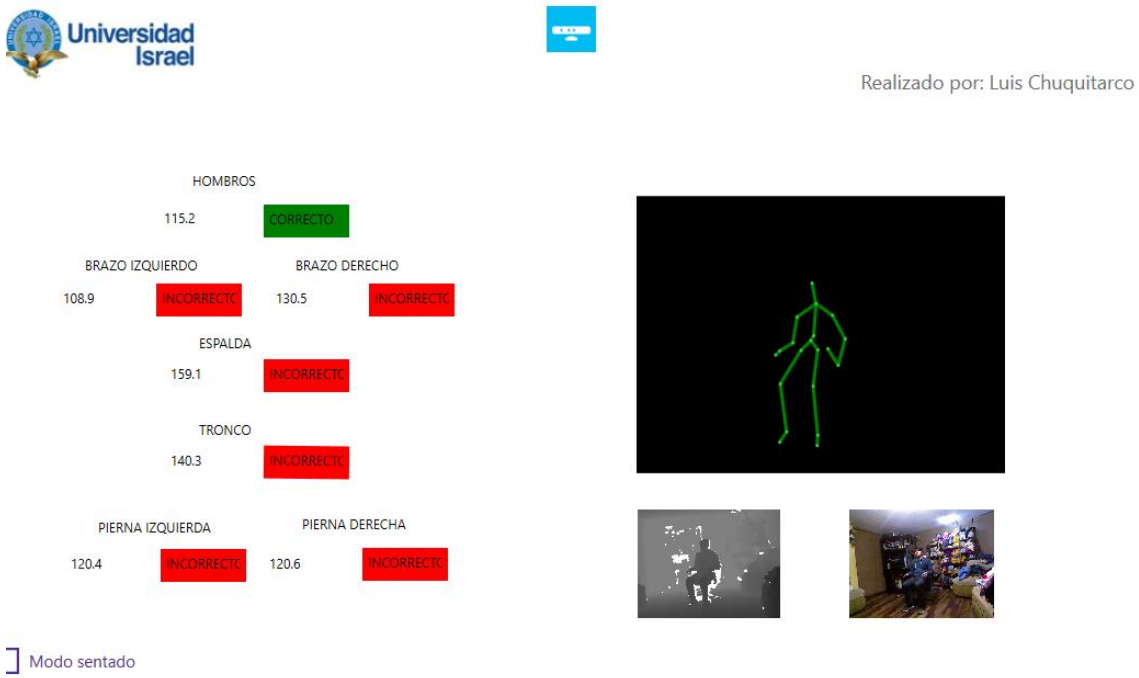


Figura 4.22. Postura Incorrecta
Fuente: Elaborado por el autor

Tabla 4.4:
Registro de postura incorrecta

	Hombros		Brazo Izquierdo		Brazo Derecho		Pierna Izquierda		Pierna Derecha		Espalda		Tronco	
Minuto	Ángulo Medido	Postura	Ángulo Medido	Postura	Ángulo Medido	Postura	Ángulo Medido	Postura	Ángulo Medido	Postura	Ángulo Medido	Postura	Ángulo Medido	Postura
7			110,941491	Incorrecto	120,859507	Incorrecto	119,278499	Incorrecto	119,84776	Incorrecto	158,479685	Incorrecto	139,687729	Incorrecto
8			116,578372	Incorrecto			116,693509	Incorrecto	30,0828826	Incorrecto	168,882616	Incorrecto	128,707434	Incorrecto
9									166,472281	Incorrecto			158,952173	Incorrecto
10			110,20216	Incorrecto	111,444412	Incorrecto	114,438742	Incorrecto	111,082531	Incorrecto	152,601354	Incorrecto	136,423436	Incorrecto
11			117,536061	Incorrecto	114,297693	Incorrecto	112,838065	Incorrecto	113,461699	Incorrecto	159,732946	Incorrecto	120,380839	Incorrecto
12					113,924213	Incorrecto	121,056293	Incorrecto	125,018244	Incorrecto	160,005463	Incorrecto		
13			113,471434	Incorrecto	110,751835	Incorrecto	118,358253	Incorrecto	116,42005	Incorrecto	155,255175	Incorrecto	136,077777	Incorrecto
14			134,980829	Incorrecto	127,24647	Incorrecto	131,749807	Incorrecto	136,949223	Incorrecto	167,346026	Incorrecto	158,717155	Incorrecto
15							131,758724	Incorrecto	132,893401	Incorrecto			160,434251	Incorrecto
16			121,209027	Incorrecto			142,949699	Incorrecto	145,45057	Incorrecto	154,477762	Incorrecto	153,062225	Incorrecto
17							127,66042	Incorrecto					135,503077	Incorrecto
18							117,697666	Incorrecto	111,08655	Incorrecto				
19							151,469345	Incorrecto	164,930004	Incorrecto				

20	121,987562	Incorrecto		
21			109,332219	Incorrecto
22			69,377687	Incorrecto
23			112,944896	Incorrecto

Fuente: Elaborado por el autor

4.3.5 Análisis y de Barras de los resultados Obtenidos

Mediante el registro que otorga la aplicación también graficara los resultados en barras de cada hora laboral, para que el evaluador tenga un control más eficaz, de esta manera se representara gráficamente que postura esta incorrecta y en que determinado tiempo como se muestra en la figura 4.23 y figura 4.24 .

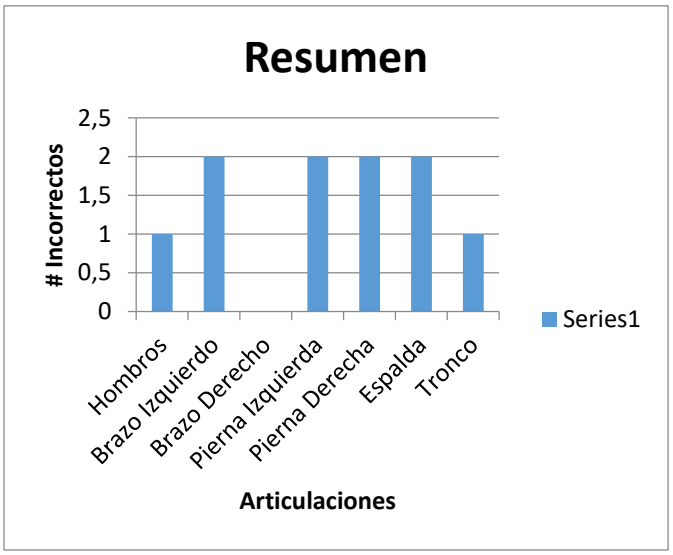


Figura 4.23. Gráfico de Barras de 8:00 – 9:00 am
Fuente: Elaborado por el autor

Mediante el grafico de barras que es calculado durante la la hora de 8:00 a 9:00 se puede observar que la articulación que mayor índice de mala postura a tenido es el brazo izquierdo, la pierna izquierda y derecha, con estas mediciones el evaludor puede determinar que el usuario tiene mayor índice de una postura incorrecta en las extremidades antes mencionadas, no se puede asumir que son pausas activas porque el tiempo sobrepasa según la norma ergonómica “inshp” que se refiere a treita segundos para tener monentos de descanso

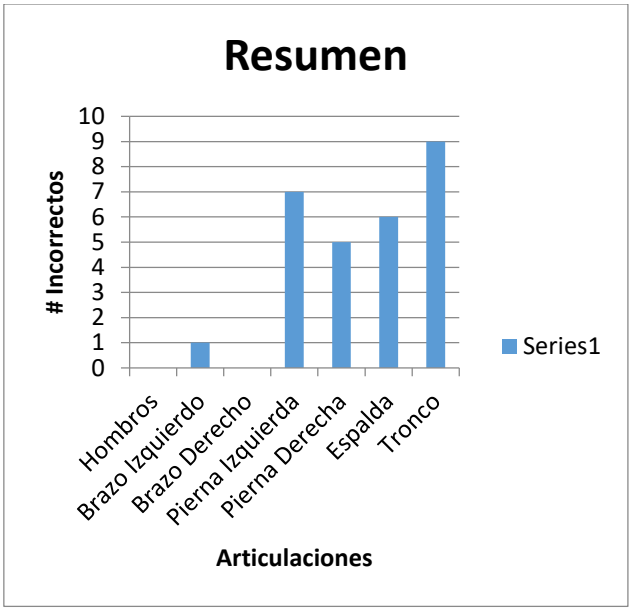


Figura 4.24. Gráfico de Barras de 9:00 – 10:00 am
Fuente: Elaborado por el autor

En el segundo grafico la hora de 9.00 a 1000 se puede observar que el usuario tiene una mayor postura incorrecta en el tronco por lo que el evaluador puede determinar que no se encuentra recto ni apoyándose en el espaldar de la silla ergonómica produciendo deformación en su angulo medido.

CONCLUSIONES

- Mediante la realización de un estudio ergonómico insht “Instituto Nacional de Seguridad y Bienestar en el trabajo” los aspectos más notables son los ángulos que forman las articulaciones del esqueleto humano, es decir la postura del usuario.

- Mediante el análisis ergonómico realizado por la aplicación para la captura de las articulaciones del usuario se puede desarrollar el algoritmo matemático basado en programación Visual Studio garantizando la captura de los 7 ángulos que se forman entre ellas desde (X° a Y°) una vez ya obtenidos estos valores permite definir si la posición de la persona se encuentra en una postura correcta o incorrecta.

- A través de esta investigación se pudo evidenciar que la cámara Kinect cuenta con varias propiedades para el diseño de diferentes aplicaciones en distintas áreas; industrial, entretenimiento y en la medicina, dado que la Kinect tiene un costo mínimo para su adquisición y su entorno de programación es amigable para el desarrollo de las aplicaciones.

- Mediante un registro de control que emite la aplicación, permitirá que el evaluador tenga un enfoque más eficaz para determinar si el usuario en sus horas laborales tiene mayor índice de una postura incorrecta, con este análisis podrá dar soluciones evitando enfermedades que puede ocasionar en una mala postura a un largo plazo.

- Se determinó que, tras la realización de las pruebas de funcionamiento del prototipo por parte del personal administrativo, la misma considera que, el dispositivo permite el análisis de una postura correcta o incorrecta.

- Mediante la norma investigada salud e higiene ocupacional el cálculo que se realiza al ángulo formado de la articulaciones se debe sumar y resta 20° a todas las mediciones calculadas, por el estándar de media que poseen las personas en países Americanos y Europeos por la altura mínima de los habitantes es de

1.75cm los hombres y las mujeres 1.65cm, para nuestro países se le debe sumar y resta 15° debido a que la estura minima de hombres es 1.65cm y de mujeres 1.50 cm a esta medida fue desarrollada la aplicación.

RECOMENDACIONES

- Se recomienda la utilización de dos cámaras de profundidad, con la posibilidad de capturar el espacio de diversos ángulos y con diversos campos de visión, se podrá realizar un escaneo más fiel y obtener información acerca de esas áreas que aparecen sombreadas.

- Asimismo, se recomienda la utilización del dispositivo Kinect V2, mejorando la detección de personas, captación del entorno y profundidad, permitiendo el rastreo de la posición de la mano (abierta, cerrada), con la creación de varias aplicaciones a la vez.

- Actualmente se necesita que el espacio quede bien despejado y sin tener obstáculos para tener un correcto seguimiento del usuario, es recomendable encontrar una manera de poder reducir este espacio a uno más cerrado.

- Mejorar el interfaz del usuario, realizar cambios en el aspecto grafico realizando una interfaz más atractiva para la persona que la utiliza tenga un mejor entendimiento del sistema

- Es recomendable la creación de una base de datos para que el evaluador tenga un historial, del personal administrativo que es evaluado para la detección de una postura correcta o incorrecta.

- Colocar la Kinect a una altura de 0.86cm desde el piso y al usuario colocar una vista perpendicular a la cámara con el ángulo de visibilidad de 43° verticalmente a un rango de 2.6m a 3m.

BIBLIOGRAFÍA

Baroja Payan, E., Juarez Rivera, V., Rojas Duran, R., & Velasquez Calderon, R. (8 de 2014). *Aplicación de la técnica RULA en el área de empaquetado mediante tecnología Kinect.* Obtenido de Aplicación de la técnica RULA en el área de empaquetado mediante tecnología Kinect.: https://www.ride.org.mx/index.php/RIDE/article/view/121/543

Bernal Iñiguez, J. (2014). *Diseño, construccion e implementacion de un sistema de captura de movimiento para analisis ergonomico de iesgo laboral de extremidades superiores.* Cuenca.

Ergonomía. (28 de Junio de 2016). *La salud y seguridad en el trabajo*. Obtenido de La salud y seguridad en el trabajo: http://www.espalda.org/divulgativa/su_espalda/trabajadores/workers.asp

Escobar, F. (17 de 1 de 2014). *Aspectos Importantes*. Obtenido de Aspectos Importantes: http://freimarescobar13.over-blog.com/2014/01/la-ergonomia-y-el-ambito-laboral.html

Instituto Nacional de Seguridad e Higiene en el Trabajo. (2003). *NP Erga-FP 35. Trbajo en posicion sentado.* España.

Larumbe Bergera, A., & Villanueva Larre, A. (2015). *Modelado 3D de cabeza mediante Kinect.* Pamplona.

Lau, D. (27 de 11 de 2013). *La ciencia detrás de Kinects o Kinect 1.0 versus 2.0*. Obtenido de La ciencia detrás de Kinects o Kinect 1.0 versus 2.0: https://www.gamasutra.com/blogs/DanielLau/20131127/205820/The_Science_Behind_Kinects_or_Kinect_10_versus_20.php

Leadbetter, R. (20 de julio de 2010). *Las especificaciones técnicas finales de Kinect*. Obtenido de Las especificaciones técnicas finales de Kinect: https://www.eurogamer.es/articles/digitalfoundry-especificaciones-finales-kinect

Lucero Guerrero, P. (2014). *Diseño, Experimentación y Evaluación de prácticas en el Área de Ergonomia, modelado Biomecánico y Análisis de movimiento para un Laboratorio de Ingeniería Biomedica en la Universidad Politecnica Salesiana* . Cuenca.

Martinez Castro, V. (28 de 02 de 2012). *Ergonomía en trabajos de oficina*. Obtenido de Ergonomía en trabajos de oficina: http://prevencionar.com/2012/02/28/ergonomia-en-trabajos-de-oficina/

Martino, L., & Lema, G. (2011). *Faceval*. Obtenido de Faceval: http://iie.fing.edu.uy/investigacion/grupos/gti/timag/trabajos/2011/faceval/home/index.html

Mas, D., & J. A. (2015). *Evaluación de la repetitividad de movimientos mediante el método JSI*. Obtenido de Evaluación de la repetitividad de movimientos mediante el método JSI: https://www.ergonautas.upv.es/metodos/jsi/jsi-ayuda.php

Microsoft. (13 de Septiembre de 2013). *Kinect para Windows SDK v1.8*. Obtenido de Kinect para Windows SDK v1.8: https://www.microsoft.com/en-us/download/details.aspx?id=40278

Murillo, A. (01 de 03 de 2012). *Diferencias entre Kinect Xbox 360 y Kinect for Windows*. Obtenido de Diferencias entre Kinect Xbox 360 y Kinect for Windows: http://www.kinectfordevelopers.com/es/2012/03/01/diferencias-entre-kinect-xbox-360-y-kinect-for-windows/

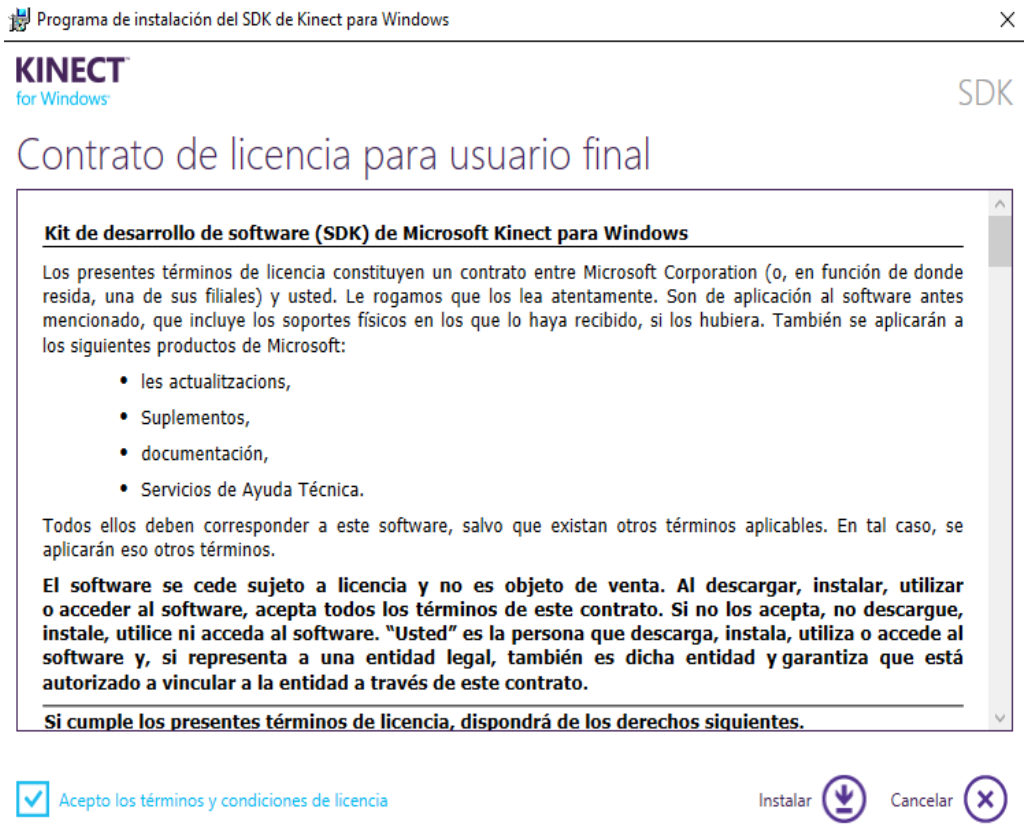
Murillo, A. (s.f.). *Kinect for developers*. Obtenido de Kinect for developers: http://www.kinectfordevelopers.com/es/2012/11/06/que-es-el-dispositivo-kinect/

Villacres Cornejo, L. (2018). *Diseño de un sistema interfaz para el reconocimiento y traducción de gestos corporales al lenguaje natural (escrito, hablado) mediante el sensor Kinect de Microsoft, para personas con capacidades diferentes.* Guayaquil.

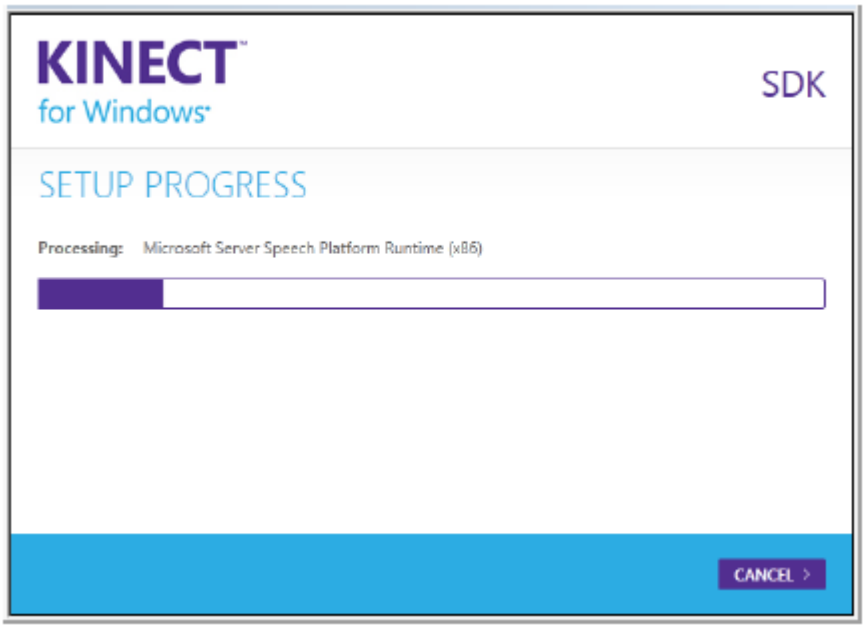
ANEXOS

ANEXO 1: MANUAL DE INSTALACIÓN

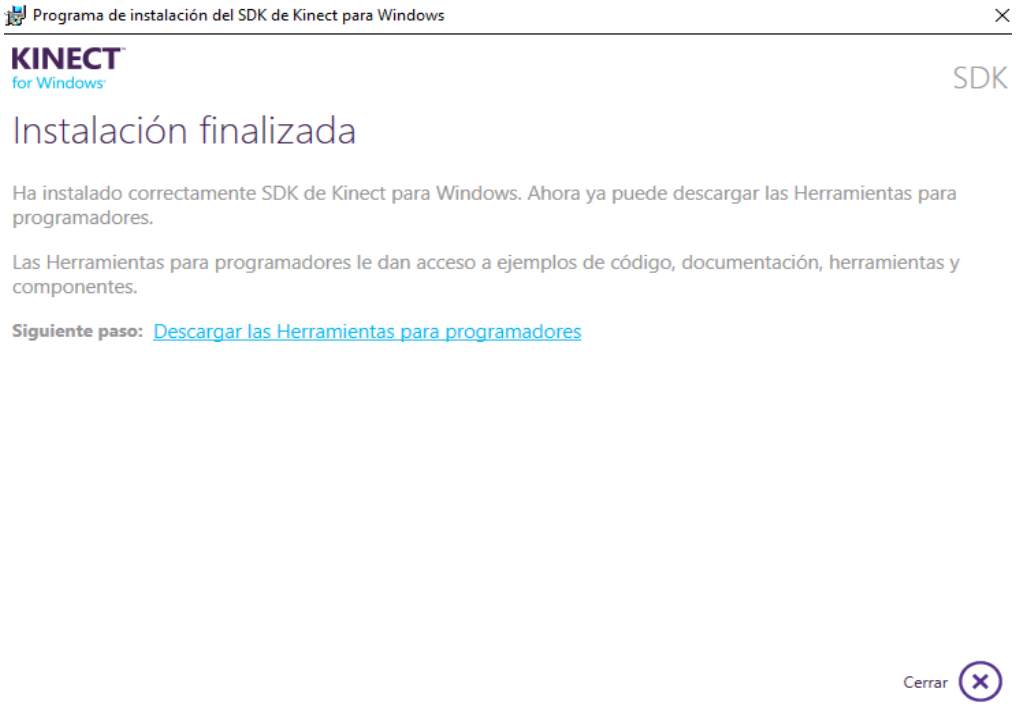
1. Instalar el SDK Kinect V1.8 de Microsoft que se encuentra en la carpeta de Instalaciones en el cd, o se puede descargar de la página oficinal de Microsoft de la siguiente url: https://www.microsoft.com/en-us/download/details.aspx?id=40278



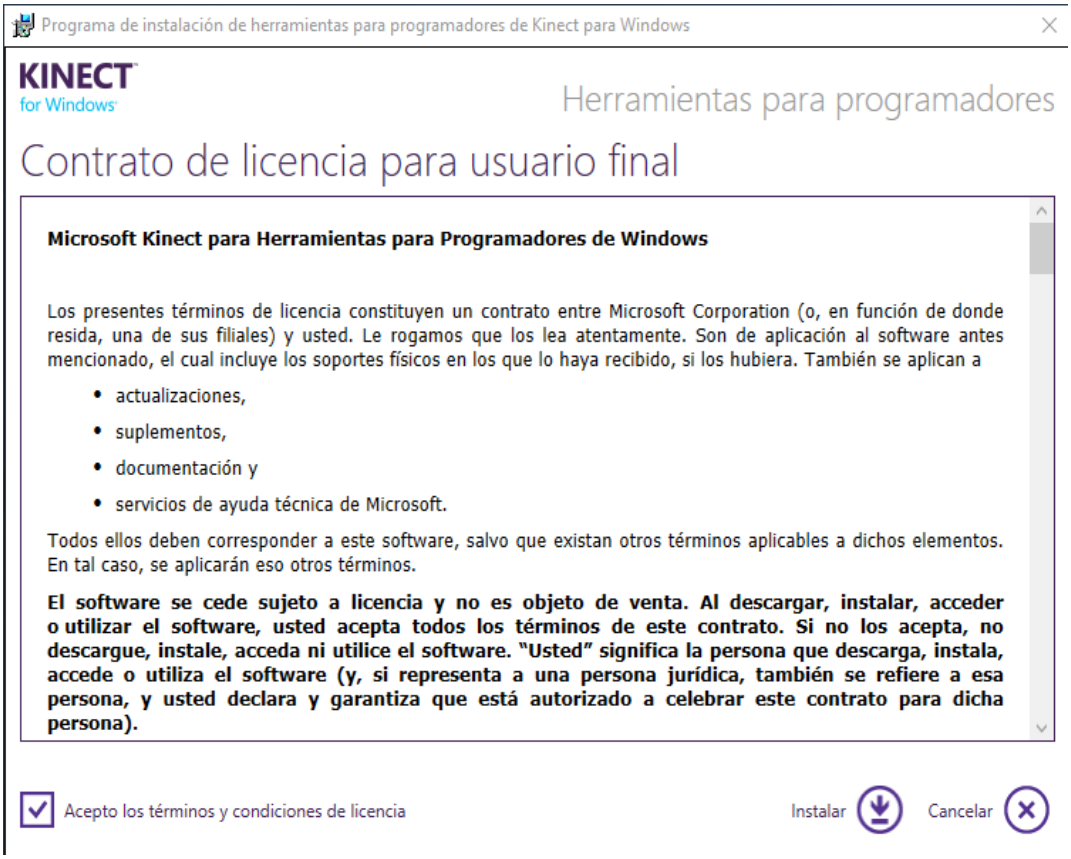
Clic en aceptar los términos e instalar.



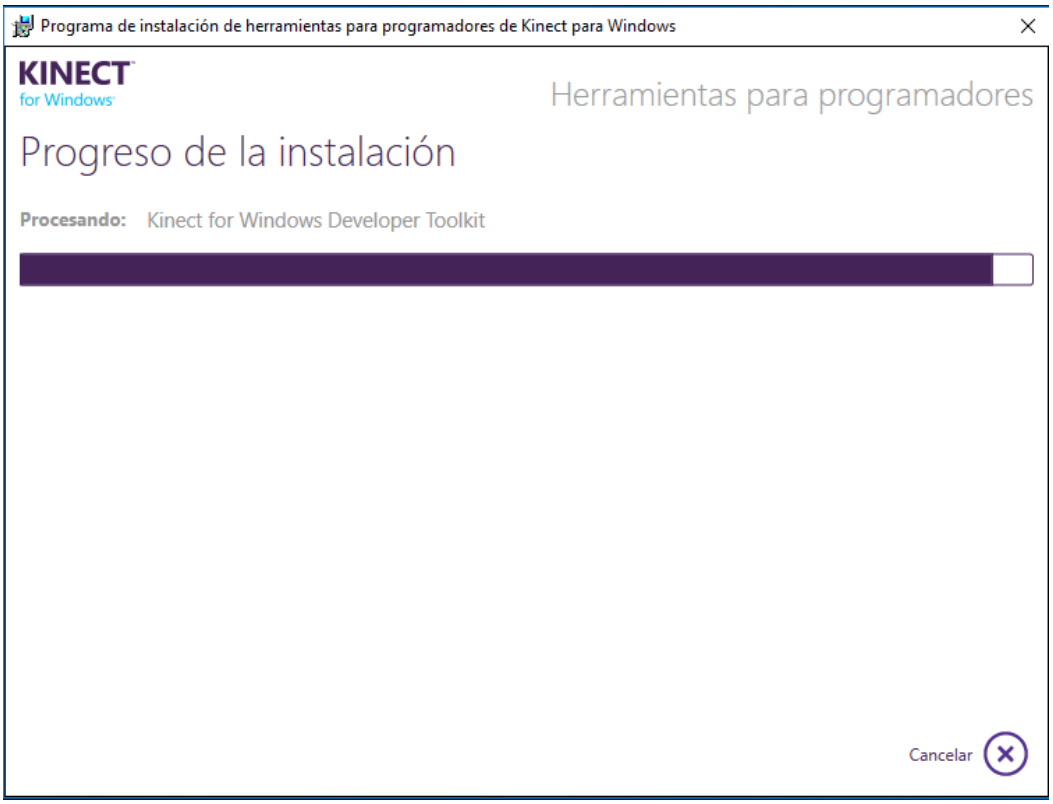
Clic en cerrar.



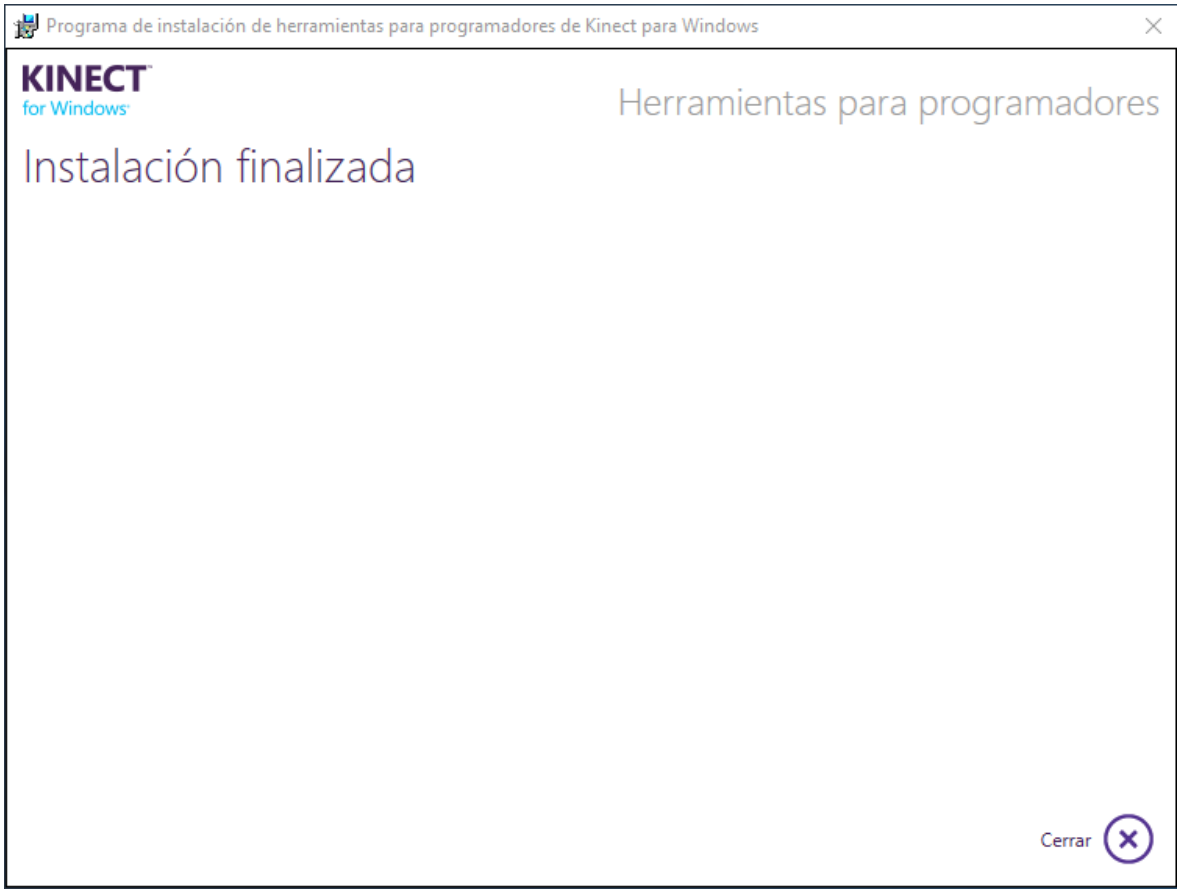
2. Instalar el Developer toolkit v1.8 de Microsoft que se encuentra en la carpeta de Instalaciones en el cd, o se puede descargar de la página oficinal de Microsoft de la siguiente url: https://www.microsoft.com/en-us/download/details.aspx?id=40276



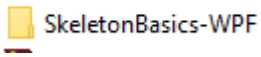
Clic en aceptar los términos e instalar.



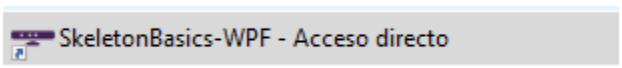
Clic en cerrar.



3. Copiar la carpeta SkeletonBasic-WPF en documentos, que se encuentra en el cd.



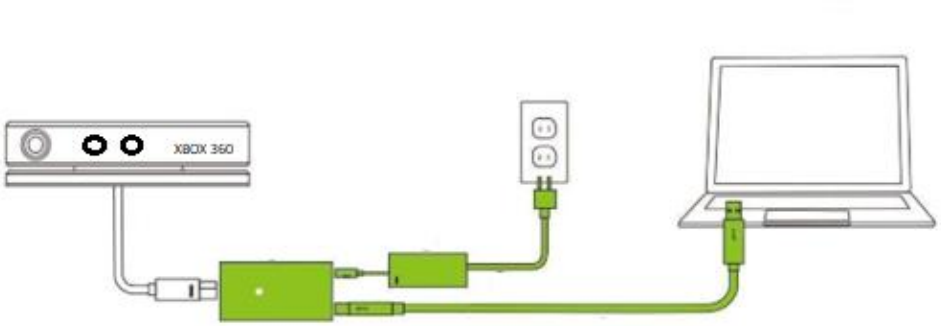
Clic en el acceso directo de la aplicación



4. Reiniciar el equipo.

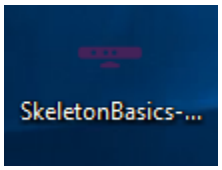
ANEXO 2: MANUAL DE USUARIO

1. Conectar el Kinect al PC con su respectivo adaptador.



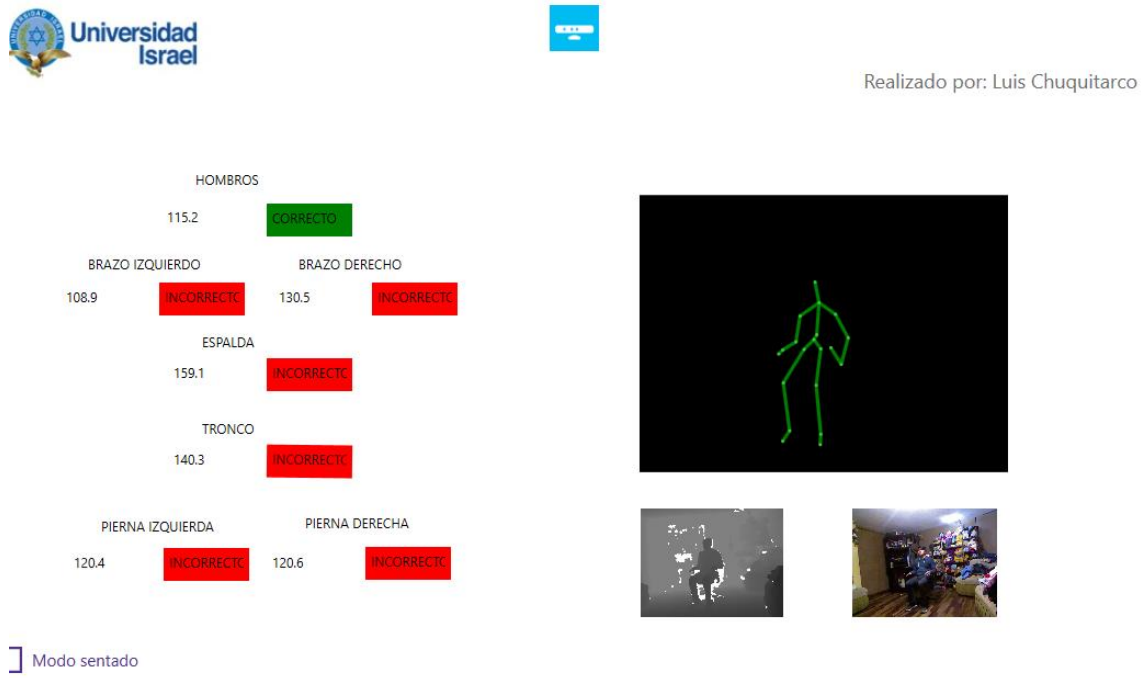
2. Abrir la aplicación.

   Clic en el icono SkeletonBasics-WPF que aparece en el escritorio o en la barra de inicio.



3. Ingresar los datos del Usuario a evaluar y luego clic empezar

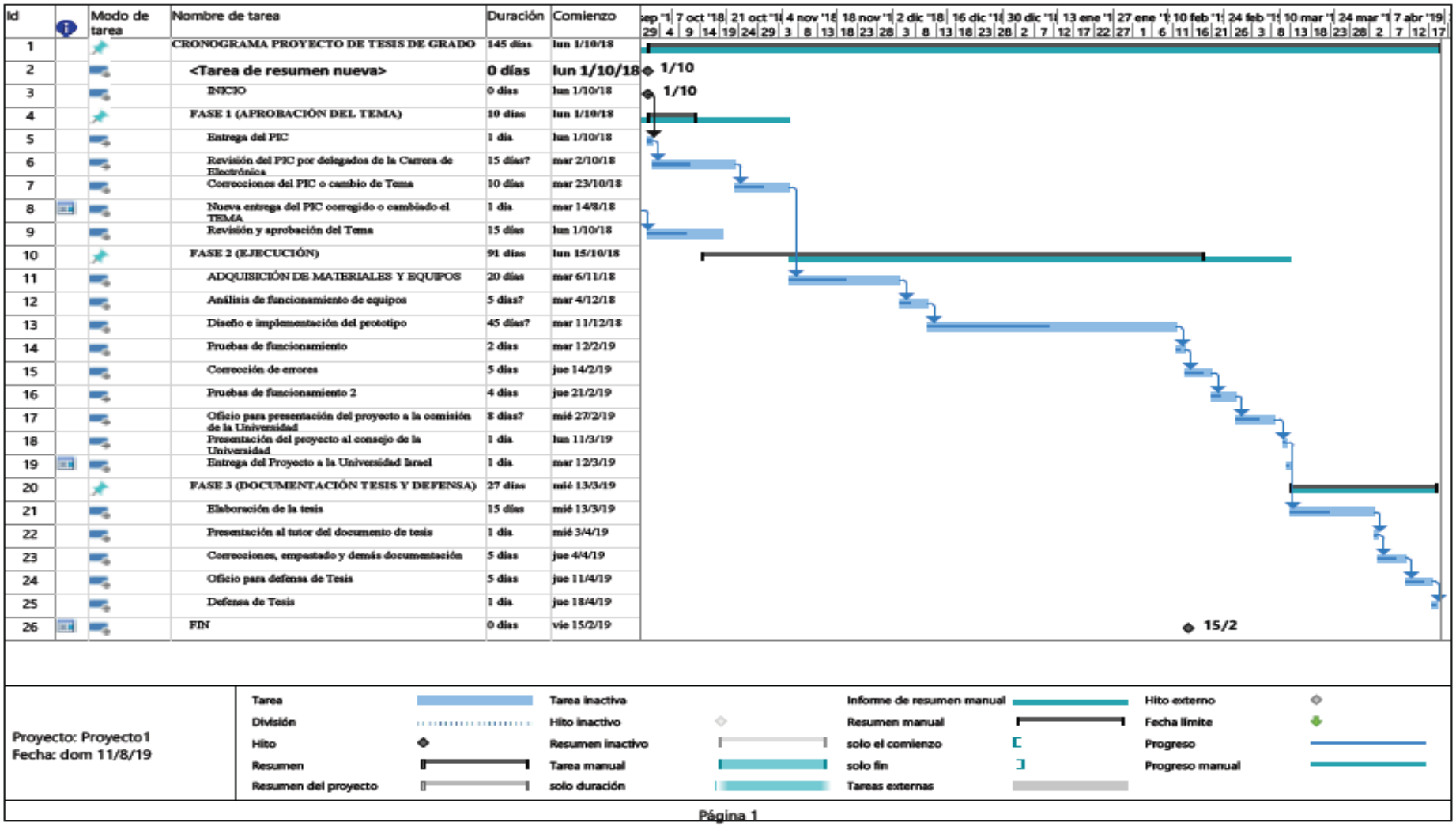
4. Dejar a la Kinect realizando el estudio durante el tiempo deseado.



5. Clic en cerrar la aplicación

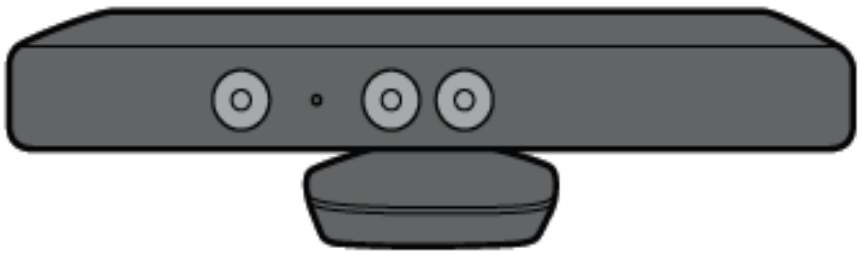
6. Ingresar en la carpeta reporte para poder revisar la documentación que se crea del registro de datos

ANEXO 3: CRONOGRAMA

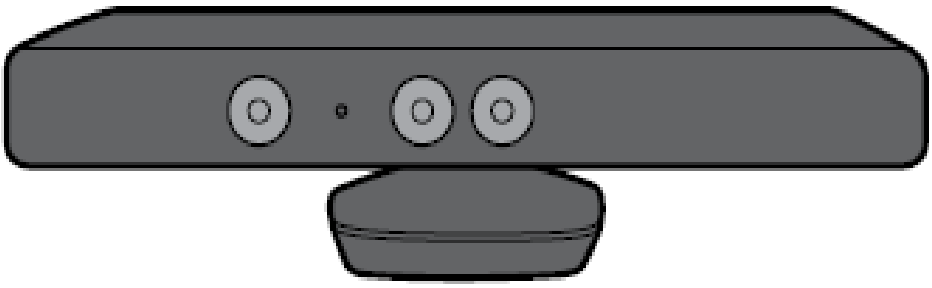


Id	Modo de tarea	Nombre de tarea	Duración	Comienzo
1		CRONOGRAMA PROYECTO DE TESIS DE GRADO	145 días	lun 1/10/18
2		<Tarea de resumen nueva>	0 días	lun 1/10/18
3		INICIO	0 días	lun 1/10/18
4		FASE 1 (APROBACIÓN DEL TEMA)	10 días	lun 1/10/18
5		Entrega del PIC	1 día	lun 1/10/18
6		Revisión del PIC por delegados de la Carrera de Electrónica	15 días?	mar 2/10/18
7		Correcciones del PIC o cambio de Tema	10 días	mar 23/10/18
8		Nueva entrega del PIC corregido o cambiado el TEMA	1 día	mar 14/8/18
9		Revisión y aprobación del Tema	15 días	lun 1/10/18
10		FASE 2 (EJECUCIÓN)	91 días	lun 15/10/18
11		ADQUISICIÓN DE MATERIALES Y EQUIPOS	20 días	mar 6/11/18
12		Análisis de funcionamiento de equipos	5 días?	mar 4/12/18
13		Diseño e implementación del prototipo	45 días?	mar 11/12/18
14		Pruebas de funcionamiento	2 días	mar 12/2/19
15		Corrección de errores	5 días	jue 14/2/19
16		Pruebas de funcionamiento 2	4 días	jue 21/2/19
17		Oficio para presentación del proyecto a la comisión de la Universidad	8 días?	mié 27/2/19
18		Presentación del proyecto al consejo de la Universidad	1 día	lun 11/3/19
19		Entrega del Proyecto a la Universidad Israel	1 día	mar 12/3/19
20		FASE 3 (DOCUMENTACIÓN TESIS Y DEFENSA)	27 días	mié 13/3/19
21		Elaboración de la tesis	15 días	mié 13/3/19
22		Presentación al tutor del documento de tesis	1 día	mié 3/4/19
23		Correcciones, empastado y demás documentación	5 días	jue 4/4/19
24		Oficio para defensa de Tesis	5 días	jue 11/4/19
25		Defensa de Tesis	1 día	jue 18/4/19
26		FIN	0 días	vie 15/2/19

ANEXO 4: DATASHEET KINECT



XBOX 360 KINECT SENSOR

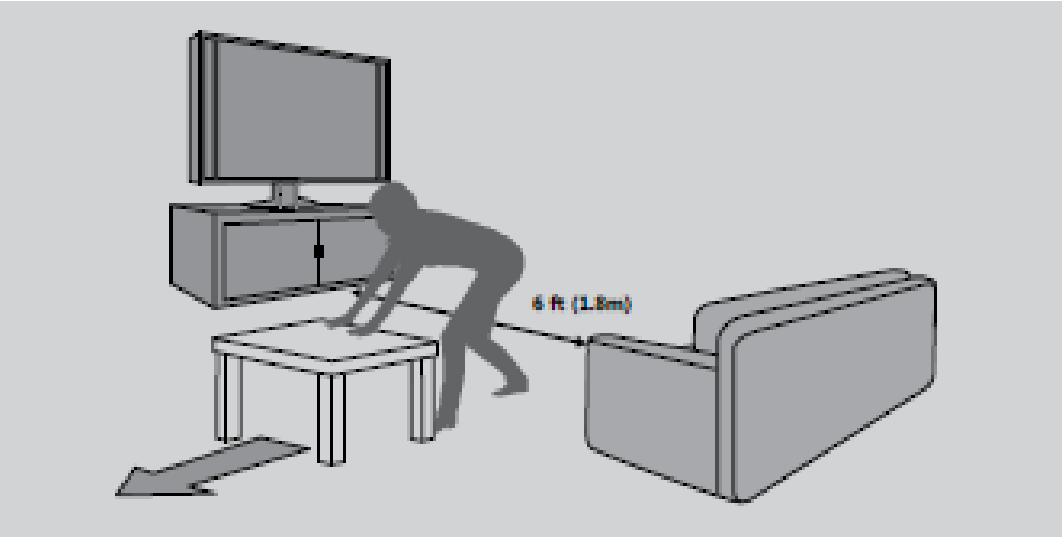


Xbox 360 Kinect Sensor

Thanks for choosing the Xbox 360® Kinect™ Sensor. The Kinect sensor offers a revolutionary new way to play: you're the controller. Just move around and see what happens. Control your Xbox 360 with a wave of your hand. The only experience you need is life experience.

The Kinect sensor is for use with the Xbox 360 video game and entertainment system. To learn more about using the Kinect sensor with a specific game, see the documentation that came with your game.

ADEQUATE SPACE FOR PLAYING



The Kinect sensor needs to be able to see you, and you need room to move. The sensor can see you when you play approximately 6 feet (2 meters) from the sensor. For two people, you should play approximately 8 feet (2.5 meters) from the sensor.

Play space will vary based on your sensor placement and other factors. See your game's instructions for more information about whether it requires only part of the sensor play space.

⚠ WARNING

Make sure you have enough space to move freely while playing

Gameplay with your Kinect sensor may require varying amounts of movement. Make sure you won't hit, run into, or trip over other players, bystanders, pets, furniture, or other objects when playing. If you will be standing and/or moving during gameplay, you will also need good footing.

Before playing:

- Look in all directions (right, left, forward, backward, down, and up) for things you might hit or trip over.
- Make sure your play space is far enough away from windows, walls, stairs, etc.
- Make sure there is nothing you might trip on—toys, furniture, or loose rugs, for example. Also, be aware of children and pets in the area. If necessary, move objects or people out of the play space.
- Don't forget to look up. Be aware of light fixtures, fans, and other objects overhead when assessing the play space.

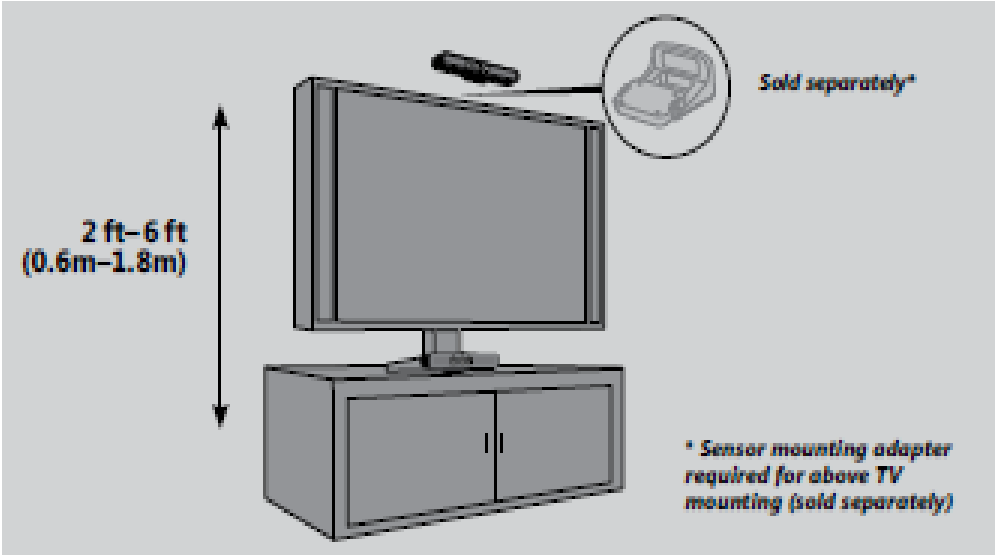
While playing:

- Stay far enough away from the television to avoid contact.
- Keep enough distance from other players, bystanders, and pets. This distance may vary between games, so take account of how you are playing when determining how far away you need to be.
- Stay alert for objects or people you might hit or trip on. People and objects can move into the area during gameplay, so always be alert to your surroundings.

Make sure you always have good footing while playing:

- Play on a level floor with enough traction for game activities.
- Make sure you have appropriate footwear for gaming (no high heels, flip flops, etc.) or are barefoot, if appropriate.

CHOOSE A LOCATION FOR YOUR SENSOR



For the best play space and sensor performance, place your sensor between 2 feet and 6 feet (0.6 and 2 meters) high, the closer to the low or high limit, the better. Also:

- Place the sensor on a stable surface.
- Make sure the sensor is aligned with the center of your TV, and as close as possible to the front edge of the table or shelf.
- Make sure to place the sensor in a location where it will not fall or be struck during gameplay.
- Do not put the sensor on your console.
- Do not place the sensor on or in front of a speaker or a surface that vibrates or makes noise.
- Keep the sensor out of direct sunlight.
- Do not use near any heat sources. Use the sensor within its specified operating temperature range of 41 °F – 95 °F (5 °C – 35 °C). If the sensor is exposed to an environment outside its prescribed range, turn it off and allow the temperature to stabilize within the specified range before using the sensor again.

IMPORTANT
Only adjust the sensor location by moving the base. Do not adjust the sensor viewing angle by hand, by tilting the sensor on its base. After setup is complete, let the sensor motors adjust the viewing angle, or you risk damaging your sensor.

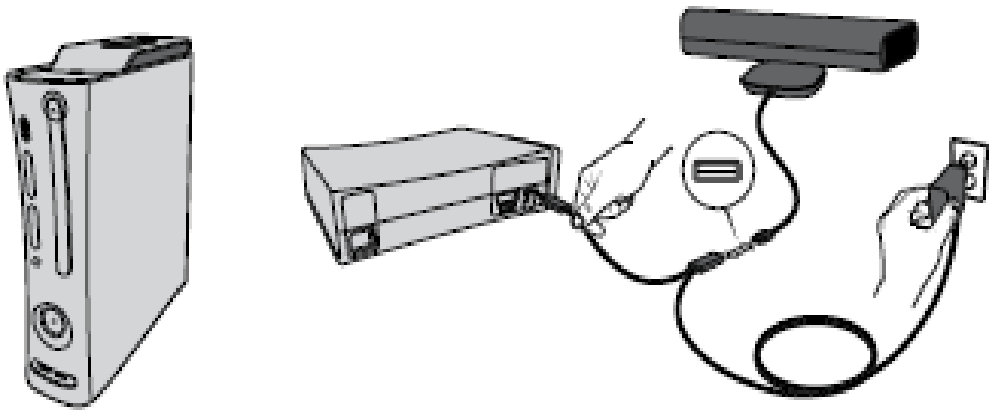
⚠ WARNING
Arrange all cables and cords so that people and pets are not likely to trip over or accidentally pull on them as they move around or walk through the area. When the sensor and console are not in use, you may need to disconnect all cables and cords from the sensor and console to keep them out of the reach of children and pets. Do not allow children to play with cables and cords.

⚠ WARNING
Avoid Glare
To minimize eyestrain from glare, try the following:

- Position yourself at a comfortable distance from your television or monitor and the Kinect sensor.
- Place your television or monitor and Kinect sensor away from light sources that produce glare, or use window blinds to control light levels.
- Choose soothing natural light that minimizes glare and eyestrain and increases contrast and clarity.
- Adjust your television or monitor brightness and contrast.

Connect the Sensor to Your Original Xbox 360 Console

The sensor only works with the back USB port on an original Xbox 360 console.



Original Xbox 360

To connect to your original Xbox 360 console:

1. Unplug any accessories from the back USB port on your console.
2. Plug the sensor into the USB/power cable.
3. Plug the USB/power cable into your console's back USB port.
4. Plug the AC adapter end of the USB/power cable into a wall outlet.

Use only the USB/power cable that is shipped with the product or is given to you by an authorized repair center.

If you have an original Xbox 360 console with no hard drive, you should also attach a storage device with at least 256 MB free space. You can use an Xbox 360 Hard Drive, Xbox 360 Memory Unit, or a USB flash drive.

ANEXO 5: CÓDIGO DE LA APLICACIÓN

```
//------------------------------------------------------------------------
-------
// <copyright file="MainWindow.xaml.cs" company="Microsoft">
//     Copyright (c) Microsoft Corporation.  All rights reserved.
// </copyright>
//------------------------------------------------------------------------
-------

using System.IO;
using System;
using System.Windows;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Media.Media3D;
using Microsoft.Kinect;

using Microsoft.Office.Interop.Excel;
using Microsoft.Samples.Kinect.SkeletonBasics.Paginas;

namespace Microsoft.Samples.Kinect.SkeletonBasics
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : System.Windows.Window
    {
        /// <summary>
        /// Width of output drawing
        /// </summary>
        private const float RenderWidth = 640.0f;

        /// <summary>
        /// Height of our output drawing
        /// </summary>
        private const float RenderHeight = 480.0f;

        /// <summary>
        /// Thickness of drawn joint lines
        /// </summary>
        private const double JointThickness = 3;

        /// <summary>
        /// Thickness of body center ellipse
        /// </summary>
        private const double BodyCenterThickness = 10;

        /// <summary>
        /// Thickness of clip edge rectangles
        /// </summary>
        private const double ClipBoundsThickness = 10;

        /// <summary>
        /// Brush used to draw skeleton center point
        /// </summary>
        private readonly Brush centerPointBrush = Brushes.Blue;

        /// <summary>
        /// Brush used for drawing joints that are currently tracked
```

```
        /// </summary>
        private readonly Brush trackedJointBrush = new
SolidColorBrush(Color.FromArgb(255, 68, 192, 68));

        /// <summary>
        /// Brush used for drawing joints that are currently inferred
        /// </summary>
        private readonly Brush inferredJointBrush = Brushes.Yellow;

        /// <summary>
        /// Pen used for drawing bones that are currently tracked
        /// </summary>
        private readonly Pen trackedBonePen = new Pen(Brushes.Green, 6);

        /// <summary>
        /// Pen used for drawing bones that are currently inferred
        /// </summary>
        private readonly Pen inferredBonePen = new Pen(Brushes.Gray, 1);

        /// <summary>
        /// Active Kinect sensor
        /// </summary>
        private KinectSensor sensor;

        /// <summary>
        /// Drawing group for skeleton rendering output
        /// </summary>
        private DrawingGroup drawingGroup;

        /// <summary>
        /// Drawing image that we will display
        /// </summary>
        private DrawingImage imageSource;

        //excel
        private Workbook wb;
        private Worksheet ws;
        private string nombre_usuario;
        //contador excel
        private int contador_excel=0;
        //contador de hojas
        private int contador_hoja_excel = 0;

        //contador de posiciones incorrectas en las articulaciones
durante una hora
        private int contador_brazo_derecho = 0;
        private int contador_brazo_izquierdo = 0;
        private int contador_espalda = 0;
        private int contador hombros = 0;
        private int contador_pierna_derecha = 0;
        private int contador_pierna_izquierda = 0;
        private int contador_espalda_rodilla = 0;
        private bool bd_generar_grafica = false;

        //bandera de inicio
        private bool bd_inicio = false;

        //angulos
        private Double angulo_brazo_derecho;
        private Double angulo_brazo_izquierdo;
        private Double angulo_espalda;
```

```
        private Double angulo_hombros;
        private Double angulo_pierna_derecha;
        private Double angulo_pierna_izquierda;
        private Double angulo_espalda_rodilla;

        /// <summary>
        /// Initializes a new instance of the MainWindow class.
        /// </summary>
        public MainWindow()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Draws indicators to show which edges are clipping skeleton
data
        /// </summary>
        /// <param name="skeleton">skeleton to draw clipping information
for</param>
        /// <param name="drawingContext">drawing context to draw
to</param>
        private static void RenderClippedEdges(Skeleton skeleton,
DrawingContext drawingContext)
        {
            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Bottom))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(0, RenderHeight - ClipBoundsThickness,
RenderWidth, ClipBoundsThickness));
            }

            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Top))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(0, 0, RenderWidth, ClipBoundsThickness));
            }

            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Left))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(0, 0, ClipBoundsThickness, RenderHeight));
            }

            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Right))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(RenderWidth - ClipBoundsThickness, 0,
ClipBoundsThickness, RenderHeight));
            }
        }

        /// <summary>
        /// Execute startup tasks
```

```
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void WindowLoaded(object sender, RoutedEventArgs e)
        {
            // Create the drawing group we'll use for drawing
            this.drawingGroup = new DrawingGroup();

            // Create an image source that we can use in our image
control
            this.imageSource = new DrawingImage(this.drawingGroup);

            // Display the drawing using our image control
            Image.Source = this.imageSource;



            // Look through all sensors and start the first connected
one.
            // This requires that a Kinect is connected at the time of
app startup.
            // To make your app robust against plug/unplug,
            // it is recommended to use KinectSensorChooser provided in
Microsoft.Kinect.Toolkit (See components in Toolkit Browser).
            foreach (var potentialSensor in KinectSensor.KinectSensors)
            {
                if (potentialSensor.Status == KinectStatus.Connected)
                {
                    this.sensor = potentialSensor;
                    break;
                }
            }

            if (null != this.sensor)
            {
                // Turn on the skeleton stream to receive skeleton frames
                this.sensor.SkeletonStream.Enable();

                // Add an event handler to be called whenever there is
new color frame data
                this.sensor.SkeletonFrameReady +=
this.SensorSkeletonFrameReady;


sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);

sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
                this.sensor.AllFramesReady += new
EventHandler<AllFramesReadyEventArgs>(this.sensor_AllFramesReady);

                // Start the sensor!
                try
                {
                    this.sensor.Start();
                }
                catch (IOException)
                {
                    this.sensor = null;
                }
            }
```

```
            if (null == this.sensor)
            {
                this.statusBarText.Text =
Properties.Resources.NoKinectReady;
            }

            //timmer segundos
            System.Windows.Threading.DispatcherTimer dispatcherTimer =
new System.Windows.Threading.DispatcherTimer();
            dispatcherTimer.Tick += new
EventHandler(dispatcherTimer_Tick);
            dispatcherTimer.Interval = new TimeSpan(0, 0, 1);
            dispatcherTimer.Start();

            //timmer minutos
            System.Windows.Threading.DispatcherTimer dispatcherTimer_min
= new System.Windows.Threading.DispatcherTimer();
            dispatcherTimer_min.Tick += new
EventHandler(dispatcherTimer_Tick_min);
            dispatcherTimer_min.Interval = new TimeSpan(0, 1, 0);
            dispatcherTimer_min.Start();


System.Windows.Application.Current.Properties["angulo_postura_correcta_br
azos"] = 90;

System.Windows.Application.Current.Properties["angulo_postura_correcta_pi
ernas"] = 90;

System.Windows.Application.Current.Properties["angulo_postura_correcta_es
palda"] = 180;

System.Windows.Application.Current.Properties["angulo_postura_correcta_ho
mbros"] = 120; //hombros relajados

            //contadores de tiempo

System.Windows.Application.Current.Properties["timer_brazo_derecho"] = 0;

System.Windows.Application.Current.Properties["timer_brazo_izquierdo"] =
0;

System.Windows.Application.Current.Properties["timer_espalda"] = 0;

System.Windows.Application.Current.Properties["timer_hombros"] = 0;
            System.Windows.Application.Current.Properties["timer_tronco"]
= 0;

System.Windows.Application.Current.Properties["timer_pierna_derecha"] =
0;

System.Windows.Application.Current.Properties["timer_pierna_izquierda"] =
0;

            //banderas de los contadores de tiempo

System.Windows.Application.Current.Properties["bd_timer_brazo_derecho"] =
0;

System.Windows.Application.Current.Properties["bd_timer_brazo_izquierdo"]
= 0;
```

```
System.Windows.Application.Current.Properties["bd_timer_espalda"] = 0;

System.Windows.Application.Current.Properties["bd_timer_hombros"] = 0;

System.Windows.Application.Current.Properties["bd_timer_tronco"] = 0;

System.Windows.Application.Current.Properties["bd_timer_pierna_derecha"]
= 0;

System.Windows.Application.Current.Properties["bd_timer_pierna_izquierda"
] = 0;

            //archivo excel

            Microsoft.Office.Interop.Excel.Application app = new
Microsoft.Office.Interop.Excel.Application();
            app.Visible = true;
            app.WindowState = XlWindowState.xlMaximized;

            //hoja de información del paciente
            this.wb = app.Workbooks.Add(XlWBATemplate.xlWBATWorksheet);
            this.ws = this.wb.Worksheets[1];
            this.ws.Name = "Información";

            var dialog = new Inicio();
            if (dialog.ShowDialog() == true) {
                Tuple<string, string, string, string, string> result =
dialog.ResponseText;
                this.ws.Range["A1"].Value = "Nombre";
                this.ws.Range["B1"].Value = result.Item1+"
"+result.Item2;
                this.nombre_usuario = result.Item1 + " " + result.Item2;

                this.ws.Range["A2"].Value = "Sexo";
                this.ws.Range["B2"].Value = result.Item3;

                this.ws.Range["A3"].Value = "Edad";
                this.ws.Range["B3"].Value = result.Item4;

                this.ws.Range["A4"].Value = "Estatura";
                this.ws.Range["B4"].Value = result.Item5;


                //this.ws.Range["A5"].Value = "Detalle";
                //this.ws.Range["B5"].Value = "Tiempo";
                //this.ws.Range["C5"].Value = "Ángulo Cálculado";
                //this.ws.Range["D5"].Value = "Rangos recomendados";
                //this.ws.Range["E5"].Value = "Fecha y hora";

                this.bd_inicio = true;
            }

            nueva_hoja_excel();

        }

        //evento timer segundos
        private void dispatcherTimer_Tick(object sender, EventArgs e)
        {
            if (bd_inicio) {
```

```
                // code goes here
                string aux_tc;
                string aux_bd_t;

                string[] list = new string[] { "brazo_derecho",
"brazo_izquierdo", "espalda", "hombros", "tronco", "pierna_derecha",
"pierna_izquierda" };

                foreach (string item in list)
                {
                    Console.WriteLine( item);
                    aux_bd_t = App.Current.Properties["bd_timer_" +
item].ToString();
                    aux_tc = App.Current.Properties["timer_" +
item].ToString();
                    if (int.Parse(aux_bd_t) == 1)
                    {

System.Windows.Application.Current.Properties["timer_" + item] =
int.Parse(aux_tc) + 1;
                    }
                    //else
                    //{
                        //cambio de una postura incorrecta a correcta
                        //if ((int.Parse(aux_tc) + 1) > 1 * 10)
                        //{
                            //guardar la hoja de excel
                            //DateTime currentDate1 = DateTime.Now;
                            //this.contador_excel += 1;
                            //this.ws.Range["A" + (5 +
contador_excel)].Value = "Postura Incorrecta en: " + item;
                            //this.ws.Range["B" + (5 +
contador_excel)].Value = aux_tc;
                            //this.ws.Range["C" + (5 +
contador_excel)].Value = angulo;
                            //this.ws.Range["D" + (5 +
contador_excel)].Value = currentDate1;

                        //}

//System.Windows.Application.Current.Properties["timer_" + item] = 0;
                    //}
                }
            }

        }

        //evento timer minutos
        private void dispatcherTimer_Tick_min(object sender, EventArgs e)
        {
            DateTime currentDate = DateTime.Now;
            //if ((currentDate.Minute % 5) == 0)

            if (currentDate.Minute == 0) {
                //nueva hoja
                nueva_hoja_excel();
            }

            if (bd_inicio)
            {
                // code goes here
```

```
                string aux_tc;
                string aux_bd_t;

                string[] list = new string[] { "brazo_derecho",
"brazo_izquierdo", "espalda", "hombros", "tronco", "pierna_derecha",
"pierna_izquierda" };

                foreach (string item in list)
                {
                    Console.WriteLine(item);
                    aux_bd_t = App.Current.Properties["bd_timer_" +
item].ToString();
                    aux_tc = App.Current.Properties["timer_" +
item].ToString();
                    if (int.Parse(aux_tc) > 45)
                    {
                        Double angulo = 0;
                        string rangos_angulos = "";

                        //guardar la hoja de excel
                        DateTime currentDate1 = DateTime.Now;
                        this.contador_excel += 1;
                        this.ws.Range["A" + (currentDate1.Minute +
3)].Value = currentDate1.Minute;

                        switch (item)
                        {
                            case "brazo_derecho":
                                label10.Content = "INCORRECTO";
                                label10.Background = Brushes.Red;
                                angulo = this.angulo_brazo_derecho;
                                rangos_angulos = "75-105";
                                this.contador_brazo_derecho += 1;
                                this.ws.Range["F" + (currentDate1.Minute
+ 3)].Value = angulo;
                                this.ws.Range["G" + (currentDate1.Minute
+ 3)].Value = "Incorrecto";
                                break;
                            case "brazo_izquierdo":
                                label11.Content = "INCORRECTO";
                                label11.Background = Brushes.Red;
                                angulo = this.angulo_brazo_izquierdo;
                                rangos_angulos = "75-105";
                                this.contador_brazo_izquierdo += 1;
                                this.ws.Range["D" + (currentDate1.Minute
+ 3)].Value = angulo;
                                this.ws.Range["E" + (currentDate1.Minute
+ 3)].Value = "Incorrecto";
                                break;
                            case "espalda":
                                label12.Content = "INCORRECTO";
                                label12.Background = Brushes.Red;
                                angulo = this.angulo_espalda;
                                rangos_angulos = "165-195";
                                this.contador_espalda +=1;
                                this.ws.Range["L" + (currentDate1.Minute
+ 3)].Value = angulo;
                                this.ws.Range["M" + (currentDate1.Minute
+ 3)].Value = "Incorrecto";
                                break;
                            case "hombros":
```

```
                                label17.Content = "INCORRECTO";
                                label17.Background = Brushes.Red;
                                angulo = this.angulo_hombros;
                                rangos_angulos = "105-135";
                                this.contador_hombros += 1;
                                this.ws.Range["B" + (currentDate1.Minute
+ 3)].Value = angulo;
                                this.ws.Range["C" + (currentDate1.Minute
+ 3)].Value = "Incorrecto";
                                break;
                            case "tronco":
                                label22.Content = "INCORRECTO";
                                label22.Background = Brushes.Red;
                                angulo = this.angulo_espalda_rodilla;
                                rangos_angulos = "105-135";
                                this.contador_espalda_rodilla += 1;
                                this.ws.Range["N" + (currentDate1.Minute
+ 3)].Value = angulo;
                                this.ws.Range["O" + (currentDate1.Minute
+ 3)].Value = "Incorrecto";
                                break;
                            case "pierna_derecha":
                                label13.Content = "INCORRECTO";
                                label13.Background = Brushes.Red;
                                angulo = this.angulo_pierna_derecha;
                                rangos_angulos = "75-105";
                                this.contador_pierna_derecha += 1;
                                this.ws.Range["J" + (currentDate1.Minute
+ 3)].Value = angulo;
                                this.ws.Range["K" + (currentDate1.Minute
+ 3)].Value = "Incorrecto";
                                break;
                            case "pierna_izquierda":
                                label14.Content = "INCORRECTO";
                                label14.Background = Brushes.Red;
                                angulo = this.angulo_pierna_izquierda;
                                rangos_angulos = "75-105";
                                this.contador_pierna_izquierda += 1;
                                this.ws.Range["H" + (currentDate1.Minute
+ 3)].Value = angulo;
                                this.ws.Range["I" + (currentDate1.Minute
+ 3)].Value = "Incorrecto";
                                break;
                            default:
                                //what you want when nothing is selected
                                break;
                        }
                        //MessageBox.Show("Postura Incorrecta en: " +
item);
                    }
                    else
                    {
                        switch (item)
                        {
                            case "brazo_derecho":
                                label10.Content = "CORRECTO";
                                label10.Background = Brushes.Green;
                                break;
                            case "brazo_izquierdo":
                                label11.Content = "CORRECTO";
                                label11.Background = Brushes.Green;
```

```
                                break;
                            case "espalda":
                                label12.Content = "CORRECTO";
                                label12.Background = Brushes.Green;
                                break;
                            case "hombros":
                                label17.Content = "CORRECTO";
                                label17.Background = Brushes.Green;
                                break;
                            case "tronco":
                                label22.Content = "CORRECTO";
                                label22.Background = Brushes.Green;
                                break;
                            case "pierna_derecha":
                                label13.Content = "CORRECTO";
                                label13.Background = Brushes.Green;
                                break;
                            case "pierna_izquierda":
                                label14.Content = "CORRECTO";
                                label14.Background = Brushes.Green;
                                break;
                            default:
                                //what you want when nothing is selected
                                break;
                        }
                    }

System.Windows.Application.Current.Properties["timer_" + item] = 0;
                }
            }
        }

        private void gen_grafica() {
            //guardar los valores
            this.ws.Range["A70"].Value = "Hombros";
            this.ws.Range["A71"].Value = "Brazo Izquierdo";
            this.ws.Range["A72"].Value = "Brazo Derecho";
            this.ws.Range["A73"].Value = "Pierna Izquierda";
            this.ws.Range["A74"].Value = "Pierna Derecha";
            this.ws.Range["A75"].Value = "Espalda";
            this.ws.Range["A76"].Value = "Tronco";

            this.ws.Range["B70"].Value = this.contador_hombros;
            this.ws.Range["B71"].Value = this.contador_brazo_izquierdo;
            this.ws.Range["B72"].Value = this.contador_brazo_derecho;
            this.ws.Range["B73"].Value = this.contador_pierna_izquierda;
            this.ws.Range["B74"].Value = this.contador_pierna_derecha;
            this.ws.Range["B75"].Value = this.contador espalda;
            this.ws.Range["B76"].Value = this.contador_espalda_rodilla;

            //generar grafica resumen de posturas
            var charts = this.ws.ChartObjects() as
Microsoft.Office.Interop.Excel.ChartObjects;
            var chartObject = charts.Add(60, 10, 300, 300) as
Microsoft.Office.Interop.Excel.ChartObject;
            var chart = chartObject.Chart;

            // Set chart range.
            var range = this.ws.get_Range("A70", "B76");
            chart.SetSourceData(range);
```

```
            // Set chart properties.
            chart.ChartType =
Microsoft.Office.Interop.Excel.XlChartType.xlColumnStacked;
            chart.ChartWizard(Source: range,
                Title: "Resumen",
                CategoryTitle: "Articulaciones",
                ValueTitle: "# Incorrectos");

            //reiniciar contadores posturas
            this.contador_brazo_derecho = 0;
            this.contador_brazo_izquierdo = 0;
            this.contador_hombros = 0;
            this.contador_espalda = 0;
            this.contador_espalda_rodilla = 0;
            this.contador_pierna_derecha = 0;
            this.contador_pierna_izquierda = 0;
        }

        private void nueva_hoja_excel() {

            //generar la grafica
            if (this.bd_generar_grafica)
            {
                gen_grafica();
            }
            else {
                this.bd_generar_grafica = true;
            }


            //nueva hoja
            DateTime currentDate = DateTime.Now;
            //this.contador_hoja_excel += 1;
            //MessageBox.Show("Postura Incorrecta en: " +
contador_hoja_excel);
            this.wb.Worksheets.Add();
            this.ws = this.wb.Worksheets[1];

            this.ws.Name = "Hora " + currentDate.Hour + " - " +
(currentDate.Hour + 1);
            this.ws.Activate();

            this.ws.Range["B1"].Value = "Hombros";
            this.ws.Range["D1"].Value = "Brazo Izquierdo";
            this.ws.Range["F1"].Value = "Brazo Derecho";
            this.ws.Range["H1"].Value = "Pierna Izquierda";
            this.ws.Range["J1"].Value = "Pierna Derecha";
            this.ws.Range["L1"].Value = "Espalda";
            this.ws.Range["N1"].Value = "Tronco";

            this.ws.Range["A2"].Value = "Minuto";
            this.ws.Range["B2"].Value = "Ángulo Medido";
            this.ws.Range["C2"].Value = "Postura";
            this.ws.Range["D2"].Value = "Ángulo Medido";
            this.ws.Range["E2"].Value = "Postura";
            this.ws.Range["F2"].Value = "Ángulo Medido";
            this.ws.Range["G2"].Value = "Postura";
            this.ws.Range["H2"].Value = "Ángulo Medido";
            this.ws.Range["I2"].Value = "Postura";
            this.ws.Range["J2"].Value = "Ángulo Medido";
```

```
            this.ws.Range["K2"].Value = "Postura";
            this.ws.Range["L2"].Value = "Ángulo Medido";
            this.ws.Range["M2"].Value = "Postura";
            this.ws.Range["N2"].Value = "Ángulo Medido";
            this.ws.Range["O2"].Value = "Postura";

        }

        /// <summary>
        /// Execute shutdown tasks
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void WindowClosing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            if (null != this.sensor)
            {
                this.sensor.Stop();
            }

            //guardar el reporte
            DateTime currentDate = DateTime.Now;
            String nombre = "Reporte " + this.nombre_usuario + " " +
currentDate.Date + "/" + currentDate.Month + "/" + currentDate.Year;
            nombre = nombre.Replace('/', '_');
            nombre = nombre.Replace(':', '_');
            gen_grafica();
            this.wb.SaveAs("C:\\Users\\PERSONAL\\Documents\\Reporte\\"
+nombre + ".xlsx");

                }

        /// <summary>
        /// Event handler for Kinect sensor's SkeletonFrameReady event
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        ///
        void sensor_AllFramesReady(object sender, AllFramesReadyEventArgs
e)
        {
            sensor_ColorFrameReady(e);
            sensor_DepthFrameReady(e);
        }

        void sensor_DepthFrameReady(AllFramesReadyEventArgs e)
        {
            using (DepthImageFrame depthFrame = e.OpenDepthImageFrame()){
                if (depthFrame == null)
                {
                    return;
                }
                else {
                    //grafica
                    depthimage.Source = DepthToBitmapSource(depthFrame);
                }

            }
        }
        void sensor_ColorFrameReady(AllFramesReadyEventArgs e)
```

```
        {
            using (ColorImageFrame colorFrame = e.OpenColorImageFrame())
{
                if (colorFrame == null)
                {
                    return;
                }
                else {
                    //grafica
                    colorimage.Source = ColorToBitmapSource(colorFrame);
                }
            }

        }

        BitmapSource DepthToBitmapSource(DepthImageFrame imageFrame){
            short[] pixelData = new short[imageFrame.PixelDataLength];
            imageFrame.CopyPixelDataTo(pixelData);

            BitmapSource bmap = BitmapSource.Create(
             imageFrame.Width,
             imageFrame.Height,
             96, 96,
             PixelFormats.Gray16,
             null,
             pixelData,
             imageFrame.Width * imageFrame.BytesPerPixel);
            return bmap;
        }

        BitmapSource ColorToBitmapSource(ColorImageFrame imageFrame){
            byte[] pixelData = new byte[imageFrame.PixelDataLength];
            imageFrame.CopyPixelDataTo(pixelData);

            BitmapSource bmap = BitmapSource.Create(
             imageFrame.Width,
             imageFrame.Height,
             96, 96,
             PixelFormats.Bgr32,
             null,
             pixelData,
             imageFrame.Width * imageFrame.BytesPerPixel);
            return bmap;
        }

        private void SensorSkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
        {
            Skeleton[] skeletons = new Skeleton[0];

            using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
            {
                if (skeletonFrame != null)
                {
                    skeletons = new
Skeleton[skeletonFrame.SkeletonArrayLength];
                    skeletonFrame.CopySkeletonDataTo(skeletons);
                }
            }

            using (DrawingContext dc = this.drawingGroup.Open())
```

```
            {
                // Draw a transparent background to set the render size
                dc.DrawRectangle(Brushes.Black, null, new Rect(0.0, 0.0,
RenderWidth, RenderHeight));

                if (skeletons.Length != 0)
                {
                    foreach (Skeleton skel in skeletons)
                    {
                        RenderClippedEdges(skel, dc);

                        if (skel.TrackingState ==
SkeletonTrackingState.Tracked)
                        {
                            this.DrawBonesAndJoints(skel, dc);

                            // angulo del brazo derecho
                            Joint codo_derecho =
skel.Joints[JointType.ElbowRight];
                            Joint hombro_derecho =
skel.Joints[JointType.ShoulderRight];
                            Joint muñeca_derecho =
skel.Joints[JointType.WristRight];
                            if (codo_derecho.TrackingState ==
JointTrackingState.Tracked && hombro_derecho.TrackingState ==
JointTrackingState.Tracked && muñeca_derecho.TrackingState ==
JointTrackingState.Tracked)
                            {
                                Vector3D vector_codo_muñeca = new
Vector3D(muñeca_derecho.Position.X - codo_derecho.Position.X,
muñeca_derecho.Position.Y - codo_derecho.Position.Y,
muñeca_derecho.Position.Z - codo_derecho.Position.Z);
                                Vector3D vector_codo_hombro = new
Vector3D(hombro_derecho.Position.X - codo_derecho.Position.X,
hombro_derecho.Position.Y - codo_derecho.Position.Y,
hombro_derecho.Position.Z - codo_derecho.Position.Z);
                                this.angulo_brazo_derecho =
Vector3D.AngleBetween(vector_codo_hombro, vector_codo_muñeca);
                                label.Content = Math.Truncate(10 *
angulo_brazo_derecho) / 10;
                                if (angulo_brazo_derecho > (90-15) &&
angulo_brazo_derecho < (90+15))
                                {
                                    //label10.Content = "CORRECTO";
                                    //label10.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_brazo_derecho"] =
0;
                                }
                                else
                                {

System.Windows.Application.Current.Properties["bd_timer_brazo_derecho"] =
1;
                                }
                                //Console.WriteLine("Left knee: " +
j.Position.X + ", " + j.Position.Y + ", " + j.Position.Z);
                            }
                            else {
                                //label10.Content = "CORRECTO";
                                //label10.Background = Brushes.Green;
```

```
System.Windows.Application.Current.Properties["bd_timer_brazo_derecho"] =
0;
                            }

                            // angulo del brazo izquierdo
                            Joint codo_izquierdo =
skel.Joints[JointType.ElbowLeft];
                            Joint hombro_izquierdo =
skel.Joints[JointType.ShoulderLeft];
                            Joint muñeca_izquierdo =
skel.Joints[JointType.WristLeft];
                            if (codo_izquierdo.TrackingState ==
JointTrackingState.Tracked && hombro_izquierdo.TrackingState ==
JointTrackingState.Tracked && muñeca_izquierdo.TrackingState ==
JointTrackingState.Tracked)
                            {
                                Vector3D vector_codo_muñeca = new
Vector3D(muñeca_izquierdo.Position.X - codo_izquierdo.Position.X,
muñeca_izquierdo.Position.Y - codo_izquierdo.Position.Y,
muñeca_izquierdo.Position.Z - codo_izquierdo.Position.Z);
                                Vector3D vector_codo_hombro = new
Vector3D(hombro_izquierdo.Position.X - codo_izquierdo.Position.X,
hombro_izquierdo.Position.Y - codo_izquierdo.Position.Y,
hombro_izquierdo.Position.Z - codo_izquierdo.Position.Z);
                                this.angulo_brazo_izquierdo =
Vector3D.AngleBetween(vector_codo_hombro, vector_codo_muñeca);
                                label3.Content = Math.Truncate(10 *
angulo_brazo_izquierdo) / 10;
                                if (angulo_brazo_izquierdo > (90-15) &&
angulo_brazo_izquierdo < (90+15))
                                {
                                    //label11.Content = "CORRECTO";
                                    //label11.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_brazo_izquierdo"]
= 0;
                                }
                                else
                                {

System.Windows.Application.Current.Properties["bd_timer_brazo_izquierdo"]
= 1;
                                }
                            }
                            else {
                                //label11.Content = "CORRECTO";
                                //label11.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_brazo_izquierdo"]
= 0;
                            }


                            // angulo espina, hombro, cuello espalda
                            Joint espalda =
skel.Joints[JointType.ShoulderCenter];
                            Joint espina = skel.Joints[JointType.Spine];
                            Joint cabeza = skel.Joints[JointType.Head];
                            if (espalda.TrackingState ==
JointTrackingState.Tracked && espina.TrackingState ==
```

```
JointTrackingState.Tracked && cabeza.TrackingState ==
JointTrackingState.Tracked)
                            {
                                Vector3D vector_cabeza_espalda = new
Vector3D(cabeza.Position.X - espalda.Position.X, cabeza.Position.Y -
espalda.Position.Y, cabeza.Position.Z - espalda.Position.Z);
                                Vector3D vector_espina_espalda = new
Vector3D(espina.Position.X - espalda.Position.X, espina.Position.Y -
espalda.Position.Y, espina.Position.Z - espalda.Position.Z);
                                this.angulo_espalda =
Vector3D.AngleBetween(vector_cabeza_espalda, vector_espina_espalda);
                                label5.Content = Math.Truncate(10 *
angulo_espalda) / 10;
                                if (angulo_espalda > (180-15) &&
angulo_espalda < (180+15))
                                {
                                    //label12.Content = "CORRECTO";
                                    //label12.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_espalda"] = 0;
                                }
                                else
                                {

System.Windows.Application.Current.Properties["bd_timer_espalda"] = 1;
                                }

                            }
                            else {
                                //label12.Content = "CORRECTO";
                                //label12.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_espalda"] = 0;
                            }



                            // angulo hombros
                            //Joint espalda =
skel.Joints[JointType.ShoulderCenter];
                            //Joint hombro_izquierdo =
skel.Joints[JointType.ShoulderLeft];
                            //Joint hombro_derecho =
skel.Joints[JointType.ShoulderRight];
                            if (espalda.TrackingState ==
JointTrackingState.Tracked && hombro_izquierdo.TrackingState ==
JointTrackingState.Tracked && hombro_derecho.TrackingState ==
JointTrackingState.Tracked)
                            {
                                Vector3D vector_espalda_hombro_izquierdo
= new Vector3D(hombro_izquierdo.Position.X - espalda.Position.X,
hombro_izquierdo.Position.Y - espalda.Position.Y,
hombro_izquierdo.Position.Z - espalda.Position.Z);
                                Vector3D vector_espalda_hombro_derecho =
new Vector3D(hombro_derecho.Position.X - espalda.Position.X,
hombro_derecho.Position.Y - espalda.Position.Y, hombro_derecho.Position.Z
- espalda.Position.Z);
                                this.angulo_hombros =
Vector3D.AngleBetween(vector_espalda_hombro_derecho,
vector_espalda_hombro_izquierdo);
```

```
                                label16.Content = Math.Truncate(10 *
angulo_hombros) / 10;
                                if (angulo_hombros > (120-15) &&
angulo_hombros < (120+15))
                                {
                                    //label17.Content = "CORRECTO";
                                    //label17.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_hombros"] = 0;
                                }
                                else
                                {

System.Windows.Application.Current.Properties["bd_timer_hombros"] = 1;
                                }
                            }
                            else {
                                //label17.Content = "CORRECTO";
                                //label17.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_hombros"] = 0;
                            }


                           // angulo pierna derecha
                           Joint rodilla_derecha =
skel.Joints[JointType.KneeRight];
                            Joint cadera_derecha  =
skel.Joints[JointType.HipRight];
                            Joint tobillo_derecha  =
skel.Joints[JointType.AnkleRight];
                            if (rodilla_derecha.TrackingState ==
JointTrackingState.Tracked && cadera_derecha.TrackingState ==
JointTrackingState.Tracked && tobillo_derecha.TrackingState ==
JointTrackingState.Tracked)
                            {
                                Vector3D vector_rodilla_cadera = new
Vector3D(cadera_derecha.Position.X - rodilla_derecha.Position.X,
cadera_derecha.Position.Y - rodilla_derecha.Position.Y,
cadera_derecha.Position.Z - rodilla_derecha.Position.Z);
                                Vector3D vector_rodilla_tobillo = new
Vector3D(tobillo_derecha.Position.X - rodilla_derecha.Position.X,
tobillo_derecha.Position.Y - rodilla_derecha.Position.Y,
tobillo_derecha.Position.Z - rodilla_derecha.Position.Z);
                                this.angulo_pierna_derecha =
Vector3D.AngleBetween(vector_rodilla_cadera, vector_rodilla_tobillo);
                                this.angulo_pierna_derecha =
(this.angulo_pierna_derecha + 90) / 2;
                                label8.Content = Math.Truncate(10 *
angulo_pierna_derecha) / 10;
                                if (angulo_pierna_derecha > (90-15) &&
angulo_pierna_derecha < (90+15))
                                {
                                    //label13.Content = "CORRECTO";
                                    //label13.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_pierna_derecha"]
= 0;
                                }
                                else
                                {
```

```
System.Windows.Application.Current.Properties["bd_timer_pierna_derecha"]
= 1;
                                }
                            }
                            else {
                                //label13.Content = "CORRECTO";
                                //label13.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_pierna_derecha"]
= 0;
                            }

                            // angulo pierna izquierda
                            Joint rodilla_izquierda =
skel.Joints[JointType.KneeLeft];
                            Joint cadera_izquierda =
skel.Joints[JointType.HipLeft];
                            Joint tobillo_izquierda =
skel.Joints[JointType.AnkleLeft];
                            if (rodilla_izquierda.TrackingState ==
JointTrackingState.Tracked && cadera_izquierda.TrackingState ==
JointTrackingState.Tracked && tobillo_izquierda.TrackingState ==
JointTrackingState.Tracked)
                            {
                                Vector3D vector_rodilla_cadera = new
Vector3D(cadera_izquierda.Position.X - rodilla_izquierda.Position.X,
cadera_izquierda.Position.Y - rodilla_izquierda.Position.Y,
cadera_izquierda.Position.Z - rodilla_izquierda.Position.Z);
                                Vector3D vector_rodilla_tobillo = new
Vector3D(tobillo_izquierda.Position.X - rodilla_izquierda.Position.X,
tobillo_izquierda.Position.Y - rodilla_izquierda.Position.Y,
tobillo_izquierda.Position.Z - rodilla_izquierda.Position.Z);
                                this.angulo_pierna_izquierda =
Vector3D.AngleBetween(vector_rodilla_cadera, vector_rodilla_tobillo);
                                this.angulo_pierna_izquierda =
(this.angulo_pierna_izquierda + 90) / 2;
                                label9.Content = Math.Truncate(10 *
angulo_pierna_izquierda) / 10;
                                if (angulo_pierna_izquierda > (90-15) &&
angulo_pierna_izquierda < (90+15))
                                {
                                    //label14.Content = "CORRECTO";
                                    //label14.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_pierna_izquierda"
] = 0;
                                }
                                else
                                {

System.Windows.Application.Current.Properties["bd_timer_pierna_izquierda"
] = 1;
                                }
                            }
                            else {
                                //label14.Content = "CORRECTO";
                                //label14.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_pierna_izquierda"
] = 0;
```

```
                            }



                            // angulo tronco
                            //espalda central, cadera derecha, rodilla
derecha
                            if (rodilla_derecha.TrackingState ==
JointTrackingState.Tracked && espalda.TrackingState ==
JointTrackingState.Tracked && cadera_derecha.TrackingState ==
JointTrackingState.Tracked)
                            {
                                Vector3D vector_espalda_cadera = new
Vector3D(espalda.Position.X - cadera_derecha.Position.X,
espalda.Position.Y - cadera_derecha.Position.Y, espalda.Position.Z -
cadera_derecha.Position.Z);
                                Vector3D vector_rodilla_cadera = new
Vector3D(rodilla_derecha.Position.X - cadera_derecha.Position.X,
rodilla_derecha.Position.Y - cadera_derecha.Position.Y,
rodilla_derecha.Position.Z - cadera_derecha.Position.Z);
                                this.angulo_espalda_rodilla =
Vector3D.AngleBetween(vector_espalda_cadera, vector_rodilla_cadera);
                                this.angulo_espalda_rodilla=
(this.angulo_espalda_rodilla+115)/ 2;
                                label21.Content = Math.Truncate(10 *
angulo_espalda_rodilla) / 10;
                                if (angulo_espalda_rodilla > (115 - 15)
&& angulo_espalda_rodilla < (115 + 15))
                                {
                                    //label22.Content = "CORRECTO";
                                    //label22.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_tronco"] = 0;
                                }
                                else
                                {

System.Windows.Application.Current.Properties["bd_timer_tronco"] = 1;
                                }
                            }
                            else {
                                //label22.Content = "CORRECTO";
                                //label22.Background = Brushes.Green;

System.Windows.Application.Current.Properties["bd_timer_tronco"] = 0;
                            }




                        }
                        else if (skel.TrackingState ==
SkeletonTrackingState.PositionOnly)
                        {
                            dc.DrawEllipse(
                            this.centerPointBrush,
                            null,
                            this.SkeletonPointToScreen(skel.Position),
                            BodyCenterThickness,
                            BodyCenterThickness);
                        }
```

```
                    }
                }

                // prevent drawing outside of our render area
                this.drawingGroup.ClipGeometry = new
RectangleGeometry(new Rect(0.0, 0.0, RenderWidth, RenderHeight));
            }
        }

        /// <summary>
        /// Draws a skeleton's bones and joints
        /// </summary>
        /// <param name="skeleton">skeleton to draw</param>
        /// <param name="drawingContext">drawing context to draw
to</param>
        private void DrawBonesAndJoints(Skeleton skeleton, DrawingContext
drawingContext)
        {
            // Render Torso
            this.DrawBone(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
            this.DrawBone(skeleton, drawingContext,
JointType.ShoulderCenter, JointType.ShoulderLeft);
            this.DrawBone(skeleton, drawingContext,
JointType.ShoulderCenter, JointType.ShoulderRight);
            this.DrawBone(skeleton, drawingContext,
JointType.ShoulderCenter, JointType.Spine);
            this.DrawBone(skeleton, drawingContext, JointType.Spine,
JointType.HipCenter);
            this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipLeft);
            this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipRight);

            // Left Arm
            this.DrawBone(skeleton, drawingContext,
JointType.ShoulderLeft, JointType.ElbowLeft);
            this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
            this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);

            // Right Arm
            this.DrawBone(skeleton, drawingContext,
JointType.ShoulderRight, JointType.ElbowRight);
            this.DrawBone(skeleton, drawingContext, JointType.ElbowRight,
JointType.WristRight);
            this.DrawBone(skeleton, drawingContext, JointType.WristRight,
JointType.HandRight);

            // Left Leg
            this.DrawBone(skeleton, drawingContext, JointType.HipLeft,
JointType.KneeLeft);
            this.DrawBone(skeleton, drawingContext, JointType.KneeLeft,
JointType.AnkleLeft);
            this.DrawBone(skeleton, drawingContext, JointType.AnkleLeft,
JointType.FootLeft);

            // Right Leg
            this.DrawBone(skeleton, drawingContext, JointType.HipRight,
JointType.KneeRight);
```

```
            this.DrawBone(skeleton, drawingContext, JointType.KneeRight,
JointType.AnkleRight);
            this.DrawBone(skeleton, drawingContext, JointType.AnkleRight,
JointType.FootRight);

            // Render Joints
            foreach (Joint joint in skeleton.Joints)
            {
                Brush drawBrush = null;

                if (joint.TrackingState == JointTrackingState.Tracked)
                {
                    drawBrush = this.trackedJointBrush;
                }
                else if (joint.TrackingState ==
JointTrackingState.Inferred)
                {
                    drawBrush = this.inferredJointBrush;
                }

                if (drawBrush != null)
                {
                    drawingContext.DrawEllipse(drawBrush, null,
this.SkeletonPointToScreen(joint.Position), JointThickness,
JointThickness);
                }
            }
        }

        /// <summary>
        /// Maps a SkeletonPoint to lie within our render space and
converts to Point
        /// </summary>
        /// <param name="skelpoint">point to map</param>
        /// <returns>mapped point</returns>
        private System.Windows.Point SkeletonPointToScreen(SkeletonPoint
skelpoint)
        {
            // Convert point to depth space.
            // We are not using depth directly, but we do want the points
in our 640x480 output resolution.
            DepthImagePoint depthPoint =
this.sensor.CoordinateMapper.MapSkeletonPointToDepthPoint(skelpoint,
DepthImageFormat.Resolution640x480Fps30);
            return new System.Windows.Point(depthPoint.X, depthPoint.Y);
        }

        /// <summary>
        /// Draws a bone line between two joints
        /// </summary>
        /// <param name="skeleton">skeleton to draw bones from</param>
        /// <param name="drawingContext">drawing context to draw
to</param>
        /// <param name="jointType0">joint to start drawing from</param>
        /// <param name="jointType1">joint to end drawing at</param>
        private void DrawBone(Skeleton skeleton, DrawingContext
drawingContext, JointType jointType0, JointType jointType1)
        {
            Joint joint0 = skeleton.Joints[jointType0];
            Joint joint1 = skeleton.Joints[jointType1];
```

```
            // If we can't find either of these joints, exit
            if (joint0.TrackingState == JointTrackingState.NotTracked ||
                joint1.TrackingState == JointTrackingState.NotTracked)
            {
                return;
            }

            // Don't draw if both points are inferred
            if (joint0.TrackingState == JointTrackingState.Inferred &&
                joint1.TrackingState == JointTrackingState.Inferred)
            {
                return;
            }

            // We assume all drawn bones are inferred unless BOTH joints
are tracked
            Pen drawPen = this.inferredBonePen;
            if (joint0.TrackingState == JointTrackingState.Tracked &&
joint1.TrackingState == JointTrackingState.Tracked)
            {
                drawPen = this.trackedBonePen;
            }

            drawingContext.DrawLine(drawPen,
this.SkeletonPointToScreen(joint0.Position),
this.SkeletonPointToScreen(joint1.Position));
        }

        /// <summary>
        /// Handles the checking or unchecking of the seated mode combo
box
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void CheckBoxSeatedModeChanged(object sender,
RoutedEventArgs e)
        {
            if (null != this.sensor)
            {
                if
(this.checkBoxSeatedMode.IsChecked.GetValueOrDefault())
                {
                    this.sensor.SkeletonStream.TrackingMode =
SkeletonTrackingMode.Seated;
                }
                else
                {
                    this.sensor.SkeletonStream.TrackingMode =
SkeletonTrackingMode.Default;
                }
            }
        }
    }
     }
```