INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELECTRICA

UNIDAD PROFESIONAL “ADOLFO LÓPEZ MATEOS”

“IMPLEMENTACIÓN DE UN CONTROL GESTUAL EN MÓVIL INALÁMBRICO”

T E S I S

QUEPARA OBTENER EL TITULO DE:

INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTAN:

JORGE ALBERTO CORTES RODRIGUEZ

FRANCISCO ANTONIO SANDOVAL

IRVING RICARDO TRUJILLO PASCUAL

ASESORES:

M. EN C. DAVID VAZQUEZ ALVAREZ

ING. JAFETH A. ALONSO CARREON

MÉXICO, D.F. 2014

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELECTRICA

UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

TEMA DE TESIS

QUE PARA OBTENER EL TITULO DE **INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**
POR LA OPCIÓN DE TITULACIÓN **TESIS COLECTIVA Y EXAMEN ORAL INDIVIDUAL**
DEBERÁ (N) DESARROLLAR **C. JORGE ALBERTO CORTES RODRIGUEZ**
C. FRANCISCO ANTONIO SANDOVAL
C. IRVING RICARDO TRUJILLO PASCUAL

"IMPLEMENTACIÓN DE UN CONTROL GESTUAL EN MÓVIL INALÁMBRICO"

DESARROLLAR UN MÓVIL QUE SERÁ CONTROLADO UTILIZANDO UN SENSOR KINECT, PARA QUE EJECUTE LAS TAREAS ENVIADAS MEDIANTE COMUNICACIÓ INALÁMBRICA.

- INTRODUCCIÓN
- JUSTIFICACIÓN
- CONTROL GESTUAL Y SENSORES DE MOVIMIENTO
- SISTEMAS DE CONTROL Y DETECCIÓN
- PLATAFORMAS DE DESARROLLO
- INTEGRACIÓN DE CÓDIGOS PARA LA IMPLEMENTACIÓN DEL CONTROL GESTUAL
- CONCLUSIONES
- ANEXOS
- BIBLIOGRAFÍA

MÉXICO D.F. A 02 DE DICIEMBRE DE 2013

ASESORES

M. EN C. DAVID VAZQUEZ ALVAREZ **ING. JAFETH A. ALONSO CARREON**

ING. PATRICIA LORENA RAMIREZ RANGEL
JEFE DEL DEPARTAMENTO DE
INGENIERÍA EN COMUNICACIONES Y ELECTRONICA

ÍNDICE

Justificación. i
Objetivo. iii
Introducción. iv

CAPITULO 1
Control Gestual y Sensores de movimiento. 1

1.1 El Control Gestual. 1
1.1.1 Desarrollo Futuro. 2
1.2 Historia del Control Gestual. 3
1.2.1 Aplicación en los Ordenadores. 3
1.3 Sensores de Movimiento. 4
1.3.1 Características. 4
1.3.2 Tipos de Sensores. 5
1.3.3 Funcionamiento de los Sensores de Movimiento. 5
1.4 Ciencia de la Computación. 7

CAPITULO 2
Sistemas de Control y Detección. 8

2.1 El Sensor Kinect y su Composición. 8
2.1.1 Funcionamiento. 9
2.2 Detección de Articulaciones Mediante el Sensor Kinect. 11
2.3 Reconocimiento de Gestos. 11

CAPITULO 3
Plataformas de Desarrollo. 13

3.1 Microsoft Visual Studio. 13
3.1.1 Lenguaje de Programación C#. 14
3.1.2 Características del Lenguaje C#. 14
3.2 Microsoft SDK Kinect. 14
3.2.1 Características de la Versión 1.7. 15
3.2.2 ¿Por qué Kinect? 16
3.3 ARDUINO. 16
3.3.1 Bibliotecas en ARDUINO. 17
3.3.2 Microcontroladores para ARDUINO. 18

CAPITULO 4
Integración de códigos para la implementación del Control Gestual. 20

4.1 Reconocimiento de puntos en el cuerpo. 20
4.1.1 Detectar Posturas. 22

4.2 Reconocimiento de Gestos. 23
4.2.1 Control de Velocidad. 26
4.3 Programación de ARDUINO. 27
4.3.1 Conexión con ARDUINO. 28
4.4 El Móvil. 30

CONCLUSIONES. 32

ANEXO A
Instalación de Microsoft SDK Kinect en Visual Studio. 34

ANEXO B
Código en C# para la detección de puntos en el cuerpo humano. 40

ANEXO C
Código para la detección de gestos. 44

ANEXO D
Artículo presentado en Congreso. 54

REFERENCIAS. 59

JUSTIFICACIÓN

Desde el principio de la Revolución Industrial hasta nuestro tiempo, los seres humanos han estado interesados en medios que permitan realizar tareas mediante máquinas, que sean accesibles y tengan un nivel de seguridad y rendimiento óptimo. El objetivo de la tecnología siempre ha sido el ayudar a resolver problemas y también el hacer más cómoda la vida de las personas.

Antes, los controles comunes eran utilizados en todos los equipos; para encendido y apagado, volumen, ejecutar comandos, mover un objeto, etc. Pero tenían la desventaja de que se extraviaban, podían llenarse de grasa, se caían o golpeaban y entonces dejaban de funcionar. Es por eso que surgió una nueva tecnología en donde el control es uno mismo, realizando movimientos con las manos o el cuerpo para ejecutar las funciones requeridas.

Esta tecnología se puede aplicar en varias áreas para usos diferentes. Por lo que se considera una tecnología con futuro y expansión. Permite tener un control preciso sobre tareas complicadas o que requieran un alto grado de estabilidad, también ayuda a mantener nuestro cuerpo en movimiento ayudando a evitar el sedentarismo que es causa de algunas enfermedades que afectan actualmente a las personas.

El control a distancia es una herramienta que tiene muchas aplicaciones en todas las ciencias y trabajos: actualmente existe "cirugía a distancia"; esta es una técnica en la cual un cirujano lleva a cabo una cirugía usando una computadora que controla instrumentos muy pequeños fijados a un robot. El cirujano se sienta en una estación informática cercana o grandes distancias como por ejemplo diferentes estados o países y dirige los movimientos de un robot. Otra aplicación existente, es la de un robot para localizar desperfectos en tuberías de la industria petrolera. Se trata de un dispositivo neumático que se opera a control remoto a través de un software y con una cámara infrarroja que permite ver en detalle el interior de la cañería. También existe el robot desarma bombas, el robot es dirigido en sus desplazamientos, con un controlador de tipo gamepad, un joystick clásico y un sensor de presión controlado por el pie usado en los zapatos. Todas estas aplicaciones, sumándole la nuestra, hacen que esta tecnología sea un paso importante en el desarrollo tecnológico.

Se realizó el proyecto de controlar un móvil a distancia a través de la computadora, utilizando el sensor Kinect. El enfoque va dirigido al control gestual, con el cual se pueden lograr movimientos más precisos.

Este proyecto presenta el prototipo de un vehículo que se controla a través del movimiento del brazo izquierdo. Esto permite dejar las demás extremidades libres para realizar trabajos diferentes o enviar comandos nuevos al vehículo. Utilizando la cámara del sensor Kinect se aprovechó la tecnología existente para darle un uso innovador que pueda contribuir al desarrollo de nuevas aplicaciones.

El propósito de este proyecto es aminorar la diferencia tecnológica entre México y otros países, que generan su propia tecnología y que continuamente renuevan sus herramientas productivas y técnicas. Muchas de las innovaciones que se integran a los sistemas productivos son generadas en las universidades. Por desgracia, en México este panorama es diferente, las empresas tienden a buscar tecnologías formada en el extranjero, por corporaciones que no generan investigación en México, y por lo tanto, esto contribuye a la dependencia tecnológica.

Es por eso que, con el propósito de romper con esta dependencia tecnológica, nosotros como alumnos del Instituto Politécnico Nacional queremos dar apoyo a la investigación. Con ello la ESIME Zacatenco podrá darse a conocer con el desarrollo de este proyecto, el cuál es un robot móvil controlado gestualmente, que puede ser adecuado y configurado para múltiples usos, además de la aportación del sistema de "Control Gestual", que puede definir los movimientos para realizar tareas específicas.

El desarrollo de este proyecto dentro de la ESIME del IPN nos llevaría a estar a la vanguardia y, sobre todo, dentro de la competitividad y el progreso de las técnicas de control por computadora.

OBJETIVO GENERAL

Desarrollar un móvil que será controlado utilizando un sensor Kinect, para que ejecute las tareas enviadas mediante comunicación inalámbrica.

OBJETIVOS ESPECÍFICOS

- Desarrollar un sistema capaz de tener, como única forma de control, la información obtenida por medio del sensor Kinect.

- Lograr que un usuario tele-opere el vehículo para que se desplace por el entorno.

- Documentar el uso de sistemas básicos de Control Gestual, para su posible aprovechamiento como base para otros proyectos, que puedan beneficiarse de éste tipo de sensores.

INTRODUCCIÓN

En tiempos recientes se ha llevado a cabo la automatización de diferentes tipos de procesos como los industriales o los administrativos. Esto trae como beneficio la reducción del tiempo, gracias al avance en los sistemas de cómputo y al surgimiento de nuevas tecnologías de computación inteligente como los sistemas de Control por computadora y la lógica difusa entre otras. Estos nuevos desarrollos han facilitado el trabajo del ser humano haciéndolo más eficiente.

Actualmente la mayoría de los sistemas de control por computadora que se emplean se encuentran instalados en las plantas manufactureras para efectuar tareas de control de la calidad de los productos, esto se puede llevar a cabo mediante algoritmos de conteo y detección de patrones, los cuales son seleccionados por un experto de acuerdo a las necesidades de cada proceso. Para este tipo de actividades es suficiente utilizar una sola cámara o sensor ya que es necesario conocer la ubicación de un objeto en el espacio.

Por otra parte están tareas como en una línea de producción automatizada donde un conjunto de robots elaboran un producto en etapas. Para ello, éstos deben ser previamente programados con rutinas específicas que ejecutarán repetidamente. Sin embargo, sí se deseara construir una celda de manufactura flexible donde los robots tuvieran la capacidad de hacer diversos tipos de ensambles sin la necesidad de tener que reprogramarlos cada vez, se podría realizar incorporando a éstos algunas de las tecnologías de computación inteligente como las mencionadas anteriormente, con el fin de brindar mayor flexibilidad a este tipo de mecanismo.

En la presente tesis describe una metodología para la manipulación a distancia de objetos mediante un robot móvil, controlado mediante movimientos frente a un sensor Kinect. El sensor capta los movimientos realizados por el usuario y los interpreta como comandos para enviarlos al robot, el cual ejecutará la acción correspondiente como moverse o manipular un objeto.

Para encontrar el primer mando a distancia como tal tenemos que remontarnos a 1950. Ese año el fundador de la empresa Zenith Electronics Corporation estaba cansado de tener que levantarse a cambiar de canal y especialmente irritado se quedaba por el tema de los anuncios.

Como era el dueño de su propia empresa, pidió a sus ingenieros que le crearan un dispositivo que hiciera eso que quería sin tener que levantarse del sofá, y de esta peculiar manera nació el Lazy Bones, el cual movía los controles del televisor con ayuda de motores y un cable.

Era el año 1955 se quería tener el mismo objetivo pero sin tener los molestos cables y motores. Para conseguir ese mando, Zenith Eugene Polley ideó un sistema compuesto por cuatro células fotosensibles dispuestas en las esquinas del televisor y que se activaban con una especie de pistola de luz. Había que tener la precisión de un francotirador para no fallar al cambiar de canal o subir el volumen, pero ahí estaba un interesante embrión del mando a distancia más clásico. [1]

Posteriormente aparecieron nuevas tecnologías que revolucionaron la forma en que interactuábamos con las máquinas. Un gran ejemplo son los equipos con pantalla táctil que actualmente son muy comunes, y ahora el control gestual que se encuentra en expansión.

El mando a distancia está siendo utilizado en diversas áreas y aplicaciones que han revolucionado el manejo de los aparatos. Estas aplicaciones están inmersas en:

- ❖ Medicina
  - ✓ Cirugía a distancia

- ❖ Aplicaciones militares
  - ✓ Vehículos no tripulados

- ❖ Seguridad
  - ✓ Robot desarma-bombas
  - ✓ Robot de rescate en zona de derrumbe

- ❖ Ciencia
  - ✓ Vehículos de exploración

- ❖ Comodidad hogareña
  - ✓ Pantallas inteligentes (T.V)
  - ✓ Teléfonos celulares
  - ✓ Videojuegos

CAPITULO 1
CONTROL GESTUAL Y SENSORES DE MOVIMIENTO

Un mando a distancia es un dispositivo electrónico que permite realizar una operación remota sobre una máquina.

La mayoría de los mandos a distancia para aparatos domésticos utilizan diodos de emisión cercana a infrarrojo para emitir un rayo de luz que alcance el dispositivo. Esta luz es invisible para el ojo humano, pero transporta señales que pueden ser detectadas por el aparato.

El control basado en gestos tiene mucho camino por recorrer, Kinect está consiguiendo que lo veamos como algo cotidiano al traerlo al mercado del ocio, gracias también a los desarrollos alternativos que lo intentan exprimir.

1.1 EL CONTROL GESTUAL

En apenas cinco años las pantallas táctiles han redefinido la forma de interactuar con la tecnología, una relación de contacto que deja un hueco al control gestual y cierne una nueva amenaza sobre la supervivencia del control de ratón de sobremesa, una especie ya en peligro de extinción.

El manejo de dispositivos a través de sensores de movimiento, un sistema que funciona con éxito en la industria del videojuego, se convertirá antes de que finalice la década en un nuevo estándar para ordenadores personales, tabletas y teléfonos a los que se pedirá que, además de ser "inteligentes", agudicen sus sentidos.

Una prueba de esta evolución se pudo ver en el año 2013 en la conferencia anual para desarrolladores de software de Microsoft Build, en San Francisco, donde se presentó la primera actualización del sistema operativo Windows 8 (Windows 8.1).

Entre la lista de novedades se desveló, casi de forma anecdótica, la aplicación culinaria "Food and Drink", diseñada para que el usuario pueda seguir paso a paso una receta mientras cocina sin manchar la pantalla de su tableta por tener los dedos embadurnados de harina.

"Food and Drink" permite al aprendiz de chef cambiar de página con un movimiento de la mano frente al dispositivo, algo similar a lo que se puede hacer en la consola Xbox 360 por mediación de Kinect, pero sin que sea necesario comprar periférico alguno.

La aplicación reacciona a los cambios de luz que captan las lentes cuando se pasa algo por delante, una mano por ejemplo. Una tecnología extrapolable a ordenadores y teléfonos. De hecho, el Samsung Galaxy S4 ya ofrece esa posibilidad. [2]

"Food and Drink" es la única herramienta creada, por el momento, por Microsoft con esas características, confirmó Sareen, y no se ha facilitado aún a los desarrolladores de software ajenos a la compañía los códigos para que puedan incorporar ese control gestual, de fácil adaptación, a otro tipo de software.

Esta tecnología es algo en mente de muchas empresas del sector. En noviembre, el fabricante de chips Qualcomm, con sede en San Diego, publicó la guía para hacer posible el control gestual para dispositivos con sistema operativo Android equipados con su último procesador, Snapdragon. Qualcomm hizo incluso una demostración con otra aplicación de cocina y anticipó que pronto esas cámaras serán capaces de reconocer la cara del usuario y seguir sus movimientos.

El competidor de Qualcomm, Intel, anunció en la feria Computex de Taiwán la disponibilidad de la cámara Creative Senz3D, un periférico con visión 3D que imita las prestaciones de Kinect, el sensor que reacciona a los movimientos y la voz del usuario lanzado por Microsoft para su consola Xbox 360. Intel tiene previsto que lo que nace como periférico pase a ser un componente más de serie en los futuros ordenadores y tabletas.

Dos empresas de Israel, EyeSight y Pointgrab, se encuentran en la vanguardia del control gestual para cámaras 2D, un mercado incipiente que copan entre las dos y en el que tienen firmados contratos con tecnológicas como Lenovo, Fujitsu, Acer, Toshiba, Asus y Samsung. En ambos casos, su software es capaz no solo de cambiar de pantalla cuando el usuario agita la mano, también reconoce otros movimientos que hacen las veces del tradicional "clic" del botón izquierdo del ratón, o un zoom en un mapa con tan solo juntar o separar los dedos. [2]

1.1.1 DESARROLLO FUTURO

Si bien las posibilidades de desarrollar aplicaciones increíbles con la tecnología de Control Gestual son bastante amplias, el software y el dispositivo aún se encuentran en una fase muy incipiente. La precisión es un punto fundamental a mejorar, sobre todo cuando se pretende incursionar en ámbitos tan importantes como la salud y la ingeniería. El Control Gestual aún está lejos de ser empleado para realizar una cirugía pero como dispositivo de control gestual, en el ámbito tecnológico general, ha hecho importantes avances.

Muchas compañías han demostrado su interés por esta nueva tecnología y algunas ya la han incorporado en sus productos. Por ejemplo, HP anunció la integración nativa de la tecnología del Control Gestual en sus nuevas laptops y Google incorporó soporte para control gestual en la nueva versión de Google Earth. En el ámbito de los videojuegos, el

Control Gestual constituye una excelente opción para ejecutar ideas más ambiciosas y juegos nunca antes vistos.

No es muy posible que el Control Gestual se convierta en corto plazo en el sustituto del mouse pero sí que será adoptado como dispositivo complementario de control y, a medida que sigan desarrollándose aplicaciones, se masificará su uso llegando a ser algo tan cotidiano como un mouse.

1.2 HISTORIA DEL CONTROL GESTUAL

Kinect supuso hace dos años toda una revolución para la industria de los videojuegos. Al jugador le permitía interactuar con la pantalla del televisor o el ordenador con los gestos de su cuerpo, por lo que se ampliaban con creces las capacidades del mando. Este dispositivo, pensado para la videoconsola de Microsoft Xbox 360, se basa en una barra sensora que crea un campo de rayos infrarrojos combinado con un sensor de imagen del tipo CMOS, para detectar los movimientos en tres dimensiones. Sin embargo, pronto se le encontraron otros usos relacionados con la gestión de ordenadores, dando paso a la era del 'control gestual' de dispositivos tecnológicos. Hoy son muchas las empresas que diseñan tecnologías de este tipo, que diversos fabricantes ya aplican a sus equipos. El próximo paso serán las tabletas y los móviles.

Kinect era en realidad un diseño de la empresa israelí PrimeSense que Microsoft adaptó a la Xbox dentro del llamado "Project Natal". Pero PrimeSense, lejos de vender la patente del hardware a Microsoft, dejó claro que la tecnología tenía usos más amplios y que estaba en contacto con fabricantes de ordenadores para crear nuevos productos.

Por su lado, Sony desarrolló su propia tecnología llamada Move para la consola PlayStation 3 que lanzó al mercado casi al mismo tiempo que lo hacía Microsoft con la Xbox 360. Move era más similar al mando Wii Remote de la consola Wii de Nintendo, verdadero pionero en el control de videojuegos por movimientos, pero no iba conectado por un cable con la videoconsola, sino que trabajaba de forma inalámbrica. Se basaba en una cámara llamada PlayStation Eye, que se situaba encima de la pantalla (igual que la barra de Kinect), y en el mando Motion Controller, que estaba plagado de sensores de movimiento y tenía en su parte superior una esfera que se iluminaba en diferentes colores para poder ser detectada por la cámara. [3]

1.2.1 APLICACIÓN EN LOS ORDENADORES

Tras PrimeSense, otras empresas han creado desarrollos de barras de gestos para ordenadores. Una de ellas es la californiana Leap Motion, que a finales de junio llegó a un

acuerdo con un fondo de inversión para dotarse de 20 millones de euros, con el fin de impulsar la comercialización de su tecnología de control gestual externa para ordenadores.

De momento, esta se encarna en un dispositivo llamado Leap Motion Controller, que se conecta por USB o Bluetooth (en Mac OS X). Sin necesidad de calibrado, detecta la mano del usuario frente a la webcam del ordenador, con la que se coordina para procesar órdenes como cambiar de aplicación o de pestaña en el navegador, subir o bajar el volumen de los altavoces, realizar dibujos a mano alzada**,** etc., como se puede ver en este vídeo. Está pensado para los sistemas operativos Mac OS X y Windows 8 y tiene un precio de salida de 80 dólares, por el momento solo para Estados Unidos. El fabricante Acer ha llegado a un acuerdo con Leap Motion para incorporar de manera interna el dispositivo en algunos de sus terminales.

Por su parte, los ordenadores de la gama Vaio de Sony incorporan desde hace un año una tecnología de control gestual ahora adaptada a Windows 8, que apuesta en especial por estas tecnologías como factor de diferenciación frente a Mac OS X.

1.3 SENSORES DE MOVIMIENTO

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, movimiento, etc.

Los sensores de movimiento son aparatos basados en la tecnología de los rayos infrarrojos o las ondas ultrasónicas para poder captar en tiempo real los movimientos que se generan en un espacio determinado. Estos sensores de movimiento, adscritos sobre todo a cámaras de seguridad, puertas en almacenes y centros comerciales, etc; son uno de los dispositivos más reconocidos e importantes dentro de la seguridad electrónica, que tanto ha apostado por, sobre todo, dos aspectos fundamentales: el tamaño y la funcionalidad de cada uno de los equipos que usan durante el proceso. [4]

1.3.1 CARACTERÍSTICAS

- ***Rango de medida:*** dominio en la magnitud medida en el que puede aplicarse el sensor.
- ***Precisión:*** es el error de medida máximo esperado.
- ***Offset o desviación de cero:*** valor de la variable de salida cuando la variable de entrada es nula. Si el rango de medida no llega a valores nulos de la variable de entrada, habitualmente se establece otro punto de referencia para definir el offset.
- ***Linealidad*** o correlación lineal.

- ***Sensibilidad de un sensor:*** relación entre la variación de la magnitud de salida y la variación de la magnitud de entrada.
- ***Resolución:*** mínima variación de la magnitud de entrada que puede apreciarse a la salida.
- ***Rapidez de respuesta:*** puede ser un tiempo fijo o depender de cuánto varíe la magnitud a medir. Depende de la capacidad del sistema para seguir las variaciones de la magnitud de entrada.
- ***Derivas:*** son otras magnitudes, aparte de la medida como magnitud de entrada, que influyen en la variable de salida. Por ejemplo, pueden ser condiciones ambientales, como la humedad, la temperatura u otras como el envejecimiento (oxidación, desgaste, etc.) del sensor.
- ***Repetitividad:*** error esperado al repetir varias veces la misma medida.

Un sensor es un tipo de transductor que transforma la magnitud que se quiere medir o controlar, en otra, que facilita su medida. Pueden ser de indicación directa (ej. un termómetro de mercurio) o pueden estar conectados a un indicador (posiblemente a través de un convertidor analógico a digital, un computador y un display) de modo que los valores detectados puedan ser leídos por un humano. [5]

1.3.2 TIPOS DE SENSORES

El sensor de movimiento en cuanto a su zona de instalación se clasifica entre apto para exterior o para aplicación interior.

El detector de movimiento, en cuanto al sistema de detección que utiliza se clasifica como:

- Detector de movimiento de rayos infrarrojos pasivo.
- Detector de movimiento de microondas.
- Detector de movimiento dual de rayos infrarrojos y de microondas.
- Detector de movimiento de ultrasonidos.

1.3.3 FUNCIONAMIENTO DE LOS SENSORES DE MOVIMIENTO

Todo detector de movimiento consta de una unidad emisora y receptora, conectada a otra unidad central. Ambas unidades generalmente están alejadas entre sí.

❖ **Detector de movimiento de infrarrojos pasivo**

Va equipado con uno dos sensores infrarrojos, que transmiten su señal de salida a la unidad central. Las variaciones de temperatura uniformes de la zona a vigilar no son consideradas.

Una rápida variación de la radiación infrarroja, producida por la entrada en escena de un intruso, dispara la situación de alarma.

La unidad central responde ejecutando automáticamente la secuencia de actuación, previamente programada. El detector de rayos infrarrojos, tiene menor alcance de detección que el detector de microondas.

❖ **Detector de movimientos de microondas**

Van equipados con un emisor de microondas, y un detector doppler que puede contar hasta con dos canales de recepción, para evitar falsas alarmas. La unidad central memoriza el nivel de respuesta a la señal emitida recibida de la zona a proteger, estableciendo un nivel de alarma. Si este nivel, es alcanzado por la variación de la señal recibida, debida a la entrada de un intruso.

La unidad central pasa a situación de alarma, y ejecuta automáticamente la secuencia de actuación previamente programada. Existen en el mercado, equipos que incluso llegan a detectar intrusos que se mueven gateando, o arrastrándose.

❖ **Detector dual de rayos infrarrojos y de microondas**

Incorpora los dos sistemas de detección, y sólo se llega a la situación de alarma, si los dos sistemas alcanzan este nivel. Con este sistema se evita una gran cantidad de falsas alarmas.

❖ **Detector de movimiento de ultrasonidos**

Va equipado con un emisor, y un receptor de ultrasonidos. La variación de la frecuencia de la onda recibida respecto a la emitida, provoca el disparo de la alarma en la unidad central, y la ejecución de las acciones previamente programadas. Algunos equipos incorporan un regulador de luminosidad que pilota automáticamente mediante un regulador electrónico, los puntos de luz previamente determinados.

El sensor de Kinect es una barra horizontal conectado a un pivote, diseñado para estar en una posición longitudinal. El dispositivo tiene una cámara RGB, sensor de profundidad y un micrófono multi-array bidireccional que conjuntamente capturan el movimiento de los cuerpos en 3D, además de ofrecer reconocimiento facial y aceptar comandos de voz.

El sensor de Kinect reproduce video a una frecuencia de 30 Hz, en colores RGB 32-bit y resolución VGA de 640×480 pixels, el canal de video monocromo es de 16-bit, resolución QVGA de 320×240 pixels con hasta 65,536 niveles de sensibilidad. El límite del rango visual del sensor de Kinect está entre 1.2 y 3.5 metros de distancia, con un ángulo de vista de 57° horizontalmente y un ángulo de 43° verticalmente, mientras que el pivote puede

orientarse hacia arriba o abajo ampliando hasta 27°. El array del micrófono tiene cuatro cápsulas, y opera con cada canal procesando 16-bit de audio con un ratio de frecuencia de 16 kHz. [6]

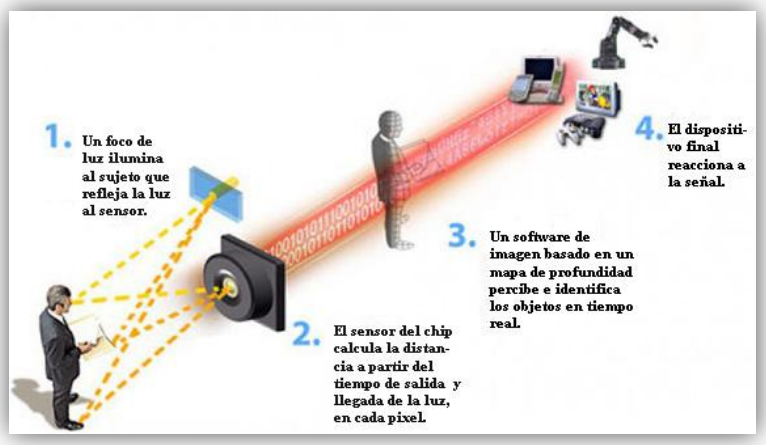

Figura 1.1.- Representación del funcionamiento de un sensor

1.4 CIENCIAS DE LA COMPUTACIÓN

Los investigadores del MIT construyeron un dispositivo que controla la computadora a través de movimientos de la mano y los dedos.

Investigadores de Fraunhofer Institute de Alemania crearon un elegante sistema de control de PCs con Windows que se basa en gestos de las manos.

Microsoft tiene un producto increíblemente prometedor en sus manos, y algo aún mejor: Una plataforma estable.

No hay duda de que los dispositivos basados en gestos serán la tendencia del futuro. Los usos son tan amplios que de inmediato pasó lo obvio: muchas personas ya han invertido tiempo y esfuerzo en dominar el sistema.

Microsoft está ayudando con su SDK, pero para aprovechar realmente esta oportunidad única Microsoft debería financiar proyectos de Kinect y ver lo que construyen. [7]

CAPITULO 2
SISTEMAS DE CONTROL Y DETECCIÓN

Lo esencial para que funcione el móvil, es el uso de una cámara. Esta puede ser del tipo cámara web (webcam), tanto que venga incluida en la computadora o portátil; o bien, usar la cámara del dispositivo Kinect, que es el que utilizaremos nosotros.

En el caso de las cámaras web sería necesario realizar un proceso de configuración y programarla para que detecte puntos del cuerpo, sombras y demás. Cosa que el sensor Kinect ya trae incluido y por eso nos basaremos en él.

Con ayuda del software "Microsoft SDK de Kinect", versión 1.7, realizamos la detección de puntos clave en el esqueleto del usuario.

Kinect es un accesorio para el Microsoft XBox 360 que consiste en una cámara sensible a la profundidad, lo que le permite identificar lo que ve en un contexto de 3 dimensiones. En principio fue pensado como accesorio para videojuegos, pero sus capacidades permiten que también sea un dispositivo útil para otros fines. De hecho, apenas algunos días después de su lanzamiento, comenzaron a aparecer drivers de Kinect creados por desarrolladores independientes. Reconociendo este potencial, Microsoft liberó posteriormente un SDK oficial. [8]

2.1 EL SENSOR KINECT Y SU COMPOSICIÓN

El sensor de Kinect es un equipo alargado conectado a un pivote, diseñado para estar en una posición horizontal. El dispositivo tiene una cámara RGB, un sensor de profundidad y un micrófono multi-array bidireccional que, en conjunto, capturan imágenes y movimientos de los cuerpos en 3D, además de ofrecer reconocimiento facial y aceptar comandos de voz.

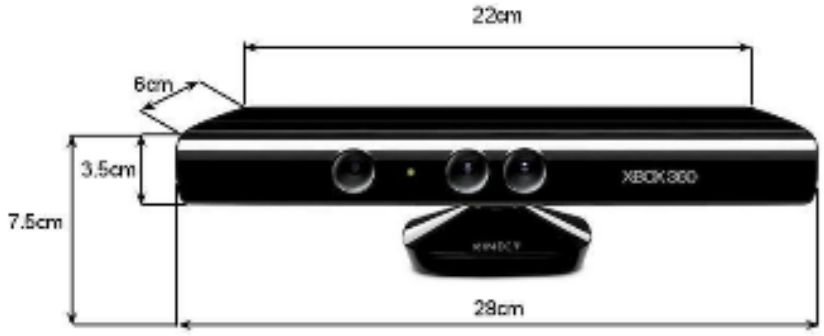

Figura 2.1.- Dimensiones del sensor Kinect

El sensor de Kinect adquiere imágenes de video con un sensor CMOS de colores a una frecuencia de 30 Hz, en colores RGB de 32 bits y resolución VGA de 640×480 pixeles. El canal de video monocromo CMOS es de 16 bits, resolución QVGA de 320×240 pixeles con hasta 65,536 niveles de sensibilidad.

Para calcular distancias entre un cuerpo y el sensor, el sensor emite un haz láser infrarrojo que proyecta un patrón de puntos sobre los cuerpos cuya distancia se determina. Una cámara infrarroja capta este patrón y por hardware calcula la profundidad de cada punto. El rango de profundidad del sensor de Kinect está entre 0.4 y 4 mts. Existen 2 modos (Default y Near) para determinar distancias. Se ha elegido el modo "Default" ya que permite medir hasta 4 metros de distancia con respecto al sensor.

El ángulo de vista (FOV) es de 58° horizontales y 45° verticales. Por otro lado el pivote permite orientar en elevación, hacia arriba o hacia abajo incrementando el FOV hasta en 27°. El array del micrófono tiene cuatro cápsulas, y opera con cada canal procesando en 16 bits de audio con un ratio de frecuencia de 16 kHz.

2.1.1 FUNCIONAMIENTO

La cámara de Kinect funciona con hardware y software especializado para el reconocimiento de imagen. La cámara tiene dos funcionalidades principales, genera un mapa de 3D de la imagen que tiene en su campo visual y reconoce humanos en movimiento entre los objetos de la imagen a partir de diferentes segmentos de las articulaciones del cuerpo y un esquema en escala de grises del rostro. La cámara de profundidad tiene un papel muy importante ya que permite diferenciar a una persona a pesar de que ésta utilice prendas de vestir que tengan el mismo color que el fondo

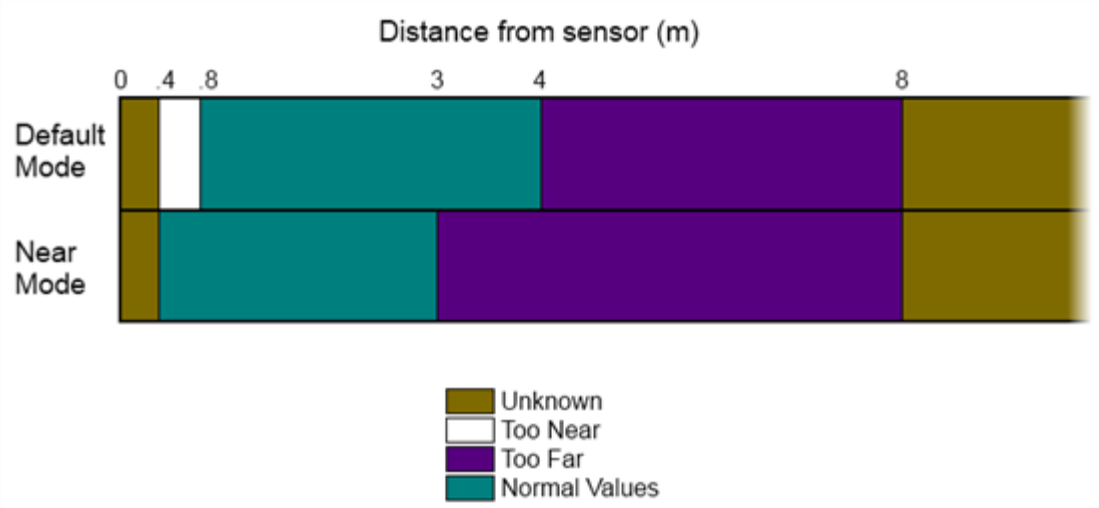

Figura 2.2.- Rango de reconocimiento de la cámara Kinect

Existe software antiguo que utilizaba las diferencias de textura y color para poder distinguir los objetos del fondo. PrimeSense, quien desarrolló Kinect, y Canesta, recién adquirida por Microsoft, utilizan un modelo distinto. La cámara transmite una luz invisible para el ojo humano, cercana a los infrarrojos en el espectro, y puede conocer el tiempo que tarda la luz en retornar al sensor después de ser reflejada en los objetos. Canesta es una empresa localizada en Sunnyvale, California, un fabricante de chips que hace que los dispositivos electrónicos reaccionen a los movimientos del usuario, crea interfaces que no requieren de periféricos con botones que sean ajustados con la mano y conectados por cable.

El sensor de profundidad actúa como un sonar, la operación no es muy complicada, es posible cuando se conoce el tiempo que le toma la salida y llegada a la luz después de ser reflejada en un objeto, conociendo la velocidad absoluta de la luz, es posible obtener la distancia a la cual se localiza el objeto. En un campo amplio con objetos, el sensor Kinect trata de reconocer a qué distancia se encuentran los objetos, distinguiendo el movimiento en tiempo real. Kinect puede llegar a distinguir la profundidad de cada objeto con diferencias de un centímetro y su altura y ancho con diferencias de tres milímetros. El hardware de Kinect está compuesto por la cámara y el proyector de luz infrarroja, añadido al firmware y a un procesador que utiliza algoritmos para procesar las imágenes tridimensionales.

Con el procesador es posible interpretar los movimientos que se registran en los objetos capturados por la cámara de Kinect en eventos con significado que aparecen en la pantalla. Estos movimientos son buscados por el algoritmo y luego contextualizados, es decir, se buscará la identificación de los movimientos en tiempo real para producir los eventos que se requieran.

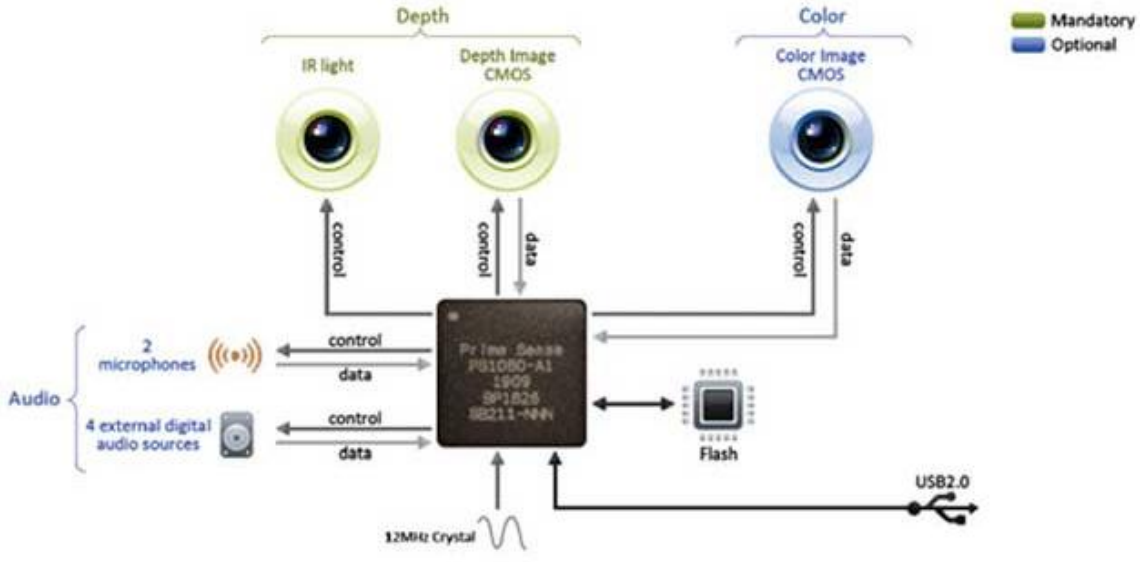

Figura 2.3.- Composición interna de un sensor Kinect

2.2 DETECCIÓN DE ARTICULACIONES MEDIANTE EL SENSOR KINECT

Normalmente, el flujo de datos "crudos" que suministra el sensor Kinect proporciona la información de 20 puntos de una persona cuando está dentro del rango visual admisible. De esta manera, es posible obtener la información posicional en tiempo real y en los 3 ejes cartesianos de estos 20 puntos por persona. Cabe destacar que la información recolectada está en un formato imagen de 640x480 pixeles tomada a una velocidad de 30 FPS (Frames per Second o Cuadros por segundo).

Con el fin de obtener todas las trayectorias que realiza una persona, es necesario registrar y archivar los 20 puntos que entrega el dispositivo. Cabe destacar que esta es la funcionalidad más importante que tiene el sensor para la aplicación buscada, dado que otorga la posibilidad de tener la posición tridimensional en cada instante de tiempo. Si se toma una sucesión dinámica de estas entregas, se podrá obtener todas las trayectorias de todos las partes del cuerpo de una persona. [9]

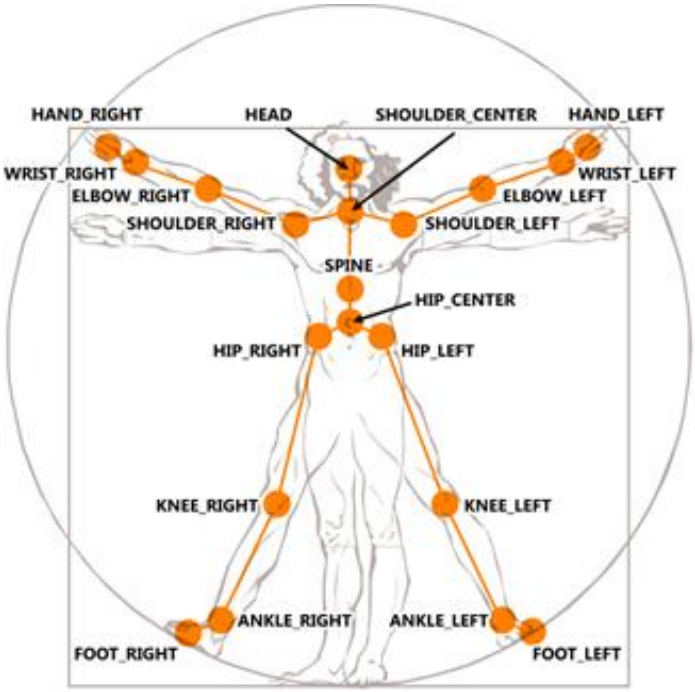

Figura 2.4.- Conjunto de 20 puntos detectado por el sensor Kinect

2.3 RECONOCIMIENTO DE GESTOS

Cuando nos referimos a reconocimiento de gestos, en este caso, se trata de asignar a ciertos movimientos consecutivos de partes del cuerpo una determinada acción (saltar, saludar, girar, etc.).

Al igual que nos pasa con la detección de posturas el reconocimiento de gestos también tiene muchas técnicas diferentes que se pueden aplicar para lograr una mejor identificación o una implementación más sencilla. Una técnica muy utilizada para estos casos en utilizar redes neuronales las cuales se pueden entrenar para ir alcanzando cada vez más precisión y calidad de detección.

También podemos usar técnicas como definir algorítmicamente el gesto, al igual que hicimos con la postura, o comparar con una serie de plantillas ya definidas. La técnica de las plantillas es muy eficiente para gestos que siempre se realizan de la misma forma y que con cualquier otra técnica sería muy difícil de detectar.

Para ilustrar la técnica de comparación de plantillas podemos poner el golpeo de una bola jugando al tenis. Este movimiento es bastante complejo como para definirlo algorítmicamente y es más fácil si tenemos una serie de plantillas (gestos) e ir comparando cada captura con ellas. El movimiento capturado no será igual a la plantilla que tengamos pero se puede permitir un margen de error y si el resto del movimiento se sigue pareciendo al resto de plantillas podemos darlo por válido.

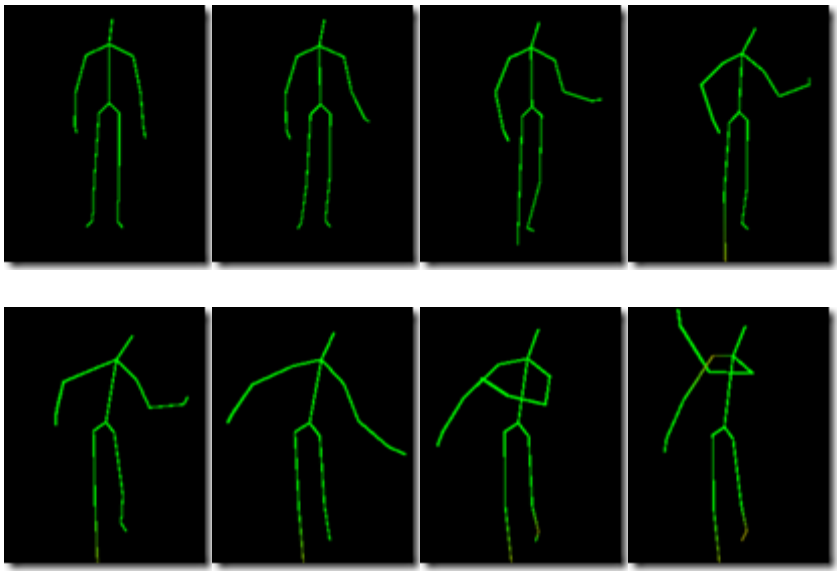

Figura 2.5.- Gestos captados frente al sensor Kinect

CAPITULO 3
PLATAFORMAS DE DESARROLLO

Para poder decirle al móvil las acciones que debe realizar y que el sensor conozca que comandos debe enviar, es necesario trabajar con plataformas para crear los códigos del programa.

En este proyecto se trabaja con Visual Studio 2010, ayudado por las librerías de SDK Kinect y, posteriormente, utilizar la plataforma ARDUINO para enviar los comandos al móvil y que este realicé la acción debida.

3.1 MICROSOFT VISUAL STUDIO

Visual Studio es una plataforma de desarrollo para sistemas operativos Windows. Soporta varios lenguajes de programación tales como C++, C#, J# y Visual Basic .NET, al igual que entornos de desarrollo web como ASP.NET. Aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

El entorno de desarrollo integrado (IDE) de Visual Studio presenta un conjunto de herramientas destinadas a ayudarle a escribir y modificar el código para los programas, así como a detectar y corregir errores en los programas.

Elegimos esta plataforma por su fácil manejo y, sobre todo, debido a que el sensor Kinect cuenta con una plataforma propia que puede ser fácilmente unida y utilizada con Visual Studio.

Para este proyecto se utilizó la versión 2010, puesto que es la más compatible y estable en el momento. Es por eso que los códigos del programa se realizaron en esta versión.

Figura 3.1.- Logotipo de Visual Studio

3.1.1 LENGUAJE DE PROGRAMACIÓN C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

El nombre C Sharp fue inspirado por la notación musical, donde '#' (sostenido, en inglés sharp) indica que la nota (C es la nota do en inglés) es un semitono más alta, sugiriendo que C# es superior a C/C++. Además, el signo '#' se compone de cuatro signos '+' pegados.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco Mono - DotGNU, el cual genera programas para distintas plataformas como Windows, Unix, Android, iOS, Windows Phone, Mac OS y GNU/Linux.

3.1.2 CARACTERÍSTICAS DEL LENGUAJE C#

El estándar ECMA-334 lista las siguientes características en el diseño para C#:

- ✓ Lenguaje de programación orientado a objetos simple, moderno y de propósito general.
- ✓ Inclusión de principios de ingeniería de software tales como revisión estricta de los tipos de datos, revisión de límites de vectores, detección de intentos de usar variables no inicializadas, y recolección de basura automática.
- ✓ Capacidad para desarrollar componentes de software que se puedan usar en ambientes distribuidos.
- ✓ Portabilidad del código fuente.
- ✓ Fácil migración del programador al nuevo lenguaje, especialmente para programadores familiarizados con C, C++ y Java.
- ✓ Soporte para internacionalización.
- ✓ Adecuación para escribir aplicaciones de cualquier tamaño: desde las más grandes y sofisticadas como sistemas operativos hasta las más pequeñas funciones.
- ✓ Aplicaciones económicas en cuanto a memoria y procesado.

3.2 MICROSOFT SDK KINECT

Un kit de desarrollo de software o SDK (Software Development Kit) es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador crear

aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, plataformas de hardware, computadoras, videoconsolas, sistemas operativos, etc.

Es algo tan sencillo como una interfaz de programación de aplicaciones o API (Application Programing Interface) creada para permitir el uso de cierto lenguaje de programación, o puede, también, incluir hardware sofisticado para comunicarse con un determinado sistema embebido. Las herramientas más comunes incluyen soporte para la detección de errores de programación como un entorno de desarrollo integrado o IDE (Integrated Development Environment) y otras utilidades. Los SDK frecuentemente incluyen, también, códigos de ejemplo y notas técnicas de soporte u otra documentación de soporte para ayudar a clarificar ciertos puntos del material de referencia primario.

El SDK no sólo incluye controladores sino también interfaces de programación de aplicaciones (API's), interfaces de dispositivo, documentos de instalación y materiales con recursos. Esto representa otro emocionante hito para la tecnología que ha capturado la imaginación de millones y se ha convertido en el dispositivo electrónico de ventas más rápidas de todos los tiempos.

Para este proyecto utilizamos la versión más actualizada del SDK de Kinect, la versión 1.7. Con plantillas y programas actualizados.

3.2.1 CARACTERÍSTICAS DE LA VERSION 1.7

El SDK de Kinect tiene varias características adicionales:

- **Drivers (para que el PC reconozca la Kinect)**: necesitamos que el ordenador reconozca todas las capacidades del dispositivo, las cámaras y los micrófonos.

- **NUI:** Se engloban en este área todos los métodos relacionados con las cámaras

- **Imagen de la cámara:** Una imagen de lo que está captando la cámara principal en estos momentos.

- **Datos de profundidad:** Es también una imagen, pero en escala de grises que es capaz de detectar la profundidad de una figura.

- **Seguimiento de esqueleto:** Es una de las características más destacadas de Kinect, ya que es capaz de detectar 20 articulaciones de 2 personas diferentes orientadas al sensor.

- **Speech:** Esta área engloba todos los métodos relacionados con la escucha y comandos de voz.

- **Captura de audio con detección de ruido:** El hecho de que Kinect tenga 4 micrófonos permite aislar el ruido y detectar la fuente de las órdenes concretas.

- **Comandos de voz:** Además, usando la tecnología de reconocimiento de voz, se pueden crear diccionarios personalizados para que el programa reconozca nuestras órdenes.

3.2.2 ¿POR QUÉ KINECT?

La cámara del sensor Kinect cuenta ya con la electrónica necesaria para que el sensor funcione a nuestras necesidades; Solo necesitamos programar, en Visual Studio, el código necesario para que detecte ciertos puntos del esqueleto humano y que los comandos sean enviados al móvil.

SDK de Kinect, otorga la facilidad de contar con las librerías que incluyen dichas instrucciones, como es la de detectar puntos en el cuerpo y reconocer gestos realizados. Esto hace que la programación sea menos complicada y más eficaz.

3.3 ARDUINO

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

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo costo que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (boot loader) que corre en la placa.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

Figura 3.2.- Logotipo de ARDUINO

3.3.1 BIBLIOTECAS EN ARDUINO

Para hacer uso de una biblioteca en Sketch (el IDE de Arduino), basta con hacer clic sobre "Import Library" en el menú, escoger una biblioteca y se añadirá el #include correspondiente. Las bibliotecas estándar que ofrece Arduino son las siguientes:

- Serial: Lectura y escritura por el puerto serie.

- EEPROM: Lectura y escritura en el almacenamiento permanente.
    - read(), write()

- Ethernet: Conexión a Internet mediante "Arduino Ethernet Shield". Puede funcionar como servidor que acepta peticiones remotas o como cliente. Se permiten hasta cuatro conexiones simultáneas.
    - Servidor: Server(), begin(), available(), write(), print(), println()
    - Cliente: Client(), connected(), connect(), write(), print(), println(), available(), read(), flush(), stop()

- Firmata: Comunicación con aplicaciones de ordenador utilizando el protocolo estándar del puerto serie.
- LiquidCrystal: Control de LCDs con chipset Hitachi HD44780 o compatibles. La biblioteca soporta los modos de 4 y 8 bits.
- Servo: Control de servo motores. A partir de la versión 0017 de Arduino la biblioteca soporta hasta 12 motores en la mayoría de placas Arduino y 48 en la Arduino Mega.
    - attach(), write(), writeMicroseconds(), read(), attached(), detach()

El manejo de la biblioteca es bastante sencillo. Mediante attach(número de pin) añadimos un servo y mediante write podemos indicar los grados que queremos que tenga el motor (habitualmente de 0 a 180).

- SoftwareSerial: Comunicación serie en pines digitales. Por defecto Arduino incluye comunicación sólo en los pines 0 y 1 pero gracias a esta biblioteca podemos realizar esta comunicación con el resto de pines.

- Stepper: Control de motores paso a paso unipolares o bipolares.
    - Stepper(steps, pin1, pin2), Stepper(steps, pin1, pin2, pin3, pin4), setSpeed(rpm), step(steps)

El manejo es sencillo. Basta con iniciar el motor mediante Stepper indicando los pasos que tiene y los pines a los que está asociado. Se indica la velocidad a la que queramos que gire en revoluciones por minuto con setSpeed(rpm) y se indican los pasos que queremos que avance con step(pasos).

- Wire: Envío y recepción de datos sobre una red de dispositivos o sensores mediante Two Wire Interface (TWI/I2C).

Además las bibliotecas Matrix y Sprite de Wiring son totalmente compatibles con Arduino y sirven para manejo de matrices de leds.

También se ofrece información sobre diversas bibliotecas desarrolladas por contribuidores diversos que permiten realizar muchas tareas.

3.3.2 MICROCONTROLADORES PARA ARDUINO

Los microcontroladores Arduino Diecimila, Arduino Duemilanove y Arduino Mega están basados en Atmega168, Atmega 328 y Atmega128.

Tabla 3.1: Características de los microcontroladores Atmega para ARDUINO

	Atmega168	Atmega328	Atmega1280
Voltaje operativo	5 V	5 V	5 V
Voltaje de entrada recomendado	7 - 12 V	7 - 12 V	7 - 12 V
Voltaje de entrada límite	6 - 20 V	6 - 20 V	6 - 20 V
Pines de entrada y salida digital	14 (6 proporcionan PWM)	14 (6 proporcionan PWM)	54 (14 proporcionan PWM)
Pines de entrada analógica	6	6	16
Intensidad de corriente	40 mA	40 mA	40 mA
Memoria Flash	16KB (2KB reservados para el bootloader)	32KB (2KB reservados para el bootloader)	128KB (4KB reservados para el bootloader)

SRAM	1 KB	2 KB	8 KB
EEPROM	512 bytes	1 KB	4 KB
Frecuencia de reloj	16 MHz	16 MHz	16 MHz

CAPITULO 4
INTEGRACIÓN DE CÓDIGOS PARA LA IMPLEMENTACIÓN DEL CONTROL GESTUAL

En este capítulo se verá el desarrollo propuesto para la implementaciones de control gestual; la detección de puntos en el cuerpo, declaración de comandos para gestos, envío de información y la compilación desde Visual Studio que es el compilador que se eligió para realizar los códigos, ya que el SDK de Kinect está configurado para trabajar en productos Microsoft, además de que las funciones utilizadas están basadas en lenguaje C.

4.1 RECONOCIMIENTO DE PUNTOS EN EL CUERPO

Para este proyecto fue necesario contar con un sensor Kinect, y Descargar el SDK de Kinect para Windows y tener alguna versión de Visual Studio 2010. Para programar con el SDK utilizamos la herramienta de desarrollo Visual Studio 2010 y Windows 7.

El SDK de Kinect nos permite obtener los puntos de articulaciones (Joints) del esqueleto y su posición en el espacio de una forma sencilla.

Para poder usar el Detector del Esqueleto (Skeletal tracking) añadimos la opción *UseSkeletalTracking* cuando inicializamos el sensor y creamos el evento *SkeletonFrameReady* que nos permitirá capturar los datos del esqueleto una vez que se obtengan.

```
kinect = new Runtime();
kinect.Initialize(RuntimeOptions.UseDepthAndPlayerIndex|RuntimeOptio
ns.UseSkeletalTracking);
kinect.SkeletonFrameReady      +=      new      EventHandler      <
SkeletonFrameReadyEventArgs > (kinect_SkeletonFrameReady);
```

En el método *kinect_SkeletonFrameReady* que se habrá creado será donde implementemos el código para obtener las partes del cuerpo. Para ello tenemos que obtener los datos del *Skeleton* a partir del *SkeletonFrame* que se captura. El *SkeletonFrame* puede tener varios esqueletos pero sólo nos interesa los que tengan datos. Utilizaremos un *foreach* para realizar la detección de gestos por cada esqueleto correcto que genere el sensor. El siguiente paso es saber si ese esqueleto es correcto o no. Eso se hace mediante la propiedad TrackingState (NotTracked, PositionOnly o Tracked).

```
void kinect_SkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
{
         foreach (SkeletonData skeleton in
e.SkeletonFrame.Skeletons)
          if (skeleton.TrackingState ==
SkeletonTrackingState.Tracked)
    {
```

El SDK sólo nos da la opción de acceder a los datos de 2 esqueletos a la vez. El resto de posiciones del vector serán esqueletos vacíos (NotTracked) por ello tenemos que ir recorriendo el vector hasta conseguir el correcto.

Ya tenemos el esqueleto, ahora obtenemos las posiciones de las partes del cuerpo. Para ello tenemos un enumerado llamado *JointID* que utilizamos para obtener del vector de *Joints* el elemento que queremos de una forma más natural. Por ejemplo vamos a tomar la posición de la mano derecha.

```
float HandRightX = skeleton.Joints[JointID.HandRight].Position.X;
float HandRightY = skeleton.Joints[JointID.HandRight].Position.Y;
float HandRightZ = skeleton.Joints[JointID.HandRight].Position.Z;
```

Con la propiedad *position* obtenemos la posición con respecto al sensor. Con las posiciones de las distintas partes podremos detectar gestos y posturas usando diferentes técnicas.

Para trabajar mejor con las posiciones definimos una estructura de datos llamada Vector3. Tendrá 3 propiedades, una por cada valor de dimensión facilitando la tarea de almacenar y acceder a los valores de posición de la parte del cuerpo correspondiente.

```
public struct Vector3
{
    public float X;
    public float Y;
    public float Z;
}
```

Cambiamos el código anterior utilizando la nueva estructura de datos para la mano derecha y la cabeza:

```
Vector3 handRight = new Vector3( );

handRight.X = skeleton.Joints[JointID.HandRight].Position.X;
handRight.Y = skeleton.Joints[JointID.HandRight].Position.Y;
handRight.Z = skeleton.Joints[JointID.HandRight].Position.Z;

Vector3 head = new Vector3( );

head.X = skeleton.Joints[JointID.Head].Position.X;
head.Y = skeleton.Joints[JointID.Head].Position.Y;
head.Z = skeleton.Joints[JointID.Head].Position.Z;
```

4.1.1 DETECTAR POSTURAS

Ahora se a definir algorítmicamente la postura que se quiere detectar. Se restan las 3 dimensiones de mano y cabeza entre si y, si la distancia resultante está dentro de un rango de error que definimos nosotros, se da por buena la postura.

```
   bool HandOnHead(Vector3 hand, Vector3 head)
           {
              float distance = (hand.X - head.X) + (hand.Y - head.Y) +
(hand.Z - head.Z);
                        if (Math.Abs(distance) > 0.05f)
                        return false;
                        else
                         return true;
           }
```

Lo siguiente es crear lo que se llama el detector de posturas. En él se lleva la cuenta de posturas en proceso de detección, ya que una postura se tiene que mantener por un periodo de tiempo antes de convertirse en una postura detectada.

Para esto necesitamos unas propiedades donde definimos el número de veces que se tiene que identificar una postura antes de considerarse postura detectada, un acumulador para llevar la cuenta de las posturas detectadas, una propiedad para saber qué postura está en proceso de detección y otra para almacenar la última postura detectada. También se tendrá un enumerado de posturas para que sea más fácil hacer referencia a ellas.

```
const int PostureDetectionNumber = 10;
int accumulator = 0;
Posture postureInDetection = Posture.None;
Posture previousPosture = Posture.None;
public enum Posture
{
    None,
    HandOnHead
}
```

Al detector de posturas se le pasará la postura que se identificó algorítmicamente. Dentro se comprueba si la postura es la misma que está en proceso de detección.

Finalmente se llama a la función dentro del bucle anterior, justo después del código de obtención de la posición de mano y cabeza, mostrando el resultado en un *Label* que se añadió previamente a la ventana principal.

```
if (HandOnHead(handRight, head))
{
      if (PostureDetector(Posture.HandOnHead))
```

```
            label1.Content = previousPosture.ToString();
        }
          else
             if(PostureDetector(Posture.None))
                label1.Content = previousPosture.ToString();
```

Estas funciones sirven para detectar los puntos del esqueleto, y verlos en el monitor.

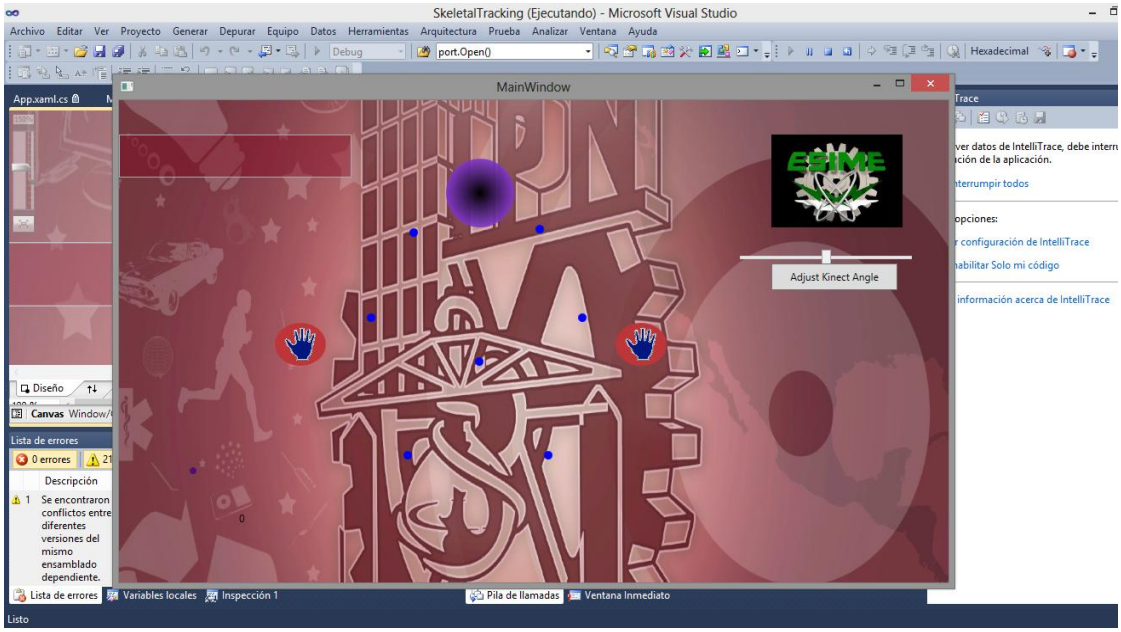

Figura 4.1.- Visualización de la detección del esqueleto en Visual Studio 2010

4.2 RECONOCIMIENTO DE GESTOS

Cuando nos referimos a reconocimiento de gestos, en este caso, se trata de asignar a ciertos movimientos consecutivos de partes del cuerpo una determinada acción (saltar, saludar, girar, etc.).

Al igual que pasa con la detección de posturas el reconocimiento de gestos también tiene muchas técnicas diferentes que se pueden aplicar para lograr una mejor identificación o una implementación más sencilla. Una técnica muy utilizada para estos casos en utilizar redes neuronales las cuales se pueden entrenar para ir alcanzando cada vez más precisión y calidad de detección.

También se pueden usar técnicas como definir algorítmicamente el gesto, al igual que con la postura, o comparar con una serie de plantillas ya definidas. La técnica de las plantillas es muy eficiente para gestos que siempre se realizan de la misma forma y que con cualquier otra técnica sería muy difícil de detectar.

Para poder identificar gestos se necesitan 2 listas, una de tipo Vector3 para almacenar las posiciones que se obtengan y otra para llevar la cuenta de los gestos que se han identificado.

```
List<Vector3> positionList = new List<Vector3>();
List<Gesture> gestureAcceptedList = new List<Gesture>();
```

Después se crean las propiedades que se usarán para definir el gesto, para la longitud, el ancho y duración del movimiento. También se tendrá otra propiedad para saber cuándo ha sido detectado el último gesto y otra por si se requiere añadir un tiempo mínimo entre detecciones de gesto. Por ultimo, un enumerado para nombrar los gestos.

```
const float SwipeMinimalLength = 0.4f;
const float SwipeMaximalHeight = 0.2f;
const int SwipeMininalDuration = 250;
const int SwipeMaximalDuration = 1500;
DateTime lastGestureDate = DateTime.Now;
int MinimalPeriodBetweenGestures = 0;

public enum Gesture
{
     None,
     Swipe
}
```

Lo siguiente es crear la función que se va a encargar de recorrer la lista de posiciones que se tiene, e ir comprobando si existe un conjunto de ellas que pueda ser el gesto que se quiere detectar.

Entre estas comprobaciones se encuentran las mencionadas anteriormente de conservar la misma altura de la mano y dirección correcta de movimiento.

Si cumple eso hay que comprobar si la longitud del gesto es la adecuada para más tarde calcular los tiempos para saber si la velocidad es la idónea y entra dentro del rango del periodo entre gestos (en este caso el periodo entre gestos es 0).

Por último se muestra que se ha detectado el gesto en una etiqueta, se añade el gesto a la lista de gestos aceptados, se asigna la fecha actual a la propiedad correspondiente a la fecha del último gesto detectado y se borra la lista de posiciones que se tiene.

```
void Swipe()
        {
            int start = 0;
            for (int index = 0; index < positionList.Count-1;
index++)
            {
                if ((Math.Abs(positionList[0].Y -
positionList[index].Y) > SwipeMaximalHeight) ||
(positionList[index].X - positionList[index + 1].X > -0.01f))
```

```
                {
                    start = index;
                }
                if ((Math.Abs(positionList[index].X -
positionList[start].X) > SwipeMinimalLength))
                {
                    double totalMilliseconds =
(positionList[index].date-
positionList[start].date).TotalMilliseconds;
                    if (totalMilliseconds >=
SwipeMininalDuration && totalMilliseconds <=
SwipeMaximalDuration)
                    {
                        if
(DateTime.Now.Subtract(lastGestureDate).TotalMilliseconds >
MinimalPeriodBetweenGestures)
                        {
                            label1.Content = "Swipe "+
gestureAcceptedList.Count;
                            gestureAcceptedList.Add(Gesture.Swip
e);
                            lastGestureDate = DateTime.Now;
                            positionList.Clear();
                        }
                    }
                }
            }
        }
```

Ahora se añade el evento *SkeletonFrameReady* en el lugar indicado donde se colocan las posiciones en la lista de posiciones, posteriormente se llama al detector de gestos y después se eliminan las posiciones sobrantes de la lista si se ha alcanzado el tope máximo que se asignó a 20.

```
positionList.Add(new Vector3()
{
    X = handRight.X,
    Y = handRight.Y,
    Z = handRight.Z,
    date = DateTime.Now
}
    );
Swipe();
if (positionList.Count() > 20)
{
    positionList.RemoveAt(0);
}
```

De esta forma se tiene el código necesario para ir implementando los gestos que se requiere que el programa reconozca para realizar las funciones que se desean y sean enviadas al móvil.

El programa se basa sólo en ciertos puntos del cuerpo y no en los veinte que incluye el SDK. Con esos puntos se hizo una asignación dependiendo de la posición correspondiente, basados en un tablero con doce casillas. Cada casilla corresponde a una acción, y también se agregan puntos de descanso en donde no se efectúa ninguna acción. Las acciones se efectuarán dependiendo de la casilla en donde se coloque la mano izquierda.

Figura 4.2.- Representación de comandos dependiendo la posición de la mano izquierda

4.2.1 CONTROL DE VELOCIDAD

Con la mano izquierda se tiene un control gestual para la dirección que debe tomar el móvil al desplazarse, pero también se incluye un control que responderá a los movimientos de la mano derecha, el control de la velocidad.

Se incluyó una barra la cual va aumentando los valores de la velocidad en valores de diez. En la parte izquierda se mostrarán los valores de la velocidad para llevar un mejor control.

Figura 4.3.- Visualización con barra de control de velocidad

4.3 PROGRAMACIÓN DE ARDUINO

En este software se programaron los comandos con los que se controlará el móvil. Se puede decir que es el corazón del proyecto. Se le asignan una tarea o ejecución que se deberá llevar a cabo al momento de recibir la instrucción.

Pretendemos que el móvil pueda desplazarse en los cuatro puntos cardinales; Norte, Sur, Este y Oeste. Cada punto representado por su letra inicial (ej. Norte es N).

```
void loop()
{
 if(Serial.available())
 {
   int c= Serial.read();

   if(c=='N')
   {
     digitalWrite(13,HIGH);
     digitalWrite(12,LOW);
     digitalWrite(11,LOW);
     digitalWrite(10,HIGH);
   }

   if(c=='O')
   {
     digitalWrite(13,HIGH);
     digitalWrite(12,LOW);
     digitalWrite(11,LOW);
     digitalWrite(10,LOW);
   }
```

```
    if(c=='S')
    {
      digitalWrite(13,LOW);
      digitalWrite(12,HIGH);
      digitalWrite(11,HIGH);
      digitalWrite(10,LOW);
    }

    if(c=='E')
    {
      digitalWrite(13,LOW);
      digitalWrite(12,LOW);
      digitalWrite(11,LOW);
      digitalWrite(10,HIGH);
    }

    else if(c=='D')
    {
      digitalWrite(13,LOW);
      digitalWrite(12,LOW);
      digitalWrite(11,LOW);
      digitalWrite(10,LOW);
    }

  }
}
```

Con estos comandos se declara que el móvil puede desplazarse según la señal que reciba. Esta señal es enviada a través de los códigos previamente programados desde Visual Studio, en la función "Process Gesture"

4.3.1 CONEXIÓN CON ARDUINO

Gracias a ARDUINO, es posible indicar al móvil las tareas que debe realizar. ARDUINO cuenta con un circuito con un microcontrolador que se encarga de interpretar las instrucciones enviadas, para que el móvil realice las tareas debidas.

Figura 4.1.- Placa de ARDUINO con microcotrolador

ARDUINO se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data).

En la plataforma ARDUINO se asignan parámetros para la ejecución de los comandos. Primero recibe la señal de puerto serial COM6, en este puerto es conectado en uno de los módulos de recepción inalámbrica Xbee, el cual transmitirá al segundo módulo colocado en un circuito conectado al móvil.

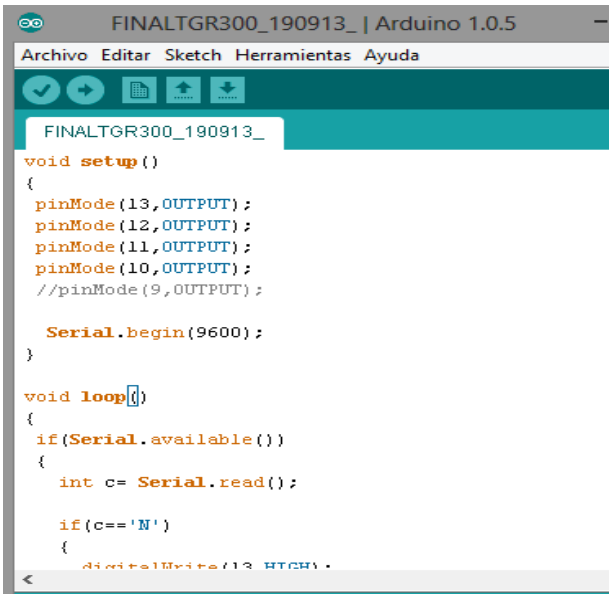

Figura 4.2.- Visualización de plataforma ARDUINO

Al llegar la señal al segundo Xbee, este la transmite al circuito de ARDUINO que ejecutará la traducción al móvil. La traducción está basada en cuatro órdenes representadas con los números del diez al trece. Cada número corresponde a una dirección en la que el móvil se desplazará.

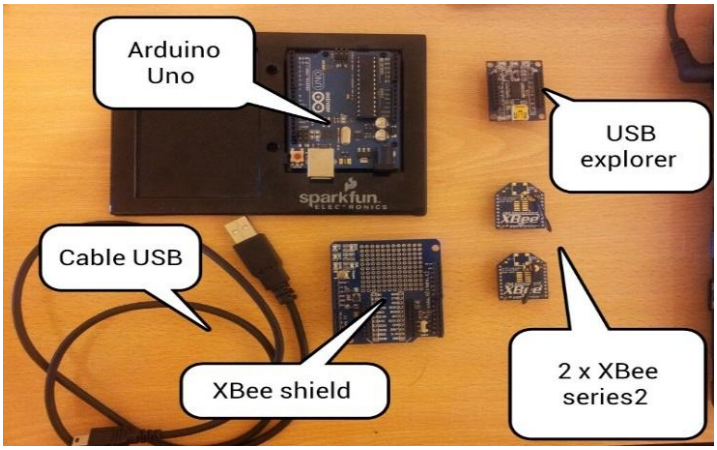

Figura 4.3.- Equipo utilizado para transmitir comandos de forma inalámbrica

La transmisión de la señal se puede representar en un diagrama, donde se conoce el camino por el que circula. Primero el sensor Kinect que captura la imagen y se muestra en la computadora. Según los gestos/comando que realicemos, estos serán transmitidos por medio de los transmisores XBee hacia la placa ARDUINO, la cual interpretará las lecturas para indicarle al móvil los movimientos que debe realizar.

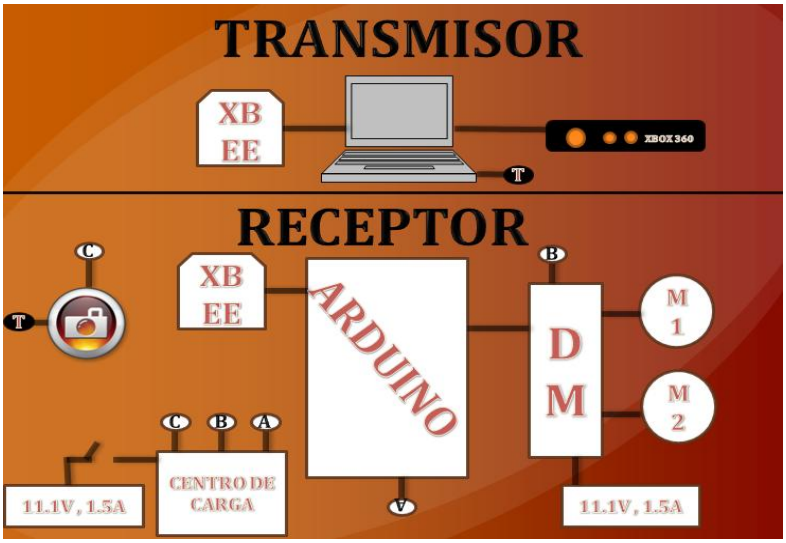

Figura 4.4.- Diagrama del funcionamiento de la transmisión

4.4 EL MÓVIL

Finalmente procedemos al ensamblado de los componentes. Se tiene el programa en Visual Studio con la captura de los puntos y el reconocimiento de los gestos, los sensores Xbee que manda la información, y la plataforma de ARDUINO que ejecutará los comandos. Ahora se agrega un último circuito compuesto por cuatro relevadores que funcionará como puente H.

Figura 4.5.- Circuito con 4 relevadores para el cambio de dirección

El circuito de la Figura 4.4 muestra los relevadores encargados de cambiar la ruta en que se desplazará el móvil (Adelante, Atrás, Izquierda o Derecha). Estos reciben la señal de los sensores y efectúan los cambios necesarios para que el móvil avance correctamente o, en todo caso, se detenga.

El móvil se construyó como un prototipo a un robot de exploración.

Figura 4.6.- Robot móvil finalizado

CONCLUSIONES

La realización de este proyecto nos introdujo a conocer las técnicas para el desarrollo de un control gestual, la detección de puntos del cuerpo humano y técnicas de transmisión de señales de forma inalámbrica.

El desarrollo de esta aplicación fue sencilla gracias al apoyo de las librerías del SDK de Kinect, que ayudaron detectando los puntos del cuerpo humano por medio de los sensores, lo cual facilitó el desarrollo del software para la aplicación. Principalmente fue implementado en Visual Studio 2010, esto también resulto conveniente para nosotros debido a que reafirmamos nuestros conocimientos en dicho software, los cuales hemos obtenido gracias a las materias impartidas en la ESIME.

ARDUINO y el transmisor Xbee fueron herramientas esenciales para nuestro proyecto. Sin ellas, no hubiera sido posible la comunicación inalámbrica. Nos ayudaron a adentrarnos más en el mundo de las comunicaciones inalámbricas y conocer como enviar y transmitir señales de esta manera.

Al proyecto se le pueden anexar muchas mejoras para que realice movimientos con mayor detalle y precisión. En nuestro caso, la implementación del control por medio de la voz nos sirvió como un auxiliar en caso de problemas, además de conocer a detalle la aplicación de este proceso.

El Control Gestual está en una etapa de desarrollo, y nuestra aplicación es tan solo un tentempié a las futuras tecnologías que pueden desarrollarse con ella. Nos podría ayudar a reducir tiempo y costos tanto para el usuario como para una empresa; el usuario ahorraría tiempo debido a que no será necesario un teclado o control físico para realizar las tareas, y la empresa podrá ahorrar costos de uso y equipo.

Además el Control Gestual no solo se aplica a un área, puede estar inmersa en distintos desarrollos de forma que nos puede ayudar a manipular mejor los objetos. Lo único que necesitamos para aplicar el Control Gestual es una cámara, como las que ya vienen incluidas en las computadoras personales, y un individuo que sea el control.

Se deja este desarrollo con la intención de que puedan desarrollarse nuevas aplicaciones y que sirva para resolver problemas, como siempre ha sido la intención de la ingeniería.

ANEXOS

ANEXO A
INSTALACIÓN DE MICROSOFT SDK KINECT EN VISUAL STUDIO

Una vez que tenemos el sensor Kinect, un ordenador con Windows y Visual Studio 2010 podemos proceder a descargar el SDK de Kinect para Windows eligiendo la versión de 32-bit o 64-bit.

Cuando esté descargado hay que tener en cuenta antes de instalar que no debemos tener instalado ningún otro driver de Kinect, el sensor no debe estar conectado ni Visual Studio abierto; Si esto se cumple se puede proceder a la instalación del SDK ejecutando el archivo descargado.

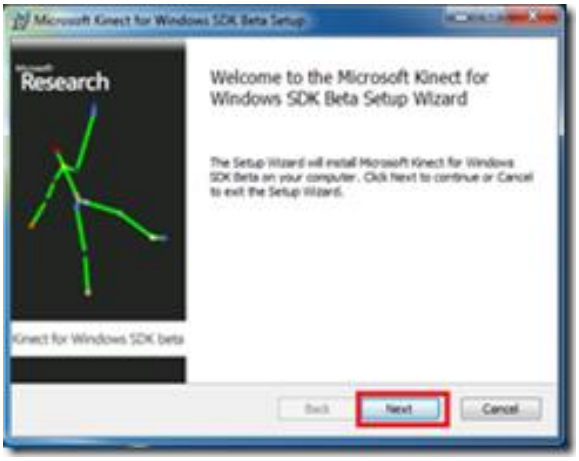

Figura 1A.- Pantalla de inicio del instalador

Después de pasar la pantalla de inicio de la instalación, donde se informa que se va a instalar el "SDK de Kinect para Windows Beta", aparece un mensaje avisando que ciertos ejemplos que vienen con el SDK necesitan el último directX SDK. Se siguen los pasos del asistente para instalar correctamente el SDK.

Una vez instalado el SDK, se procede a conectar el sensor Kinect al ordenador.

Se toma la fuente de alimentación y se conecta a la corriente.

El siguiente paso es conectar el otro extremo del cable (con el *LED* en verde) de la fuente de alimentación al sensor por medio del cable USB.

Por último se conecta el cable USB que sale de la fuente de alimentación a un puerto USB libre del ordenador.

Si está conectado correctamente tiene que parpadear el *LED* verde del sensor.

Figura 2A.- Pasos para la conexión del sensor Kinect al puerto de nuestra computadora

Cuando se conecta aparece en pantalla un mensaje informando que se está instalando un driver de un dispositivo nuevo.

Figura 3A.- Mensaje de instalador del driver

Si se hace clic se puede ver más información sobre el proceso. Se muestra el estado de la instalación de los driver para utilizar el dispositivo, el control de audio, la cámara y el control del array de los micrófonos. Cuando se acaba de instalar los diferentes drivers ya esta listo para usarse el sensor.

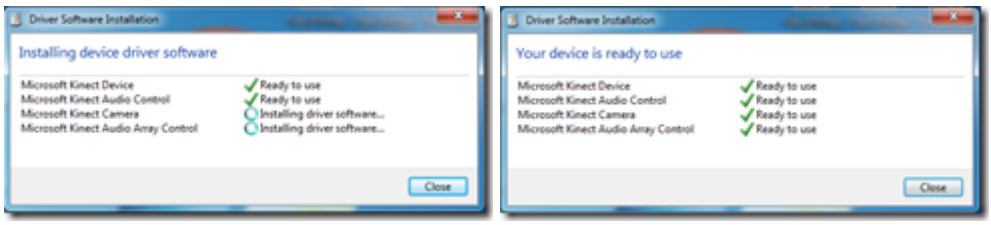

Figura 4A.- Pantallas de informe de instalación de los drivers

Si se revisa en el "Panel de control -> Sonido y hardware -> Administrador de dispositivos", aparece el sensor instalado:

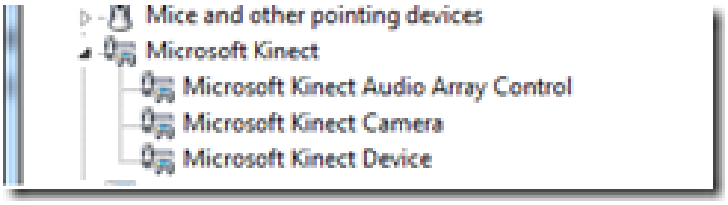

Figura 5A.- Se muestra el sensor instalado

Para asegurarse que funciona, dirijase a "C:\Program Files\Microsoft Research KinectSDK" o la ruta que se haya elegido para la instalación del SDK y ejecute el programa *Skeletal Viewer* o *Shape Game* que son 2 programas de ejemplo que se instalan con el SDK.

Ahora se procede a crear un proyecto para iniciar la programación con el SDK de Kinect para Windows y que servirá como proyecto base para las aplicaciones.

Una vez abierto Visual Studio 2010 se crea un proyecto WPF (Windows Presentation Foundation). Se le da un nombre y aceptar.

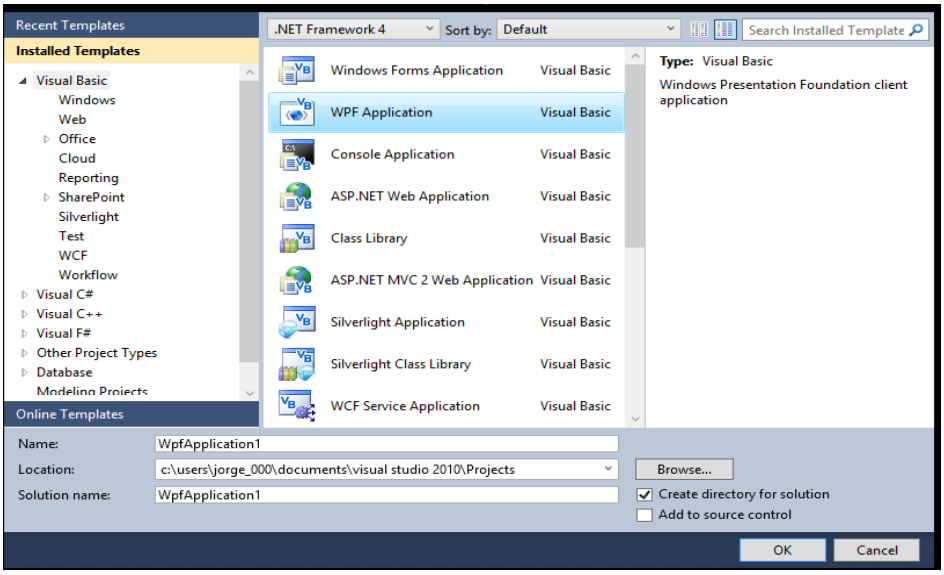

Figura 6A.- Creación de un nuevo proyecto WPF en Visual Studio 2010

Cuando finalice el proceso de creación y preparación de la solución se abrirá el panel de diseño con la ventana principal (MainWindows.xaml) de la aplicación y el explorador de proyectos a la derecha. Se despliega el nodo de MainWindow.xaml para acceder al fichero MainWindow.xaml.cs (el code-behind de la ventana) para empezar un programa en él.

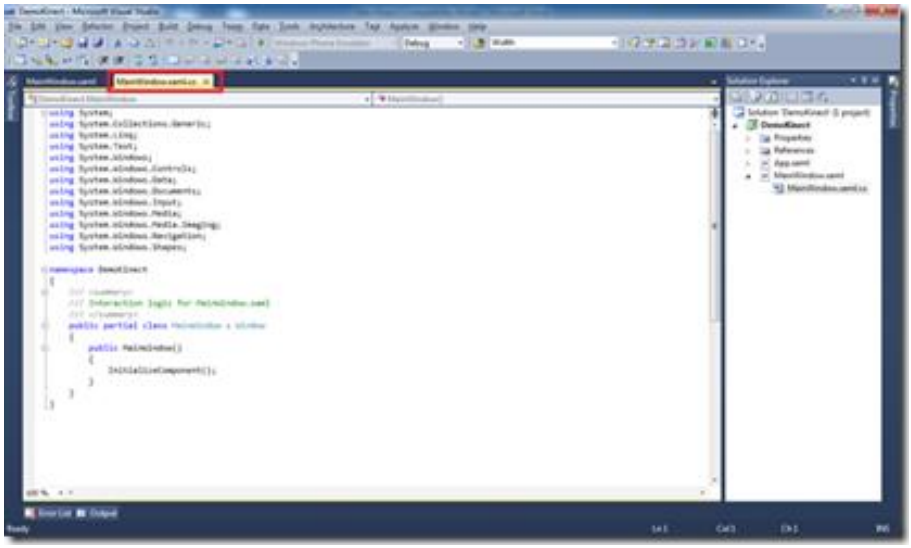

Figura 7A.- Ventana de MainWindow.xaml.cs

Para poder programar con el SDK de Kinect se debe agregar una referencia al proyecto. Presione con el botón derecho del ratón encima de "References" elija "añadir referencia". Aparecerá un cuadro de diálogo en el que se selecciona la referencia al SDK "Microsoft.Research.Kinect" y aceptar. Después se muestra cómo se ha agregado una nueva referencia al proyecto que corresponde al SDK.

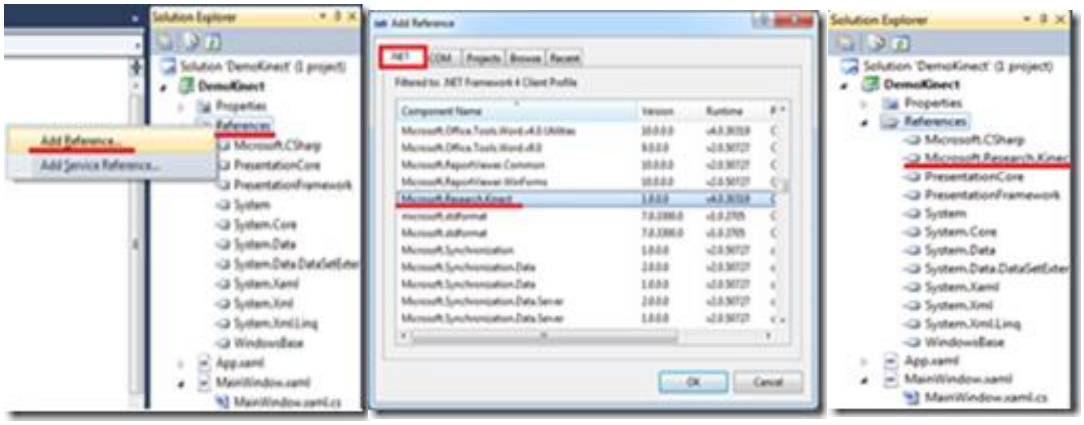

Figura 8A.- Pasos para agregar la Referencia a Microsoft Kinect

Con la referencia agregada se añade el espacio de nombres junto con los demás "usings" del fichero para poder utilizar las clases y métodos del SDK.

```
using Microsoft.Research.Kinect.Nui;
```

Dentro del constructor (método MainWindow()) de la ventana, se añade el evento Loaded, donde se introduce el código para inicializar el sensor, y el evento Closed para finalizar el uso. Esto creará, presionando la tecla "Tab", los 2 métodos (MainWindow_Loaded y MainWindow_Closed) que capturarán los eventos creados y se ejecutará el código que se escriba en ellos cuando estos eventos se produzcan.

```
public MainWindow()

{

InitializeComponent();

Loaded += new RoutedEventHandler(MainWindow_Loaded);

Closed += new EventHandler(MainWindow_Closed);

}
```

Se crea un atributo de la clase Runtime, clase que hace referencia al sensor Kinect y que se inicializará más adelante.

```
Runtime kinect;
```

Dentro del método MainWindow_Loaded se crea la instancia para controlar e inicializar el sensor con las opciones que se quieren usar. Se tienen 4 opciones para especificar el uso de la cámara RGB, de la de profundidad y del Skeletal Tracking.

La cámara de profundidad se puede usar con la opción de sólo profundidad o profundidad con índice de jugador. Si pone la segunda opción los datos de la cámara de profundidad contendrán también un índice para saber si los datos corresponden a un jugador u otro o a ninguno. Como ejemplo se inicializa con 3 opciones. Color, profundidad con índice de jugador y Skeletal Tracking.

```
void MainWindow_Loaded(object sender, RoutedEventArgs e){

kinect = new Runtime();

kinect.Initialize(RuntimeOptions.UseColor |
RuntimeOptions.UseDepthAndPlayerIndex |
RuntimeOptions.UseSkeletalTracking);

}
```

Por último en el método MainWindow_Closed se finaliza el uso del sensor. Cuando se cierre la aplicación ya estará disponible.

```
void MainWindow_Closed(object sender, EventArgs e)
{
kinect.Uninitialize();
}
```

Con esto se tiene la estructura del proyecto preparada para empezar a utilizar las funcionalidades que ofrece el SDK en una aplicación.

ANEXO B

CÓDIGO EN C# PARA DETECCIÓN DE PUNTOS EN EL CUERPO HUMANO

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Media;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;
using Coding4Fun.Kinect;
using System.IO;
using System.IO.Ports;
using System.ComponentModel;
using SkeletalTracking.Properties;
using Microsoft.Speech.AudioFormat;
using Microsoft.Speech.Recognition;


namespace SkeletalTracking
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        KinectSensor _sensor;
        SerialPort port = new SerialPort("COM3", 9600, Parity.None, 8,
StopBits.One);
        SpeechRecognitionEngine speechengine;


        /*
        public MainWindow()
        {
            InitializeComponent();
        }
        */


        public void SkeletalTracking()
        {
            InitializeComponent();
        }
```

```
        bool closing = false;
        const int skeletonCount = 6;
        Skeleton[] allSkeletons = new Skeleton[skeletonCount];

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            //this.squeezeSound.Stream = Properties.Resources.Squeeze;
            //En esta linea se indica que cada vez que el estado del dispositivo
cambie se mandara llamar al evento KinectSensors_StatusChanged
            //KinectSensor Es un eventualizador para el Audio
            //KinectSensorChooser1 Es un eventualizador para el Esqueleto
            kinectSensorChooser1.KinectSensorChanged += new
DependencyPropertyChangedEventHandler(kinectSensorChooser1_KinectSensorChanged);
            KinectSensor.KinectSensors.StatusChanged += new
EventHandler<StatusChangedEventArgs>(KinectSensors_StatusChanged);

            //Este metodo conecta activa lo que hara es asignar el primer
dispositivo encontrado a nuestra variable _sensor,ademas de inicializarlo e
iniciar el reconocimiento de voz

            //kinectSensorChooser1.KinectSensorChanged += new
DependencyPropertyChangedEventHandler(kinectSensorChooser1_KinectSensorChanged);

        }

        void kinectSensorChooser1_KinectSensorChanged(object sender,
DependencyPropertyChangedEventArgs e)
        {
            KinectSensor old = (KinectSensor)e.OldValue;

            StopKinect(old);

            KinectSensor sensor = (KinectSensor)e.NewValue;

            if (sensor == null)
            {
                return;
            }

            var parameters = new TransformSmoothParameters
            {
                Smoothing = 0.3f,
                Correction = 0.0f,
                Prediction = 0.0f,
                JitterRadius = 1.0f,
                MaxDeviationRadius = 0.5f
            };

            sensor.SkeletonStream.Enable(parameters);

            sensor.SkeletonStream.Enable();

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

sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
            conectaActiva();
```

```
            try
            {
                sensor.Start();

                port.Close();
                port.Open();
            }

            catch (System.IO.IOException)
            {
                kinectSensorChooser1.AppConflictOccurred();
            }
        }

void sensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)
        {
            if (closing)
            {
                return;
            }

            //Get a skeleton
            Skeleton first = GetFirstSkeleton(e);

            if (first == null)
            {
                return;
            }

            GetCameraPoint(first, e);

            //set scaled position
            ScalePosition(Head, first.Joints[JointType.Head]);
            ScalePosition(LeftHand, first.Joints[JointType.HandLeft]);
            //ScalePosition(RightHand, first.Joints[JointType.HandRight]);
            ScalePosition(CodoIzquierdo, first.Joints[JointType.ElbowLeft]);
            ScalePosition(CodoDerecho, first.Joints[JointType.ElbowRight]);
            ScalePosition(HombroIzquierdo, first.Joints[JointType.ShoulderLeft]);
            ScalePosition(HombroDerecho, first.Joints[JointType.ShoulderRight]);
            ScalePosition(Cintura, first.Joints[JointType.HipCenter]);
            ScalePosition(RodillaDerecha, first.Joints[JointType.KneeRight]);
            ScalePosition(RodillaIzquierda, first.Joints[JointType.KneeLeft]);
            ScalePosition2(BD, first.Joints[JointType.HandRight]);

            ProcessGesture(first.Joints[JointType.Head],
first.Joints[JointType.HandLeft], first.Joints[JointType.HandRight],
first.Joints[JointType.ElbowLeft], first.Joints[JointType.ElbowRight],
first.Joints[JointType.ShoulderLeft], first.Joints[JointType.ShoulderRight],
first.Joints[JointType.HipCenter], first.Joints[JointType.KneeRight],
first.Joints[JointType.KneeLeft]);

        }

        Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)
        {
            using (SkeletonFrame skeletonFrameData = e.OpenSkeletonFrame())
            {
                if (skeletonFrameData == null)
```

```
                {
                    return null;
                }


                skeletonFrameData.CopySkeletonDataTo(allSkeletons);

                //get the first tracked skeleton
                Skeleton first = (from s in allSkeletons
                                  where s.TrackingState ==
SkeletonTrackingState.Tracked
                                  select s).FirstOrDefault();

                return first;



            }
        }
```

ANEXO C

CÓDIGO PARA LA DETECCIÓN DE GESTOS

```
private void ProcessGesture(Joint head, Joint handleft, Joint handright, Joint
elbowleft, Joint elbowright, Joint shoulderleft, Joint shoulderright, Joint
hipcenter, Joint kneeright, Joint kneeleft)
        {
            port.Write("D");
            if (handright.Position.Y > 0.6 && handleft.Position.Y <
elbowleft.Position.Y && handleft.Position.X > shoulderleft.Position.X &&
handleft.Position.X < shoulderright.Position.X)
            {
                textBox4.Text = "MÁX.";
                port.Write("b");
                port.Write("S");
                textBox1.Text = "RETROCEDE";
            }else
                if (handright.Position.Y > 0.6 && handleft.Position.Y >
shoulderleft.Position.Y && handleft.Position.X > shoulderleft.Position.X &&
handleft.Position.X < shoulderright.Position.X)
            {
                textBox4.Text = "MÁX.";
                port.Write("b");
                port.Write("N");
                textBox1.Text = "AVANZA";
            }else
                    if (handright.Position.Y > 0.6 && handleft.Position.Y <
shoulderleft.Position.Y && handleft.Position.Y > hipcenter.Position.Y &&
handleft.Position.X < hipcenter.Position.X)
            {
                textBox4.Text = "MÁX.";
                port.Write("b");
                port.Write("O");
                textBox1.Text = "IZQUIERDA";
            }else
                        if (handright.Position.Y > 0.6 && handleft.Position.Y <
shoulderleft.Position.Y && handleft.Position.Y > hipcenter.Position.Y &&
handleft.Position.X > hipcenter.Position.X)
            {
                textBox4.Text = "MÁX.";
                port.Write("b");
                port.Write("E");
                textBox1.Text = "DERECHA";
            }
            else
                if (handright.Position.Y < 0.6 && handright.Position.Y > 0.5 &&
handleft.Position.Y < elbowleft.Position.Y && handleft.Position.X >
shoulderleft.Position.X && handleft.Position.X < shoulderright.Position.X)
                //if (handright.Position.Y > 0.05)
                {
                    textBox4.Text = "90";
                    port.Write("b");
                    port.Write("S");
                    textBox1.Text = "RETROCEDE";
                }else
                    if (handright.Position.Y < 0.6 && handright.Position.Y > 0.5
&& handleft.Position.Y > shoulderleft.Position.Y && handleft.Position.X >
shoulderleft.Position.X && handleft.Position.X < shoulderright.Position.X)
```

```
                    //if (handright.Position.Y > 0.05)
                {
                textBox4.Text = "90";
                port.Write("b");
                port.Write("N");
                textBox1.Text = "AVANZA";
                }else
                        if (handright.Position.Y < 0.6 && handright.Position.Y >
0.5 && handleft.Position.Y < shoulderleft.Position.Y && handleft.Position.Y >
hipcenter.Position.Y && handleft.Position.X < hipcenter.Position.X)
                    //if (handright.Position.Y > 0.05)
                {
                textBox4.Text = "90";
                port.Write("b");
                port.Write("O");
                textBox1.Text = "IZQUIERDA";
                }else
                            if (handright.Position.Y < 0.6 &&
handright.Position.Y > 0.5 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X >
hipcenter.Position.X)
                    //if (handright.Position.Y > 0.05)
                {
                textBox4.Text = "90";
                //port.Write("b");
                port.Write("E");
                textBox1.Text = "DERECHA";
                }else

                    if (handright.Position.Y < 0.5 && handright.Position.Y > 0.4
&& handleft.Position.Y < elbowleft.Position.Y && handleft.Position.X >
shoulderleft.Position.X && handleft.Position.X < shoulderright.Position.X)
                    {
                        textBox4.Text = "80";
                        port.Write("c");
                        port.Write("S");
                        textBox1.Text = "RETROCEDE";
                    }
                    else
                        if (handright.Position.Y < 0.5 && handright.Position.Y >
0.4 && handleft.Position.Y > shoulderleft.Position.Y && handleft.Position.X >
shoulderleft.Position.X && handleft.Position.X < shoulderright.Position.X)
                    {
                        textBox4.Text = "80";
                        port.Write("c");
                        port.Write("N");
                        textBox1.Text = "AVANZA";
                    }
                    else
                            if (handright.Position.Y < 0.5 &&
handright.Position.Y > 0.4 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X <
hipcenter.Position.X)
                    {
                        textBox4.Text = "80";
                        port.Write("c");
                        port.Write("O");
                        textBox1.Text = "IZQUIERDA";
                    }
```

```
                    else
                                if (handright.Position.Y < 0.5 &&
handright.Position.Y > 0.4 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X >
hipcenter.Position.X)
                    {
                        textBox4.Text = "80";
                        port.Write("c");
                        port.Write("E");
                        textBox1.Text = "DERECHA";
                    }
                    else
                     if (handright.Position.Y < 0.4 && handright.Position.Y > 0.3
&& handleft.Position.Y < elbowleft.Position.Y && handleft.Position.X >
shoulderleft.Position.X && handleft.Position.X < shoulderright.Position.X)
                        {
                            textBox4.Text = "70";
                            port.Write("d");
                            port.Write("S");
                            textBox1.Text = "RETROCEDE";
                        }
                        else
                         if (handright.Position.Y < 0.4 && handright.Position.Y >
0.3 && handleft.Position.Y > shoulderleft.Position.Y && handleft.Position.X >
shoulderleft.Position.X && handleft.Position.X < shoulderright.Position.X)
                        {
                            textBox4.Text = "70";
                            port.Write("d");
                            port.Write("N");
                            textBox1.Text = "AVANZA";
                        }
                        else
                             if (handright.Position.Y < 0.4 &&
handright.Position.Y > 0.3 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X <
hipcenter.Position.X)
                        {
                            textBox4.Text = "70";
                            port.Write("d");
                            port.Write("O");
                            textBox1.Text = "IZQUIERDA";
                        }
                        else
                                 if (handright.Position.Y < 0.4 &&
handright.Position.Y > 0.3 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X >
hipcenter.Position.X)
                        {
                            textBox4.Text = "70";
                            port.Write("d");
                            port.Write("E");
                            textBox1.Text = "DERECHA";
                        }
                        else
                         if (handright.Position.Y < 0.3 && handright.Position.Y >
0.2 && handleft.Position.Y < elbowleft.Position.Y && handleft.Position.X >
shoulderleft.Position.X && handleft.Position.X < shoulderright.Position.X)
                            {
                                textBox4.Text = "60";
```

```
                                port.Write("e");
                                port.Write("S");
                                textBox1.Text = "RETROCEDE";
                            }
                            else
                             if (handright.Position.Y < 0.3 &&
handright.Position.Y > 0.2 && handleft.Position.Y > shoulderleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                            {
                                textBox4.Text = "60";
                                port.Write("e");
                                port.Write("N");
                                textBox1.Text = "AVANZA";
                            }
                            else
                                 if (handright.Position.Y < 0.3 &&
handright.Position.Y > 0.2 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X <
hipcenter.Position.X)
                            {
                                textBox4.Text = "60";
                                port.Write("e");
                                port.Write("O");
                                textBox1.Text = "IZQUIERDA";
                            }
                            else
                                    if (handright.Position.Y < 0.3 &&
handright.Position.Y > 0.2 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X >
hipcenter.Position.X)
                            {
                                textBox4.Text = "60";
                                port.Write("e");
                                port.Write("E");
                                textBox1.Text = "DERECHA";
                            }
                            else
                             if (handright.Position.Y < 0.2 &&
handright.Position.Y > 0.1 && handleft.Position.Y < elbowleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                                {
                                    textBox4.Text = "50";
                                    port.Write("f");
                                    port.Write("S");
                                    textBox1.Text = "RETROCEDE";
                                }
                                else
                                 if (handright.Position.Y < 0.2 &&
handright.Position.Y > 0.1 && handleft.Position.Y > shoulderleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                                {
                                    textBox4.Text = "50";
                                    port.Write("f");
                                    port.Write("N");
                                    textBox1.Text = "AVANZA";
                                }
```

```
                                else
                                     if (handright.Position.Y < 0.2 &&
handright.Position.Y > 0.1 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X <
hipcenter.Position.X)
                                {
                                    textBox4.Text = "50";
                                    port.Write("f");
                                    port.Write("O");
                                    textBox1.Text = "IZQUIERDA";
                                }
                                else
                                         if (handright.Position.Y < 0.2 &&
handright.Position.Y > 0.1 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X >
hipcenter.Position.X)
                                {
                                    textBox4.Text = "50";
                                    port.Write("f");
                                    port.Write("E");
                                    textBox1.Text = "DERECHA";
                                }
                                else
                                 if (handright.Position.Y < 0.1 &&
handright.Position.Y > 0 && handleft.Position.Y < elbowleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                                    {
                                        textBox4.Text = "40";
                                        port.Write("g");
                                        port.Write("S");
                                        textBox1.Text = "RETROCEDE";
                                    }
                                else
                                     if (handright.Position.Y < 0.1 &&
handright.Position.Y > 0 && handleft.Position.Y > shoulderleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                                    {
                                        textBox4.Text = "40";
                                        port.Write("g");
                                        port.Write("N");
                                        textBox1.Text = "AVANZA";
                                    }
                                else
                                         if (handright.Position.Y < 0.1 &&
handright.Position.Y > 0 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X <
hipcenter.Position.X)
                                    {
                                        textBox4.Text = "40";
                                        port.Write("g");
                                        port.Write("O");
                                        textBox1.Text = "IZQUIERDA";
                                    }
                                else
                                            if (handright.Position.Y < 0.1 &&
handright.Position.Y > 0 && handleft.Position.Y < shoulderleft.Position.Y &&
```

```
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X >
hipcenter.Position.X)
                                    {
                                        textBox4.Text = "40";
                                        port.Write("g");
                                        port.Write("E");
                                        textBox1.Text = "DERECHA";
                                    }
                                else
                                     if (handright.Position.Y < 0 &&
handright.Position.Y > -0.1 && handleft.Position.Y < elbowleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                                    {
                                        textBox4.Text = "30";
                                        port.Write("h");
                                        port.Write("S");
                                        textBox1.Text = "RETROCEDE";
                                    }
                                else
                                      if (handright.Position.Y < 0 &&
handright.Position.Y > -0.1 && handleft.Position.Y > shoulderleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                                    {
                                        textBox4.Text = "30";
                                        port.Write("h");
                                        port.Write("N");
                                        textBox1.Text = "AVANZA";
                                    }
                                else
                                       if (handright.Position.Y < 0 &&
handright.Position.Y > -0.1 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X <
hipcenter.Position.X)
                                    {
                                        textBox4.Text = "30";
                                        port.Write("h");
                                        port.Write("O");
                                        textBox1.Text = "IZQUIERDA";
                                    }
                                else
                                          if (handright.Position.Y < 0 &&
handright.Position.Y > -0.1 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y >hipcenter.Position.Y && handleft.Position.X >
hipcenter.Position.X)
                                    {
                                        textBox4.Text = "30";
                                        port.Write("h");
                                        port.Write("E");
                                        textBox1.Text = "DERECHA";
                                    }
                                else
                                    if (handright.Position.Y < -0.1 &&
handright.Position.Y > -0.2 && handleft.Position.Y < elbowleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                                        {
                                            textBox4.Text = "20";
```

```
                                        port.Write("i");
                                        port.Write("S");
                                        textBox1.Text = "RETROCEDE";
                                    }
                                else
                                    if (handright.Position.Y < -0.1 &&
handright.Position.Y > -0.2 && handleft.Position.Y > shoulderleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                                    {
                                        textBox4.Text = "20";
                                        port.Write("i");
                                        port.Write("N");
                                        textBox1.Text = "AVANZA";
                                    }
                                else
                                    if (handright.Position.Y < -0.1 &&
handright.Position.Y > -0.2 && handleft.Position.Y < shoulderleft.Position.Y &&
handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X <
hipcenter.Position.X)
                                    {
                                        textBox4.Text = "20";
                                        port.Write("i");
                                        port.Write("O");
                                        textBox1.Text = "IZQUIERDA";
                                    }
                                else
                                    if (handright.Position.Y < -
0.1 && handright.Position.Y > -0.2 && handleft.Position.Y <
shoulderleft.Position.Y && handleft.Position.Y > hipcenter.Position.Y &&
handleft.Position.X > hipcenter.Position.X)
                                    {
                                        textBox4.Text = "20";
                                        port.Write("i");
                                        port.Write("E");
                                        textBox1.Text = "DERECHA";
                                    }
                                else
                                    if (handright.Position.Y < -0.2 &&
handright.Position.Y > -0.3 && handleft.Position.Y < elbowleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                                    {
                                        textBox4.Text = "10";
                                        port.Write("j");
                                        port.Write("S");
                                        textBox1.Text = "RETROCEDE";
                                    }
                                    else
                                    if (handright.Position.Y < -0.2 &&
handright.Position.Y > -0.3 && handleft.Position.Y > shoulderleft.Position.Y &&
handleft.Position.X > shoulderleft.Position.X && handleft.Position.X <
shoulderright.Position.X)
                                    {
                                        textBox4.Text = "10";
                                        port.Write("j");
                                        port.Write("N");
                                        textBox1.Text = "AVANZA";
                                    }
```

```
                                                else

                                                if (handright.Position.Y < -0.2
&& handright.Position.Y > -0.3 && handleft.Position.Y < shoulderleft.Position.Y
&& handleft.Position.Y > hipcenter.Position.Y && handleft.Position.X <
hipcenter.Position.X)
                                                {
                                                    textBox4.Text = "10";
                                                    port.Write("j");
                                                    port.Write("O");
                                                    textBox1.Text = "IZQUIERDA";
                                                }
                                                else
                                                    if (handright.Position.Y < -
0.2 && handright.Position.Y > -0.3 && handleft.Position.Y <
shoulderleft.Position.Y && handleft.Position.Y > hipcenter.Position.Y &&
handleft.Position.X > hipcenter.Position.X)
                                                {
                                                    textBox4.Text = "10";
                                                     port.Write("j");
                                                     port.Write("E");
                                                    textBox1.Text = "DERECHA";
                                                }
                                                else
                                                 if (handright.Position.Y > -0.4)
                                                    {
                                                        textBox4.Text = "0";
                                                         port.Write("b");
                                                         port.Write("D");
                                                        textBox1.Text = "ALTO";
                                                    }



                                                        }

     void GetCameraPoint(Skeleton first, AllFramesReadyEventArgs e)
        {

            using (DepthImageFrame depth = e.OpenDepthImageFrame())
            {
                if (depth == null ||  kinectSensorChooser1.Kinect == null)
                {
                    return;
                }


                //Map a joint location to a point on the depth map
                //head
                DepthImagePoint headDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.Head].Position);
                //left hand
                DepthImagePoint leftDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HandLeft].Position);
                //right hand
                DepthImagePoint rightDepthPoint =
```

```
depth.MapFromSkeletonPoint(first.Joints[JointType.HandRight].Position);
                DepthImagePoint eleftDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.ElbowLeft].Position);
                DepthImagePoint elrightDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.ElbowRight].Position);
                DepthImagePoint sleftDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.ShoulderLeft].Position);
                DepthImagePoint srightDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.ShoulderRight].Position);
                DepthImagePoint cDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HipCenter].Position);
                DepthImagePoint krDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HipCenter].Position);
                DepthImagePoint klDepthPoint =

depth.MapFromSkeletonPoint(first.Joints[JointType.HipCenter].Position);

                //Map a depth point to a point on the color image
                //head
                ColorImagePoint headColorPoint =
                    depth.MapToColorImagePoint(headDepthPoint.X,
headDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //left hand
                ColorImagePoint leftColorPoint =
                    depth.MapToColorImagePoint(leftDepthPoint.X,
leftDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                //right hand

                ColorImagePoint elrightColorPoint =
                    depth.MapToColorImagePoint(elrightDepthPoint.X,
elrightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                ColorImagePoint eleftColorPoint =
                    depth.MapToColorImagePoint(eleftDepthPoint.X,
eleftDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                ColorImagePoint srightColorPoint =
                    depth.MapToColorImagePoint(srightDepthPoint.X,
srightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                ColorImagePoint sleftColorPoint =
                    depth.MapToColorImagePoint(sleftDepthPoint.X,
sleftDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                ColorImagePoint rightColorPoint =
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                ColorImagePoint cColorPoint =
```

```
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                ColorImagePoint krColorPoint =
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);
                ColorImagePoint klColorPoint =
                    depth.MapToColorImagePoint(rightDepthPoint.X,
rightDepthPoint.Y,
                    ColorImageFormat.RgbResolution640x480Fps30);

                //Set location
                CameraPosition(Head, headColorPoint);
                CameraPosition(LeftHand, leftColorPoint);
                //CameraPosition(BD, rightColorPoint);

                CameraPosition(CodoIzquierdo, eleftColorPoint);
                CameraPosition(CodoDerecho, elrightColorPoint);
                CameraPosition(HombroIzquierdo, sleftColorPoint);
                CameraPosition(HombroDerecho, srightColorPoint);
                CameraPosition(Cintura, cColorPoint);
                CameraPosition(RodillaDerecha, krColorPoint);
                CameraPosition(RodillaIzquierda, klColorPoint);

            }
        }
```

ANEXO D
ARTÍCULO PRESENTADO EN CONGRESO

EL 16
PON 72

"IMPLEMENTACIÓN DE UN CONTROL GESTUAL EN UN MÓVIL INALÁMBRICO"

David Vázquez Álvarez, Jafeth Ascensión Alonso Carreón, Jorge Alberto Cortés Rodríguez, Antonio Sandoval Francisco, Irving Ricardo Trujillo Pascual

Escuela Superior de Ingeniería Mecánica y Eléctrica, Instituto Politécnico Nacional, D.F., México.

dvazqueza@ipn.mx, jalonso@ipn.mx, jacortes1601@gmail.com, antoniosandovalfrancisco@hotmail.com, rick_thelostdreams@hotmail.com

RESUMEN

A través de la detección de gestos el usuario podrá controlar un móvil que se desplace en una dirección previamente determinada para cierto gesto.

INTRODUCCIÓN

El control es algo que el ser humano ha buscado a través del tiempo, para poder manipular objetos y que estos realicen tareas que resultan ser complicadas o que requieren un alto nivel de esfuerzo. Los sistemas que cuentan con un control a distancia son muy comunes hoy en día y se siguen desarrollando nuevas técnicas para facilitar el control.

La tecnología de control gestual se puede aplicar en varias áreas para diferentes usos. Por lo que se considera una tecnología con futuro y expansión. Esto permite tener un control preciso sobre tareas complicadas, asegura que el objeto imite totalmente los movimientos del cuerpo para una mayor confiabilidad, también es una tecnología que permite que personas con alguna discapacidad, como la ausencia de las manos, puedan operar estos objetos puesto que no requiere el contacto físico, solo de los movimientos del cuerpo.

ROC&C'2013 / EL 16 PONENCIA RECOMENDADA POR EL COMITE DE ELECTRONICA DEL IEEE SECCION MEXICO Y PRESENTADA EN LA REUNION INTERNACIONAL DE OTOÑO, ROC&C'2013. ACAPULCO, GRO., DEL 10 AL 14 DE NOVIEMBRE DEL 2013.

Aunque aún se siguen utilizando, los controles remotos físicos, como los del televisor, tienen la desventaja de que pueden extraviarse, llenarse de grasa o sufrir golpes y dejar de funcionar. Después surgieron los dispositivos con pantalla táctil, pero aunque son muy populares hoy en día, también tienen las desventajas de que, con el paso del tiempo, la pantalla queda manchada con las marcas de los dedos, también hay programas que no soportan esta tecnología y quedan desconfigurados cuando alguien quiere utilizarlos en modo táctil. Con el control gestual se evitan todos estos inconvenientes, solamente se necesita de una cámara o un sensor y el usuario que operará el equipo. La persona es el control que se necesita para hacer funcionar el programa.

DESARROLLO

El equipo y software utilizado en este proyecto fueron: sensor Kinect, Microsoft Visual Studio, SDK Kinect para Windows, Digi XBee y ARDUINO.

La cámara del sensor Kinect ya cuenta con la electrónica necesaria para lo que se desea realizar, que son las tres cámaras en su parte frontal; una infrarroja, de RGB y de profundidad, tal y como se muestra en la figura 1. Gracias a esto, sólo es necesario instalar los controladores en una computadora para que reconozca el sensor Kinect.

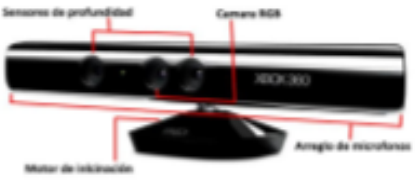

Figura 1.- Composición del sensor Kinect

El software de Visual Studio es recomendado ya que es distribuido por Windows, además de ser solicitado cuando descargamos el Kit de Desarrollo de Software (SDK) del Kinect. El SDK incluye las librerías necesarias para la programación del sensor Kinect. Estas librerías son aplicadas en este proyecto en Visual Studio y ahora es posible configurar los comandos que se desean realizar.

El código del programa ejecuta la función "Skeletal Tracking", esta función es la que reconoce veinte puntos en el cuerpo del usuario (Figura 2). En base a estos puntos es que también será capaz de reconocer los gestos conociendo la posición en la que se encuentran. Las librerías del SDK Kinect ya cuentan con los comandos para este proceso.

Se utiliza la versión 1.7 del SDK de Kinect. Esta versión incluye soporte para programar en lenguaje C#. Se eligió este lenguaje porque es orientado a objetos y es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

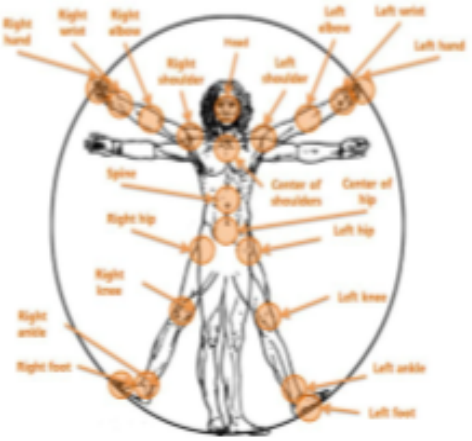

Figura 2.- Detección de 20 puntos en el cuerpo

Otra función importante en el código es "Process Gesture". Esta función realiza el reconocimiento de gestos a partir de una orientación cartesiana en los ejes "X" e "Y". Una vez que reconoce los puntos del cuerpo humano, se dicta una condición sobre esos puntos. Por ejemplo, se le indica que si el punto correspondiente a la mano se encuentra en un lugar mayor al punto correspondiente a la cabeza, entonces ejecute algún comando. En la figura 3 se muestra la forma en que el sensor Kinect detecta los gestos del cuerpo.

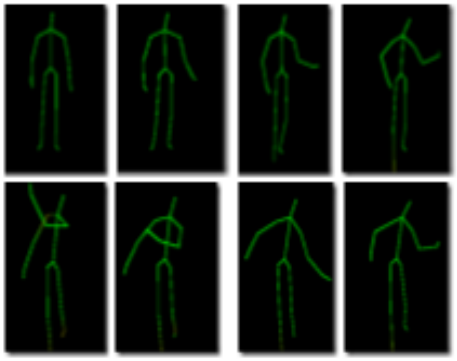

Figura 3.- Reconocimientos de gestos con sensor Kinect

Para entender mejor cómo funciona el reconocimiento del gesto se debe saber que el ordenador lee las posiciones de izquierda a derecha, es decir, un punto que se desplaza hacia la derecha se dice que va aumentando, y un punto desplazado hacia la izquierda disminuye. De igual manera en el eje Y se lee en dirección ascendente.

El programa se basa sólo en ciertos puntos del cuerpo y no en los veinte que incluye el SDK. Con esos puntos se hizo una asignación dependiendo de la posición correspondiente, basados en un tablero con doce casillas. Cada casilla corresponde a una acción, y también se agregan puntos de descanso en donde no se efectúa ninguna acción. Las acciones se efectuarán dependiendo de la casilla en donde se coloque la mano izquierda. Esto se puede visualizar en la figura 4.

Figura 4.- Representación del comandos dependiendo la posición de la mano izquierda

Con los comandos programados y el reconocimiento funcionando se pasa a transmitir la señal al receptor que efectuará las tareas. En este proyecto se construyó un pequeño vehículo móvil de cuatro llantas y se agregó un reconocimiento de gesto a la mano derecha. Este sólo servirá para ordenar al móvil que se detenga e ignore las demás órdenes que se le envíen con la mano izquierda. (Figura 5)

Figura 5.- Comando efectuados para los posición de la mano derecha

La conexión se realizó de forma inalámbrica gracias a los sensores Xbee. Los sensores utilizados tienen un alcance interior de 30m. y 100m. en el exterior. Esta señal transmitida llega al circuito eléctrico plataforma ARDUINO, el cual es el corazón del proyecto, al ser este el encargado de interpretar las instrucciones enviadas desde la computadora y traducirlas al móvil para que realice las acciones debidas.

ARDUINO se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data).

En la plataforma ARDUINO se le asignan parámetros para la ejecución de los comandos. Primero recibe la señal de puerto serial COM6, en este puerto es conectado en uno de los módulos de recepción inalámbrica Xbee, el cual transmitirá al segundo módulo colocado en un circuito conectado al móvil. Este proceso se representa en la figura 6.

Figura 6.- Visualización de plataforma ARDUINO

Al llegar la señal al segundo Xbee, este la pasa al circuito de ARDUINO que ejecutará la traducción al móvil. La traducción está basada en cuatro órdenes representadas con los números del diez al trece. Cada número corresponde a una dirección en la que el móvil se desplazará.

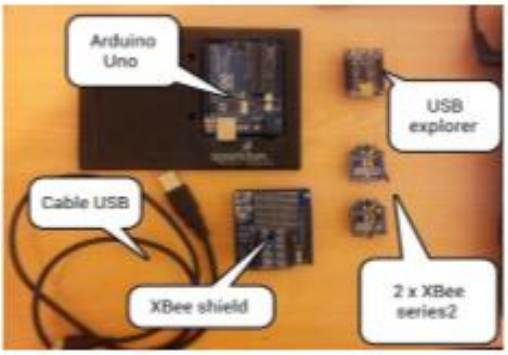

Figura 7.- Equipo utilizado para transmitir comandos de forma inalámbrica

Finalmente procedemos al ensamblado de los componentes. Se tiene el programa en Visual Studio con la captura de los puntos y el reconocimiento de los gestos, los sensores Xbee que mandas la información, y la plataforma de ARDUINO que ejecutará los comandos. Ahora se agrega un último circuito compuesto por cuatro relevadores que funcionará como puente H. (Figura 8)

Figura 8.- Circuito con 4 relevadores para el cambio de dirección

El circuito de la Figura 8 muestra los relevadores encargados de cambiar la ruta en que se desplazará el móvil (Adelante, Atrás, Izquierda o Derecha). Estos reciben la señal de los sensores y efectúan los cambios necesarios para que el móvil avance correctamente o, en todo caso, se detenga.

Figura 9.- Prototipo móvil terminado

CONCLUSIONES

El desarrollo de esta aplicación fue sencilla gracias al apoyo de las librerías del SDK de Kinect, que ayudaron detectando los puntos del cuerpo humano por medio de los sensores, lo cual facilitó el desarrollo del software para la aplicación.

Al proyecto se le pueden anexar muchas mejoras para que realice movimientos con mayor detalle y precisión.

Se deja este desarrollo con la intención de que puedan desarrollarse nuevas aplicaciones y que sirva para resolver problemas, como siempre ha sido la intención de la ingeniería.

BIBLIOGRAFÍA

[1] Kinect for Windows. 2013. http://www.microsoft.com/en-us/kinectforwindows/Discover/Features.aspx

[2] Microsoft en Español. 2013. http://blogs.technet.com/b/microsoftlatam/archive/2011/06/24/conoce-m-225-s-sobre-el-sdk-de-kinect-para-windows.aspx

[3] Comenzando a desarrollar aplicaciones con Kinect. 2013. http://blanchardspace.wordpress.com/2013/03/23/comenzando-a-desarrollar-aplicaciones-con-kinect/

[4] Plantilla para crear programas con interfaz gráfica - Kinect SDK. 2013.

http://www.zaerpo.com/2012/10/plantilla-para-crear-programas-con.html

[5] [KINECT] Howto: "Hola Mundo" con C# y Kinect. 2013. http://geeks.ms/blogs/elbruno/archive/2010/12/31/kinect-howto-hola-mundo-con-c-y-kinect.aspx

[6] Reto SDK Kinect. Detectar posturas con Skeletal tracking. 2013. http://blogs.msdn.com/b/esmsdn/archive/2011/08/09/reto-sdk-de-kinect-detectar-poses-con-skeletal-tracking.aspx

[7] Working with Kinect Studio. 2013. http://msdn.microsoft.com/en-us/magazine/jj650892.aspx

[8] ARDUINO – Home Page. 2013 http://www.arduino.cc/es/

[9] Xbee – Modulos de Transmición Inalámbrica. 2013. http://www.xbee.cl/

CURRICULUM

M. en C. David Vázquez Álvarez.

Maestro en Ciencias en Ingeniería de Telecomunicaciones de la Sección de Estudios de Posgrado e Investigación (SEPI) del Instituto Politécnico Nacional (IPN). Actualmente es Subdirector Administrativo de la Escuela Superior de Ingeniería Mecánica y Eléctrica (ESIME) unidad Profesional Adolfo López Mateos (Zacatenco). Áreas de Interés Actual: Fibras Ópticas, Radio Definido por Software, Radio Cognitivo, Sistemas de Comunicaciones Móviles e Inalámbricas.

M. en C. Jafeth Ascensión Alonso Carreón.

Maestro en Ciencias en Ingeniería de Telecomunicaciones de la Sección de Estudios de Posgrado e Investigación (SEPI) del Instituto Politécnico Nacional (IPN). Actualmente es Presidente de la especialidad de Computación en la Escuela Superior de Ingeniería Mecánica y Eléctrica (ESIME) unidad Profesional Adolfo López Mateos (Zacatenco). Áreas de Interés Actual: Reconocimiento e interpretación de imágenes y fotografía.

Jorge Alberto Cortés Rodríguez

Actualmente es alumno de 9° semestre en la especialidad de computación de la carrera de Ingeniería en Comunicaciones y Electrónica de la Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Profesional Adolfo López Mateos impartida en el Instituto Politécnico Nacional (IPN). Áreas de interés: Aplicación de Redes, Programación, Sistemas de Comunicación.

Antonio Sandoval Francisco

Actualmente es alumno de 9° semestre en la especialidad de computación de la carrera de Ingeniería en Comunicaciones y Electrónica de la Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Profesional Adolfo López Mateos impartida en el Instituto Politécnico Nacional (IPN). Áreas de interés: Sistemas en tiempo real, Agentes inteligentes expertos, Redes.

Trujillo Pascual Irving Ricardo

Actualmente es alumno de 9° semestre en la especialidad de computación de la carrera de Ingeniería en Comunicaciones y Electrónica de la Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Profesional Adolfo López Mateos impartida en el Instituto Politécnico Nacional (IPN). Áreas de interés: Redes, Desarrollo de aplicaciones móviles, Telecomunicaciones

REFERENCIAS

[1] El mando a distancia. 2013. http://www.xataka.com/historias-de-la-tecnologia/el-mando-a-distancia-historia-de-una-desaparicion

[2] El control Gestual será la siguiente revolución informática. 2013. http://www.eleconomistaamerica.mx/ahorro-energetico/noticias/4954515/06/13/El-control-gestual-sera-la-siguiente-evolucion-informatica.html

[3] Después de la pantalla tactil, el control gestual. 2013. http://www.consumer.es/web/es/tecnologia/software/2013/08/26/217464.php

[4] Sensores de Movimiento – Material Eléctrico | Voltimum ES. 2013. http://www.voltimum.es/news/8080/cm/sensores-de-movimiento.html

[5] Sensores de Movimiento. 2013. http://sensmovimiento.blogspot.mx/p/caracteristicas-de-un-sensor.html

[6] ¿Cómo funciona un detector de movimiento? 2013. http://intrepido1.over-blog.es/article-como-funciona-detector-movimiento-85924119.html

[7] SDK de Kinect: Historia de un éxito accidental de Microsoft. 2013. http://www.pcworld.com.mx/Articulos/13303.htm

[8] SG. Hola Mundo Kinect. 2013. http://sg.com.mx/revista/hola-mundo-kinect

[9] Funcionamiento del sensor de movimiento Kinect. 2013. http://ideasgeek.net/2010/11/10/funcionamiento-del-sensor-de-movimiento-en-kinect/

[10] Kinect for Windows. 2013. http://www.microsoft.com/en-us/kinectforwindows/Discover/Features.aspx

[11] Kinect for Windows. 2013. http://www.microsoft.com/en-us/kinectforwindows/develop/new.aspx

[12] Microsoft en Español. 2013. http://blogs.technet.com/b/microsoftlatam/archive/2011/06/24/cionoce-m-225-s-sobre-el-sdk-de-kinect-para-windows.aspx

[13] El nuevo SDK de Kinect para Windows. 2013. http://www.elotrolado.net/noticia_el-nuevo-sdk-del-kinect-para-windows-permite-realizar-escaners-3d_21753

[14] Hola Mundo con Kinect SDK. 2013. http://robertoluis.wordpress.com/2011/12/21/hola-mundo-con-kinect-sdk/

[15] Comenzando a desarrollar aplicaciones con Kinect. 2013. http://blanchardspace.wordpress.com/2013/03/23/comenzando-a-desarrollar-aplicaciones-con-kinect/

[16] Plantilla para crear programas con interfaz gráfica - Kinect SDK. 2013. http://www.raerpo.com/2012/10/plantilla-para-crear-programas-con.html

[17] [KINECT] Howto: "Hola Mundo" con C# y Kinect. 2013. http://geeks.ms/blogs/elbruno/archive/2010/12/31/kinect-howto-hola-mundo-con-c-y-kinect.aspx

[18] Reto SDK Kinect. Detectar posturas con Skeletal tracking. 2013. http://blogs.msdn.com/b/esmsdn/archive/2011/08/09/reto-sdk-de-kinect-detectar-poses-con-skeletal-tracking.aspx

[19] Working with Kinect Studio. 2013. http://msdn.microsoft.com/en-us/magazine/jj650892.aspx