SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Reconocimiento de Movimientos de Manos para la Manipulación de una Interfaz Computacional

presentada por

Ing. José Luis Molina Salgado

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dr. Máximo López Sánchez

Codirector de tesis

Dr. Juan Gabriel González Serna

Cuernavaca, Morelos, México. Junio de 2018.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

SEP

SECRETARIA DE EDUCACIÓN PUBLICA

TECNOLOGICO NACIONAL DE MEXICO

Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Morelos a 25 de junio del 2018
OFICIO No. DCC/204/2018

Asunto: Aceptación de documento de tesis

DR. GERARDO V. GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del **Ing. José Luis Molina Salgado**, con número de control M16CE009, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis profesional titulado "**Reconocimiento de movimientos de manos para la manipulación de una interfaz computacional"** y hemos encontrado que se han realizado todas las correcciones y observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

DIRECTOR DE TESIS

Dr. Máximo López Sánchez
Doctor en Ciencias de la Computación
7498547

CO-DIRECTOR DE TESIS

Dr. Juan Gabriel González Serna
Doctor en Ciencias de la Computación
7820329

REVISOR 1

Dra. Andrea Magadán Salazar
Doctorado en Ciencias Computacionales
10654097

REVISOR 2

Dra. Azucena Montes Rendón
Doctora en Ciencias
4001014

S.E.P. CENTRO NACIONAL DE INVESTIGACION Y DESARROLLO TECNOLOGICO
RECIBIDO
13:18
SERVICIOS ESCOLARES

C.p. M.T.I. María Elena Gómez Torres - Jefa del Departamento de Servicios Escolares.
Estudiante
Expediente

NACS/lmz

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Centro Nacional de Investigación
y Desarrollo Tecnológico

Cuernavaca, Mor., 25 de junio de 2018
OFICIO No. SAC/288/2018

Asunto: Autorización de impresión de tesis

ING. JOSÉ LUIS MOLINA SALGADO
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS
DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Reconocimiento de movimientos de manos para la manipulación de una interfaz computacional"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
EXCELENCIA EN EDUCACIÓN TECNOLÓGICA®
"CONOCIMIENTO Y TECNOLOGÍA AL SERVICIO DE MÉXICO"

SEP TecNM
CENTRO NACIONAL
DE INVESTIGACIÓN
Y DESARROLLO
TECNOLÓGICO
SUBDIRECCIÓN
ACADÉMICA

DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO

C.p. M.T.I. María Elena Gómez Torres .- Jefa del Departamento de Servicios Escolares.
Expediente

GVGR/mcr

Agradecimientos

A mi madre que siempre me ha dado todo su apoyo y cariño incondicional.

A mi hermano que ha estado ahí para aconsejarme cuando lo necesito.

Al Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) por aceptarme en el programa de maestría en Ciencias Computacionales.

A mi director de tesis, Dr. Máximo López Sánchez por sus consejos, observaciones y por la confianza que siempre ha tenido en mí.

A mi codirector de tesis, Dr. Juan Gabriel González Serna por su aportación durante la realización de la tesis.

A mis revisores la Dra. Andrea Magadán Salazar y la Dra. Azucena Montes Rendón por sus observaciones y sugerencias que me ayudaron a mejorar mi trabajo de tesis.

Al programa de becas del Consejo Nacional de Ciencia y Tecnología (CONACYT) por brindarme el apoyo económico y poder cumplir una de mis metas en la vida.

A mis compañeros de generación por sus sugerencias, observaciones, consejos y amistad.

A los jóvenes que participaron en la realización de mis pruebas por colaborar durante su desarrollo, siempre con entusiasmo y mostrando un gran interés.

A excelentes personas que conocí en el centro de investigación tanto de vigilancia, intendencia, administrativos, directivos e investigadores

Abstract

Nowadays, the use of technological systems helps the execution of people's daily tasks, some of these are: the use of smartphones to buy plane or bus tickets, to watch Internet content on television, to use computer applications to control irrigation systems in greenhouses, control the mouse pointer through the movements of the head, control the water temperature of the shower, etc.

The use of natural movements in the control of devices and technological systems has required the development of new tools such as the recognition of patterns in the movement of people, voice recognition, recognition of hand gestures, among others.

In this work were investigate different human movement detection techniques, there are systems that detect the movement of the extremities of people and they are an association with an instruction to a computer system or device. This work is focused on the recognition of hand movements to do everyday tasks through natural movements.

In accordance with the investigation of different methods and devices which are able of detecting hand movements of detecting hand movements, a method was identified (which uses a depth sensing device), this allows the development of systems to detect hand movements and assign different actions to said movements. Making use of this technique, the developers only need to have basic knowledge in artificial vision because with this method it is possible to define in a textual way the movements through a language of gesture definition.

The use of this technique offers the ability to develop systems not limited to the number of registered movements, this results in the development of new detection systems with a greater number of gestures defined for a greater number of tasks.

As a result of this research, a method was obtained that is able to identify hand movements based on the use of defined positions, these are defined through the orientation of the palm, the fingers, the distance between them and their bending, likewise were carried out and analyzed different tests that allowed to determine that the proposed method is an improvement to the common detection systems because with this one obtains good results and the systems are developed in a simpler way.

Resumen

Hoy en día, el uso de sistemas tecnológicos facilita la ejecución de muchas de las tareas cotidianas de las personas, algunas de estas son: el uso de teléfonos inteligentes para comprar boletos de avión o autobús, visualizar contenido de internet en la televisión, utilizar aplicaciones de computadora para controlar sistemas de riego en invernaderos, control del puntero del mouse a través de movimientos de la cabeza, controlar la temperatura del agua de la regadera, etc.

El uso de movimientos naturales en el control de dispositivos y sistemas tecnológicos ha precisado el desarrollo de nuevas herramientas tales como el reconocimiento de patrones en el movimiento de personas, el reconocimiento de voz, reconocimientos de gestos de las manos, entre otros.

En este trabajo se investigaron diferentes técnicas de detección de movimiento humano, existen sistemas que detectan el movimiento de las extremidades de las personas y lo asocian con una instrucción a un sistema o dispositivo computacional. En concreto, este trabajo está enfocado en el reconocimiento de movimientos de manos que puedan ser utilizados para ejecutar tareas cotidianas a través de movimientos naturales.

De acuerdo con la investigación de los diferentes métodos y dispositivos capaces de realizar la detección de movimientos de manos, se identificó un método que utiliza un dispositivo sensor de profundidad, permite desarrollar sistemas de detección de movimientos de manos y asignar diferentes acciones a dichos movimientos; haciendo uso de esta técnica, los desarrolladores solo necesitan tener conocimientos básicos en el área de visión artificial ya que con este método es posible definir de manera textual los movimientos a través de un lenguaje de definición de gestos.

El uso de esta técnica proporciona la capacidad de desarrollar sistemas no limitados a la cantidad de movimientos registrados, esto da lugar al desarrollo de nuevos sistemas de detección con un número mayor de gestos definidos para ejecutar un número más grande de tareas.

Como resultado de esta investigación, se obtuvo un método que es capaz de identificar movimientos de manos basándose en el uso de posiciones definidas, estas se definen a través de la orientación de la palma, de los dedos, de la distancia entre ellos y su flexión, así mismo se realizaron y analizaron diferentes pruebas que permitieron determinar que el método propuesto es una mejora a los sistemas de detección común ya que con este se obtienen buenos resultados y los sistemas son desarrollados de una manera más sencilla.

.

Contenido

Capítulo I. Introducción 2
1.1 Antecedentes 2
1.2 Objetivos 3
1.3 Justificación 3
1.4 Alcances 3
1.5 Limitaciones 4
1.6 Metodología de solución 4
1.7 Organización de la tesis 7
Capítulo 2. Marco conceptual 9
2.1 Visión artificial 9
2.2 Herramientas de desarrollo 10
2.3 Comunicación 11
Capítulo 3 Estado del arte 13
3.1 Métodos utilizados en las investigaciones 24
3.2 Descripción de los campos del estado del arte. 25
3.3 Tabla comparativa del estado del arte 26
3.4 Comparación y selección del sensor 28
Capítulo 4. Metodología de solución 31
4.1 Fase 1: Diseño del sistema 32
4.1.2 Comparación y selección del método de detección de movimientos. 32
4.1.3 Comparación y selección del sensor. 32
4.1.4 Selección de los movimientos del sistema. 33
4.1.5 Diseño del sistema 43
4.2 Fase 2: Desarrollo del sistema 48
4.2.1 Desarrollo del módulo de inicialización de servicio y aplicación. 48
4.2.2 Desarrollo del módulo de definición de posiciones. 49
4.2.3 Desarrollo del módulo de definición de movimientos 50
4.2.4 Desarrollo del módulo de confirmación de movimientos 51
4.2.5 Desarrollo del módulo de ejecución de comandos 51
4.2.6 Desarrollo del módulo de envío de comandos 53

4.3 Fase 3: Diseño y desarrollo del sistema receptor de comandos 54
4.3.1 Modelado del sistema receptor. 54
4.3.2 Desarrollo del módulo de comunicación. 55
4.3.3 Desarrollo del módulo de ejecución de comandos. 56
4.4 Ambientes de trabajo de los sistemas. 56
Capítulo 5. Pruebas y resultados 59
5.1 Fase 4: Realización de pruebas y análisis de resultados. 59
5.1.2 Pruebas realizadas con 15 usuarios de diferentes tonos de piel. 59
5.1.3 Resultados de las pruebas 63
5.1.4 Reporte del análisis de resultados 70
Capítulo 6. Conclusiones y trabajos futuros. 80
6.1 Conclusiones 80
6.2 Trabajos futuros 81
Bibliografía 83
Anexos 90
Código del sistema de detección de posiciones. 90
Código del sistema de detección de movimientos. 93
Código del sistema de envío de datos. 96
Código del sistema de recepción de datos. 97
Artículo 1: 99
Artículo 2: 105

Índice de Figuras

Figura 1. Fases y procesos de la metodología de solución. 5
Figura 2. Muestra del color de piel del usuario (Han & Rashid, 2016). 13
Figura 3. Proceso de obtención de imagen (Han & Rashid, 2016). 14
Figura 4. Proceso del reconocimiento (Han & Rashid, 2016). 14
Figura 5. Adquisición de la imagen (Tchoketch Kenbir & Bouhedda, 2016). 15
Figura 6. Método de reconocimiento (Tchoketch Kenbir & Bouhedda, 2016). 16
Figura 7. Esquema de funcionamiento (Sueaseenak & Khawdee, 2017). 17
Figura 8. Colocación del sensor en el usuario (Sueaseenak & Khawdee, 2017). 18
Figura 9. Captura de la posición de la mano (Gunawardane, 2016). 19

Figura 10. Esquema de funcionamiento (Gunawardane, 2016). 19
Figura 11. Método utilizado por (Emrehan, 2016). 21
Figura 12. Definición de la parte del cuerpo (Emrehan, 2016). 21
Figura 13. Descriptores de la mano (López, 2016). 22
Figura 14. Descripción de las señas (Microsoft, 2017). 24
Figura 15. Captura de la imagen (Microsoft, 2017). 24
Figura 16. Intel® RealSense™ SR300 (Intel, 2018). 29
Figura 17. Kinect para Windows v2 (Microsoft, 2018). 29
Figura 18. Metodología de solución. 31
Figura 19. Definición de la posición de la mano (Microsoft, 2017). 34
Figura 20. Posición para la letra A. 35
Figura 21. Posición para la letra B. 36
Figura 22. Posición para la letra C. 36
Figura 23. Posición para la letra E. 37
Figura 24. Posición para la letra F. 37
Figura 25. Posición para la letra G. 38
Figura 26. Posición para la letra H. 38
Figura 27. Posición de la letra I. 38
Figura 28. Posición AdelanteA. 39
Figura 29. Posición AdelanteB. 39
Figura 30. Posición AtrasA. 40
Figura 31. Posición AtrasB. 40
Figura 32. Posición CerrarA. 41
Figura 33. Posición CerrarB. 41
Figura 34. Posición PPA. 42
Figura 35. Posición PPB. 42
Figura 36. Diseño del sistema. 44
Figura 37. Módulo de inicialización del servicio de detección y aplicación. 45
Figura 38. Módulo de definición de posiciones de la mano. 46
Figura 39. Módulo de definición de movimientos. 47
Figura 40. Diagrama de flujo del sistema de detección de movimiento. 48
Figura 41. Modelo del sistema receptor de comandos. 54
Figura 42. Modelo del sistema receptor. 55
Figura 43. Objetivo de movimientos realizados. 60
Figura 44. Muestra de los colores de piel. 61
Figura 45. Pruebas con luminosidad alta. 61
Figura 46. Pruebas con luminosidad baja. 61
Figura 47. Movimientos asignados. 74
Figura 48. Opciones para agregar. 74
Figura 49. Dificultad de los movimientos. 74
Figura 50. Movimiento más difícil. 75

Figura 51. Movimiento más fácil. 75
Figura 52. Comparativa de sensores. 76
Figura 53. Movimientos obtenidos RealSense™ SR300 (Intel, 2018). 77
Figura 54. Movimientos obtenidos Kinect V2 (Microsoft, Developer, 2018). 77

Índice de Tablas

Tabla 1. Comparativa de las investigaciones presentadas en el estado del arte. . 26
Tabla 2. Comparativa de las investigaciones presentadas en el estado del arte. . 27
Tabla 3. Sensores soportados (Microsoft, 2017). 28
Tabla 4. Posiciones de la mano. 33
Tabla 5. Pruebas del módulo de comunicación. 62
Tabla 6. Pruebas del Usuario 1 con sensor Intel® RealSense™ SR300 63
Tabla 7. Pruebas del Usuario 1 con sensor Microsoft® Kinect V2. 63
Tabla 8. Pruebas del Usuario 2 con sensor Intel® RealSense™ SR300. 63
Tabla 9. Pruebas del Usuario 2 con sensor Microsoft® Kinect V2. 64
Tabla 10. Pruebas del Usuario 3 con sensor Intel® RealSense™ SR300. 64
Tabla 11. Pruebas del Usuario 3 con sensor Microsoft® Kinect V2. 64
Tabla 12. Pruebas del Usuario 4 con sensor Intel® RealSense™ SR300. 64
Tabla 13. Pruebas del Usuario 4 con sensor Microsoft® Kinect V2. 65
Tabla 14. Pruebas del Usuario 5 con sensor Intel® RealSense™ SR300. 65
Tabla 15. Pruebas del Usuario 5 con sensor Microsoft® Kinect V2. 65
Tabla 16. Pruebas del Usuario 6 con sensor Intel® RealSense™ SR300. 65
Tabla 17. Pruebas del Usuario 6 con sensor Microsoft® Kinect V2. 66
Tabla 18. Pruebas del Usuario 7 con sensor Intel® RealSense™ SR300. 66
Tabla 19. Pruebas del Usuario 7 con sensor Microsoft® Kinect V2. 66
Tabla 20. Pruebas del Usuario 8 con sensor Intel® RealSense™ SR300. 66
Tabla 21. Pruebas del Usuario 8 con sensor Microsoft® Kinect V2. 67
Tabla 22. Pruebas del Usuario 9 con sensor Intel® RealSense™ SR300. 67
Tabla 23. Pruebas del Usuario 9 con sensor Microsoft® Kinect V2. 67
Tabla 24. Pruebas del Usuario 10 con sensor Intel® RealSense™ SR300. 67
Tabla 25. Pruebas del Usuario 10 con sensor Microsoft® Kinect V2. 68
Tabla 26. Pruebas del Usuario 11 con sensor Intel® RealSense™ SR300. 68
Tabla 27. Pruebas del Usuario 11 con sensor Microsoft® Kinect V2. 68
Tabla 28. Pruebas del Usuario 12 con sensor Intel® RealSense™ SR300. 68
Tabla 29. Pruebas del Usuario 12 con sensor Microsoft® Kinect V2. 69
Tabla 30. Pruebas del Usuario 13 con sensor Intel® RealSense™ SR300. 69
Tabla 31. Pruebas del Usuario 13 con sensor Microsoft® Kinect V2. 69
Tabla 32. Pruebas del Usuario 14 con sensor Intel® RealSense™ SR300. 69
Tabla 33. Pruebas del Usuario 14 con sensor Microsoft® Kinect V2. 70
Tabla 34. Pruebas del Usuario 15 con sensor Intel® RealSense™ SR300. 70

Tabla 35. Pruebas del Usuario 15 con sensor Microsoft® Kinect V2. 70
Tabla 36. Efectividad por tono de piel con sensor Intel® RealSense™ SR300. ... 71
Tabla 37. Efectividad por tono de piel con sensor Microsoft® Kinect V2. 71
Tabla 38. Efectividad por género con sensor Intel® RealSense™ SR300. 71
Tabla 39. Efectividad por género con sensor Microsoft® Kinect V2. 71
Tabla 40. Efectividad por luminosidad con sensor Intel® RealSense™ SR300. ... 72
Tabla 41 Efectividad por luminosidad con sensor Microsoft® Kinect V2. 72
Tabla 42. Efectividad por movimientos con sensor Intel® RealSense™ SR300. .. 72
Tabla 43 Efectividad por movimientos con sensor Microsoft® Kinect V2. 73
Tabla 44. Tabla comparativa de resultados. 76
Tabla 45 Cumplimiento de los objetivos específicos. 80

Capítulo I

Introducción

Capítulo I. Introducción

En la actualidad la tecnología se encuentra presente en muchas de nuestras actividades cotidianas, es común encontrar avances tecnológicos en distintas áreas como la salud, seguridad, educación, transporte, etc., algunos de estos son creados con el fin de otorgar comodidades, mientras que otros son auxiliares o realizan tareas de riesgo para las personas o trabajadores.

En el transcurso de los últimos años se ha explorado la posibilidad de manipular dispositivos o interfaces de manera indirecta, esto ha llevado a los investigadores a crear alternativas de manipulación como pinzas para el manejo de objetos radioactivos, pantallas *touch*, interfaces cerebro computadora, etc.; sin embargo, en el campo de la manipulación a través de gestos los sistemas creados son limitados y específicos para cada objetivo, es por esto que uno de los retos presentes hoy en día es poder utilizar los movimientos corporales (específicamente el de las manos) para la manipulación de distintas interfaces o artefactos.

El presente trabajo tiene como finalidad presentar una investigación que se encuentra enmarcada en el uso de un sensor de profundidad para la detección de señas y gestos de la mano derecha, esto con el objetivo de realizar la manipulación indirecta de interfaces computacionales, que permita el desarrollo de aplicaciones no limitativas a la cantidad de gestos ingresados en el sistema y, que la adición de nuevas señas o gestos se realice de una manera sencilla para los próximos desarrolladores.

1.1 Antecedentes

En el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) se desarrolló un trabajo de tesis que tiene por título “Generación de Palabras a Partir de la Lengua de Señas Mexicana”, este se enfoca en el diseño, desarrollo e implementación de un sistema de reconocimiento de gestos de manos apoyándose del uso de un dispositivo con sensores infrarrojos. Este trabajo se desarrolló debido a que las personas con discapacidad auditiva tienen problemas a la hora de comunicarse con personas que desconocen el lenguaje de señas. Las pruebas fueron realizadas evaluando la similitud entre los gestos almacenados en el sistema y los gestos realizados por los usuarios haciendo uso del sensor *Leap Motion Controller* (Nájera, 2017).

Los resultados de las pruebas funcionales del trabajo de investigación dieron como resultado un 55.06 % de precisión en el sistema desarrollado.

1.2 Objetivos

En esta sección se muestran los objetivos del trabajo de tesis.

1.2.1 Objetivo general

- Desarrollar un sistema computacional que identifique los movimientos de manos definidos para manipular interfaces computacionales de manera remota.

1.2.2 Objetivos específicos

- Realizar un módulo que identifique al menos cuatro posiciones de las manos.
- Definir una lista de movimientos que puedan ser usados para manipular una interfaz.
- Definir un método de conexión inalámbrica que permita el envío de datos entre el sistema computacional y la interfaz remota en tiempo real.

1.3 Justificación

La detección y rastreo de personas y objetos es un campo muy investigado; sin embargo, los métodos utilizados para llevar a cabo esta tarea no cubren todos los escenarios, por lo que es necesario utilizar un método específico para cada una de ellas.

Utilizando controles remotos es posible manipular interfaces computacionales (robots, drones, etc.). Estos dispositivos tienen un alcance limitado a su cobertura (Bluetooth, Radiofrecuencia, etc.) además de que su manipulación requiere de mover palancas, botones o pantallas táctiles.

Actualmente un método de detección de movimiento no es capaz de llevar a cabo esta tarea de una manera sencilla, ya que no se cuenta con reglas preestablecidas para facilitar el desarrollo de sistemas de detección de movimiento. En una técnica de detección de movimientos, la formalización de patrones de movimiento, así como la definición de una gramática visual son una solución formal que permitirá utilizar métodos preestablecidos en sistemas de manipulación de interfaces, los que podrán ser implementados en el control remoto de dispositivos o interfaces computacionales.

1.4 Alcances

- El sistema tendrá la capacidad de detectar movimientos de las manos de una persona.
- El sistema identificará acciones en base a los movimientos detectados.
- El sistema se comunicará con una interfaz remota previamente definida.

- El sistema tendrá la capacidad de realizar tareas en conjunto de una interfaz computacional remota.
- Las imágenes bajo análisis serán adquiridas en un ambiente interior controlado.
-

1.5 Limitaciones

- El sistema no será capaz de comunicarse con más de una interfaz de manera simultánea.
- El sistema no será capaz de conectarse con cualquier interfaz remota.
-

1.6 Metodología de solución

La figura 1 muestra las fases y los procesos de la metodología de solución, así como las conexiones de cada proceso y cada fase.

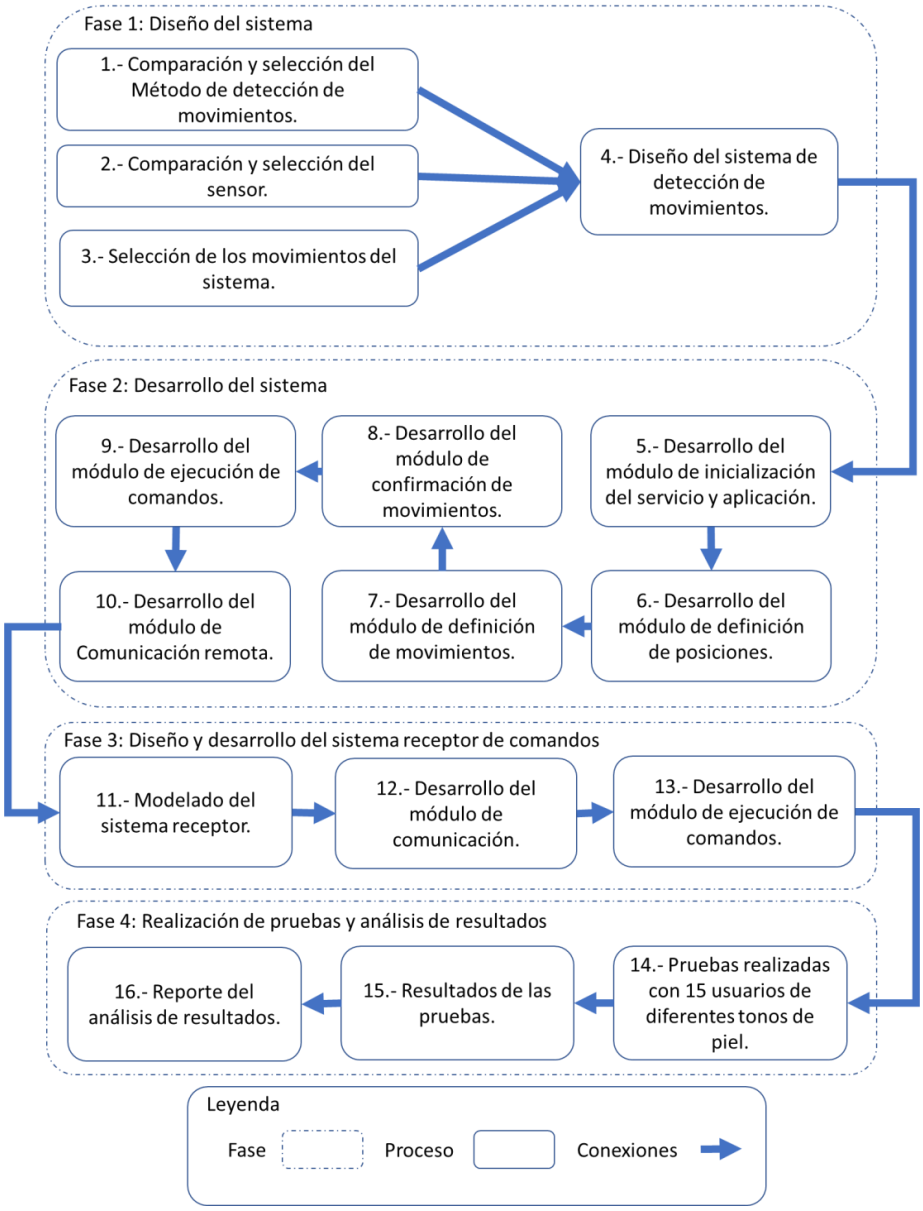

Figura 1. Fases y procesos de la metodología de solución.

Fase 1: Diseño del sistema

1. **Comparación y selección del método de detección de movimientos:** En este proceso se investigaron distintos métodos de detección de movimiento y se seleccionó uno, con el cual se pueden crear sistemas de detección de movimientos de manos basados en la descripción de las posiciones.
2. **Comparación y selección del sensor:** En este proceso se realizó una investigación acerca de diferentes dispositivos que pueden ser utilizados

para la detección de gestos y se seleccionó el indicado para el desarrollo del sistema de acuerdo con su capacidad de detección.

3. **Selección de los movimientos del sistema:** En este proceso se determinaron 4 movimientos de manos, los que están compuestos por dos posiciones cada uno, estos gestos se determinaron para realizar la manipulación de la aplicación “Spotify”.
4. **Diseño del sistema:** En este proceso se diseñó el sistema a realizar, tomando en cuenta el método y sensor seleccionado, así como el lenguaje de programación y el sistema operativo destino.

Fase 2: Desarrollo del sistema:

5. **Desarrollo del módulo de inicialización del servicio y aplicación:** En este proceso se programaron instrucciones para iniciar el servicio de detección del Proyecto Praga y se realiza la ejecución de la aplicación multimedia.
6. **Desarrollo del módulo de definición de posiciones:** En este proceso se definieron las ocho posiciones que utilizaran los movimientos a detectar.
7. **Desarrollo del módulo de definición de movimientos:** En este proceso se realizó un módulo que toma las posiciones conjuntas de un movimiento y las almacena en variables para poder ser reconocidas.
8. **Desarrollo del módulo de confirmación de movimientos:** En este proceso se programaron instrucciones para activar el sensor y comparar las imágenes de entrada con los movimientos definidos en el sistema.
9. **Desarrollo del módulo de ejecución de comandos:** En este proceso se programaron instrucciones para la manipulación de una aplicación multimedia que es ejecutada sobre el sistema Windows.
10. **Desarrollo del módulo de comunicación remota:** En este proceso se realizó la programación del módulo que envía los comandos generados a una aplicación receptora conectada de forma remota.

Fase 3: Diseño y desarrollo del sistema receptor de comandos

11. **Modelado del sistema receptor:** En este proceso se modeló un sistema que tuviera la capacidad de recibir los comandos enviados por el sistema de detección de gestos y que a su vez realizara la ejecución de los comandos recibidos.
12. **Desarrollo del módulo de comunicación:** En este proceso se programó un módulo de comunicación que es capaz de recibir los comandos enviados por el sistema de detección de gestos.
13. **Implementación del módulo de ejecución de comandos:** En este proceso se implementó el mismo módulo que se creó para el sistema de

detección de movimiento y para la ejecución de comandos, mismos que realizan la manipulación de la aplicación “Spotify”.

Fase 4: Realización de pruebas y análisis de resultados.

14. **Pruebas realizadas con 15 usuarios:** En este proceso se llevaron a cabo las pruebas del sistema de detección de gestos, se seleccionaron 15 usuarios de distintos tonos de piel y se probó el sistema con dos sensores diferentes.
15. **Resultados de las pruebas:** En este proceso se agruparon y clasificaron las notas de los resultados obtenidos por cada uno de los usuarios con los que se realizaron las pruebas.
16. **Reporte del análisis de resultados:** En este proceso se analizaron los resultados obtenidos y se generó un reporte que muestra el porcentaje de efectividad del sistema.

1.7 Organización de la tesis

Este trabajo de tesis se encuentra dividido en siete capítulos, en los cuales se describe el trabajo de investigación realizado:

El capítulo uno muestra la información general relacionada con este trabajo de investigación como son antecedentes del proyecto, descripción del problema, metodología de solución, alcances y limitaciones.

El capítulo dos aborda el marco conceptual que contiene los conceptos más relevantes utilizados durante el desarrollo de este trabajo de investigación.

El capítulo tres contiene el estado del arte donde se muestran las investigaciones realizadas que se relacionan con el este trabajo.

El capítulo cuatro aborda la metodología de solución empleada para alcanzar los objetivos planeados en el capítulo uno.

El capítulo cinco describe las pruebas realizadas con el sistema de detección de gestos y el sistema de ejecución de comandos.

El capítulo seis aborda la conclusión general de este trabajo de investigación y los trabajos futuros.

El capítulo siete contiene la bibliografía y los anexos que complementan el trabajo de investigación.

Capítulo II

Marco conceptual

Capítulo 2. Marco conceptual

El presente capítulo aborda los conceptos teóricos requeridos para la comprensión de este trabajo.

2.1 Visión artificial

En esta categoría se muestran los conceptos relacionados al área de visión artificial.

Gestos

Un gesto es definido como el movimiento de las manos, brazos, cabeza, etc. para expresar algún tipo de idea o sentimiento (Cambridge Dictionary, 2017).

Reconocimiento de gestos

El reconocimiento gestual es el proceso de registrar un gesto del mundo real, de analizar los parámetros capturados y de dar como resultado qué gesto humano se ha realizado. El sistema de captura de movimiento es la solución electrónica o mecánica que se encarga de transformar el movimiento puramente físico en información de tipo digital (Moreno, 2010).

Clasificador

El término clasificador se utiliza en referencia al algoritmo que sirve para asignar un elemento entrante no etiquetado en una categoría concreta conocida. Dicho algoritmo, permite ordenar o disponer por clases elementos entrantes a partir de cierta información característica de estos. Una manera de implementar un clasificador es seleccionar un conjunto de ejemplos etiquetados y tratar de definir una regla que permita asignar una etiqueta a cualquier otro dato de entrada. En ocasiones, el término clasificador también es utilizado para referirse a la función matemática que implementa el algoritmo de clasificación (Mills, 2010).

Procesamiento de imágenes

El procesamiento de imágenes tiene como objetivo mejorar el aspecto de las imágenes y hacer más evidentes en ellas ciertos detalles que se desean hacer notar. La imagen puede haber sido generada de muchas maneras, por ejemplo, fotográficamente, o electrónicamente, por medio de monitores de televisión. En general, el procesamiento de las imágenes se puede hacer por medio de métodos ópticos o bien por medio de métodos digitales en una computadora (Biblioteca Digital ILCE, 2018).

NUI

Interfaz natural de usuario (Natural User Inteface, NUI) es la capacidad de interactuar con una máquina usando únicamente el cuerpo humano. NUI es una interacción con el tacto, los gestos, por el sonido o por los sentidos (Onysus, 2018).

Project Prague
Project Prague es un SDK que le permite crear experiencias NUI basadas en la entrada de gestos manuales. Proporciona APIs para C #, C ++, lo que le permite diseñar e implementar fácilmente sus propios gestos de mano personalizados e integrarlos en sus aplicaciones (Microsoft, Microsoft Docs, 2017).

Sensor de profundidad (RGB-D)
Son un tipo específico de dispositivos de detección de profundidad que trabajan en asociación con una cámara RGB, que son capaces de aumentar la imagen convencional con información de profundidad (relacionada con la distancia al sensor) en una base por píxel (GLobal, 2018).

Casco convexo
Teniendo un conjunto de puntos en el plano, su envoltura convexa (o casco convexo) está definida por el polígono convexo de área mínima que cubre todos los puntos (esto es, todos los puntos están dentro del polígono). Esta definición también se puede generalizar para puntos en el espacio o para puntos en alguna dimensión mayor, cambiando el polígono por un poliedro (world of πer, 2018).

Algoritmo Scan de Graham
Este algoritmo utiliza una lista en la que va almacenando y ordenando correctamente los puntos que constituyen los extremos del cierre convexo (Fernández Fadrique & Baquero Vega, 2018).

Algoritmo de Ramer–Douglas–Peucker
Es un algoritmo para reducir el número de puntos utilizados en la aproximación de una curva. El objetivo del algoritmo es, dada una curva compuesta por segmentos, encontrar una curva similar aproximada con menos puntos. El algoritmo define una diferencia basada en la máxima distancia entre la curva original y la curva simplificada. La curva simplificada consiste en una reducción de los puntos que definían la curva original (Hershberger, 1992).

Transformada de Hough
La transformada de Hough es una herramienta que permite detectar curvas en una imagen. Es una técnica muy robusta frente al ruido y a la existencia de huecos en la frontera del objeto. A la hora de aplicar la transformada de Hough a una imagen es necesario obtener primero una imagen binaria de los píxeles que forman parte de la frontera del objeto (Segmentación. Transformada de Hough. , 2006).

2.2 Herramientas de desarrollo
En esta categoría se muestran las herramientas utilizadas en el desarrollo del trabajo de tesis.

Windows

El sistema operativo Windows de Microsoft es un sistema operativo que para instalarlo y utilizarlo en un equipo informático es necesario comprar una licencia. Tras el sistema operativo Mac OS de Apple, Windows tomó su nombre debido a que fue uno de los primeros en incorporar una interfaz gráfica de usuario basada en ventanas. A lo largo de su historia han existido varias versiones de Windows, a las que le han ido añadiendo mejoras en su entorno gráfico, en su rendimiento, en el aprovechamiento de las características hardware del equipo y en las aplicaciones de utilidades para facilitar la labor al usuario (Reyes, 2017).

SDK

Un SDK (Software Development Kit), o kit de desarrollo de software, es un conjunto de herramientas que ayudan a la programación de aplicaciones para un entorno tecnológico particular. Es decir, las aplicaciones desarrolladas sobre el SDK estarán destinadas a algún sistema operativo, plataforma hardware, consola de videojuegos o paquete de software en especial (Software Factory, 2018).

API

Una API (siglas de 'Application Programming Interface') es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes, de la misma manera en que la interfaz de usuario facilita la interacción humano-software (Merino, 2018).

Visual Studio

Visual Studio es un conjunto de herramientas y otras tecnologías de desarrollo de software basadas en componentes, que sirven para crear aplicaciones eficaces y de alto rendimiento, permitiendo a los desarrolladores crear sitios y aplicaciones web, así como otros servicios web en cualquier entorno que soporte la plataforma. En palabras más específicas, Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C# y Visual C++ utilizan todo el mismo entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y facilita la creación de soluciones en varios lenguajes (MSN, 2018).

2.3 Comunicación

En esta categoría se muestra el concepto utilizado para la comunicación remota.

Conexión remota

La conexión remota es una tecnología que permite el acceso a distancia a un dispositivo computacional desde otro punto del planeta, esto se logra a través de una conexión de red establecida por fibra óptica, ethernet, Wi-Fi, etc. (KANEDA, 2018).

Capítulo III

Estado del arte

Capítulo 3 Estado del arte

Los sistemas computacionales para la detección de gestos han sido diseñados de distintas maneras, estos pueden ser creados con cámaras de video o con dispositivos elaborados a base de distintos sensores que son especialmente diseñados para dicho trabajo. A continuación, se muestran diferentes investigaciones realizadas con estas técnicas.

Como se puede observar en (Han & Rashid, 2016), se desarrolló un sistema de reconocimiento de gestos basado en una arquitectura ARM Cortex-A8 en la cual se instaló un sistema operativo Linux donde se realizó un estudio y la medición en tiempo real de las respuestas a las pruebas realizadas. Los resultados de estas pruebas indicaron que el sistema realizado otorga una respuesta en tiempo real del reconocimiento de los gestos de las manos y comandos de voz, lo que permite el control de dispositivos ZigBee sobre un canal de comunicación inalámbrico.

Este método comprende el uso de una cámara web para tomar las imágenes. El método utilizado es el siguiente.

- Paso 1: El sistema toma una muestra de la piel del usuario y crea un perfil de color basado en ella, este perfil de color ayuda al sistema para realizar la extracción de fondo de la imagen (Figura 2).

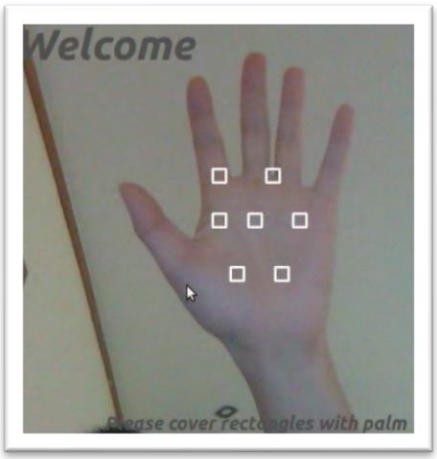

Figura 2. Muestra del color de piel del usuario (Han & Rashid, 2016).

De acuerdo con la muestra de color el sistema es capaz de extraer el fondo haciendo uso del umbral. En este caso la muestra provee de 7 colores al perfil.

- Paso 2: El sistema produce una imagen binaria tomando en cuenta cada color del perfil.

Figura 3. Proceso de obtención de imagen (Han & Rashid, 2016).

La figura 3 muestra el proceso para obtener la imagen del movimiento a identificar, dónde se suman las imágenes binarias y se filtra el ruido para poder obtener una imagen suavizada de la mano.

- Paso 3: Se necesitan cuatro etapas clave para reconocer un movimiento de la imagen suavizada, la figura 4 muestra el procedimiento.

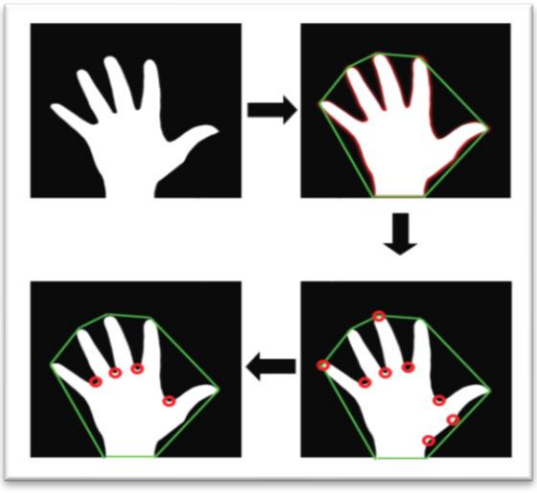

Figura 4. Proceso del reconocimiento (Han & Rashid, 2016).

Primero se busca el contorno en la imagen binaria usando el algoritmo de seguimiento de bordes, posteriormente se busca el casco convexo de un punto usando el algoritmo de R. L. Graham y el algoritmo F. Yao, posteriormente se buscan los puntos más alejados del vértice convexo usando el algoritmo de Duglas-Peuker, la última etapa consiste en descartar los defectos de convexidad irrelevantes. Con esto se puede obtener el alcance y la posición de la mano, de acuerdo con la diferencia en el número de las puntas de los dedos se pueden crear diferentes gestos.

En el artículo (Tchoketch Kenbir & Bouhedda, 2016), se muestra un sistema para la detección de gestos y generación de instrucciones que se basa en el uso de MatLab en conjunto con un dispositivo construido con Arduino, donde las imágenes adquiridas son tratadas con la transformada de Houg para definir cuáles son los objetivos que se desean. Las señales de control se generan y se envían a través de una conexión inalámbrica para hacer que el dispositivo se mueva en la dirección deseada.

Este método hace uso de una cámara web para tomar la captura de las imágenes para posteriormente procesarlas.

Paso 1: El sistema adquiere una imagen a través de una cámara web de una laptop, esta es una imagen colorida de 256x256 pixeles, la elección de esta imagen de baja resolución permite que el procesamiento y el reconocimiento sean más rápidos, con estos parámetros permiten que el control en tiempo real sea alcanzado (Figura 5).

Figura 5. Adquisición de la imagen (Tchoketch Kenbir & Bouhedda, 2016).

Paso 2: Dentro del procesamiento, el problema en visión por computadora más común es localizar un objeto en particular dentro de una imagen. Este problema es resuelto mediante el uso de la Transformada Circular de Hough (Por sus siglas en inglés CHT).

Para llevar a cabo la localización y detección de un objeto, son utilizadas algunas técnicas de procesamiento, la morfología matemática es una herramienta para extraer los componentes de la imagen que son útiles en la representación y descripción de una región y forma. La erosión y dilatación son las operaciones morfológicas fundamentales que se usan para definir las operaciones de cierre y apertura.

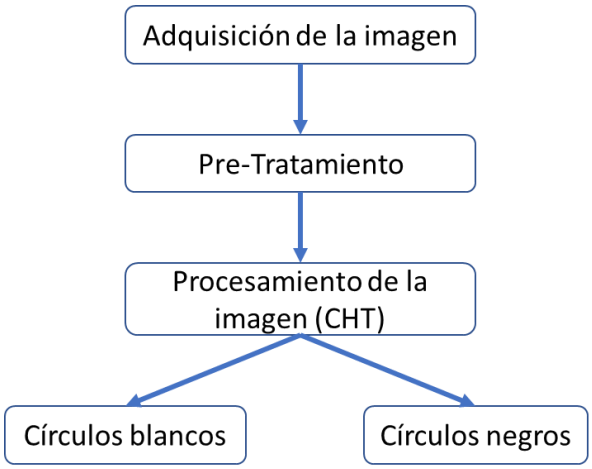

Figura 6. Método de reconocimiento (Tchoketch Kenbir & Bouhedda, 2016).

Una vez que el radio y el centro de los objetos son identificados, el objeto corresponde a un mínimo de dos círculos blancos y dos círculos negros con el mismo centro, estas son las condiciones que se revisan en la imagen adquirida para poder localizar el objeto (Figura 6).

Otro trabajo que se relaciona es (Gergely , Ujbányi, & József , 2016)**,** en el cual el sistema que se desarrolló tiene como objetivo la detección de gestos a partir del movimiento de las manos, este sistema incorpora sensores de proximidad para medir las distancias entre un dispositivo y las manos del usuario, posteriormente se lleva a cabo un procesamiento de todos los datos en un sistema computacional ejecutado en una aplicación de escritorio.

Otros dispositivos con los que se puede realizar la detección de gestos y que permiten la interacción con sistemas computacionales son los que se muestran en (Csapo & Kristjánsson, 2016)**,** donde se analizan los dispositivos táctiles y apticos, además que se dan a conocer los distintos experimentos que se realizaron con el brazalete “Myo” utilizado controlar la retroalimentación auditiva en tiempo real. El objetivo de esta investigación es encontrar una manera de que los usuarios con discapacidades visuales puedan interactuar con sistemas de cómputo.

Hoy en día el uso de las NUI cada vez es más común en los sistemas de interacción, sin embargo, aún no existe una técnica que otorgue resultados perfectos a la hora de realizar la detección del cuerpo humano; en (Lee & Tanaka, 2013) se explica la propuesta de un sistema con un método de interacción natural, además del diseño de los gestos de la mano utilizando la identificación de dedos mediante dispositivos de múltiples sensores como Kinect.

Los autores en (Sueaseenak & Khawdee, A Performance of Modern Gesture Control Device with Application in Pattern Classification, 2017) ponen a prueba el funcionamiento del brazalete “Myo” en conjunto con una aplicación de extracción de características basada en electromiografía (EMG) de varios canales, de esta manera son capaces de extraer la información de los músculos del brazo durante la representación de 6 gestos de la mano incluyendo mano abierta, mano cerrada, mano flexionada y la mano en posición normal. Los criterios utilizados fueron el índice de decibelios y el criterio de dispersión de los datos del EMG con lo que demuestran que el uso del dispositivo es adecuado para la detección de gestos.

Este método utiliza un sensor en forma de brazalete (“Brazalete MYO”), con este sensor se capturan señales eléctricas provenientes de los músculos del brazo de una persona.

Figura 7. Esquema de funcionamiento (Sueaseenak & Khawdee, 2017).

En la figura 7 se muestra el esquema del funcionamiento del sistema que indica los procesos necesarios para realizar la detección de un movimiento.

- Paso 1: El sistema realiza la recolección de datos provenientes del sensor de siete personas, los datos se obtienen de seis gestos de la mano (Mano cerrada, mano abierta, mano flexionada, doble toque y posición normal de la mano). Después de colocar el sensor en los músculos de la persona (Figura 8), la señal proveniente del sensor es enviada a una computadora a través de una conexión bluetooth.

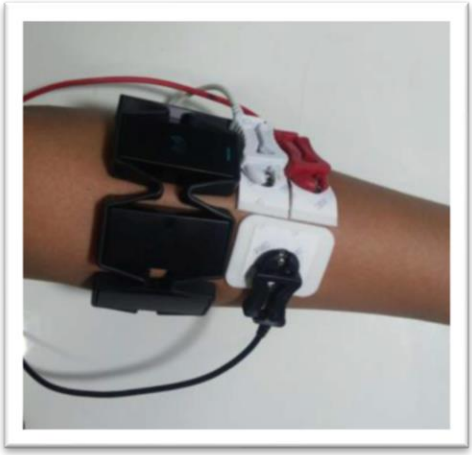

Figura 8. Colocación del sensor en el usuario (Sueaseenak & Khawdee, 2017).

La señal proveniente del sensor es en forma de ondas, sin embargo, mediante el uso del software MYO es posible grabar los datos y almacenarlos en un archivo de hoja de cálculo.

- Paso 2: El sistema realiza una extracción de características mediante una colección de datos de tres canales, cada canal se encuentra conectado a un tipo de músculo superficial, en este proceso se utiliza una función de electromiografía y el método denominado Valor Absoluto Medio.
- Paso 3: El sistema verifica que las señales obtenidas y procesadas, coincidan con las señales muestra de cada uno de los gestos de la mano que previamente se recolectaron.

La investigación (Gunawardane, 2016) muestra el desarrollo de un sistema de detección de gestos utilizando *Leap Motion Controller*, este sistema realiza el seguimiento de tres de los dedos de una mano, así como la muñeca para controlar un mecanismo de dedos blandos que imita el movimiento de la mano del usuario, todo esto se logró gracias a la precisión de los sensores y la programación del sistema.

Este método es utilizado en conjunto con un dispositivo sensor llamado Leap Motion Controller, a través del sensor y el uso del software Processing se lleva a cabo la detección de gestos.

- Paso 1: El sistema realiza la captura de la posición de la mano (Figura 9), los datos son adquiridos por el software Processing en dónde se lleva a cabo la extracción de características, identificación y toma de decisión de las salidas de control. La figura 10 muestra un esquema del funcionamiento del sistema.

Figura 9. Captura de la posición de la mano (Gunawardane, 2016).

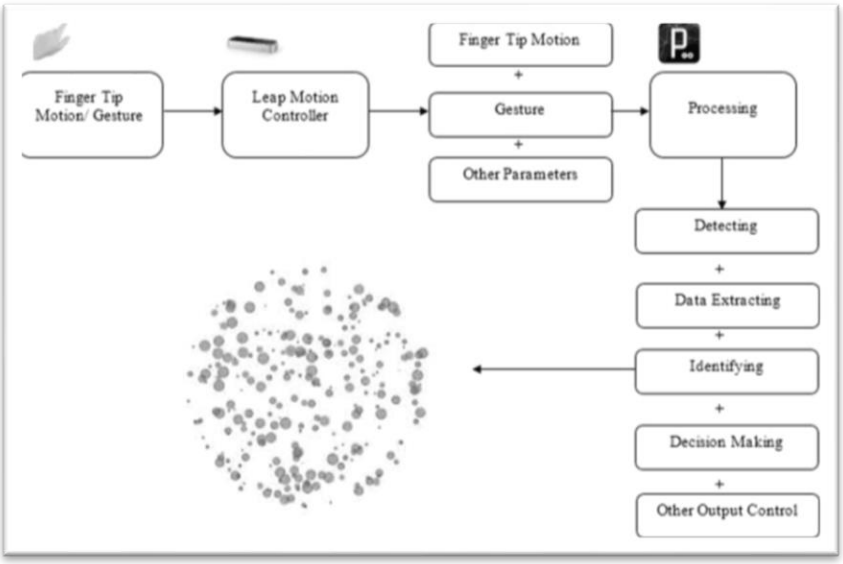

Figura 10. Esquema de funcionamiento (Gunawardane, 2016).

El principal requisito para el diseño del sistema es detectar la punta del dedo, extraer la posición, orientación y los datos de la velocidad de movimiento. El rastreo de la punta de los dedos y de la muñeca fueron recreados en el software Processing.

- Paso 2: El sistema utiliza el dedo índice para extraer los datos de posición orientación y la velocidad, posteriormente extrae cuatro vectores del marco de coordenadas del Leap Motion Controller para cada dedo y para la muñeca.
- Paso 3: Utilizando la relación que existe entre la muñeca y la distancia de la punta de los dedos, se generan gestos que son utilizados como datos de control para una mano robótica.

En algunas ocasiones la intervención humana para llevar a cabo una tarea es indispensable, sin embargo, día con día se crean nuevas alternativas para minimizar el riesgo humano, tal es el caso de (Bendale & Kharat, 2017) en donde desarrollaron un sistema de control de gestos para el control de un brazo robótico que tiene como finalidad la manipulación de materiales peligrosos en estaciones nucleares. El sistema está compuesto por componentes electrónicos que detectan los movimientos del usuario y un brazo robótico que replica dichos movimientos en tiempo real, comunicando los dispositivos a través de una conexión inalámbrica.

El artículo (Deng, Chang, & Jian, 2017) muestra un panorama en donde se da a conocer un sistema multimodal en donde se hace uso de dos dispositivos para la manipulación de una interfaz computacional, utilizan un sensor para el rastreo del ojo humano y un sensor de profundidad para el control de gestos de la mano, con esto demuestran que el hacer uso de distintos sensores en conjunto mejora la precisión y la experiencia del usuario al realizar la manipulación de interfaces.

Algunos investigadores centran sus trabajos en el desarrollo de sistemas que ayudan a la población, tal es el caso de (Antón, Kurillo, Goñi, Illarramendi, & Bajcsy, 2017) en donde se apoyan del uso de las nuevas tecnologías para la detección de gestos humanos para desarrollar un sistema de ayuda a personas que necesitan una rehabilitación, el sistema hace uso del sensor de profundidad Kinect para la detección de gestos, además de realizar una conexión remota entre dos de estos sistemas, uno para la persona que necesitan la rehabilitación y por otro lado un fisioterapeuta que indica gráficamente cuales son los movimientos que debe realizar el paciente.

En ocasiones se desarrollan sistemas de detección de movimientos con la finalidad de hacer la manipulación de robots, estos replican el movimiento humano con la ayuda de sensores como en el caso de (Emrehan, 2016)**,** donde se presenta un sistema que permite al usuario comunicarse con un robot humanoide utilizando señales no verbales. Utiliza un sensor Kinect para enseñar acciones humanas a un robot humanoide mediante el uso de técnicas de inteligencia artificial.

Este método hace uso de un sensor de profundidad (Kinect) para la captura y procesamiento de las imágenes, la figura 11 muestra un esquema del funcionamiento del sistema.

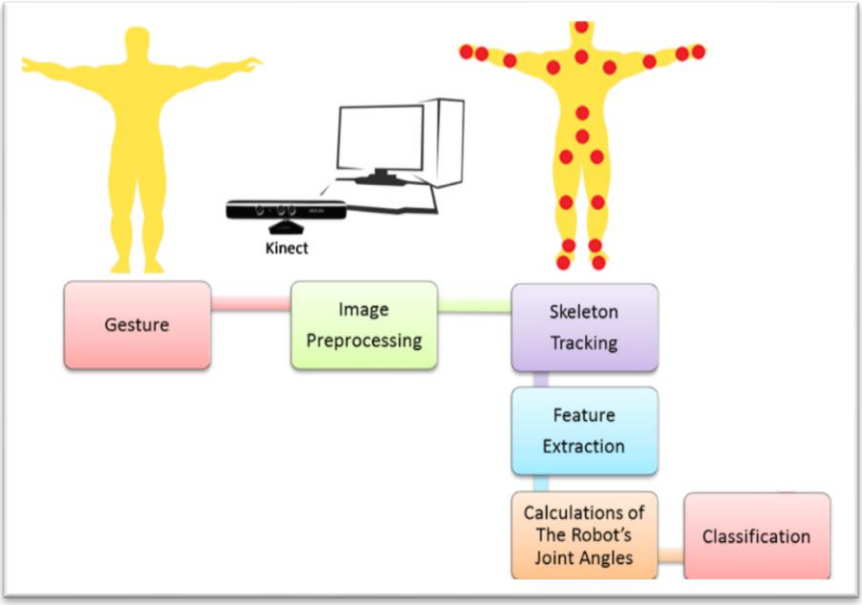

Figura 11. Método utilizado por (Emrehan, 2016).

- Paso 1: El sistema adquiere la imagen desde el sensor de profundidad Kinect, la cual contiene datos de imagen RGB y de profundidad a través de sensores infrarrojos colocados en el mismo sensor.
- Paso 2: El sistema realiza un preprocesamiento de la imagen haciendo uso del SDK proporcionado por Microsoft para el sensor Kinect.
- Paso 3: Teniendo en cuenta que, en Kinect, las partes del cuerpo consisten en una combinación de dos puntos de unión diferentes, las coordenadas cartesianas se definen por medio de los puntos terminales de las partes del cuerpo. Cada parte del cuerpo definida entre puntos de unión de Kinect se evalúa como un vector de posición de inicio y fin, las que son coordenadas previamente conocidas en el espacio 3D.

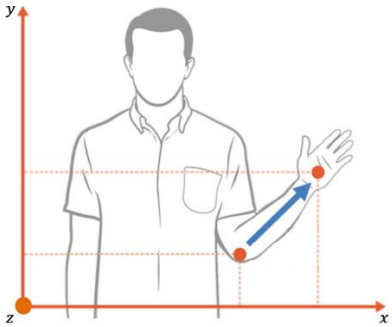

Figura 12. Definición de la parte del cuerpo (Emrehan, 2016).

La figura 12 muestra la definición de la parte del cuerpo humano. Los vectores de posición definidos por las posiciones conjuntas de Kinect no pueden

transferirse directamente a un robot humanoide, por esto en este trabajo se utiliza un algoritmo de transformación para obtener los ángulos del brazo y así poder enviarle instrucciones a un robot humanoide.

La investigación realizada por (López, 2016) es una muestra más de las capacidades de los sistemas de detección de movimiento, el autor en este trabajo realiza un sistema que permite la identificación de letras del alfabeto haciendo uso del lenguaje de señas, el sistema realiza la captura de imágenes a través de un sensor *RealSense* y el uso del kit de desarrollo provisto por Intel.

Este método se caracteriza por utilizar un sensor de profundidad RealSense y el SDK provisto por Intel para el propio sensor, a través de estos componentes se logra hacer la detección de señas y gestos de las manos.

- Paso 1: Primeramente, el sistema realiza la captura de imágenes, tanto RGB e infrarroja para posteriormente ser procesadas a través del uso del SDK de RealSense.
- Paso 2: Para realizar la detección de la mano se toman en cuenta los descriptores de la mano como se muestra en la figura 13, estos descriptores son puntos de interés extraídos a partir del SDK del sensor.

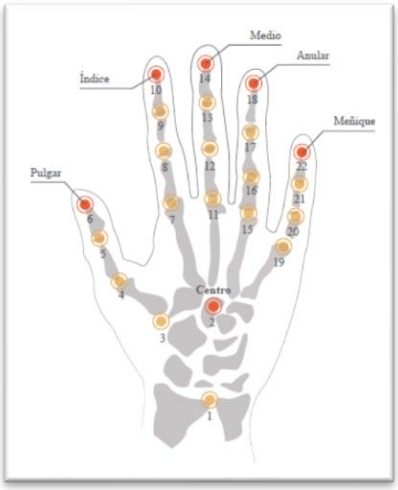

Figura 13. Descriptores de la mano (López, 2016).

El SDK en conjunto con el procesador de imágenes de RealSense permite realizar reconocimiento de manos, seguimiento y reconocimiento de rostros, escaneo 3D, eliminación de fondo, seguimiento de objetos y reconocimiento de voz.

- Paso 3: El sistema realiza la clasificación con redes neuronales y máquinas de vector soporte en dónde, aquel conjunto que tenga mejor relación entre

error de validación, tiempo de ejecución y complejidad del algoritmo es el elegido.

En la investigación de (Eyal Krupka, 2017) se muestra un conjunto de herramientas que permiten la detección de movimientos en tiempo real a través de sensores de profundidad, esto se logra definiendo un conjunto de proposiciones básicas. Este trabajo demuestra que es posible realizar definiciones de una manera más sencilla y con un costo computacional bajo.

En esta investigación se propone un método de detección de movimientos alternativo que hace uso de los sensores de profundidad.

En este trabajo se analizan y prueban diferentes técnicas de detección de movimientos, si bien existen diferentes algoritmos de procesamiento y clasificación, no todas las alternativas son convenientes para cualquier desarrollo.

En este trabajo se crea un lenguaje de descripción de gestos, que permite generar diferentes definiciones de gestos sin la necesidad de llevar a cabo un reentrenamiento del sistema, se mencionan tres ventajas de este lenguaje.

1. Facilita la retroalimentación del gesto.
2. Evita la división de código entre los componentes (Durante la programación).
3. Facilita el tratamiento de la composición de los gestos.

Se obtuvieron los mejores resultados al hacer uso de un método basado en bosque aleatorio de múltiples etapas para discriminar con éxito entre 6 poses predefinidas. Se presenta una gran ventaja al obtener buenos resultados ya que este método no genera un alto costo computacional, esto permite que las detecciones sean más rápidas y equipos de cómputo de gama media.

El Proyecto Praga (Microsoft, Microsoft Docs, 2017) es el resultado de esta investigación, es un SDK que permite el uso del **Lenguaje de Descripción de Gestos** y permite una identificación de las señas y gestos de las manos de manera sencilla.

- Paso 1: En el sistema se realiza la descripción de las señas que serán utilizadas, todo esto haciendo uso del lenguaje de descripción de gestos (Figura 14).

```
var rotateSet = new HandPose("RotateSet", new FingerPose(new[] { Finger.Thumb, Finger.Index }, FingerFlexion.Open, PoseDirection.Forward)
                                          new FingertipPlacementRelation(Finger.Index, RelativePlacement.Above, Finger.Thumb),
                                          new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching, Finger.Thumb));

var rotateGo = new HandPose("RotateGo", new FingerPose(new[] { Finger.Thumb, Finger.Index }, FingerFlexion.Open, PoseDirection.Forward),
                                        new FingertipPlacementRelation(Finger.Index, RelativePlacement.Right, Finger.Thumb),
                                        new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching, Finger.Thumb));

var rotateRight = new Gesture("RotateRight", rotateSet, rotateGo);
```

Figura 14. Descripción de las señas (Microsoft, 2017).

- Paso 2: El sistema captura la imagen a través de un sensor de profundidad 3D (Kinect o RealSense) y compara el gesto con los gestos descritos en el sistema (Figura 15).

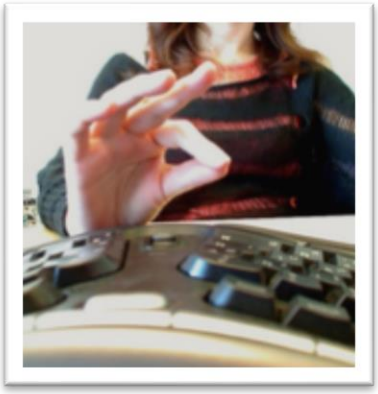

Figura 15. Captura de la imagen (Microsoft, 2017).

- Paso 3: El sistema ejecuta las acciones programadas al detectar si el gesto está descrito o no.

3.1 Métodos utilizados en las investigaciones

Con las investigaciones realizadas con anterioridad se muestran distintas formas de llevar a cabo la detección y el análisis del movimiento humano, estos métodos poseen características similares, sin embargo, se pueden dividir en 3 categorías principales.

- Sistemas desarrollados que utilizan sensores que están conectados al usuario.
    - Estos sistemas se componen por un sistema que detecta las señales emitidas por uno o varios sensores que porta el usuario y de esta manera poder identificar los movimientos.
- Sistemas desarrollados que utilizan directamente la visión artificial
    - Estos sistemas generalmente están compuestos por una cámara de video para capturar las imágenes, un método de procesamiento de imágenes, el uso de algoritmos de clasificación, entrenamiento de los

sistemas y una comparación de los datos de entrada con los datos de la base de conocimiento.

- Sistemas desarrollados que utilizan un lenguaje de programación para interpretar la información proveniente de sensores de profundidad.
    - Estos sistemas están principalmente caracterizados por hacer uso de sensores de profundidad, estos dispositivos cuentan con un Kit de Desarrollo de Software (SDK) que provee las herramientas necesarias para el desarrollo de aplicaciones en distintas plataformas.

3.2 Descripción de los campos del estado del arte.

A continuación, se describe cada uno de los criterios que se emplearon para realizar la tabla comparativa.

- **Trabajo:** Nombre del articulo o trabajo investigado.
- **Año:** Año de la publicación.
- **Sensor:** Nombre del sensor principal utilizado.
- **SDK:** Indica si el sensor cuenta con herramientas de desarrollo propias.
- **PDI:** Indica si es necesario llevar a cabo el Procesamiento Digital de Imágenes para llegar a los resultados.
- **Entrenamiento:** Indica si es necesario entrenar a los clasificadores con una muestra de imágenes de la mano.
- **Definición de posiciones por descripción:** Indica si es posible definir una posición únicamente describiéndola.

3.3 Tabla comparativa del estado del arte

La tabla 1 y 2 muestran una comparativa de las investigaciones presentadas en el estado del arte.

Tabla 1. Comparativa de las investigaciones presentadas en el estado del arte.

Trabajo	Base de datos	Sensor	SDK	PDI	Entrenamiento	Descripción de posiciones
Gesture and Voice Control of Internet of Things (Han & Rashid, 2016).	Si	Cámara Web	No	Si	Si	No
Gesture Control of Mobile Robot Based Arduino Microcontroller (Tchoketch Kenbir & Bouhedda, 2016).	Si	Cámara Web	No	Si	Si	No
Cost-effective hand gesture computer control (Gergely , Ujbányi, & József , 2016).	No	Ultrasónico	No	No	Si	No
Evaluation of Human-Myo Gesture Control (Csapo & Kristjánsson, 2016).	Si	Brazalete Myo	No	No	Si	No
Finger Identification and Hand Gesture Recognition (Lee & Tanaka, 2013).	Si	Kinect	Si	Si	Si	No
A Performance of Modern Gesture Control Device with Application in Pattern (Sueaseenak & Khawdee, A Performance of Modern Gesture Control Device with Application in Pattern Classification, 2017).	Si	Brazalete Myo	No	No	Si	No
The Development of a Gesture Controlled Soft Robot (Gunawardane, 2016)	Si	Leap Motion Controller	Si	No	SI	No
A design of Gesture controlled mobile robot with robotic arm for nuclear environment (Bendale & Kharat, 2017).	No	Flex	No	No	Si	No

Tabla 2. Comparativa de las investigaciones presentadas en el estado del arte.

Trabajo	Base de datos	Sensor	SDK	PDI	Entrenamiento	Descripción de posiciones
Gaze Modulated Disambiguation Technique for Gesture Control in 3D Virtual Objects Selection (Deng, Chang, & Jian, 2017).	Si	Leap Motion Controller & Eye Tracker	Si	No	SI	No
Real-time Communication for Kinect-based Telerehabilitation (Antón, Kurillo, Goñi, Illarramendi, & Bajcsy, 2017)	No	Kinect	Si	Si	Si	No
Gesture imitation and recognition using Kinect sensor and extreme learning machines. (Emrehan, 2016)	Si	Kinect	Si	Si	Si	No
Reconocimiento en tiempo real de la lengua dactilológica colombiana basado en aprendizaje supervisado usando la cámara 3D RealSense (López, 2016)	Si	RealSense	Si	Si	Si	No
Toward realistic hands gesture interface: Keeping it simple for developers and machines (Eyal Krupka, 2017)	Si	RealSense & Kinect	Si	Si	Si	Si
Reconocimiento de movimientos de manos para la manipulación de una interfaz computacional	No	RealSense & Kinect	Si	No	No	Si

3.4 Comparación y selección del sensor

Debido a que el método seleccionado es el SDK del Proyecto Praga solo se puede recurrir al uso de tres sensores diferentes (Tabla 3).

Tabla 3. Sensores soportados (Microsoft, 2017).

Marca	Rango de detección de gestos	Experiencia
Intel® RealSense™ SR300	20-60 [cm]	Mejor
Intel® RealSense™ F200	20-60 [cm]	Buena
Kinect para Windows v2	60-110 [cm]	Buena

3.4.1 Intel® RealSense™ SR300.

La cámara Intel® RealSense™ SR300 integra un subsistema de audio de doble micrófono, proporciona una cámara a todo color de 1080p a 30 fotogramas por segundo (Por sus siglas en inglés FPS) y ofrece una distancia óptima entre 0.2 m y 1.5 m para una mejor percepción de la profundidad (Figura 16).

Características:

- Los gestos con las manos pueden usarse para múltiples propósitos de interfaz de usuario.
- Capacidad de escaneo 3D que reconstruye la forma y la apariencia de un objeto del mundo real en una malla triangular en 3D.
- Una función de segmentación de fondo que elimina o reemplaza la imagen detrás de la cabeza y el hombro del usuario.
- Capacidad de seguimiento y reconocimiento facial que identifica la presencia del usuario y las características faciales, admitiendo hasta 78 puntos de referencia para una alta precisión y reconocimiento de expresiones.

Figura 16. Intel® RealSense™ SR300 (Intel, 2018).

3.4.2 Kinect para Windows v2

Kinect permite a los usuarios controlar e interactuar con una consola de video juegos sin necesidad de tener contacto físico con un control tradicional, mediante una interfaz natural de usuario que reconoce gestos, comandos de voz, y objetos e imágenes (Figura 17).

Características:

- Cámara que puede ver en 3D.
- Una videocámara tradicional.
- Arreglo de 4 micrófonos.
- Reconoce hasta 6 esqueletos activos.
- Reconocimiento de voz.
- Conjunto de APIs multiplataforma.
- Plugin de Unity.

Figura 17. Kinect para Windows v2 (Microsoft, 2018).

Capítulo IV

Metodología de solución

Capítulo 4. Metodología de solución

En la figura 18 se describen las fases que comprenden la metodología de solución para el presente tema de tesis, así como cada una de las actividades contenidas en cada fase.

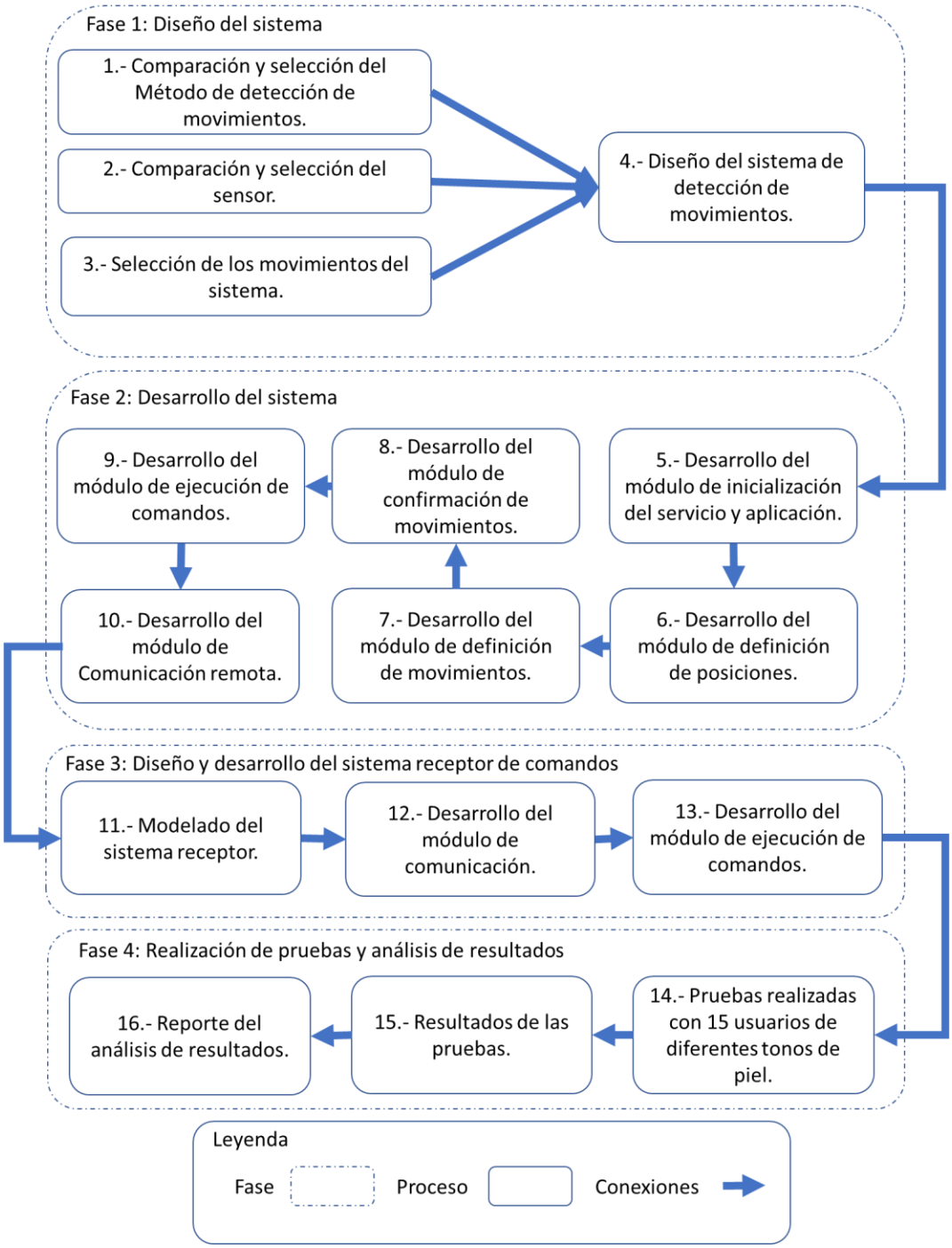

Figura 18. Metodología de solución.

4.1 Fase 1: Diseño del sistema

La finalidad de esta fase comprende el diseño del sistema de detección de movimientos de la mano derecha, para este sistema se investigaron distintos métodos y dispositivos capaces de realizar dicha detección.

4.1.2 Comparación y selección del método de detección de movimientos.

Las metodologías comparadas se muestran a continuación y se encuentran presentadas de en el capítulo 3:

a) Método utilizado en (Han & Rashid, 2016)
b) Método utilizado en (Tchoketch Kenbir & Bouhedda, 2016)
c) Método utilizado en (Emrehan, 2016)
d) Método utilizado en (Sueaseenak & Khawdee, 2017)
e) Método utilizado en (Gunawardane, 2016)
f) Método utilizado en (López, 2016)
g) Método propuesto por (Eyal Krupka, 2017)

4.1.2.1 Método seleccionado.

Para determinar el método a utilizar se tomó en cuenta la cantidad de procedimientos realizados, la facilidad de registrar diferentes señas o posiciones, así como la eficiencia del método.

El método seleccionado es el propuesto por (Eyal Krupka, 2017) ya que este es un método que es capaz de detectar movimientos de las manos, no requiere crear una base de datos, no es necesario llevar a cabo un entrenamiento del sistema, permite definir las posiciones de la mano a través de una descripción con el uso del Lenguaje de Definición de Gestos y según las investigaciones es un método que obtiene resultados de eficiencia arriba del 80%, además, con este método el desarrollo de futuros sistemas no requerirá de una gran experiencia en temas de visión por computadora.

Este método consiste en llevar a cabo los siguientes pasos.

1) Describir las posiciones de las manos.
2) Asignar un conjunto de dos a cinco posiciones para cada gesto.
3) Iniciar el sensor de captura de imágenes.
4) Iniciar el proceso de reconocimiento de gestos.
5) Ejecutar las instrucciones asignadas a los gestos.

4.1.3 Comparación y selección del sensor.

Tomando en cuenta las características de los sensores soportados y las recomendaciones del Proyecto Praga, se decidió hacer uso del sensor Kinect v2 y del sensor SR300 para este trabajo, así de esta manera se puede tener un reporte del funcionamiento de ambos para futuras investigaciones y desarrollos.

Nota: Debido a que el sensor Intel® RealSense™ F200 se encuentra descontinuado, se omite en la investigación y el uso de este puesto que el sensor SR300 es la mejora a dicho sensor.

4.1.4 Selección de los movimientos del sistema.

Para el desarrollo de este proceso se realizó la definición de posiciones de la mano que deberían ser reconocidos por el sistema a desarrollar. En un inicio se tomó como base algunas de las posiciones de la mano utilizadas en el LSM (Lenguaje de Señas Mexicano), las señas (posiciones) utilizadas se muestran en la Tabla 4.

Tabla 4. Posiciones de la mano.

Posición	Significado	Posición	Significado
	Letra A		Letra F
	Letra B		Letra G
	Letra C		Letra H
	Letra E		Letra I

Las definiciones de las posiciones de la mano son programadas para ser ejecutados sobre lenguaje C# utilizando las librerías proporcionadas por el Proyecto Praga, para esto se debe tomar en cuenta que la posición de la mano que está definida por cuatro características que se muestran en la figura 19.

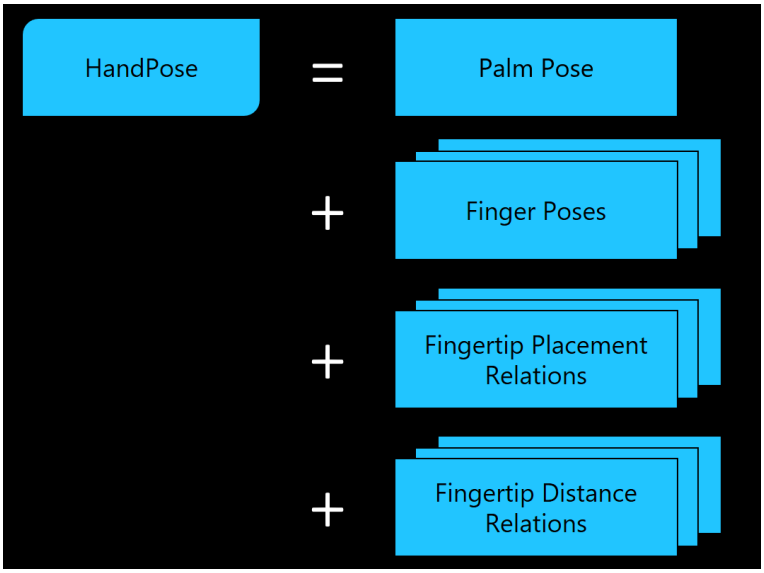

Figura 19. Definición de la posición de la mano (Microsoft, 2017).

La posición de la palma (Palm Pose) está definida por la **Dirección** y la **Orientación** en dónde cada una de ellas puede tomar distintos valores como:

1. Izquierda
2. Derecha
3. Arriba
4. Abajo
5. Adelante
6. Atrás

Por ejemplo, la posición de la palma puede ser definida como:

```
new PalmPose(new AnyHandContext(), PoseDirection.Down | PoseDirection.Forward)
```

La posición de los dedos (Finger Poses) está definida por la **Dirección** y la **Flexión** de los dedos de la mano en donde la dirección puede tomar valores como:

1. Izquierda
2. Derecha
3. Arriba
4. Abajo
5. Adelante
6. Atrás

Mientras que la flexión de los dedos puede tomar valores como:

1. Abierto
2. Abierto estirado

3. Doblado
4. Doblado metido

Por ejemplo, de la definición de la posición de los dedos es la siguiente:

```
new FingerPose(new[] { Finger.Thumb, Finger.Index,
Finger.Middle, Finger.Ring }, FingerFlexion.Open)
```

Las relaciones entre la colocación de la punta del dedo (Fingertip Placement Relations) están definidas por la **Ubicación Relativa**, la cual puede tomar valores como:

1. Izquierda
2. Derecha
3. Encima
4. Abajo
5. Detrás
6. Al frente

Por ejemplo, la definición de las relaciones entre la colocación de la punta de los dedos es la siguiente:

```
new FingertipPlacementRelation(Finger.Index,
relativePlacement.Above, Finger.Thumb)
```

Por último, las relaciones entre la distancia de los dedos (Fingertip Distance Relations) que están definidas por la distancia relativa que puede tomar los siguientes valores:

1. Tocando
2. Sin tocar

Por ejemplo, la definición para las relaciones entre la distancia de los dedos es la siguiente:

```
new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb)
```

Tomando en cuenta estas definiciones se generó el código para cada una de las 8 posiciones de la mano a detectar, a continuación, se muestran las posiciones y el código generado para hacer la detección.

Figura 20. Posición para la letra A.

El siguiente código fue generado para la posición de la letra A (Figura 20).

```
var letra_a = new HandPose("LetraA",
                new FingerPose(new[] { Finger.Index, Finger.Middle,
Finger.Ring, Finger.Pinky }, FingerFlexion.FoldedTucked),
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
                  new FingerPose(Finger.Thumb, FingerFlexion.Open,
                        PoseDirection.Left));
```

Figura 21. Posición para la letra B.

El siguiente código fue generado para la posición de la letra B (Figura 21).

```
var letra_b = new HandPose("LetraB",
                new FingerPose(Finger.Thumb,
FingerFlexion.FoldedTucked),
                new FingerPose(new[] { Finger.Index,
Finger.Middle, Finger.Ring, Finger.Pinky }, FingerFlexion.Open),
                new FingertipDistanceRelation (Finger.Index,
RelativeDistance.Touching, Finger.Middle),
                new FingertipDistanceRelation (Finger.Middle,
RelativeDistance.Touching, Finger.Ring),
                new FingertipDistanceRelation(Finger.Ring,
RelativeDistance.Touching, Finger.Pinky),
                new       FingertipPlacementRelation(Finger.Middle,
RelativePlacement.Above, Finger.Thumb));
```

Figura 22. Posición para la letra C.

El siguiente código fue generado para la posición de la letra C (Figura 22).

```
var letra_c = new HandPose("letraC",
```

```
                new FingerPose(new[] { Finger.Thumb, Finger.Index,
Finger.Middle, Finger.Ring, Finger.Pinky }, FingerFlexion.Open,
PoseDirection.Left),
                new FingertipPlacementRelation(Finger.Index,
RelativePlacement.Above, Finger.Thumb),
                new         FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb));
```

Figura 23. Posición para la letra E.

El siguiente código fue generado para la posición de la letra E (Figura 23).

```
var letra_e = new HandPose("letraE",
                new       FingerPose(new       AllFingersContext(),
FingerFlexion.FoldedTucked) );
```

Figura 24. Posición para la letra F.

El siguiente código fue generado para la posición de la letra F (Figura 24).

```
var letra_f = new HandPose("LetraF",
                new FingerPose(new AllFingersContext(),
FingerFlexion.Open),
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.Touching, Finger.Thumb),
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Middle),
                new FingertipDistanceRelation(Finger.Middle,
RelativeDistance.Touching, Finger.Ring),
                new          FingertipDistanceRelation(Finger.Ring,
RelativeDistance.Touching, Finger.Pinky));
```

Figura 25. Posición para la letra G.

El siguiente código fue generado para la posición de la letra G (Figura 25).

```
var letra_g = new HandPose("LetraG",
                new FingerPose(Finger.Index, FingerFlexion.Open,
PoseDirection.Left),
                new FingerPose(Finger.Thumb, FingerFlexion.Open,
PoseDirection.Up),
                new FingerPose(new[] {Finger.Middle, Finger.Ring,
Finger.Pinky}, FingerFlexion.Folded));
```

Figura 26. Posición para la letra H.

El siguiente código fue generado para la posición de la letra H (Figura 26).

```
var letra_h = new HandPose("LetraH",
                new FingerPose(new[] {
Finger.Index,Finger.Middle}, FingerFlexion.Open,
PoseDirection.Left),
                new FingerPose(Finger.Thumb, FingerFlexion.Open,
PoseDirection.Up),
                     new      FingerPose(new[]      {Finger.Ring,
Finger.Pinky }, FingerFlexion.Folded));
```

Figura 27. Posición de la letra I.

El siguiente código fue generado para la posición de la letra I (Figura 27).

```
var letra_i = new HandPose("letraI",
                new FingerPose(new[] { Finger.Thumb,
Finger.Middle, Finger.Ring, Finger.Index }, FingerFlexion.Folded),
                new FingerPose(Finger.Pinky, FingerFlexion.Open));
```

Después de probar las señas ingresadas se determinó que un movimiento de la mano es la suma de 2 o más posiciones detectadas, por lo tanto, para poder realizar la detección de 4 movimientos de las manos se tuvieron que definir 8 nuevas posiciones de la mano.

La idea principal de manipulación es el control de algunas funciones de la aplicación Spotify de Windows 10, esta aplicación es un reproductor de música. Por esto los movimientos se definieron como: **Adelante, Atrás, Cerrar y Play/Pause**, las posiciones definidas se muestran a continuación.

Figura 28. Posición AdelanteA.

El siguiente código fue generado para la posición AdelanteA (Figura 28).

```
var AdelanteA = new HandPose("AdelanteA",
                new FingerPose(Finger.Thumb, FingerFlexion.Open,
PoseDirection.Left),
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
                new      FingerPose(new[]      {      Finger.Index,
Finger.Middle,         Finger.Ring,         Finger.Pinky         },
FingerFlexion.Folded));
```

Figura 29. Posición AdelanteB.

El siguiente código fue generado para la posición AdelanteB (Figura 29).

```
var AdelanteB = new HandPose("AdelanteB",
                new FingerPose(Finger.Thumb, FingerFlexion.Open,
PoseDirection.Up),
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
                new      FingerPose(new[]      {      Finger.Index,
Finger.Middle,         Finger.Ring,         Finger.Pinky         },
FingerFlexion.Folded));
```

Figura 30. Posición AtrasA.

El siguiente código fue generado para la posición AtrasA (Figura 30).

```
var AtrasA = new HandPose("AtrasA",
                new FingerPose( Finger.Thumb, FingerFlexion.Open,
PoseDirection.Up),
                new FingerPose(Finger.Pinky, FingerFlexion.Open,
PoseDirection.Forward),
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
                new FingerPose(new[] { Finger.Index,
Finger.Middle, Finger.Ring }, FingerFlexion.Folded));
```

Figura 31. Posición AtrasB.

El siguiente código fue generado para la posición AtrasB (Figura 31).

```
var AtrasB = new HandPose("AtrasB",
                new FingerPose(Finger.Thumb, FingerFlexion.Open,
PoseDirection.Left),
                new FingerPose(Finger.Pinky, FingerFlexion.Open),
```

```
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
                new      FingerPose(new[]      {      Finger.Index,
Finger.Middle, Finger.Ring }, FingerFlexion.Folded));
```

Figura 32. Posición CerrarA.

El siguiente código fue generado para la posición CerrarA (Figura 32).

```
var CerrarA = new HandPose("CerrarA",
                new FingerPose(new[]{Finger.Index, Finger.Ring,
Finger.Pinky}, FingerFlexion.Open),
                new FingerPose( Finger.Middle, FingerFlexion.Open,
PoseDirection.Up),
                new FingertipDistanceRelation ( Finger.Index,
RelativeDistance.Touching, Finger.Middle),
                new FingertipDistanceRelation(Finger.Middle,
RelativeDistance.Touching, Finger.Ring),
                new FingertipDistanceRelation(Finger.Ring,
RelativeDistance.Touching, Finger.Pinky));
```

Figura 33. Posición CerrarB.

El siguiente código fue generado para la posición CerrarB (Figura 33).

```
var CerrarB = new HandPose("CerrarB",

                new FingerPose(new[] { Finger.Index, Finger.Ring,
Finger.Pinky }, FingerFlexion.Open),
```

```
                new FingerPose(Finger.Middle, FingerFlexion.Open,
PoseDirection.Down),
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.Touching, Finger.Middle),
                new FingertipDistanceRelation(Finger.Middle,
RelativeDistance.Touching, Finger.Ring),
                new          FingertipDistanceRelation(Finger.Ring,
RelativeDistance.Touching, Finger.Pinky));
```

Figura 34. Posición PPA.

El siguiente código fue generado para la posición PPA (Figura 34).

```
var PPA = new HandPose("PPA",
            new FingerPose(Finger.Thumb, FingerFlexion.Open,
PoseDirection.Up),
            new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
            new FingerPose(new[] { Finger.Index, Finger.Middle,
Finger.Ring, Finger.Pinky }, FingerFlexion.Folded,
PoseDirection.Right));
```

Figura 35. Posición PPB.

El siguiente código fue generado para la posición PPB (Figura 35).

```
var PPB = new HandPose("PPB",
          new         FingerPose(new         AllFingersContext(),
FingerFlexion.Folded));
```

4.1.5 Diseño del sistema

En este proceso se generaron distintos diagramas que describen el contenido y funcionamiento del sistema. En la figura 36 se puede observar la vista general del sistema, mientras que en las figuras 37, 38 y 39 se describen módulos del sistema de una manera particular.

Descripción de la figura 36:

- **Módulo de iniciación del servicio y aplicación**: Este módulo es el encargado de realizar el arranque del servicio de detección de movimientos y de ejecutar el reproductor de música, en caso de que alguno no pueda ser iniciado, el sistema envía un mensaje de error y lo finaliza.
- **Módulo de definición de posiciones de la mano**: Una vez recibida la confirmación de inicio, el sistema almacena en variables las 8 posiciones diferentes para la generación de los 4 movimientos.
- **Módulo de definición de movimientos**: Teniendo las 8 posiciones de la mano registradas, el sistema genera 4 movimientos a partir de las entradas A+B de cada movimiento y los almacena en una variable cada uno.
- **Módulo de confirmación de movimientos**: El sistema recibe como entrada los datos provenientes del sensor de profundidad y compara las posiciones recibidas con los movimientos almacenados, si existe una coincidencia envía el comando correspondiente.
- **Módulo de ejecución de comandos**: El módulo recibe el comando proveniente del movimiento detectado y lo ejecuta para controlar la aplicación multimedia. **Nota**: Este módulo se desactiva si el módulo de envío de comandos se encuentra activo.
- **Módulo de envío de comandos**: El módulo recibe el comando proveniente del movimiento detectado y lo envía a un equipo remoto para ser ejecutado a través de una aplicación receptora.

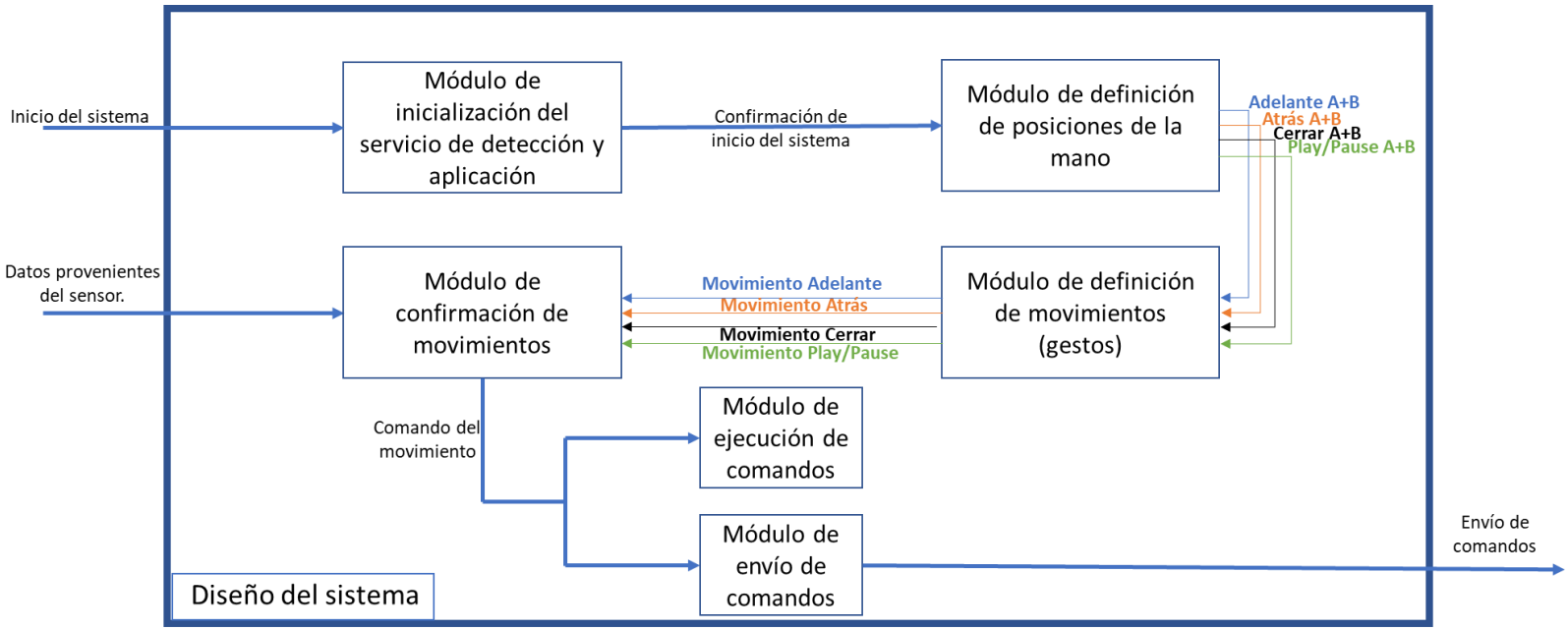

Figura 36. Diseño del sistema.

Descripción de la figura 37:

- **Inicialización del servicio de detección**: Este módulo se encarga de buscar y ejecutar los servicios del SDK del Proyecto Praga, en caso de existir un error el sistema manda un mensaje y finaliza la aplicación.
- **Inicialización de la aplicación multimedia**: Este módulo se encarga de ejecutar la aplicación multimedia Spotify y en caso de no existir o no poder ejecutarla, el sistema manda un mensaje y finaliza la aplicación.
- **Confirmación de inicio**: Este módulo es el encargado de recibir la confirmación del inicio del servicio, así como de la aplicación, este módulo envía la confirmación de inicio si esta es verdadera.

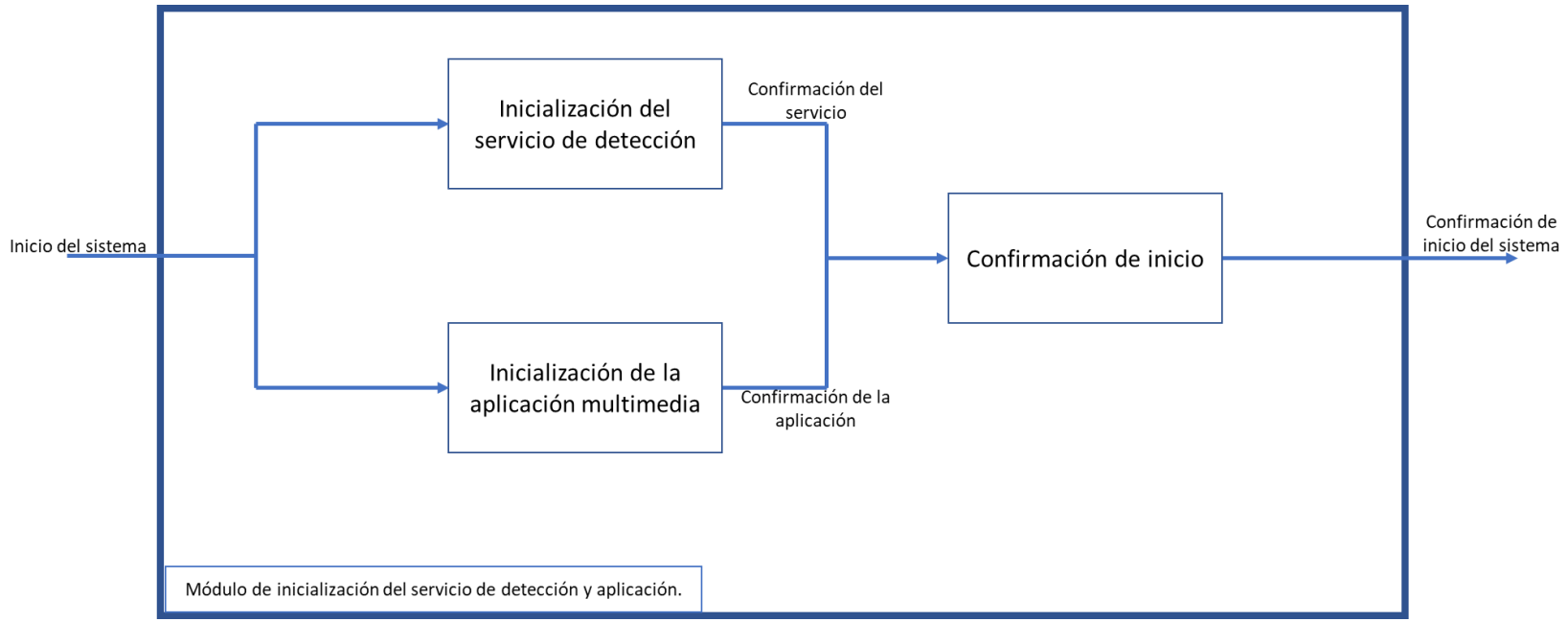

Figura 37. Módulo de inicialización del servicio de detección y aplicación.

Descripción de la figura 38:

- **Definición de la posición AdelanteA**: En este proceso se define la posición inicial del movimiento Adelante.
- **Definición de la posición AdelanteB**: En este proceso se define la posición final del movimiento Adelante.
- **Definición de la posición AtrásA**: En este proceso se define la posición inicial del movimiento Atrás.
- **Definición de la posición AtrásB**: En este proceso se define la posición final del movimiento Atrás.
- **Definición de la posición CerrarA**: En este proceso se define la posición inicial del movimiento Cerrar.
- **Definición de la posición CerrarB**: En este proceso se define la posición final del movimiento Cerrar.
- **Definición de la posición Play/PauseA**: En este proceso se define la posición inicial del movimiento Play/Pause.
- **Definición de la posición Play/PauseB**: En este proceso se define la posición final del movimiento Play/Pause.

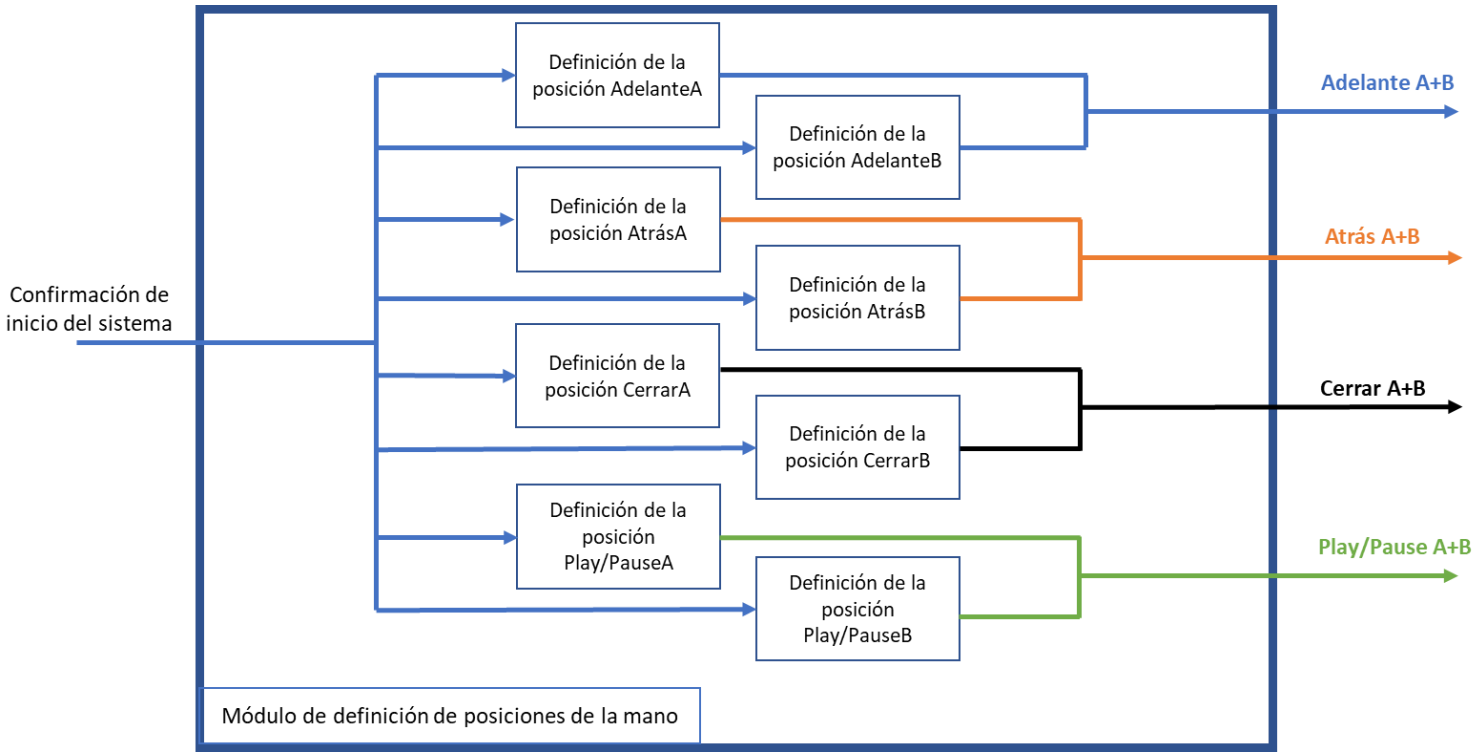

Figura 38. Módulo de definición de posiciones de la mano.

Descripción de la figura 39:

- **Definición del movimiento Adelante**: En este proceso se define un movimiento tomando las posiciones definidas (posición A como inicio del movimiento y posición B como fin del movimiento).
- **Definición del movimiento Atrás**: En este proceso se define un movimiento tomando las posiciones definidas (posición A como inicio del movimiento y posición B como fin del movimiento).
- **Definición del movimiento Cerrar**: En este proceso se define un movimiento tomando las posiciones definidas (posición A como inicio del movimiento y posición B como fin del movimiento).

- **Definición del movimiento Play/Pause**: En este proceso se define un movimiento tomando las posiciones definidas (posición A como inicio del movimiento y posición B como fin del movimiento).

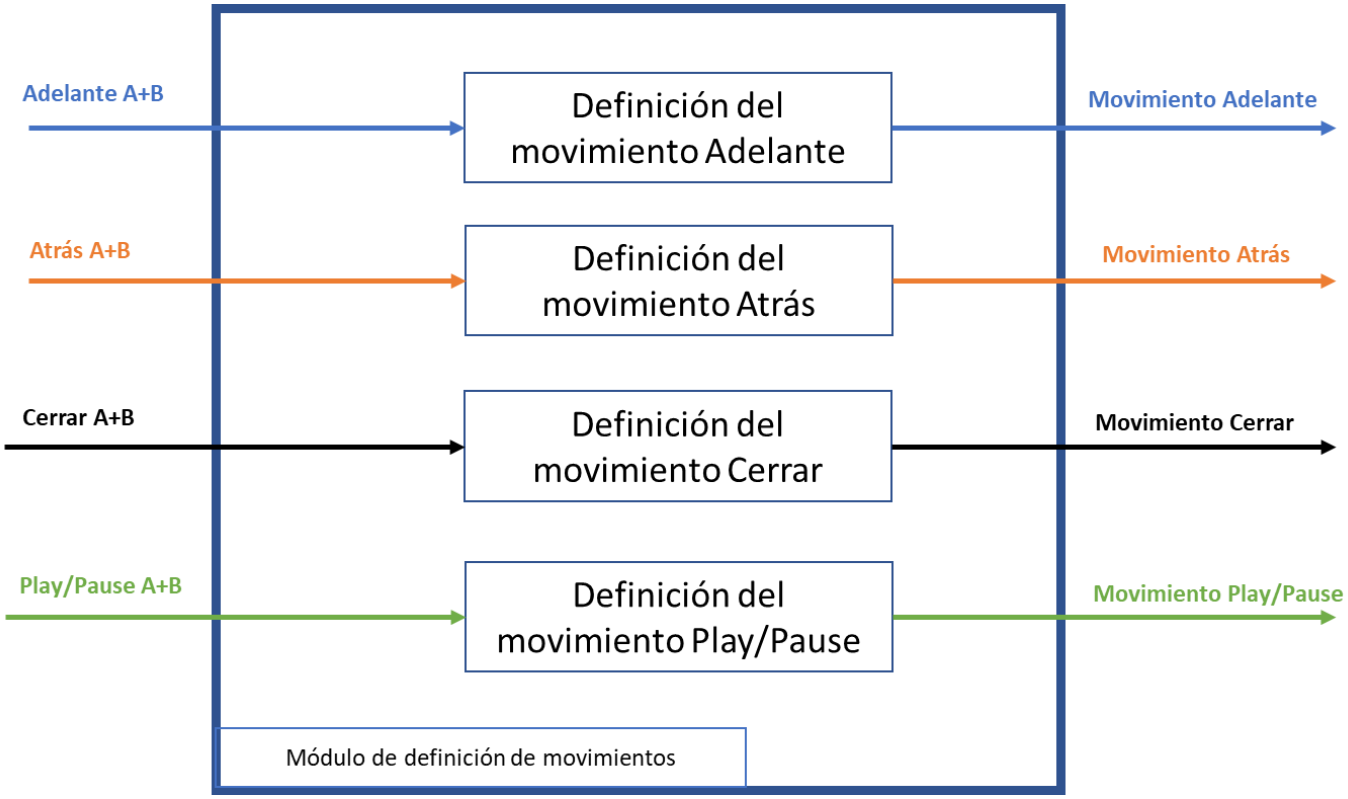

Figura 39. Módulo de definición de movimientos.

4.2 Fase 2: Desarrollo del sistema

La finalidad de esta fase comprende el desarrollo de todos los módulos descritos en el diseño del sistema, estos módulos fueron programados en el lenguaje C# utilizando el entorno de desarrollo Visual Studio 2017 sobre la plataforma Windows 10 (Figura 40).

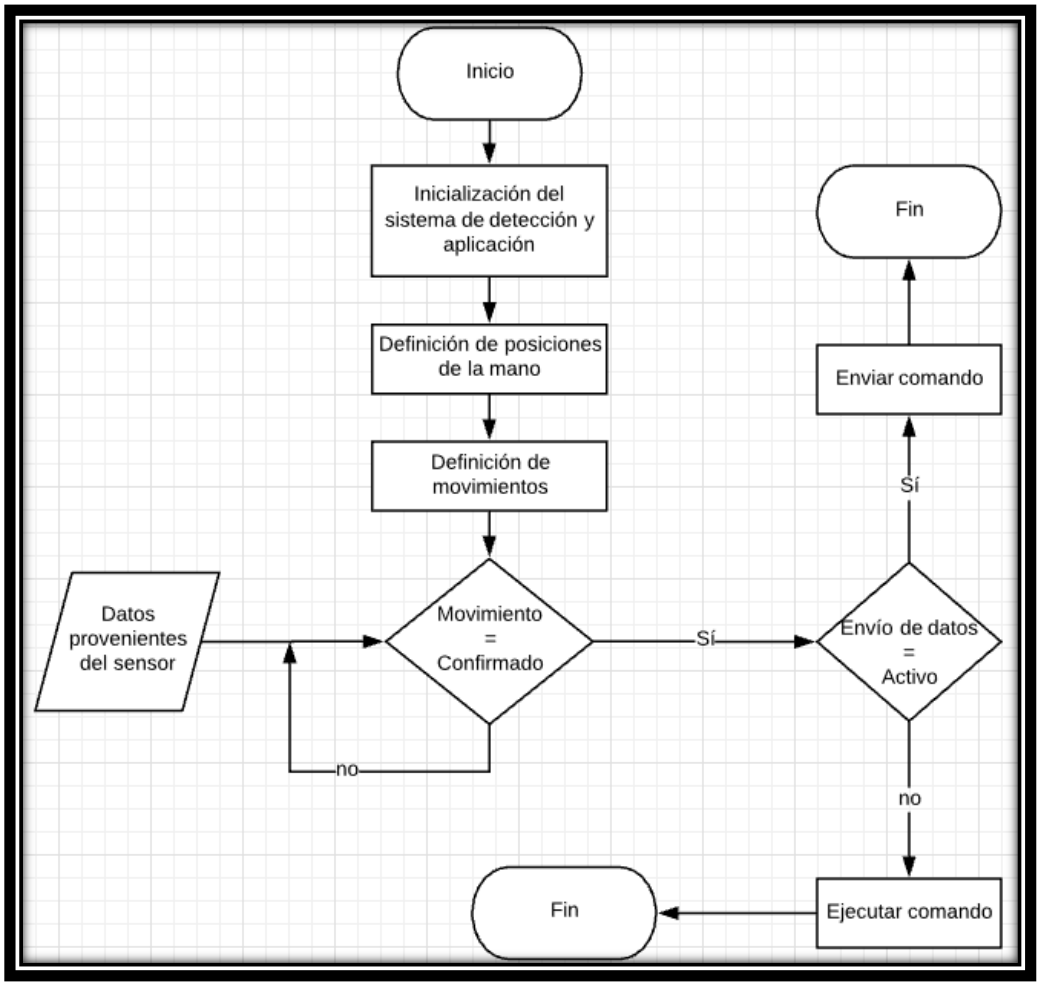

Figura 40. Diagrama de flujo del sistema de detección de movimiento.

4.2.1 Desarrollo del módulo de inicialización de servicio y aplicación.

Para desarrollar este módulo se escribieron instrucciones que ejecutan el servicio del SDK del proyecto Praga y la aplicación Spotify. El módulo escrito es el siguiente:

```
public MainWindow()
        {
            MainWindow ventana = this;
            //Iniciar servicio del sensor
            ProcessStartInfo Servicio = new ProcessStartInfo();
            Servicio.UseShellExecute = true;
            Servicio.FileName = "Microsoft.Gestures.Service.exe";
```

```
            Servicio.WorkingDirectory = @"C:\Users\Luis
Salgado\AppData\Roaming\Microsoft\Prague\PragueVersions\LatestVers
ion";
            Process.Start(Servicio);
            Thread.Sleep(3000);
            Process.Start(Servicio);
            Thread.Sleep(1000);
            InitializeComponent();
            Dinamic();
        }
```

4.2.2 Desarrollo del módulo de definición de posiciones.

En este módulo se escribieron las definiciones de las 8 posiciones de la mano, se utilizó el Lenguaje de Descripción de Gestos. El módulo escrito es el siguiente:

```
      //Definición de las posiciones de la mano

            //Avanzar

            var AdelanteA = new HandPose("AdelanteA",
                new FingerPose(Finger.Thumb, FingerFlexion.Open,
PoseDirection.Left),
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
                new FingerPose(new[] { Finger.Index,
Finger.Middle, Finger.Ring, Finger.Pinky },
FingerFlexion.Folded));
            var AdelanteB = new HandPose("AdelanteB",
                new FingerPose(Finger.Thumb, FingerFlexion.Open,
PoseDirection.Up),
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
                new FingerPose(new[] { Finger.Index,
Finger.Middle, Finger.Ring, Finger.Pinky },
FingerFlexion.Folded));

            //<--Avanzar

            //Retroceder

            var AtrasA = new HandPose("AtrasA",
                new FingerPose( Finger.Thumb, FingerFlexion.Open,
PoseDirection.Up),
                new FingerPose(Finger.Pinky, FingerFlexion.Open,
PoseDirection.Forward),
```

```
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
                new FingerPose(new[] { Finger.Index,
Finger.Middle, Finger.Ring }, FingerFlexion.Folded));
            var AtrasB = new HandPose("AtrasB",
                new FingerPose(Finger.Thumb, FingerFlexion.Open,
PoseDirection.Left),
                new FingerPose(Finger.Pinky, FingerFlexion.Open),
                new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
                new FingerPose(new[] { Finger.Index,
Finger.Middle, Finger.Ring }, FingerFlexion.Folded));

            //<--Retroceder

            //Cerrar app
            var CerrarA = new HandPose("CerrarA",
                new FingerPose(n
            //Play Pause

            var PPA = new HandPose("PPA",
            new FingerPose(Finger.Thumb, FingerFlexion.Open,
PoseDirection.Up),
            new FingertipDistanceRelation(Finger.Index,
RelativeDistance.NotTouching, Finger.Thumb),
            new FingerPose(new[] { Finger.Index, Finger.Middle,
Finger.Ring, Finger.Pinky }, FingerFlexion.Folded,
PoseDirection.Right));

            var PPB = new HandPose("PPB",
           new FingerPose(new AllFingersContext(),
FingerFlexion.Folded));

            //<--Play Pause

            //<--Definición de las posiciones de la mano
```

4.2.3 Desarrollo del módulo de definición de movimientos

Para la definición de los gestos es necesario almacenar cada uno de ellos en variables, dando como parámetros el nombre y el orden de las posiciones que conforman el gesto. El módulo escrito es el siguiente:

```
          //Definición de los gestos
             var Adelante = new Gesture("Adelante", AdelanteA,
AdelanteB);
             var Atras = new Gesture("Atras", AtrasA, AtrasB);
             var Cerrar = new Gesture("Cerrar", CerrarA, CerrarB);
             var PlayPause = new Gesture("PlayPause", PPA, PPB);
             //<--Definición de los gestos
```

4.2.4 Desarrollo del módulo de confirmación de movimientos

El módulo de confirmación de movimientos activa el sensor de profundidad y comienza a recibir los datos de entrada, queda a la espera de detectar alguno de los gestos almacenados en las variables. El módulo escrito es el siguiente:

```
             //Activación del servicio de detección
             var gesturesService =
GesturesServiceEndpointFactory.Create();
             await gesturesService.ConnectAsync();

             //Detección del gesto
             await gesturesService.RegisterGesture(Adelante);
             await gesturesService.RegisterGesture(Atras);
             await gesturesService.RegisterGesture(Cerrar);
             await gesturesService.RegisterGesture(PlayPause);
             //<--Detección del gesto
```

4.2.5 Desarrollo del módulo de ejecución de comandos

Este módulo se encarga de ejecutar los comandos correspondientes al movimiento detectado, se utiliza la librería ImputSimulator para simular el teclado multimedia a través de código de programación en C#, además fue agregado un botón para salir de la aplicación. El módulo escrito es el siguiente:

```
//Ejecución de comandos dinamicos
             //Avanzar
             Adelante.Triggered += (sender, args) =>
             {

InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_NEXT_TRACK);

             };
             //retroceder
             Atras.Triggered += (sender, args) =>
             {

InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_PREV_TRACK);

             };
             //AbrirApp
```

```
            Cerrar.Triggered += (sender, args) =>
            {

                try
                {
                    Process[] proc =
Process.GetProcessesByName("Microsoft.Gestures.Service");
                    proc[0].Kill();
                }
                catch (Exception ex)
                {
                    MessageBox.Show("¡¡No se pudo detener el
servicio!!" + "  Error: " + ex);
                }
                Thread.Sleep(1500);
                System.Environment.Exit(0);

            };

            //Play Pause
            PlayPause.Triggered += (sender, args) =>
            {

InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_PLAY_PAUSE);
            };
            //<--Activacion del servicio de detección

            Salir.Click += Salir_Click;

        } //<--Run program

    private void Salir_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            Process[] proc =
Process.GetProcessesByName("Microsoft.Gestures.Service");
            proc[0].Kill();
        }
        catch (Exception ex)
        {
            MessageBox.Show("¡¡No se pudo detener el servicio!!" +
"  Error: " + ex);
        }
        Thread.Sleep(1500);        System.Environment.Exit(0);
}
```

4.2.6 Desarrollo del módulo de envío de comandos

Este módulo fue desarrollado para enviar los comandos a una aplicación receptora con una dirección IP fija a través de una conexión TCP. El módulo escrito es el siguiente:

```
        void SendData(int data)
          {
              String Comand = "";
              switch (data)
              {
                  case 1:
                      Comand = "Adelante";
                          break;

                  case 2:
                      Comand = "Atras";
                      break;

                  case 3:
                      Comand = "Cerrar";
                      break;

                  case 4:
                      Comand = "PP";
                      break;
              }
              try
              {
                  TcpClient Cliente = new TcpClient();
                  Cliente.Connect("127.0.0.1", 80);

                  Stream stream = Cliente.GetStream();
                  ASCIIEncoding ascii = new ASCIIEncoding();
                  byte[] enviar = ascii.GetBytes(Comand);
                  // Thread.Sleep(2000);
                  stream.Write(enviar, 0, enviar.Length);
                  //MessageBox.Show("Mensaje enviado");
                  Cliente.Close();

              }
              catch (Exception)
              {
                  MessageBox.Show("No se puede establecer la
conexion con el host remoto");

              }
```

4.3 Fase 3: Diseño y desarrollo del sistema receptor de comandos

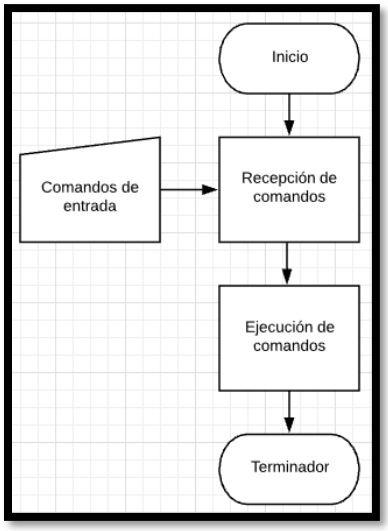

Figura 41. Modelo del sistema receptor de comandos.

Esta fase comprende el diseño de una aplicación (Figura 41) que se ejecuta sobre el sistema Windows 10 y que se encarga de recibir comandos provenientes del sistema de detección de movimiento y ejecutar acciones sobre la aplicación Spotify.

4.3.1 Modelado del sistema receptor.

El sistema receptor es una aplicación sencilla que únicamente recibe los comandos y ejecuta instrucciones de control sobre el sistema Windows 10.

Descripción de la figura 42:

- **Módulo de recepción de comandos**: Este módulo es el encargado de recibir comandos de control provenientes de un equipo remoto a través de una conexión de red.
- **Módulo de ejecución de comandos**: El módulo recibe el comando proveniente del movimiento detectado y lo ejecuta para controlar la aplicación multimedia.

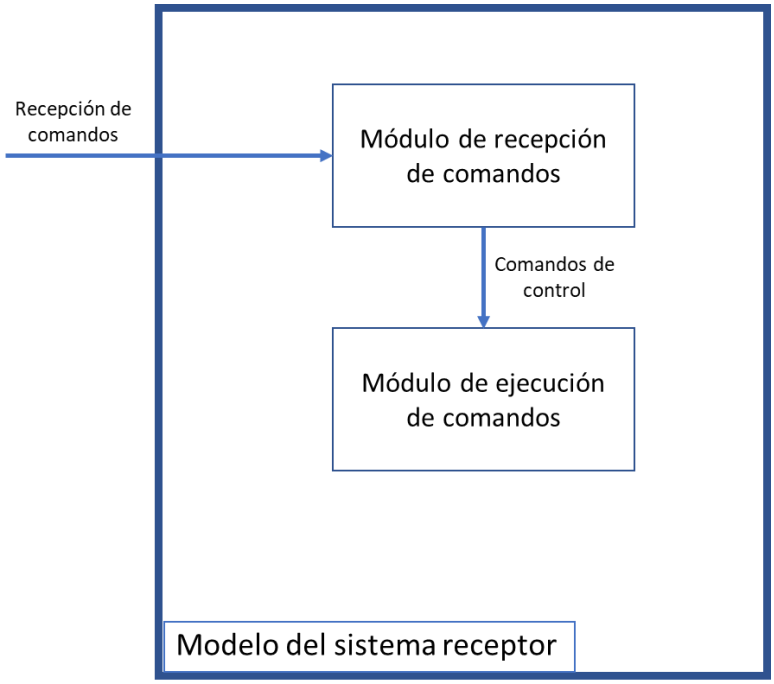

Figura 42. Modelo del sistema receptor.

4.3.2 Desarrollo del módulo de comunicación.

En este proceso se desarrolló un módulo de comunicación capaz de recibir los datos provenientes del sistema de detección de movimiento, se establece una comunicación a través de una conexión remota usando el protocolo TCP y envía los comandos al módulo de ejecución. El módulo escrito es el siguiente:

```
void Recibir()
        {
            String Command = "";

            try
            {
                IPAddress ip = IPAddress.Parse("192.168.43.27");
                TcpListener listener = new TcpListener(ip, 80);
                listener.Start();

                Socket socket = listener.AcceptSocket();
                 byte[] bit = new byte[100];
                 int i = socket.Receive(bit);
                 for (int a = 0; a < i; a++)
                 {
                     Command = Command + Convert.ToChar(bit[a]);
                 }

                ejecutar(command);
```

```
                socket.Close();
                listener.Stop();
                Recibir();
            }
            catch (Exception)
            {
                MessageBox.Show("Error de conexion");
            }
```

4.3.3 Desarrollo del módulo de ejecución de comandos.

En este proceso se desarrolló un módulo de ejecución de comandos para el sistema receptor. El módulo escrito es el siguiente:

```
void ejecutar(command) {
                    switch (Command)
                    {
                        case "Adelante":

InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_NEXT_TRACK);
                            break;

                        case "Atras":

InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_PREV_TRACK);
                            break;
                        case "PP":

InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_PLAY_PAUSE);
                        case "Cerrar":
                            System.Environment.Exit(0);
break;                  }               }
```

4.4 Ambientes de trabajo de los sistemas.

El ambiente al que está dirigido el sistema de detección de movimientos debe contar con las siguientes características.

a) Computadora con sistema Windows 10.
b) Procesadores Intel Core i5 de cuarta generación o superior.
c) Memoria Ram 4 Gb o superior.
d) SDK Proporcionado por el Proyecto Praga.
e) 5 Gb de espacio de almacenamiento disponible.
f) Sensor de profundidad Kinect v2 o Intel RealSense SR300 (Preferentemente).

El ambiente en el que se trabajo es el siguiente.

a) Computadora con sistema operativo Windows 10.
b) Procesador Intel Core i7 de sexta generación.
c) Memoria Ram de 8 Gb.
d) SDK del Proyecto Praga.
e) Espacio de almacenamiento mayor a 5 Gb.
f) Sensor de profundidad Kinect v2 y Sensor Intel RealSense SR300.
g) Conexión de red.

El ambiente al que está dirigido el sistema de recepción de comandos debe contar con las siguientes características.

a) Sistema operativo Windows 10.
b) Procesador Core i3 de cuarta generación o superior.
c) Memoria Ram de 4 Gb.
d) 100 Mb de espacio de almacenamiento.
e) Conexión de red.

El ambiente en el que se trabajo es el siguiente.

a) Sistema operativo Windows 10.
b) Procesador Core i5 de quinta generación.
c) Memoria Ram de 4 Gb.
d) Espacio de almacenamiento mayor a 100 Mb.
e) Conexión de red.

Capítulo V

Pruebas y resultados

Capítulo 5. Pruebas y resultados

En este capítulo se dan a conocer las pruebas realizadas y los resultados obtenidos a través de su ejecución.

5.1 Fase 4: Realización de pruebas y análisis de resultados.

En esta fase se realizaron las pruebas en el sistema de reconocimiento de movimientos de manos, además una evaluación de los resultados obtenidos.

Durante el desarrollo de la investigación se tuvo la necesidad de experimentar con más de un sensor de profundidad puesto que las investigaciones indicaban que existía una diferencia en la capacidad de detección entre distintos sensores. Se identificaron métodos y dispositivos capaces de reconocer movimientos de manos, de estos se seleccionó el **SDK del proyecto Praga** (Microsoft, Docs, 2018) como método de detección y los dispositivos **Intel® RealSense™ SR300** (Intel, Software Intel, 2018) y el sensor **Microsoft® Kinect V2** (Microsoft, Developer, 2018) y así poder obtener resultados comparables entre ellos.

5.1.2 Pruebas realizadas con 15 usuarios de diferentes tonos de piel.

El sistema fue probado por 15 usuarios de diferentes géneros, edades y colores de piel, en donde cada usuario realizó (10 ocasiones) un movimiento frente al Sensor Intel® RealSense™ SR300 (Intel, Software Intel, 2018), así como frente al sensor Microsoft® Kinect V2 (Microsoft, Developer, 2018) obteniendo un total de 150 pruebas realizadas por movimiento y por sensor.

Se decidió realizar la manipulación de una interfaz multimedia con la que se pudiera poner a prueba el funcionamiento del sistema de reconocimiento. Para cumplir los objetivos de la investigación se agregaron 4 movimientos, compuestos cada uno de ellos, de 2 posiciones de la mano. Los movimientos solo son identificados, si y solo si, la segunda posición de un movimiento es precedida por la primera posición.

Pruebas

En esta sección se muestra el proceso que se llevó a cabo para la realización de las pruebas tanto del sistema de reconocimiento de manos como el sistema de comunicación remota.

Sistema de reconocimiento de manos

Las pruebas fueron realizadas con 15 personas seleccionadas al azar, las cuales efectuaron los 4 movimientos bajo estudio, los movimientos fueron realizados en 10 ocasiones por cada uno de los usuarios dando un total de 600 movimientos llevados a cabo ante cada sensor (Figura 43).

Figura 43. Objetivo de movimientos realizados.

Cada movimiento se conforma de 2 posiciones:

1. Movimiento Adelante = Posición AdelanteA + Posición AdelanteB.
2. Movimiento Atrás = Posición AtrasA + Posición AtrasB.
3. Movimiento Cerrar = Posición CerrarA + Posición CerrarB.
4. Movimiento Play/Pause = Posición PPA + Posición PPB.

Las tablas 5 y 6 muestran las pruebas realizadas con 15 personas en las que los aspectos que se tomaron en cuenta fueron:

- Género.
- Tono de piel.
- Luminosidad.
- Movimiento (Número de Aciertos/Intentos).

Hubo la necesidad de identificar distintos colores de piel por lo cual se realizó una clasificación para los 15 usuarios, los colores de piel se clasificaron de la letra **A** a la letra **G** como se observa en la figura 44.

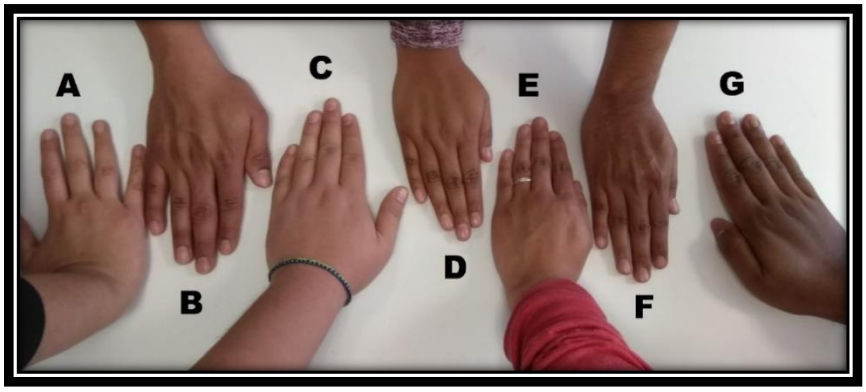

Figura 44. Muestra de los colores de piel.

Se tuvo la oportunidad de realizar las pruebas de baja luminosidad únicamente con 5 personas, esto por la disponibilidad de los usuarios (La figura 45 muestra una de las pruebas realizadas con luminosidad alta y la figura 46 muestra una de las pruebas con luminosidad baja).

Figura 45. Pruebas con luminosidad alta.

Figura 46. Pruebas con luminosidad baja.

Antes de realizar las pruebas, se calculó la distancia mínima y máxima de detección de cada uno de los sensores, y según la información proporcionada por el Proyecto Praga (Microsoft, Docs, 2018), los rangos de distancia permitidos para los sensores son:

- Intel® RealSense™ SR300 camera con un rango de 20-60 cm.
- Kinect V2 con un rango de 60-110 cm.

Sin embargo, al realizar las pruebas se determinó que los rangos permitidos en los sensores son:

- Intel® RealSense™ SR300 camera con un rango de 17-62 cm (5 cm mayor).
- Kinect V2 con un rango de 64-111 cm (3 cm menor).

Se les mostró a los usuarios cuales eran los movimientos que debían realizar para que los llevaran a cabo en 10 ocasiones cada uno.

Sistema de comunicación remota

Para realizar las pruebas del módulo de comunicación se crearon 2 aplicaciones (escritas en lenguaje C#); una aplicación que envía (Origen) los comandos de una computadora a otra y la otra aplicación es la encargada de recibir los comandos (Destino).

La aplicación origen encargada del envío de comandos tiene incluida un ciclo con el cual envía un comando en 20 ocasiones esperando 2 segundos entre cada comando. Por otro lado, la aplicación destino incluye la interpretación de los comandos recibidos, ambas aplicaciones cuentan con excepciones en caso de que exista algún fallo en la comunicación. Las aplicaciones se ejecutaron en 10 ocasiones obteniendo un total de **200 comandos enviados**, las aplicaciones se ejecutaron en 2 computadoras con sistema operativo **Windows 10** (Microsoft, Windows, 2018) y conectadas a una red WLAN con direcciones IP estáticas.

Tabla 5. Pruebas del módulo de comunicación.

Ejecución No.	Comandos enviados	Comandos recibidos
1	20	20
2	20	20
3	20	19
4	20	20
5	20	19
6	20	20
7	20	19
8	20	20
9	20	20
10	20	20
Total	200	197

La tabla 5 muestra que de los 200 comandos enviados por la aplicación origen, el 98.5% fueron recibidos por la aplicación destino, debido a esto se considera necesario volver a programar los módulos de comunicación. Sin embargo, teniendo en cuenta el porcentaje de datos recibido se determina que el módulo de comunicación es adecuado para el envío de comandos entre aplicaciones.

5.1.3 Resultados de las pruebas

En esta sección se muestran los resultados obtenidos de las pruebas realizadas a cada uno de los usuarios, utilizando los sensores seleccionados (Tabla 6 a la 35).

Tabla 6. Pruebas del Usuario 1 con sensor Intel® RealSense™ SR300

Usuario: **1** Género: **Masculino** Color de piel: **E** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	10/10
Atrás	17-62 cm	10/10
Cerrar	17-62 cm	10/10
Play/Pause	17-62 cm	6/10
	Total	36/40
Porcentaje de aciertos		**90%**

Tabla 7. Pruebas del Usuario 1 con sensor Microsoft® Kinect V2.

Usuario: **1** Género: **Masculino** Color de piel: **E** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	8/10
Atrás	64-111 cm	6/10
Cerrar	64-111 cm	9/10
Play/Pause	64-111 cm	2/10
	Total	25/40
Porcentaje de aciertos		**62.5%**

Tabla 8. Pruebas del Usuario 2 con sensor Intel® RealSense™ SR300.

Usuario: **2** Género: **Masculino** Color de piel: **F** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	10/10
Atrás	17-62 cm	10/10
Cerrar	17-62 cm	8/10
Play/Pause	17-62 cm	9/10
	Total	37/40
Porcentaje de aciertos		**92.5%**

Tabla 9. Pruebas del Usuario 2 con sensor Microsoft® Kinect V2.

Usuario: **2** Género: **Masculino** Color de piel: **F** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	1/10
Atrás	64-111 cm	0/10
Cerrar	64-111 cm	1/10
Play/Pause	64-111 cm	6/10
	Total	8/40
Porcentaje de aciertos		**20%**

Tabla 10. Pruebas del Usuario 3 con sensor Intel® RealSense™ SR300.

Usuario: **3** Género: **Femenino** Color de piel: **A** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	10/10
Atrás	17-62 cm	10/10
Cerrar	17-62 cm	10/10
Play/Pause	17-62 cm	9/10
	Total	39/40
Porcentaje de aciertos		**97.5%**

Tabla 11. Pruebas del Usuario 3 con sensor Microsoft® Kinect V2.

Usuario: **3** Género: **Femenino** Color de piel: **A** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	10/10
Atrás	64-111 cm	10/10
Cerrar	64-111 cm	2/10
Play/Pause	64-111 cm	9/10
	Total	31/40
Porcentaje de aciertos		**77.5%**

Tabla 12. Pruebas del Usuario 4 con sensor Intel® RealSense™ SR300.

Usuario: **4** Género: **Masculino** Color de piel: **D** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	10/10
Atrás	17-62 cm	7/10
Cerrar	17-62 cm	10/10
Play/Pause	17-62 cm	6/10
	Total	33/40
Porcentaje de aciertos		**82.5%**

Tabla 13. Pruebas del Usuario 4 con sensor Microsoft® Kinect V2.

Usuario: **4** Género: **Masculino** Color de piel: **D** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	2/10
Atrás	64-111 cm	7/10
Cerrar	64-111 cm	0/10
Play/Pause	64-111 cm	7/10
	Total	16/40
Porcentaje de aciertos		**40%**

Tabla 14. Pruebas del Usuario 5 con sensor Intel® RealSense™ SR300.

Usuario: **5** Género: **Masculino** Color de piel: **E** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	8/10
Atrás	17-62 cm	6/10
Cerrar	17-62 cm	10/10
Play/Pause	17-62 cm	8/10
	Total	32/40
Porcentaje de aciertos		**80%**

Tabla 15. Pruebas del Usuario 5 con sensor Microsoft® Kinect V2.

Usuario: **5** Género: **Masculino** Color de piel: **E** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	10/10
Atrás	64-111 cm	8/10
Cerrar	64-111 cm	1/10
Play/Pause	64-111 cm	5/10
	Total	24/40
Porcentaje de aciertos		**60%**

Tabla 16. Pruebas del Usuario 6 con sensor Intel® RealSense™ SR300.

Usuario: **6** Género: **Femenino** Color de piel: **G** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	10/10
Atrás	17-62 cm	6/10
Cerrar	17-62 cm	9/10
Play/Pause	17-62 cm	10/10
	Total	35/40
Porcentaje de aciertos		**87.5%**

Tabla 17. Pruebas del Usuario 6 con sensor Microsoft® Kinect V2.

Usuario: **6** Género: **Femenino** Color de piel: **G** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	5/10
Atrás	64-111 cm	7/10
Cerrar	64-111 cm	6/10
Play/Pause	64-111 cm	7/10
	Total	25/40
Porcentaje de aciertos		**62.5%**

Tabla 18. Pruebas del Usuario 7 con sensor Intel® RealSense™ SR300.

Usuario: **7** Género: **Masculino** Color de piel: **C** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	10/10
Atrás	17-62 cm	8/10
Cerrar	17-62 cm	10/10
Play/Pause	17-62 cm	10/10
	Total	38/40
Porcentaje de aciertos		**95%**

Tabla 19. Pruebas del Usuario 7 con sensor Microsoft® Kinect V2.

Usuario: **7** Género: **Masculino** Color de piel: **C** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	10/10
Atrás	64-111 cm	1/10
Cerrar	64-111 cm	10/10
Play/Pause	64-111 cm	8/10
	Total	29/40
Porcentaje de aciertos		**72.5%**

Tabla 20. Pruebas del Usuario 8 con sensor Intel® RealSense™ SR300.

Usuario: **8** Género: **Masculino** Color de piel: **C** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	8/10
Atrás	17-62 cm	8/10
Cerrar	17-62 cm	10/10
Play/Pause	17-62 cm	10/10
	Total	36/40
Porcentaje de aciertos		**90%**

Tabla 21. Pruebas del Usuario 8 con sensor Microsoft® Kinect V2.

Usuario: **8** Género: **Masculino** Color de piel: **C** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	2/10
Atrás	64-111 cm	0/10
Cerrar	64-111 cm	7/10
Play/Pause	64-111 cm	8/10
	Total	17/40
Porcentaje de aciertos		**42.5%**

Tabla 22. Pruebas del Usuario 9 con sensor Intel® RealSense™ SR300.

Usuario: **9** Género: **Femenino** Color de piel: **D** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	9/10
Atrás	17-62 cm	10/10
Cerrar	17-62 cm	9/10
Play/Pause	17-62 cm	10/10
	Total	38/40
Porcentaje de aciertos		**95%**

Tabla 23. Pruebas del Usuario 9 con sensor Microsoft® Kinect V2.

Usuario: **9** Género: **Femenino** Color de piel: **D** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	10/10
Atrás	64-111 cm	4/10
Cerrar	64-111 cm	6/10
Play/Pause	64-111 cm	12/10
	Total	32/40
Porcentaje de aciertos		**80%**

Tabla 24. Pruebas del Usuario 10 con sensor Intel® RealSense™ SR300.

Usuario: **10** Género: **Masculino** Color de piel: **B** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	10/10
Atrás	17-62 cm	5/10
Cerrar	17-62 cm	10/10
Play/Pause	17-62 cm	5/10
	Total	30/40
Porcentaje de aciertos		**75%**

Tabla 25. Pruebas del Usuario 10 con sensor Microsoft® Kinect V2.

Usuario: **10** Género: **Masculino** Color de piel: **B** Luminosidad: **Alta**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	7/10
Atrás	64-111 cm	2/10
Cerrar	64-111 cm	0/10
Play/Pause	64-111 cm	2/10
	Total	11/40
Porcentaje de aciertos		**27.5%**

Tabla 26. Pruebas del Usuario 11 con sensor Intel® RealSense™ SR300.

Usuario: **11** Género: **Femenino** Color de piel: **B** Luminosidad: **Baja**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	9/10
Atrás	17-62 cm	10/10
Cerrar	17-62 cm	8/10
Play/Pause	17-62 cm	10/10
	Total	37/40
Porcentaje de aciertos		**92.5%**

Tabla 27. Pruebas del Usuario 11 con sensor Microsoft® Kinect V2.

Usuario: **11** Género: **Femenino** Color de piel: **B** Luminosidad: **Baja**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	8/10
Atrás	64-111 cm	8/10
Cerrar	64-111 cm	7/10
Play/Pause	64-111 cm	7/10
	Total	30/40
Porcentaje de aciertos		**75%**

Tabla 28. Pruebas del Usuario 12 con sensor Intel® RealSense™ SR300.

Usuario: **12** Género: **Masculino** Color de piel: **G** Luminosidad: **Baja**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	10/10
Atrás	17-62 cm	5/10
Cerrar	17-62 cm	9/10
Play/Pause	17-62 cm	9/10
	Total	33/40
Porcentaje de aciertos		**82.5%**

Tabla 29. Pruebas del Usuario 12 con sensor Microsoft® Kinect V2.

Usuario: **12** Género: **Masculino** Color de piel: **G** Luminosidad: **Baja**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	6/10
Atrás	64-111 cm	0/10
Cerrar	64-111 cm	9/10
Play/Pause	64-111 cm	7/10
	Total	22/40
Porcentaje de aciertos		**55%**

Tabla 30. Pruebas del Usuario 13 con sensor Intel® RealSense™ SR300.

Usuario: **13** Género: **Masculino** Color de piel: **C** Luminosidad: **Baja**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	10/10
Atrás	17-62 cm	7/10
Cerrar	17-62 cm	10/10
Play/Pause	17-62 cm	7/10
	Total	34/40
Porcentaje de aciertos		**85%**

Tabla 31. Pruebas del Usuario 13 con sensor Microsoft® Kinect V2.

Usuario: **13** Género: **Masculino** Color de piel: **C** Luminosidad: **Baja**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	3/10
Atrás	64-111 cm	5/10
Cerrar	64-111 cm	9/10
Play/Pause	64-111 cm	8/10
	Total	25/40
Porcentaje de aciertos		**62.5%**

Tabla 32. Pruebas del Usuario 14 con sensor Intel® RealSense™ SR300.

Usuario: **14** Género: **Femenino** Color de piel: **A** Luminosidad: **Baja**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	8/10
Atrás	17-62 cm	10/10
Cerrar	17-62 cm	8/10
Play/Pause	17-62 cm	10/10
	Total	36/40
Porcentaje de aciertos		**90%**

Tabla 33. Pruebas del Usuario 14 con sensor Microsoft® Kinect V2.

Usuario: **14** Género: **Femenino** Color de piel: **A** Luminosidad: **Baja**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	2/10
Atrás	64-111 cm	7/10
Cerrar	64-111 cm	4/10
Play/Pause	64-111 cm	5/10
	Total	18/40
Porcentaje de aciertos		**45%**

Tabla 34. Pruebas del Usuario 15 con sensor Intel® RealSense™ SR300.

Usuario: **15** Género: **Masculino** Color de piel: **G** Luminosidad: **Baja**		
Movimiento	Distancia	Aciertos
Adelante	17-62 cm	9/10
Atrás	17-62 cm	8/10
Cerrar	17-62 cm	9/10
Play/Pause	17-62 cm	9/10
	Total	35/40
Porcentaje de aciertos		**87.5%**

Tabla 35. Pruebas del Usuario 15 con sensor Microsoft® Kinect V2.

Usuario: **15** Género: **Masculino** Color de piel: **G** Luminosidad: **Baja**		
Movimiento	Distancia	Aciertos
Adelante	64-111 cm	3/10
Atrás	64-111 cm	5/10
Cerrar	64-111 cm	5/10
Play/Pause	64-111 cm	4/10
	Total	17/40
Porcentaje de aciertos		**42.5%**

5.1.4 Reporte del análisis de resultados

En esta sección se muestra un análisis de los resultados obtenidos con los diferentes usuarios.

Las tablas 36 y 37 muestran los resultados de efectividad tomando en cuenta el tono de piel de cada usuario.

En la tabla 36 es posible observar que no existe una decadencia de la efectividad en la detección si el tono de piel del usuario es más obscuro, es posible observar que la efectividad del tono F es muy similar a la efectividad del tono A.

Tabla 36. Efectividad por tono de piel con sensor Intel® RealSense™ SR300.

Tonalidad	Porcentaje de aciertos
Usuarios con tono de piel A	93.75%
Usuarios con tono de piel B	83.5%
Usuarios con tono de piel C	88.33%
Usuarios con tono de piel D	88.75%
Usuarios con tono de piel E	85%
Usuarios con tono de piel F	92.5%
Usuarios con tono de piel G	85.83%

En la tabla 37 es posible observar que la efectividad de la detección es variante, sin embargo, de igual manera se puede observar que la capacidad de detección no está ligada al tono de piel del usuario.

Tabla 37. Efectividad por tono de piel con sensor Microsoft® Kinect V2.

Tonalidad	Porcentaje de aciertos
Usuarios con tono de piel A	61.25%
Usuarios con tono de piel B	51.25%
Usuarios con tono de piel C	59.16%
Usuarios con tono de piel D	50%
Usuarios con tono de piel E	61.25%
Usuarios con tono de piel F	20%
Usuarios con tono de piel G	53.16%

En las tablas 38 y 39 se muestran los resultados de efectividad tomando en cuenta el género de los usuarios.

Tabla 38. Efectividad por género con sensor Intel® RealSense™ SR300.

Género	Porcentaje de aciertos
Masculino	86%
Femenino	92.5

Tabla 39. Efectividad por género con sensor Microsoft® Kinect V2.

Género	Porcentaje de aciertos
Masculino	48.5%
Femenino	68%

En las tablas anteriores se observa que la efectividad del sistema con ambos sensores tiende a mejorar cuando el usuario es de género femenino, sin embargo, debido a que el número de usuarios del género masculino es el doble que, del género femenino, no es posible determinar que el género influye

directamente en la capacidad de detección; por lo tanto, se considera necesario realizar otra serie de pruebas en donde se cuente con un número de usuarios masculinos igual que de usuarios femeninos.

Las tablas 40 y 41 muestran los resultados de efectividad tomando en cuenta la luminosidad del ambiente. Es posible observar que la luminosidad no influye en los resultados ya que los resultados aumentan o disminuyen dependiendo del sensor utilizado.

Tabla 40. Efectividad por luminosidad con sensor Intel® RealSense™ SR300.

Luminosidad	Porcentaje de aciertos
Alta	88.45%
Baja	87.5%

Tabla 41 Efectividad por luminosidad con sensor Microsoft® Kinect V2.

Luminosidad	Porcentaje de aciertos
Alta	54.5%
Baja	56%

En las tablas 42 y 43 se muestran los resultados de efectividad tomando en cuenta los movimientos realizados.

Tabla 42. Efectividad por movimientos con sensor Intel® RealSense™ SR300.

Usuario	Adelante	Atrás	Cerrar	Play/Pause
1	10/10	10/10	10/10	6/10
2	10/10	10/10	8/10	9/10
3	10/10	10/10	10/10	9/10
4	10/10	7/10	10/10	6/10
5	8/10	6/10	10/10	8/10
6	10/10	6/10	9/10	10/10
7	10/10	8/10	10/10	10/10
8	8/10	8/10	10/10	10/10
9	9/10	10/10	9/10	10/10
10	10/10	5/10	10/10	5/10
11	9/10	10/10	8/10	10/10
12	10/10	5/10	9/10	9/10
13	10/10	7/10	10/10	7/10
14	8/10	10/10	8/10	10/10
15	9/10	8/10	9/10	9/10
Total	141/150	120/150	140/150	128/150
Porcentaje de aciertos	**94%**	**80 %**	**93.33%**	**85.33%**

Como se puede observar en la tabla 42 el porcentaje de aciertos del sistema va desde el 80% hasta el 94%; sin embargo, también se observa que dos de los movimientos presentan resultados arriba del 90% y dos movimientos presentan resultados menores al 90%.

Tabla 43 Efectividad por movimientos con sensor Microsoft® Kinect V2.

Usuario	Adelante	Atrás	Cerrar	Play/Pause
1	8/10	6/10	9/10	2/10
2	1/10	0/10	1/10	6/10
3	10/10	10/10	2/10	9/10
4	2/10	7/10	0/10	7/10
5	10/10	8/10	1/10	5/10
6	5/10	7/10	6/10	7/10
7	10/10	1/10	10/10	8/10
8	2/10	0/10	7/10	8/10
9	10/10	4/10	6/10	8/10
10	7/10	2/10	0/10	2/10
11	8/10	8/10	7/10	7/10
12	6/10	0/10	9/10	7/10
13	3/10	5/10	9/10	8/10
14	2/10	7/10	4/10	5/10
15	3/10	5/10	5/10	4/10
Total	87/150	70/150	76/150	93/150
Porcentaje de aciertos	**58%**	**46.67%**	**50.67%**	**62%**

En la tabla 43 se observa que los resultados obtenidos en las pruebas con el sensor Kinect V2 (Microsoft, Developer, 2018) presentan una variación muy grande en la cantidad de aciertos entre los usuarios. Además, bajo desempeño ya que la capacidad de detección es considerablemente menor a la capacidad del sensor RealSense™ (Intel, Software Intel, 2018) puesto que no supera el 65% de detección por movimiento.

Se observó que algunos usuarios tenían un grado de dificultad a la hora de realizar alguno de los movimientos registrados en el sistema, por lo que se elaboró una encuesta de usabilidad.

Primeramente, se preguntó a los usuarios si estaban de acuerdo con las opciones que fueron agregadas al sistema, es decir, las acciones que se realizaban con los movimientos asignados. El 100% de los usuarios estuvieron de acuerdo (Figura 47).

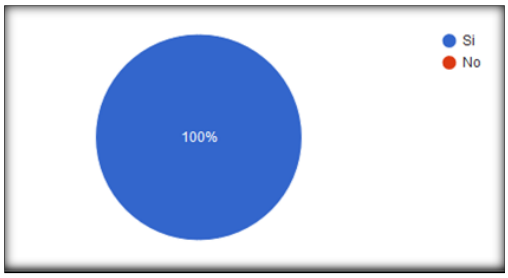

Figura 47. Movimientos asignados.

La segunda pregunta es opcional, en ella se les pide a los usuarios alguna sugerencia de acciones que podría realizar el sistema, la figura 48 muestra las sugerencias de los usuarios.

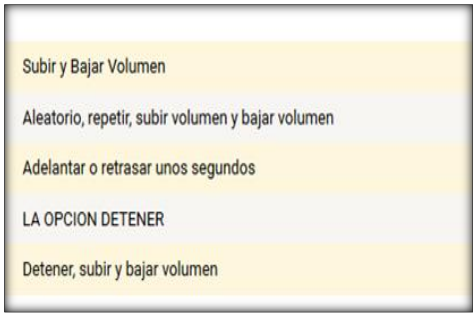

Figura 48. Opciones para agregar.

Teniendo en mente la problemática observada, se les preguntó a los usuarios que grado de facilidad tuvieron para realizar los movimientos, la figura 49 muestra que para ningún usuario fue muy fácil realizarlo, sin embargo, a ningún usuario le fue muy difícil realizar los movimientos.

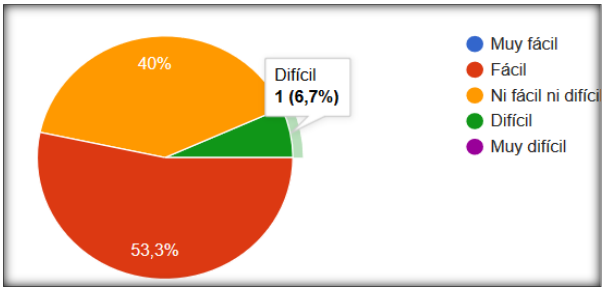

Figura 49. Dificultad de los movimientos.

Se les preguntó cuáles fueron los movimientos más fáciles y más difíciles de realizar y se obtuvieron las siguientes respuestas (Figura 50 y 51).

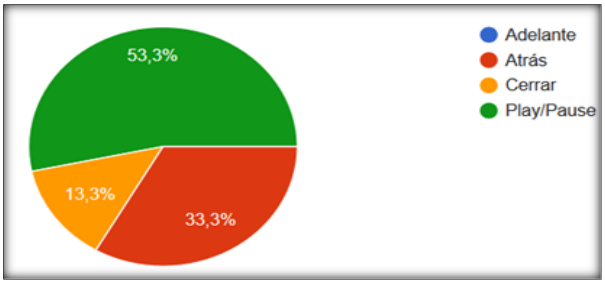

Figura 50. Movimiento más difícil.

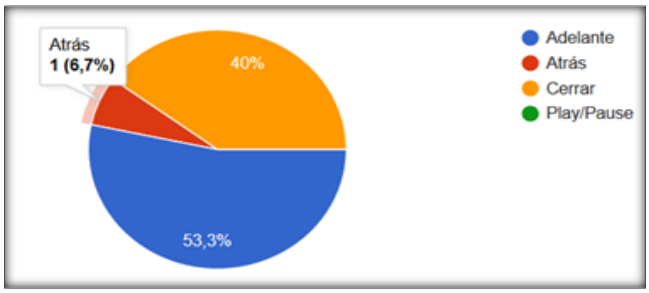

Figura 51. Movimiento más fácil.

Dadas las respuestas obtenidas de la encuesta identificó que se pueden establecer nuevos movimientos, estos permitirán a los usuarios no esforzar las manos y a su vez tendrá como consecuencia un mejor reconocimiento de los movimientos en el sistema.

De acuerdo con los resultados obtenidos de las pruebas realizadas al sistema de detección se realizó una tabla (Tabla 44 comparativa, una gráfica (Figura 52) comparativa y una comparación de los movimientos objetivo (Figura 43) y los movimientos obtenidos para cada clase en cada uno de los sensores utilizados (Figura 53 y 54).

Es posible observar que la diferencia entre la detección va desde el 72.5% hasta el 17.5% sin embargo, la diferencia está muy descontrolada debido a la capacidad de detección del sensor Kinect V2 (Microsoft, Developer, 2018) cuando se trata de detectar las manos.

Tabla 44. Tabla comparativa de resultados.

Usuario	Kinect V2	RealSense SR300	Diferencia
1	62.5%	90%	27.5%
2	20%	92.5%	72.5%
3	77.5%	97.5%	20%
4	40%	82.5%	42.5%
5	60%	80%	20%
6	62.5%	87.5%	25%
7	72.5%	95%	22.5%
8	42.5%	90%	47.5%
9	70%	95%	25%
10	27.5%	75%	47.5%
11	75%	92.5%	17.5%
12	55%	82.5%	27.5%
13	62.5%	85%	22.5%
14	45%	90%	45%
15	42.5%	87.5%	45%
Total	**54.33%**	**88.16%**	**33.83%**

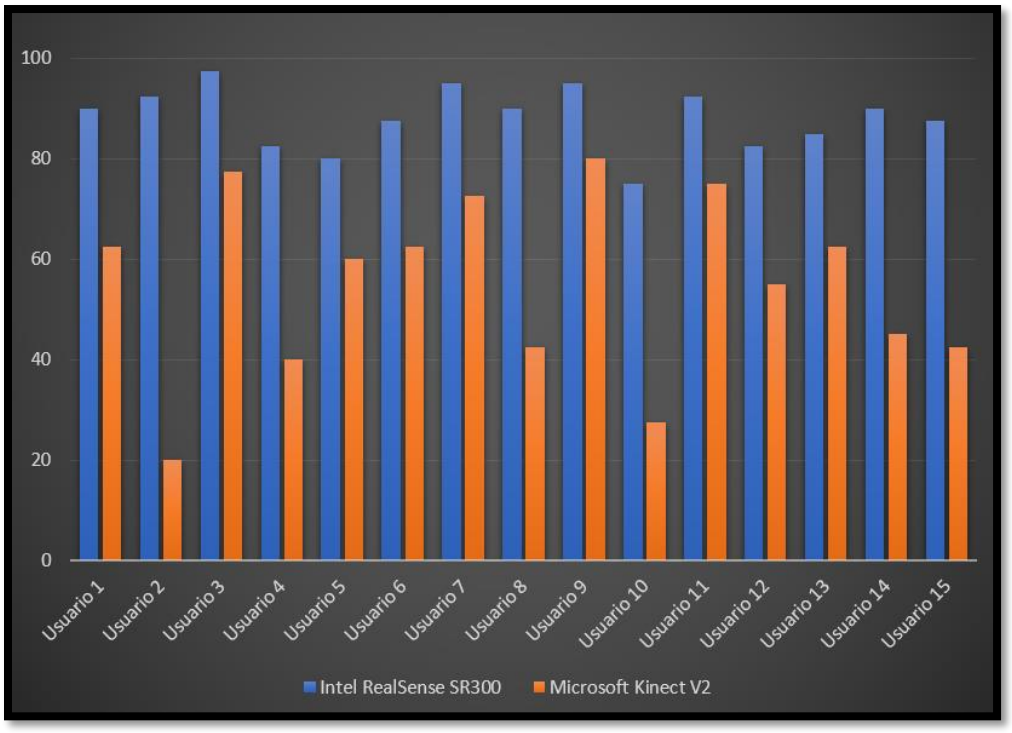

Figura 52. Comparativa de sensores.

Figura 53. Movimientos obtenidos RealSense™ SR300 (Intel, 2018).

Figura 54. Movimientos obtenidos Kinect V2 (Microsoft, Developer, 2018).

Con todas las pruebas realizadas y los resultados obtenidos se determinó que, las principales consideraciones a tomar en cuenta para desarrollar un sistema de detección de movimiento de manos son:

- Definir movimientos que sean fáciles de realizar para los usuarios.
- Seleccionar el mejor sensor para llevar a cabo la detección (RealSense™ SR300).

Se realizaron 600 movimientos por sensor utilizando el mismo sistema para ambos y se obtuvo una eficiencia del 88.16% para Intel® RealSense™ SR300 (Intel, 2018) y 54.33% para Microsoft® Kinect V2 (Microsoft, Developer, 2018), es por esto que, el mejor sensor para llevar a cabo el reconocimiento de movimientos es el sensor **Intel® RealSense™ SR300** (Intel, Software Intel, 2018) debido a que, con este sensor, el sistema tiene un mayor porcentaje de efectividad.

Capítulo VI

Conclusiones y trabajos futuros

Capítulo 6. Conclusiones y trabajos futuros.

En este capítulo se exponen las conclusiones y aportaciones que han resultado de este trabajo de investigación. Además, se exponen ideas y sugerencias con la finalidad de mejorar el sistema.

6.1 Conclusiones.

En la investigación realizada durante el trabajo de tesis, se observó que existen diversos métodos y dispositivos que permiten identificar movimientos de manos, sin embargo, la mayoría de estos requieren de la pericia en temas de visión artificial; no obstante, se identificó un método que permite desarrollar sistemas de detección sin poseer un gran conocimiento en el área.

Cumplimiento de los objetivos

En la tabla 45 se muestran los objetivos específicos y como se cumplieron.

Tabla 45 Cumplimiento de los objetivos específicos.

Objetivo específico	Completado con
Definir una lista de movimientos que puedan ser usados para manipular una interfaz.	✓ Módulo de definición de posiciones
Realizar un módulo que identifique al menos cuatro posiciones de las manos	✓ Módulo de definición de movimientos
Definir un método de conexión inalámbrica que permita el envío de datos entre el sistema computacional y la interfaz remota en tiempo real.	✓ Módulo de envío de comandos ✓ Módulo de comunicación

El objetivo general de este trabajo **"Desarrollar un sistema computacional que identifique movimientos de manos definidos para manipular interfaces computacionales de manera remota"** se cumplió con la investigación y el desarrollo de una aplicación para Windows que se encarga de detectar los movimientos de las manos, enviar los comandos asociados a ellos y ejecutar los comandos en una computadora conectada remotamente.

En este trabajo se logró un acercamiento al **Lenguaje de Definición de Gestos** para el desarrollo de sistemas de detección de movimiento de manos, que permite definir gestos que a su vez pueden ser utilizados para manipular diferentes interfaces computacionales.

Las pruebas realizadas al sistema otorgan un 88.16% (RealSense SR300) y 54.33% (Kinect v2) de capacidad de detección con 4 gestos definidos (compuestos por 2 posiciones cada uno).

Con esta investigación se aporta un método de detección de movimiento que permite desarrollar sistemas no limitados a la cantidad de movimientos registrados, es posible definir posiciones y gestos a través del uso de un Lenguaje de Definición de Gestos, no necesita de la creación de bases de datos de muestras, no necesita el entrenamiento de clasificadores, es posible controlar interfaces de manera remota y facilita la adición de nuevas posiciones y gestos.

Este trabajo posibilitó observar distintos comportamientos tanto del sistema como de los usuarios, que a su vez permiten identificar nuevos temas de investigación para mejorar la efectividad de los sistemas a desarrollar.

Durante el desarrollo de la investigación se realizó la publicación de algunos artículos en diferentes áreas (Ver anexos).

6.2 Trabajos futuros

A continuación, se muestran los trabajos futuros relacionados con este trabajo de investigación.

- Diseñar, desarrollar y probar sistemas que definan gestos que se compongan con más de dos posiciones cada uno.
- Investigar el uso del SDK para el desarrollo de sistemas con detección de profundidad de las manos.
- Desarrollar un sistema que utilice la aún experimental detección de ambas manos que no se tocan del SDK del proyecto Praga.
- Investigar la compatibilidad del SDK con las nuevas generaciones de sensores de profundidad.
- Desarrollar un sistema de definición de posiciones que identifique la posición que se presenta frente al sensor y que genere el código (Lenguaje de Definición de Gestos) que corresponda a dicha posición.
- Desarrollar una interfaz remota que pueda replicar los movimientos de la mano derecha del usuario.
- Investigar la capacidad de movimiento que tienen las manos de las personas para determinar qué tan sencillo o difícil es realizar un movimiento en específico.

Capítulo VII

Bibliografía

Bibliografía

Aguirre, S. (2013). SAAC-Droid, una herramienta de ayuda a la comunicación.

Antón, D., Kurillo, G., Goñi, A., Illarramendi, A., & Bajcsy, R. (2017). Real-time communication for Kinect-based telerehabilitation. *Future Generation Computer Systems 75*.

Bendale, T., & Kharat, V. (2017). A design of Gesture controlled mobile robot with robotic arm for nuclear environment. *International Conference on Nascent Technologies in the Engineering Field (ICNTE-2017)*.

Biblioteca Digital ILCE. (18 de 05 de 2018). Obtenido de http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen2/ciencia3/084/htm/sec_9.htm

Brico Geek. (s.f.). Recuperado el 17 de Septiembre de 2017, de http://tienda.bricogeek.com/sensores/217-sensor-optico-qrd1114.html

Cambridge Dictionary. (12 de 10 de 2017). Obtenido de http://dictionary.cambridge.org/es/diccionario/ingles/gesture

Carrod electronica. (2014). Recuperado el 17 de Septiembre de 2017, de https://www.carrod.mx/products/sensor-reflectivo-de-un-canal-cny70

Cetronic componentes electrónicos. (s.f.). Recuperado el 17 de Septiembre de 2017, de http://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=84-TSOP4838&cPath=1353

Conéctica. (17 de Julio de 2013). Recuperado el 20 de Septiembre de 2017, de https://conectica.com/2013/07/17/glassup-la-alternativa-economica-para-las-google-glass/

Csapo, A., & Kristjánsson, Á. (2016). Evaluation of Human-Myo Gesture Control. *7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2016)*, 415-420.

Deng, S., Chang, J., & Jian, J. (2017). Gaze Modulated Disambiguation Technique for Gesture Control in 3D Virtual Objects Selection. *Cybernetics (CYBCONF), 2017 3rd IEEE International Conference on*.

Emrehan, Y. (2016). Gesture imitation and recognition using Kinect sensor and extreme learning machines. *Measumrement 94*, 852-861.

eneso. (2017). Recuperado el 21 de 09 de 2017, de http://www.eneso.es/verbo/

Eneso, T. d. (2002). *http://www.eneso.es/producto/verbo*. Obtenido de Tecnología para personas con diversidad funcional: http://www.eneso.es/

EPSON. (2017). Recuperado el 20 de Septiembre de 2017, de https://www.epson.es/products/see-through-mobile-viewer/moverio-bt-200

Eyal Krupka, K. K. (2017). Toward realistic hands gesture interface: Keeping it simple for developers and machines. *Microsoft Research*.

Fernández Fadrique, F. J., & Baquero Vega, A. (26 de 06 de 2018). *Convexidad*. Obtenido de http://asignatura.us.es/fgcitig/contenidos/practicas/envolvente/tema5.html

Fritzing. (s.f.). Recuperado el 20 de Septiembre de 2017, de http://fritzing.org/

Fundation, B. (21 de 09 de 2017). *Bluetooth*. Obtenido de https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works

Gergely , S., Ujbányi, T., & József , K. (2016). Cost-effective hand gesture computer control. *7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2016)*, 239-234.

GIGA. (2015). Recuperado el 21 de 09 de 2017, de http://giga.cps.unizar.es/affectivelab/araboard.html

GLobal, I. (18 de 05 de 2018). Obtenido de https://www.igi-global.com/dictionary/human-motion-analysis-and-simulation-tools/50427

Gunawardane, M. M. (2016). The Development of a Gesture Controlled Soft Robot Gripping Mechanism. *Faculty of Engineering Technology, The Open University of Sri Lanka*.

Han, X., & Rashid, M. (2016). Gesture and Voice Control of Internet of Things. *11th Conference on Industrial Electronics and Applications (ICIEA)*, 1991-1995.

Héctor Daniel, P., Sofía, A., & José Luis, F. (2013). SAAC-Droid, una herramienta de ayuda a la comunicación. *MULTICIENCIAS Vol. 13, Nº 3,* , 313-320.

Hershberger, J. (1992). Speeding Up the Douglas-Peucker Line-Simplication Algorithm. *The Universty of British Columbia*.

Hornyak, T. (10 de Diciembre de 2013). *Cnet*. Recuperado el 20 de Septiembre de 2017, de https://www.cnet.com/news/wink-glasses-fog-up-in-a-flash-forcing-you-to-blink/

INEGI. (2014). *ENADID*. Recuperado el 20 de 08 de 2017, de http://www.beta.inegi.org.mx/proyectos/enchogares/especiales/enadid/2014/

INEGI. (s.f.). *INEGI.* Recuperado el 20 de 08 de 2017, de http://internet.contenidos.inegi.org.mx/contenidos/productos/prod_serv/contenidos/espanol/bvinegi/productos/nueva_estruc/702825090203.pdf

Intel. (20 de 05 de 2018). *Intel*. Obtenido de https://software.intel.com/en-us/realsense/sr300

Intel. (20 de 02 de 2018). *Software Intel*. Obtenido de https://software.intel.com/es-es/realsense/sr300

IPN, N. I. (2015). Interfaz cerebro-computadora para comunicar a discapacitados. *Gaceta Politécnica - IPN 1141*, 5.

irisbond. (2016). Recuperado el 21 de 09 de 2017, de http://www.irisbond.com/productos/irisbond-duo

IRISBOND, C. S. (2016). Obtenido de IRISBOND: http://www.irisbond.com

José Alberto, M., Héctor, G., & Aída, C. (2013). Sistema de comunicación alternativa para personas con parálisis cerebral que saben leer y escribir con el apoyo de un dispositivo móvil con Android. *Revista Tecnología Digital Vol. 3 No. 1, 2013*, 9-18.

KANEDA. (18 de 05 de 2018). Obtenido de http://kaneda123.over-blog.es/article-que-conexion-remota-85904632.html

Lee, U., & Tanaka, J. (2013). Finger Identification and Hand Gesture Recognition. *Asia-Pacific Computer and Human Interaction*.

López, E. O. (9 de 08 de 2016). Reconocimiento en tiempo real de la lengua dactilológica colombiana basado en aprendizaje supervisado usando la cámara 3D RealSense. Colombia: Universidad de los Andes.

Mancilla, J. A. (2013). Sistema de comunicación alternativa para personas con parálisis cerebral que saben leer y escribir con el apoyo de un dispositivo móvil con Android.

María, T., Juan Jesús, R., & Antonio, B. (2016). Interfaz cerebro-computadora para comunicar a discapacitados.

Martínez, A. G. (2014). Sistema alternativo y aumentativo de comunicación para personas con problemas fonéticos con el apoyo de un dispositivo móvil con Android.

Merino, M. (18 de 05 de 2018). *TICBeat*. Obtenido de http://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve/

Microsoft. (25 de Noviembre de 2017). *Microsoft Docs*. Obtenido de https://docs.microsoft.com/en-us/gestures/

Microsoft. (20 de 05 de 2018). Obtenido de https://www.microsoft.com/en-us/store/d/refurbished-kinect-sensor-for-xbox-one/8t3g646dzrpb/39pk?activetab=pivot%3aoverviewtab

Microsoft. (20 de 02 de 2018). *Developer*. Obtenido de https://developer.microsoft.com/en-us/windows/kinect

Microsoft. (20 de 02 de 2018). *Docs*. Obtenido de https://docs.microsoft.com/en-us/gestures/

Microsoft. (20 de 02 de 2018). *Windows*. Obtenido de https://www.microsoft.com/es-mx/software-download/windows10

Mills, P. (2010). Efficient statistical classification of satellite measurements. *International Journal of Remote Sensing* .

Moreno, G. V. (2010). *Sistema de Reconocimiento Gestual para Unidad de Medición Inercial*. Obtenido de https://riunet.upv.es/bitstream/handle/10251/8609/MEMORIA_PFC_GUSTAVO_VIDAL.pdf

Morse, C. (15 de 08 de 2017). *Codigo Morse*. Obtenido de http://www.codigomorse.com/

MSN. (18 de 05 de 2018). *MSN*. Obtenido de https://www.msn.com/es-es/noticias/microsoftstore/%C2%BFqu%C3%A9-es-y-para-qu%C3%A9-sirve-visual-studio-2017/ar-AAnLUC9

Nájera, L. O. (2017). Generación de Palabras a Partir de la Lengua de Señas Mexicana.

Nano ID. (2017). Recuperado el 17 de Septiembre de 2017, de https://nanoid.mx/products/sensor-optico-reflectivo-infrarrojo-tcrt5000l-sin-pcb

Onysus. (18 de 05 de 2018). Obtenido de https://www.onysus.com/interfaz-natural-de-usuario-nui-en-una-cascara-de-nuez/

Pinola, M. (12 de 06 de 2017). *Lifewire*. Obtenido de https://www.lifewire.com/what-is-wi-fi-2377430

Proyectos Electronics. (s.f.). Recuperado el 17 de Septiembre de 2017, de https://catorcepi.wikispaces.com/file/view/3678453-SENSOR-INFRARROJO-Teoria-y-practica.pdf

RAE. (20 de 08 de 2017). *Real Academia Española*. Obtenido de http://corpus.rae.es/lfrecuencias.html

Reyes, F. P. (11 de 2017). Sistema de recomendación basado en el perfil de usuario en personas con discapacidad de lenguaje mediante un dispositivo que controla el cursor en pantalla. CENIDET.

samsung. (2014). Recuperado el 21 de 09 de 2017, de https://news.samsung.com/global/samsung-electronics-introduces-eyecan-next-generation-mouse-for-people-with-disabilities

Segmentación. Transformada de Hough. . (2006). Universidad de Jaén.

Software Factory. (18 de 05 de 2018). Obtenido de http://www.4rsoluciones.com/blog/que-es-un-kit-de-desarrollo-de-software-sdk-2/

Spotify. (20 de 02 de 2018). *Spotify*. Obtenido de https://www.spotify.com/mx/

Sueaseenak, D., & Khawdee, C. (2017). A Performance of Modern Gesture Control Device with Application in Pattern Classification. *International Conference on Control, Automation and Robotics*, 428-431.

Sueaseenak, D., Khawdee, C., Pakornsirikul, N., & Sukjamsri, C. (2017). A Performance of Modern Gesture Control Device with Application in Pattern Classification. *International Conference on Control, Automation and Robotics*, 428-431.

Tchoketch Kenbir, S., & Bouhedda. (2016). Gesture Control of Mobile Robot Based Arduino. *8th International Conference on Modelling, Identification and Control (ICMIC-2016)*, 1081-1085.

UABCS. (2016). *Universidad Autónoma de Baja California Sur*. Obtenido de http://conacytprensa.mx/index.php/tecnologia/tic/11603-neurosoft-una-aplicacion-para-conectarse-con-el-mundo

UcMerced. (25 de 08 de 2017). *UcMerced Library*. Obtenido de http://library.ucmerced.edu/node/10249

USERS. (2011). Módulo de captura de esquemáticos. En *Electrónica práctica* (pág. 85). Buenos Aires: redUSERS.

Vuzix. (2017). Recuperado el 20 de Septiembre de 2017, de https://www.vuzix.com/Products/M100-Smart-Glasses

WHO. (s.f.). *World Health Organization*. Recuperado el 25 de 08 de 2017, de http://www.who.int/es/

Wiki de robótica. (s.f.). Recuperado el 17 de Septiembre de 2017, de http://wiki.robotica.webs.upv.es/wiki-de-robotica/sensores/sensores-proximidad/sensor-infrarrojos/

Wikipedia. (3 de Septiembre de 2017). Recuperado el 20 de Septiembre de 2017, de https://es.wikipedia.org/wiki/Google_Glass

world of πer. (26 de 06 de 2018). Obtenido de http://pier.guillen.com.mx/algorithms/07-geometricos/07.7-envoltura_convexa.htm

Anexos

Anexos

En este apartado se muestra la información anexa al documento de tesis.

Código del sistema de detección de posiciones.

```
using System;
using System.Diagnostics;
using System.Threading;
using System.Windows;
using Microsoft.Gestures;
using Microsoft.Gestures.Endpoint;
using System.Windows.Media.Imaging;

namespace Cliente.Gestos.Static
{
    /// <summary>
    /// Lógica de interacción para MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            ProcessStartInfo Servicio = new ProcessStartInfo();
            Servicio.UseShellExecute = true;
            Servicio.FileName = "Microsoft.Gestures.Service.exe";
            Servicio.WorkingDirectory = @"C:\Users\Luis
Salgado\AppData\Roaming\Microsoft\Prague\PragueVersions\LatestVersion";
            Process.Start(Servicio);
            Thread.Sleep(3000);
            Process.Start(Servicio);
            Thread.Sleep(1000);

            InitializeComponent();
            Static();

        }

        private async void Static()
        {
            var URI = @"D:\Dropbox\Cenidet\Semestre 4\Visual projects\Cliente.Gestos.Static\Gestos
Img\";

            var letra_a = new HandPose("LetraA",
               new FingerPose(new[] { Finger.Index, Finger.Middle, Finger.Ring, Finger.Pinky },
FingerFlexion.FoldedTucked),
               new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching,
Finger.Thumb),
               new FingerPose(Finger.Thumb, FingerFlexion.Open, PoseDirection.Left));
            var letra_b = new HandPose("LetraB",
                new FingerPose(Finger.Thumb, FingerFlexion.FoldedTucked),
                new FingerPose(new[] { Finger.Index, Finger.Middle, Finger.Ring, Finger.Pinky },
FingerFlexion.Open),
                new FingertipDistanceRelation (Finger.Index, RelativeDistance.Touching,
Finger.Middle),
                new FingertipDistanceRelation (Finger.Middle, RelativeDistance.Touching,
Finger.Ring),
                new FingertipDistanceRelation(Finger.Ring, RelativeDistance.Touching, Finger.Pinky),
                new FingertipPlacementRelation(Finger.Middle, RelativePlacement.Above,
Finger.Thumb));
            var letra_c = new HandPose("letraC",
                new FingerPose(new[] { Finger.Thumb, Finger.Index, Finger.Middle, Finger.Ring,
Finger.Pinky }, FingerFlexion.Open, PoseDirection.Left),
                new FingertipPlacementRelation(Finger.Index, RelativePlacement.Above, Finger.Thumb),
                new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching,
Finger.Thumb));
            var letra_d = new HandPose("letraD",
```

```
                new FingerPose(new[] { Finger.Thumb, Finger.Middle, Finger.Ring, Finger.Pinky },
FingerFlexion.OpenStretched, PoseDirection.Left),
                new FingerPose(Finger.Index, FingerFlexion.Open),
                new FingertipPlacementRelation(Finger.Middle, RelativePlacement.Above,
Finger.Thumb),
                new FingertipDistanceRelation(Finger.Middle, RelativeDistance.Touching,
Finger.Thumb),
                new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching,
Finger.Middle));
            var letra_e = new HandPose("letraE",
                new FingerPose(new AllFingersContext(), FingerFlexion.FoldedTucked) );
            var letra_f = new HandPose("LetraF",
                new FingerPose(new AllFingersContext(), FingerFlexion.Open),
                new FingertipDistanceRelation(Finger.Index, RelativeDistance.Touching,
Finger.Thumb),
                new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching,
Finger.Middle),
                new FingertipDistanceRelation(Finger.Middle, RelativeDistance.Touching,
Finger.Ring),
                new FingertipDistanceRelation(Finger.Ring, RelativeDistance.Touching,
Finger.Pinky));
            var letra_g = new HandPose("LetraG",
                new FingerPose(Finger.Index, FingerFlexion.Open, PoseDirection.Left),
                new FingerPose(Finger.Thumb, FingerFlexion.Open, PoseDirection.Up),
                new FingerPose(new[] {Finger.Middle, Finger.Ring, Finger.Pinky},
FingerFlexion.Folded));
            var letra_h = new HandPose("LetraH",
                new FingerPose(new[] { Finger.Index,Finger.Middle}, FingerFlexion.Open,
PoseDirection.Left),
                new FingerPose(Finger.Thumb, FingerFlexion.Open, PoseDirection.Up),
                new FingerPose(new[] {Finger.Ring, Finger.Pinky }, FingerFlexion.Folded));


            var Letra_A = new Gesture("Letra_A", letra_a);
            var Letra_B = new Gesture("Letra_B", letra_b);
            var Letra_C = new Gesture("Letra_C", letra_c);
            var Letra_D = new Gesture("Letra_D", letra_d);
            var Letra_E = new Gesture("Letra_E", letra_e);
            var Letra_F = new Gesture("Letra_F", letra_f);
            var Letra_G = new Gesture("Letra_G", letra_g);
            var Letra_H = new Gesture("Letra_H", letra_h);


            var gesturesService = GesturesServiceEndpointFactory.Create();
            await gesturesService.ConnectAsync();

            await gesturesService.RegisterGesture(Letra_A);
            await gesturesService.RegisterGesture(Letra_B);
            await gesturesService.RegisterGesture(Letra_C);
           // await gesturesService.RegisterGesture(Letra_D);
            await gesturesService.RegisterGesture(Letra_E);
            await gesturesService.RegisterGesture(Letra_F);
            await gesturesService.RegisterGesture(Letra_G);
            await gesturesService.RegisterGesture(Letra_H);

            Letra_A.Triggered += (sender, args) =>
            {

                Dispatcher.Invoke(() => Gestos.Source = new BitmapImage(new Uri(URI+
"Letra_A.jpg")));

            };

            Letra_B.Triggered += (sender, args) =>
            {

                Dispatcher.Invoke(() => Gestos.Source = new BitmapImage(new Uri(URI +
"Letra_B.jpg")));
```

```
            };

            Letra_C.Triggered += (sender, args) =>
            {

                Dispatcher.Invoke(() => Gestos.Source = new BitmapImage(new Uri(URI +
"Letra_C.jpg")));

            };

            Letra_D.Triggered += (sender, args) =>
            {

                Dispatcher.Invoke(() => Gestos.Source = new BitmapImage(new Uri(URI +
"Letra_D.jpg")));

            };

            Letra_E.Triggered += (sender, args) =>
            {

                Dispatcher.Invoke(() => Gestos.Source = new BitmapImage(new Uri(URI +
"Letra_E.jpg")));

            };

            Letra_F.Triggered += (sender, args) =>
            {

                Dispatcher.Invoke(() => Gestos.Source = new BitmapImage(new Uri(URI +
"Letra_F.jpg")));

            };

            Letra_G.Triggered += (sender, args) =>
            {

                Dispatcher.Invoke(() => Gestos.Source = new BitmapImage(new Uri(URI +
"Letra_G.jpg")));

            };

            Letra_H.Triggered += (sender, args) =>
            {

                Dispatcher.Invoke(() => Gestos.Source = new BitmapImage(new Uri(URI +
"Letra_H.jpg")));

            };
            /*
            Letra_A.Triggered += (sender, args) =>
            {

                Dispatcher.Invoke(() => Gestos.Source = new BitmapImage(new Uri(URI +
"Letra_I.jpg")));

            };
            */
            Salir.Click += Salir_Click;

        }


        private void Salir_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                Process[] proc = Process.GetProcessesByName("Microsoft.Gestures.Service");
                proc[0].Kill();
```

```
            }
            catch (Exception ex)
            {
                MessageBox.Show("¡¡No se pudo detener el servicio!!" + "  Error: " + ex);
            }
            Thread.Sleep(1500);
            System.Environment.Exit(0);
        }
    }
}
```

Código del sistema de detección de movimientos.

```
using System;
using System.Windows;
using Microsoft.Gestures;
using Microsoft.Gestures.Endpoint;
using System.Diagnostics;
using System.Threading;
using WindowsInput;

namespace Cliente.Gestos
{
    /// <summary>
    /// Lógica de interacción para MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {

        public MainWindow()
        {
            MainWindow ventana = this;
            //Iniciar servicio del sensor
            ProcessStartInfo Servicio = new ProcessStartInfo();
            Servicio.UseShellExecute = true;
            Servicio.FileName = "Microsoft.Gestures.Service.exe";
            Servicio.WorkingDirectory = @"C:\Users\Luis
Salgado\AppData\Roaming\Microsoft\Prague\PragueVersions\LatestVersion";
            Process.Start(Servicio);
            Thread.Sleep(3000);
            Process.Start(Servicio);
            Thread.Sleep(1000);
            InitializeComponent();
            Dinamic();
        }

        private async void Dinamic()
        {


            //Definicion de las posiciones de la mano

            //Avanzar

            var AdelanteA = new HandPose("AdelanteA",
                new FingerPose(Finger.Thumb, FingerFlexion.Open, PoseDirection.Left),
                new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching,
Finger.Thumb),
                new FingerPose(new[] { Finger.Index, Finger.Middle, Finger.Ring, Finger.Pinky },
FingerFlexion.Folded));
            var AdelanteB = new HandPose("AdelanteB",
                new FingerPose(Finger.Thumb, FingerFlexion.Open, PoseDirection.Up),
                new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching,
Finger.Thumb),
                new FingerPose(new[] { Finger.Index, Finger.Middle, Finger.Ring, Finger.Pinky },
FingerFlexion.Folded));

            //<--Avanzar
```

```
            //Retroceder

            var AtrasA = new HandPose("AtrasA",
                new FingerPose( Finger.Thumb, FingerFlexion.Open, PoseDirection.Up),
                new FingerPose(Finger.Pinky, FingerFlexion.Open, PoseDirection.Forward),
                new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching,
Finger.Thumb),
                new FingerPose(new[] { Finger.Index, Finger.Middle, Finger.Ring },
FingerFlexion.Folded));
            var AtrasB = new HandPose("AtrasB",
                new FingerPose(Finger.Thumb, FingerFlexion.Open, PoseDirection.Left),
                new FingerPose(Finger.Pinky, FingerFlexion.Open),
                new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching,
Finger.Thumb),
                new FingerPose(new[] { Finger.Index, Finger.Middle, Finger.Ring },
FingerFlexion.Folded));

            //<--Retroceder

            //Cerrar app
            var CerrarA = new HandPose("CerrarA",
                new FingerPose(new[]{Finger.Index, Finger.Ring, Finger.Pinky}, FingerFlexion.Open),
                new FingerPose( Finger.Middle, FingerFlexion.Open, PoseDirection.Up),
                new FingertipDistanceRelation ( Finger.Index, RelativeDistance.Touching,
Finger.Middle),
                new FingertipDistanceRelation(Finger.Middle, RelativeDistance.Touching,
Finger.Ring),
                new FingertipDistanceRelation(Finger.Ring, RelativeDistance.Touching,
Finger.Pinky));
            var CerrarB = new HandPose("CerrarB",
                new FingerPose(new[] { Finger.Index, Finger.Ring, Finger.Pinky },
FingerFlexion.Open),
                new FingerPose(Finger.Middle, FingerFlexion.Open, PoseDirection.Down),
                new FingertipDistanceRelation(Finger.Index, RelativeDistance.Touching,
Finger.Middle),
                new FingertipDistanceRelation(Finger.Middle, RelativeDistance.Touching,
Finger.Ring),
                new FingertipDistanceRelation(Finger.Ring, RelativeDistance.Touching,
Finger.Pinky));
            //<--Cerrar app

            //Play Pause

            var PPA = new HandPose("PPA",
            new FingerPose(Finger.Thumb, FingerFlexion.Open, PoseDirection.Up),
            new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching, Finger.Thumb),
            new FingerPose(new[] { Finger.Index, Finger.Middle, Finger.Ring, Finger.Pinky },
FingerFlexion.Folded, PoseDirection.Right));

            var PPB = new HandPose("PPB",
           new FingerPose(new AllFingersContext(), FingerFlexion.Folded));

            //<--Play Pause

            //<--Definicion de las posiciones de la mano

            //Deinicion de los gestos
            var Adelante = new Gesture("Adelante", AdelanteA, AdelanteB);
            var Atras = new Gesture("Atras", AtrasA, AtrasB);
            var Cerrar = new Gesture("Cerrar", CerrarA, CerrarB);
            var PlayPause = new Gesture("PlayPause", PPA, PPB);
            //<--Definicion de los gestos

            //Activacion del servicio de deteccion
            var gesturesService = GesturesServiceEndpointFactory.Create();
            await gesturesService.ConnectAsync();

            //Deteccion del gesto
```

```
            await gesturesService.RegisterGesture(Adelante);
            await gesturesService.RegisterGesture(Atras);
            await gesturesService.RegisterGesture(Cerrar);
            await gesturesService.RegisterGesture(PlayPause);
            //<--Deteccion del gesto

            //Ejecucion de comandos dinamicos
            //Avanzar
            Adelante.Triggered += (sender, args) =>
            {

                    InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_NEXT_TRACK);

            };
            //retroceder
            Atras.Triggered += (sender, args) =>
            {

                    InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_PREV_TRACK);

            };
            //AbrirApp
            Cerrar.Triggered += (sender, args) =>
            {

                try
                {
                    Process[] proc = Process.GetProcessesByName("Microsoft.Gestures.Service");
                    proc[0].Kill();
                }
                catch (Exception ex)
                {
                    MessageBox.Show("¡¡No se pudo detener el servicio!!" + "  Error: " + ex);
                }
                Thread.Sleep(1500);
                System.Environment.Exit(0);

            };

            //Play Pause
            PlayPause.Triggered += (sender, args) =>
            {
                InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_PLAY_PAUSE);
            };
            //<--Activacion del servicio de deteccion

            Salir.Click += Salir_Click;

        } //<--Run program


    private void Salir_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            Process[] proc = Process.GetProcessesByName("Microsoft.Gestures.Service");
            proc[0].Kill();
        }
        catch (Exception ex)
        {
            MessageBox.Show("¡¡No se pudo detener el servicio!!" + "  Error: " + ex);
        }
        Thread.Sleep(1500);
        System.Environment.Exit(0);
    }


}//clas Mein guindow
}
```

Código del sistema de envío de datos.

```
using System;
using System.Text;
using System.Windows;
using System.Net.Sockets;
using System.IO;

namespace EnvioDatos
{
    /// <summary>
    /// Lógica de interacción para MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            Vup.Click += Vup_Clk;
            Vdown.Click += Vdown_Clk;
            Play.Click += Play_Clk;
            Open.Click += Open_Clk;
        }

        void SendData(int data)
        {
            String Comand = "";
            switch (data)
            {
                case 1:
                    Comand = "nt";
                        break;

                case 2:
                    Comand = "pt";
                    break;

                case 3:
                    Comand = "pp";
                    break;

                case 4:
                    Comand = "op";
                    break;
            }
            try
            {
                TcpClient Cliente = new TcpClient();
                Cliente.Connect("127.0.0.1", 80);

                Stream stream = Cliente.GetStream();
                ASCIIEncoding ascii = new ASCIIEncoding();
                byte[] enviar = ascii.GetBytes(Comand);
                // Thread.Sleep(2000);
                stream.Write(enviar, 0, enviar.Length);
                //MessageBox.Show("Mensaje enviado");
                Cliente.Close();

            }
            catch (Exception)
            {
                MessageBox.Show("No se puede establecer la conexion con el host remoto");

            }
        }
        private void Vup_Clk(object sender, RoutedEventArgs e)
        {
            SendData(1);
        }
```

```
        private void Vdown_Clk(object sender, RoutedEventArgs e)
        {
            SendData(2);

        }
        private void Play_Clk(object sender, RoutedEventArgs e)
        {
            SendData(3);

        }
        private void Open_Clk(object sender, RoutedEventArgs e)
        {
            SendData(4);

        }

    }
}
```

Código del sistema de recepción de datos.

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


namespace Recibe_Datos
{
    /// <summary>
    /// Lógica de interacción para MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            Iniciar.Click += Iniciar_Click;
        }

        void Recibir()
        {

            String Command = "";

            try
            {
                IPAddress ip = IPAddress.Parse("192.168.43.27");
                TcpListener listener = new TcpListener(ip, 80);
                listener.Start();

                Socket socket = listener.AcceptSocket();
                 byte[] bit = new byte[100];
                 int i = socket.Receive(bit);
                 for (int a = 0; a < i; a++)
```

```
                {
                    Command = Command + Convert.ToChar(bit[a]);
                }
                switch (Command)
                {
                    case "forward":
                        InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_NEXT_TRACK);
                        break;

                    case "backward":
                        InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_PREV_TRACK);
                        break;
                    case "play":
                        InputSimulator.SimulateKeyDown(VirtualKeyCode.MEDIA_PLAY_PAUSE);
                        break;
                    case "open":
                        MessageBox.Show("Nada");
                        break;
                    case "exit":
                        System.Environment.Exit(0);
                        break;
                }


                socket.Close();
                listener.Stop();
                Recibir();
            }
            catch (Exception)
            {
                MessageBox.Show("Error de conexion");
            }
           // main();
        }

        private void Iniciar_Click(object sender, RoutedEventArgs e)
        {

            Recibir();
        }
    }
}
```

Artículo 1:

ESTUDIO COMPARATIVO DE SISTEMAS DE DETECCIÓN Y ANÁLISIS DE MOVIMIENTO HUMANO

ISC. José Luis Molina Salgado[1], Dr. Máximo López Sánchez[2] y Ing. David Ulises Ríos Mendoza[3]

***Resumen*— Las Interfaces Naturales de Usuario (NUI: Natural User Interfaces) se basan en los movimientos que el usuario puede realizar de manera natural. Debido a esto no es posible utilizar cualquier sistema para la interacción humano-computadora, por ello es preciso analizar los sistemas de detección humana tomando en cuenta las diferencias entre hardware y software para comparar los sistemas existentes que utilizan el reconocimiento de patrones para la identificación de movimiento. En este estudio se analizaron distintos sistemas de detección humana, algunos de estos sistemas utilizan cámaras de video o sensores de profundidad, esto permite tener sistemas distintos en hardware y software de los cuales se obtiene una comparación más destacada en cuanto a detección humana.**

***Palabras clave*— NUI, sensores, gestos, profundidad, métodos de detección.**

Introducción

Cuando se trata de manipular interfaces computacionales, existen diferentes maneras de hacerlo, existen dispositivos físicos como los teclados, ratones, palancas, botones etc. sin embargo también existen interfaces con las que no es necesario ejercer una fuerza sobre ellos o bien no es necesario que se tenga contacto físico con ellos, esta investigación recopila información de distintas investigaciones y desarrollos realizados que tienen como objetivo la manipulación de interfaces, esto con el fin de entender el funcionamiento y el método de desarrollo de cada una de estas. Hoy en día la tecnología permite que los desarrollos de este tipo de sistemas sean muy variados y por eso es preciso saber si las técnicas modernas y las antiguas tienen un funcionamiento adecuado cuando se necesita desarrollar un sistema que sea capaz de detectar movimientos y gestos de un usuario.

Estado del arte

Las Interfaces Naturales de Usuario (NUI) son un tipo de interfaz con las que se puede interactuar con un sistema o aplicación omitiendo el uso de interfaces físicas de entrada, estos pueden ser ratón, teclado, panel táctil entre otros las NUI se caracterizan por hacer uso del cuerpo humano (manos, dedos, torso etc.) para interactuar con los sistemas y aplicaciones, esto se logra a través de la detección del movimiento del usuario.

Los sistemas computacionales para la detección de gestos han sido diseñados de distintas maneras, estos pueden ser creados con cámaras de video o con dispositivos elaborados a base de distintos sensores que son especialmente diseñados para dicho trabajo. A continuación, se muestran diferentes trabajos realizados con estas técnicas.

Como se puede observar en (Han & Rashid, 2016), se desarrolló un sistema de reconocimiento de gestos basado en una arquitectura ARM Cortex-A8 en la cual se instaló un sistema operativo Linux donde se realizó un estudio y la medición en tiempo real de las respuestas a las pruebas realizadas. Los resultados de estas pruebas indicaron que el sistema realizado otorga una respuesta en tiempo real del reconocimiento de los gestos de las manos y comandos de voz, lo que permite el control de dispositivos ZigBee sobre un canal de comunicación inalámbrico.

En el artículo (Tchoketch Kenbir & Bouhedda, 2016), se muestra un sistema para la detección de gestos y generación de instrucciones que se basa en el uso de MatLab en conjunto de un dispositivo construido con Arduino, donde las imágenes adquiridas son tratadas con la transformada de Houg para definir cuáles son los objetivos que se desean. Las señales de control se generan y se envían a través de una conexión inalámbrica para hacer que el dispositivo se mueva en la dirección deseada.

Otro trabajo que se relaciona es (Gergely , Ujbányi, & József , 2016) en el cual el sistema que se desarrollo tiene como objetivo la detección de gestos a partir del movimiento de las manos, este sistema incorpora sensores de proximidad para medir las distancias entre un dispositivo y las manos del usuario posteriormente se lleva a cabo un procesamiento de todos los datos en un sistema computacional ejecutado en una aplicación de escritorio.

Otros dispositivos con los que se puede realizar la detección de gestos que permiten la interacción con sistemas computacionales son los que se muestran en (Csapo & Kristjánsson, 2016) donde se analizan los dispositivos táctiles

[1] El ISC. José Luis Molina Salgado es Estudiante de Maestría en el Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos. joseluis.molina@cenidet.edu.mx **(autor corresponsal)**
[2] El Dr. Máximo López Sánchez es Profesor Investigador en el Centro Nacional de Investigación y Desarrollo Tecnológico maximo@cenidet.edu.mx
[3] El Ing. David Ulises Ríos Mendoza es Estudiante de Maestría en el Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos. david.rios@cenidet.edu.mx

y hápticos y se da a conocer los distintos experimentos que se realizaron con el brazalete "Myo" para controlar la retroalimentación auditiva en tiempo real. El objetivo de esta investigación es encontrar una manera de que los usuarios con discapacidades visuales puedan interactuar con sistemas de cómputo.

Hoy en día el uso de las NUI cada vez es más común en los sistemas de interacción sin embargo aún no existe una técnica que otorgue resultados perfectos a la hora de realizar la detección del cuerpo humano, en (Lee & Tanaka, 2013) se explica la propuesta de un método de interacción natural con un sistema además del diseño de los gestos de la mano utilizando la identificación de dedos utilizando dispositivos de múltiples sensores como Kinect.

Los autores en (Sueaseenak, Khawdee, Pakornsirikul, & Sukjamsri, 2017) ponen a prueba el funcionamiento del brazalete "Myo" en conjunto de una aplicación de extracción de características basada en electromiografía (EMG) de varios canales, de esta manera son capaces de extraer la información de los músculos del brazo durante la representación de 6 gestos de la mano incluyendo mano abierta, mano cerrada mano flexionada y la mano en posición normal. Los criterios utilizados fueron el índice de decibelios y el criterio de dispersión de los datos del EMG con lo que demuestran que el uso del dispositivo es adecuado para la detección de gestos.

La investigación (Gunawardane, 2016) muestra el desarrollo de un sistema de detección de gestos utilizando Leap Motion Controller, este sistema realiza el seguimiento de tres de los dedos de una mano, así como la muñeca para controlar un mecanismo de dedos blandos que imita el movimiento de la mano del usuario, todo esto se logró gracias a la precisión de los sensores y la programación del sistema.

En algunas ocasiones la intervención humana para llevar a cabo una tarea es indispensable, sin embargo, día con día se crean nuevas alternativas para minimizar el riesgo humano, tal es el caso de (Bendale & Kharat, 2017) en donde desarrollaron un sistema de control de gestos para el control de un brazo robótico que tiene como finalidad la manipulación de materiales peligrosos en estaciones nucleares. El sistema está compuesto por componentes electrónicos que detectan los movimientos del usuario y un brazo robótico que replica dichos movimientos en tiempo real, los dispositivos se comunican a través de una conexión inalámbrica.

El articulo (Deng, Chang, & Jian, 2017) muestra un panorama en donde se da a conocer un sistema multimodal en donde se hace uso de dos dispositivos para la manipulación de una interfaz computacional, utilizan un sensor para el rastreo del ojo humano y un sensor de profundidad para el control de gestos de la mano, con esto demuestran que el hacer uso de distintos sensores en conjunto, mejora la precisión y la experiencia del usuario al realizar la manipulación de interfaces.

Algunos investigadores centran sus trabajos en el desarrollo de sistemas que ayudan a la población tal es el caso de (Antón, Kurillo, Goñi, Illarramendi, & Bajcsy, 2017) en donde se apoyan del uso de las nuevas tecnologías para la detección de gestos humanos para desarrollar un sistema de ayuda a personas que necesitan una rehabilitación, el sistema hace uso del sensor de profundidad Kinect para la detección de gestos, además de realizar una conexión remota entre dos de estos sistemas, uno para la persona que necesita la rehabilitación y por otro lado un fisioterapeuta que indica gráficamente cuales son los movimientos que debe realizar el paciente.

Métodos utilizados en las investigaciones

Con las investigaciones realizadas con anterioridad se muestran distintas formas de llevar a cabo la detección y el análisis del movimiento humano, estos métodos poseen características similares, sin embargo, se pueden dividir en 2 categorías principales.

- Sistemas desarrollados que utilizan directamente la visión artificial
  - Estos sistemas generalmente están compuestos por una cámara de video para capturar las imágenes, un método de procesamiento de imágenes, el uso de algoritmos de clasificación, entrenamiento de los sistemas y una comparación de los datos de entrada con los datos de la base de conocimiento.

- Sistemas desarrollados que utilizan un lenguaje de programación para interpretar la información proveniente de sensores de profundidad.
  - Estos sistemas están principalmente caracterizados por hacer uso de sensores de profundidad, estos dispositivos cuentan con un Kit de Desarrollo de Software (SDK) que provee las herramientas necesarias para el desarrollo de aplicaciones en distintas plataformas.

A continuación, se muestras algunos de los métodos utilizados en las investigaciones.

En la figura 1 se observa que las fases del sistema constan de la recolección de datos se realiza a través de una cámara, la información pasa a una tarjeta de desarrollo en donde se lleva a cabo un procesamiento digital de

imágenes (Figura 2) que permite tanto el entrenamiento de los algoritmos y la detección de los gestos proporcionados por la persona que hace uso del sistema.

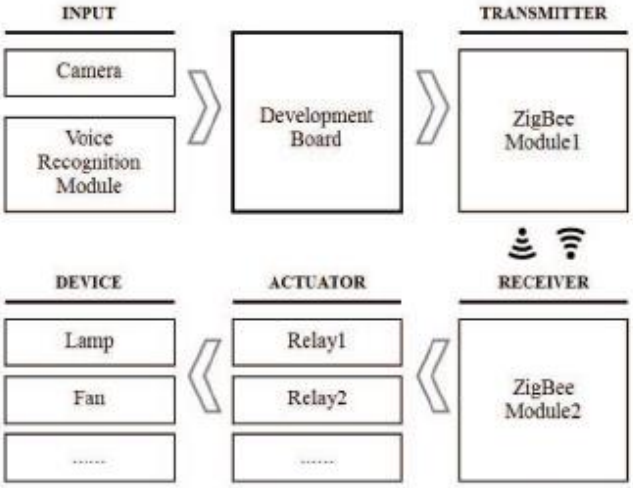

Figura 1. Método utilizado en (Han & Rashid, 2016).

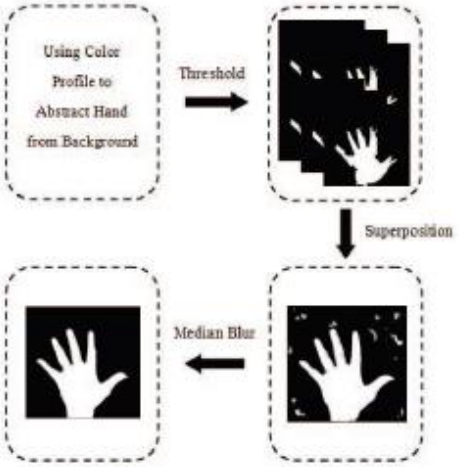

Figura 2. Procesamiento digital de imágenes (Han & Rashid, 2016).

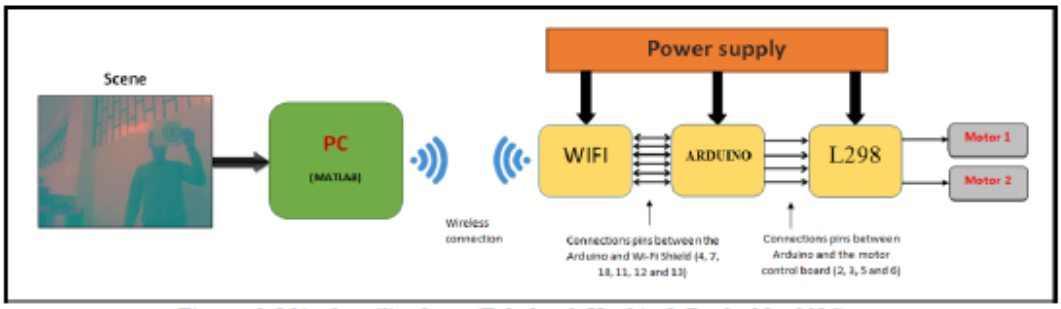

Figura 3. Método utilizado en (Tchoketch Kenbir & Bouhedda, 2016).

En la figura 3 se muestra que la recolección de datos se realiza a través de una cámara, estos pasan a un procesamiento llevado a cabo en Matlab en donde se manipulan para obtener la información de detección la cual es convertida en instrucciones que se envían a través de una conexión inalámbrica a un dispositivo que ejecuta las instrucciones que el usuario haya indicado como entrada al sistema.

En la figura 4 se puede observar que el sistema utiliza un dispositivo que cuenta con sensores de profundidad para realizar la detección, estos sensores cuentan con un procesador de imágenes que junto con el SDK se encargan de extraer la información de entrada, esto permite que con solo realizar la programación de la detección puedan obtenerse los datos de entrada y programar las acciones de salida.

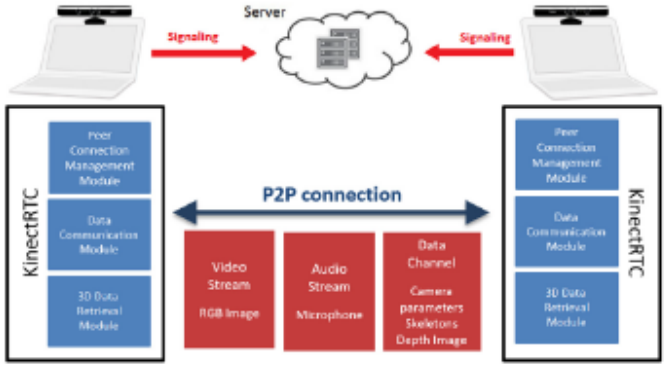

Figura 4. Método usado en (Antón, Kurillo, Goñi, Illarramendi, & Bajcsy, 2017).

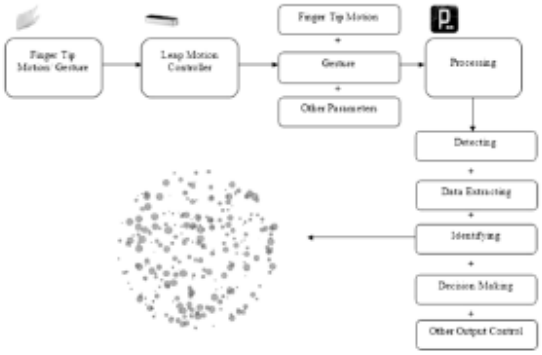

Figura 5. Método usado en (Gunawardane, 2016).

La figura 5 muestra un sistema en el cual el método para la detección de movimiento está basado en el uso del sensor Leap Motion Controller, a través de los datos de entrada el sensor se encarga de analizarlos y enviar la información del gesto detectado con sus respectivos parámetros, posteriormente se ingresan al sistema Processing en donde se lleva a cabo la detección del gesto para poder otorgar la salida correspondiente a los datos de entrada.

Los sistemas de detección y análisis de movimiento humano pueden estar compuestos por diferentes métodos, sin embargo, las nuevas tecnologías dan la oportunidad de trabajar en sistemas de detección sin la necesidad de realizar el procesamiento digital de imágenes, el entrenamiento de algoritmos de clasificación y la desventaja que tienen las cámaras convencionales cuando se trata de utilizarlas en sistemas con poca o mucha luminosidad.

En la tabla 1 se hace una comparación de los sistemas de detección de movimiento.
*PDI → Procesamiento Digital de Imágenes

Investigación	Sensor principal	SDK	PDI escrito por el programador	Entrenamiento de algoritmos	Resultados exitosos
(Antón, Kurillo, Goñi, Illarramendi, & Bajcsy, 2017)	Kinect Sensor V1	✓	X	X	✓
(Bendale & Kharat, 2017)	Flex Sensor	X	X	X	✓
(Csapo & Kristjánsson, 2016)	Myo Bracelet	X	X	X	✓
(Deng, Chang, & Jian, 2017)	Leap Motion Controller	✓	X	X	✓
(Gergely , Ujbányi, & József , 2016)	Ultrasonic Sensor	X	X	X	✓
(Gunawardane, 2016)	Leap Motion Controller	✓	X	X	✓
(Han & Rashid, 2016)	Camera	X	✓	✓	✓
(Lee & Tanaka, 2013)	Kinect Sensor V1	✓	X	X	✓
(Sueaseenak, Khawdee, Pakornsirikul, & Sukjamsri, 2017)	Myo Bracelet	X	X	X	✓
(Tchoketch Kenbir & Bouhedda, 2016)	Camera	X	✓	✓	✓

Tabla 1. Comparación de sistemas de detección.

Es claramente notable que para llegar a tener éxito cuando se desarrolla un sistema de detección de movimiento no es indispensable utilizar un método o un dispositivo en concreto, son muchas las formas de hacerlo, sin embargo, cada una de estas formas se puede optimizar para el enfoque concreto del desarrollo deseado.

Conclusiones

Tomando en cuenta la presente investigación se comprende que los sistemas para la detección de movimiento pueden ser desarrollados con diferentes metodologías o con una combinación de dos o más, así mismo se pueden hacer uso de distintas herramientas que permitan la detección adecuada para una aplicación en específico.

Cabe mencionar que los nuevos dispositivos de detección facilitan el desarrollo de este tipo de sistemas con lo cual se abren nuevas posibilidades para la generación de aplicaciones que sean capaces de identificar movimientos, gestos o acciones del usuario.

Referencias

Antón, D., Kurillo, G., Goñi, A., Illarramendi, A., & Bajcsy, R. (2017). Real-time communication for Kinect-based telerehabilitation. Future Generation Computer Systems 75.

Bendale, T., & Kharat, V. (2017). A design of Gesture controlled mobile robot with robotic arm for nuclear environment. International Conference on Nascent Technologies in the Engineering Field (ICNTE-2017).

Csapo, A., & Kristjánsson, Á. (2016). Evaluation of Human-Myo Gesture Control. 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2016), 415-420.
Deng, S., Chang, J., & Jian, J. (2017). Gaze Modulated Disambiguation Technique for Gesture Control in 3D Virtual Objects Selection. Cybernetics (CYBCONF), 2017 3rd IEEE International Conference on.
Gergely, S., Ujbányi, T., & József , K. (2016). Cost-effective hand gesture computer control. 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2016), 239-234.
Gunawardane, M. M. (2016). The Development of a Gesture Controlled Soft Robot Gripping Mechanism. Faculty of Engineering Technology, The Open University of Sri Lanka.
Han, X., & Rashid, M. (2016). Gesture and Voice Control of Internet of Things. 11th Conference on Industrial Electronics and Applications (ICIEA), 1991-1995.
Lee, U., & Tanaka, J. (2013). Finger Identification and Hand Gesture Recognition. Asia-Pacific Computer and Human Interaction.
Sueaseenak, D., Khawdee, C., Pakornsirikul, N., & Sukjamsri, C. (2017). A Performance of Modern Gesture Control Device with Application in Pattern Classification. International Conference on Control, Automation and Robotics, 428-431.
Tchoketch Kenbir, S., & Bouhedda. (2016). Gesture Control of Mobile Robot Based Arduino. 8th International Conference on Modelling, Identification and Control (ICMIC-2016), 1081-1085.

Artículo 2:

(IJACSA) International Journal of Advanced Computer Science and Applications,
Vol. 8, No. 10, 2017

Design and Implementation of a Communication System and Device Aimed at the Inclusion of People with Oral Communication Disabilities

Dr. Máximo López Sánchez
Computer Science
Cenidet-TecNM
Cuernavaca, Mexico

José Luis Molina Salgado
Computer Science
Cenidet-TecNM
Cuernavaca, Mexico

Dr. Juan Gabriel González Serna
Computer Science
Cenidet-TecNM
Cuernavaca, Mexico

Melisa Hernández Salinas
Computer Science
Cenidet-TecNM
Cuernavaca, Mexico

***Abstract*—Disability is part of human condition; it discriminates people who have this complication. The present work was carried out due to this and an experience in our research center. A prototype was designed and build that allows eye signals to be sent to a mobile device, where through a computer system, it was possible to generate an appropriate dialogue mechanism to respond to this challenge. The results allow us to open up an area of opportunity for a contribution in the inclusion of people with disabilities.**

***Keywords*—Communication; system; disabilities; device; oral**

I. Introduction

Disability is a part of human condition: almost all people suffer some sort of passing or permanent disability at some point in their life, and those who reach senility will experience growing functioning difficulties in their organism, which in many cases causes contact loss with their familiar environment, just as in the case of young people, the critical mass is no longer used due to the fact that they have no way of communicating their knowledge.

Information from the World Health Organization in a 2010 report estimates that over one billion people live with some form of disability. According to the World Health Survey, about 785 million people, (15.6%), aged 15 and over live with a disability. It shows that, out that of the estimated total of people with disabilities, 110 million, (2.2%), have very significant functioning difficulties [1].

On the other hand, the prevalence of disability in Mexico in 2014 was 6%, according to [2], (National Survey of Demographic Dynamics 2014), information. This means that 7.1 million of the country's inhabitants cannot or find it very difficult to do any of the evaluated activities. Reported disabilities: concentrate 42.4% (walk, walk up or down using their legs and seeing, even when wearing glasses); learning, remembering or concentrating, listening and moving or using arms or hands, together total 36.4%, while bathing, dressing, eating, talking or communicating, add up to 21.2% [3].

It's important to mention that during the 2014-2016 period, a student with oral communication problems attended his master's degree program at CENIDET - TecNM, (National Centre for Research and Technological Development - National Technological of Mexico), experimenting with his own disability during his research of Master's Thesis.

II. Related Works

Some computer systems, that help people with speech problems to communicate with the people in their environment, were checked. These are: 1) **Verbo** [4]. It is a software that can be installed in computers with Windows or Android operating systems and can be adapted to different pushbutton devices such as special keyboards, keypads, and eyepieces among others. The user constructs the phrase he/she wishes to communicate using pictograms. Another similar system is 2) **IRISBOND** [5]. That can be used by people suffering from ALS (atherosclerosis), spinal cord injury, paraplegia or other conditions, thanks to the implementation of an ocular tracking technology with which the user can manipulate the different systems installed on Windows such as: Facebook, WhatsApp, Microsoft Word, etc., it also incorporates software that allows sentence building through a virtual keyboard. 3) The third option is **AraBoard** [6]. This allows the creation, edition and use of communication boards for different devices (computers, smartphones or tablets), as well as different operating systems, in this system, users with speech problems structure their messages selecting pictograms. 4) **Eyecan** [7]. The project was introduced as a mouse that is controlled by sight and together with a virtual keyboard quadriplegic users can write messages to communicate. 5) **Alternative communication system for people with cerebral paralysis who can read and write with the support of an Android mobile device** [8]. It

proposes the use of an Arduino made button and software that installs on a mobile device running Android, which uses a virtual keyboard where the user selects one by one the letters or words that make up the phrase that he wants to express one by one to ease letter selection, a swipe on the keyboard, first vertical and then horizontal, is made, stops are indicated via a button that adapts to the user's mouth and is activated by means of oral muscle movement. 6) Lastly, **alternative and augmentative communication system for people with phonetic problems, with support of an Android mobile device** [9] and 7) **SAAC - Droid, a tool to help communication** [10]. They are systems that are installed on Android devices and use touchscreens. The first structure dialogues through pictogram selection, while the second uses a virtual keyboard. Users of these systems must have mobility in upper limbs that allows device use. Our alternative communication proposal consists of two parts: development of a non-invasive device and a computer system.

III. Device Development

To solve the problem in terms of communication, a methodology was designed, (Fig. 1), which has five activities that range from a current solution search to the implementation of a functional device.

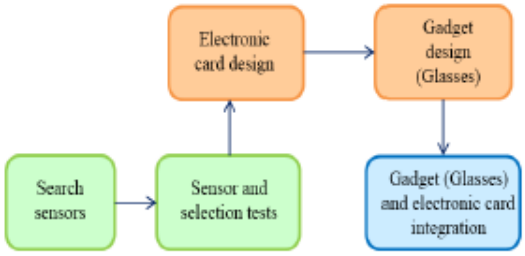

Fig. 1. Methodology.

A. Sensor Search

In the search for sensors, it was determined that the most adapted option to the solution of our problem were the reflexive optical sensors, due to the fact that people with motor disabilities have limited movements in their body parts, this is, that they have the ability to perform certain hand, finger, head and mainly eye movements. So, we focused our work on this last movement.

Considering this type of sensor, we found that the most relevant ones are the following:

1) CNY70

The CNY70 is a short-range infrared sensor based on a light emitter and a receiver, both pointing in the same direction, whose operation is based on object reflection capacity and beam detection reflected by the receiver [11].

2) IR Sharp Sensors

An Infrared sensors family for object detection offers distance information according to the range in which it receives signals, specifically in the case of the GP2D02 and GP2D12 models. These sensors work through light triangulation, which bounces off an object and according to the distance of that object, the angle tends to differ [12].

3) QRD1114

It is an optical sensor composed of an infrared diode and a phototransistor, which is responsible for detecting infrared light reflection. In general, its use achieves light change detection in black and white colors; its measurement range is between 0.5 to 1 centimeters [13].

4) TSOP4838

It is an infrared sensor used to receive infrared signals from remote controls; they are normally used in home appliances. The sensor has an amplifier circuit and a 38 Khz oscillator inside, which allows signal reception even in the presence of intense light sources [14].

5) Diode Emitter and Phototransistor

The IR LED (emitter diode) is responsible for emitting a sort of electromagnetic radiation better known as infrared light. The phototransistor differs from a common transistor because its base has been replaced by a photosensitive crystal that regulates the flow of collector-emitter current according to the light shed on it [15].

6) TCRT5000 and TCRT5000L

The TCRT5000 and TCRT5000L are reflective sensors that include an infrared emitter and a phototransistor in a lead pack that blocks visible light. The package includes two mounting clips [16].

B. Sensor Selection

So as to select the right sensor for our solution we considered the following points:

- Size: Size was considered because the sensor will be used for eye blinking detection without obstructing the user's vision.
- Distance detection: This variable was taken into account, since the sensor must detect eye blinking, thus, it must be at a certain distance from the eyewear lens to the user's eye.
- Operational voltage: an adequate voltage was taken into account due to the characteristics of device operation and the energy expenditure it represents.
- Response time: It was considered because response speed is required to be as fast as possible.

From these characteristics, the following comparative table was obtained (Table 1), where each of the features are analyzed and from which the best suited sensor for the job was selected.

TABLE I. COMPARATIVE TABLE (SENSORS)

Sensor	Characteristics			
	Dimensions	*Range detection*	*Operation Voltage*	*Response time*
CNY70	7 x 7 x 6 mm	< .5 mm	5V	
GP2D02	37x14x14.4mm	10 a 80 cm.	4.4 a 7V	70 µs
GP2D12	37x14x14.4mm	10 a 80 cm.	4.5 a 5.5V	32 µs
QRD1114	6.1x4.39x4.65 mm	0.050" (1.27 mm)	5V	10 µs, 50 µs
TSOP4838	3 cm x 0.5 cm	35m	3 a 6V	9 µs
Emitting Diode and Phototransistor	5 y 3 mm	——	3 a 5V	——
TCRT5000L	10.2 x 5.8 x 7 mm	2.5 mm	5V	10 µs

C. Electronic Card Design

For the electronic card design, two PCB (Printed Circuit Board) design software were used, these are:

1) PCB Wizard

It is an educational area designed program, which allows electronic circuit schematics creation and thus easily obtain one or two-sided printed circuit designs [17].

2) Fritzing

It was created under the Processing and Arduino principles, allowing designers to document their prototypes based on Arduino and create printed circuit diagrams for later manufacture.

It is an open source software and has libraries with the majority of components that include the Arduino ones, connection boards, LED, motors, displays, among others [18].

D. Gadget Design (Glasses)

Within this stage a previous analysis was carried out on some commercial glasses type devices to take into account size, weight, ergonomics and materials used in them.

The analyzed devices are the following:

- Google Glass [19].
- Epson Moverio bt-200 [20].
- Wink Glasses mod. 2013 [21].
- Glassup [22].
- Vuzix Smarth Glasses [23].

For gadget modeling, AutoCAD software, which provides tools to develop 3D models, was used.

Development steps were as follows:

- A first pencil sketch was designed (Fig. 2).
- The first sketch was modeled in AutoCAD (Fig. 3).
- The 3D model was printed.
- The reflective optical sensors were placed on the lens of the glasses.
- The design was modified due to the tests performed with the sensors (Fig. 4).
- A final prototype was established (Fig. 5).

Fig. 2. First pencil sketch glasses.

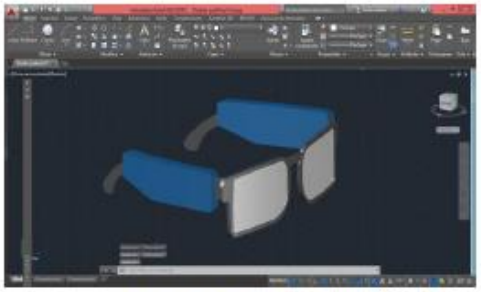

Fig. 3. First sketch in AutoCAD.

Fig. 4. Second design seen from the Cube Print platform ready for printing.

Fig. 5. Final prototype.

E. Electronic Card and Gadget Integration

Once the lens was printed, the electronic card was inserted inside the plate holder, which is placed on the right side of the glasses and the corresponding connections to the battery holder, placed on the left side, were made, so as to finally have a functional prototype which integrates all the parts to fulfill their integral function.

To obtain better sensor readings, these must have a calibration function, so that the detected information received from the sensors is adapted to the characteristics of the user. For this, tests were carried out to verify reading efficiency in sunlight, white LED and dark environments (Table 2).

Once the gadget is duly integrated, data detected by the sensor is sent wirelessly to a system where it is converted to letters, so that later on the user can form words and thus establish communication with people in their environment.

Environment 1 = Environment with sunlight

Environment 2 = White led light

Environment 3 = Darkness

TABLE II. EXPOSURE OF SENSORS TO DIFFERENT ENVIRONMENTS

User	Environment 1	Environment 2	Environment 3
1	10/10	10/10	10/10
2	10/10	10/10	10/10
3	10/10	10/10	10/10
4	7/10	7/10	7/10
5	10/10	10/10	10/10
6	10/10	10/10	10/10
7	10/10	10/10	10/10
8	10/10	10/10	10/10
9	10/10	10/10	10/10
10	10/10	10/10	10/10
Percentage	**97%**	**97%**	**97%**

It should be noted that for these trials, tests were made on 10 people aged between 22 to 28, who were not necessarily disabled, and each user was told to blink 10 times to determine the amount of blinks detected by the sensors, and based on this, know if the sensor position calibration in the lens of the gadget was correct.

IV. APPLICATION (SOFTWARE)

A computer system was developed to use the device with a mobile, it is able to communicate wirelessly with the device in charge of sending the user's signals, allows obtained information gathering based on user blinking by executing actions such as: write a letter, predict a word, formulate sentences and interpret in a sonorous way what the user writes. The developed system consists of:

- Wireless communication module.
- Data transformation algorithm.
- Dictionaries.
- Interpretation algorithm.
- Matching algorithm.
- Audio playback module.

A. Wireless Communication Module

The system makes use of the computer device's wireless technologies on which it is installed; this system is compatible with Wi-Fi or Bluetooth communication.

"Bluetooth® is a low-power wireless connectivity technology used to stream audio, transfer data and broadcast information between devices." [24].

"Wi-Fi is a wireless networking protocol that allows devices to communicate without Internet cables. It is technically an industry term that represents the type of wireless local area network (LAN) protocol based on the 802.11 IEEE network standards." [25].

Technologies such as Bluetooth and Wi-Fi are ideal for a system that requires communication at a distance no greater than 10 meters; in addition, these technologies are quite common in everyday computing devices.

The system is programmed to send and receive information from the signal detection device, so that the data can be obtained in real time and does not depend on the complementary use of some other type of technology. Fig. 6 shows a schematic of the module composition.

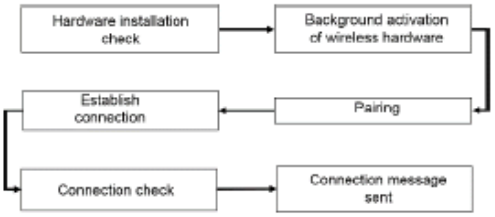

Fig. 6. Connection module.

B. Data Transformation Algorithm

The system has a module responsible for transforming data that is received from the device, this data is entered into different variables to be processed and know what instructions the user wants executed. The module is in charge of identifying the instruction taking into account the time that elapses between the sending of the signals coming from the device and where the obtained signals take values from A and B.

The listed instructions are executed according to data reception time, where: Instruction 1 (i1), Instruction 2 (i2) ... Instruction n (in), are established by knowing the time (t) in milliseconds (ms) that exists between receiving a value A (t1) with a value B (t2).

The instructions are executed if and only if:

1) i1

$$100\ \text{ms} < t1 - t2 <= 1000\ \text{ms}$$

2) i2

$$1000\ ms < t1 - t2 <= 2000\ ms$$

3) i3

$$2000\ ms < t1 - t2 <= 3000\ ms$$

4) in

$$n\ ms < t1 - t2 <= n\ ms$$

The following shows the work scheme of the data transformation algorithm (Fig. 7).

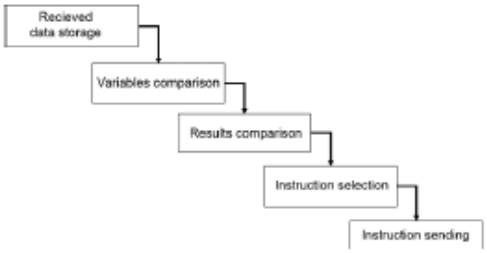

Fig. 7. Data transformation algorithm.

C. Dictionaries

The system has three types of dictionaries, General, Temporal and Custom, which are used for storing relevant information, (words), that will be used by the user, making their writing easier (Fig. 8).

"A Data Dictionary is a collection of names, definitions, and attributes about data elements that are being used or captured in a database, information system, or part of a research project. It describes the meanings and purposes of data elements within the context of a project, and provides guidance on interpretation, accepted meanings and representation [26].

Fig. 8. Corpus and dictionaries.

At first the general dictionary was composed of a corpus of the most used words in Spanish and is available on the official website of the Royal Spanish Academy [27], this information is the basis for custom and temporary dictionary creation, (Fig. 9). Later the English language was added, and it is intended to incorporate as many languages as necessary.

D. Interpretation Algorithm

"Morse code is a code or communication system that allows telegraphic communication through the transmission of electrical impulses of various lengths or by visual means, such as light, sound or mechanics. This code consists of a series of points; stripes and spaces, which when combined together can form words, numbers and other symbols" [28].

Within the system, Morse code allows anyone in the world to make use of it, since implemented Morse code is the international version; it has a great variety of numbers, symbols and letters that adapt to most languages.

According to instructions received from the data transformation algorithm, character writing can be established in Morse code, considering points and lines for word and sentence generation. The international Morse code also allows the user to write with a single alphabet in different languages. Fig. 9 shows the code used in the system.

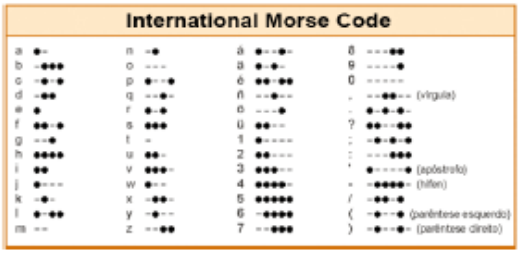

Fig. 9. Morse code (Daisy, 2017).

E. Matching Algorithm

This algorithm is in charge of using and manipulating different dictionaries, when an alphabetic character written by the user is presented, the algorithm makes an input data selection to find one or more word matches that can be used by the user, if the user wants to write a word of two or more characters, the algorithm receives as input a single character that serves as an indicator to perform the search for a suggestion.

The word matches can be defined as:

- C = Initial character
- P = Word of the dictionary
- Pc = Complement of the word = P – C

Thus,

- Co = Coincidence = C + Pc

Subsequently a selection of suggestions considered as foremost is made, based on all coincidences found in dictionaries and evaluating the nominal frequency of the corpus of words, to obtain the words with greater use probability in the selected language. Fig. 10 is a graphical representation of the matching algorithm operation.

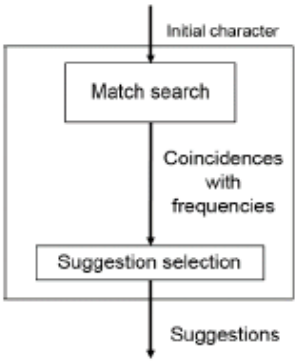

Fig. 10. Matching algorithm.

F. Audio Playback Module

To reproduce in a sonorous way the texts elaborated or prepared by the user, the system has a text to voice conversion module, to achieve this, it uses tools provided by Google. This module is responsible for storing the textual information in an **O** variable and passing it as a parameter to a process that executes the audio playback tools so that the user and the people around him can listen to the previously written messages.

V. Tests

For testing we relied on the "**Con Nosotros**" foundation, which specializes in basic education of people with cerebral paralysis. Test reactions were applied to enrolled students at the institution.

A test plan was built to validate the system; this consisted of three phases where an evaluation of each phase was carried out.

The first phase aimed to enable the user to identify the precise interpretation time of each selected option. They were asked to practice at least ten actions for each of the available options. The results were evaluated and if at least 70% of successful values were obtained, the second phase would proceed; Table 3 shows the results obtained.

TABLE III. Selected Options Table

User	Point	Dash	Selection mark
1	10/10	10/10	10/10
2	10/10	10/10	10/10
3	9/10	8/10	7/10
Average	9.6/10	9.3/10	9/10
Percentage	**96.6%**	**93.3%**	**90%**

As can be seen in Table 3, users 1 and 2 results were 100%, unlike user 3, since the latter suffered slight involuntary head movements, which prevented him from maintaining his head position in a single place, as a result of this, the inclination of his head created a shadow that didn't allow continuous light illumination in the controlled environment, that is why the signals obtained by the sensors placed in the gadget didn't have correct eye blinking detection.

The second phase aimed to enable the user to select a button in the application interface. The test in this phase consisted in asking the user to select three buttons that had been previously determined, ten times, these were located in different places on the interface, they had to make seven attempts at pressing each button correctly so that this test could be successfully completed. The results of this test can be seen in Table 4.

TABLE IV. Second Phase Results

User	Button "A"	Button "R"	Button "Z"
1	10/10	10/10	10/10
2	10/10	10/10	10/10
3	7/10	9/10	9/10
Average	9/10	9.66/10	9.66/10
Percentage	**90%**	**96.6%**	**96.6%**

Lastly, in the third phase, a full use of the application was made. The test consisted in asking the user to write longer phrases, which were made up of more than three words and would then be reproduced using a speech synthesizer; the phrases that were used for the tests are shown in Table 5:

TABLE V. Phrases

No.	Phrase
1	Hello, good day
2	My name is "User name"
3	My favorite food is "User's Favorite Food"

Test results were:

As can be seen in Table 6, users 1 and 2 correctly completed any of the 3 sentences, while user 3, due to the involuntary movements of his head, failed to coordinate the movement of his eyes to select the desired button.

As for the average writing times, writing each sentence in ten tries was considered, taking the average time of the first five attempts and the last five attempts. Tables 7 and 8 shows the results obtained.

TABLE VI. Completed Phrases

User	Phrase 1	Phrase 2	Phrase 3
1	10/10	10/10	10/10
2	10/10	10/10	10/10
3	10/10	8/10	7/10
Average	10/10	9.3/10	9/10
Percentage	**100%**	**93.3%**	**90%**

TABLE VII. First Five Attempts

User	Elapsed time in phrase 1	Elapsed time in phrase 2	Elapsed time in phrase 3
1	6.46	8.09	8.19
2	8.55	9.33	9.56
3	10.57	11.37	12.51
Average	**8.49**	**9.46**	**10.22**

TABLE VIII. LAST FIVE ATTEMPTS

User	Elapsed time in phrase 1	Elapsed time in phrase 2	Elapsed time in phrase 3
1	1.21	2.12	2.1
2	2.07	3.04	4.03
3	2.27	6.24	10.07
Average	**2.07**	**3.43**	**5.26**

A reduction of more than 6 minutes for sentences 1 and 2 can be seen, a little more than 4 minutes in sentence 3.

VI. CONCLUSIONS

During this project's making, an effective proposal was sought for phrase and sentence writing, it was considered that one way is to obtain input as binary data, so taking into account the availability of Morse code, it was decided to implement a methodology in where its only necessary to select the desired object on the screen, from the first of the row and scroll the objects until you reach the desired element, this way writing time was reduced in the application.

When testing in controlled environments, with users suffering from cerebral paralysis, it was observed that this system is useful only for users who, besides having full control over their blinking, do not present involuntary movements that affect excessively their head position, since when tilting the head down, the sensors enter the shadow generated as a result of this inclination, giving an incorrect eye blinking reading due to this change of light.

However, according to the opinion of the three users the application is simple to use and easy to memorize when looking for a desired letter.

There is an area of improvement in both the device and the computer system and work will continue. On the other hand, there are some limitations to carrying out more tests, due to the fact that because it is often difficult for institutions that tend to this stratum of people to have enough time to apply tests.

ACKNOWLEDGEMENT

The result of this work was sponsored by cenidet-TecNM

REFERENCES

[1] WHO, "World Health Organization," [Online]. Available: http://www.who.int/es/. [Last access:25 08 2017].

[2] INEGI, "ENADID," 2014. [Online]. Available: http://www.beta.inegi.org.mx/proyectos/enchogares/especiales/enadid/2014/. [Last access:20 08 2017].

[3] INEGI, "INEGI," [Online]. Available: http://internet.contenidos.inegi.org.mx/contenidos/productos/prod_serv/contenidos/espanol/bvinegi/productos/nueva_estruc/702825090203.pdf. [Last access:20 08 2017].

[4] "eneso," 2017. [Online]. Available: http://www.eneso.es/verbo/. [Last access:21 09 2017].

[5] "irisbond," 2016. [Online]. Available: http://www.irisbond.com/productos/irisbond-duo. [Last access:21 09 2017].

[6] "GIGA," 2015. [Online]. Available: http://giga.cps.unizar.es/affectivelab/araboard.html. [Last access:21 09 2017].

[7] "samsung," 2014. [Online]. Available: https://news.samsung.com/global/samsung-electronics-introduces-eyecan-next-generation-mouse-for-people-with-disabilities. [Last access:21 09 2017].

[8] J. A. M. Mancilla, Sistema de comunicación alternativa para personas con parálisis cerebral que saben leer y escribir con el apoyo de un dispositivo móvil con Android, 2013.

[9] G. C. Martínez, "Sistema alternativo y aumentativo de comunicación para personas con problemas fonéticos con el apoyo de un dispositivo móvil con Android," 2014.

[10] S. Aguirre, "SAAC-Droid, una herramienta de ayuda a la comunicación.," 2013.

[11] "Carrod electronica," 2014. [Online]. Available: https://www.carrod.mx/products/sensor-reflectivo-de-un-canal-cny70. [Last access:17 09 2017].

[12] "Wiki de robótica," [Online]. Available: http://wiki.robotica.webs.upv.es/wiki-de-robotica/sensores/sensores-proximidad/sensor-infrarrojos/. [Last access:17 09 2017].

[13] "Brico Geek," [Online]. Available: http://tienda.bricogeek.com/sensores/217-sensor-optico-qrd1114.html. [Last access:17 09 2017].

[14] "Cetronic componentes electrónicos," [Online]. Available: http://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=84-TSOP4838&cPath=1353. [Last access:17 09 2017].

[15] "Proyectos Electronics," [Online]. Available: https://catorcepi.wikispaces.com/file/view/3678453-SENSOR-INFRARROJO-Teoria-y-practica.pdf. [Last access:17 09 2017].

[16] "Nano ID," 2017. [Online]. Available: https://nanoid.mx/products/sensor-optico-reflectivo-infrarrojo-tcrt5000l-sin-pcb. [Last access:17 09 2017].

[17] USERS, "Módulo de captura de esquemáticos," de Electrónica práctica, Buenos Aires, redUSERS, 2011, p. 85.

[18] "Fritzing," [Online]. Available: http://fritzing.org/. [Last access:20 09 2017].

[19] "Wikipedia," 3 09 2017. [Online]. Available: https://es.wikipedia.org/wiki/Google_Glass. [Last access:20 09 2017].

[20] "EPSON," 2017. [Online]. Available: https://www.epson.es/products/see-through-mobile-viewer/moverio-bt-200. [Last access:20 09 2017].

[21] T. Hornyak, "Cnet," 10 12 2013. [Online]. Available: https://www.cnet.com/news/wink-glasses-fog-up-in-a-flash-forcing-you-to-blink/. [Last access:20 09 2017].

[22] "Conéctica," 17 Julio 2013. [Online]. Available: https://conectica.com/2013/07/17/glassup-la-alternativa-economica-para-las-google-glass/. [Last access:20 09 2017].

[23] "Vuzix," 2017. [Online]. Available: https://www.vuzix.com/Products/M100-Smart-Glasses. [Last access:20 09 2017].

[24] Fundation, "Bluetooth," 21 09 2017. [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works.

[25] M. Pinola, "Lifewire," 12 06 2017. [Online]. Available: https://www.lifewire.com/what-is-wi-fi-2377430.

[26] UcMerced, "UcMerced Library," 25 08 2017. [Online]. Available: http://library.ucmerced.edu/node/10249.

[27] RAE, "Real Academia Española," 20 08 2017. [Online]. Available: http://corpus.rae.es/lfrecuencias.html.

[28] Morse, "Codigo Morse," 15 08 2017. [Online]. Available: http://www.codigomorse.com/.