UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

CAPTURA DE MOVIMIENTOS CON KINECT PARA MANIPULAR UN ROBOT HUMANOIDE

TESIS

PARA OBTENER EL TÍTULO DE
INGENIERO EN ELECTRÓNICA

PRESENTA:
ÁNGEL AGUSTÍN GARIBO MORANTE

ASESOR:
MAESTRO EN INGENIERÍA
SALVADOR RAMÍREZ ZAVALA

MORELIA, MICHOACÁN FEBRERO DEL 2018

Dedicatoria

A mis seres queridos.

Dedico este trabajo a los que estuvieron en esta larga trayectoria, mi familia, mis hermanas que me apoyaron y confiaron en que podía hacerlo, a mis padres que siempre han estados y sin ellos nada de esto sería posible, A las personas de las que aprendí, a mis profesores y mis amigos a lo largo de mi vida que con sus consejos y conocimiento me acompañaron en el camino, unos estuvieron solo un tiempo, y otros aún lo hacen, con los que compartí buenos y malos momentos, pero de todos aprendí algo útil. A mi futura esposa y madre de mi hija, que siempre ha creído en mi aun cuando yo no lo he hecho, a mi hija, de un año, que con solo su sonrisa me hace sentir que todo es posible.

Agradecimientos

Agradezco a mi papa Agustín Garibo Naranjo por todo el apoyo y confianza brindada en mí, que sin su apoyo nada de esto sería posible y a mama María Morante Martínez que estuvo para escucharme a pesar de la distancia.

Agradezco a mi esposa Gabriela Estrada Carrillo por su paciencia, dedicación, apoyo. Por su amor hacia mí y nuestra hija, y por creer en nosotros.

Agradezco a mi buen y gran amigo Misael Mayo por acompañarme en este gran viaje de formación profesional y como persona, desde hace muchos años ya.

Y, por último, pero no menos importante a mí asesor el M.I. Salvador Ramírez Zavala que me brindó las herramientas y el apoyo para que este trabajo fuera posible, además de su grata amistad y orientación tanto técnica como personal.

Contenido

Lista de Figuras

Lista de tablas

Abstract

In the present work is present a human-machine interface (IHM) or Machine Men Interface (MMI) and a graphic interface in Matlab or Graphical User Interface (GUI). English) to manipulate a humanoid robot through.

For this MHI developed in the present work, the wireless control of a humanoid robot based on servomotors is made, the movement of its articulations is manipulated so that the detected human body postures are replicated as accurately as possible. The MICROSOFT Kinect sensors, which deliver data via USB to a depth and color computer, are processed by Matlab to obtain the coordinates of each limb of the body. With these coordinates and the programming of various algorithms that rely on basic trigonometry, the position and angles of the limbs are calculated to determine the position in which a person is standing in front of the Kinect, afterwards a specific command is sent via Bluetooth from the computer to the robot controller to move the servomotors and replicate the captured posture.

The articulations of the robot can be manipulated or deactivated by the GUI by a common user, so other Kinect parameters can also be configured such as height, resolution, frames captured before a detection, sensor type, among other functionalities.

Resumen

En el presente trabajo se presenta una interfaz hombre-maquina IHM o Machine Men Interface (MMI, por sus siglas en ingles) y una interfaz gráfica en Matlab o Graphical User Interface (GUI, por sus siglas en ingles). para manipular un robot humanoide mediante

Para esta IHM desarrollada en el presente trabajo, se hace el control inalámbrico de un Robot humanoide hecho a base de servomotores, se manipula el movimiento de sus extremidades para que se repliquen, lo más exacto posible, las posturas detectadas del cuerpo humano haciendo uso de los sensores del Kinect de MICROSOFT, que entrega datos vía USB a un ordenador de profundidad y color, dichos datos son procesados mediante Matlab para obtener posteriormente las coordenadas de cada extremidad del cuerpo. Con estas coordenadas y con la programación de diversos algoritmos que se apoyan de trigonometría básica se calcula la posición y ángulos de las extremidades para determinar la postura en la que se encuentre una persona parada frente al Kinect, posteriormente se envía un comando específico vía bluetooth desde el ordenador al controlador del Robot para que mueva los servomotores y se replique la postura capturada.

Las articulaciones del robot pueden ser manipuladas o desactivadas mediante la GUI por un usuario común, así como también pueden ser configurados otros parámetros de Kinect como la altura, la resolución, cuadros capturados antes de una detección, tipo de sensor, entre otras funcionalidades más.

Palabras clave.

Interfaz, Humano, Máquina, Arduino, Matlab, servomotor, Bluetooth. Kinect.

Capítulo 1 Introducción

1.1. Introducción a las Interfaz Humano Máquina.

En la actualidad y con el crecimiento tan acelerado de las nuevas tecnologías es más común encontrarnos con computadoras o robots que interactúan directamente con los seres humanos sin intervenir algún contacto físico directo. Nos encontramos con celulares o computadoras que pueden ser controladas con la voz o con la detección de los gestos de nuestra cara, hay videojuegos que sin necesidad de tener un mando físico en las manos se pueden realizar las mismas actividades que teniendo uno o incluso aún más, como dirigir un avatar que simula nuestros movimientos en tiempo real, permitiendo patear un balón imaginario, manejar un coche de carreras, tener una pelea de box con un adversario sin sufrir ningún daño físico, o inclusive volar, en un mundo virtual. También, en entornos industriales, médicos, o incluso militares existen robot que son dirigidos en tiempo real por los movimientos de un humano, permitiendo manipular objetos peligrosos o residuos tóxicos, realizar una cirugía con ultra precisión, desactivar una bomba, pilotar un avión sin que se arriesgue la vida de un ser humano. Es aquí que el concepto de Interfaz Hombre Maquina o IHM, cobra importancia.

Existen diferentes tipos de IHM, y son utilizadas en diferentes áreas como ya se mencionó, y cada vez serán más comunes este tipo de tecnologías en la vida cotidiana. En nuestra casa las simples tareas diarias serán cada vez más simples, llegará un momento en el que un obrero, un militar, un doctor, o un ingeniero trabaje en su casa desde un cuarto confortable y seguro realizando tareas que aún se requiera la intervención y razonamiento critico de un ser humano y no de una computadora, o mejor aún, una combinación de ambos, controlando mediante alguna IHM maquinaria pesada, brazos que realicen una cirugía, un robot humanoide que carguen objetos pesados, se metan a zonas inseguras como un edificio derrumbado, manipulen desechos radioactivos o industriales, todo ello sin que se arriesgue ninguna vida humana, aumentando la calidad de vida de las personas y contribuyendo a un desarrollo social y tecnológico.

Pero no hay que irnos muy lejos, todo esto ya se está viviendo hoy en día, aunque a una escala menor, pero cada vez son más adelantos que se tienen en estas áreas, cada vez son más sofisticadas estas tecnologías, haciendo uso de software o hardware de última tecnología, o como mero medio recreativo o educacional.

En el presente trabajo se muestra una sencilla pero eficaz IHM, en la cual se capturan datos por medio de un sensor de la posición de las articulaciones de un humano y se procesan por una computadora, para a partir de ello enviar una instrucción vía inalámbrica a una máquina.

A continuación, se presenta una definición de una IHM dada por la norma ISO 9241-110, además algunas de las áreas que emplean, los requisitos como el operar y observar, y la facilidad para poder usarse que deben tener las IHM, y después algunos antecedentes en proyectos actuales que emplean este tipo de interfaces.

1.1.1. Definición de una Interfaz humano máquina.

Una interfaz de usuario asistida por ordenador, actualmente una interfaz de uso, también conocida como interfaz hombre-máquina (IHM), forma parte del programa informático que se comunica con el usuario. En ISO 9241-110, el término interfaz de usuario se define como "todas las partes de un sistema interactivo (software o hardware) que proporcionan la información y el control necesarios para que el usuario lleve a cabo una tarea con el sistema interactivo" [1].

Figura 1 Ejemplo sencillo de una IHM.

La interfaz de usuario / interfaz hombre-máquina (IHM) es el punto de acción en que un hombre entra en contacto con una máquina. El caso más simple es el de un interruptor: No se trata de un humano ni de una "máquina" (la lámpara), si no una interfaz entre los dos, vease la Figura 1 donde se muestra dicho caso de IHM. Para que una interfaz hombre-máquina (HMI) sea útil y significativa para las personas, debe estar adaptada a sus requisitos y capacidades. Por ejemplo, programar un robot para que encienda la luz sería demasiado complicado y un interruptor en el techo no sería práctico para una luz en un sótano.

1.1.2. La Interfaz Hombre-Máquina (IHM) en diferentes áreas

Pensando sistemáticamente, la interfaz del usuario es una de las interfaces hombre-máquina (HMI): Hombre ↔ interfaz hombre - máquina ↔ máquina. Distintas ciencias se dedican a este tema, como las tecnologías de la información, la investigación cognitiva y la psicología. El conocimiento básico para un diseño de interfaz que le resulte fácil de utilizar al usuario se recoge en la disciplina científica de la ergonomía. Las áreas de actividad en sí son la ergonomía cognitiva, la ergonomía de sistemas y la ergonomía del software. [1].

1.1.3. Operar y observar; requisitos para una IHM

La interfaz del usuario, además de una "interfaz humano-máquina" (HMI), también se denomina "interfaz hombre-máquina" (MMI) y permite que el operador, en ciertas circunstancias, vaya más allá del manejo de la máquina y observe el estado del equipo e intervenga en el proceso. La información ("comentarios") se proporciona por medio de paneles de control con señales luminosas, campos de visualización o botones, o por medio de software que utiliza un sistema de visualización que se ejecuta en una terminal, por ejemplo. Con un interruptor de una lámpara, la información visual se proporciona a partir de la impresión de "luz" y la configuración del interruptor en "encendido" y "oscuridad" con el interruptor "apagado". En la cabina del conductor de un vehículo también se encuentran múltiples interfaces de usuario, desde los controles (pedales, volante, interruptores y palancas de los intermitentes, etc.) a través de reconocimientos visuales de la "máquina", el vehículo (pantalla de velocidad, marcha, canal de la radio, sistemas de navegación, etc.).

1.1.4. Facilidad de uso de la interfaz hombre-máquina

El éxito de un producto técnico depende de más factores aparte del precio, la fiabilidad y el ciclo de vida; también depende de factores como la capacidad de manipulación y la facilidad de uso para el usuario. Lo ideal sería que una interfaz hombre-máquina (HMI) se explicara por sí misma de forma intuitiva, sin necesidad de formación. El interruptor de la luz, a pesar de su popularidad y simplicidad, no es la interfaz de usuario ideal sino una solución intermedia entre dos objetivos contradictorios. En este caso, el interruptor debe estar situado cerca del dispositivo que se va a encender, por ejemplo, en la lámpara en sí (para que no tenga que buscarlo). O de lo contrario, debe estar cerca de la puerta (donde se encuentra normalmente) para que no tenga que buscarlo en la oscuridad. Otra interfaz popular, pero que tampoco resulta ideal, es la pantalla táctil: En este caso, puede iniciar un programa para el correo electrónico, por ejemplo, tocando la pantalla y luego recibe el correo. Sin embargo, cuando pulsa el icono, el dedo cubre el icono en sí. Esto generalmente no crea problemas, pero no es posible dibujar o escribir con precisión en la pantalla con los dedos.

1.2. Antecedentes y desarrollos HMI actuales.

Actualmente se están llevando novedosos estudios y muchos con gran avance en esta área, algunos en estudios médicos, o militares, o meramente científicos, algunos de los más interesantes desarrollados y que se han dado a conocer al público son los siguientes:

Goldfinger

Goldfinger es una innovadora interfaz hombre-máquina con forma de guante, alimentada energéticamente por los propios movimientos corporales y cuyos componentes electrónicos están integrados en el tejido, en vez de simplemente adheridos al guante.

Goldfinger ha sido diseñado y fabricado gracias a la colaboración entre el Instituto Politécnico de Turín en Italia y el Instituto Tecnológico de Massachusetts (MIT) en Estados Unidos, en un proyecto dirigido por Giorgio De Pasquale, del Departamento de Ingeniería Mecánica y Aeroespacial de la citada universidad italiana [2].

El principal punto de innovación en comparación con otros dispositivos es que Goldfinger posee la capacidad de autoabastecerse energéticamente. Genera energía eléctrica mediante el movimiento de los dedos del usuario, y ello le permite una autonomía de operación mucho mayor. Además, no necesita los cables eléctricos que sí se requerirían para el suministro eléctrico. El método de transmisión inalámbrica y la integración de muchos componentes de alta tecnología dentro del tejido del guante (de manera que el usuario no perciba su presencia) constituyen dos aspectos altamente innovadores de este prototipo. Entre estos componentes, por ejemplo, se encuentran cables conductores insertados en la textura del tejido, transductores piezoeléctricos con una alta flexibilidad e interruptores eléctricos hechos dentro del propio tejido, en vez de con los componentes electrónicos tradicionales [2]. La Figura 2 muestra al prototipo.

Figura 2 La nueva interfaz en forma de guante. Foto: Instituto Politécnico de Turín [2].

Exoesqueleto controlado con la mente para personas con ciertas discapacidades

Una nueva investigación realizada en Estados Unidos por científicos de la Universidad Rice, la Universidad de Houston y el TIRR Memorial Hermann también en Houston, tiene como meta ayudar a las víctimas de derrame cerebral, o con lesiones cerebrales similares, a recuperar esa capacidad motora en un grado lo mayor posible.

El equipo de José Luis Contreras Vidal, director del Laboratorio de Sistemas de Interfaz Cerebro-Máquina no Invasivos, dependiente de la Universidad de Houston, y Marcia O'Malley, directora del Laboratorio de Mecatrónica e Interfaces Hápticas de la Universidad Rice, es el primero en lograr reconstruir con éxito movimientos 3D propios del acto de caminar, y otros ejecutados por las manos, a partir de señales cerebrales registradas de modo no invasivo mediante electroencefalograma (EEG). La tecnología parece que permitirá a sus usuarios controlar brazos y piernas robóticas mediante sus pensamientos.

El equipo multidisciplinario espera desarrollar y validar una interfaz cerebro-máquina no invasiva que se espera que mejore de forma drástica la adaptación a sus prótesis de personas con extremidades amputadas, y también que revolucione la rehabilitación motora de personas con lesiones cerebrales. La nueva neuro tecnología interpretará ondas cerebrales que permitirán, por ejemplo, que un paciente que ha padecido un derrame cerebral controle a voluntad un exoesqueleto dentro del cual se coloca el brazo, desde las yemas de los dedos hasta el codo [3]. En la Figura 3 se muestran ensayos con la tecnología.

Investigadores crean un sistema impulsado por Kinect para pesar a los astronautas a la vista.

Con la vista puesta en los astronautas, los investigadores han creado un sistema que usa el Kinect de Microsoft para medir el peso de una persona simplemente mirándolos.

Figura 3 Ensayos de la tecnología. (Foto: Bruce French/TIRR Memorial Hermann) [3]

Los investigadores están utilizando Microsoft Kinect para calcular el peso de una persona con solo mirarlas, tal vez incluso en el espacio exterior. El científico informático Carmelo Velardo y un equipo del Centro de Robótica Espacial Humana del Instituto Italiano de Tecnología crearon el sistema, que utiliza la cámara de seguimiento del cuerpo de Kinect para generar un modelo 3D de un individuo dado. A continuación, se utiliza una base de datos de 28,000 personas para calcular el peso en función de las medidas físicas del sujeto, generando resultados que el equipo dice que son 97 por ciento precisos. Velardo considera que el sistema es ideal para su uso en viajes espaciales: sin gravedad, las escalas normales son inútiles, y los astronautas deben trepar sobre un taburete oscilante montado en un muelle para medir su masa

corporal. El sistema Kinect es más pequeño y más eficiente en energía que la solución actual, con la hipótesis de Velardo de que incluso podría construirse en las paredes de una estación espacial [4].

Roki, un exoesqueleto creado en México para ayudar a las personas a que puedan volver a caminar

Ahora tenemos otro proyecto muy interesante, es creado por Roki Robotics, empresa que se encuentra en Zapopan Jalisco y su creación lleva el nombre de Roki, un exoesqueleto robótico que se diseña a la medida, con la única finalidad de ayudar a la rehabilitación de las personas que deben mantener el equilibrio y apoyarse con muletas o andadera.

La idea original viene por parte de Norberto Velázquez y sus compañeros en 2006, donde creaban robots que pudieran jugar fútbol, pero fue hasta 2013, cuando conocieron a varias personas en sillas de ruedas y pensaron que la tecnología podría ser ideal para ayudarlos [5].

Figura 4. Prototipo Roki [5].

Roki es un traje robótico que cuenta con cuatro motores que se colocan en las rodillas y caderas. Las dos piernas se deben sujetar a las extensiones robóticas, mientras en la parte superior tiene una mochila en la que estará la batería que ofrece hasta cuatro horas de uso continuo. Entre los requisitos que se piden es pesar menos de 101 Kg, tener una estatura entre los 150 cm a 198 cm, ser menor de 60 años, tener una lesión medular inferior a T4, control de brazos y manos para el equilibrio y no contar con contracturas musculares [5].

Además de estos proyectos mencionados, existe muchos más desarrollos actuales muy interesantes, en los que se trabaja con Robots controlados por una HMI. Pero solo se mencionan algunos de ellos que tienen relación con el presente trabajo.

1.3. Objetivos

En esta sección se plantean los objetivos de este trabajo de tesis, los cuales se dividen en objetivo general y objetivos específicos.

1.3.1. Objetivo general

Complementar y reforzar la formación académica como Ingeniero en electrónica, sintetizando parte del conocimiento adquirido en un proyecto prototipo de utilidad a la comunidad estudiantil, o como parteaguas para el desarrollo de nuevas tecnologías que hagan uso de los elementos en el presente trabajo, en el cual se pretende desarrollar una Interfaz Humano Máquina.

1.3.2. Objetivos específicos

Realizar una IHM, la cual este constituida por como mínimo un elemento sensor, un ordenador que sea capaz de procesar los datos capturados y determinar una acción de control simple que sea enviada de forma inalámbrica entre el ordenador y un robot (Máquina) que manipule algún objeto u herramienta, o que simplemente se mueva esto sin que intervenga de forma totalmente directa o física el usuario (Humano), solo mediante el acoplamiento indirecto entre el usuario y el robot (Interfaz).

1.4. Justificación

En la actualidad existe una demanda cada vez mayor de este tipo de tecnologías, que ayuden a facilitar muchas tareas en diferentes áreas, como lo son en la medicina, la militar, en la construcción, en diferentes ingenierías, en lo domestico o inclusive en lo recreativo. Donde con el uso de una interfaz donde sin la intervención directa de un humano, pero si haciendo uso de su razonamiento critico ante situaciones que a una computadora o un robot autónomo les son difíciles o incluso imposible enfrentar. Ante este escenario surge la necesidad y la inquietud de realizar un prototipo funcional que sea controlado mediante este tipo de interfaz, donde se combinan las habilidades de los seres humanos, y el potencial de las maquinas.

Como parte de la formación universitaria en la rama de Ingeniería Electrónica en la Universidad Michoacana de San Nicolás De Hidalgo se adquieren diversos conocimientos técnicos sobre electrónica, entre los que destacan las subramas de Electrónica Analógica, Electrónica Digital, Instrumentación, Programación de Microcontroladores, Procesamiento Digital de señales, Control, en la cual se derivan muchas otras subramas de gran importancia como lo son el Control Analógico, Control Digital, Control en el Espacio de estados, Control con lógica Difusa, por mencionar algunas. Entonces se pretende conjuntar algunas de estas áreas del conocimiento, por lo que como parte de la formación y reforzamiento de los conocimientos adquiridos la Ingeniería en Electrónica se desarrolla este proyecto como tema de tesis, haciendo uso de tecnologías de bajo costo, pero funcionales que cumplan con el objetivo del trabajo.

Además, este tipo de herramientas con IHM, si bien es cada vez más comunes en la actualidad, en la Universidad Michoacana de San Nicolás De Hidalgo son muy escasas y existen pocos desarrollos de las mismas, por lo que se pretende este trabajo sea de utilidad para futuras generaciones de estudiantes de la Facultad de Ingeniería Eléctrica que quieran indagar más en esta área y les sirva de parteaguas, y orientación, ya sea para mejorar este trabajo o para desarrollar uno nuevo. Dejando el mismo a un nivel muy funcional.

Cabe mencionar que alguien quiere continuar con el presente trabajo el software desarrollado en este trabajo queda libre y se autoriza a quien así lo desee tomar fragmentos del código, o el código completo y hacer las modificaciones y mejoras que considere necesarias, solo con la consigna de que hagan mención de su autor, su servidor. Entonces si algún estudiante quisiera indagar más en este tema, le será muy asequible continuar con este trabajo, en comparativa con otras herramientas similares disponibles en el mercado que tienen un costo un poco elevado para un estudiante de recursos económicos limitados.

1.5. Metodología

La metodología de este trabajo se desarrolla bajo el siguiente esquema:
-Se especificaron los requerimientos que debe cumplir la persona que cumplirá el rol de Humano de la IHM, los cuales son que su interacción con la máquina sea del tipo indirecta, con el mimo contacto físico, y sea únicamente entrelazado con la maquina por medio de la interfaz a desarrollar, además de que cumpla con los rangos de distancia y condiciones del ambiente establecidos por el sensor.
-Se seleccionó como elemento sensor del interfaz humano máquina, al Kinect de la XBOX 360, consola de videojuegos de Microsoft, y se cumplieron requerimientos del mismo para usarlo conectado a la PC, los cuales fueron el uso de un adaptador del puerto que posee a puerto USB tipo b, y el software y controladores necesarios para hacer uso del mismo.
-Se planteó el equipo de cómputo a utilizar, verificando que cumple con los requisitos de procesamiento necesarios para realizar el proyecto, a partir de los requerimientos del software utilizado.
-Se hicieron primeras pruebas con el Kinect con captura de datos de los sensores de video.
- Se seleccionó como elemento actuador un prototipo de robot bípedo hecho por un exestudiante de la Facultad de Ingeniería Electrónica, el Ing. Salvador Álvarez Zalapa, que utiliza servomotores para realizar el movimiento de sus articulaciones.
- Se hizo como principal cambio al Robot mencionado anteriormente su tarjeta principal que implementaba una basada en el DSPIC30F4013 por la tarjeta del fabricante Arduino MEGA 2560, debido a sus ventajas respecto a la anterior y que satisface los requerimientos del proyecto actual.

-Se seleccionó la tecnología Bluetooth como vía de comunicación inalámbrica para transmitir comandos de la PC al robot, en específico se trabajó con el módulo HC-06, además se realizaron pruebas de comunicación entre la PC y la tarjeta principal por medio de Matlab.
- Se realizó una interfaz gráfica con la herramienta GUIDE de Matlab, para acercarse más al objetivo de capturar datos y realizar un control de una máquina por medio de la interfaz o acoplamiento entre hombre y máquina indirecto.
-Se realizó un algoritmo final para la tarjeta principal del robot y para la interfaz gráfica utilizada sistema final en el que el usuario con solo hacer unas

1.6. Contenido de la tesis

En el **capítulo 1** se presenta una introducción a las IHM, así como también el propósito de dicho proyecto analizando la problemática en cuanto a las problemáticas y demandas actuales que requieren el uso de estas tecnologías, un pequeña comparativa de costos del trabajo presentado y las opciones en el mercado. y como herramienta para otros estudiantes.

En el **capítulo 2** se presentan de manera general, las características de los elementos que conforman a la IHM, como lo son el ser humano y los requisitos que debe cumplir, además, el módulo Kinect como elemento sensor, las características de la PC (Personal Competer, por sus siglas en inglés) y el software de MATLAB 2015.a donde se desarrolló la interfaz gráfica y primeras pruebas, la comunicación inalámbrica entre la PC y el robot, el robot y sus módulos que lo conforma (tarjeta principal, tarjeta servocontroladora, y servomotores), para finalmente llegar a un diagrama de un sistema de captura de posiciones humanas reales con Kinect, procesamiento de las capturas obtenidas con una PC y la manipulación de robot humanoide vía inalámbrica.

En el **capítulo 3** se presenta la descripción de los elementos del sistema dando un enfoque su hardware y conexiones, y en el caso del humano a sus características, Para las diferentes conexiones se encuentra el módulo Kinect de Microsoft, el bluetooth de la PC, la conexión del Robot, que abarca desde la conexión de los servomotores, su tarjeta principal de Arduino MEGA 2560, la conexión el módulo de bluetooth HC-06 del robot, la conexión de la tarjeta servo controladora Mini Maestro 24 CH de Pololo con los servomotores MG995 y la conexión a su vez, con la tarjeta principal, y las características de la fuente de alimentación.

El **capítulo 4** Primeramente, para la captura de posiciones humanas se menciona los requisitos de software necesarios y controladores para poder usar el Kinect con la PC, además se explica un poco sobre el desarrollo del software realizado mediante un ejemplo con los comandos de la biblioteca de Kinect para Matlab.

También, se explica la manera de hacer las configuraciones previas de la tarjeta servocontroladora Mini maestro 24 canales con una GUI del fabricante, así como el software desarrollado para que la tarjeta MEGA 2560 trabaje con la tarjeta servo controladora.

Además, se explica una manera sencilla, mediante un Script en Matlab, para establecer la detección de postura utilizando el Kinect, interpretación de los ángulos detectados, para finalmente enviar comandos para la posición de cada servomotor vía bluetooth a la tarjeta principal y que esta lo interprete y se comunique vía puerto serie con la tarjeta servo controladora que a su vez determinará el valor del ancho del pulso para cada canal correspondiente a cada servomotor.

Finalmente se explica el desarrollo de una GUI, mediante la herramienta GUIDE de Matlab, que facilita al usuario configuraciones previas del Kinect y mayor interacción con el sistema final o IHM.

En el **capítulo 5** se presentan los resultados obtenidos, algunos ejemplos de la detección de diferentes posturas en algunas personas, así como la respuesta del robot humanoide ante dichas posturas. Se presentan diferentes configuraciones en la GUI para los sensores del Kinect y la comparación script versus GUI.

Por último, en el **capítulo 6** se presentan las conclusiones generales de esta tesis, desde del hardware y software desarrollado, los resultados obtenidos de las pruebas con diferentes personas, los problemas y limitaciones en la realización de este trabajo, así como los posibles trabajos y mejoras futuras a realizar en el prototipo.

Capítulo 2.
Uso del Kinect para una Interfaz Humano Máquina (IHM)

2.1. Introducción a la interfaz IHM con uso del Kinect

Las Interfaces humano Máquina tienen mucha aplicación en la actualidad y son de gran utilidad, desde sector industrial, en el hogar o incluso de forma recreativa, existen muchos tipos de IHM, desde las más sencillas como el interruptor para prender una lámpara, hasta más complejas donde un humano interactúa, ya sea por medios físicos como paneles de control con botones, una interfaz gráfica asistida por ordenador, o inclusive sin ningún contacto físico y por medio de sensores sin contacto físico o inalámbricos (de presencia, proximidad, ultrasónicos, micrófonos, radares, cámaras de video, cámaras térmicas o incluso una combinación de varios de estos elementos) con mecanismos más complejos como robots, exoesqueletos, vehículos no tripulados, o inclusive procesos industriales etc. En la Figura 5 se muestra un ejemplo general de IHM

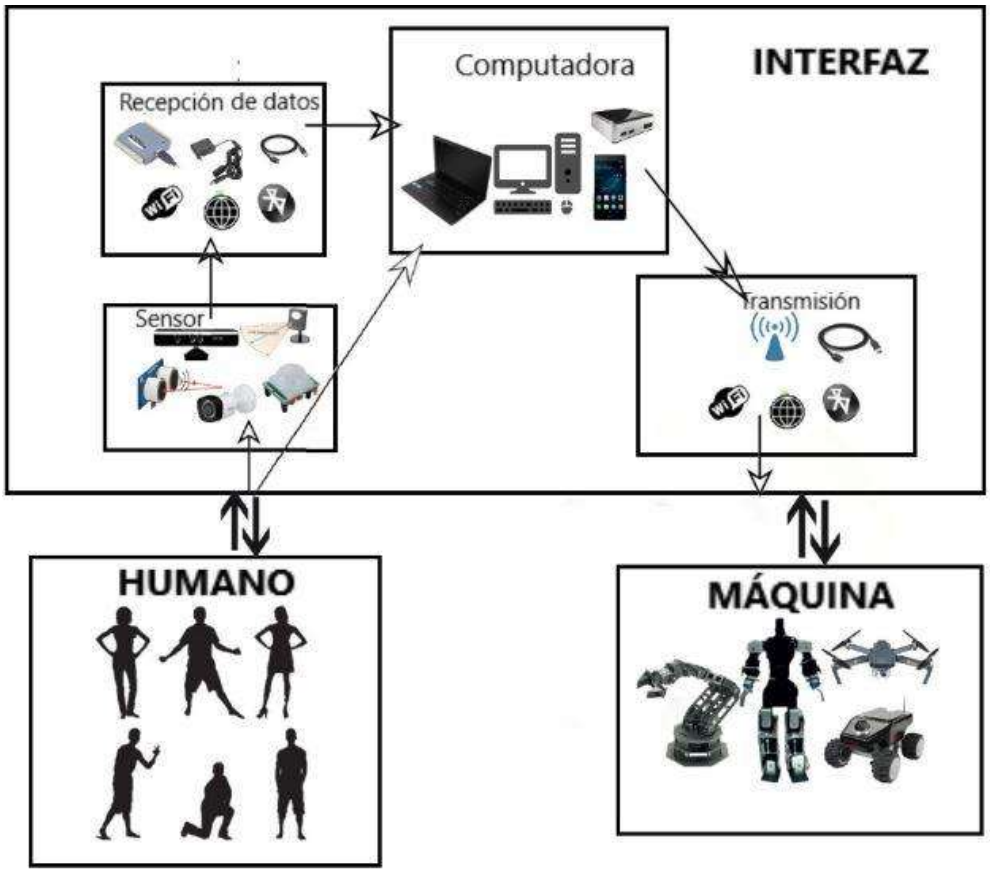

Figura 5 Interfaz Humano Máquina.

El Kinect es un módulo que emplea varios sensores, como lo son dos micrófonos, un sensor de color, una cámara infrarroja, y un sensor de profundidad, y que ha sido implementado en diferentes sistemas IHM que con uso de una computadora o PC (Personal Competer, por sus siglas en inglés) se hace el procesamiento de las capturas de información obtenida por los sensores (audio o imágenes) y a partir de esto enviar una instrucción para manipular una máquina como puede ser un robot, un vehículo no tripulado ya sea terrestre o incluso aéreo, dicha transmisión puede ser alámbrica (por medio de cables) o inclusive por medio de una tecnología inalámbrica como lo es el wifi, bluetooth, internet, radiofrecuencia, et. En la Figura 5 se muestran varios elementos que pudieran conformar una IHM. Algunos de los cuales son explicados a continuación, dando más atención a los elementos implementados la IHM de este trabajo.

2.2. Humano

El humano es uno de los elementos principales de un sistema IHM, y aunque existen sistemas o interfaces completamente automatizadas y muy complejos que ya no requieren la presencia o interacción con un ser humano, estos no se estudian en este trabajo.

El ser humano tiene mucha importancia en procesos o sistemas (ya sean procesos industriales, sistemas domóticos, sistemas médicos automatizados por mencionar solo algunos) donde a una computadora le sea imposible elegir que acción debe realiza a partir de estímulos exteriores, ya sea porque el algoritmo o programa que se deba encargar de elegir dicha acción no esté diseñado para funcionar con la información adquirida, o la información sea nula o este "contaminada" de alguna forma, o si no fueron consideradas otras variables, para lo cual la computadora puede o no elegir una acción a ejecutarse, o que la acción elegida no sea optima o la ideal para el proceso o peor aún sea una acción totalmente equivocada y ello pueda tener consecuencias como un mal producto, elección de la temperatura no deseada para una habitación, o inclusive una mala cirugía que puede tener daños irreversibles en un paciente, hablando de un sistema medico automatizado).

Por lo anterior el humano tiene gran importancia en muchos sistemas, y es donde cobran mucha importancia las IHM, donde se aprovechan las habilidades humanas adquiridas de forma empírica, en conjunto con el potencial que puede otorgar una máquina, como procesado de información que no es posible en un humano, o uso de herramientas para fin específico donde el humano por si solo no pudiera realizar, como levantar cargas pesadas, o peligrosas para su salud, realizar cirugías de extra precisión donde se aprovechen los conocimientos de un doctor especialista con el pulso nanométrico de un brazo robótico.

En este sistema IHM realizado la persona o humano no requiere ser especialista en un área, pues el sistema solo es un prototipo interactivo, tampoco se requiere que tenga una complexión física especifica.

2.3. La interfaz

La Interfaz será la encargada de realizar de algún modo la unión o acoplamiento entre el Humano y la Máquina, remontándonos al ejemplo más sencillo antes mencionado en el cual un interruptor tenía el rol de Interfaz en un sistema donde una persona deseaba ya sea apagar o prender una lámpara.

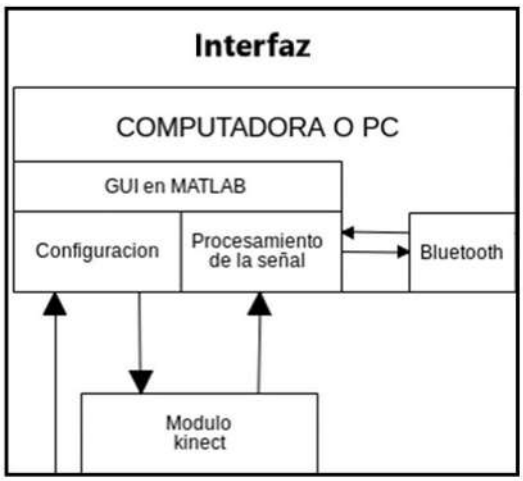

Figura 6 Esquema de la interfaz desarrollada.

La interfaz puede llegar ser tan simple o muy compleja como se mencionó anteriormente, en este trabajo se implementó un sistema con un esquema similar al mostrado en la Figura 6, donde se requiere un Humano, del cual ya se habló *grosso modo* anteriormente y una Máquina, ahora se explica más a detalle la parte de la Interfaz de este sistema.

Los elementos que la pueden conformar pueden ser diferente, pero principalmente se requiere una computadora donde el humano interactuara directamente con ella y esta a su vez con la Máquina, o para una interfaz un poco más compleja (la implementada en este trabajo) donde el humano además de interactuar con la Maquina a través de una computadora de forma física donde el **sensor** seria el teclado y el mouse, lo hace también de forma indirecta o indirecta por medio de un **sensor** inalámbrico, del cual se explica un poco más a continuación.

2.3.1. El sensor de la Interfaz

El sensor como ya se dijo con contacto con o sin contacto físico (de presencia, proximidad, ultrasónicos, micrófonos, radares, cámaras de video, cámaras térmicas o incluso una combinación de varios de estos elementos), a continuación, se describen las características de solo algunos de ellos, como lo es el sensor PIR, y el ultrasónico, y las características de algunos módulos comerciales que implementan estos sensores, hasta llegar al módulo utilizado en este trabajo.

Sensor PIR

Entre los sensores de presencia se encuentra el módulo HC-SR501 al cual también se le conoce como sensor PIR, "infrarrojo pasivo", "piroeléctrico" o "movimiento IR Los sensores PIR permiten “sentir” el movimiento, casi siempre es usado para detectar si un humano se movió dentro o fuera del rango de los sensores. Son pequeños, económicos, de bajo consumo, fáciles de usar y no se desgastan. Por esa razón, se encuentran comúnmente en electrodomésticos y artilugios utilizados en hogares o negocios.

Los PIR están hechos básicamente de un sensor piroeléctrico (que se puede ver a continuación como el metal redondo con un cristal rectangular en el centro), que puede detectar niveles de radiación infrarroja. Todo emite un poco de radiación de bajo nivel, y cuanto más caliente es algo, más radiación se emite. El sensor en un detector de movimiento en realidad está dividido en dos mitades. La razón de esto es que estamos buscando detectar el movimiento (cambio) no los niveles medios de IR. Las dos mitades están cableadas para que se cancelen mutuamente. Si una mitad ve más o menos radiación IR que la otra, la salida oscilará alta o baja [6]. En la Figura 7 se observa como es este módulo.

Figura 7 Módulo PIR HC-SR501 [6].

Algunas características básicas son:

- **Salida:** Pulso digital alto (3V) cuando se dispara (movimiento detectado) digital bajo cuando está inactivo (no se detecta movimiento). Las longitudes de pulso están determinadas por resistencias y condensadores en la PCB y difieren de un sensor a otro.
- **Rango de sensibilidad:** Hasta 6 metros.
- **Fuente de alimentación:** Voltaje de entrada de 5V-12V para la mayoría de los módulos (tienen un regulador de 3.3V)
- **Costo:** Entre $20 y $180 pesos M.N.

Como se observa, este elemento como sensor, cumple con algunos de los requerimientos para el trabajo propuesto, como lo son bajo costo, voltaje de alimentación y de lógica típicos en la electrónica digital, pero tiene como principal inconveniente la limitante de que o bien detecta o no detecta presencia (1 o 0 lógicos), por lo que las acciones de control con los datos obtenidos con este sensor serían muy limitadas a partir de esta información.

Sensor ultrasónico

Los sensores de ultrasonido son muy útiles para medir distancias y detectar obstáculos. El funcionamiento es simple, envía una señal ultrasónica inaudible y nos entrega el tiempo que demoro en ir y venir hasta el obstáculo más cercano que detecto.

El módulo HC-SR04 emplea un sensor ultrasónico que está conformados por dos cilindros puestos uno al lado del otro, uno de ellos es quien emite la señal ultrasónica, mientras que el otro es quien la recibe, es un sistema muy simple pero no por eso deja de ser efectivo. En la Figura 8 Se muestra como es físicamente este módulo. El sensor hc-sr04 en particular tiene una sensibilidad muy buena del orden de los 3mm, y un alcance deteniendo en cuenta que la

Figura 8 Sensor ultrasónico HC-SR04

mayoría de las aplicaciones donde este sensor es utilizado es para medir o detectar obstáculos o distancias mayores a varios centímetros, y su sensibilidad es muy buena.

Características del sensor ultrasónico hc-sr04 [7]

- **Alimentación:** 5 volts.
- **Interfaz de cuatro hilos**: (vcc, trigger, echo, GND).
- **Rango de medición:** 2 cm a 400cm.
- **Corriente de alimentación:** 1.5mA.
- **Frecuencia de pulso:** 40Khz.
- **Apertura del pulso ultrasónico:** 15°.
- **Señal de disparo:** 10us.
- **Costo:** Entre $20 y $150 pesos M.N.

Como se observa este sensor tiene unas características bastante aceptables, como su gran alcance, lógica de voltajes, bajo costo, más interacción con el usuario, esto es, si el usuario está más o menos alejado del sensor el tiempo registrado de la señal enviada es mayor o menor, por

lo que ya con este sensor se podría hacer una interacción mayor con el usuario y el actuador, como que entre a más cercano este el usuario más rápido sea el movimiento del actuador. O se mueva de un lado a otro si el usuario se acerca o aleja. Con todas estas prestaciones es una buena opción como sensor, pero tiene la desventaja de que se necesita de una tarjeta adicional para la transmisión de datos desde el sensor hacia la computadora, o se podría trabajar directamente con una tarjeta que adquiera y procese los datos del sensor y se comunique directamente con la máquina, pero el objetivo de este trabajo es utilizar además una computadora para el procesado de la información.

El módulo Kinect®, como elemento sensor

Kinect® de Microsoft® es un complemento de la consola de videojuegos XBOX® que literalmente, nos convierte en "Controles humanos" y somos nosotros con nuestras manos, cabeza, pies, cara y voz, quienes controlamos cada aspecto del juego o aplicación que estemos corriendo en la consola. Funciona por medio de cámaras, sensores y micrófonos de alta tecnología que detectan cualquier movimiento que hagamos con el cuerpo y lo interpreta en acciones que harán los personajes del juego que esté en acción.

Algunos de los componentes son los que se muestran la |Figura 10:

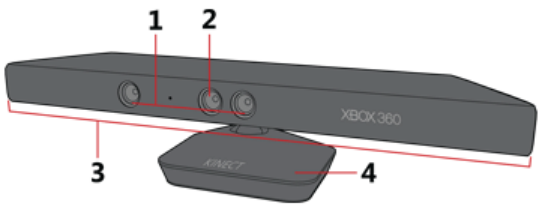

Figura 9 Kinect de Xbox de Microsoft [8].

1 sensores de profundidad 3-D

Los sensores tridimensionales hacen un seguimiento de cuerpo dentro del área de juego.

2 cámara RGB

Una cámara RGB (rojo, verde, azul) ayuda a identificarlo y capta imágenes y videos del juego.

3 varios micrófonos

Se usa un conjunto de micrófonos en el borde frontal inferior del sensor Kinect para reconocimiento de voz y charla.

4 inclinación motorizada

Un impulso mecánico en la base del sensor Kinect inclina de manera automática el sensor hacia arriba o abajo según sea necesario. [8]

Y algunas de sus características son:

Campo de visión

Campo de visión horizontal: 57 grados
Campo de visión vertical: 43 grados
rango de inclinación física: ± 27 grados
Rango de profundidad del sensor: 1,2 – 3,5 metros

Data Streams (Flujo de datos)

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

Sistema de Seguimiento

Rastrea hasta 6 personas, incluyendo 2 jugadores activos
Rastrea 20 articulaciones por jugador activo
Capacidad para mapear jugadores activos en tiempo real

Sistema de audio

Chat en vivo y voz dentro del juego.
Sistema de cancelación de eco que aumenta la entrada de voz
Reconocimiento de voz múltiple

Como se observa las características del módulo Kinect son muchas y bastante buenas como lo son sus múltiples sensores calidad de los mismos y rangos de trabajo. Entre los que destacan sus cámaras y el sensor de profundidad 3d, el cual tiene un rango aceptable (1.2-3.5 metros), por lo que se puede posicionar el usuario a diferentes distancias del sensor. El flujo de datos es bastante aceptable (30 fotogramas por segundo), para esta aplicación esta velocidad de captura es buena, pues en un segundo pueden ser capturados múltiples cuadros o imágenes para su análisis y procesamiento. Además, cuenta con la prestación de que se puede controlar la inclinación del mismo por software, ya que tiene un motor integrado para realizar dicha acción, cuenta con sensores audio, que en este trabajo no interesa su uso, pero para futuros trabajos y mejora de la IHM podrían ser utilizados.

Es necesario mencionar que la principal desventaja de usar el Kinect puede decirse que es su costo muy por encima respecto a los sensores otros sensores mencionados, aunque sus prestaciones son muy superiores a los módulos mencionados, pero se puede adquirir uno usado y que sea funcional alrededor de $400 pesos M.N. Además, otra desventaja es que no se puede conectar directamente a la PC y necesario adquirir un adaptador especial para poderlo conectar, como el que se muestra en la |Figura 10, cuyo precio ronda los $200 pesos M.N.

|Figura 10 Adaptador de Kinect® para PC.

2.3.2. La computadora

La computadora es un elemento clave en este sistema IHM desarrollado, pues permite el procesamiento de la información adquirida por el Kinect su manipulación ya sea mediante un programa no visible por el usuario, o mejor aún mediante una interfaz de usuario grafica o GUI (Graphical User Interface, por sus siglas en ingles), en el uso e implementación de la GUI radica el uso de una PC, ya que como antes se mencionó, podría hacerse el procesamiento de la información y comunicación con la Maquina directamente por medio un microcontrolador o tarjeta de desarrollo, que es más barata pero sus prestaciones generalmente son menores, que si bien se puede adaptar una pantalla exterior para hacer una GUI, por lo general dicha GUI no sería muy avanzada o se carecería de las prestaciones que ofrece una PC promedio, como mayor velocidad de procesamiento, implementación directa de otros dispositivos y tecnologías, como teclado, pantalla incorporada, adaptador inalámbrico integrado, sistema operativo, además de los diferentes entornos de desarrollo que pueden ser utilizados, de los cuales se habla a continuación.

Entornos de desarrollo que permiten el uso de Kinect

El módulo de Kinect y las herramientas de desarrollador son compatible con el sistema operativo Windows y Linux, este proyecto se desarrolla en el primero, debido a que se tiene mayor soporte y compatibilidad debido a que Microsoft es desarrollador de Kinect y de Windows, además que el equipo de cómputo utilizado corre bajo este sistema operativo.

Existen varias alternativas entornos de desarrollo para Windows para trabajar con Kinect, entre las que destacan Visual Studio, Code Composser Studio de Texas Instruments®, Processing y Arduino IDE de *open source*, y Matlab que además de ser capaces de programar convencionalmente bajo los mismos, también permiten el desarrollo de GUIs. En el presente trabajo se utiliza el entorno de Arduino IDE y Matlab. debido a sus prestaciones como lo son su alto poder de procesamiento, amplio soporte, y versión gratuita para estudiantes, a continuación, se describen algunas de las características de estos entornos

Matlab 2015a

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux [9].

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multi dominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas o bibliotecas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets) [9].

Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

Los requerimientos de hardware necesarios, de Matlab en su versión 2015a, que se explicará más adelante, por ahora solo los requerimientos son mencionados en la Tabla 1 [10].

Tabla 1 Requerimientos mínimos de hardware de MATLAB.

<table>
<tr><th colspan="5">Familias y productos de MATLAB y Simulink de 32 y 64 Bit</th></tr>
<tr><th>Sistemas operativos</th><th>Procesadores</th><th>Espacio de disco</th><th>RAM</th><th>Gráficos</th></tr>
<tr><td>Windows 10</td><td rowspan="6">Cualquier procesador x86 Intel o AMD con soporte de instrucciones SSE2</td><td rowspan="6">1 GB para solo MATLAB 3-4 GB para una instalación común</td><td rowspan="6">2 GB</td><td rowspan="6">No se requiere una tarjeta gráfica especifica. Se recomienda una tarjeta gráfica con aceleración por hardware con soporte OpenGL 3.3. con 1 GB de memoria GPU</td></tr>
<tr><td>Windows 8.1</td></tr>
<tr><td>Windows 8</td></tr>
<tr><td>Windows 7 Service pack 1</td></tr>
<tr><td>Windows Vista Service Pack 2</td></tr>
</table>

2.4. Transmisión de datos entre la PC y la Máquina

Para la comunicación entre el ordenador y la Máquina donde se transmitirán los comandos específicos a partir del procesamiento de la información proporcionada por el usuario, o

directamente configuraciones desde la computadora, por medio de una GUI. Para ello hay múltiples opciones, desde simplemente utilizar cables o adaptadores necesarios para interconectar la computadora y la máquina, o utilizar tecnologías inalámbricas como pueden ser el wifi local, o wifi con acceso a internet, el Bluetooth, comunicación por radio frecuencia o por microondas

En este sistema se trabaja la se trabaja con la tecnología bluetooth, algunas de las razones son por su amplio soporte y compatibilidad con diferentes dispositivos, un alcance bastante bueno, un bajo costo y curiosidad por implementar dicha tecnología. A continuación, se presenta una introducción a la tecnología Bluetooth, y al módulo que implementa la Máquina de este sistema, cabe mencionar en este punto, que si la comunicación que se utilizará en la Interfaz, en específico entre la computadora y la Máquina, ambos deben hacer uso de esta tecnología.

2.4.1. Introducción a la tecnología bluetooth

En primer lugar, como curiosidad decir que el sobrenombre Bluetooth de la tecnología que expondremos en nuestro trabajo es un nombre tomado de un Rey Danés del siglo X, llamado Harald Blåtand (Bluetooth), que fue famoso por sus habilidades comunicativas, y por haber logrado el comienzo de la cristianización en su cerrada sociedad Vikinga [11].

Una de las características más importantes de la especificación Bluetooth es que debería permitir dispositivos de muchos fabricantes para trabajar juntos. Por ese motivo, Bluetooth no solo define un sistema de radio, sino también una pila de software que permite a las aplicaciones encontrar otros dispositivos Bluetooth en el área, descubrir qué servicios pueden ofrecer y usar esos servicios. Bluetooth permite que hasta ocho dispositivos se conecten entre sí en un grupo llamado *piconet* [12].

Los dispositivos Bluetooth funcionan a 2,4 GHz, en la banda ISM, sin licencia y disponible a nivel mundial. Ese es el ancho de banda reservado para uso general por aplicaciones industriales, científicas y médicas en todo el mundo.Dado que esta banda de radio puede ser utilizada libremente por cualquier transmisor de radio siempre que cumpla con las regulaciones, la intensidad y la naturaleza de la interferencia no pueden predecirse. Por lo tanto, la inmunidad a la interferencia es un tema muy importante para Bluetooth. En general, la inmunidad a la interferencia se puede obtener mediante la supresión o eliminación de interferencias. La supresión puede obtenerse mediante codificación o propagación de secuencia directa, pero el rango dinámico de señales interferentes en redes ad hoc puede ser enorme, por lo que las ganancias de codificación y procesamiento prácticamente no suelen ser adecuadas [12].

2.4.2. Bluetooth en la computadora

Existen diferentes adaptadores de Bluetooth para computadora disponibles en el mercado, inclusive algunas ya traen dispositivos Bluetooth integrados, tal es el caso de la computadora utilizada en este trabajo, las configuraciones necesarias se mencionan el capítulo 4, "Desarrollo del software"

2.4.3. Bluetooth en la Máquina

Entre los dispositivos disponibles en el mercado con tecnología Bluetooth se encuentra este módulo HC-06, versión mejorada de su hermano HC-05. Este módulo se muestra en la Figura 11. Y tiene características muy interesantes entre las que destacan las siguientes:

Figura 11 Módulo Bluetooth HC-06.

Características Módulo bluetooth HC-06 [13]:

- **Modo esclavo y modo maestro**.
- Puede configurarse mediante comandos AT (Deben escribirse en mayúscula)
- **Frecuencia:** 2.4 GHz, banda ISM.
- **Antena** PCB incorporada.
- **Potencia de emisión:** ≤ 6 dBm, Clase 2
- **Alcance:** 5 m a 10 m
- **Sensibilidad:** ≤ -80 dBm a 0.1% VER
- **Velocidad:** Asincrónica: 2 Mbps (máx.) /160 kbps, sincrónica: 1 Mbps/1 Mbps
- **Seguridad:** Autenticación y encriptación (Password por defecto: 1234)
- **Baudrate ajustable:** 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- Módulo montado en tarjeta con regulador de voltaje y 4 pines suministrando acceso a VCC, GND, TXD, y RXD
- **Voltaje de alimentación:** 3.3VCD – 6VCD.
- **Voltaje de operación:** 3.3VCD
- **Costo:** Va de los $50 a los $300 pesos M.N.

El módulo de bluetooth HC-06 ofrece una buena relación de precio y características, ya que es un módulo Maestro-Esclavo, quiere decir que además de recibir conexiones desde una PC o Tablet, también es capaz de generar conexiones hacia otros dispositivos bluetooth. Esto nos permite, por ejemplo, conectar dos módulos de bluetooth y formar una conexión punto a punto para transmitir datos entre dos microcontroladores o dispositivos. Además de tener un consumo de corriente bajo, su principal limitante es su voltaje de operación de 3.3 VCD.

2.5. La Máquina

La máquina del sistema como se mostró en la Figura 5, puede ser desde un robot, un brazo robótico, un vehículo móvil, terrestre como un coche, o aéreo como un avión o un *dron*., inclusive un exoesqueleto, o el hardware de un proceso industrial cualquiera, o en un caso más sencillo la misma computadora.

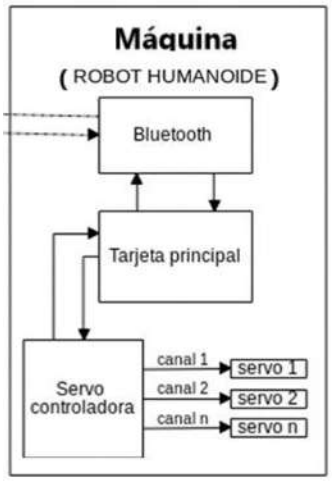

Figura 12 Diagrama de la Máquina

En la Figura 12 se muestra un diagrama de la Máquina, con un enfoque a los elementos utilizados en este trabajo, en el cual se utilizó un robot humanoide el cual se escribe a continuación.

2.5.1. Robot humanoide

Se utiliza como máquina de la IHM un robot humanoide desarrollado por el ex estudiante de esta universidad y facultad, Salvador Álvarez Zalapa, el cual fue presentado como trabajo de tesis para obtener el grado de Ingeniero en Electrónica con título “CONSTRUCCIÓN DE UN ROBOT BÍPEDO HUMANOIDE CON CONTROL DE EQUILIBRIO POR LÓGICA DIFUSA”, al cual le aplicó un control lógico difuso, que consistía en censar la inclinación actual

del robot mediante un módulo acelerómetro y giroscopio y mediante el control difuso determinar el valor de posición de los servomotores de los pies para mantener el equilibrio del Robot.

Entre los principales cambios y mejoras destacan el reordenamiento del cableado de los servomotores, , el cambio de los servomotores dañados, construcción de una base para mejorar la facilidad en las pruebas, y la sustitución de la tarjeta principal, pues para el control con lógico difuso, y comunicación con la tarjeta servo controladora usaba el DSPIC30F4013, y el presente trabajo se usa tarjeta de desarrollo Arduino mega 2560, de la cual se habla más a detalle en la sección 0.

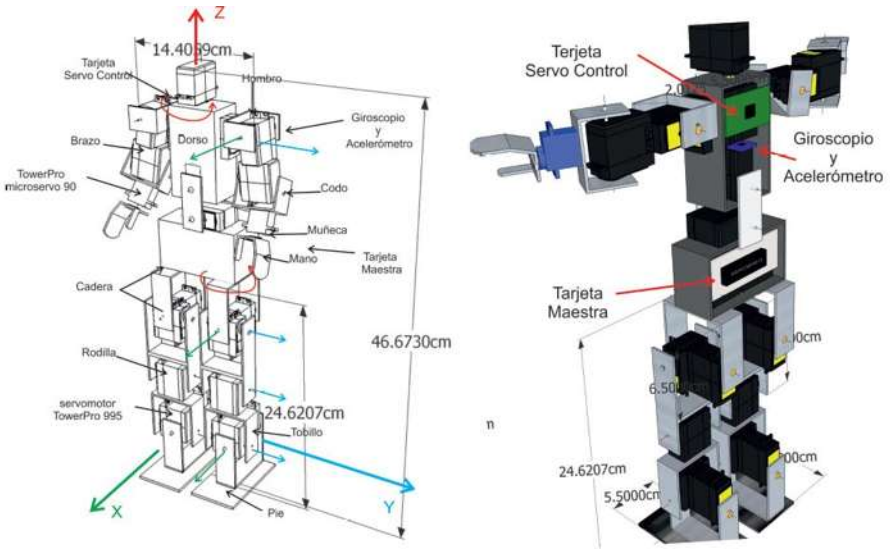

Figura 13 Diseño del robot bípedo y su distribución espacial [14].

El diseño del robot bípedo cuenta con un total de 23 GDL (Grados De Libertad) como se muestra en la por pierna, 2 más en la cadera, 4 más por cada brazo y uno más en la cabeza. En la Figura 13 se muestra la distribución espacial del modelo y ubicación del hardware principal, así como los ejes sobre los actúan cada una de las articulaciones de este robot. La estructura general del RBH está formada por servomotores que fueron modificados para formar parte de esta, aluminio y plástico [14].

Los elementos que conforman al robot humanoide son el módulo Bluetooth, los servomotores MG995, y una tarjeta servo controladora Mini Maestro 24 Canales, el módulo bluetooth HC-06, del cual ya se escribió anteriormente, y que si bien forma parte del apartado de transmisión de datos entra la computadora y la Máquina, es su vez parte física de la Máquina, por lo cual ya no se escribe de este módulo en esta sección. Los otros elementos se describen a continuación.

2.5.2. La tarjeta Principal

La tarjeta Principal es el nombre que se le da a la tarjeta que controlara a los otros elementos de la Máquina (el robot humanoide), los cuales son una tarjeta servo controladora, los servomotores y un dispositivo Bluetooth.
Existen muchas tarjetas de desarrollo de diferentes fabricantes como lo son de Texas Instruments que tiene las LaunchPad de la familia MSP serie 430, o de Microchip que implementan PIC de la familia F45xx, también se encuentran las mini computadoras como del fabricante Raspberry, ó tarjetas con dispositivos AVR, entre los que se encuentre el fabricante Arduino que ha tenido un crecimiento exponencial en los últimos años, por ser de software de uso libre, y además un soporte muy grande alrededor del mundo. De estas dos últimos fabricantes se describe a continuación la RASPBERRY PI 3. Con la que se realizaron varias pruebas como parte de este trabajo, y la Arduino MEGA 2560 que fue la que finalmente se implementó

Mini computadora Raspberry Pi 3
Inicialmente se trabajó con la tarjeta RASPBERRY PI 3®, la cual es una computadora de placa única con LAN inalámbrica y conectividad Bluetooth. de bajo costo desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. Soporta múltiples Sistemas Operativos (SO), pero el que se encuentra con mayor soporte es el SO Raspbian®. En la Figura 14 se muestra esta tarjeta Raspberry.

Figura 14 Raspberry PI 3 Modelo B.

Algunas de sus características son [15]**:**

- CPU Quad Core 1.2GHz Broadcom BCM2837 de 64 bits
- 1GB de RAM
- BCM43438 LAN inalámbrica y Bluetooth de baja energía (BLE) a bordo
- GPIO extendido de 40 pines
- 4 puertos USB 2
- Salida estéreo de 4 polos y puerto de video compuesto
- HDMI de tamaño completo
- Puerto de cámara CSI para conectar una cámara Raspberry Pi
- Puerto de pantalla DSI para conectar una pantalla táctil Raspberry Pi

- Puerto Micro SD para cargar su sistema operativo y almacenar datos
- Fuente de alimentación micro USB conmutada actualizada de hasta 2.5 A

Se desarrollo un programa en el IDE de Python 3.4, el cual realizaba una secuencia de encendido y apagado de un LED conectado a un GPIP (puerto que puede ser utilizado como entrada o salida digital), dicho programa se configuro para que se auto ejecutara al arranque de la Raspberry.

Posterior a ello se hicieron pruebas con el puerto serie para mandar comandos a la tarjeta controladora de los servomotores del Robot, pero se tuvo el inconveniente de que este se encontraba mapeado físicamente al dispositivo Bluetooth integrado, por lo que para poder usarlo es necesario dejar inhabilitado el Bluetooth, con lo que se podría utilizar únicamente su conectividad WIFI, por lo cual se decide utilizar otra tarjeta como Principal la cual se describe a continuación.

Módulo microcontrolador Arduino MEGA 2560®

la tarjeta Arduino Mega 2560 la cual es desarrollada bajo el esquema de *open-source* y construida con un microcontrolador modelo Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales. Esta tarjeta es programada en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede comunicarse a un PC a través del puerto serial (conversión con USB) utilizando lenguajes como Flash, Processing, MaxMSP, etc [16].

Figura 15 Arduino MEGA 2560 R y logo.

El Arduino Mega tiene 54 pines de entradas/salidas digitales (14 de las cuales pueden ser utilizadas como salidas PWM), 16 entradas análogas, 4 UARTs (Universal, Asynchronous Received-Transmitter, por sus siglas en ingles), cristal oscilador de 16MHz, conexión USB, Jack de alimentación, conector ICSP y botón de reset. Arduino Mega incorpora todo lo necesario para que el microcontrolador trabaje; simplemente conéctalo a tu PC por medio de un cable USB o con una fuente de alimentación externa (9 hasta 12VCD) [16].

Además, utiliza un microcontrolador ATMega8U2 en vez del circuito integrado FTDI. Esto permite mayores velocidades de transmisión por su puerto USB y no requiere drivers para Linux o MAC, pero si para Windows, además cuenta con la capacidad de ser reconocido por el PC como un teclado, mouse, joystick, etc. [16]. En la Figura 15 se muestra como luce dicha tarjeta.

Algunas de sus características son:

- **Microcontrolador** ATmega2560.
- **Voltaje de entrada** de 5-12V.
- 54 pines digitales de Entrada/Salida (14 de ellos son salidas PWM).
- 16 entradas análogas.
- 256KB de memoria flash.
- **Velocidad del reloj** de 16Mhz.
- **Costo:** De entre $250 y $800 pesos M.N.

Entorno de desarrollo para la tarjeta principal

Para el desarrollo del código de la tarjeta principal, la Arduino MEGA 2560, se utiliza la *IDE* oficial del fabricante Arduino versión 1.8.3, funciona bajo la filosofía de software de código abierto Arduino, su programación hace que sea más fácil escribir código y subirlo a la tarjeta. Se ejecuta en Windows, Mac OS X, y Linux. El entorno está escrito en Java y está basado en Processing y otro software de código abierto. Este software se puede usar con cualquier placa Arduino. En la Figura 15 se muestra el logotipo de esta IDE.

2.5.3. El servomotor

El Robot con el que se trabajó cuenta con actuadores de tipo servomotor de alta velocidad MG995. Puede usar cualquier código de servo, hardware o biblioteca para controlarlos. El Actuador MG995 incluye armas y hardware para comenzar.

Figura 16 Servomotor MG995

Características del servomotor MG995:

- Peso: 55 g

- Dimensión: 40.7 x 19.7 x 42.9 mm aprox.
- Par de parada: 8.5kgf · cm (4.8 V), 10kgf · cm (6 V)
- Velocidad de operación: 0.2s/60 (4.8 V), 0.16s/60 (6 V)
- Voltaje de funcionamiento: 4,8 V a 7,2 V
- Ancho de banda muerta: 5µs
- Diseño de cojinete de bolas doble estable y a prueba de golpes
- Rango de temperatura: 0 ºC-55ºC 31150-MP
- Ángulo de rotación: 120 grados. (+ - 60 desde el centro)
- Metal Gears para una vida más larga [17]

Como se observa tiene un par o torque que va desde los 8.5kgf x cm, operando con 4.8 volts, hasta los 10kgf x cm, con 6 volts de alimentación, la programación es compatible con múltiples controladores, para ella es necesario enviarle una señal PWM (Pulse-Width Modulation, por sus siglas en inglés), dependiendo del ancho del pulso, siempre y cuando este dentro del rango, se tiene rotación de un motor, que después de reducir la velocidad para aumentar el par mediante un sistema de engranaje, se logra una rotación de su engrane de salida que es 0 a 120 grados aprox. Según el fabricante. En laFigura 16

2.5.4. La tarjeta servo controladora

Como parte complementaria al hardware del Robot, y para evitar "consumir" muchos pines y recursos del microcontrolador principal, el Robot implementa una tarjeta controladora llamada "Mini Maestro 24 Channel" del fabricante Pololu®, en la cual son conectados todos los servomotores, y mediante un protocolo y comunicación específica, e los cuales se escribe más adelante, y conjunto con un microcontrolador principal es posible controlar de forma más eficiente hasta 24 servomotores, enviando la trama adecuada.

Los Mini Maestros son tarjetas servo controladoras altamente versátiles (y compactos) y placas de E / S de propósito general. Admiten tres métodos de control: USB para la conexión directa a una computadora, serie TTL para usar con sistemas integrados y secuencias de comandos internas para aplicaciones autocontenidos y sin controlador de host.

Los canales se pueden configurar como salidas de servo para su uso con servos de control de radio (RC) o controles electrónicos de velocidad (ESC), como salidas digitales, o como entradas analógicas / digitales. Los Mini Maestros también cuentan con frecuencias de pulso configurables de 1 a 333 Hz y pueden generar una amplia gama de pulsos, lo que permite una capacidad de respuesta y rango máximos de los servos modernos. Las unidades se pueden conectar en cadena con controladores de servo y motor Pololu adicionales en una única línea serie. En la Figura 24 se puede observar dicha tarjeta.

Principales características:

- **Tres métodos de control:** USB, serie TTL (5V) y scripting interno
- **Resolución de ancho de pulso de** salida de 0.25µs (corresponde a aproximadamente 0.025 para un servo típico, que está más allá de lo que el servo podría resolver)
- **Frecuencia de pulso configurable**: de 1 a 333 Hz
- **Amplio rango de pulsos:** de 64 a 4080 µs
- Velocidad individual y control de aceleración para cada canal
- Las funciones de canal alternativo permiten que los canales se usen como:
  Salidas digitales de uso general (0 o 5 V)
  Entradas analógicas o digitales (los canales 0 - 11 pueden ser entradas analógicas, los canales 12+ pueden ser entradas digitales)
- **Completa guía del usuario** [18].

2.6. Conclusiones del capítulo

A manera de conclusión, en la
Figura 17, se muestra un diagrama de cómo está forma general, la IHM desarrollada.

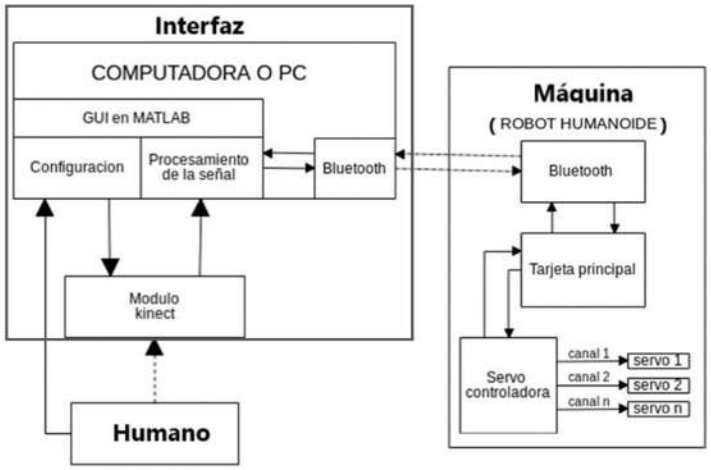

Figura 17 Diagrama general de la IHM.

Las características de los elementos que conforman una IHM, como lo son el ser humano y los requisitos que debe cumplir, además, el módulo Kinect como elemento sensor, las características de la PC (Personal Competer, por sus siglas en inglés) y el software de MATLAB 2015.a donde se desarrolló la interfaz gráfica y primeras pruebas, la comunicación inalámbrica entre la PC y el robot, el robot y sus módulos que lo conforma (tarjeta principal, tarjeta servocontroladora, y servomotores), para finalmente llegar a un diagrama de un sistema de captura de posiciones humanas reales con Kinect, procesamiento de las capturas obtenidas con una PC y la manipulación de robot humanoide vía inalámbrica.

Capítulo 3. Desarrollo de Hardware.

3.1. Introducción al hardware

En esta sección se describe como se hicieron las conexiones físicas (cableado y componentes adicionales) de los elementos que componen el sistema, se advierte sobre consideraciones especiales o recomendaciones a seguir para finalmente llegar a un producto operativo al final de este capítulo.

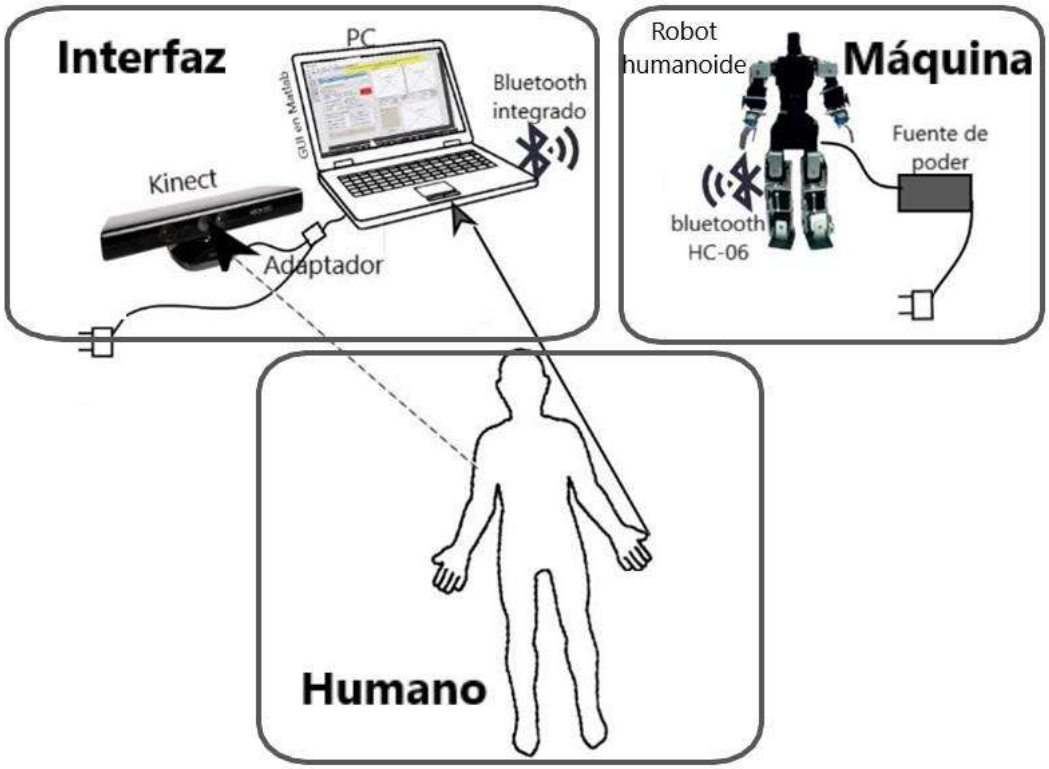

Figura 18 Diagrama del hardware de la IHM.

La Figura 18 se muestra un esquema general de la conexión entre los elementos que conforman el sistema; Kinect V1 para Xbox 360 (el sensor), el adaptador, PC (el procesamiento de las señales adquiridas), Bluetooth HC-06 (la comunicación inalámbrica), y en la Maquina se encuentre con el tarjeta maestra o principal Arduino MEGA 2560, que indica instrucciones a la tarjeta servo controladora Mini Maestro 24 la cual mueve a los servomotores MG995.

3.2. El Humano

La persona que cumpla el rol de “Humano” de la IHM, debe cumplir con algunas características básicas, como lo son estar vestido con ropa no muy grueso (como cobertores o chamarras de

invierno), para un rastreo completo debe de estar colocado parado frente al Kinect en un rango de entre 1.2 a 3.5 metros de distancia y dentro de su límite de visión campo de visión horizontal de 57 grados y vertical de 43 grados. Además de esto debe tener el conocimiento mínimo necesario para utilizar la interfaz para poder hacer la selección del sensor de video del Kinect, algunas configuraciones iniciales (opcionales) iniciar la detección de la postura

3.3. El hardware de la Interfaz

Los elementos que conforman la Máquina como ya se mencionó, son el módulo Bluetooth HC-06, la tarjeta maestra o principal (MEGA 2560), y la tarjeta servo controladora (Mini Maestro 24) que a su vez mueve o controla a los actuadores (servomotores MG995). A continuación, se muestra información de estos elementos referente al hardware, como lo son la conexión y algunas advertencias y recomendaciones.

3.3.1. El Kinect y el adaptador

El Kinect tiene la función de ser el elemento sensor del sistema, que como ya se menciono es un módulo conformado por varios sensores (sensor de profundidad, sensor de color, sensores de sonido). A continuación, en la Figura 19, se muestra un diagrama de conexiones que incluye al Kinect y al adaptador para PC.

Figura 19 Conexión del Kinect con la PC

Primeramente, se comienza con el elemento sensor, el módulo Kinect®, el cual es elemento más sencillo de conectar, pues solo basta adquirir un adaptador de puerto del Kinect que usa Xbox, a puerto USB, las conexiones se muestran en la Figura 19..

Los pasos a seguir para la conexión del Kinect son:

- 1 conseguir un Kinect
- 2 conseguir el adaptador para conectar con PC

- 3 conectar el Kinect al puerto hembra del adaptador.
- 4 conectar el cable macho USB a la computadora y su fuente de voltaje al tomacorriente.

Como se observa los pasos a seguir son muy sencillos, al conectar correctamente el led del Kinect debe prender, y se debe mostrar un cuadro de dialogo en la PC indicando que se están instalando algunos drivers, los otros drivers y herramientas necesarias se abordaran en el capítulo siguiente.

3.3.2. Conexiones de la Computadora o PC

La PC puede ser cualquiera que cumpla con los requerimientos de software que se mostraron en la sección Entornos de desarrollo que permiten el uso de Kinect, esta puede ser una computadora de escritorio, una laptop, o una minicomputadora. En este trabajo se utilizó una laptop que cumpliera con los requerimientos de la sección antes mencionada

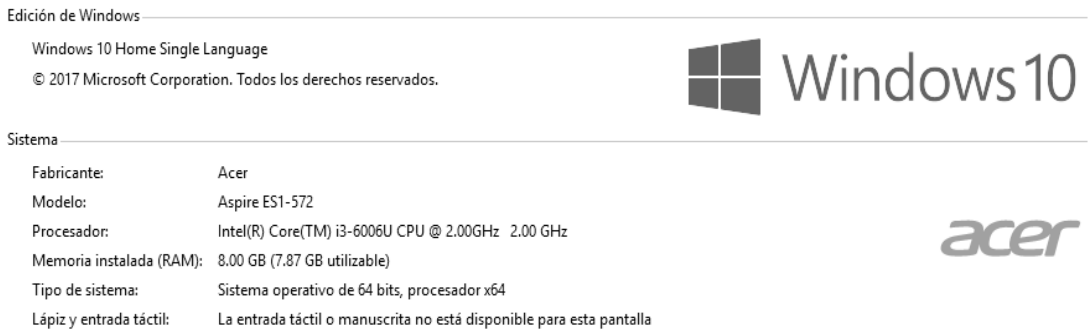

Figura 20 Características de la PC.

Ahora bien, analizando la PC utilizada se tienen algunas de las características de software y hardware que aparecen en la Figura 20.

Las conexiones necesarias son únicamente con el Módulo Kinect, y conexión al tomacorriente, ya que el dispositivo Bluetooth para la comunicación entre la Máquina lo tiene integrado, solo se tienen que hacer las configuraciones que se explican en el Capítulo 4 Desarrollo del Software.

3.4. El hardware de la Máquina

En esta sección se describen los componentes físicos de la Máquina (el robot humanoide) la cual está compuesta por un módulo Bluetooth, una tarjeta principal, los servomotores y una

tarjeta servo controladora. Se mencionan las interconexiones que se hacen en los diferentes elementos y algunas recomendaciones y advertencias a considerar.

3.4.1. El Bluetooth HC-06

El módulo HC-06 tiene 4 pines de conexión disponibles, uno TX para la transmisión de datos por protocolo UART, el pin RX para la recepción de datos con el mismo protocolo, y los pines de alimentación VCD y GND.

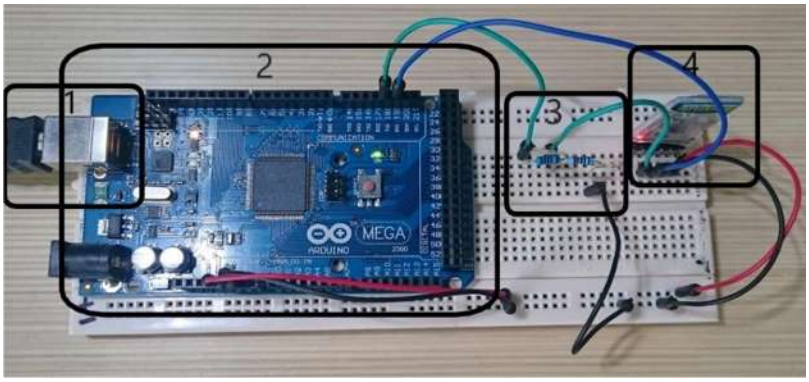

Figura 21 Conexión de prueba del módulo HC-06 con la tarjeta MEGA2650.

Existe una versión anterior al módulo HC-06 utilizado, el HC-05, que entre las principales diferencias se encuentra que el segundo puede ser configurado como modo maestro y modo esclavo, a diferencia del H-06, que solo permite ser usado como maestro. Estos modos se describen a continuación:

Figura 22 Módulos Bluetooth HC-05 y HC-06.

Otra de las grandes diferencias es que posee el HC-06 posee un pin menos que su antecesor, lo que facilita su programación, ya que este pin, llamado "*key*" o "*enable*", era utilizado como pin de activación para entrar al modo de comandos AT, que en el capítulo siguiente se habla más a fondo de este modo de comandos. Pero resultaba tedioso y un tanto complicada la rutina de inicialización del módulo, por lo que en el HC-06 no presenta este problema, pero presenta en la placa un orificio por si se desea soldar un terminal para entrar al

modo de programación AT por defecto, el cual está configurado a una velocidad de transmisión de 37400 *baudios*. Y es útil ingresar a este modo cuando no se conoce la velocidad actual a la que está configurado el módulo. En Figura 22 se muestra como lucen ambos módulos para evitar confusiones.

Para hacer las primeras pruebas de este módulo se utilizó uno de los tres puertos Serie TTL (0-5V) disponibles de la tarjeta Arduino MEGA, en la Figura 21, se muestra la conexión realizada para hacer las pruebas.

En la Figura 21 se observa un cable USB tipo B, el cual es conectado a la PC, por lo que no es necesario una fuente de poder externa, sino que la PC cumple dicha función.

ADVERTENCIA. En la Figura 21.2 se observa que fue utilizado el puerto serie 1, o Serial1 en Arduino. Donde el cable TX de la MEGA se conectó a un divisor de voltaje, para dicho divisor no es muy importante el valor de las resistencias, pero si la proporción entre ellas. Ahora se explicará un poco más esto. Como los voltajes de los pines de salida son lógicos de 0 a 5 volts, y de los puertos serie son TTL en la tarjeta MEGA, y se especifica por el fabricante que el módulo HC-06 trabaja con una lógica de voltajes de 0-3.3V, no así su alimentación que puede ser de los 3.3 a los 6 volts. Entonces para establecer una buena conexión y evitar daño en el módulo por sobre voltaje es necesario dicho divisor de voltaje. La relación entre las resistencias debe ser de 2/3, es decir, conectar el pin TX de la MEGA a una resistencia R1 igual a 1/3 de la resistencia total, en serie con una resistencia R2 igual a 2/3 de la resistencia total total la cual a su vez debe de estar conectada en el otro extremo a tierra. En medio de ambas resistencias se tiene que tomar una derivación conectada a RX del módulo HC-06. De este modo se tiene que cuando el pulso TX este en alto (5 volts), se tendría un voltaje de 3.3 volts a RX del HC-06. Para entender mejor esto observar la Figura 23, y en Figura 21.3 aparece dicho divisor.

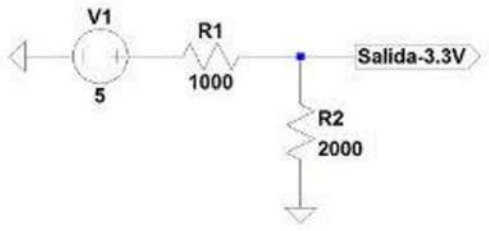

Figura 23 Divisor de voltaje de 5 a 3.3 Volts.

Por último, en la Figura 21 se muestra la conexión de la señal de TX proveniente de la tarjeta MEGA y después de pasar por el divisor de voltaje, con el pin RX del HC-06, y el pin TX del HC-06 directamente al pin RX de la MEGA, aquí no es obligatorio un circuito elevador

de voltaje de 3.3 a 5volts ya que la tarjeta MEGA interpreta ese nivel de voltaje como un 1 lógico. Pero si se requiriera un circuito elevador, se recomienda usar una compuerta OR, interconectando sus dos pines de entrada, así, a su salida se tendrían 5 volts cuando reciba 3.3v, ya que también interpreta ese nivel de voltaje como un 1 lógico. Los últimos cables son los de la alimentación del HC-06, uno conectado a pin de 5 volts de la MEGA y el otro a tierra.

3.4.2. La tarjeta servo controladora y los servomotores.

La tarjeta servo controladora (Mini Maestro 24 canales) se selecciona en base a velocidad de transmisión de datos y capacidad de la misma; ya que permite controlar hasta 24 servomotores, cuenta con entradas y salidas analógicas y digitales, salidas de PWM, señales de control para control electrónico de velocidad, también dispone de comunicación USB para conexión directa al PC, comunicación TTL serie para usar con sistemas embebidos y funcionamiento autónomo mediante scripts internos.

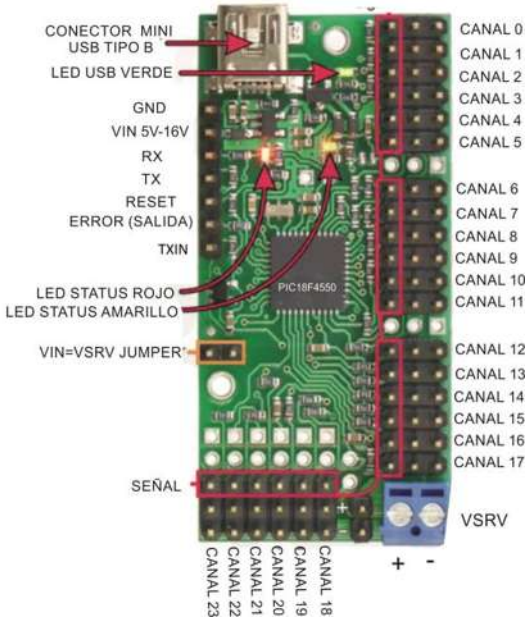

Figura 24 Mini Maestro de 24 canales

La alimentación de la tarjeta servo controladora puede hacerse desde el conector USB o una fuente externa con voltaje de 5-16V conectada a las terminales VIN (+) y GND (-). Se puede tener ambas fuentes conectadas al mismo tiempo, en cuyo caso el microcontrolador se alimenta de la fuente externa. La conexión de alimentación de los servos (VSRV) puede ser independiente de la alimentación general de la tarjeta removiendo el jumper VIN=VSR, de cualquier otro modo el voltaje colocado ya sea en la terminal VIN o VSRV serán iguales; para este proyecto se utilizan dos fuentes una para la tarjeta servo controladora y demás hardware y otra

exclusivamente para los servomotores; debido a que la alimentación del servo pasa directamente sin tener que regularse.

Del hardware general de esta tarjeta se tiene que cuando el pin RST se encuentra en bajo (0 Volts) se genera un reseteo general en la tarjeta Internamente este pin está en alto (5 Volts) con una resistencia de *pull-up* colocada en la tarjeta por lo que puede estar desconectado este pin. El pin ERROR es una salida, que se pone en alto (5 Volts) cuando led de estatus rojo se enciende, indicando que existe algún error de comunicación. El pin TXIN es una línea de entrada que hace que sea fácil encadenar varias tarjetas servocontroladoras. Cualquier byte recibido por esta línea se almacena en el búfer de transmisión y posteriormente se transmite por la línea TX.

Dentro de las opciones de comunicación de la tarjeta se encuentran USB Dual Port, USB encadenado y UART (esta última empleada en este proyecto).

Comunicación UART: la línea RX se usa como receptora no invertida TTL (0-5 V) serie, los comandos recibidos por este puerto son únicamente comandos preestablecidos por el fabricante, estos comandos pueden variar dependiendo del protocolo interno usado, estos protocolos internos empleados han sido nombrados por el fabricante de esta tarjeta como: Protocolo Compacto, Protocolo Pololu, Protocolo Mini SSC [19].

Indicadores LED La tarjeta servo control tiene tres indicadores LED:
Esta tarjeta cuenta con tres leds incorporados, ver Figura 24, el **led verde** indica el estado del dispositivo USB. Cuando no está conectado a USB estará apagado. Al conectarlo parpadeará lentamente. El parpadeo continúa mientras se instala el controlador en la PC, después de que haya instalado correctamente el controlador el led verde permanece encendido y parpadeará solo si hay actividad en la línea USB [20].

El **led rojo** de error indica que hay errores en la transmisión. Se encenderá cuando ocurra un error y se apagará al limpiar el bit de error. El **led amarillo** indica el control del estado de la trasmisión. Cuando el Maestro está en modo auto baudio (por defecto 9600 baudios) y aún no se ha detectado la velocidad, el LED amarillo parpadeará lentamente. Durante este tiempo la tarjeta no transmite pulsos a los servos. Una vez que está listo para controlar los servos, el led amarillo se enciende brevemente de forma periódica. La frecuencia de los destellos es proporcional al período de servo (la cantidad de tiempo entre los pulsos en un solo canal), con un período de 20 ms el parpadeo se produce aproximadamente una vez por segundo. El número de destellos indicara un estado diferente un destello indica que ningún servo está habilitado (no se envían pulsos) y todos los canales de salida están bajos, mientras que un doble destello indica que al menos uno de los servos está habilitado o uno de los canales de salida está siendo impulsado en alto.

Cuando se resetea por alguna razón que no sea la alimentación, el led rojo y/o el amarillo parpadean cuatro veces para indicar la condición de reset.

Condición de los leds:
-Amarillo apagado, rojo parpadeando: Un reset de alimentación. Ocurre cuando los 5V de corriente del Maestro descienden por debajo de los 3.0 V, seguramente por la descarga de las baterías o por una alimentación inadecuada.
-Amarillo pardeando, rojo apagado: Reset del Maestro por bajo voltaje en la línea RST.
-Amarillo y rojo parpadean a la vez: Un fallo de *firrmware* hace que el "*watchdog*" salte. Esto También ocurre inmediatamente después de una actualización de firmware, como parte normal del proceso de actualización

Comunicación
Esta tarjeta dispone de tres modos de comunicación. Tiene las líneas TX y RX que permiten enviar y recibir datos TTL (0 Volts - 5 Volts), segundo dispone de dos puertos virtuales serie por donde el ordenador se conecta a través del USB. Uno de estos puertos se llama Command Port (Puerto virtual que puede ser visualizado desde el software Maestro Control Center el cual se interconecta con la tarjeta) y el otro TTL Port.

USB Dual Port: En este modo el Command Port se usa para enviar Comandos a la tarjeta y recibir respuestas de él. La velocidad en baudios es irrelevante. El TTL Port se usa para enviar bytes por la línea TX y recibirlos por la RX. La velocidad de comunicación que se establezca en el programa al configurar el puerto TTL determina la velocidad en baudios utilizada para recibir y enviar bytes RX y TX. Esto permite monitorear por USB la posición de los servos y el poder utilizar simultáneamente las líneas de TX y RX como puerto serie de propósito general para que pueda comunicarse con otros tipos de dispositivos serie TTL [19].

USB encadenado: En este modo el Comando Port se usa en ambos casos para transmitir bytes por la línea TX y enviar comandos a la tarjeta y esta responde a los comandos enviados por el Command Port no por la línea TX. Los bytes recibidos por la línea RX son reenviados al Command Port, pero no se interpretan como bytes de comandos. La velocidad de transmisión en la terminal al abrir el Command Port determina la velocidad de recepción y envío de bytes por RX y TX. El puerto TTL no se usa. Este modo permite que un único puerto COM del ordenador pueda controlar múltiples dispositivos compatibles con el protocolo utilizado [19].

UART: En este modo, las líneas TX y RX se usan para enviar comandos a la tarjeta y recibir respuestas del mismo. Cada byte recibido por RX se envía al Command Port, pero los bytes enviados desde el Command Port son ignorados. La velocidad en baudios se detecta

automáticamente por la tarjeta cuando se recibe un byte 0xAA por RX, o puede ser establecida por el usuario bits/segundo (bps). Este modo permite que sea usando un microcontrolador u otro dispositivo TTL serie dicho método es el implementado en este trabajo [19].

TTL Serie: La línea RX puede recibir bytes cuando está conectada a una lógica no invertida (de 0 a 5V - "TTL"). Los bytes recibidos en RX pueden ser comandos o una secuencia arbitraria de datos que pasa al ordenador mediante el puerto USB, dependiendo del modo serie configurado. La tarjeta proporciona los niveles lógicos (0 a 5V) a la salida de su línea de transmisión TX. Los bytes enviados por la tarjeta en TX pueden ser respuestas a comandos que soliciten información o una secuencia arbitraria de datos que está recibiendo desde un ordenador por el puerto USB, dependiendo del modo ajustado. Si no se requiere recibir bytes TTL de la tarjeta se puede dejar desconectada la línea TX. La transmisión es asíncrona, lo que significa que la recepción y el envío llevan bits a velocidad independiente. El formato de datos es de 8 bits, un bit de paro y sin paridad es decir 8-N-1. En la Figura 25se muestra el asincronismo, no invertido de un byte serie TTL: Una línea serie TTL invertida tiene como estado predeterminado (no activo) en alto [19].

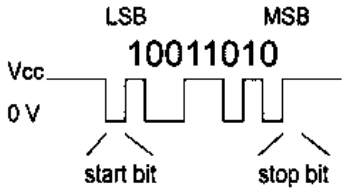

Figura 25 Señal no invertida de un byte serie TTL.

La forma de conectar es muy sencilla, en la Figura 26 se muestran el nombre correspondiente a cada pin y un ejemplo de conexión en el canal 23 correspondiente a la parte baja de la cintura según la
Tabla 2.

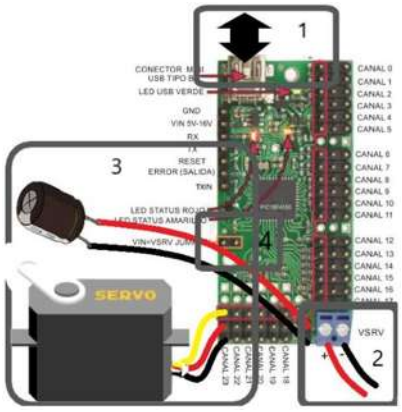

Figura 26 Tarjeta servo control y otros elementos.

En la Figura 26.1 se indica la conexión necesaria del cable USB hacia la computadora, para poder empezar a configurar y trabajar con la tarjeta. Posterior a esto se deben descargar los controladores necesarios dependiendo del S.O. (Sistema Operativo) con el que se esté trabajando, esto se explica de forma más detallada en el **Capítulo 3.**

En la Figura 26.2 se indica dónde debe ser conectada la fuente para los servomotores y en la Figura 26.4 se muestran dos pines, los cuales al conectar un puente entre ambos, se indica a la tarjeta que su lógica se alimentara con la misma fuente que los servomotores, lo cual no es recomendable al menos que se tenga certeza de que la fuente está regulada y no habrá caídas de voltaje al estar operando los servomotores. Por eso en este trabajo se alimenta directamente de la placa de Arduino, que, aunque esta se alimente de la misma fuente que los servomotores, proporciona una salida regulada.

En la Figura 26.3 se muestra cómo debe ir conectado un servomotor, la línea amarilla corresponde a la señal PWM, la línea negra a tierra, y la línea roja corresponde al voltaje de alimentación de los servomotores, también se muestran los pineas correspondientes a otros canales para los otros servomotores, numerados del 0 al 23. A continuación, en la
Tabla 2 se muestra el canal asignado a cada servomotor y el nombre asignado al mismo según la posición que ocupa en el Robot.

Tabla 2 Número de canal y nombre de cada servomotor según posición.

Número de canal	Nombre
0	No existe
1	Tobillo izq. Lados
2	Rodilla Izquierda
3	Tobillo izq. frente/atrás
4	Pierna izq. Lados
5	Pierna izq. Frente/atrás
6	Cabeza
7	Pierna derecha. Frente/atrás
8	Pierna derecha lados
9	Rodilla derecha
10	Tobillo derecha, frente/atrás
11	Tobillo derecho lados
12	Codo derecho
13	Mano derecha
14	Cintura alta
15	Rotación hombro izquierda

16	Apertura hombro izquierda
17	Rotación hombro derecha
18	Rotación pierna derecha
19	Rotación pierna izquierda
20	Apertura hombro derecho
21	Codo izquierdo
22	Muñeca izquierda
23	Cintura baja

RECOMENDACIÓN: Conectar un capacitor de al menos 1,000uF (mil microfaradios) o más, para evitar dañar la fuente de poder, ya que cuando trabajen todos los servomotores o varios a la vez, los picos de consumo de corriente pueden ser muy alto, por lo que pueden causar una caída considerable en el voltaje, y debido a que todos los elementos están conectados a una misma fuente esto podría causar que la tarjeta principal o la servo controladora se apaguen por falta de energía y causar fallas en el funcionamiento o incluso daños en los componentes.

3.4.3. La fuente de poder

Como fuente de poder se reutilizó una procedente de un CPU de escritorio, en la Figura 27.1 se muestra la fuente de poder utilizada, y en la Figura 27.2 las características de voltaje y corriente. Donde se puede observar que para la salida de 5 volts es capaz de proporcionar 18 amperios. Por lo que entrega una potencia de 90 watts.

Figura 27 Fuente de poder

En la Figura 28, cuadro 1, se muestran los pines correspondientes a las salidas de voltaje de 5 voltios utilizada. Y en el cuadro 2 se muestra la conexión necesaria para que la fuente

prenda. Esta alimentación se utiliza para los servos de la tarjeta controladora y para la placa de principal, misma alimenta a la servocontroladora y al módulo bluetooth.

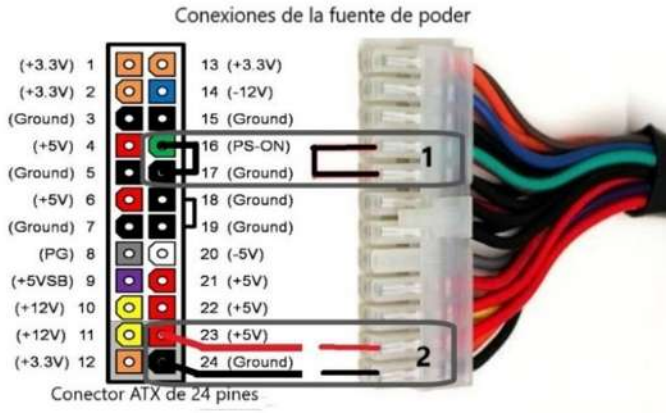

Figura 28 Conexiones de la fuente de poder

Según la página oficial del fabricante [21] el consumo de corriente es el siguiente:

Consumo de corriente en ralentí 10mA (Amperios x10-3)
Consumo de corriente de operación sin carga 100mA
Máxima corriente de drenaje 1000mA aprox.

ADVERTENCIA. Para un consumo de hasta 23 servos el consumo de corriente sin carga podría llegar a ser de hasta 2.3 A, con una fuente de voltaje de 5 voltios, la potencia demandada a la fuente es de 10.15 watts. Aunque este consumo aumenta, con la carga no a todas las articulaciones o servos levantan mucho peso, como es el caso de los brazos, cabeza, e incluso las piernas al estar el robot sobre la base que se le fabrico, aunque este consumo evidentemente se elevara al poner al robot manteniéndose parado sobre sus propias piernas. Se deben de tener en cuenta el consumo que podría llegar a tener de hasta 23 amperios. Aunque como ya se mencionó, esto es poco probable que ocurra.

Según las especiaciones de la fuente, proporciona hasta 18 Amperes en la salida de 5 volts, y hasta 90 watts, se puede decir que la fuente cumple con los requerimientos del sistema, en el que los servomotores son los elementos de mayo consumo, pero que debido a que el robot se encuentra en reposo y algunas articulaciones no cargan mucho peso, es muy difícil que se llegue a demandar toda la potencia de la fuente. El consumo de los otros elementos es despreciable en comparación a los servomotores, ya que el de la tarjeta principal es alrededor de 50mA [16], el de la tarjeta servocontroladora de apenas 40mA [22], y el del módulo bluetooth de entre 30 y 40mA [6], por lo que suman un total de hasta 0.13 amperios.

3.4.4. Conexión final del robot

Por último, en Figura 29, se muestra un diagrama resumido de las conexiones necesarias en el robot y los módulos que lo componen.

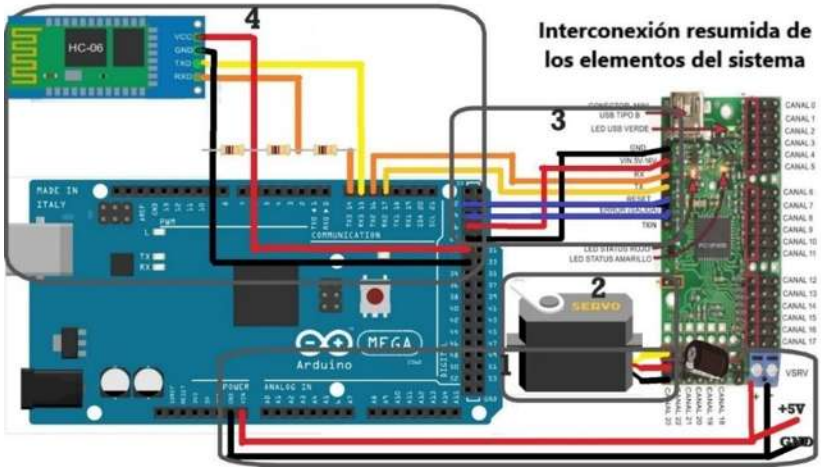

Figura 29 Conexión general del Robot y sus módulos.

En la Figura 29.1 se muestra la conexión de la fuente de poder con la tarjeta Arduino MEGA, y con la tarjeta servo controladora Mini Maestro 24, en la Figura 29.2 un ejemplo de conexión de un servomotor, en la Figura 29.3 las conexiones entre la tarjeta MEGA y la servo controladora, por último en la Figura 29.4 la conexión con el módulo de bluetooth HC-06.

3.5. Conclusiones del capítulo 3

En este capítulo se presentó la descripción de los elementos del sistema dando un enfoque a su hardware y conexiones, y en el caso del humano a sus características,

Los principales cambios del hardware del robot, los cuales fueron mejorar el cableado, etiquetado del mismo, sustitución de servomotores dañados, montaje en una base para facilidad de uso, no se utilizó el acelerómetro giroscopio, y se cambió la tarjeta principal por sus nuevas prestaciones, por todo esto se tiene un hardware del robot con disponibilidad de agregar más módulos, como una pantalla compatible con la tarjeta principal, disponible en el mercado, botones para el control directo de sus articulaciones, o posteriormente combinar el control lógico difuso que implementaba con la captura de posiciones humanas y manipulación de articulaciones.

Para las diferentes conexiones se encuentra el módulo Kinect de Microsoft, el bluetooth de la PC, la conexión del Robot, que abarca desde la conexión de los servomotores, su tarjeta

principal de Arduino MEGA 2560, la conexión el módulo de bluetooth HC-06 del robot, la conexión de la tarjeta servo controladora Mini Maestro 24 CH de Pololo con los servomotores MG995 y la conexión a su vez, con la tarjeta principal, y las características de la fuente de alimentación.

Capítulo 4. Desarrollo del Software

4.1. Introducción al software

En este capítulo se muestra todo software desarrollado para tener un sistema funcional, empezando ejemplos prácticos funcionales para los primeros pasos con los módulos implementados y las plataformas implementadas para su desarrollo, hasta el software final desarrollado para la IHM.

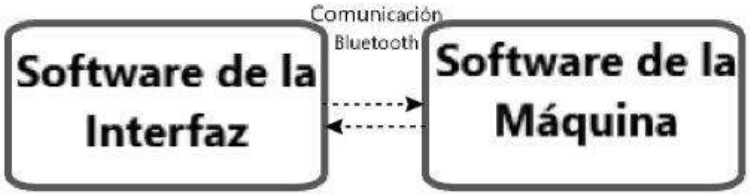

Figura 30 Diagrama general del software desarrollado.

En la **¡Error! No se encuentra el origen de la referencia.** se muestra un diagrama g eneral del software. Como se observa el humano no aparece en el diagrama, pues no tiene cabida como parte del software pues es un ente físico, pero es una parte crucial para que el software desarrollado funciones. Además, se puede observar que el software de la Interfaz se comunica o tiene relación con el software de la Máquina, más sin embargo parte de configuraciones previas y código desarrollado no tiene dependencia con otro software.

4.2. Software para la Interfaz del sistema IHM

El software para la Interfaz abarca desde los controladores y programas previos requeridos para poder utilizar el Kinect en la computadora, los programas desarrollados en Script de Matlab para el reconocimiento de imágenes con el Kinect, las configuraciones del dispositivo Bluetooth integrado de la computadora y la utilización del mismo con Matlab para enviar datos hacia la tarjeta principal, hasta llegar a un Script y a una Interfaz de Usuario Grafica o GUI final en Matlab donde intervienen todos los elementos que conforman la Interfaz.

4.2.1. Instalación de los drivers y herramientas de Kinect para Windows y Matlab.

Para las primeras pruebas del Kinect en Matlab primeramente es necesario instalar los drivers del Kinect para Windows, algunas herramientas de desarrollador, y las bibliotecas de Kinect para Matlab.

Para esto primeramente se debe conectar el Kinect a la PC y esperar que se instalen los primeros controladores, después se deben seguir los siguientes pasos:

-Verificar si se tiene instalada una versión de NET framework igual o superior a 4, y de Visual Studio 2012 o superior sino descargarlas e instalarlas.
-Descargar e instalar Kinect for Windows SDK v1.8 disponible en la página de Microsoft [23].
-Descargar e instalar Kinect for Windows Developer Toolkit v1.8 disponible en la página de Microsoft [24]
-Opcionalmente se puede instalar Kinect for Windows Language Packs v11.0 para tener soporte en español [25].
-Para poder trabajar con Matlab se tiene que descargar e instalar la biblioteca o Toolbox, desde su tienda de aplicaciones, "Image Acquisition Toolbox Support Package for Kinect For Windows Sensor" [26].

4.2.2. Ejemplo para empezar a trabajar con el Kinect en Matlab

Una vez cumpliendo los requerimientos anteriores ya se puede empezar a trabajar con el Kinect en Matlab. Un ejemplo con las instrucciones a seguir se puede encontrar en la página de Matlab donde además se muestran los comandos disponibles para la biblioteca de Kinect para Matlab anteriormente mencionada [27]

Se pueden ver que los dos sensores en el Kinect® para Windows® están representados por dos ID de dispositivo, uno para el sensor de color y uno para el sensor de profundidad. En ese ejemplo, el Dispositivo 1 es el sensor de color y el Dispositivo 2 es el sensor de profundidad. Este ejemplo muestra cómo crear un objeto de entrada de video para que el sensor de color adquiera imágenes RGB y luego que el sensor de profundidad adquiera datos esqueléticos [26].

Los pasos a seguir son los siguientes:

1. Crea el objeto de entrada de video para el sensor de color. DeviceID 1 se utiliza para el sensor de color.
```
vid = videoinput('kinect',1,'RGB_640x480');
```

2. Observe las propiedades específicas del dispositivo en el dispositivo fuente, que es el sensor de color en la cámara Kinect. (este comando causaba un error critico en Matlab 2015b)
```
src = getselectedsource(vid);
```

Se mostrará algo como a continuación:

```
src

Display Summary for Video Source Object:

      General Settings:
        Parent = [1x1 videoinput]
        Selected = on
        SourceName = ColorSource
        Tag =
        Type = videosource

      Device Specific Properties:
        Accelerometer = [0.0 -1.0 0.0]
        AutoExposure = on
        AutoWhiteBalance = on
        BacklightCompensation = AverageBrightness
        Brightness = 0.2156
        CameraElevationAngle = 3
        Contrast = 1
        ExposureTime = 1.0
        FrameInterval = 0
        FrameRate = 30
        Gain = 0
        Gamma = 2.2
        Hue = 0
        PowerLineFrequency = Disabled
        Saturation = 1
        Sharpness = 0.5
        WhiteBalance = 2700
```

Como se puede ver en la salida, el sensor de color tiene un conjunto de propiedades específicas del dispositivo las cuales se muestran en la. Tabla 3.

Tabla 3 Propiedades del sensor de color del Kinect.

Propiedad específica del dispositivo - Sensor de color	**Descripción**
Accelerometer	Devuelve un vector 3D de datos de aceleración para los sensores de color y profundidad. Los datos se actualizan mientras el dispositivo se está ejecutando o presentando una vista previa.
	Este vector 1 x 3 representa los valores x, y y z de la aceleración en unidades de gravedad g (9.81 m / s ^ 2). Por ejemplo,
	[0.06 -1.00 -0.09]
	representa valores de x como 0.06 g, y como -1.00 g, y z como -0.09 g.

AutoExposure	Use para configurar la exposición automáticamente. Esto controla si otras propiedades relacionadas están activadas. Los valores pueden ser **enable** (predeterminado) y **disable**.
	on significa que la exposición se establece automáticamente, y estas propiedades no se pueden configurar y emitirá una advertencia: **FrameInterval, ExposureTime** y **Gain**.
	significa que estas propiedades no se pueden configurar y lanzarán una advertencia: **PowerLineFrequency, BacklightCompensation**, y **Brightness**.
AutoWhiteBalance	Use para habilitar o deshabilitar la configuración automática de balance de blancos.
	on (predeterminado) significa que configurará automáticamente el balance de blancos y no se podrá establecer la propiedad **WhiteBalance**.
	off significa que la propiedad **WhiteBalance** es configurable.
BacklightCompensation	Configura los modos de compensación de contraluz para ajustar la cámara para capturar imágenes dependiendo de las condiciones ambientales.
	Tenga en cuenta que esta propiedad solo es válida si **AutoExposure** está establecido en **enable**. El valor predeterminado es **averageBrightness**.
	Los valores son:
	AverageBrightness. Favorece un nivel de brillo promedio
	CenterPriority. Favorece el centro de la escena
	LowLightsPriority Favorece un bajo nivel de luz
	CenterOnly. Favorece solo al centro
Brightness	Indica el nivel de brillo. El rango de valores es de 0.0 a 1.0, y el valor predeterminado es 0.2156.
	Tenga en cuenta que esta propiedad solo es válida si **AutoExposure** está establecido en **Enabled**.
CameraElevationAngle	Controla el ángulo de la lente del sensor. Este es el ángulo de la cámara en relación con el suelo. El valor debe ser una propiedad entera con un rango de -27 a 27 grados. El valor predeterminado es el último valor establecido, ya que esta es una configuración que se conserva. Solo configúrelo si desea cambiar el ángulo de la cámara. Esta propiedad también se comparte con el sensor de profundidad.
Contrast	Indica el nivel de contraste. Los valores deben estar en el rango de 0.5 a 2, con un valor predeterminado de 1.
ExposureTime	Indica el tiempo de exposición en incrementos de 1 / 10,000 de segundo. El rango de valores es de 0 a 4000, y el valor predeterminado es 0.
	Tenga en cuenta que esta propiedad solo es válida si **AutoExposure** está configurado para **Disabled**.
FrameInterval	Indica el intervalo de cuadro en unidades de 1 / 10,000 de segundo. El rango de valores es de 0 a 4000, y el valor predeterminado es 0.
	Tenga en cuenta que esta propiedad solo es válida si la **AutoExposure** está configurada como **Disabled**.

FrameRate	Marcos por segundo para la adquisición. Esta propiedad es de solo lectura y los valores posibles para el sensor de color son 12, 15 y 30 (valor predeterminado). Refleja la velocidad de fotogramas real cuando se ejecuta.
Gain	Indica un multiplicador para los valores de color RGB. El rango de valores es de 1.0 a 16.0, y el predeterminado es 1.0.
	Tenga en cuenta que esta propiedad solo es válida si la **AutoExposure** está configurada como **Disabled**.
Gamma	Indica la medición gamma. Los valores deben estar en el rango de 1 a 2.8, con un valor predeterminado de 2.2.
Hue	Indica la configuración de matiz. Los valores deben estar en el rango de -22 a 22, con un valor predeterminado de 0.
PowerLineFrequency	Opción para reducir el parpadeo causado por la frecuencia de una línea de alimentación. Los valores están deshabilitados, **FiftyHertz** y **SixtyHertz**. El valor predeterminado es **Disabled**.
	Tenga en cuenta que esta propiedad solo es válida si la AutoExposure está configurada como **Disabled**.
Saturation	Indica el nivel de saturación. Los valores deben estar en el rango de 0 a 2, con un valor predeterminado de 1.
Sharpness	Indica nivel de nitidez. Los valores deben estar en el rango de 0 a 1, con un valor predeterminado de 0.5.
WhiteBalance	Indica la temperatura del color en grados Kelvin. El rango de valores es de 2700 a 6500 y el valor predeterminado es 2700.
	Tenga en cuenta que esta propiedad solo es válida si **AutoWhiteBalance** está configurado como Deshabilitado.

3. Opcionalmente puede establecer algunas de estas propiedades que se muestran en el paso anterior. Por ejemplo, puede que esté adquiriendo imágenes en una situación de poca luz. Puede ajustar la adquisición para esto estableciendo la propiedad **BacklightCompensation** en **LowLightsPriority**, lo que favorece un nivel de luz bajo.
```
src.BacklightCompensation = 'LowLightsPriority';
```

4. Obtenga una vista previa de la secuencia de color llamando a la vista previa en el objeto del sensor de color creado en el paso 1.
```
preview(vid);
```

   Cuando haya terminado la vista previa, cierre la ventana de vista previa.
```
closepreview(vid);
```

5. Crea el objeto de entrada de video para el sensor de profundidad. Tenga en cuenta que se crea un segundo objeto (vid2) y DeviceID 2 se usa para el sensor de profundidad.
```
vid2 = videoinput('kinect',2,'Depth_640x480');
```

6. Observe las propiedades específicas del dispositivo en el dispositivo fuente, que es el sensor de profundidad en el Kinect.

```
 src = getselectedsource(vid2);
src

Display Summary for Video Source Object:

      General Settings:
        Parent = [1x1 videoinput]
        Selected = on
        SourceName = DepthSource
        Tag =
        Type = videosource

      Device Specific Properties:
        Accelerometer = [0.0 -1.0 0.0]
        BodyPosture = Standing
        CameraElevationAngle = 4
        DepthMode = Default
        FrameRate = 30
        IREmitter = on
        SkeletonsToTrack = [1x0 double]
        TrackingMode = off
```

Como puede ver en la salida, el sensor de profundidad tiene un conjunto de propiedades específicas del dispositivo asociadas con el seguimiento del esqueleto. Estas propiedades son específicas del sensor de profundidad y se muestran en la
Tabla 4.

Tabla 4 Propiedades del sensor de profundidad del Kinect.

Propiedad específica del dispositivo - Sensor de profundidad	**Descripción**
Accelerometer	Devuelve un vector 3D de datos de aceleración para los sensores de color y profundidad. Los datos se actualizan mientras el dispositivo se está ejecutando o presentando una vista previa.
BodyPosture	Indica si los esqueletos rastreados están de pie o sentados. Los valores son **Standing** (da 20 puntos de datos del esqueleto) y **Seated** o sentado (proporciona datos de 10 puntos del esqueleto, usando índices del 2 al 11). **Standing** es el predeterminado.
	Tenga en cuenta que, si **BodyPosture** está configurado en modo Sentado, y el Modo de seguimiento está configurado en **Position**, no se devuelve ninguna posición, ya que **Position** es la ubicación de la articulación de la cadera y la articulación de la cadera no se rastrea en modo Sentado.
CameraElevationAngle	La información para esta propiedad es la misma que para el sensor de color
DepthMode	Indica el rango de profundidad en el mapa de profundidad. Los valores son Predeterminados (rango de 50 a 400 cm) y Cerca (rango de 40 a 300 cm).

FrameRate	Son los cuadros por segundo para la adquisición. Esta propiedad es de solo lectura y se fija en 30 para el sensor de profundidad para todos los formatos. Refleja la velocidad de fotogramas real cuando se ejecuta.
IREmitter	Controla si el emisor IR está encendido o apagado. Inicialmente, el valor predeterminado está activado. Sin embargo, esta es una propiedad adhesiva, por lo que el valor predeterminado es el último valor establecido. Si lo configura en apagado, permanecerá desactivado en usos futuros hasta que cambie la configuración.
	Una ventaja de esta propiedad es que es útil cuando se usan múltiples dispositivos Kinect para evitar interferencias.
SkeletonsToTrack	Indica el ID de seguimiento de esqueleto devuelto como parte de los metadatos. Los valores son:
	[ ] Default tracking
	[**TrackingID1**] Sigue 1 esqueleto con ID de seguimiento = **TrackingID1**
	[**TrackingID1 TrackingID2**] Sigue 2 esqueletos con ID de seguimiento; **TrackingID1** y **TrackingID2**
TrackingMode	Indica el estado de seguimiento. Los valores son:
	Skeleton rastrea esqueleto completo con articulaciones
	Position solo rastrea la posición de la cadera
	Off deshabilita el seguimiento de posición del esqueleto(default)
	Tenga en cuenta qué si **BodyPosture** está configurado en modo **Seating**, y el Modo de seguimiento está configurado en **Position**, no se devuelve ninguna posición, ya que **Position** es la ubicación de la articulación de la cadera y la articulación de la cadera no se rastrea en modo **Sentado**.

7. Inicie el segundo objeto de entrada de video (la secuencia de profundidad).

```
start(vid2);
```

8. Se accede a los datos del esqueleto como metadatos en el flujo de profundidad. Puede usar getdata para acceder a él.

```
% Obtenga los datos en el objeto.
[frame, ts, metaData] = getdata (vid2);
% Mira los metadatos para ver los parámetros en los datos del esqueleto.
metadata
```

```
metaData =

10x1 struct array with fields:
    AbsTime: [1x1 double]
    FrameNumber: [1x1 double]
    IsPositionTracked: [1x6 logical]
    IsSkeletonTracked: [1x6 logical]
    JointDepthIndices: [20x2x6 double]
    JointImageIndices: [20x2x6 double]
    JointTrackingState: [20x6 double]
    JointWorldCoordinates: [20x3x6 double]
    PositionDepthIndices: [2x6 double]
    PositionImageIndices: [2x6 double]
    PositionWorldCoordinates: [3x6 double]
    RelativeFrame: [1x1 double]
    SegmentationData: [640x480 double]
    SkeletonTrackingID: [1x6 double]
    TriggerIndex: [1x1 double]
```

Estos campos de metadatos están relacionados con el seguimiento de los esqueletos y se muestran en la Tabla 5

Tabla 5 Metadatos obtenidos de Sensor de Profundidad del Kinect.

MetaData	Descripción
AbsTime	Este es un dato tipo double de 1x1 y representa la marca de tiempo completa, incluidas la fecha y la hora, en formato de reloj MATLAB.
FrameNumber	Este es un dato tipo doble de 1 x 1 y representa el número de cuadro.
IsPositionTracked	Esta es una matriz booleana 1 x 6 de valores verdadero / falso para el seguimiento de la posición de cada uno de los seis esqueletos. A 1 indica que se hace un seguimiento de la posición y un 0 indica que no.
IsSkeletonTracked	Esta es una matriz booleana de 1 x 6 de valores verdadero / falso para el estado de seguimiento de cada uno de los seis esqueletos. A 1 indica que se rastrea y 0 indica que no.
JointDepthIndices	Si la propiedad BodyPosture se establece en Permanente, esta es una matriz doble de 20 x 2 x 6 de coordenadas x e y para 20 puntos en píxeles en relación con la imagen de profundidad, para los seis posibles esqueletos. Si BodyPosture está configurado como Sentado, esto sería un doble de 10 x 2 x 6 para 10 articulaciones.
JointImageIndices	Si la propiedad **BodyPosture** está establecida en Permanente, esta es una matriz doble de 20 x 2 x 6 de coordenadas x e y para 20 puntos en píxeles en relación con la imagen en color, para los seis posibles esqueletos. Si **BodyPosture** está configurado como Sentado, se tendrá una matriz tipo double de 10 x 2 x 6 para 10 articulaciones.
JointTrackingState	Esta matriz de 20 x 6 enteros contiene valores enumerados para la precisión de seguimiento de cada articulación para los seis esqueletos. Los valores incluyen:
	0 no rastreado
	1posición inferida
	2 posición rastreada
JointWorldCoordinates	Esta es una matriz doble de 20 x 3 x 6 de coordenadas x, y, y z para 20 puntos, en metros desde el sensor, para los seis esqueletos posibles, si **BodyPosture** se establece en Permanente. Si está configurado como Sentado, esto sería un doble de 10 x 3 x 6 para 10 articulaciones.
PositionDepthIndices	Una matriz doble de 2 x 6 de coordenadas X e Y de cada esqueleto en píxeles en relación con la imagen de profundidad.
PositionImageIndices	Una matriz doble de 2 x 6 de coordenadas X e Y de cada esqueleto en píxeles en relación con la imagen en color.

PositionWorldCoordinates	Una matriz doble de 3 x 6 de las coordenadas X, Y y Z de cada esqueleto en metros relativos al sensor.
RelativeFrame	Esta matriz doble de 1 x 1 representa el número de fotograma relativo a la ejecución de un disparador si se usa la activación.
SegmentationData	Doble matriz de tamaño de imagen con cada píxel asignado a un esqueleto rastreado / detectado, representado por los números 1 a 6. Este mapa de segmentación es un mapa de bits con valores de píxel correspondientes al índice de la persona en el campo de visión que está más cerca del cámara en esa posición de píxel. Un valor de 0 significa que no hay un esqueleto rastreado.
SkeletonTrackingID	Esta matriz entera de 1 x 6 contiene las ID de seguimiento de los seis esqueletos. El Kinect genera los ID de seguimiento.
TriggerIndex	Este es un valor doble de 1 x 1 y representa el trigger al que está asociado el evento si se usa la activación.

9. Puede ver cualquier propiedad individual al profundizar en los metadatos. Por ejemplo, para mirar la propiedad IsSkeletonTracked.

```
metaData.IsSkeletonTracked
```

```
ans =

     1     0     0     0     0     0
```

En este caso, significa que, de los seis posibles esqueletos, hay un esqueleto que se rastrea y está en la primera posición. Si tiene varios esqueletos, esta propiedad es útil para confirmar cuáles están siendo rastreados.

10. Obtenga las ubicaciones conjuntas para la primera persona en coordenadas mundiales utilizando la propiedad JointWorldCoordinates. Dado que esta es la persona en la posición 1, el índice usa 1.

```
  metaData.JointWorldCoordinates(:,:,1)
```

```
ans =

   -0.1488   -0.3257    2.1674
   -0.1488   -0.2257    2.1674
   -0.1368   -0.0098    2.2594
   -0.1324    0.1963    2.3447
   -0.1024   -0.0058    2.2574
   -0.1622   -0.3361    2.1641
   -0.1844   -0.6279    1.9877
   -0.4044   -0.6779    1.9877
    0.0981   -0.0125    2.2683
    0.2364    0.2775    2.2117
    0.3775    0.5872    2.2022
    0.4075    0.6372    2.2022
   -0.2532   -0.4492    2.0742
   -0.1869   -0.8425    1.8432
   -0.1869   -1.2941    1.8432
   -0.1969   -1.3541    1.8432
   -0.0360   -0.4436    2.0771
    0.0382   -0.8358    1.8286
    0.1096   -1.2114    1.5896
    0.1196   -1.2514    1.5896
```

Las columnas representan las coordenadas X, Y y Z en metros de los 20 puntos en el esqueleto 1.

11. Opcionalmente puede ver los datos de segmentación como una imagen.

```
% Ver los datos de segmentación como una imagen.
  imagesc (metaDataDepth.SegmentationData);

%Configure el mapa de color para enviar el código de color a las personas
detectadas.
  colormap(jet);
```

Índices conjuntos de BodyPosture

La propiedad BodyPosture, en el paso 5, indica si los esqueletos rastreados están de pie o sentados. Los valores son Standing (da datos de esqueleto de 20 puntos) y Seated (proporciona datos de esqueleto de 10 puntos.

Este es el orden de las articulaciones devuelto por el adaptador Kinect:

Columna Centro = 1;
Espina dorsal = 2;
Centro de hombro = 3;
Cabeza = 4;
Hombro a la izquierda = 5;
Codo izquierdo = 6;
Muñeca izquierda = 7;
Mano izquierda = 8;
Hombro a la derecha = 9;
Codo derecho = 10;
Muñeca derecha = 11;
Mano derecha = 12;
Cadera izquierda = 13;
Rodilla izquierda = 14;
Tobillo izquierdo = 15;
Pie izquierdo = 16;
Cadera derecha = 17;
Rodilla a la derecha = 18;
Tobillo a la derecha = 19
Pie derecho = 20

4.2.3. Configuraciones para utilizar el bluetooth integrado de la PC

Se explican las configuraciones para el dispositivo bluetooth en que utiliza la computadora, el cual ya trae integrado, pero en caso no ser así, los pasos a seguir son muy similares a los descritos a continuación.

- Activa el dispositivo y haz que se pueda detectar.
- Selecciona el botón Inicio y luego selecciona Configuración > Dispositivos Bluetooth y otros dispositivos.
- Activa Bluetooth y luego selecciona Agregar Bluetooth u otro dispositivo > Bluetooth. Elige el dispositivo y sigue otras instrucciones que aparezcan, como el código de emparejamiento y luego selecciona Listo.

Ahora bien, si se desea cambiar la velocidad de transmisión, ya que por defecto es 9600 baudios, y cambiar otras configuraciones del puerto COM referente al dispositivo bluetooth se deben seguir las siguientes instrucciones:

- Activa el dispositivo y haz que se pueda detectar.
- Abrir la herramienta “Administrador de dispositivos”
- Buscar la sección Puertos (COM y LPT) y desplegarla
- Dar clic derecho sobre el puerto COM correspondiente al dispositivo Bluetooth
- Ir a la pestaña “Configuración del Puerto” y aplicar las configuraciones necesarias.
- Dar clic en aceptar y cerrar las ventanas.

En la Figura 31 se muestra un ejemplo de dichas configuraciones.

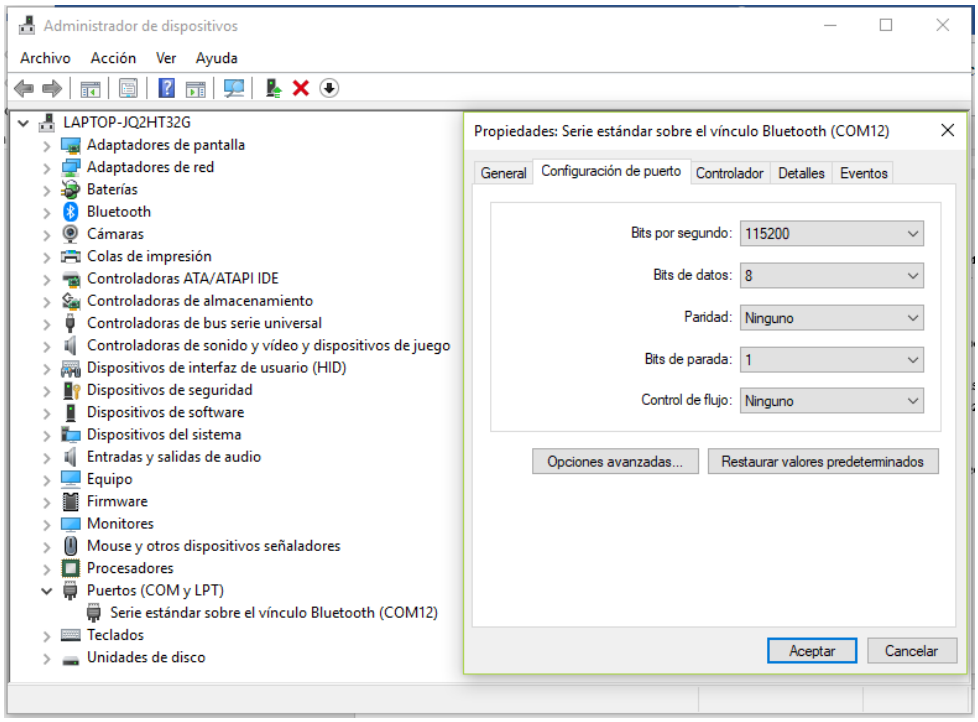

Figura 31 Configuraciones del puerto serie del Bluetooth integrado de la PC.

4.2.4. Comunicación de la PC con la tarjeta principal vía Bluetooth usando Matlab

Este código aparece en el Anexo A-2 y tiene como función establecer una conexión inalámbrica con el dispositivo bluetooth de la tarjeta principal, el módulo HC-06, y enviar uno o más comandos para vómer uno o varios servomotores. La trama debe ser compatible con la requerida por la tarjeta principal, cuyo código aparece en el Anexo A-7

En la Figura 32, se presenta un diagrama con el flujo del programa en Matlab para las primeras conexiones vía bluetooth de la PC con la tarjeta principal (Arduino MEGA), donde se manda un comando específico.

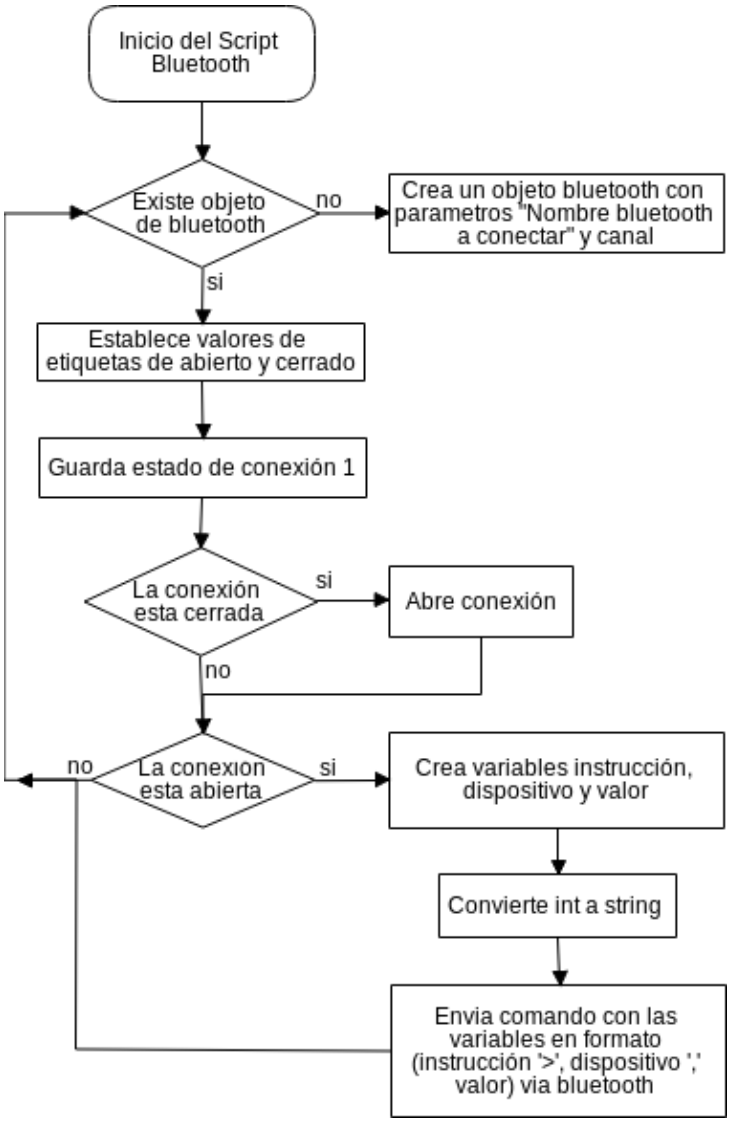

Figura 32 Conexión en Matlab con el Robot vía Bluetooth

Primero pregunta si no existe un objeto bluetooth, de no existir lo crea y recibe como parámetros el nombre del dispositivo Bluetooth al que se quiere desea conectar, en este trabajo se eligió "Roboto" y en la sección 4.3 se muestra como configurar dicho módulo con ese nombre. Él otro parámetro es el canal de ese dispositivo bluetooth si es que permite múltiples conexiones.

4.2.5. Script desarrollado en Matlab para un sistema funcional

El diagrama de la Figura 33 muestra el flujo que sigue el código desarrollado en un Script de Matlab, en el cual tienen cabida todos los elementos que forman la Interfaz del sistema IHM, donde se indica la inicialización de del Bluetooth interno de la PC para poder enviar instrucciones a la Máquina,

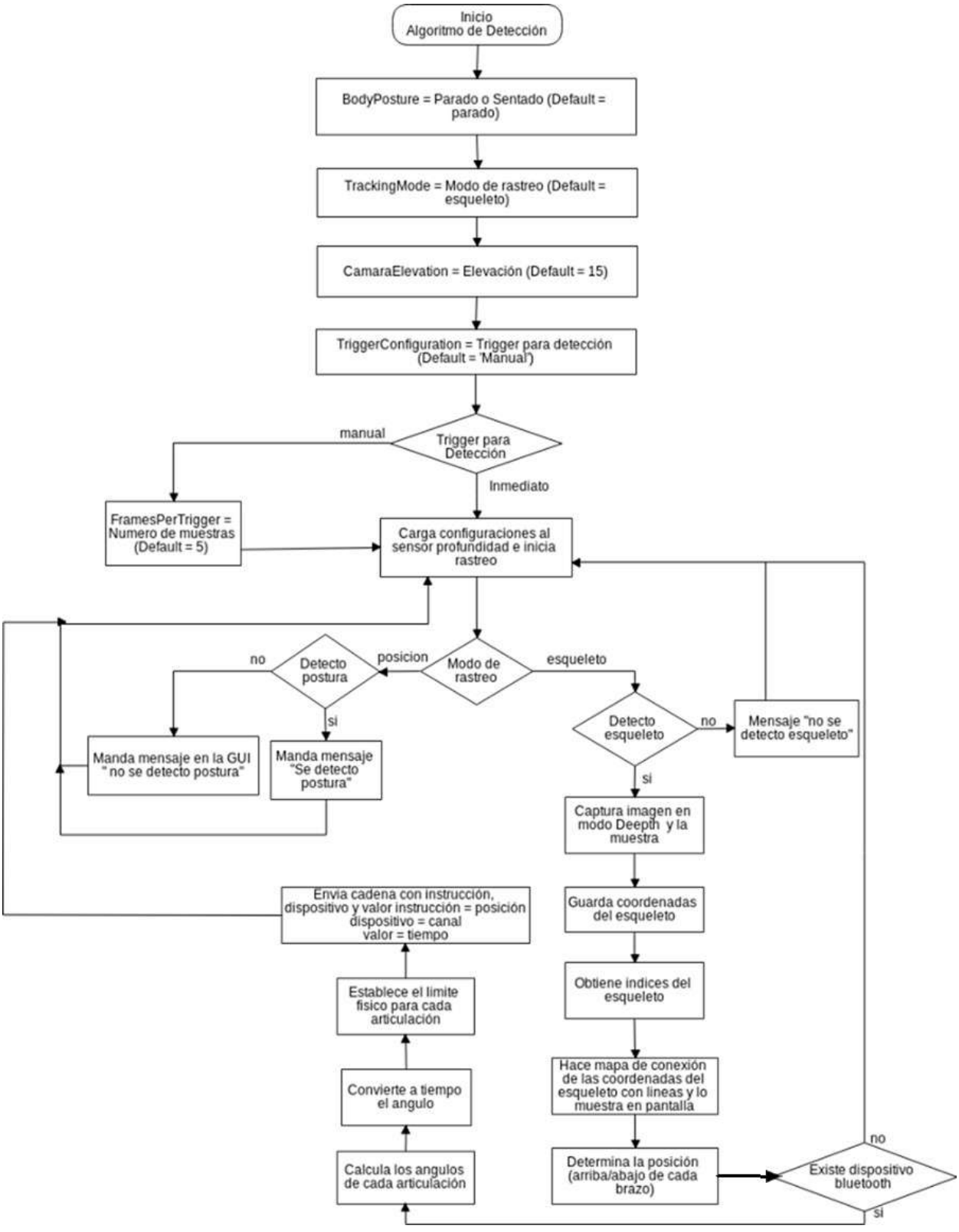

Figura 33 Diagrama del flujo del algoritmo de detección de movimientos del Kinect.

El diagrama mencionado principalmente describe el núcleo del programa desarrollado para la detección de posturas humanas, que incluye las configuraciones previas que se deben especificar, el inicio de detección de posturas usando funciones que la biblioteca del Kinect para Matlab ofrece, se almacenan las coordenadas obtenidas en la detección, y se procede a hacer un cálculo de los ángulos que hay entre diferentes extremidades, como las piernas, los brazos, antebrazos, etc.

Posteriormente a la detección de ángulos los datos se pasan por una etapa de filtrado o parametrización donde se establecen los límites físicos mostrados en la Tabla 7, después de esto finalmente se envían datos vía Bluetooth a la tarjeta principal del robot, en un formato entendible por la misma, para la manipulación de los servomotores a una determinada posición haciendo uso de la tarjeta controladora, que está conectada en un puerto serie (Serial2) de la tarjeta principal, otro de los puerto serie utilizados es el correspondiente al módulo de Bluetooth HC.06 (puerto Serial3), y uno más de los puertos de la tarjeta principal es uno que se encuentra físicamente mapeado con el puerto USB, y que se utiliza principalmente para depuración del código cuando la tarjeta se encuentra conectada vía USB con la computadora, utilizando un monitor de puerto serie (Arduino proporciona una herramienta llamada Monitor Serial)

A continuación, en la sección siguiente se muestra información útil referente al cálculo de ángulos en triángulos escalenos (todos sus lados diferentes), explicando el procedimiento con un ejemplo básico y aplicando estos conceptos en el código desarrollado

4.2.5.1. Calculo de los ángulos de las articulaciones de la postura detectada

A continuación, se detalla como calcular los ángulos de las articulaciones de las posturas detectadas por el Kinect para llegar a un resultado como en la Figura 34, donde es necesario primero calcular la longitud de cada extremidad del cuerpo, posteriormente por ley de cosenos o senos, calcular los ángulos, y en algunas ocasiones es necesario hacer una rotación de la recta para poder calcular un ángulo final.

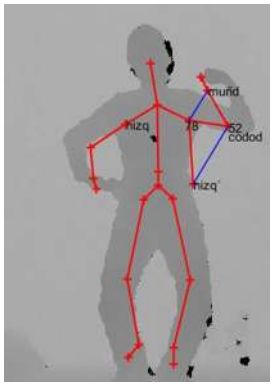

Figura 34 Detección con ángulos en axila y codo derecho

Cálculo de la longitud de una línea recta dados dos puntos.

Distancia entre dos puntos dados. Sean P1(xl, yl) y P2(x2, y2) (Figura 35) dos puntos dados cualesquiera Vamos a determinar la distancia d entre Pi y P2, siendo d = $\overline{|P_1P_2|}$. Por PI P2 tracemos las perpendiculares P1A y P2D a ambos ejes coordenados, como se indica en la //, y sea E su punto de intersecci6n. Consideremos el triángulo rectángulo P1EP2. Por el teorema de Pitágoras tenemos [28]:

$\boldsymbol{d2} = \overline{P_1P_2^2}. = \overline{P_2E^2} = \quad \overline{P_1E^2}$

TEOREMA 1. La distancia d entre dos puntos P1(x1, yl) y P2(x2, y2) está dada por la formula [28]

d = $\sqrt{(x1 - x2)^2 + (y1 - y2)^2}$

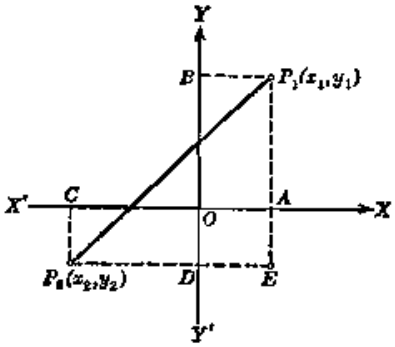

Figura 35 Línea recta entre dos puntos [28] .

Es necesario realizar el cálculo de cuanto mide cada extremidad, con los datos entregados por las funciones del Kinect en Matlab, ya longitud no es fija y depende de cada persona y de la cercanía o lejanía que tenga con el Kinect, y aunque no se estén midiendo las extremidades en unidades reales como metros (que si es posible calcular pero en este trabajo no se hacen dichos cálculo) se puede hacer el cálculo de la longitud de una extremidad (pierna, brazo, antebrazo, hombros, manos, etc.) en pixeles.

Por ejemplo, para calcular la longitud, en pixeles, del brazo y antebrazo derecho y recordando el orden de las articulaciones devueltos por el Kinect, tenemos:

Hombro a la derecha = 9;
Codo derecho = 10;
Muñeca derecha = 11;

Entonces en Matlab y aplicando la ecuación anterior en Matlab tenemos:
ya que el brazo derecho está formado entre Hombro a la derecha y codo derecho y el antebrazo derecho por el codo derecho y a muñeca derecha entonces:

```
brazod=sqrt((esqueleto(9,1)-esqueleto(10,1))^2+(esqueleto(9,2)-
esqueleto(10,2))^2);
antebrazod=sqrt((esqueleto(10,1)-esqueleto(11,1))^2+(esqueleto(10,2)-
esqueleto(11,2))^2);
```

Calculo de ángulos y lados de un triángulo escaleno por Ley de Senos y cosenos

Teorema de la ley de senos. Los lados de un triángulo son proporcionales a los senos de los ángulos opuestos [29].

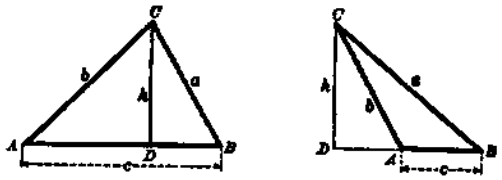

Figura 36 Triangulo escaleno 1 ángulos agudos, 2 ángulo obtuso

En el libro de Trigonometría plana y esférica de Granville se muestra de la demostración las siguientes igualdades:

$$\frac{\boldsymbol{a}}{\boldsymbol{sen(A)}} = \frac{\boldsymbol{b}}{\boldsymbol{sen(B)}} = \frac{\boldsymbol{c}}{\boldsymbol{sen(C)}}$$

Teorema de la Ley de cosenos. En un triángulo cualquiera el cuadrado de un lado es igual a la suma de los cuadrados de los otros dos menos el doble producto de estos dos lados por el coseno del ángulo que forman. En el libro de Trigonometría plana y esférica de Granville [29] se muestra la demostración de las siguientes igualdades, para entender esto mejor, ver la Figura 36:

$$\boldsymbol{a^2 = b^2 + c^2 - 2bc\ cos\ A}$$
$$\boldsymbol{b^2 = a^2 + c^2 - 2ac\ cos\ B}$$
$$\boldsymbol{c^2 = a^2 + b^2 - 2ab\ cos\ C}$$

Ahora bien, llevando esto al trabajo realizado, para calcular los ángulos de interés de las articulaciones veamos cómo queda aplicado para calcular el ángulo del codo derecho en Matlab:

```
ang_codo=radtodeg(acos((imahombro_munecad^2-brazod^2-antebrazod^2)/(-
2*(brazod)*(antebrazod))));
```

Obsérvese que aparece la variable `imahombro_munecad` que corresponde al lado imaginario del triángulo que se formaría entre el hombro derecho y la muñeca derecha, para así completar los tres lados del triángulo y poder aplicar la ley de cosenos despejada de la forma:

$$\boldsymbol{A = arcos((a^2 - b^2 - c^2)/(-2 * b * c))}$$

Donde **A** es el ángulo del codo**, a** corresponder al lado imaginario entre la muñeca y el hombro derecho, **b** corresponde al brazo derecho y **c** al antebrazo derecho.

Rotación de ejes de una línea recta en un plano 2D.

Rotaciones: Cada punto del plano puede unirse al origen de coordenadas mediante una recta, que forma un cierto ángulo θ con el eje x. Una rotación de centro el origen y ángulo α transforma cada punto P en otro P' cuyo ángulo con el eje x es α + θ. En la Figura 37 se muestra mejor esto.

Figura 37 Rotación de ángulo de una recta respecto a un punto en el origen o eje coordenado.

La matriz G mostrada en la Figura 37 muestra a una matriz de rotación, sean un punto coordenado P1(x1, y2), una rotación de 90 grados tomado como el otro punto de la recta el eje coordenado. Entonces el punto P1'se calcula de la siguiente manera:
P1'=x1', y1'

Donde

x1' = x1*cos(90) – y1*sen(90)
y1' = x1'*sen(90) – y1*cos(90)

En algunas ocasiones es necesario hacer una rotación a una articulación para poder hacer el cálculo de los ángulos, por ejemplo, si se quisiera calcular el ángulo de la axila derecha, los lados que se podrían seleccionar podrían ser la recta formada entre los dos hombros, la el brazo derecho y la recta imaginaria entra el codo y el hombro izquierdo, pero se tendría el inconveniente de que al sobrepasar el codo de la línea horizontal formada por los hombros, entonces se llegaría casi al límite de un ángulo de un triángulo que es de 180 grados, por lo que al sobrepasarlo el ángulo ya no seguiría aumentando sino que empezaría a disminuir, pues ahora el triángulo formado por las extremidades sería otro.

Para solucionar esto si bien se puede indicar en Matlab que, si se sobrepasa la horizontal formada por los hombros, el ángulo resultante sea 180 + (180-sngulo resultante)
Otra solución es rotar 90 grados la línea formada por los hombros, y ahora la limitante de 180 grados quedaría cuando el brazo este totalmente vertical y sobrepase ese límite en dirección a la cabeza. En la Figura 34 se muestran un gráfico con un ejemplo de detección con estos lados y ángulos, para poder entender mejor lo explicado.
Al aplicar esto en Matlab tenemos:

```
xprima=((cos(90)*(esqueleto(5,1)-esqueleto(9,1)))+(sin(90)*(esqueleto(5,2)-
esqueleto(9,2))))+ esqueleto(9,1)

yprima=((sin(90)*(esqueleto(5,1)-esqueleto(9,1)))+(cos(90)*(esqueleto(5,2)-
esqueleto(9,2))))+esqueleto(9,2)

hombrosrot=sqrt((xprima-esqueleto(9,1))^2+(yprima-esqueleto(9,2))^2);
```

Obsérvese que para poder aplicar la rotación fue necesario restar a las coordenadas del hombro izquierdo el valor de las coordenadas del hombro derecho (se realizó una translación de punto coordenado), para tomar como referencia coordenada el punto correspondiente al hombro derecho y al final se suma al resultado de la rotación el valor de la coordenada que había sido restado para no alterar la posición final de la recta.

4.2.6. Interfaz gráfica en GUIDE de Matlab vía bluetooth

La interfaz se desarrolló en una herramienta de Matlab llamada GUIDE, la cual permite crear diferentes objetos como paneles, botones, selectores de tabla, tablas, graficas, etiquetas, selectores, entre otros. Además de desplazarlos y ajustar el tamaño a conveniencia del desarrollador. Y además crear funciones relacionadas al evento cuando un botón o selector es clicado o seleccionado. Si se requiere más información sobre cómo crear una GUI en GUIDE de Matlab, se puede consultar un manual [30] donde ejemplifiquen como hacer diferentes funciones a partir de objetos creados gráficamente.

La interfaz desarrollada se muestra en la Figura 38 y sus respectivos bloques, cuya descripción se muestra a continuación.

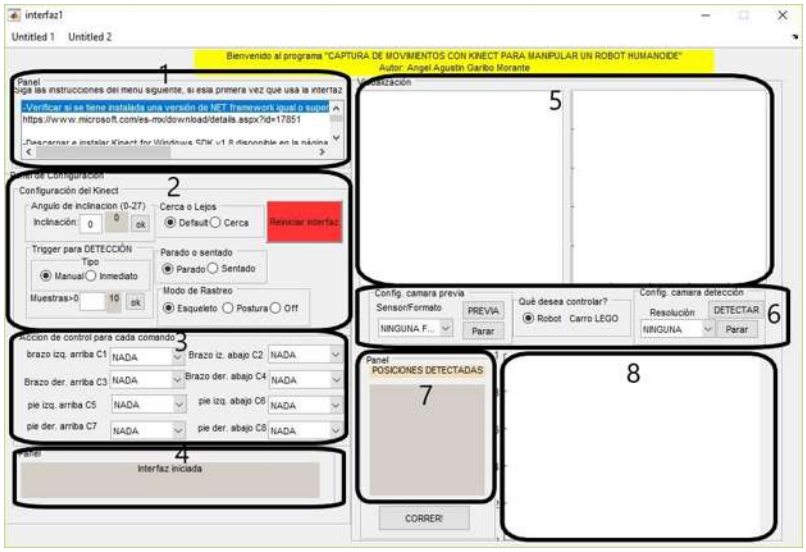

Figura 38 Paneles de la interfaz de usuario gráfica desarrollada.

El cuadro 1 es correspondiente a las instrucciones a seguir para cuando se hace uso del Kinect por primera vez.

En el cuadro 2 se muestran las configuraciones posibles a los sensores del Kinect, a la inclinación, al número de muestras tomadas, tipo de detección, etc. El diagrama de la Figura 39

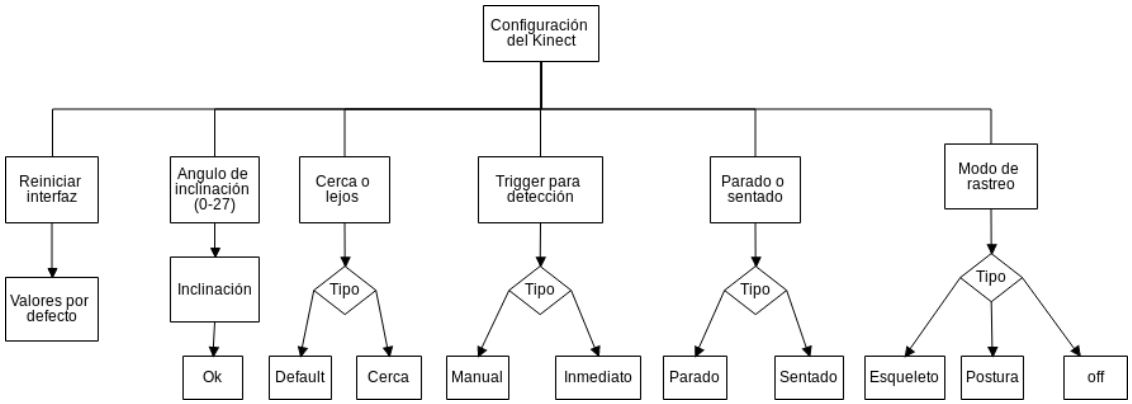

Figura 39 Configuraciones del Kinect disponibles en la interfaz

En el cuadro.3 es un bloqué correspondiente a una funcionalidad que se agregará en trabajos futuros, para poder trabajar con el robot de LEGO Mindstorms EV3 [31] Donde se le asignará un comando de (avanzar, retroceder, girar a la derecha o izquierda, avanzar o retroceder con giro, o manipular herramienta al detectar determinada postura. Esto se dejará para trabajos fututos.

En el cuadro .4 se muestra una seudo terminal, donde se muestra la instrucción o configuración realizada después de que esta surtió efecto. Útil para ver el flujo del programa.

En el cuadro 5 en la parte de la izquierda es la ventana donde se mostrarán visualizaciones previas de todos los objetos de video disponibles para en el Kinect los cuales se muestran en la Tabla 6:

En el cuadro 6, se encuentran unos paneles donde se selecciona el tipo de objeto de video con el que se desea trabajar, ya sea solo para visualización previa sin detección de posturas, en la que se encuentran disponibles todos los formatos de video de la Tabla 6, y el panel de la izquierda donde se selecciona únicamente el formato Depth (Profundidad), para detección de profundidad y sus tres resoluciones disponibles, que es el tipo de video que soporta detección y rastreo de posturas y esqueletos y es con el que se trabaja.

A continuación, en Figura 40 se muestra el diagrama correspondiente a las configuraciones previas a la captura de video que se pueden hacer en la GUI desarrollada (Graphical User Interfaz)

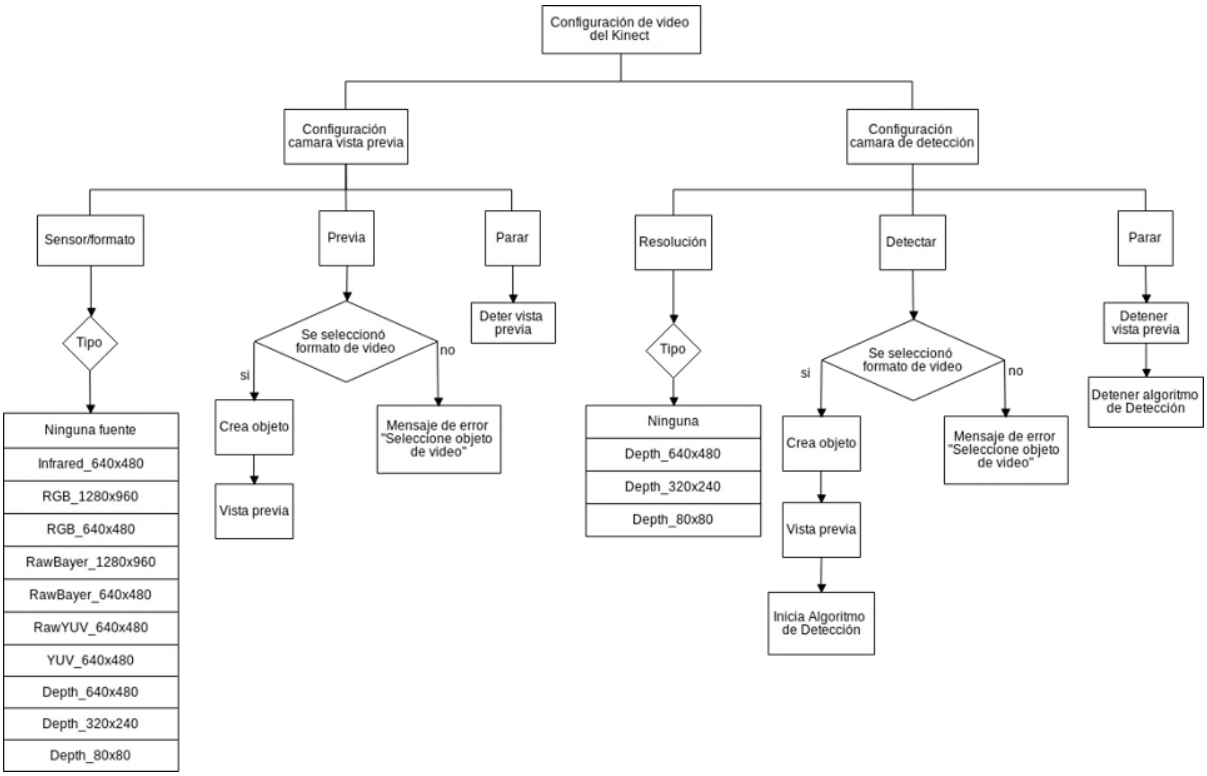

Figura 40 Diagrama de la selección de los objetos de video el Kinect disponibles en la interfaz.

Tabla 6 Fuentes de Video y resoluciones disponibles en el Kinect.

Fuente de video	**Resoluciones de video disponibles**
Infrared	640x480
RGB	1280x960 640x480
RawBayer	1280x960 640x480
RawYUV	640x480
YUV	640x480
Depth	640x480 320x240 80x60

En el cuadro 7 se muestra un panel donde se mostrarán algunas de las posturas detectadas cuando se haya seleccionado un formato de video Depth y se haya dado clic en el botón DETECTAR. Al presionar dicho botón se sigue un flujo del programa como se muestra en la Figura 33.

Por último, en el cuadro 8 aparece una sección para graficar que se usará para trazar una ruta seguida por el Robot LEGO Mindstorms [31] o cualquier otro robot móvil que se quisiera implementar, después de habérsele ordenado un comando de los anteriormente mencionados a partir de una postura detectada.

Y las configuraciones disponibles de los objetos de video se muestran en// cabe mencionar que pueden ser usados todos los sensores de video disponibles del Kinect, ver Tabla 5, pero únicamente para la detección de postura y esqueleto se utiliza el de profundidad (Depth) a diferentes resoluciones.

4.3. Software de la Máquina del sistema IHM

El software de la Máquina abarca todo el código desarrollado para la tarjeta principal desde las configuraciones previas del módulo Bluetooth HC-06 con comandos AT, configuraciones previas de la tarjeta controladora con el software del fabricante, y recepción de datos de la tarjeta principal provenientes de la PC vía bluetooth, hasta llegar a un software de la Máquina final funcional.

4.3.1. Software desarrollado y configuraciones para el Módulo Bluetooth HC-06

Como ya se mencionó en el capítulo 2, este módulo se configura con comandos AT, escribiéndolos directamente por el pin RX, para ello se explica un poco más en qué consisten dichos comandos:

Formato de los comandos AT. Todos los comandos están constituidos por variable tipo strings (cadenas de texto) de código ASCII. Los caracteres que forman el comando deben trasmitirse, desde el microcontrolador al módulo HC-06, en un solo tiempo, sin pausas (a 9600 bps N,8,1, en caso que el módulo tenga su valor de velocidad default) y sin ningún carácter adicional de terminación como CR/LF.

Prueba de ping. El Bluetooth no toma ninguna medida, simplemente confirma con "OK" y le informa que la comunicación fue exitosa.

Enviar: AT

Respuesta: OK

Establecer la velocidad en baudios

Establece la velocidad de transmisión en baudios UART de Bluetooth. La velocidad en baudios se establece mediante un índice hexadecimal de '1' a 'C'.

Los índices son: 1: 1200, 2: 2400, 3: 4800, 4: 9600, 5: 19200, 6: 38400, 7: 57600,

8:115200, 9: 230400, A: 460800, B: 921600, C: 1382400
Enviar: AT + BAUD <índice>
Respuesta: OK <velocidad en baudios>

Establecer el nombre del dispositivo Bluetooth.

Enviar: AT + NAME <nombre del dispositivo>
Respuesta: OKsetname

Establecer el código PIN de Bluetooth

Establece el código de seguridad necesario para conectarse al dispositivo.

Enviar: AT + PIN <código de 4 dígitos>
Respuesta: OK <código de 4 dígitos>

Verifique la revisión del firmware

Enviar: AT +VERSION
Respuesta: Linvor V1.8

Código desarrollado para probar módulo HC-06

Consiste en un código conocido como "echo" o "eco", donde lo que es leído por un puerto serie de la tarjeta principal, se envía a otro puerto serie, es este caso el puerto serie USB usado para conectar la tarjeta a la computadora. Este código se muestra en el

RECOMENDACIÓN. Al inicio del código se puede observar la secuencia de arranque para forzar al bluetooth al modo comandos AT, pero físicamente se encuentra desconectado del pin que permite ejecutar dicha secuencia. Conectar si se desconoce a que *baudrate* está configurado su módulo bluetooth.

4.3.2. Primeras pruebas de la tarjeta Mini Maestro

Para empezar a controlar los servos que queramos de forma sencilla existe un software del fabricante de la tarjeta disponible para varias plataformas llamado Maestro Control Center.

Primero deben ser instalados los drivers de la tarjeta siguiendo las instrucciones que se pueden encontrar en una guía del fabricante [22] en la sección "comenzando".

Una vez instalados y conectada la tarjeta Vía USB, es posible conectar con la tarjeta, seleccionando la pestaña "Connected to:" y elegir nuestra tarjeta, en este punto ya es posible conectar un servomotor mandarle instrucciones desde la interfaz para hacer pruebas de desplazamiento de su posición, aceleración y velocidad, pero se recomienda antes ir a la pestaña "Serial Settings", después en el apartado "Serial Mode" seleccionar "UART, fixed baud rate" y

seleccionar la velocidad con la que se quiere trabajar por comunicación UART (Universal Asynchronous Receiver-Transmitter, o Transmisor-Receptor Asíncrono Universal, por sus siglas en ingles) , lo que coloquialmente se conoce como comunicación serial o serie. En este trabajo se configuró a 115200 baudios por considerarse una velocidad relativamente rápida y donde los errores en el envió-recepción no son tan grandes como para velocidades mayores. En la Figura 41 se muestran estas configuraciones.

RECOMENDACIÓN. Es necesario hacer esta reconfiguración si no se sabe a qué velocidad ha sido configurada la tarjeta previamente, y para indicar a la tarjeta con qué tipo de comunicación de los disponibles mencionados [19] va a trabajar De lo contrario se puede omitir y saltarse directamente a la transmisión de datos vía puerto serie de la controladora (los pines TX y RX de la tarjeta).

Mediante esta interfaz de Pololu, es posible obtener el valor en ms correspondiente a la posición inicial que se le quisiera especificar al robot, útil cuando se trabaje en la plataforma de la tarjeta principal donde se desarrollará el código, en este trabajo como se utilizó una placa de Arduino la IDE (Integrated Development Environment, o Entorno de Desarrollo Integrado, por sus siglas en inglés) con la que se trabajo fue la IDE Arduino versión 1.8.3 de la que más adelante se escribe.

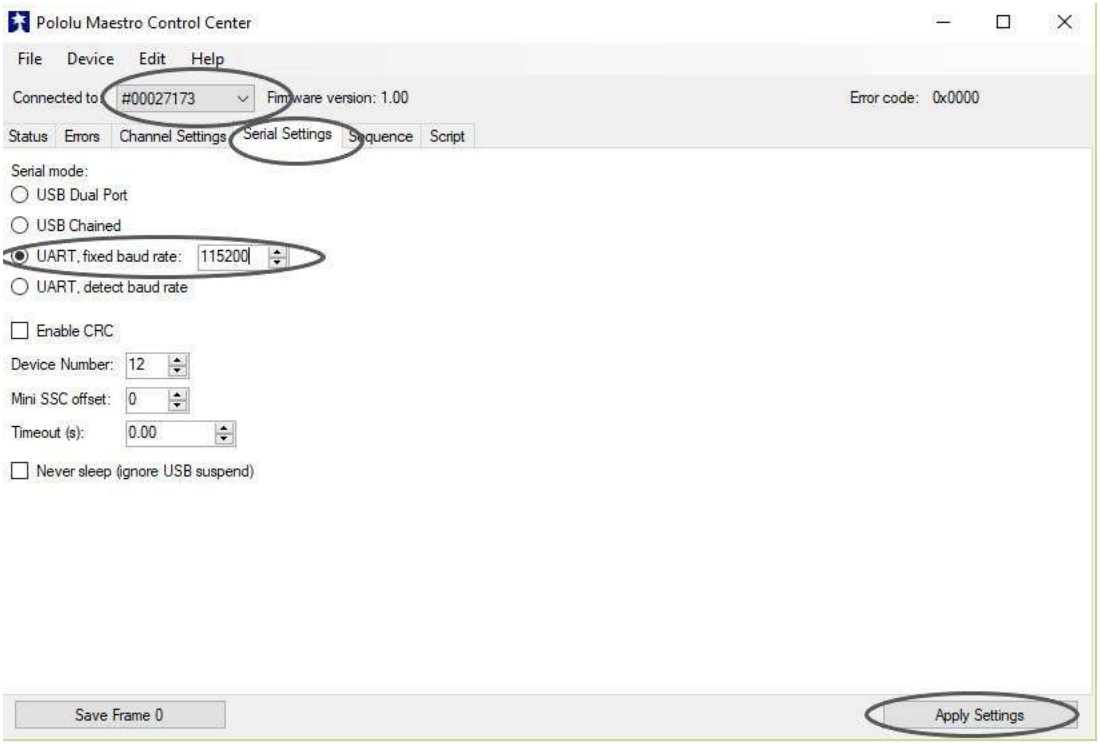

Figura 41 Configuraciones En Pololu Maestro Control Center.

En la Figura 42 se muestran los valores en milisegundos de correspondientes a la posición inicial de los diferentes servomotores del robot.

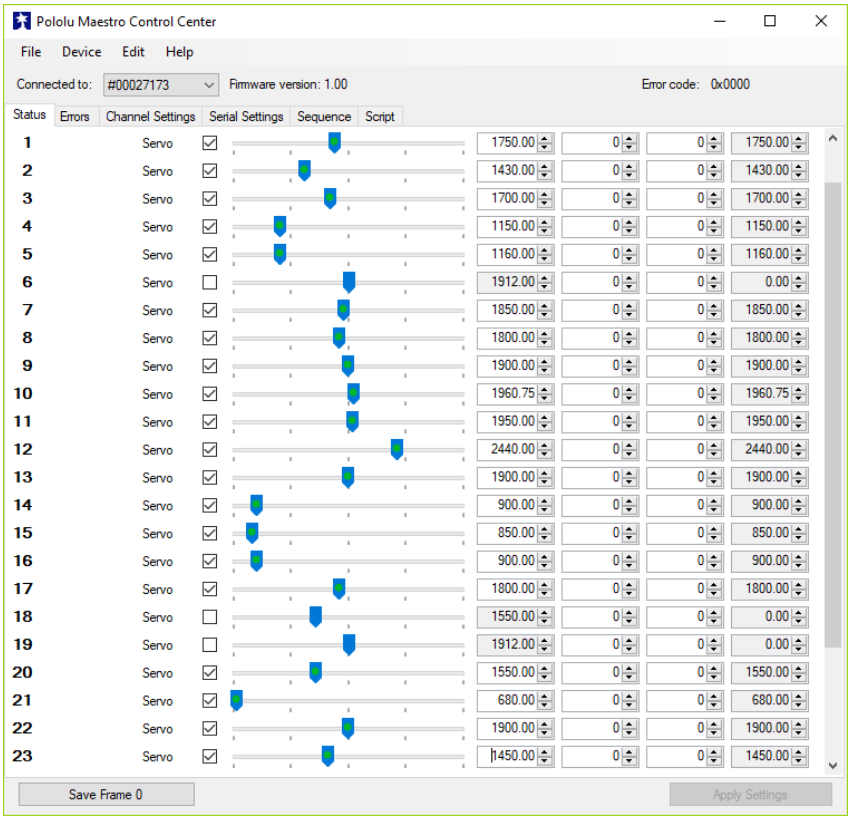

Figura 42 Obtención de las posiciones iniciales con el software de Pololu.

En la Figura 43se muestra cómo es que lucen las articulaciones del robot con estos valores de posición inicial

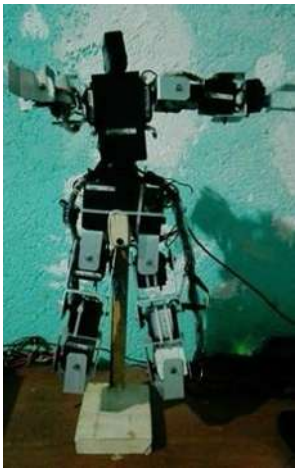

Figura 43 Posición inicial del Robot.

Y en la Tabla 7 los límites críticos (cuando una articulación del robot pega con otra y ello puede ocasionar sobre calentamiento y destrucción del servomotor afectado) y los límites físicos (posiciones límite de las articulaciones para un humano promedio).

Tabla 7 Limites en cicrosegundos de los servomotores, y posición inicial.

Número de canal	Límites críticos		Límites físicos		Valor inicial
	Min.	Max.	Min.	Max.	
1	1405	2700	1405	2100	1750
2	705	2495	705	1550	1430
3	1400	2440	1530	2050	1700
4	950	1700	1000	1100	1150
5	860	1460	860	1460	1160
6	656	3168	656	3168	1912
7	1320	3130	1320	3130	1850
8	1300	380	1600	3080	1800
9	1200	2450	1200	2130	1900
10	1630	2515	1630	2400	1950
11	656	3168	1550	2250	1950
12	800	3168	800	2500	2440
13	656	2720	656	2720	1900
14	660	1800 *(14)	660	1800 *(14)	900
15	750 *(15)	3168	890	3168	850
16	656	2660	790	2200 *(16)	900
17	656	3168	656	3168	1800
18	No aplica				
19					
20	1050	3168	1600 *(20)	3000	1550
21	660	2240	660	2240	680
22	656	2710	656	2710	1900
23	1050	2150	1050	2150	1450

ADVERTENCIAS. En la Tabla 7 se muestran unas advertencias en algunas canales marcadas con *, las cuales son:

*(14) revisar si se atasca el hardware.

*(15) valor puede llegar a ser 750 sii canal 16 es mayor a 2175.

*(16) canal 16 puede llegar a 2500 sii canal 15 es mayor a1300.

*(20) canal 20 puede llegar a 1200 sii canal 17 es menor a 2500.

4.3.3. Primeras pruebas de la tarjeta MEGA 2560 con la controladora Mini Maestro

En esta sección se describe cómo usar la tarjeta principal con la servocontroladora, para lo cual es necesario descargar la IDE oficial de Arduino, preferentemente la versión 1.8.3 o superior.

Al conectar la tarjeta a la computadora, en Windows, automáticamente se descargarán e instalaran los controladores de la tarjeta y se podrá usar, se deben hacer la selección del puerto COM correspondiente a la tarjeta MEGA, y la selección de la tarjeta como se muestra en la Figura 44.

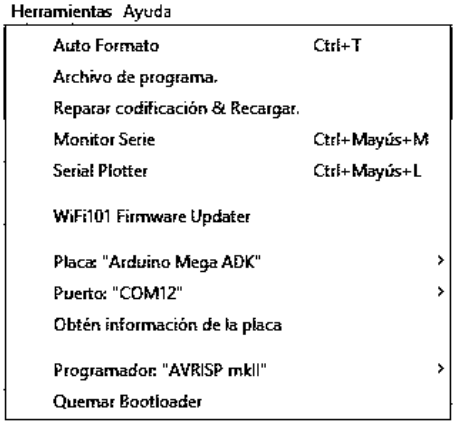

Figura 44 Selección del puerto y placa en la IDE.

RECOMENDACIÓN. Nótese que se seleccionó COM12, esto puede cambiar si se tienen más dispositivos conectados a la computadora, checar administrador de tareas para comprobar el puerto.

A continuación, hay dos caminos a seguir, enviar una secuencia especifica de control para un determinado protocolo de comunicación serie de los especificados por el fabricante en [20], y mandar carácter por carácter, o utilizar una biblioteca para Arduino proporcionada por el mismo fabricante. En este trabajo se utilizaron ambos caminos, el primero solo para realizar las primeras pruebas y explicar la trama, y para practicidad se implementaron algunas funciones de la biblioteca que hacen implementan el protocolo Pololu. El código desarrollado para esta sección se encuentra en el ANEXO A-4

Comandos de Protocolos

Para el control de la tarjeta servo controladora se usan los comandos serie. Si la tarjeta servo controladora está en “UART, detect baud rate”, primero se debe enviar el indicador de baudios con el byte 0xAA a la línea RX y después los comandos y datos necesarios. La comunicación

se logra mediante el envío de paquetes de comandos que constan de un byte de comando seguido de los bytes de datos que se requieren. Los bytes de comando tienen siempre los bits más significativos altos (128_255 decimal, 0x80_0xFF hexadecimal) mientras los menos significativos desactivados (0 y 127 decimal, 0x00_0x7F hexadecimal). Esto significa que cada byte de datos sólo puede llevar siete bits de información.

La tarjeta servo controladora identifica los protocolos Pololu, Compact y Mini-SSC en tiempo real y no es necesario usar parámetros para detectar el protocolo usado, por lo que se puede decidir entre uno y otro protocolo sin necesidad de configuración de registros en la tarjeta servo controladora

Protocolo Compacto. Es el más simple y compacto de los dos protocolos. El paquete con este protocolo es simple: Comando byte (con MSB alto), los bytes de datos necesarios, Por ejemplo, para ajustar el destino del servo de 0 a 1500 µs (0° a 60°), se deberá enviar la secuencia de bytes:

En hexadecimal: 0x84, 0x00, 0x70, 0x2E.
En decimal: 132, 0, 112, 46.

El byte 0x84 corresponde al comando Set Target, el primer byte de datos 0x00 es el número de servo y los dos últimos bytes de datos contienen el destino en cuartos de micro segundos.

Protocolo Pololu. Este protocolo es compatible con los usados por otros dispositivos controladores de servomotores y es el que se ha implementado en este trabajo ya que en este protocolo se envía un identificador antes de enviar algún otro dato. Como tal, se puede encadenar a más controladores o dispositivos serie en una sola línea y con este Protocolo enviar comandos específicamente al controlador deseado sin confundir a los otros dispositivos de la línea. Para utilizar el protocolo Pololu, se debe transmitir 0xAA (170 decimal) como primer byte (comando) seguido de un byte de datos con el número de dispositivo. El número de dispositivo predeterminado para la tarjeta servo controladora es 12, pero es un parámetro que puede ser cambiado. Cualquier tarjeta en la línea cuyo número de dispositivo coincida con el especificado aceptará el comando, todos los demás dispositivos lo ignorarán. Los bytes restantes en el paquete de comandos son los mismos que los de un paquete de protocolo compacto y que se enviarán, con una diferencia clave: el byte de protocolo de comando compacto es ahora un byte de datos para el comando 0xAA. Por lo tanto, el paquete de comandos es: 0xAA, byte número de dispositivo, byte comando con MSB (More Significant Bite) desactivado, bytes precisos de datos.

Por ejemplo, lo mismo que antes para la tarjeta servo controladora con el número de dispositivo 12, se enviará la secuencia:

En hexadecimal: 0xAA, 0x0C, 0x04, 0x00, 0x70, 0x2E.

En decimal: 170, 12, 4, 0, 112, 46.

De modo que el 0x04 es el byte 0x84 haciendo la analogía con el protocolo compacto. Protocolo Mini SSC Este protocolo permite el control de hasta 254 servos diferentes con el encadenado de tarjetas controladoras. Solo es necesario tres bytes para ajustar el destino del servo, por lo que es un buen protocolo si se requiere enviar comandos de forma rápida.

Protocolo Mini SSC. se transmite un 0xFF (255 en decimal) como primer byte de comando seguido del byte de numero de servo y otro de bits con el destino a realizar. El paquete sería el siguiente: 0xFF, byte número de servo, byte destino servo.

Por ejemplo: si se busca ajustar el destino del servo 0 a la posición neutral, se debe enviar la secuencia siguiente:

En hexadecimal: 0xFF, 0x00, 0x7F.

En decimal: 255, 0, 127.

Por lo que puede usarse este protocolo para configuraciones donde se requiere varios servomotores, de modo que estos pueden ser interconectados en bloques contiguos de servos numerados de 0 a 254.

Comandos para los servomotores

La tarjeta servo controladora tiene varios comandos para ajustar el canal, tomar la posición actual y ajustar límites de velocidad y aceleración, como son: Set Target, Set Speed, Set Aceleration, Get Position Set Target (Protocolo compacto / Pololu) Este comando sirve para la inicialización de la tarjeta servo controladora y varía dependiendo el protocolo usado.

Protocolo compacto: 0x84, número de canal, destino bits bajos (LB), destino bits altos (HB).

Protocolo Pololu: 0xAA, número de dispositivo, 0x04, número de canal, destino bits bajos, destino bits altos

Los 7 bits bajos del 3er byte representan el destino, los 7 bits bajos del cuarto byte de datos representan el destino. El destino no es entero negativo. Si el canal está configurado para servo, el destino representa el ancho de pulso a transmitir en cuartos de microsegundo. El valor destino 0 indica a la tarjeta servo controladora que pare el envío de pulsos al servo. Si el canal está configurado como salida digital los valores menores de 6000 indican que debe poner la línea en bajo, en cambio sí son mayores la línea estará en alto [20].

Por ejemplo, si el canal 2 está configurado para servo y se desea ajustar el destino a 1500 µs (1500x4 = 6000 = 01011101110000 en binario), se enviará la siguiente secuencia de bytes:

En binario: 10000100, 00000010, 01110000, 00101110
En hexadecimal: 0x84, 0x02, 0x70, 0x2E
En decimal: 132, 2, 112, 46

Set Speed. Este Comando limita la velocidad del servo al cambiar el valor del canal de salida. El límite de velocidad se da en unidades de 0.25 µs/10 ms.

Protocolo compacto: 0x87, número de canal, velocidad LB, velocidad HB.
Protocolo Pololu: 0xAA, número de dispositivo, 0x07, número de canal, velocidad bits bajos, velocidad bits altos.

Por ejemplo, el comando 0x87,0x05,0x0C,0x01 establece la velocidad del canal 5 en un valor de 140, que corresponde a una velocidad de 3.5 µs/ms (140x0.25 µs/10 ms). Esto significa que, si envía un comando de Set Target para ajustar el destino del servo de 1000 µs a 1350 µs, tardará 100 ms ((1350µs-1000µs) /3:5 µs/ms)) para realizar ese ajuste. Una velocidad de 0 es una velocidad ilimitada, así el objetivo será inmediato para llegar a la posición. Tomando en cuenta que la velocidad real de los movimientos cinemáticos también está limitada por el propio diseño del servo, la tensión de alimentación y de la carga, por lo que este parámetro no servirá para ir más rápido de lo que físicamente es capaz. Con la configuración de velocidad mínima de 1, la salida de servo tarda 4 segundos ((2000µs-1000µs) = (1x0:25µs/10)) para pasar de 1 a 2 ms (0° a 180°). La configuración de velocidad no tiene ningún efecto en los canales configurados como entradas o salidas digitales [20].

Set Acceleration. Este comando limita la aceleración en la salida del canal servo. La aceleración puede tener un valor entre 0 y 255 en unidades de (0.25 µs) / (80 ms) / (10 ms). El valor 0 corresponde ningún límite de aceleración. La aceleración limitada causa que la velocidad del servo ascienda lentamente hasta alcanzar la velocidad máxima para luego descender hasta llegar a la posición destino, resultando de ello, un movimiento relativamente suave de un punto a otro. Con la aplicación de límites en velocidad y aceleración, solo harán falta unos pocos ajustes para llegar al destino de posición deseado con movimientos de aspecto natural y que de lo contrario serían bastante más complicados de crear [20].

Protocolo compacto: 0x89, número de canal, aceleración bits bajos, aceleración bits altos.
Protocolo Pololu: 0xAA, número de dispositivo, 0x09, número de canal, aceleración bits bajos, aceleración bits altos.

Configurando la aceleración mínima a 1, el servo tarda 3.2 segundos (((2000µs - 1000µs) / (1 * 0.25µs/80ms)) /10ms) en moverse desde una posición de 1ms a un destino de 2ms (0° a 180°). La aceleración no tiene efecto en los canales configurados como entradas o salidas digitales [20].

Get Position Con este comando se obtiene la posición en la cual se encuentra un servomotor partiendo de la señal generada por la tarjeta servo controladora.

Protocolo compacto: 0x90, número canal.

Protocolo Pololu: 0xAA, número de dispositivo, 0x10, número de canal.

Respuesta: un byte LB de posición y un byte HB de posición ambos en 8 bits. El comando pregunta al dispositivo que se comunica con la tarjeta servocontroladora por la posición en que se encuentra el canal [20].

El valor se envía de inmediato en dos bytes tras haber recibido el comando. Si el canal especificado esta como servo el valor representa el ancho de pulso que la tarjeta servo controladora está emitiendo por el canal, reflejando los efectos de comandos previos como velocidad y en la placa. Si el canal esta configurados como salida digital y el valor es menor de 6000, la línea está en bajo, si el valor es mayor de 6000 es que la línea está en alto. Si el canal está configurado como entrada, la posición representa el voltaje medido en el canal [20].

Los canales 0-11 son analógicos y el rango de valores será entre 0 y 1023 para voltajes de 0 a 5V. Las entradas en los canales 12-23 son digitales y sus valores serán de 0 o 1023 exactamente. Se debe tomar en cuenta que el formato de este comando difiere del destino, velocidad y aceleración dado que no existe ninguna restricción sobre el bit alto, la posición es formateada como un entero sin signo estándar dos bytes primero el bajo luego el alto. Por ejemplo, a la posición 2567 tendría como respuesta de 0x07, 0x0A [20]..

4.3.4. Software de la tarjeta principal para recibir e interpretar la trama recibida vía bluetooth y controlar los servomotores

La Máquina, en este caso el robot humanoide, como ya se mencionó está conformado por varios elementos, entre los que se encuentran la tarjeta principal Arduino MEGA 2560, los servocontroladores MG995 y la tarjeta servocontroladora Mini Maestro 24 Canales, además del módulo Bluetooth HC-06. Ahora bien, para conjuntar todos esos elementos hace falta desarrollar un software, en este trabajo fue un código en la IDE de Arduino, que sea capaz de hacer que la máquina cumpla el objetivo del trabajo, el ser manipulada a través de una Interfaz vía inalámbrica, de la que ya se escribió.

Este programa se encarga de recibir información e interpretar loso comandos de la PC, después de que la interfaz y el algoritmo programado determino el ángulo de las articulaciones una vez que se detectó una postura y se rastrearon sus articulaciones,

La información que recibe la tarjeta principal es el comando de ‘pos’ que indica que se requiere mover la posición mediante el comando set Target de la biblioteca de Pololu para Arduino, un numero para identificar a cada servomotor y un valor que corresponde el ángulo con de cada articulación, dicho ángulo no se recibe en grados, sino en unidades de tiempo del ancho del pulso que se requiere para llevar un determinado servomotor a una determinada posición, además dicho tiempo del ancho del pulso se debe recibir multiplicado por un factor cuatro, pues así lo requieren las funciones de la biblioteca. El programa en la tarjeta interpreta toda esta información recibida y envía instrucciones a la tarjeta servocontroladora para que esta cumpla su función de controlar el pulso de salida de cada canal correspondiente a cada servomotor.

Para la interpretación y debida conexión de la tarjeta principal con la PC, primero debe de estar corriendo el programa en la tarjeta principal, para posteriormente lanzar el Script en el Anexo A-3, el flujo del programa desarrollado para la tarjeta principal se muestra en la Figura 45, y el código se encuentra en el Anexo A -6.

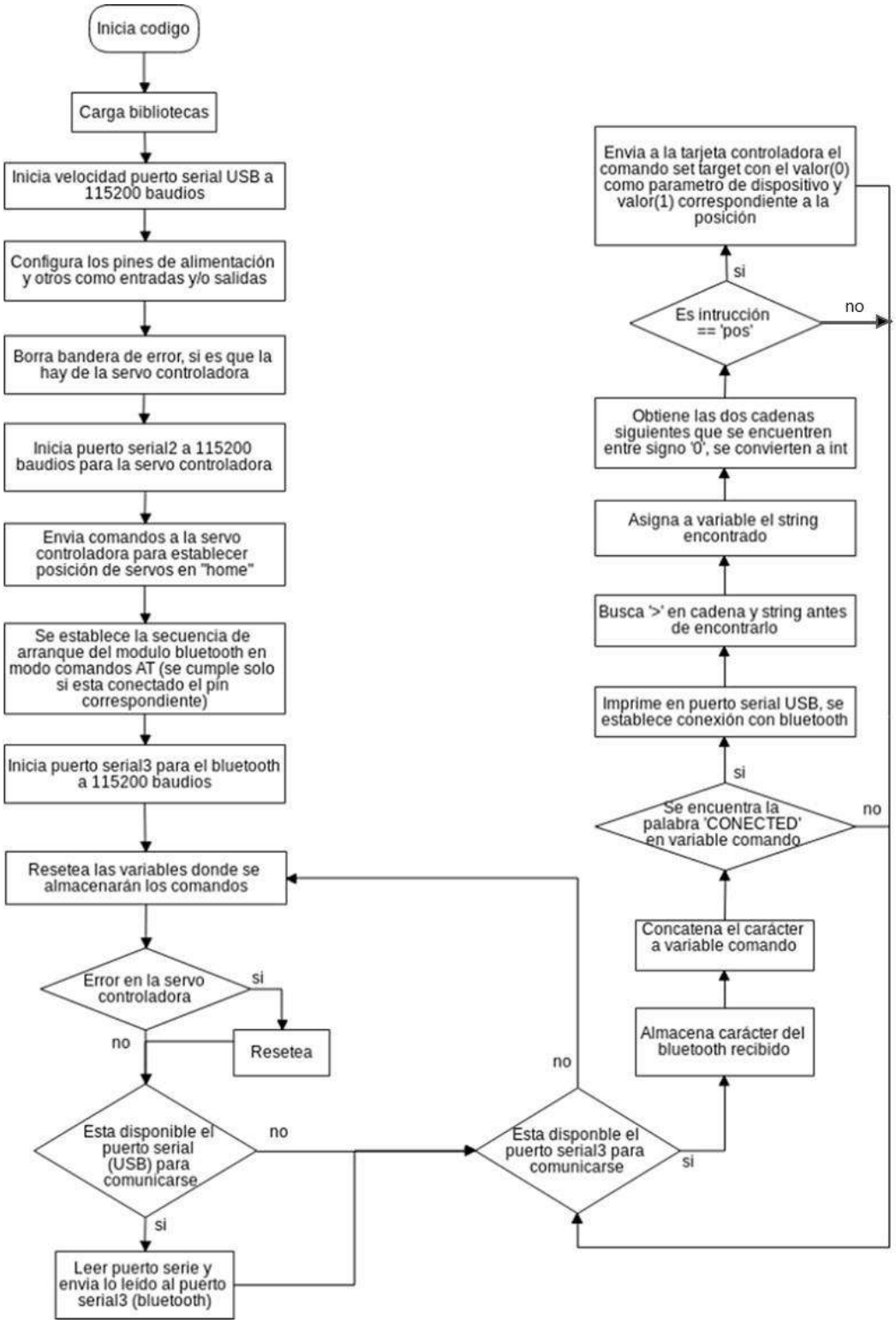

Figura 45 Conexión de la tarjeta MEGA con Matlab vía bluetooth, y detección de la trama recibida para controlar un servomotor

4.4. Conclusiones del capítulo

Como se mencionó anteriormente, existen diferentes entornos para desarrollar software en una PC, y su elección depende de los elementos de hardware que se van a implementar, o de las características que se requieran hablando de software, como facilidad para programar, soporte, bibliotecas adicionales, poder de procesamiento, y herramientas extra que ofrezcan.

En el desarrollo de software una de los principales inconvenientes que se tuvo fue la utilización de las funciones que Matlab provee para utilizar dispositivos Bluetooth conectados a la computadora, pues tardaban mucho tiempo en enviar cadenas de datos con comandos hacía la tarjeta principal del robot humanoide o Máquina, aun configurado a una velocidad muy alta de 115200 baudios. Que, si bien para cadenas cortas el tiempo que tardaba en enviarlas no era muy grande, este se incrementaba considerablemente cuando las cadenas con las instrucciones incrementaban su longitud

Por último, poniendo en comparativa el Script desarrollado en Matlab para una implementación final de todos los elementos que conforman la Interfaz, con la GUI desarrollada, se observó que el Script tiene tiempo de ejecución mucho más rápido que la GUI, que si bien el código implementado para su desarrollo es muy extenso (ya que se definen funciones para queda objeto como botones, graficadores, selectores, pseudoconsolas) tiene un núcleo o algoritmo para la detección de posturas y envio de instrucciones a la Máquina muy similar al Script.

Capítulo 5
Pruebas y resultados

5.1. Introducción a las Pruebas realizadas para llegar a una IHM funcional y obtener resultados.

En este capítulo se muestran las pruebas realizadas durante el proceso de desarrollo de la IHM, desde los resultados obtenidos por las IDE utilizadas (Matlab, Mini Maestro 24 control center, y Arduino 1.8.3) divididas de la misma forma en que se ha estado trabajando pruebas de la Interfaz y de la Máquina, y finalmente se muestran pruebas y los resultados de la Interfaz Humano Máquina final.

5.2. Pruebas en la Interfaz previas

Se presentan las pruebas de la Interfaz del software desarrollado en Matlab, desde las primeras detecciones de posturas con el Kinect y las pruebas para obtención de ángulo de todas las articulaciones.

5.2.1. Primeras capturas y pruebas con Kinect

En esta sección se muestran algunas capturas realizadas con el Kinect, después de que se cumplieron con los requisitos de controladores y herramientas antes mencionados. A continuación, se muestran los resultados de emplear diferentes fuentes de video.

En las capturas siguientes se utilizó el de color del Kinect, con configuración de Cámara Infrarroja con 640x480, configuraciones RGB con una resolución de 1280x980 pixeles, RGB con resolución de 640x480RawBayer con resolución de 1280x980, RawYUV a 640x480, YUV a 640x480.

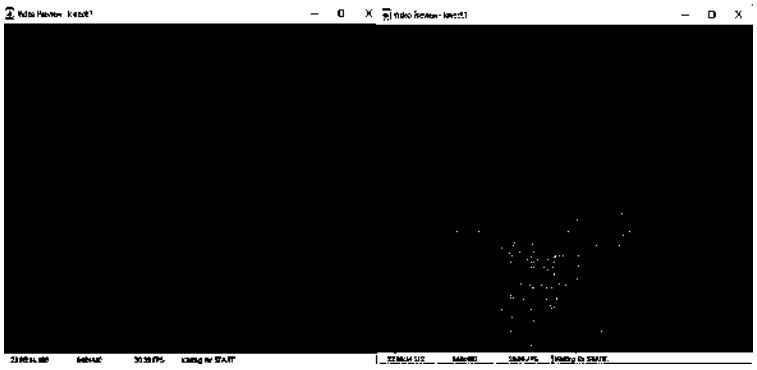

Figura 46 Cámara infrarroja a 640x480 pixeles. 1) 2 metros de distancia. 2) 1 metro

En la Figura 46.1 se muestra una captura con una persona colocada a 2 metros de distancia frente al Kinect, y en la Figura 46.2 colocada a 1 metro de distancia, se puede observar que la imagen se ve obscura aun cuando la captura fue realizada con buenas condiciones de luz visible. Pero la cámara infrarroja funciona bajo el principio de emitir haces de luz infrarroja la cual no es visible para el ojo humano, pero si para una cámara digital convencional, como lo es el caso del Kinect, por esta razón al estar la persona situada más cerca del Kinect se le puede observar más iluminada, pues la infrarroja incide sobre ella con más potencia

Figura 47 Sensor Color, 1) RGB 1280x980, 2) RawBayer 640x480, 3) RawYUV 640X480

Y finalmente el sensor de profundidad Depth con resolución de 640x480, 320x240 y 80x60 pixeles

Figura 48 Sensor de profundidad, 1) 640x480 lejos, 2) 640x480 cerca, 3) 320x240 lejos, 4) 80x60 lejos

5.2.2. Primeras pruebas para detección de posturas

Las pruebas realizadas consistieron en la utilización del sensor de profundidad del Kinect y utilizando algunas de las funciones del Toolbox de Matlab para rastrear las articulaciones del cuerpo.

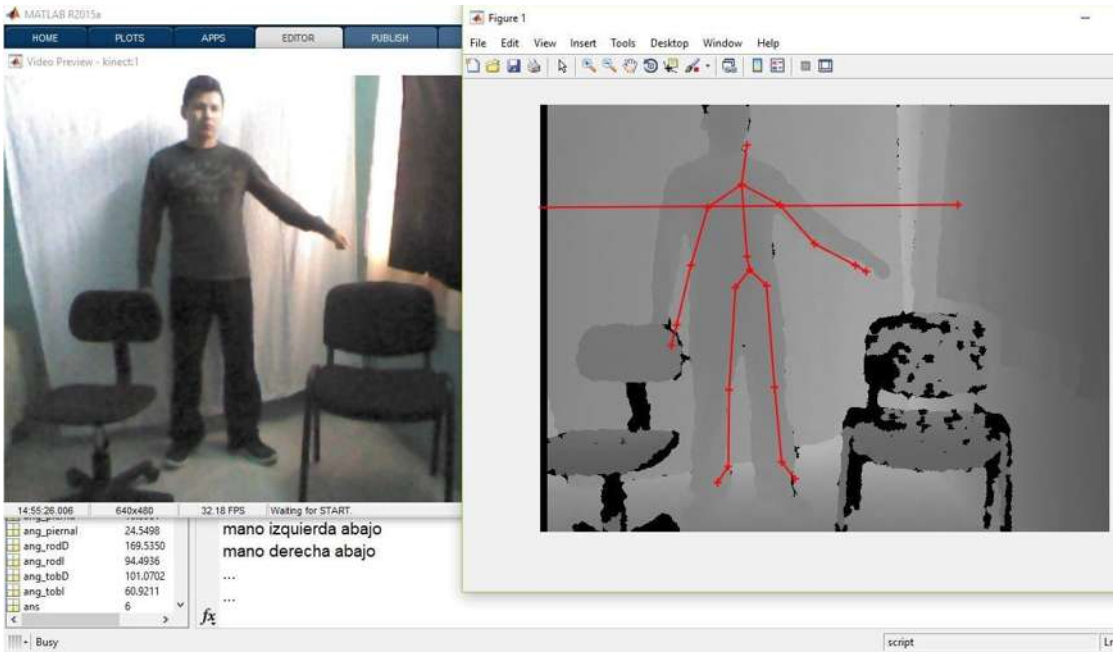

Figura 49 Rastreo con obstáculos y manos abajo.

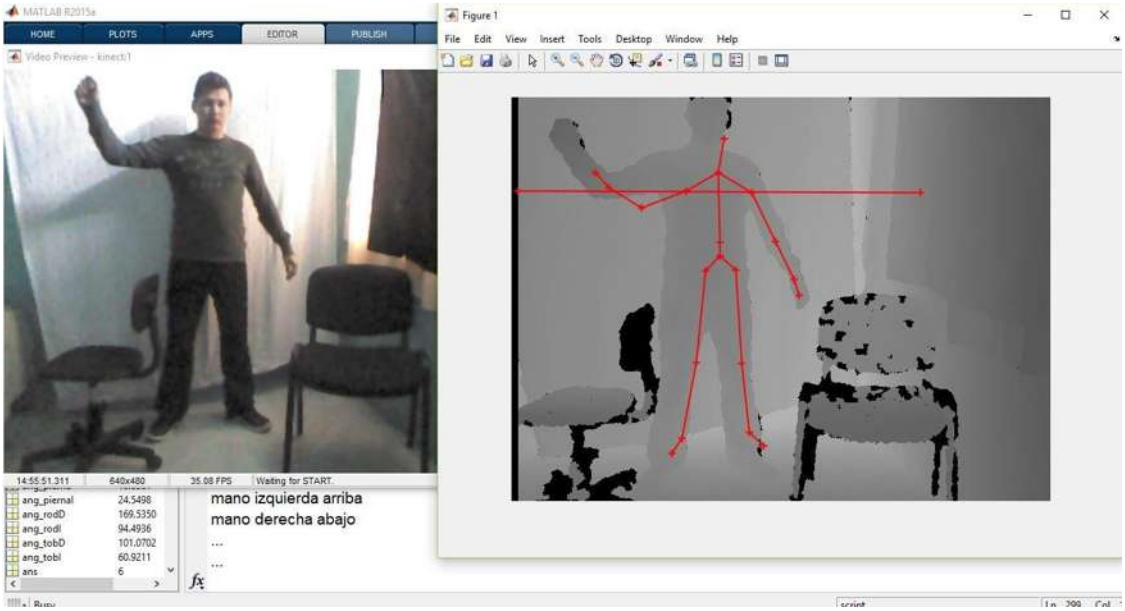

Figura 50 Rastreo de postura con obstáculos y mano izquierda arrida

En la Figura 49 y Figura 50 se muestran algunas capturas con de rastreo de las extremidades principales de una persona con obstáculos (sillas), adicional a esto se trabajó con las coordenadas de cada articulación, utilizando sentencias condicionales en Matlab para determinar si la persona tenia los brazos por arriba o por debajo de una línea horizontal imaginaria a la altura de los hombros. En ambas figuras se alcanza a apreciar el texto que indica la ubicación de cada mano.

5.2.3. Detección de ángulos en las articulaciones de una persona

Las pruebas realizadas consistieron en determinar que ángulo tiene cada articulación (axilas, codos, rodillas, piernas y cuello) con respecto a dos extremidades como referencia.

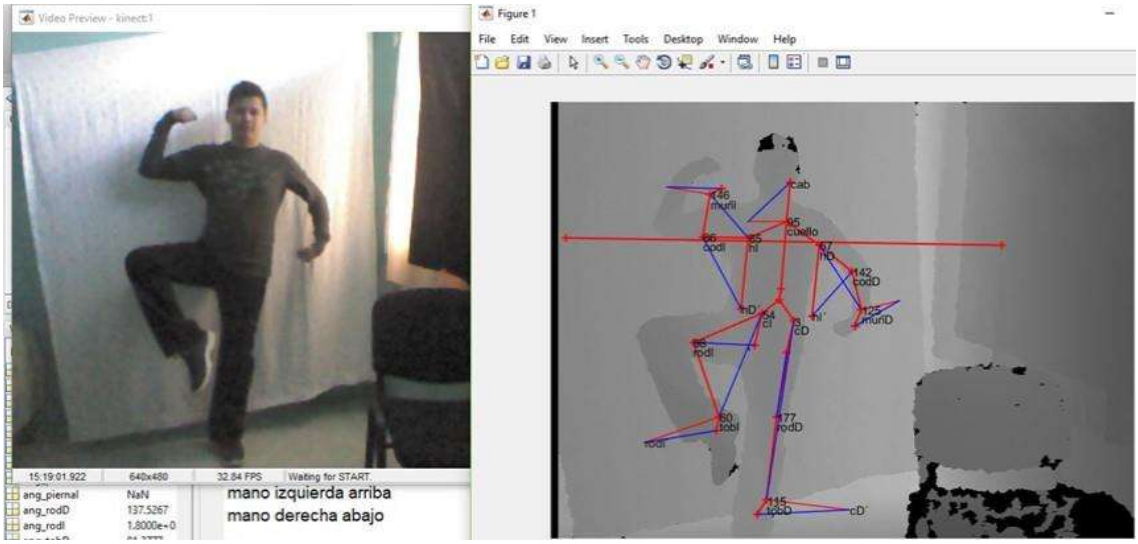

Figura 51 Detección de ángulos personan 1 de sexo masculino, y detección de posición de brazos

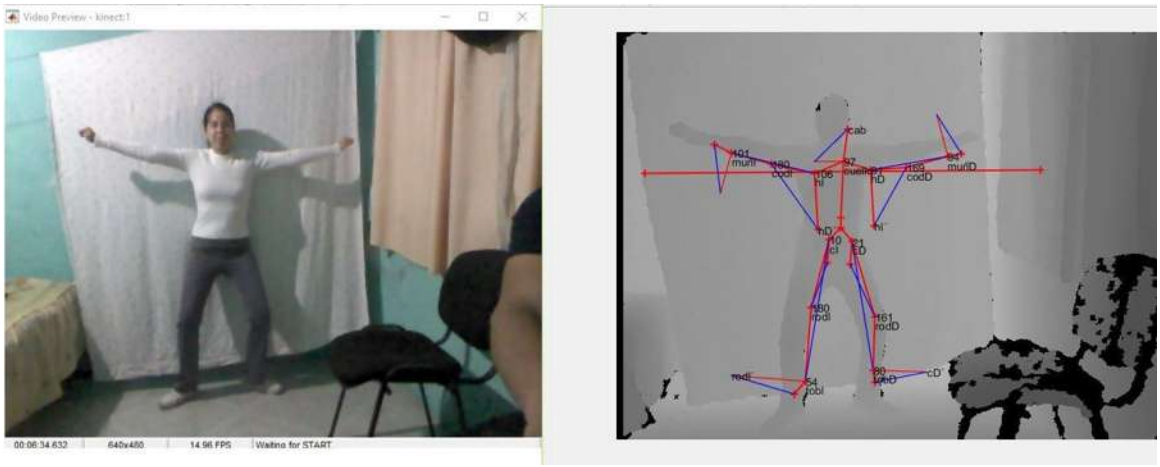

Figura 52 Detección de ángulos persona 2 de sexo femenino.

En la Figura 51 se muestra un ejemplo de detección de ángulos de una persona de sexo masculino, y en la Figura 52, ambas con una silla en la escena, a continuación, en la se muestran los ángulos calculado a partir de las coordenadas de las articulaciones y extremidades de la persona, utilizando ley de senos y ley de cosenos, además también se muestra que el sistema detectó la mano izquierda arriba y la mano derecha abajo.

Tabla 8 Ángulos de las articulaciones de la Persona 1.

Articulación Persona 1	Ángulo en grados
Cuello	95
Axila Izquierda	85
Codo Izquierdo	86
Muñeca Izquierda	146
Axila Derecha	57
Codo Derecho	142
Muñeca Derecha	125
Pierna Izquierda	54
Rodilla Izquierda	98
Tobillo Izquierdo	60
Pierna Derecha	3
Rodilla Derecha	177
Tobillo Derecho	114

Tabla 9 Ángulos de las articulaciones de la Persona 2.

Articulación Persona 2	Ángulo en grados
Cuello	97
Axila Izquierda	106
Codo Izquierdo	180
Muñeca Izquierda	101
Axila Derecha	91
Codo Derecho	169
Muñeca Derecha	84
Pierna Izquierda	10
Rodilla Izquierda	180
Tobillo Izquierdo	64
Pierna Derecha	21
Rodilla Derecha	161
Tobillo Derecho	80

Como se observa en la Figura 51 y Figura 52, y los ángulos obtenidos en las tabla Tabla 8 y Tabla 9, el algoritmo de detección de posturas y cálculo de ángulos resulta ser muy eficaz, si bien algunas de las líneas trazadas correspondientes a algunas extremidades no están exactamente situadas sobre lo que sería la persona esto es debido al trazado de las mismas, más no representa problema para la detección de la postura de las personas ni de sus ángulos.

5.3. Pruebas a la Máquina

Las pruebas realizadas en la Máquina incluyen las configuraciones y respuestas obtenidas en los comandos AT, el envío de comandos desde la tarjeta principal a la servocontroladora y la respuesta obtenida.

A continuación, se muestran las pruebas realizadas desde la herramienta "Monitor Serial" que proporciona la IDE de Arduino 1.8.3, la cual tiene la función de ser un monitor del puerto serie de las placas de Arduino cuando se conectan vía USB a la computadora. El código implementado se puede ver en el ANEXO A-1. Y la respuesta en la Figura 53.

5.3.1. Pruebas con comandos AT

Los comandos AT son cadenas de texto que se envían al módulo bluetooth vía puerto serie, y son como los que se muestran en la Figura 53.

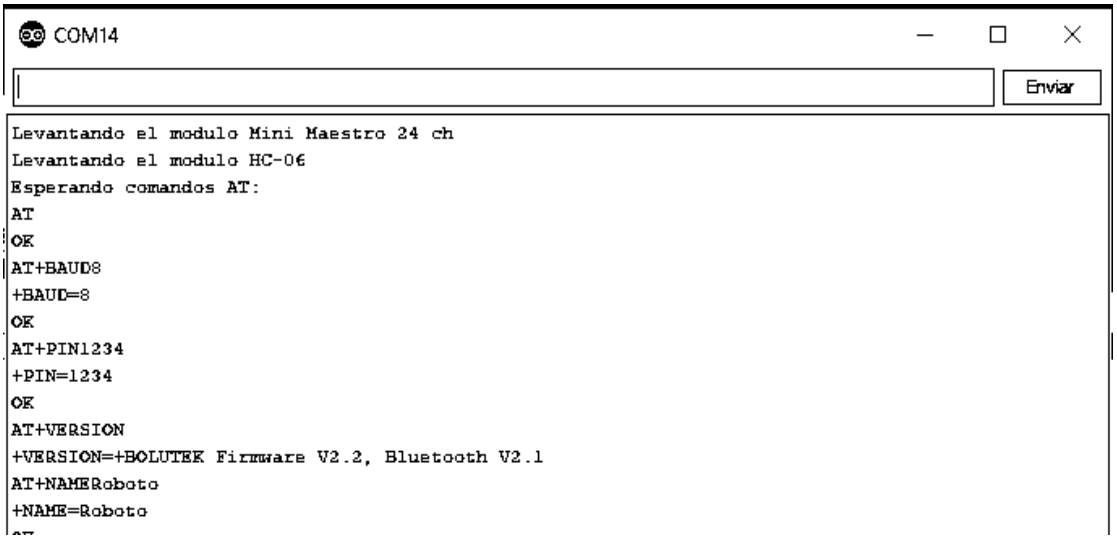

Figura 53 Comandos AT en Monitor Serial de Arduino.

Mediante los comandos Serial.write, Serial.print se pueden escribir información al puerto serie, la primera solo admite escritura en formato byte a byte, y la segunda en formato String con codificación ASCII, y Serial.read

Los comandos enviados fueron “AT”, cuya respuesta fue “OK”, esto indica que se tiene configurada correctamente la velocidad del Puerto Serial3 que es el usa el módulo HC.06. Si se recibieran caracteres basura se debería cambiar la velocidad, una forma como ya se mencionó es entrar en modo AT forzado a una velocidad de 37400 baudios, siguiendo la secuencia del código del Anexo A-5, para utilizarla es necesario soldar un cable a la terminal KEY o ENA del módulo.

Se envío el comando AT+BAUD68 correspondiente a una velocidad de 115200, se recibió como respuesta “+BAUD=8” y “OK” ello indica que ya estaba trabajando a esta velocidad pues ya había sido configurado previamente.

El comando AT+PIN1234 configura como contraseña o pin de seguridad el código 1234, entonces cuando se requiera hacer el emparejamiento con otro dispositivo Bluetooth, en este trabajo por ejemplo cuando se emparejo con la PC este PIN fue solicitado. La respuesta de este comando fue “+PIN=1234” Y “OK”.

Otro de los comandos enviados fue “AT+VERSION” cuya respuesta se puede observar en la Figura 53. Que muestra la versión del firmware 2.2 y del bluetooth 2.1.

El último comando fue “AT+NAMERoboto” para asignarle el nombre de Roboto al dispositivo. La respuesta obtenida fue “+NAME=Roboto”

Como se observa una vez teniendo conectado correctamente el dispositivo y habiendo configurado la velocidad del dispositivo, enviar y hacer configuraciones por medio de los comandos AT resulta muy sencillo.

5.3.2. Pruebas de comandos de la tarjeta principal a la tarjeta servo controladora

Estas pruebas consisten en el envío de un comando haciendo uso de la sintaxis del protocolo Pololu, cuya secuencia se puede ver en la Figura 54, para lo cual se usó un osciloscopio. El comando consistió en indicar controlar la posición del canal 23 en 1450us.

El comando enviado fue

En hexadecimal: 0xAA, 0x17, 0x04, 0x00, 0xAA, 0x05.

En decimal: 170, 23, 4, 0, 170, 5.

En la Figura 54 se muestra el ancho del pulso generado. Cuya salida del canal 23 es un ancho de 1450us, y se observa en la Figura 55

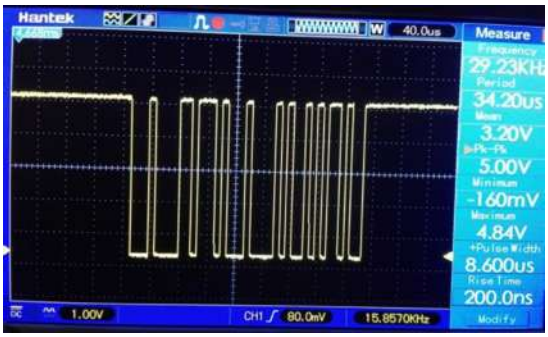

Figura 54 Envío de posición 1450 us al canal 23.

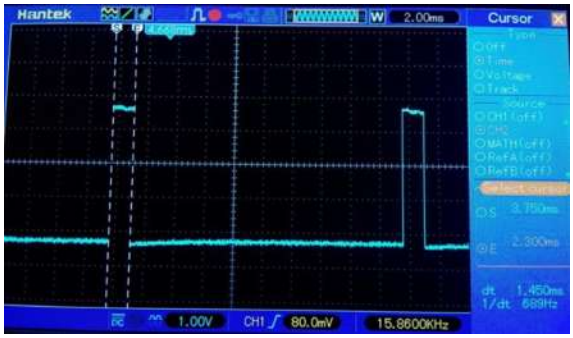

Figura 55 Respuesta del canal 23.

5.4. Pruebas y resultados de la Interfaz Humano Máquina

En esta sección se presentan las pruebas y resultados obtenidos del sistema Interfaz Humano Máquina desarrolla. Las cuales se dividen en resultados obtenidos por el Script en Matlab (sin interfaz gráfica) y mediante la interfaz de Usuario Grafica o GUI.

El script es un programa que se realiza en Matlab, el cual se ejecuta línea a línea desde el inicio al fin, siguiendo estructuras de programación y sentencias muy similares al lenguaje C.

5.4.1. Resultados de la IHM con el Script en Matlab

A continuación, en la Figura 56, se muestra ocho secuencias del sistema funcionando, en las cuales se hicieron las pruebas a una persona de sexo femenino, con ropa que no es muy holgada o gruesa, a una distancia de aproa 2 metros, a la izquierda se encuentra el Humano, al centro la Interfaz y a la derecha la Máquina.

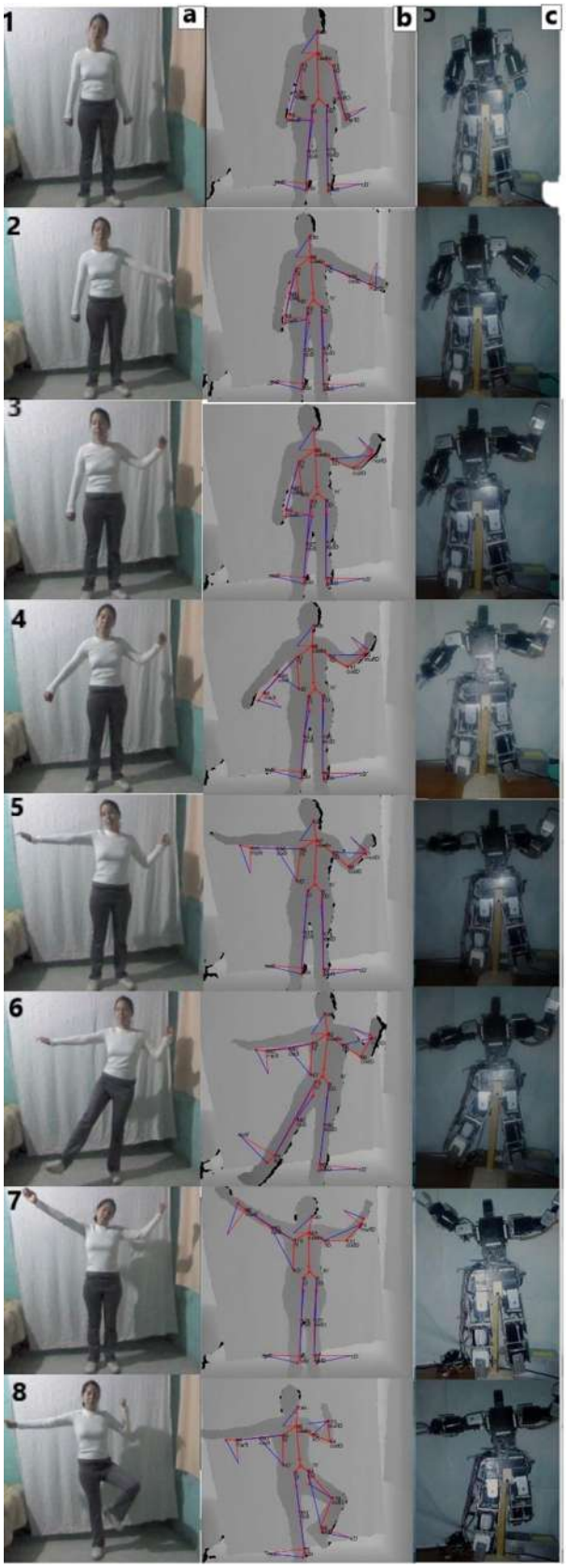
a
b
c
1
2
3
4
5
6
7
8

Figura 56 Secuencia de detección IHM

En la Figura 56, se muestra una secuencia de 8 movimientos realizada por una persona, en el centro la detección y rastreo de las extremidades usando el sensor de profundidad del Kinect, y en la derecha se muestra la respuesta del para las capturas de movimientos.

Adicional a esto se realizó una prueba de rendimiento para calcular el tiempo que tarda el programa desde que detecta una persona, realiza el rastreo de sus extremidades y hace el procesamiento de la misma, para lo cual se utilizó la función tic toc de Matlab el cual rondaba de entre 0.092 y 0.125 segundos.

5.5. Resultados de la captura de posiciones con la Interfaz Gráfica de Usuario

En la Figura 57 se muestra la Interfaz gráfica desarrollada, con el ejemplo de detección de una persona.

Figura 57 Interfaz Gráfica de Usuario con detección.

En la Figura 57 se puede observar en la terminal de comandos que fue seleccionada la opción trigger manual en modo de detección, y en el panel de detección se observa mano izquierda arriba y mano derecha abajo.

Para calcular el tiempo que tarda en realizar cada detección y hacer el procesamiento de la misma rondaba de entre 0.092 y 0.125 segundos

.

5.6. Conclusiones del capítulo.

Esta IHM desarrollada se caracteriza por su gran simplicidad, el usuario solo tiene que saber que debe pararse frente al Kinect, a una distancia apropiada (aprox. 2 metros), en una habitación con luz moderada y que no tenga objetos grandes o muebles que se interpongan entre los sensores del Kinect y él, y será suficiente con ejecutar la GUI desarrollada y con configuraciones básicas e intuitivas le será posible empezar a manipular el Robot, pues una vez hecho lo anterior ahora solo bastará con moverse naturalmente y el sistema se encargará del resto.

Se observó que el tiempo de ejecución del programa (Script) fue mucho menor que el tiempo de la Interfaz Gráfica de Usuario. Específicamente la sección desde donde swe detecta una posición, se procesa la información y se hacen los cálculos necesarios para obtener los ángulos de todas las articulaciones del esqueleto detectado de un Humano.

Capítulo 6
Conclusiones

El presente trabajo, “CAPTURA DE MOVIMIENTOS CON KINECT PARA MANIPULAR UN ROBOT HUMANOIDE” se concluye diciendo que se cumplieron todos los objetivos planteados en el inicio del trabajo, pues se logró a desarrollar una Interfaz Humano Máquina, la cual está constituida por más de un sensor (el módulo Kinect), una computadora que procesa la información adquirida por el sensor, las acciones de control que determina son comandos para manipular la posición de los servomotores de una Máquina (robot humanoide).

EL sistema final Interfaz Humano Maquina resulto ser bastante eficiente en la detección de posturas de personas, como se observó el tiempo de detección fue muy reducido a pesar de la gran cantidad de cálculos realizados por la computadora. Además, cabe mencionar su gran simplicidad de uso, pues el usuario solo tiene que saber que debe pararse frente al Kinect, a una distancia apropiada (aprox. 2 metros), en una habitación con luz moderada y que no tenga objetos grandes o muebles que se interpongan entre los sensores del Kinect y él, y será suficiente con ejecutar la GUI desarrollada y con configuraciones básicas e intuitivas le será posible empezar a manipular el Robot, pues una vez hecho lo anterior ahora solo bastará con moverse naturalmente y el sistema se encargará del resto.

Cabe mencionar los principales cambios del hardware del robot, los cuales fueron mejorar el cableado, etiquetado del mismo, sustitución de servomotores dañados, montaje en una base para facilidad de uso, no se utilizó el acelerómetro giroscopio, y se cambió la tarjeta principal por sus nuevas prestaciones, por todo esto se tiene un hardware del robot con disponibilidad de agregar más módulos, como una pantalla compatible con la tarjeta principal, disponible en el mercado, botones para el control directo de sus articulaciones, o posteriormente combinar el control lógico difuso que implementaba con la captura de posiciones humanas y manipulación de articulaciones.

Trabajos futuros

Para trabajos futuros se contempla:

-Incorporar una Pantalla al robot y control de posiciones local.

-Reincorporar el Sensor acelerómetro/giroscopio al robot.

-Implementar un control de equilibrio en conjunto con detección y replica de posturas en el robot.

-Aumentar la velocidad de transmisión de datos vía inalámbrica.

-Utilizar una Mini computadora e incorporarla al robot para el procesamiento de la información del Kinect.

Referencias

[1] © Ing. Punzenberger COPA-DATA GmbH, «COPADATA,» © Ing. Punzenberger COPA-DATA GmbH, [En línea]. Available: https://www.copadata.com/es/soluciones-hmi-scada/interfaz-hombre-maquina-hmi/. [Último acceso: 19 Enero 2018].

[2] Noticias De La Ciencia, «www.noticiasdelaciencia.com,» NCYT Amazing, 28 Diciembre 2015. [En línea]. Available: http://noticiasdelaciencia.com/not/17583/innovadora-interfaz-hombre-maquina-en-forma-de-guante/. [Último acceso: 20 Enero 2018].

[3] (Amazings/NCYT), «Noticias De La Ciencia y Tecnología,» 25 Septiembre 2014. [En línea]. Available: http://noticiasdelaciencia.com/not/5233/exoesqueleto-controlado-con-la-mente-para-personas-con-ciertas-discapacidades/. [Último acceso: 20 Enero 2018].

[4] B. Bishop, «THE VERGE,» © 2018 Vox Media, Inc. All Rights Reserved, 25 Diciembre 2011. [En línea]. Available: https://www.theverge.com/2011/12/25/2659309/kinect-weighs-astronauts-sight-space. [Último acceso: 20 Enero 2018].

[5] MARTINPIXEL, «XATACA MEXICO,» WSL WeblogsSL, 27 Julio 2017. [En línea]. Available: https://www.xataka.com.mx/ciencia/roki-un-exoesqueleto-creado-en-mexico-para-ayudar-a-las-personas-para-que-puedan-volver-a-caminar. [Último acceso: 10 Novienbre 2017].

[6] L. ada, «Learn Adafruit,» INGENIERÍA EN NYC Adafruit ®, [En línea]. Available: https://www.google.com.mx/search?num=30&newwindow=1&ei=UU5pWrHtIY70zgL-so_gBg&q=Visi%C3%B3n+de+conjunto+por+lady+ada+Los+sensores+PIR+le+permiten+sentir+el+movimiento%2C+casi+siempre+usado+para+detectar+si+un+humano+se+movi%C3%B3+dentro+o+fuera+del+rango+. [Último acceso: 19 Enero 2018].

[7] MAYLAMPS MECHATRONICS, «MAYLAMPS MECHATRONICS,» Naylamp Mechatronics Perú, [En línea]. Available: http://www.naylampmechatronics.com/sensores-proximidad/10-sensor-ultrasonido-hc-sr04.html. [Último acceso: 10 Noviembre 2017].

[8] Microsoft, «Support xbox,» MICROSOFT, [En línea]. Available: https://support.xbox.com/es-MX/xbox-360/accessories/kinect-sensor-components. [Último acceso: 10 Noviembre 2017].

[9] A. Gilat, Matlab: Intriducción con ejemplos prácticos, Barcelona. ESPAÑA: REVERTÉ, 206.

[10] The Mathwork, Inc, «Mathwork,» 2015. [En línea]. Available: https://la.mathworks.com/content/dam/mathworks/mathworks-dot-com/support/sysreq/files/SystemRequirements-Release2015a_Windows.pdf. [Último acceso: 10 Noviembre 2017].

[11] N. J. Muller, Tecnología Bluetooth, España: Mcgraw Hill Editorial, 2002.

[12] D. D. a. J. Zhu, «THE BLUETOOTH SYSTEM,» University of Victoria, Victoria, Canadá, 2001.

[13] Guangzhou HC Information Technology Co., Ltd. , «ElectronicosCaldas,» [En línea]. Available: http://www.electronicoscaldas.com/datasheet/HC-06_Wavesen.pdf. [Último acceso: 11 Noviembre 2017].

[14] I. S. A. Zalapa, «TESIS,» de *CONSTRUCCIÓN DE UN ROBOT BÍPEDO HUMANOIDE CON CONTROL DE EQUILIBRIO POR LÓGICA DIFUSA*, Morelia, Michoacán, México, Enero de 2015, p. 20.

[15] Raspberry, «RasPberry BLOG,» RASPBERRY PI FOUNDATION REINO UNIDO CARIDAD REGISTRADA, 2018. [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Último acceso: 11 Noviembre 2017].

[16] Arduino, «ARDUINO,» MCI Electronics, [En línea]. Available: http://arduino.cl/arduino-mega-2560/. [Último acceso: 10 Noviembre 2017].

[17] MARLIN P MARLIN P. JONES & ASSOC & ASSOC., INC., «MPJA,» [En línea]. Available: https://www.mpja.com/download/31150mp.pdf. [Último acceso: 11 Noviembre 2017].

[18] © 2001–2018 Pololu Corporation, «Pololu,» 2010. [En línea]. Available: https://www.pololu.com/product/1356. [Último acceso: 12 Noviembre 2018].

[19] Pololu Corporation, «Pololu Robotics & Electronics,» [En línea]. Available: https://www.pololu.com/docs/0J40/5. [Último acceso: 13 Noviembre 2017].

[20] Pololu Corporation, «Pololu Robotics & Electronics,» [En línea]. Available: https://www.pololu.com/docs/0J40/4.e. [Último acceso: 13 Noviembre 2017].

[21] ©Torq Pro & Tower Pro. , «Tower Pro,» [En línea]. Available: http://www.towerpro.com.tw/product/mg995/. [Último acceso: 13 Noviembre 2017].

[22] Pololu Corporation, «Pololu Robotics & Electronics,» [En línea]. Available: https://www.pololu.com/docs/0J40/3. [Último acceso: 2017 Noviembre 14].

[23] Microsoft Corporation, «Microsoft,» Microsoft Corporation, Noviembre 2017. [En línea]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=40278. [Último acceso: 14 Noviembre 2017].

[24] Microsoft Corporations, «Mirosoft,» Microsoft Corporations, Noviembre 2017. [En línea]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=40276. [Último acceso: 14 Noviembre 2017].

[25] Microsoft Corporation, «Microsoft,» Microsoft Corporation, Noviembre 2017. [En línea]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=34809. [Último acceso: 14 Noviembre 2017].

[26] MathWorks Image Acquisition Toolbox Team, «Mathworks,» The MathWorks, Inc., 17 Enero 2014. [En línea]. Available: https://la.mathworks.com/matlabcentral/fileexchange/40445-image-acquisition-toolbox-support-package-for-kinect-for-windows-sensor. [Último acceso: 15 Noviembre 2017].

[27] The MathWorks, Inc., «MathWorks,» The MathWorks, Inc., 2015. [En línea]. Available: https://la.mathworks.com/help/imaq/acquiring-image-and-skeletal-data-using-the-kinect.html. [Último acceso: 15 Noviembre 2017].

[28] C. H. Lehmann, «Sistemas de Coordenadas,» de *Geomatria Analitica*, MCxico, D. F., LtMUSA, S. A. de C. V., 1989, pp. 11-12.

[29] W. A. Granville, «Trigonometría,» de *Geometría plana y esférica*, Boston, Massachusett, USA, GRANVILLE SMITH MIKESH, 1980, pp. 127-160.

[30] P. Corcuera, Creación de interfaces de interfaces de usuario con Matlab, Cantabria: Universidad de Cantabria.

[31] The LEGO Group. , «LEGO,» The LEGO Group. , 2013. [En línea]. Available: https://www.lego.com/es-es/mindstorms/about-ev3. [Último acceso: 16 Nobiembre 2017].

[32] B. W. Evans, Manual de programación Arduino, San Francisco, California: Creative Commonds, 2007.

ANEXO A

1) Código Desarrollado en Matlab para empezar a detectar esqueleto

```
%%%Código en Matlab para empezar con las detecciones
% % % El sensor de Kinect para Windows aparece como dos dispositivos separados en IMAQHWINFO.
  hwInfo = imaqhwinfo('kinect')
  hwInfo.DeviceInfo(1)
  hwInfo.DeviceInfo(2)
% Cree los objetos VIDEOINPUT para las dos secuencias
sensorColor =  videoinput('kinect',1,'RGB_640x480');
figure(1)
preview(sensorColor);
sensorP = videoinput('kinect',2,'Depth_640X480');
%preview(sensorP);
fuente2 = getselectedsource(sensorP);
fuente2.BodyPosture = 'Standing';
fuente2.TrackingMode = 'Skeleton';
fuente2.DepthMode = 'Default';
fuente2.CameraElevationAngle=0;

% Configura el modo de disparo en 'manual'
triggerconfig([sensorColor sensorP],'manual');
%triggerconfig([sensorColor sensorP], 'immediate')
%Adquirir ## de cuadros para el disparon, que se ocupara lpara la detección
% Recuerde tener una persona en frente al Kinect para ver datos de seguimiento válidos.
cuadros=4;                    %## Numero de cuadros en modo de disparo 'manual'
 sensorColor.FramesPerTrigger = 1;  %número de cuadros por disparo para el sensor de colo
sensorP.FramesPerTrigger = cuadros; %número de cuadros para el sensor de profundiad
%%repite el disparo para ambos sensores
%sensorColor.TriggerRepeat = 200;
%sensorP.TriggerRepeat = 200;
%%banderas de deteccion de esqueletos y posicion rastreadas en cero
cualquierPosicionRastreada = 0;
cualquierEsqueletoRastreada=0;

%%%Bluecle infinito
while(1)
% %Compruebe si hay esqueletos rastreados a partir de los metadatos de profundidad
 while ( ( cualquierEsqueletoRastreada )== 0)
 %disp('no hay nadie');
 start(sensorP);
% Comienza el dispositivo de profundidad. Esto comienza la adquisición, pero no inicia el registro
de los datos adquiridos.
trigger(sensorP);      % Dispare los dispositivos para iniciar el registro de datos.
disp('...');           %%se puestran en pantalla para indicar que no se ha detectado a nadie
disp('no se detecto a nadie');

x=sensorP.InitialTriggerTime; %establecce el tiempo de disparo incial para el sensor de
profundidad
x(3);
% Recuperar los datos adquiridos, Recupera los cuadros y comprueba si se siguen los esqueletos
%[DatosDeColorDelCuadro] = getdata(sensorColor);
[DatosDeProfundidadDelCuadro, timeDataDepth, metaDataDepth] = getdata(sensorP);
% metaDataDepth; % Ver datos del esqueleto de los metadatos de profundidad
%Compruebe si hay esqueletos rastreados a partir de los metadatos de profundidad
cualquierPosicionRastreada = any(metaDataDepth(cuadros).IsPositionTracked ~= 0);
cualquierEsqueletoRastreada = any(metaDataDepth(cuadros).IsSkeletonTracked ~= 0);
```

```
stop([sensorColor sensorP]); % Detiene los dispositivos

%%toma una captura del sensor de profundidad y la muestra en una ventana
figure(1)
imapro=getsnapshot(sensorP);
imshow(imapro, [0 4000] );
%%tiempo limite de para que se despliegue mensaje de error, en caso de no
%%detectar a nadie
if (x(2)==(sensorP.Timeout-1))
    sensorP.Timeout=sensorP.Timeout+5;
end
  end   %%%Termina el bluque de detección
%%%%Inica algoritmo para graficación de extremidades y calculo de ángulos.
%ver cuantos esqueletos fueron rastreados
rastrearEsqueletos = find(metaDataDepth(cuadros).IsSkeletonTracked);
%rastrearEsqueletos =1;  %forza a la bandera a indicar detección de esqueleto
%obtiene las coordenadas de los metadatos de  profundidad de los esqueletos
%rastreados
obtenerCoordenadas = metaDataDepth(cuadros).JointWorldCoordinates(:, :, rastrearEsqueletos);
% Índices de unión del esqueleto con respecto a la imagen en color
esqueleto = metaDataDepth(cuadros).JointImageIndices(:, :, rastrearEsqueletos);
% guarda el numero de esqueletos rastreados
nSkeleton = length(rastrearEsqueletos);

%Muestra una imagen Deepth (se puede cambiar el formato de imagen)
figure(1)
imapro=getsnapshot(sensorP);
imshow(imapro, [0 4000] );

% Mapa esquemático de conexión para unir las articulaciones
mapaDeConexionDelEsqueleto = [[1 2]; % espalda baja
                        [2 3]; %espalda
                        [3 4]; %cuello
                        [3 5]; %hombro izquierdo
                        [5 6]; %brazo izq parte alta
                        [6 7];%brazo izq parte baja
                        [7 8]; %mano izq
                        [3 9]; %hombro derecho
                        [9 10]; %brazo der parte alta
                        [10 11]; %brazo der parte baja
                        [11 12];%mano der
                        [1 17]; % nalga derecha
                        [17 18]; %pierna derecha parte alta
                        [18 19];%pierna derecha parte baja
                        [19 20]; %pie derecho
                        [1 13]; %nalga izq
                        [13 14]; %pierna izq parte alta
                        [14 15];%pierna izq parte baja
                        [15 16]]; %pie izq

 % dibuja el esqueleto en la imagen Depth
 for i = 1:19
     if nSkeleton > 0
         X1                  =                  [esqueleto(mapaDeConexionDelEsqueleto(i,1),1,1)
esqueleto(mapaDeConexionDelEsqueleto(i,2),1,1)];
         Y1                  =                  [esqueleto(mapaDeConexionDelEsqueleto(i,1),2,1)
esqueleto(mapaDeConexionDelEsqueleto(i,2),2,1)];
         line(X1,Y1, 'LineWidth', 1.5, 'LineStyle', '-', 'Marker', '+', 'Color', 'r');
     end
```

```
     if nSkeleton > 1
         X2                  =                  [esqueleto(mapaDeConexionDelEsqueleto(i,1),1,2)
esqueleto(mapaDeConexionDelEsqueleto(i,2),1,2)];
         Y2                  =                  [esqueleto(mapaDeConexionDelEsqueleto(i,1),2,2)
esqueleto(mapaDeConexionDelEsqueleto(i,2),2,2)];
         line(X2,Y2, 'LineWidth', 1.5, 'LineStyle', '-', 'Marker', '+', 'Color', 'g');
     end
    hold on;
 end
 %Traza una linea horizontal a la altura de los hombros
 line([(esqueleto(5,1)-200) (esqueleto(9,1)+200)],[esqueleto(5,2) esqueleto(9,2)], 'LineWidth',
1.5, 'LineStyle', '-', 'Marker', '+', 'Color', 'r');

 %%%Condicionales para detectar posición de las manos
  if(esqueleto(8,2)>esqueleto(5,2))
      disp('mano izquierda abajo')
  end
   if(esqueleto(8,2)<esqueleto(5,2))
      disp('mano izquierda arriba')
   end
   if(esqueleto(12,2)>esqueleto(9,2))
      disp('mano derecha abajo')
  end
   if(esqueleto(12,2)<esqueleto(9,2))
      disp('mano derecha arriba')
   end
  cualquierEsqueletoRastreada=0;
end
```

2) Script para conexión de la PC con Módulo HC-06 vía bluetooth

```
%info=instrhwinfo('Bluetooth'); %invoca a los dispositivos bluetooth
%info.RemoteNames;               %muestra los nombres de los dispositivos disponibles
if(exist('blu')==0)
   blu=Bluetooth('Roboto',1)
   ac='buscando bluetooth...'
   estado=blu.Status;
end

if(exist('blu')>0)
cerrado='closed';
abierto='open';
if(strcmp(cerrado,estado))
   ac='conectando...'
   fopen(blu);
   estado=blu.Status;
end
end

if(strcmp(estado,abierto))    %si hay conexión con el dispositivo
blu.ValuesSent                 %Muestra el número de paquetes enviados
instruccion = 'pos';                  %etiqueta crrespondiente a la instracción que queremos
%darle a la tarjeta principal
dispo =   12;                  %numero de canal que queremos controlar
dispoc = int2str(dispo);       %convierte el valor de int a string
valor = 1850;                  %posición (valor en microsegundos) de la posición deseada
valor=valor*4;                 %multiplica el valor por 4, ya que la biblioteca de arduino
                               % para la tarjeta servocontroladora pide el
                               % valor multiplicado por 4
```

```
valorc=int2str(valor);          %convierte de int a string
comando = [ instruccion '>' dispoc ',' (valorc) ]  %crea la cadena con el comando
fwrite(blu,comando);              %escribe y envia la cadena con el comando (byte a byte hasta
acabar)

dispo =   20;                   %numero de canal que queremos controlar
dispoc = int2str(dispo);        %convierte el valor de int a string
valor = 1550;                   %posición (valor en microsegundos) de la posición deseada
valor=valor*4;                  %multiplica el valor por 4, ya que la biblioteca de arduino
                                % para la tarjeta servocontroladora pide el
                                % valor multiplicado por 4
valorc=int2str(valor);          %convierte de int a string
comando = [ instruccion '>' dispoc ',' (valorc) ]  %crea la cadena con el comando
 fwrite(blu,comando);              %escribe y envia la cadena con el comando (byte a byte hasta
acabar)
%
comando = [ 'fin' '>' ]
fwrite(blu,comando);
%
end
```

3) Código Del Script final de la IHM

```
%%%Código Del Script final de la IHM
% % % El sensor de Kinect para Windows aparece como dos dispositivos separados en IMAQHWINFO.
  %hwInfo = imaqhwinfo('kinect')
  %hwInfo.DeviceInfo(1)
  %hwInfo.DeviceInfo(2)
% Cree los objetos VIDEOINPUT para las dos secuencias
%sensorColor =  videoinput('kinect',1,'RGB_1280x960
sensorColor =  videoinput('kinect',1,'RGB_640x480');
%sensorColor =  videoinput('kinect',1,'RawBayer_640x480');
%sensorColor =  videoinput('kinect',1,'YUV_640x480');
figure(1)
preview(sensorColor);

%sensorP = videoinput('kinect',2,'Depth_320x240');
sensorP = videoinput('kinect',2,'Depth_640X480');
%sensorP = videoinput('kinect',2,'Depth_80x60');
%preview(sensorP);
fuente2 = getselectedsource(sensorP);
fuente2.BodyPosture = 'Standing';
fuente2.TrackingMode = 'Skeleton';
fuente2.DepthMode = 'Default';
fuente2.CameraElevationAngle=0;

% Configura el modo de disparo en 'manual'
triggerconfig([sensorColor sensorP],'manual');
%triggerconfig([sensorColor sensorP], 'immediate')
%Adquirir ## de cuadros para el disparon, que se ocupara lpara la detección
% Recuerde tener una persona en frente al Kinect para ver datos de seguimiento válidos.
cuadros=4;  %## Numero de cuadros en modo de disparo 'manual'
sensorColor.FramesPerTrigger = 1;  %numero de cuadros por disparo para el sensor de colo
sensorP.FramesPerTrigger = cuadros; %numero de cuadros para el sensor de profundiad
%%repite el disparo para ambos sensores
%sensorColor.TriggerRepeat = 200;
%sensorP.TriggerRepeat = 200;
%%banderas de deteccion de esqueletos y posicion rastreadas en cero
cualquierPosicionRastreada = 0;
```

```
cualquierEsqueletoRastreada=0;

%%%Bluecle infinito
while(1)
% %Compruebe si hay esqueletos rastreados a partir de los metadatos de profundidad
  while ( ( cualquierEsqueletoRastreada )== 0)
    %disp('no hay nadie');
    start(sensorP);
% Comienza el dispositivo de profundidad. Esto comienza la adquisición, pero no inicia el registro
de los datos adquiridos.
trigger(sensorP);      % Dispare los dispositivos para iniciar el registro de datos.
disp('...');           %%se puestran en pantalla para indicar que no se ha detectado a nadie
disp('no se detecto a nadie');

x=sensorP.InitialTriggerTime; %establecce el tiempo de disparo incial para el sensor de
profundidad
x(3);
% Recuperar los datos adquiridos, Recupera los cuadros y comprueba si se siguen los esqueletos
%[DatosDeColorDelCuadro] = getdata(sensorColor);
[DatosDeProfundidadDelCuadro, timeDataDepth, metaDataDepth] = getdata(sensorP);
% metaDataDepth; % Ver datos del esqueleto de los metadatos de profundidad

%Compruebe si hay esqueletos rastreados a partir de los metadatos de profundidad
cualquierPosicionRastreada = any(metaDataDepth(cuadros).IsPositionTracked ~= 0);
cualquierEsqueletoRastreada = any(metaDataDepth(cuadros).IsSkeletonTracked ~= 0);

stop([sensorColor sensorP]); % Detiene los dispositivos

%%toma una captura del sensor de profundidad y la muestra en una ventana
figure(1)
imapro=getsnapshot(sensorP);
imshow(imapro, [0 4000] );

%%tiempo limite de para que se despliegue mensaje de error, en caso de no
%%detectar a nadie
if (x(2)==(sensorP.Timeout-1))
    sensorP.Timeout=sensorP.Timeout+5;
end

  end   %%%Termina el bluque de detección

%%%%Inica algoritmo para graficación de extremidades y calculo de ángulos.
%ver cuantos esqueletos fueron rastreados
rastrearEsqueletos = find(metaDataDepth(cuadros).IsSkeletonTracked);
%rastrearEsqueletos =1;  %forza a la bandera a indicar detección de esqueleto
%obtiene las coordenadas de los metadatos de  profundidad de los esqueletos
%rastreados
obtenerCoordenadas = metaDataDepth(cuadros).JointWorldCoordinates(:, :, rastrearEsqueletos);
% Índices de unión del esqueleto con respecto a la imagen en color
esqueleto = metaDataDepth(cuadros).JointImageIndices(:, :, rastrearEsqueletos);
% guarda el numero de esqueletos rastreados
nSkeleton = length(rastrearEsqueletos);

%Muestra una imagen Deepth (se puede cambiar el formato de imagen)
figure(1)
imapro=getsnapshot(sensorP);
imshow(imapro, [0 4000] );

% Mapa esquemático de conexión para unir las articulaciones
```

```
mapaDeConexionDelEsqueleto = [[1 2]; % espalda baja
                          [2 3]; %espalda
                          [3 4]; %cuello
                          [3 5]; %hombro izquierdo
                          [5 6]; %brazo izq parte alta
                          [6 7];%brazo izq parte baja
                          [7 8]; %mano izq
                          [3 9]; %hombro derecho
                          [9 10]; %brazo der parte alta
                          [10 11]; %brazo der parte baja
                          [11 12];%mano der
                          [1 17]; % nalga derecha
                          [17 18]; %pierna derecha parte alta
                          [18 19];%pierna derecha parte baja
                          [19 20]; %pie derecho
                          [1 13]; %nalga izq
                          [13 14]; %pierna izq parte alta
                          [14 15];%pierna izq parte baja
                          [15 16]]; %pie izq

 % dibuja el esqueleto en la imagen Depth
 for i = 1:19
     if nSkeleton > 0
         X1                  =                  [esqueleto(mapaDeConexionDelEsqueleto(i,1),1,1)
esqueleto(mapaDeConexionDelEsqueleto(i,2),1,1)];
         Y1                  =                  [esqueleto(mapaDeConexionDelEsqueleto(i,1),2,1)
esqueleto(mapaDeConexionDelEsqueleto(i,2),2,1)];
         line(X1,Y1, 'LineWidth', 1.5, 'LineStyle', '-', 'Marker', '+', 'Color', 'r');
     end
     if nSkeleton > 1
         X2                  =                  [esqueleto(mapaDeConexionDelEsqueleto(i,1),1,2)
esqueleto(mapaDeConexionDelEsqueleto(i,2),1,2)];
         Y2                  =                  [esqueleto(mapaDeConexionDelEsqueleto(i,1),2,2)
esqueleto(mapaDeConexionDelEsqueleto(i,2),2,2)];
         line(X2,Y2, 'LineWidth', 1.5, 'LineStyle', '-', 'Marker', '+', 'Color', 'g');
     end
    hold on;
 end
 %Traza una linea horizontal a la altura de los hombros
 line([(esqueleto(5,1)-200) (esqueleto(9,1)+200)],[esqueleto(5,2) esqueleto(9,2)], 'LineWidth',
1.5, 'LineStyle', '-', 'Marker', '+', 'Color', 'r');

 %%%Condicionales para detectar posición de las manos
  if(esqueleto(8,2)>esqueleto(5,2))
      disp('mano izquierda abajo')
  end
   if(esqueleto(8,2)<esqueleto(5,2))
      disp('mano izquierda arriba')
   end
   if(esqueleto(12,2)>esqueleto(9,2))
      disp('mano derecha abajo')
  end
   if(esqueleto(12,2)<esqueleto(9,2))
      disp('mano derecha arriba')
   end

  %para extremidades y articulaciones superiores derechas
  %para calcular ángulo de axila derecha
```

```
  xprima1=-((cosd(90)*(esqueleto(5,1)-esqueleto(9,1)))        -      (sind(90)*(esqueleto(5,2)-
esqueleto(9,2))))     + esqueleto(9,1);
  yprima1=-((sind(90)*(esqueleto(5,1)-esqueleto(9,1)))        +      (cosd(90)*(esqueleto(5,2)-
esqueleto(9,2))))      +esqueleto(9,2);
  line([xprima1 esqueleto(9,1)],[yprima1 esqueleto(9,2)], 'LineWidth', 1.5, 'LineStyle', '-',
'Marker', '+', 'Color', 'r');
  hombrosrot=sqrt((xprima1-esqueleto(9,1))^2+(yprima1-esqueleto(9,2))^2);
  brazod=sqrt((esqueleto(9,1)-esqueleto(10,1))^2+(esqueleto(9,2)-esqueleto(10,2))^2);
  imahombroizq_codod=sqrt((xprima1-esqueleto(10,1))^2+(yprima1-esqueleto(10,2))^2);
  line([xprima1 (esqueleto(10,1))],[yprima1 esqueleto(10,2)], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'b');
  ang_axilaD=radtodeg(          acos(           (imahombroizq_codod^2-brazod^2-hombrosrot^2)/(-
2*(brazod)*(hombrosrot)) ));
  text(esqueleto(9,1),esqueleto(9,2),int2str(ang_axilaD));

  %para calcular ángulo de codo derecho
  antebrazod=sqrt((esqueleto(10,1)-esqueleto(11,1))^2+(esqueleto(10,2)-esqueleto(11,2))^2);
  imahombro_munecad=sqrt((esqueleto(9,1)-esqueleto(11,1))^2+(esqueleto(9,2)-
esqueleto(11,2))^2);
  line([(esqueleto(9,1))  (esqueleto(11,1))],[esqueleto(9,2)  esqueleto(11,2)], 'LineWidth', 1,
'LineStyle', '-',  'Color', 'b');
  ang_codoD=radtodeg(           acos(            (imahombro_munecad^2-brazod^2-antebrazod^2)/(-
2*(brazod)*(antebrazod))) );
  text(esqueleto(10,1),esqueleto(10,2),int2str(ang_codoD));

  %para calcular ángulo de muñeca derecha
  xprima2=((cosd(90)*(esqueleto(10,1)-esqueleto(11,1)))       -      (sind(90)*(esqueleto(10,2)-
esqueleto(11,2))))     + esqueleto(11,1);
  yprima2=((sind(90)*(esqueleto(10,1)-esqueleto(11,1)))       +      (cosd(90)*(esqueleto(10,2)-
esqueleto(11,2))))      +esqueleto(11,2);
  antebrazodrot=sqrt((xprima2-esqueleto(11,1))^2+(yprima2-esqueleto(11,2))^2);
  line([esqueleto(11,1)  xprima2],[esqueleto(11,2)  yprima2], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'r');
  manoder=sqrt((esqueleto(11,1)-esqueleto(12,1))^2+(esqueleto(11,2)-esqueleto(12,2))^2);
  imadedosd_codod=sqrt((esqueleto(12,1)-xprima2)^2+(esqueleto(12,2)-yprima2)^2);
  line([(esqueleto(12,1))  xprima2],[esqueleto(12,2)  yprima2], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'b');
  ang_munecaD=radtodeg(          acos(          (imadedosd_codod^2-manoder^2-antebrazodrot^2)/(-
2*(manoder)*(antebrazodrot))) );
  text(esqueleto(11,1),esqueleto(11,2),int2str(ang_munecaD));

  text(esqueleto(5,1),esqueleto(5,2)+10,'hI');
  text(xprima1,yprima1,'hI´');
  text(esqueleto(10,1), esqueleto(10,2)+10,'codD');
  text(esqueleto(11,1), esqueleto(11,2)+10,'muñD');

  %para extremidades y articulaciones superiores izquierdas
  %para calcular ángulo de axila izquierda
  xprima3=((cosd(90)*(esqueleto(9,1)-esqueleto(5,1)))        -      (sind(90)*(esqueleto(9,2)-
esqueleto(5,2))))     + esqueleto(5,1);
  yprima3=((sind(90)*(esqueleto(9,1)-esqueleto(5,1)))        +      (cosd(90)*(esqueleto(9,2)-
esqueleto(5,2))))      +esqueleto(5,2);
  line([xprima3 esqueleto(5,1)],[yprima3 esqueleto(5,2)], 'LineWidth', 1.5, 'LineStyle', '-',
'Marker', '+', 'Color', 'r');
  hombrosrotI=sqrt((xprima3-esqueleto(5,1))^2+(yprima3-esqueleto(5,2))^2);
  brazoI=sqrt((esqueleto(5,1)-esqueleto(6,1))^2+(esqueleto(5,2)-esqueleto(6,2))^2);
  imahombroD_codoI=sqrt((xprima3-esqueleto(6,1))^2+(yprima3-esqueleto(6,2))^2);
  line([xprima3 (esqueleto(6,1))],[yprima3 esqueleto(6,2)], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'b');
```

```
  ang_axilaI=radtodeg(           acos(           (imahombroD_codoI^2-brazoI^2-hombrosrotI^2)/(-
2*(brazoI)*(hombrosrotI)) ));
  text(esqueleto(5,1),esqueleto(5,2),int2str(ang_axilaI));

  %para calcular ángulo de codo izquierda
  antebrazoI=sqrt((esqueleto(6,1)-esqueleto(7,1))^2+(esqueleto(6,2)-esqueleto(7,2))^2);
  imahombro_munecaI=sqrt((esqueleto(5,1)-esqueleto(7,1))^2+(esqueleto(5,2)-esqueleto(7,2))^2);
  line([(esqueleto(5,1))  (esqueleto(7,1))],[esqueleto(5,2)  esqueleto(7,2)],  'LineWidth',  1,
'LineStyle', '-',  'Color', 'b');
  ang_codoI=radtodeg(           acos(           (imahombro_munecaI^2-brazod^2-antebrazoI^2)/(-
2*(brazod)*(antebrazoI))) );
  text(esqueleto(6,1),esqueleto(6,2),int2str(ang_codoI));

  %para calcular ángulo de muñeca izquierda
  xprima4=((cosd(90)*(esqueleto(6,1)-esqueleto(7,1)))         -      (sind(90)*(esqueleto(6,2)-
esqueleto(7,2))))     + esqueleto(7,1);
  yprima4=((sind(90)*(esqueleto(6,1)-esqueleto(7,1)))         +      (cosd(90)*(esqueleto(6,2)-
esqueleto(7,2))))      +esqueleto(7,2);
  antebrazoIrot=sqrt((xprima4-esqueleto(7,1))^2+(yprima4-esqueleto(7,2))^2);
  line([esqueleto(7,1)  xprima4],[esqueleto(7,2)  yprima4],  'LineWidth',  1,  'LineStyle',  '-',
'Color', 'r');
  manoI=sqrt((esqueleto(7,1)-esqueleto(8,1))^2+(esqueleto(7,2)-esqueleto(8,2))^2);
  imadedosI_codoI=sqrt((esqueleto(8,1)-xprima4)^2+(esqueleto(8,2)-yprima4)^2);
  line([(esqueleto(8,1))  xprima4],[esqueleto(8,2)  yprima4], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'b');
  ang_munecaI=radtodeg(          acos(           (imadedosI_codoI^2-manoI^2-antebrazoIrot^2)/(-
2*(manoI)*(antebrazoIrot))) );
  text(esqueleto(7,1),esqueleto(7,2),int2str(ang_munecaI));

  text(esqueleto(9,1),esqueleto(9,2)+10,'hD');
  text(xprima3,yprima3,'hD´');
  text(esqueleto(6,1), esqueleto(6,2)+10,'codI');
  text(esqueleto(7,1), esqueleto(7,2)+10,'muñI');

    %para extremidades y articulaciones inferiores derechas
  %para calcular ángulo de pierna derecha
  xprima5=-((cosd(90)*(esqueleto(13,1)-esqueleto(17,1)))       -     (sind(90)*(esqueleto(13,2)-
esqueleto(17,2))))     + esqueleto(17,1);
  yprima5=-((sind(90)*(esqueleto(13,1)-esqueleto(17,1)))       +     (cosd(90)*(esqueleto(13,2)-
esqueleto(17,2))))      +esqueleto(17,2);
  line([xprima5 esqueleto(17,1)],[yprima5 esqueleto(17,2)], 'LineWidth', 1.5, 'LineStyle', '-',
'Marker', '+', 'Color', 'r');
  caderarot=sqrt((xprima5-esqueleto(17,1))^2+(yprima5-esqueleto(17,2))^2);
  piernad=sqrt((esqueleto(17,1)-esqueleto(18,1))^2+(esqueleto(17,2)-esqueleto(18,2))^2);
  imacaderaI_rodD=sqrt((xprima5-esqueleto(18,1))^2+(yprima5-esqueleto(18,2))^2);
  line([xprima5 (esqueleto(18,1))],[yprima5 esqueleto(18,2)], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'b');
  ang_pierna=radtodeg(           acos(            (imacaderaI_rodD^2-piernad^2-caderarot^2)/(-
2*(piernad)*(caderarot)) ));
  text(esqueleto(17,1),esqueleto(17,2),int2str(ang_pierna));

  %para calcular ángulo de rodilla derecha
  tibiad=sqrt((esqueleto(18,1)-esqueleto(19,1))^2+(esqueleto(18,2)-esqueleto(19,2))^2);
  imacaderaD_tobilloD=sqrt((esqueleto(17,1)-esqueleto(19,1))^2+(esqueleto(17,2)-
esqueleto(19,2))^2);
  line([(esqueleto(17,1)) (esqueleto(19,1))],[esqueleto(17,2) esqueleto(19,2)], 'LineWidth', 1,
'LineStyle', '-',  'Color', 'b');
  ang_rodD=radtodeg(           acos(            (imacaderaD_tobilloD^2-piernad^2-tibiad^2)/(-
2*(piernad)*(tibiad))) );
```

```
  text(esqueleto(18,1),esqueleto(18,2),int2str(ang_rodD));

  %para calcular ángulo de tobillo derecha
  xprima6=((cosd(90)*(esqueleto(18,1)-esqueleto(19,1)))       -     (sind(90)*(esqueleto(18,2)-
esqueleto(19,2))))     + esqueleto(19,1);
  yprima6=((sind(90)*(esqueleto(18,1)-esqueleto(19,1)))       +     (cosd(90)*(esqueleto(18,2)-
esqueleto(19,2))))      +esqueleto(19,2);
  tibiadrot=sqrt((xprima6-esqueleto(19,1))^2+(yprima6-esqueleto(19,2))^2);
  line([esqueleto(19,1) xprima6],[esqueleto(19,2) yprima6], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'r');
  pieD=sqrt((esqueleto(19,1)-esqueleto(20,1))^2+(esqueleto(19,2)-esqueleto(20,2))^2);
  imadedosD_rodD=sqrt((esqueleto(20,1)-xprima6)^2+(esqueleto(20,2)-yprima6)^2);
  line([(esqueleto(20,1)) xprima6],[esqueleto(20,2) yprima6], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'b');
  ang_tobD=radtodeg(   acos(   (imadedosD_rodD^2-pieD^2-tibiadrot^2)/(-2*(pieD)*(tibiadrot))) );
  text(esqueleto(19,1),esqueleto(19,2),int2str(ang_tobD));

  text(esqueleto(17,1),esqueleto(17,2)+10,'cD');
  text(xprima6,yprima6,'cD´');
  text(esqueleto(18,1), esqueleto(18,2)+10,'rodD');
  text(esqueleto(19,1), esqueleto(19,2)+10,'tobD');

  %para extremidades y articulaciones inferiores izquierdas
  %para calcular ángulo de pierna izquierda
  xprima7=((cosd(90)*(esqueleto(17,1)-esqueleto(13,1)))       -     (sind(90)*(esqueleto(17,2)-
esqueleto(13,2))))     + esqueleto(13,1);
  yprima7=((sind(90)*(esqueleto(17,1)-esqueleto(13,1)))       +     (cosd(90)*(esqueleto(17,2)-
esqueleto(13,2))))      +esqueleto(13,2);
  line([xprima7 esqueleto(13,1)],[yprima7 esqueleto(13,2)], 'LineWidth', 1.5, 'LineStyle', '-',
'Marker', '+', 'Color', 'r');
  caderarotD=sqrt((xprima7-esqueleto(13,1))^2+(yprima7-esqueleto(13,2))^2);
  piernaI=sqrt((esqueleto(13,1)-esqueleto(14,1))^2+(esqueleto(13,2)-esqueleto(14,2))^2);
  imacadD_rodI=sqrt((xprima7-esqueleto(14,1))^2+(yprima7-esqueleto(14,2))^2);
  line([xprima7 (esqueleto(14,1))],[yprima7 esqueleto(14,2)], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'b');
  ang_piernaI=radtodeg(            acos(            (imacadD_rodI^2-piernaI^2-caderarotD^2)/(-
2*(piernaI)*(caderarotD)) ));
  text(esqueleto(13,1),esqueleto(13,2),int2str(ang_piernaI));

  %para calcular ángulo de rodilla izquierda
  tibiaI=sqrt((esqueleto(13,1)-esqueleto(14,1))^2+(esqueleto(13,2)-esqueleto(14,2))^2);
  imacadI_tobI=sqrt((esqueleto(13,1)-esqueleto(15,1))^2+(esqueleto(13,2)-esqueleto(15,2))^2);
  line([(esqueleto(13,1)) (esqueleto(15,1))],[esqueleto(13,2) esqueleto(15,2)], 'LineWidth', 1,
'LineStyle', '-',  'Color', 'b');
  ang_rodI=radtodeg(   acos(   (imacadI_tobI^2-piernaI^2-tibiaI^2)/(-2*(piernaI)*(tibiaI))) );
  text(esqueleto(14,1),esqueleto(14,2),int2str(ang_rodI));

  %para calcular ángulo de tobillo izquierdo
  xprima8=-((cosd(90)*(esqueleto(14,1)-esqueleto(15,1)))       -     (sind(90)*(esqueleto(14,2)-
esqueleto(15,2))))     + esqueleto(15,1);
  yprima8=-((sind(90)*(esqueleto(14,1)-esqueleto(15,1)))       +     (cosd(90)*(esqueleto(14,2)-
esqueleto(15,2))))      +esqueleto(15,2);
  tibiaIrot=sqrt((xprima8-esqueleto(15,1))^2+(yprima8-esqueleto(15,2))^2);
  line([esqueleto(15,1) xprima8],[esqueleto(15,2) yprima8], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'r');
  pieI=sqrt((esqueleto(15,1)-esqueleto(16,1))^2+(esqueleto(15,2)-esqueleto(16,2))^2);
  imadedosI_rodI=sqrt((esqueleto(16,1)-xprima8)^2+(esqueleto(16,2)-yprima8)^2);
  line([(esqueleto(16,1)) xprima8],[esqueleto(16,2) yprima8], 'LineWidth', 1, 'LineStyle', '-',
'Color', 'b');
```

```
  ang_tobI=radtodeg(   acos(   (imadedosI_rodI^2-pieI^2-tibiaIrot^2)/(-2*(pieI)*(tibiaIrot))) );
  text(esqueleto(15,1),esqueleto(15,2),int2str(ang_tobI));

  text(esqueleto(14,1),esqueleto(14,2)+10,'rodI');
  text(esqueleto(13,1),esqueleto(13,2)+10,'cI');
  text(xprima8,yprima8,'rodI´');
  text(esqueleto(15,1), esqueleto(15,2)+10,'tobI');

     %para calcular ángulo de cuello
  cuello=sqrt((esqueleto(3,1)-esqueleto(4,1))^2+(esqueleto(3,2)-esqueleto(4,2))^2);
  horc=sqrt((esqueleto(5,1)-esqueleto(3,1))^2+(esqueleto(3,2)-esqueleto(3,2))^2);
  imacab_hombroI=sqrt((esqueleto(4,1)-esqueleto(5,1))^2+(esqueleto(4,2)-esqueleto(3,2))^2);
  line([(esqueleto(4,1))  (esqueleto(5,1))],[esqueleto(3,2)  esqueleto(3,2)],  'LineWidth',  1,
'LineStyle', '-',  'Color', 'r');
  line([(esqueleto(4,1))  (esqueleto(5,1))],[esqueleto(4,2)  esqueleto(3,2)],  'LineWidth',  1,
'LineStyle', '-',  'Color', 'b');
  ang_cuello=radtodeg(   acos(   ( imacab_hombroI^2-(horc)^2-cuello^2)/(-2*(horc)*(cuello))) );
  text(esqueleto(3,1),esqueleto(3,2),int2str(ang_cuello));
  text(esqueleto(3,1),esqueleto(3,2)+10,'cuello');
  text(esqueleto(4,1),esqueleto(4,2),'cab');
 hold off;

 %%resetea la bandera de rastreo para que se vuelva a inciar el buque de
 %%deteccion y con ello obtener nuevos metadatos
 cualquierEsqueletoRastreada = 0;

%info=instrhwinfo('Bluetooth'); %invoca a los dispositivos bluetooth
%info.RemoteNames;               %muestra los nombres de los dispositivos disponibles

%%Preguna si existe objeto de bluetooth, si no existe lo crea
if(exist('blu')==0)
   blu=Bluetooth('Roboto',1)    %dispositivo al que se conecta
   ac='Creando vinculo co Roboto...'
   estado=blu.Status;           %pregunta el estado de la conexion
end
%si existe objeto blu
if(exist('blu')>0)
cerrado='closed';               %define etiqueta cerrado
abierto='open';                 %deine etiqueta abierto
if(strcmp(cerrado,estado))      %si no hay conexion con el dispositivo
   ac='Estableciendo conexión...' %mensaje de conexion en proceso
   fopen(blu);                  %conecta con el dispositivo bluetooth
   estado=blu.Status;           %pregunta por el estado de la conexion
end

end

if(strcmp(abierto,estado))
%%%Limites y envio de los angulos vía bluetooth en con el factor requerido
  factor=52; %4*13 %%%este factor se calcula a partir de dividir 180
  grados entre el rango en microsegundos del ancho del pulso que requiere
  un servomotor

 valor=ang_codo; %carga a la variable valor el angulo de codo calculado
 %limites inferior del angulo (limite fisico del robot humanoide)
 if(valor<60)
     valor=60;
 end
  if(valor>180)
```

```
     valor=180;
  end

instru = 'pos';            %asigna a la instruccion la cadena 'pos'
dispo =   12;              %numero de canal correspondiente al servomotor
del codo
dispoc = int2str(dispo);   % convierte la variable en formato int a string para poder ser enviado
valorc=int2str((valor*factor));
comando = [ instru '>' dispoc ',' (valorc) ]; %comando a enviar
fwrite(blu,comando);         %envia comando por el dispositivo bluetooth

%%a partir de aqui los limites y sintazis es similar a como se describio
anteriormente
valor=ang_axila;
valor=(valor*13)+950;
 if(valor<820)
     valor=820;
 end
  if(valor>3168)
     valor=3168;
  end

instru = 'pos';
dispo =   20;
dispoc = int2str(dispo);
valorc=int2str(valor*4);
comando = [ instru '>' dispoc ',' (valorc) ];
fwrite(blu,comando);

valor=ang_muneca;
valor=valor*13+656;
 if(valor<656)
     valor=656;
 end
  if(valor>2720)
     valor=2720;
  end

instru = 'pos';
dispo =   13;
dispoc = int2str(dispo);
valorc=int2str(valor*4);
comando = [ instru '>' dispoc ',' (valorc) ];
fwrite(blu,comando);
%%SI SE REQUIERE CONTROLAR MÁS SERVOS, AGREGAR AQUI LOS LIMITES Y ENVIARLOS
%%A LA TARJETA PRINCIPAL

%%%Fin del envio de comandos, cadena necesaria para poder interpretar la
%%%trama en la tarjeta principal
comando = [ 'fin' '>' ]
fwrite(blu,comando);

tmax=2,t=0;
tic
while(t<tmax)
    t = toc
end

%pregunta el numero de bytes enviados
```

```
%while(blu.valuesSent=numData)
%numdata=blu.valuesSent
%end
end

end    %%FIN DEL BUCLE INFINITO
```

4) Código en Arduino con la biblioteca de Pololu

```
#include <PololuMaestro.h>
#define baud_servos 115200
MiniMaestro maestro(Serial2);

void setup() {
  ////INICIA PUERTO SERIE USB A 11500 BAUDIOS
  Serial.begin(115200);
  Serial.println("Levantando el módulo Mini Maestro 24 ch");
  ////////////////////////////////////////////////////////////

  ////CONFIGURACIONES PARA TARJETA CONTROLADORA MINI MAESTRO 24 CH//////
  pinMode(22, OUTPUT);       //RESET (ACTIVO EN BAJO) DEL MINI CONTROLLER
  pinMode(24, INPUT);        //ENTRADA PARA LEER BANDERA DE ERROR
  pinMode(26, OUTPUT);       //ALIMENTACION + DEL MINI CONTROLLER
  pinMode(28, OUTPUT);       //ALIMENTACION (TIERRA) DEL MINI CONTROLLER
  digitalWrite(26, HIGH);    //+
  digitalWrite(28, LOW) ;    //-
  digitalWrite(22, LOW);     //Borra bandera de inicio si es que la hay
  delay (500) ;              // Espera en bajo para poder resetear la bandera de error
  digitalWrite(22, HIGH);    //Termina el reseteo del modulo
  Serial2 .begin(baud_servos);
  delay(1000);
  ////////////////////////////////////////////////////////////////////
}
void loop() {
   maestro.setTarget(23, 1450 * 4);
   delayMicroseconds(700);
}
```

5) Código para ECHO del bluetooth para enviar los comandos AT

```
define BT1 Serial3
#define baud 115200
void setup(){
    ////INICIA PUERTO SERIE A 9600 BAUDIOS
  Serial.begin(115200);
  Serial.println("Levantando el módulo Mini Maestro 24 ch");
    ////CONFIGURACIÓN PARA INICIAR EL BLUETOOH
  pinMode(30, OUTPUT);         // PIN COMO SALIDA PARA ALIMENTACION + BLUETTOH
  pinMode(32, OUTPUT);         // PIN COMO SALIDA PARA ALIMENTACION - BLUETTOH
  pinMode(34, OUTPUT);         //Pin como salida para el Enable del módulo bluetooth
  pinMode(36, OUTPUT);        //Tierra del div de voltaje (de TX (5V) del Arduino MEGA a RX del BLUE(3.3))
  //se hace acomodo de cables con divisor con resistencias)
  digitalWrite (32, LOW);      //activa la tierra
  digitalWrite(34, HIGH);      // Enable en alto antes de prender el módulo Al poner en HIGH forzaremos
el modo AT
  delay(500);                  //esperar antes de encender el módulo
```

```
  digitalWrite (30, HIGH);    //Enciende el modulo
  digitalWrite (32, LOW);    //Enciende el modulo
  delay (500);
  digitalWrite(34, LOW);    // Enable en bajo después de prender el módulo bluetooth  para poder entrar
al modo AT normal
  BT1.begin(baud);
  Serial.println("Levantando el módulo HC-06");
  Serial.println("Esperando comandos AT:");
  delay(1000);
}
void loop() {
  while(1){
  if (Serial.available()) {
    char c=Serial.read();
    Serial.write(c);
    BT1.write(c);
  }
  if (BT1.available()>0){
    Serial.write(BT1.read());
  }
  }
}
```

6) Código final de la tarjeta principal

```
#include <PololuMaestro.h>
#define BT1 Serial3
#define baud 115200
#define baud_servos 115200
MiniMaestro maestro(Serial2);

void setup() {
  ////INICIA PUERTO SERIE A 9600 BAUDIOS
  Serial.begin(115200);
  Serial.println("Levantando el módulo Mini Maestro 24 ch");
  ////////////////////////////////////////////////////////////

  ////CONFIGURACIONES PARA TARJETA CONTROLADORA MINI MAESTRO 24 CH//////
  pinMode(22, OUTPUT);      //RESET (ACTIVO EN BAJO) DEL MINI CONTROLLER
  pinMode(24, INPUT);       //ENTRADA PARA LEER BANDERA DE ERROR
  pinMode(26, OUTPUT);      //ALIMENTACION + DEL MINI CONTROLLER
  pinMode(28, OUTPUT);      //ALIMENTACION (TIERRA) DEL MINI CONTROLLER
  digitalWrite(26, HIGH);  //+
  digitalWrite(28, LOW) ; //-

  digitalWrite(22, LOW);      //Borra bandera de inicio si es que la hay
  delay (500) ;              // Espera en bajo para poder resetear la bandera de error
  digitalWrite(22, HIGH);     //Termina el reseteo del modulo

  Serial2.begin(baud_servos);
  delay(1000);
  ////////////////////////////////////////////////////////////////////
home();
  ////CONFIGURACIÓN PARA INICIAR EL BLUETOOH
  pinMode(30, OUTPUT);        // PIN COMO SALIDA PARA ALIMENTACION + BLUETTOH
  pinMode(32, OUTPUT);        // PIN COMO SALIDA PARA ALIMENTACION - BLUETTOH
  pinMode(34, OUTPUT);        //Pin como salida para el Enable del módulo bluetooth
```

```
  pinMode(36, OUTPUT);       //Tierra del div de voltaje (de TX (5V) del Arduino MEGA a RX del BLUE(3.3))
se hace acomodo de cables con divisor con resistencias)
  digitalWrite (32, LOW);     //activa la tierra

  digitalWrite(34, HIGH);     // Enable en alto antes de prender el módulo Al poner en HIGH forzaremos
el modo AT
  delay(500);                 //esperar antes de encender el módulo
  digitalWrite (30, HIGH);    //Enciende el modulo
  digitalWrite (32, LOW);    //Enciende el modulo
  delay (500);
  digitalWrite(34, LOW);    // Enable en bajo despues de prenderel módulo bluetooth  para poder entrar
al modo AT normal
  BT1.begin(baud);
  Serial.println("Levantando el módulo HC-06");
  Serial.println("Esperando comandos AT:");
  delay(1000);
}

void loop() {
  String comando = "", instru = "poa",com;         // Cadena de caracteres de prueba
  int valores[2], cont = 0;                                    // Una forma más "compacta" de crear
seis variables.
  int posPrevia = 0, posActual = 0;
byte i;
  ////Si hay error en la controladora entra a este bucle
  if (digitalRead(24) == 1) {
    digitalWrite(22, LOW);
    delay(200);
    digitalWrite(22, HIGH);
    delay(3000);
    home();
  }
  ////////////////////////////////////////////////////
  if (Serial.available()) {
    BT1.write(Serial.read());
  }
   BT1.write(1);
  /////PARA ESCRIBIR DIRECTAMENTE COMANDOS DEL PUERTO SERIE DESDE ALGUNA TERMINAL
  while (BT1.available()) {
    Serial.println("recibiendo");
    com = (BT1.readString());
    comando.concat(com);

    if (comando.indexOf("CONNECTED") >= 0) {
        Serial.println(comando);
      Serial.println("ya se conectó");
      BT1.write(1);
    }
   Serial.println(comando);

 while(instru.equals('poa')){
    //Bluce para interpretar la cadena recibida de Matlab

    posPrevia = comando.indexOf('>',posPrevia);                    // Buscar la posición de la primera
coma en la cadena
```

```
  instru = comando.substring(0, posPrevia);             // Extraer la primera cadena de caracteres (sin
coma)
    //Serial.println(instru);
    posPrevia++;                                         // Desplazar el índice de la primera coma. Sin
esto, la próxima búsqueda resultará
    // en la misma posición.
    posActual = comando.indexOf(',', posPrevia);     // La siguiente subcadena ya no empezará desde el
principio; por lo tanto, hay que almacenar dos posiciones.
    for ( i = 0; i < 2; i++) {
      valores[i] = comando.substring(posPrevia, posActual).toInt(); // Extrae la subcadena que contiene
un número, luego se convierte en un valor numérico de variable.
 //Serial.println("valor="+(String)valores[i]);
      posPrevia = posActual + 1;                          // desplazamiento de índice.
      posActual = comando.indexOf(',', posPrevia);                  // La "posición actual" ahora es la
próxima ocurrencia del "caracter separador"
    }

    if (instru.equals('pos')) {
      maestro.setTarget(valores[0], valores[1]);
    }

    if (instru.equals('acel')) {
      maestro.setAcceleration(valores[0], valores[1]);
    }

    if (instru.equals('vel')) {
      maestro.setAcceleration(valores[0], valores[1]);
    }
    /*
        for (byte i = 0; i < 2; i++) {    // Imprime el valor de las seis variables extraídas del texto
de prueba.
          Serial.println((valores[i]));
          //valores[i]=0;
        }
        */
   }

  }
}
///función que carga posición inicial del robot
void home(){
  ////POSICIONES INICIALES DEL ROBOT
  maestro.setTarget(1, 1750 * 4);
  maestro.setTarget(2, 1430 * 4);
  maestro.setTarget(3, 1700 * 4);
  maestro.setTarget(4, 1100 * 4);
  maestro.setTarget(5, 1900 * 4);
  maestro.setTarget(6, 1700 * 4);
  maestro.setTarget(7, 1850 * 4);
  maestro.setTarget(8, 1620 * 4);
  maestro.setTarget(9, 1900 * 4);
  maestro.setTarget(10, 1840 * 4);
  maestro.setTarget(11, 1950 * 4);
  maestro.setTarget(12, 2240 * 4);
  maestro.setTarget(13, 1900 * 4);
  maestro.setTarget(14, 900 * 4);
  maestro.setTarget(15, 850 * 4);
```

```
 maestro.setTarget(16, 900 * 4);
 maestro.setTarget(17, 3000 * 4);
 //maestro.setTarget(18, 1800 * 4);
 //maestro.setTarget(19, 1750 * 4);
 maestro.setTarget(20, 1550 * 4);
 maestro.setTarget(21, 680 * 4);
 maestro.setTarget(22, 1900 * 4);
 maestro.setTarget(23, 1450 * 4);
 ////////////////////////////////////
}
```