BRAZO ROBÓTICO CONTROLADO POR MEDIO DE VISIÓN COMPUTACIONAL UTILIZANDO UN KINECT

Celina Villicaña González

Universidad Panamericana

celinavg@up.edu.mx

María Teresa Orvañanos Guerrero

Universidad Panamericana

torvananos@up.edu.mx

Eduardo Rodríguez Figueroa

Universidad Panamericana

eduardorodriguez@up.edu.mx

Resumen

La presente investigación se centra en el desarrollo de un brazo robótico controlado por medio de reconocimiento de movimiento a través de los sensores integrados en un XBOX Kinect versión 1.8 y el SDK disponible para el entorno de desarrollo C#, el brazo robótico cuenta con 3 grados de libertad, los cuales son controlados por servomotores y un microprocesador ATmega programado en lenguaje C. La estructura del brazo robótico fue diseñada por computadora utilizando el programa de diseño asistido por computadora SolidWorks, posteriormente se realizó la estructura final utilizando una impresora 3D, eligiendo este método debido a la facilidad de impresión y el peso del material. Con la realización de este proyecto se pretende medir los alcances y tiempo de respuesta que el Kinect versión 1.8 nos permite para dar paso al desarrollo de algún dispositivo controlado por reconocimiento de movimiento el cual pueda ayudar en el área médica de la fisioterapia para realizar un control sobre el paciente más eficiente e inclusive más exacto.

Palabras Claves: Brazo robótico, esqueleto, modelo 3D, servomotor, solidworks, XBOX Kinect.

Abstract

The present research focuses on the development of a robotic arm controlled by motion recognition through the sensors integrated in an XBOX Kinect version 1.8 and the SDK available for the C# development environment, the robotic arm has 3 degrees of freedom which are controlled by servomotors and a ATmega microprocessor programmed in the C language. The structure of the robotic arm was designed by computer using the solid modeling computer-aided design and computer-aided engineering program SolidWorks, later the final structure was built using a 3D printer, choosing this method due to the ease of printing and the weight of the material. The propose of this research is to measure the reaction time that the Kinect version 1.8 allows us to give way to the development of some device controlled by recognition of movement which can help in the medical area of the physiotherapy to realize a control of the patient in a more efficient and accurate way.

Keywords**:** *Atmega, robotic arm, servomotor, skeleton, solidworks, XBOX Kinect, 3D Model.*

1. Introducción

El uso de los sensores de profundidad en el ámbito de la robótica es un tema muy controversial y extenso, en el artículo [Kefer, 2011] se realiza una comparación entre los sensores de profundidad del Kinect y la información de profundidad otorgada por una cámara estereoscópica, con lo cual se llega a una mejoría en la información obtenida por el Kinect, cabe mencionar que en el artículo el SDK no había sido liberado para los desarrolladores por lo cual los métodos de obtención del Kinect no estaban optimizados, por lo tanto se podría deducir que el Kinect aporta mejores datos de la profundidad.

En el ámbito de la optimización industrial la mayoría de los enfoques dados a esta son el seguimiento de la línea de producción automatizada la cual, por lo general, se enfoca directamente en los productos más que en los trabajos realizados por personas. En el artículo [Lipovits, 2016] es realizada una investigación utilizando el Kinect para el reconocimiento de gesto realizado por los trabajadores en la línea

de producción, con la cual se busca la optimización de la misma por medio de algoritmos de minería de datos. En este artículo se realizará un brazo robótico el cual posea 3 grados de libertad con lo cual se buscará imitar el movimiento de un brazo humano, para ello se utilizó un Xbox Kinect debido a la gran gama de sensores y datos que este nos facilita además del SDK que Microsoft proporciona con la plataforma Kinect para desarrolladores.

Si bien el área de investigación del Kinect como ayuda o soporte en la ámbito de la rehabilitación de pacientes no es un tema extensamente investigado, se han realizado investigaciones relacionadas como es el caso de la investigación realizada por John E. Muñoz-Cardona, Oscar A. Henao-Gallo y José F. López-Herrera en la que "muestra la creación de un novedoso sistema para la rehabilitación física de pacientes con múltiples patologías, a través de dinámicas con videojuegos de ejercicio (exergames) y el análisis de los movimientos de los pacientes usando un software desarrollado" [Muños,2013].

2. Métodos

Para el desarrollo de este proyecto, se hizo uso del "*Kinect for Windows SDK*" versión 1.8, el cual permite obtener información mediante los diferentes sensores y cámaras con las que cuenta el Kinect, este dispositivo es capaz de procesar color y profundidad además de identificar el esqueleto humano y obtener datos de profundidad relativa entre el objeto y el Kinect.

Las cámaras y sensores con las que cuenta el dispositivo Kinect son:

- Una cámara RGB con una resolución de 1280x960
- Un emisor infrarrojo y un sensor infrarrojo de profundidad, el emisor emite la luz infrarroja y el sensor de profundidad lee la reflexión, calculando con esta la profundidad del esqueleto.
- Un arreglo de 4 micrófonos para capturar el sonido.
- Un acelerómetro de 3 ejes, el cual permite saber la posición del dispositivo, además de ajustar la misma conforme a la configuración establecida.

Para hacer uso de los múltiples sensores que el Kinect nos ofrece, lo primero que sé que se realizó es obtener el estado de nuestro Kinect, ya que se pudiera dar el

caso de una falla de hardware, con la cual se perdería la comunicación con este y por lo tanto no se podrían obtener los datos deseados, dichos estados se encuentran en el enumerador "*KinectStatus*". Para la monitorización del estado de nuestro Kinect el SDK provee la clase "*KinectSensor*". En la cual se pueden acceder a los múltiples Kinects, si se diera el caso, que se encuentran configurados y conectados a nuestro ordenador, la propiedad que nos facilita dicha información es "*KinectSensors*" de la misma clase "*KinectSensor*". En este trabajo solamente se utilizará un Kinect, por lo tanto, si es que se encontraban conectados distintos Kinects solo uno de estos fue utilizado [Microsoft Corporation, 2016]. Una vez que se obtenga el sensor Kinect conectado a nuestro ordenador, se procede a habilitar los diferentes sensores con los cuales se realiza un "streaming" de los datos que estos proporcionan. Los datos los cuales pueden ser obtenidos por nuestros diferentes sensores son: color, profundidad, esqueleto e infrarrojo. Para habilitar los sensores y el streaming de los datos deseados, se habilitan directamente en nuestro sensor Kinect que fue obtenido con anterioridad. Para este trabajo solamente se habilitarán los sensores de esqueleto y de profundidad, en los cuales se utilizan las clases "*SkeletonStream*" y "*DepthStream*" ambas contenidas en nuestra clase "*KinectSensor*" [Microsoft Corporation, 2016].

Una vez configurado el tipo de sensores que utilizara el Kinect, procedemos a configurar el tipo de rastreo de esqueleto que nuestro Kinect utilizara para sensar el esqueleto, ya que este ofrece dos formas de rastreo las cuales consisten en si la persona a sensar se encuentra sentado o de pie, a las cuales se refiere como modo sentado y modo por defecto respectivamente. Las diferencias y similitudes de sensar, ya sea en modo sentado o modo por defecto, se tomarán de la documentación ofrecida en [Microsoft Corporation, 2016]:

- El modo por defecto rastrea 20 articulaciones esqueléticas diferentes mientras tanto en el modo sentado solo se rastrearán 10 articulaciones siendo estas las del cuerpo superior.
- El modo por defecto detecta al usuario en función de la distancia del sujeto al fondo, en el modo sentado utiliza el movimiento del usuario para detectarlo y distinguirlo del fondo.

- Para que el usuario sea reconocido por el kinect, en el modo por defecto, simplemente se necesita estar frente al kinect, mientras tanto, en el modo sentado, es necesario que el usuario comience a moverse para este ser identificado.
- En el modo por defecto no es aconsejable rastrar usuarios que se encuentren sentados, mientras que en el modo sentado los usuarios parados pueden ser identificados, pero aun así en este modo el usuario necesita mover sus extremidades para ser identificado.
- En el modo sentado, en el procesamiento de los datos difiere en el módulo de segmentación, con respecto al modo predeterminado, debido a que los datos o imagen solo por el hecho de encontrarse sentado contienen mayor ruido y variabilidad.
- El modo sentado utiliza mayores recursos, además de demorar el procesamiento de cada una de las capas, por lo tanto, disminuye el número de cuadros por segundo de la imagen.
- El modo sentado provee una mayor fiabilidad al momento de reconocer esqueletos, siempre y cuando el kinect se encuentre en el modo de rango cercano.
- Ambos modos de rastre solamente pueden seguir el rastro de 2 personas.
- Solo un modo de rastreo puede ser utilizado a la vez.
- Si se cuentan con dos Kinects conectados a la misma computadora, siempre y cuando se encuentren en un proceso diferente, estos pueden ser configurados en diferentes modos de rastreo.

Dadas las similitudes y diferencias que hay entre los dos modos que nos ofrece el SDK del Kinect, deducimos que el mejor modo a utilizar sería el modo sentado utilizando el modo de rango cercano, ya que si bien el modo sentado utiliza un mayor consumo de recursos este realiza un mejor rastreo en cuanto al modo por default.

Una vez configurado el modo de rastreo que se utilizara, se añade un método a la serie de delegados que nuestro sensor Kinect pose en la propiedad

SkeletonFrameReady la cual es invocada en cada cuadro en el que el esqueleto sea, independientemente si fue encontrado uno o no, buscado. Posteriormente iniciamos la adquisición de los sensores por medio del método Start del sensor Kinect.

Obtención de Ángulos de las Articulaciones

Para obtener los ángulos que se dan en las principales articulaciones del brazo se desarrolló un software en el lenguaje de programación C# el cual permite procesar los datos obtenidos del Kinect por medio de las librerías que su SDK ofrece, además de utilizar la comunicación serial a través del puerto USB el cual permite la comunicación con nuestro sistema de control, la comunicación será detallada posteriormente.

Para poder replicar el movimiento de un brazo es necesario conocer al menos 3 de los ángulos que se forman en las principales articulaciones, el primero de ellos está comprendido como el ángulo que se forma entre el hombro, el codo y la muñeca figura 1 que puede ser considerado como el ángulo medido desde una vista frontal del brazo, el segundo ángulo se obtiene con las mismas articulaciones que el primero pero para el cálculo del ángulo se utilizó la profundidad de las articulaciones, de ahí se obtiene el ángulo desde una vista superior, el tercer ángulo está comprendido entre el codo, la muñeca y la mano, figura 2.

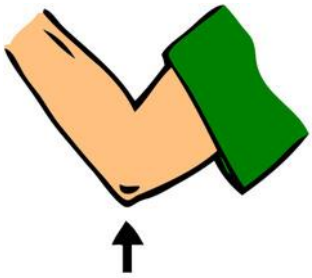

Figura 1 Localización primer ángulo usado.

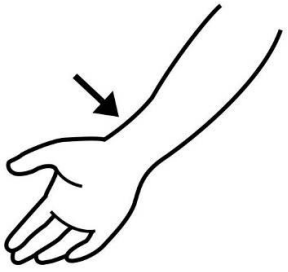

Figura 2 Localización segundo ángulo usado.

Para obtener el primer y tercer ángulo se realizó el mismo procedimiento, ya que solo era necesario hacer un cálculo con las posiciones de las articulaciones entregadas por el Kinect, y en el caso del segundo ángulo, fue necesario realizar

operaciones con la profundidad, para poder localizar el antebrazo y obtener su posición. El procedimiento para la obtención de los ángulos fue el siguiente:

- Obtener 3 distintas articulaciones, para el primer y segundo ángulo se usa: hombro, codo, y muñeca, y para el tercer ángulo: codo, muñeca y mano, los cuales se nombrarán punto A, B y C respectivamente.
- Cada articulación cuenta con 3 componentes, X(Anchura), Y(Altura) y Z(Profundidad), las cuales obtiene el Kinect, para cada vector se tomará en cuenta solo 2 componentes, en este caso X y Y. Para el segundo ángulo se tomarán en cuenta solo las componentes de X y Z.
- Para el primer y segundo caso se obtienen dos vectores resultantes, el primero de ellos es la resta de los vectores A y B, el segundo vector de la resta de B y C. Cada uno de estos vectores son conformados por las 2 componentes X y Y dejando a Z en 0 como se muestra en ecuación 1.

$$\begin{gathered}\overrightarrow{AB} = \vec{B} - \vec{A} \\ \overrightarrow{AB} = \left(B_x i + B_y j + B_z k\right) - \left(A_x i + A_y j + A_z k\right) \\ \overrightarrow{AB} = (B_x - A_x)i + \left(B_y - A_y\right)j + (0-0)k \\ \overrightarrow{BC} = \vec{C} - \vec{B} \\ \overrightarrow{BC} = \left(C_x i + C_y j + C_z k\right) - \left(B_x i + B_y j + B_z k\right) \\ \overrightarrow{BC} = (C_x - B_x)i + \left(C_y - B_y\right)j + (0-0)k\end{gathered} \tag{1}$$

- Una vez que se obtienen los vectores $\overrightarrow{AB}$ y $\overrightarrow{BC}$ se procederá a normalizarlos, ecuación 2.

$$\begin{aligned}\overrightarrow{nAB} &= \left(\frac{AB_x}{\sqrt{{AB_x}^2 + {AB_y}^2 + {AB_z}^2}}\right)i + \left(\frac{AB_y}{\sqrt{{AB_x}^2 + {AB_y}^2 + {AB_z}^2}}\right)j + \left(\frac{AB_z}{\sqrt{{AB_x}^2 + {AB_y}^2 + {AB_z}^2}}\right)k \\ \overrightarrow{nBC} &= \left(\frac{BC_x}{\sqrt{{BC_x}^2 + {BC_y}^2 + {BC_z}^2}}\right)i + \left(\frac{BC_y}{\sqrt{{BC_x}^2 + {BC_y}^2 + {BC_z}^2}}\right)j + \left(\frac{BC_z}{\sqrt{{BC_x}^2 + {BC_y}^2 + {BC_z}^2}}\right)k\end{aligned} \tag{2}$$

- Después se obtiene el producto cruz de los vectores $\overrightarrow{nAB}$ y $\overrightarrow{nBC}$, ecuaciones 3 a la 5.

$$\begin{aligned}\overrightarrow{nAB \times nBC} &= \overrightarrow{nAB} \times \overrightarrow{nBC} \\ \overrightarrow{nAB \times nBC} &= \left(nAB_x i + nAB_y j + nAB_z k\right) \times \left(nBC_x i + nBC_y j + nBC_z k\right) \\ \overrightarrow{nAB \times nBC} &= nAB_x nBC_x(i \times i) + nAB_x nBC_y(i \times j) + nAB_x nBC_z(i \times k) \\ &\quad + nAB_y nBC_x(j \times i) + nAB_y nBC_y(j \times j) + nAB_y nBC_z(j \times k) \\ &\quad + nAB_z nBC_x(k \times i) + nAB_z nBC_y(k \times j) + nAB_z nBC_z(k \times k)\end{aligned} \tag{3}$$

Donde:

$$\begin{aligned} i \times i = 0 \quad i \times j = k \quad i \times k = -j \\ j \times i = -k \quad j \times j = 0 \quad j \times k = i \\ k \times i = j \quad k \times j = -i \quad k \times k = 0 \end{aligned} \tag{4}$$

Por lo tanto:

$$\begin{aligned} \overrightarrow{nAB \times nBC} = nAB_x nBC_x(0) + nAB_x nBC_y(k) + nAB_x nBC_z(-j) + nAB_y nBC_x(-k) \\ + nAB_y nBC_y(0) + nAB_y nBC_z(i) + nAB_z nBC_x(j) + nAB_z nBC_y(-i) \\ + nAB_z nBC_z(0) \end{aligned} \tag{5}$$

- Una vez que se obtiene el producto cruz se obtiene el producto punto de los vectores $\overrightarrow{nAB}$ y $\overrightarrow{nBC}$, ecuación 6.

$$\begin{aligned} \overrightarrow{nAB} \cdot \overrightarrow{nBC} = \left(nAB_x i + nAB_y j + nAB_z k\right) \cdot \left(nBC_x i + nBC_y j + nBC_z k\right) \\ \overrightarrow{nAB} \cdot \overrightarrow{nBC} = (nAB_x)(nBC_x) + \left(nAB_y\right)\left(nBC_y\right) + (nAB_z)(nBC_z) \end{aligned} \tag{6}$$

- Además, se utiliza la función atan2, ecuación 7.

$$atan2(y,x) = \begin{cases} \tan^{-1}\left(\frac{y}{x}\right) & si \quad x > 0, \\ \frac{\pi}{2} - \tan^{-1}\left(\frac{x}{y}\right) & si \quad y > 0, \\ -\frac{\pi}{2} - \tan^{-1}\left(\frac{x}{y}\right) & si \quad y < 0, \\ \tan^{-1}\left(\frac{y}{x}\right) \pm \pi & si \quad x < 0, \\ indefinido & si \quad x = 0 \quad y = 0 \end{cases} \tag{7}$$

- Una vez que se obtiene el producto punto y el producto cruz se toma el resultado en la componente z del producto cruz y el resultado del producto punto a los cuales se les aplicara la función atan2 para la obtención del ángulo en radianes, ecuación 8.

$$\forall R = atan2\left(\overrightarrow{nAB \times nBC}, \overrightarrow{nAB} \cdot \overrightarrow{nBC}\right) \tag{8}$$

- Para obtener el tercer ángulo se seguirá el mismo proceso solo se modifica ecuación 1, quedando ecuación 9.

$$\begin{aligned} \overrightarrow{AB} = \vec{B} - \vec{A} \\ \overrightarrow{AB} = \left(B_x i + B_y j + B_z k\right) - \left(A_x i + A_y j + A_z k\right) \\ \overrightarrow{AB} = (B_x - A_x)i + (B_z - A_z)j + (0 - 0)k \end{aligned} \tag{9}$$

$$\overrightarrow{BC} = \vec{C} - \vec{B}$$

$$\overrightarrow{BC} = (C_x i + C_y j + C_z k) - (B_x i + B_y j + B_z k)$$

$$\overrightarrow{BC} = (C_x - B_x)i + (C_z - B_z)j + (0 - 0)k$$

Para dichos cálculos se recurrió al artículo [Wilson, 2012] en el cual se profundiza acerca de la obtención de ángulos utilizando las articulaciones con el mismo sistema Kinect.

PWM Ángulos y Pulsos

Para hacer uso de los servomotores, se usan los tres Timers del microprocesador Atmega16, los cuales cuentan con un PWM (Pulse-width modulation o modulación por ancho de pulsos) la función del PWM es modificar el ciclo de trabajo de una señal periódica, en este caso cuadrada.

El modo Fast PWM utiliza los registros OCRn y TCCRn de cada Timer dentro del microprocesador, el registro TCCRn es el encargado de contar de 0 a 255 cada ciclo de reloj, el registro OCRn tendrá un valor entre 0 y 255, y cuando el valor de TCCRn y OCRn sean iguales, el pulso bajara a un cero lógico (0 Volts) o subirá a un uno lógico (5 Volts) según se tenga configurado (si se inicia en alto o si se inicia en bajo), cuando el valor del registro TCCRn llegue a 255 y se reinicie en cero, se invertirá la señal de nuevo, es decir, si la señal se encontraba en un uno lógico, volverá a bajar a un cero lógico y viceversa, figura 3.

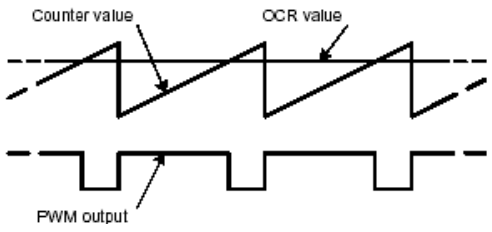

Figura 3 Salida de PWM y registros OCR y TCCR.

La ecuación 10 es la que se utiliza para calcular la frecuencia que tendrá la señal generada, donde $frecuencia_{clk}$ es la frecuencia del microprocesador la cual para este proyecto fue de 1 MHz y N es el preescaler seleccionado.

$$frecuencia_{pwm} = \frac{frecuencia_{clk}}{256 \cdot N} \quad (10)$$

Para controlar un servomotor, se debe generar un periodo de entre diez milisegundos y 30 milisegundos, el tiempo que la señal se mantiene en alto es lo que indica el ángulo con el girará el motor, los valores del pulso mínimo y máximo, que equivalen a 0º y 180º son un milisegundo y dos milisegundos de tiempo en alto, cuando se tiene un periodo de 10 ms.

$$T = \frac{1}{Frecuencia} \quad (11)$$

Con el microprocesador Atmega16, el periodo más cercano que se pudo obtener fue de 61 MHz, usando un preescaler de 64, por lo tanto, se tendrá una frecuencia de 16.30 milisegundos.

Para el cálculo del valor de OCRn, se midió que el ángulo de 0º se obtenía con un valor de 9 en el OCRn, el ángulo de 90º se obtenía con un valor de 22 en el OCRn, por lo tanto se midió que cada 6.73º se obtenía un incremento de 1 en el OCRn, y se llegó a ecuación 12.

$$OCR = (\frac{grados}{6.7307} + 9) \quad (12)$$

Para la configuración de cada Timer del microprocesador se seguirán los pasos mostrados continuación:

- Para el Timer 0:
  - ✓ Cargar en el registro TCCR0 el valor binario 0110101.
  - ✓ Debido a la configuración anterior, los bits WGM0 y WGM1 contienen un 11 binario, lo cual indica que el timer está en modo Fast PWM.
  - ✓ Los bits COM01 y COM00 tienen un 10 binario por lo tanto cuando el registro TCNT0 equivalga al registro OCR0, la señal de PWM bajará y se mantendrá de esta forma hasta que el contador vuelva a iniciar en 0
  - ✓ Se seleccionó un preescaler de 64, por lo cual en los bits CS00, CS01 y CS02 se tiene un 110 binario.

- Para el Timer 1A:
  - ✓ Se carga en el registro TCCR1A el valor binario 10000001 y en el registro TCCR1B el valor 00001011.
  - ✓ Los bits WGM13, WGM12, WGM11 y WGM10 que contienen un valor binario de 0101, indican que se usara el timer 1 en modo fast PWM usando solo 8 bits.
  - ✓ Los bits COM1A1 y COM1A0 que contiene un valor de 10, indican se iniciara con una señal en alto y al llegar el registro TCNT1 al valor del registro OCR1AL (este timer es de 16 bits, pero para este proyecto solo se usaran 8 bits de este timer, por lo tanto, solo se usan los primeros 8 bits el registro), la señal bajara.
  - ✓ Se seleccionó un preescaler de 64, por lo cual en los bits CS10, CS11 y CS12 se tiene un 110 binario.
- Para el Timer 2:
  - ✓ Cargar en el registro TCCR2 el valor binario 01101100.
  - ✓ Los bits WGM20 y WGM21 contienen un 11, al igual que en el Timer1, para indicar el modo Fast PWM.
  - ✓ Al igual que en los Timers anteriores, los bits COM21 y COM20 contienen un valor binario de 10.
  - ✓ Se seleccionó un preescaler de 64, por lo cual en los bits CS20, CS21 y CS22 se tiene un 100 binario.

Todas las configuraciones descritas fueron realizadas en base a lo especificado en [Atmel Corporation,2016].

Comunicación Serial

Una parte esencial de este proyecto fue la comunicación entre la computadora y el microprocesador, en este caso, la computadora obtenía los datos provenientes del Kinect, los procesa para encontrar los ángulos entre las articulaciones seleccionadas, pero el microprocesador requiere esa información para generar los pulsos que controlarán a cada servomotor.

Para esto es necesaria la comunicación serial asíncrona, donde cada dispositivo envía un dato en serie, y cada dispositivo tiene su propio reloj, cuando se envía un dato primero se envía un bit de inicio, los datos, y por último los bits de parada, estos bits se pueden configurar, tanto la cantidad de bits de los datos, como si son uno o dos bits de parada, además se puede configurar la paridad, es decir, si se configura con paridad, se puede definir que sea impar o par, esto para verificar que los datos se envíen correctos y completos. Para este caso se configuro un bit de inicio, ocho bits de datos, 1 bit de parada, y sin paridad, esto se define cargando en los registros necesarios los valores para la configuración que son:

- Para el registro UCSRA, se carga un 0010000 binario, esto indica que está listo para recibir datos.
- Para el registro UCSRB se carga un 10011000 binario, esto indica que se recibirá una interrupción cada que se recibe un dato, y además habilita la recepción y el envió de datos.
- Para el registro UCSRC se carga un 10000110, lo cual indica que los bits que se estarán enviando son 8, modo asíncrono y un bit de parada.
- Los registros UBRRH y UBRRL se configura la tasa de transferencia con la que se trabajará, en este caso será a 4800, por lo tanto, en el UBRRH contiene un 0 y el UBRRL contiene un 12 decimal.

Para que el microprocesador identificara que motor se moverá y cuanto, se le envía primero una letra (‘A’, ‘B’, ’C’) y el microprocesador respondía al recibirla para que la computadora ahora enviará el ángulo que se moverá.

Todas las configuraciones descritas fueron realizadas en base a lo especificado en [Atmel Corporation,2016].

Modelado de Prototipo

Para la parte física del robot, se usó el programa SolidWorks para diseñar las piezas, las cuales fueron 6 en total, ya que se consideraron un cuarto grado de libertad para su desarrollo a futuro, figura 4.

Para el tercer grado de movimiento (movimiento del antebrazo sobre un plano respecto al brazo), se colocó un motor dentro de una caja que lo sostenía, dicha

caja constaba de dos piezas, una base y una tapa por donde sale el eje motriz del motor, figura 5.

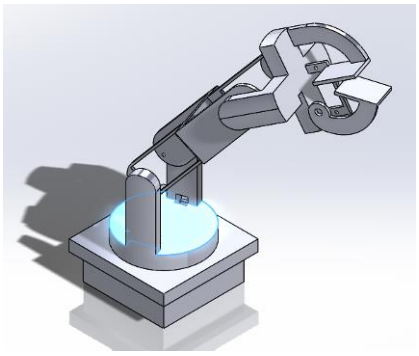

Figura 4 Ensamblaje del modelo completo.

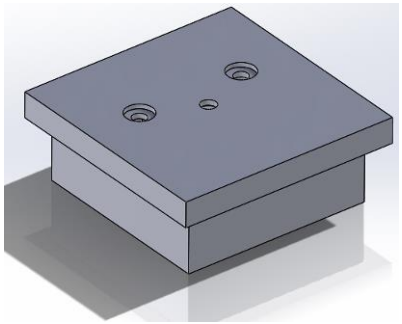

Figura 5 Ensamblaje de la caja base.

Sobre esta caja va montado una base donde encaja el eje motriz del motor que simulará el movimiento del codo, figura 6.

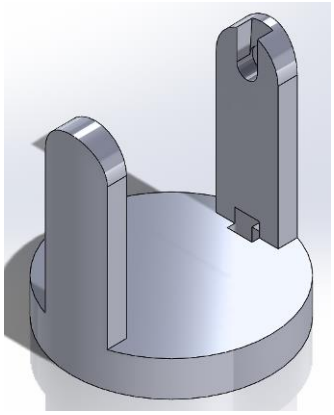

Figura 6 Base para el motor de movimiento del codo.

A esta parte se une la pieza con la que formara el antebrazo, figura 7, quien tendrá espacio para posicionar dos servomotores, cuyos ejes motores irán unidos a la base anterior, y a la parte que simulara la mano, figura 8, respectivamente.

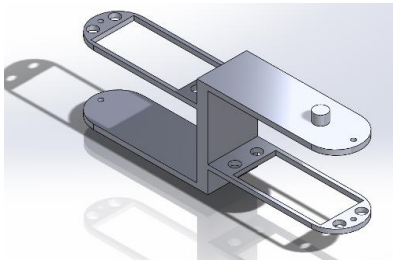

Figura 7 Antebrazo.

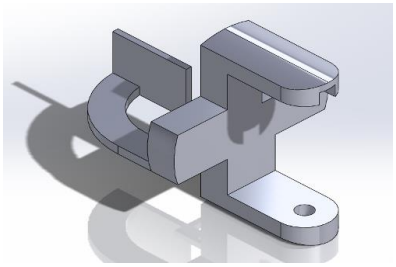

Figura 8 Mano.

3. Resultados

Los resultados del proyecto no fueron los esperados, sin embargo, estos fueron muy prometedores. Estos se dividen en dos partes esenciales, las cuales constan de, el sistema de comunicación y manipulación de los servomotores, figura 9 y figura10, el sistema de obtención de articulaciones, figura 11.

Figura 9 Estructura final del brazo.

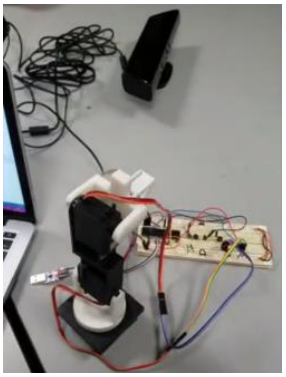

Figura 10 Sistema de comunicación y manipulación de los servomotores.

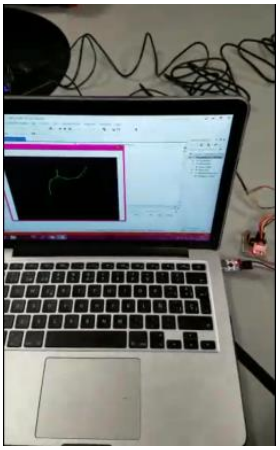

Figura 11 Sistema de obtención de articulaciones.

En el sistema de comunicación y manipulación se obtuvo una comunicación en tiempo real entre estos dos. Consiguiendo así que el movimiento de los motores fuera el más preciso posible de acuerdo con los ángulos obtenidos con nuestro algoritmo.

En la parte de nuestro sistema de obtención de articulaciones se obtuvo de manera exitosa el ángulo en los casos en los cuales nuestro algoritmo no utilizaba la profundidad de las articulaciones, debido a que nuestro sistema de obtención de articulaciones mostraba un error muy variable en la obtención de la profundidad, sin embargo, no debe haber ninguna interferencia entre el usuario y el dispositivo Kinect, ya que, si esto ocurre, el Kinect deja de captar los movimientos correctamente, además debido a la limitante del Kinect versión uno, no se pudo incluir un grado extra de libertad.

4. Discusión

A pesar de que los movimientos generados por el brazo robótico eran muy similares a los que el usuario hacía, estos podrían ser más exactos si se usa una versión más nueva del dispositivo Kinect ya que este cuenta con más y mejores sensores que entregan datos más exactos y verídicos, además de que gracias a esta nueva versión se podrían generar más grados de libertad, los cuales ayudarían a que este proyecto tenga más aplicaciones en el campo de la industria y de la producción.

5. Conclusiones

Se puede concluir que el dispositivo Kinect tiene una amplia variedad de usos gracias a la gran cantidad de información que este nos puede proporcionar y a su fácil implementación, uno de estos fue la investigación realizada en este artículo.

El trabajo realizado en este artículo nos abre las posibilidades de trabajos futuros utilizando el mismo concepto de movimiento de articulaciones para el desarrollo de algún robot o mecanismo, ya sea en un ámbito de rescate o industrialización, por mencionar algunos. Esto puede ser posible dado al tipo de comunicación utilizada en él, ya que, con unos componentes extras, por ejemplo, módulos Bluetooth, nos provee una comunicación de mayor distancia. No obstante, para mejorar la detección de los puntos de interés del esqueleto se recomendaría utilizar la versión más reciente del Kinect (Kefer & Kubinger, 2011).

6. Bibliografía y Referencias

[1] Dayra, M. Como redactar y publicar artículos científicos. Organización Panamericana de Salud. Monterrey, México. 1994.

[2] Atmel Corporation. (s.f.). 8 bit AVR Microcontroller with 16K Bytes In-System Programmable Flash: http://www.atmel.com/images/doc2466.pdf.

[3] Kefer, M., & Kubinger, W., EVALUATION OF KINECT DEPTH SENSOR FOR USE IN MOBILE ROBOTICS. Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium (págs. 147-148). Vienna: DAAAM International, 2011.

[4] Lipovits, Á., Gál, M., Kiss, P. J., & Süveges, C., Human Motion Detection in Manufacturing Process. 2nd World Congress on Electrical Engineering and Computer Systems and Science, Budapest: Electrical Engineering and Computer Systems and Science, pp. 111-118, 2016.

[5] Microsoft Corporation. (s.f.). Kinect for Windows Sensor Components and Specifications: https://msdn.microsoft.com/en-us/library/jj131033.aspx.

[6] Microsoft Corporation, Tracking Modes (Seated and Default): https://msdn.microsoft.com/en-us/library/hh973077.aspx.

[7] Muños Cardona, J. E., Henao Gallo, O. A., & López Herrea, J. F., Sistema de Rehabilitación basado en el Uso de Análisis Biomecánico y Videojuegos mediante el Sensor Kinect, Tecno Lógicas, pp. 43-54, 2013.

[8] Wilson, J. Y., Robotics, embedded101, http://www.embedded101.com/Blogs /James-Y-Wilson/entry id/167/Default.