GUARDA-CONTROL DE SISTEMAS MECANICOS PARA PREVENCION DE ACCIDENTES

Estrada Rentería Carlos de J, Quintero Rosas Verónica, Díaz-Ramírez Arnoldo

Departamento de Sistemas y Computación
Instituto Tecnológico de Mexicali
Av. Tecnológico S/N, Col Elías Calles, Mexicali, B.C., México, 21376
estradacdj@gmail.com, veroquintero@itmexicali.edu.mx, adiaz@itmexicali.edu.mx

RESUMEN

En la medida en que la población del mundo ha migrado de zonas rurales a las ciudades, la economía mundial depende más de las empresas de manufactura y a la tecnología. Sin embargo, el crecimiento de estas industrias ha venido acompañado con un incremento en el número de accidentes laborales. En México, de acuerdo a estadísticas de la Dirección General de Seguridad y Salud en el Trabajo, los accidentes reportados mediante el Instituto Mexicano del Seguro Social han ido en incremento en los últimos años, dando como resultados incapacidades temporales, permanentes y defunciones. En este artículo, se propone una guarda de control de un sistema mecánico para la prevención de accidentes en la industria, que utiliza un brazo robótico que imita el movimiento del brazo humano. El sistema controla el movimiento del brazo del usuario a distancia, para evitar el contacto directo del operario y cualquier maquinaria. Para evaluar la funcionalidad del modelo propuesto se construyó un prototipo usando una cámara Kinect. Se presentan detalles de la implementación y evaluación del prototipo.

Palabras claves: Prevención de accidentes laborales, Control mecánico remoto, Arduino, Kinect, Android

ABSTRACT.

As worldwide urban population increases, global economy has depended on high technology devices developed by several manufacturing companies. A consequence of this is the increasing number of accidents at the workplace, and their effects on employees such as disabilities and deaths. Statics from the General Direction for Safety and Health at the Workplace have also supported this issue in Mexico, throughout the Mexican Institute of Social Security. In this work, we propose a robotic arm controlled at a distance by a Kinect camera, which mimics human arm movements, and aims to avoid contact between machinery and employees. The details of implementation and evaluation are presented.

Keywords: Prevention of accidents, mechanical remote control, Arduino, Kinect, Android.

INTRODUCCIÓN

El crecimiento de la industria de la manufactura ha sido impresionante en los últimos años. Sin embargo, este crecimiento ha venido acompañado con un incremento en la tasa de accidentes laborales.

Por ejemplo, en el año 2013, el número de accidentes en la industria a nivel mundial fue de 1,20 millones, con un promedio de 3000 muertes diarias, según cifras del International Labour Organization ILO. En México, de acuerdo a cifras del Instituto Mexicano del Seguro Social [20], en el año 2012 se tuvieron 35,780 accidentes laborales. La Fig. 1, muestra el número de accidentes en la industria en los años recientes.

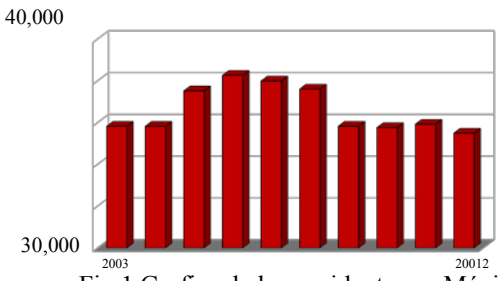

Fig.1 Grafica de los accidentes en México

Para abatir este problema se ha propuesto el uso de la tecnología en los procesos de manufactura, y en especial el uso de robots. En la actualidad existe una gran variedad de robots industriales disponibles en el mercado, que pueden tener aplicaciones tales como: soldadura, pintado, corte, prensado etc. Una característica de este tipo de robots es que solo es necesario cambiar actuadores para ser utilizados en diferentes actividades que puedan ser peligrosas para una persona, como puede observarse en la Fig. 2. Sin embargo, es común que a pesar del uso de esta tecnología para trabajos especiales, aún se requiere que el usuario use el robot por medio de un control manual para llevar a cabo el proceso de manufactura [12].

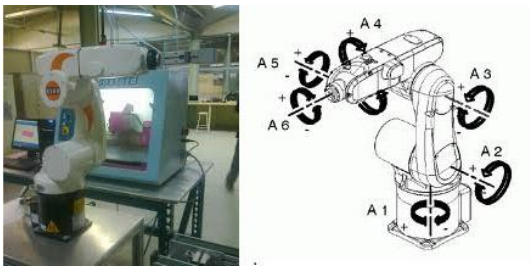

Fig.2 Ejemplo de Robot Industrial Kuka

¿Qué es un Guarda-Control?

El propósito de la colocación de guardas, es establecer un adecuado y correcto confinamiento del punto de peligro en maquinaria y equipo. Las instalaciones actuales y los proyectos futuros que involucren guardas y/o *coples* mecánicos deben cumplir estrictamente lo establecido. Por guardas a equipos se entiende un mecanismo capaz de prevenir que una persona introduzca parte de su cuerpo en la zona de riesgo, durante el ciclo de operación del equipo [13]. Puesto que a menudo se requiere llevar a cabo el control a distancia de sistemas mecánicos, éste servirá como guarda al operador, al evitar estar cerca del proceso lo que impide un peligro para el mismo.

En este artículo se propone un sistema de control para sistemas mecánicos dentro de la industria, con la finalidad de hacer mas segura la interacción de los operadores con las máquinas, eliminando o reduciendo la probabilidad de accidentes laborales. Para ello se diseñó un control *kinematico* y una aplicación móvil, la cual permite la interacción con el sistema, utilizando herramientas libres para la manipulación remota de sus movimientos. Dado a que en la mayoría de las industrias se opta por usar robots con cinco a mayor grados de libertad [3], se diseñó un control que utiliza un brazo robótico de 6 grados de libertad, que está limitado de 0-180° por los servomotores contenidos en el brazo, con eslabones fijos de metal y una pinza de dos dedos [14]. Y como medio de control a distancia, se utiliza un control, por medio de mímica para que el usuario mueva el robot tal como mueva su mano para que así; el movimiento sea más natural. Para este control se utiliza una cámara para que capte la información del usuario para utilizarla como comando para el robot y su movimiento. Para validar la funcionalidad del modelo propuesto, se construyó un prototipo utilizando la cámara de *Kinect* [5] que para el desarrollo de este control satisface los requerimientos del sistema, y a pesar de que que su resolución no es tan alta como algunas cámaras de uso industrial, la evaluación del prototipo muestra que se obtiene un buen desempeño.
Se utilizó también el kit de desarrollo (*Software Development Kit* o *SDK*) de *Microsoft* debido a su facilidad para programar y configurar la cámara *Kinect*. Además, tiene la ventaja de contar con el módulo denominado *Skeletal track,* para la imagen de una o dos personas que se estén moviendo dentro del campo de visión. El prototipo recibe instrucciones establecidas mediante el movimiento físico para mover el robot. También se tiene la opción de interactuar mediante una aplicación móvil desarrollada en *Android* [15]. Todo esto enlazado mediante un microcontrolador que define los movimientos del robot según las instrucciones dadas en el celular o en la cámara *Kinect* [10]. Ya que se tiene una manera de captar datos, se utiliza un microcontrolador. En este caso por razones académicas se utilizó Arduino [6], por su entorno de programación sencilla pero versátil. Una ventaja es que gracias a sus bibliotecas se facilita la interacción de varios actuadores.

TRABAJO PREVIO

Existen varias investigaciones utilizando interfaces hombre-máquina [2] [4], desde juegos con controles vía acelerómetro de celulares, controles utilizando *Nintendo Wii*, hasta equipos industriales de alto precio, utilizan interfaces que capturan el movimiento del cuerpo para controlar [3], comunicar y dar acciones a distintos dispositivos mecánicos, electrónicos y de comunicación. La secuencia de movimientos de un robot en general, se modela mediante parámetros [1] [2], como la mecánica y dinámica del sistema. De estos resultados se deriva el análisis matemático. Existen varias técnicas para

dicho control, una de las más utilizadas es el uso de cámaras para detectar articulaciones del cuerpo humano y estas a su vez actuar como el control en tiempo real para mover un robot o cualquier sistema mecánico.

Puede ser complejo derivar el análisis matemático y hacer un análisis mecánico y dinámico, pero este artículo propone controlar mediante un sistema informático, cualquier sistema mecánico, que incluye la utilización de la cámara *Kinect* [5] para definir la acción y *Arduino* [8] para traducir dichas acciones a señales de salida hacia motores, pistones, microcontroladores, tarjetas, Plc's, etc. En este artículo además de utilizar la cámara Kinect para definir la acción del sistema mecánico, también se utiliza una aplicación móvil que fue desarrollada utilizando *App inventor* (*Android*) para que desde cualquier parte del mundo mediante un dispositivo móvil, manipular una acción.

Es importante señalar que para utilizarlo en un sistema en particular; tan solo es necesario cambiar los parámetros de la aplicación para definir las señales de salida hacia el sistema. Tanto la cámara *Kinect* como la aplicación *Android* tienen como interfaz de acción a la tarjeta *Arduino*.

Una alternativa para el desarrollo del prototipo es haber utilizado la tarjeta *Raspberry Pi* [21], pero para fines propiamente mecánicos, no era la mejor opción debido a que *Arduino* maneja señales analógicas y *Raspberry Pi* se especializa en señales digitales. Nuestra propuesta, a diferencia de otras [11] [4], consiste en que permite la tele operación de cualquier sistema mecánico, no únicamente de un brazo robótico.

Existen algunas investigaciones [11], donde se explica el control Master-Slave, el cual es la más fiable y eficiente, pero es costoso en términos de dispositivos maestros.

Los comandos de voz son rígidos, y vulnerable al ruido. Los gestos, por otro lado, no tienen un lenguaje en particular, son universales y fáciles de entender y utilizar. Pueden ser detectados mediante sensores montados en el cuerpo como guantes, exoesqueletos, que son bastante caros. Así que la alternativa más barata es mediante la visualización de cuerpo del operador con cámara *Kinect*.

MODELO PROPUESTO

El modelo propuesto se muestra en la Fig. 3. Como puede observarse, los componentes del modelo son: Operador, Control, comunicación y Sistema Mecánico. Mismos que se describen a continuación.

El componente Operador: Se refiere a la persona operaria del sistema mecánico.

El componte Control: Es el componente que aísla al operario del sistema mecánico y elimina controles manuales; los cuales son; cámara Kinect o cualquier dispositivo móvil.

El componente Comunicación: Se refiere a la interfaz que traduce los movimientos registrados en la cámara o en el sistema móvil para mover los servomotores del Brazo Mecánico. Esta actividad la realiza la tarjeta Arduino.

El componente Sistema Mecánico: Es el brazo robótico que actuara en vez del brazo humano.

Fig. 3. Modelo Propuesto.

Para evaluar el modelo propuesto se construyó un prototipo. Los detalles de su implementación de discuten en la siguiente sección.

IMPLEMENTACIÓN DEL PROTOTIPO

I.ANDROID

Android [15], es un sistema operativo basado en el *kernel* de Linux y es muy utilizado en dispositivos móviles Fig. 4.

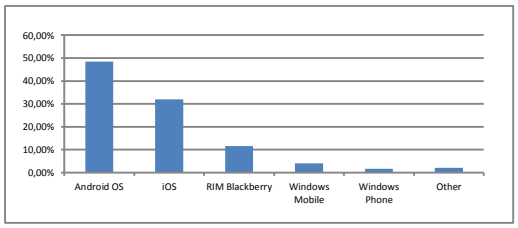

Fig. 4. Smartphone utilizados en el mundo. Fuente: The Nielsen Mobile Insights, National[23].

La forma de diseñar esta aplicación fue utilizando una herramienta que *Google* ha puesto de moda, llamado *App inventor* [16]. Su programación es muy sencilla, ya que maneja bloques (semejantes a la programación de *Matlab*), además de ser gratuito y tener mucha información, blogs, etc. El editor de bloques de la aplicación utiliza la librería *Open Blocks* de *Java* para crear un lenguaje visual a partir de bloques. Estas bibliotecas están distribuidas por *Massachusetts Institute of Technology* (MIT) bajo su licencia libre (*MIT License*).

La Fig. 5 muestra la programación a bloques para definir las acciones de los servomecanismos y la forma como se instala la aplicación (Apk) se muestra en la Fig. 6.

Fig.5. Programación de la interfaz

Se decidió utilizar el sistema *Android* [15], debido a que es el sistema más utilizado para el desarrollo de aplicaciones móviles, como puede observarse en la Fig. 4, además de su facilidad y tener una plataforma abierta para diseño.

A la fecha, se ha llegado ya al 1.000.000 de aplicaciones (de las cuales, dos tercios son gratuitas y en comparación con la *App Store* de *Apple* son más económicas) disponibles para la tienda de aplicaciones oficial de *Android*: otras tiendas no oficiales para *Android* como:

i) Tienda de aplicaciones *Samsung* Apps de Samsung

ii) *Slideme* de Java y *Amazon Appstore*.

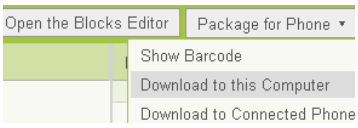

Fig. 6: Apk del instalador de la aplicación

La forma de control del sistema mecánico mediante la aplicación *App inventor*, es utilizando botones en la aplicación móvil, como se muestra en la Fig.7 . La desventaja es que no existe una retroalimentación en relación a visualizar el dispositivo que se desea controlar. Nos percatamos que *App Inventor* puede manejar video o fotografías pero no en tiempo real, es decir toma el video o foto y el usuario tiene que esperar a visualizarlo para definir qué acción tomar, esto resta tiempo y en procesos delicados o de emergencia no sería apto para utilizar.

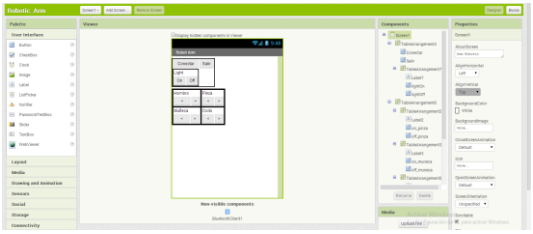

Fig. 7: Instrucciones de la aplicación mediante botones de acción.

La guarda estaría a distancia y podría manipular el sistema, pero no sería exacto porque tendría el operador que esperar a que se capture el video o la fotografía para visualizar que está pasando con el sistema y tendría latencia de respuesta, además si es una operación delicada y de observar con detalle para definir una acción esto no aplicaría.

Las instrucciones para programar dichas imágenes en la aplicación se pueden verificar en el siguiente algoritmo.

1.RecordVideo() y/o 2. TakePicture() 3. AfterPicture(robot1) 4.Define visible parametros de videoplayer 5. Stop() 6. Define escalas de presentación *ImagePath* para imágenes locales, una vez que ya se recivio la imagen.

II.- INTERFAZ (MICROCONTROLADOR)

El microcontrolador utilizado es *Atmel AVR* (*Arduino*). *Arduino* [9], es una plataforma de hardware libre, basada en una placa con un microcontrolador [8] y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares Fig. 8.

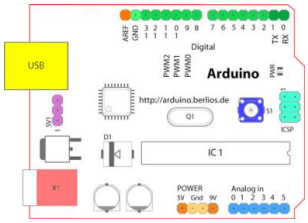

Fig. 8: Diagrama general de *Arduino*

El software puede ser descargado de forma gratuita. En la Fig. 10 se muestra la estructura general de comunicación del proyecto utilizando *Arduino*. Dicho algoritmo desarrollado Fig. 9, define una variable de tipo cadena para guardar los datos recibidos, se establece el número de los pines de salida (dos para cada motor para establecer, la opción de girar en ambos sentidos), se inicia comunicación serial mediante *Serial.begin (9600)*.

```
String dato;
int pinza0= 13;
int pinza1= 12;
int muneca0= 11;
int muneca1= 10;
int codo0= 7;
int codo1= 6;
int base0= 5;
int base1= 4;
void setup() {
Serial.begin(9600);
pinMode(pinza0, OUTPUT);
pinMode(pinza1, OUTPUT);
pinMode(muneca0, OUTPUT);
pinMode(muneca1, OUTPUT);
pinMode(codo0, OUTPUT);
pinMode(codo1, OUTPUT);
pinMode(base0, OUTPUT);
pinMode(base1, OUTPUT);
void loop() {
while (Serial.available()) {
delay(10);
if (Serial.available() > 0) {
char c = Serial.read();
dato += c;
Serial.println(c);
}
}
```

```
if(dato == "B") {
digitalWrite(muneca0, LOW);
digitalWrite(muneca1, HIGH);
}
if(dato == "3") {
digitalWrite(codo0, HIGH);
digitalWrite(codo1, LOW);
}
if(dato == "C") {
digitalWrite(codo0, LOW);
digitalWrite(codo1, HIGH);
}
if(dato == "4") {
digitalWrite(base0, HIGH);
digitalWrite(base1, LOW);
}
if(dato == "D") {
digitalWrite(base0, LOW);
digitalWrite(base1, HIGH);
}
if(dato == "5")
digitalWrite(foco, HIGH);
if(dato == "E")
digitalWrite(foco, LOW);
delay(500);
dato="";
}
}
```

Fig. 9: Algoritmo desarrollado

Una vez establecido lo anterior se definen las salidas digitales y los pines establecidos como salidas *pinMode (pinza0, OUTPUT)*. Cuando se recibe un dato, se guarda como carácter y se comprueba que la variable "dato" tenga al menos un carácter es decir *if (dato == "1")*, además de comparar la cadena de caracteres con cada uno de los casos. Finalmente se declara la variable "vacía" para volver a guardar de nuevo caracteres y compararlos con cada uno de los casos.

Fig. 10: Diagrama de comunicación

La estructura de la interfaz, está controlado mediante *Arduino*, de tal manera que tanto la comunicación vía *Kinect* o *Android* llega al serial de *Arduino*; quien es el que traduce dichas instrucciones para el movimiento de los servomotores.

Forma de conexión :

Pin 1 : Tierra

Pin2: Pinza2

Pin3: Pinza3

…

Se puede agregar *n* números de movimientos, de hecho; se pueden hacer combinaciones pero de forma manual ya que la limitante es tocar un solo botón en el celular (acción) pero en cámara *Kinect* se puede hacer simultáneo. El único problema es el voltaje ya que al hacer dos movimientos al mismo tiempo reduce la potencia en los motores. Como trabajo futuro se arreglara agregando una etapa de amplificación (*Opams 741*) [17].

III.- KINECT

La videoconsola *Xbox 360* [10], permite a los usuarios interactuar con la consola sin necesidad de tener contacto físico con un controlador, mediante una interfaz [5] que reconoce gestos, e imágenes. Las instrucciones de manipulación del control utilizado fue; *SDK* de *Microsoft* el cual tiene acceso a la cámara *RGB* y a los datos de profundidad que indican la distancia de un objeto al sensor de *Kinect* [10] por medio de la cámara de profundidad XYZ. El procesamiento de audio para el arreglo de 4 micrófonos con cancelación de ruido acústico y de eco, un "*beam formation*" para identificar la fuente del sonido y la integración con la API de *Windows speech recognition*. El sensor de profundidad es un proyector de infrarrojos combinado con un sensor monocromo que permite a *Kinect* ver en 3D. Una vez que *Kinect* detecte la presencia del usuario se le dibujara un "Stickman" Fig. 11 con los vértices, para dibujar ese "Stickman" debemos llamar al objeto de la librería SDK y definir de que punto a qué punto estarán unidos. Esto se puede verificar en el siguiente pseudocódigo.

```
int userID=userList.get (0);
 if(kinect.isTrackingSkeleton(userID))  {  kinect.drawLimb(userID, SDK.SKEL_HEAD);…….
```

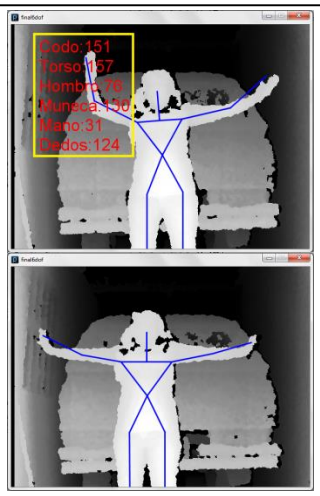

Fig. 11: Stickman generado en cámara

Este código dibuja la parte de la cabeza y el cuello los cual unirá esos dos puntos al momento de dibujar el "*stickman*", es decir el vector. Este mismo algoritmo se repite para cualquier vector que se quiera dibujar, cambiando los "tracking" a mostrar, es decir de qué punto a punto queremos visualizar en el espacio. Para activar la cámara con *VideoStream* es necesario enviar parámetros como el tipo de flujo que se quiere obtener, la resolución y el tipo de imagen que se va a capturar.

Esto es porque al definir los vectores muchas veces solo da un punto (destino final del vector), como lo muestra el siguiente pseudocódigo.

```
knect.VideoStream.Open(Mi-
crosoft.Research.Kinect
.Nui.ImageStreamType.Video, Mi-
crosoft.Research.Kinect.Nui.Im-
ageResolution.Resolu-
tion640x480,Microsoft.Re-
search.Kinect.Nui.Imag
```

Este punto fue muy crítico, ya que al no cargar estas instrucciones *Kinect* dará una señal como resultado, pero esta no podrá ser procesada por *Arduino* . Ahora que se tiene el "Stickman" [10] y se configuró la forma de reproducir más eventos (imágenes) del sensor, se puede definir la pantalla a visualizar mediante un *bitmapsource* que enlace la imagen y los puntos definidos en el espacio; como a continuación se demuestra:

```
kinect.VideoFrameReady += new EventHandler<Im-
ageFrameReadyEventArgs>(kinect_VideoFr
ameReady);
void kinect_VideoFrameReady(object sender,
ImageFrameReadyEventArgs e)
PlanarImage Image = e.ImageFrame.Image; image1.Source
= BitmapSource.Create(Image.Width, Image.Height,

96, 96, PixelFormats.Bgr32, null, Image.Bits, Image.Width
* Image.BytesPerPixel);

}
```

RESULTADOS

El reto más grande en este proyecto, no fue la manipulación de los servomecanismos, de hecho fue muy sencillo utilizar mediante el celular la aplicación *Android* enviarlo al microcontrolador y este al brazo robótico. Lo complicado fue la cámara *Kinect* [5], debido a que una vez que creamos el vector entre cada punto por medio y con respaldo de otro vector superpuesto encontramos el ángulo entre los dos vectores gracias a la ecuación:

$$\cos\alpha \frac{u_1 v_1 u_2 v_2}{\sqrt{u_1^2 + u_2^2} * \sqrt{v_1^2 + v_2^2}}$$

Las bibliotecas *math* nos ayudaron mucho a aplicar la formula, ya que el ángulo inverso que se modificó a el ángulo resultante y se invirtió por el efecto "espejo" que tiene *Kinect*.
Sin embargo los resultados obtenidos en la comunicación, para que el robot ejecutara una serie de "instrucciones" a realizar fue el siguiente ver Fig. 12.

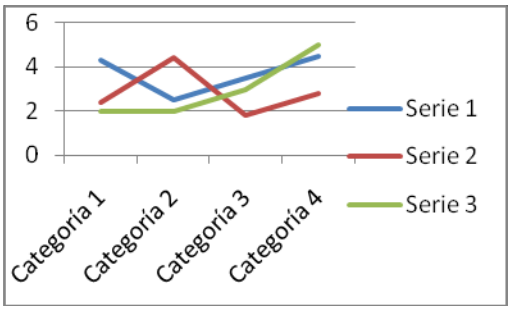

Fig. 12: Resultados por serie de procesos

Al principio las primeras instrucciones no se detectaban correctamente (serie 1,2) se iniciaban en la instrucción "mover pinza1" (serie 1) y "mover codo" (serie 2) del brazo robótico, sin embargo al correr la serie de instrucciones 3 nos percatamos que si se encuentra en modo "0" es decir, un estado de inicialización el robot ejecutaba todas las instrucciones.

No debería existir este modo de inicialización ya que en la vida real debe ser capaz de interrumpir una acción y realizar la nueva instrucción que se le envié, vía celular o kinect; ya que se pierde tiempo en inicializar todo el sistema, además que se pierden instrucciones , es decir; si tiene una serie de instrucciones a realizar (procesos) y se envía una instrucción remota este debería parar el siguiente proceso, realizar la o las ejecuciones remotas (dispositivo móvil, *Kinect*) y al terminar recordar cuales de las instrucciones (antes de ser interrumpido) realizo, y cuales seguirán.

Este resultado lo analizaremos en trabajos futuros, lo interesante es que el medio de comunicación vía remota se ejecuta correctamente con celular y *Kinect*, como lo muestra la Fig. 13.

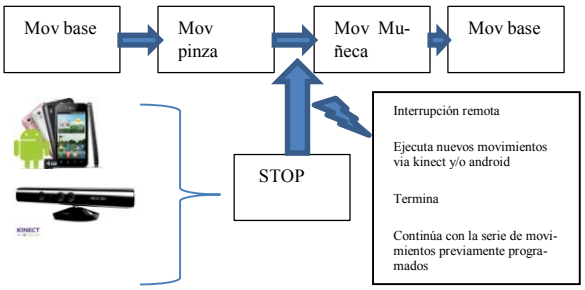

Fig. 13: Diagrama a bloques de la función de inicialización (stop).

TRABAJO FUTURO

Como trabajo futuro se intentará resolver el problema de "interrumpir un proceso" ya establecido (serie de instrucciones programadas) en el brazo robótico. Ya que es crítico si el sistema es utilizado para realizar tareas específicas y repetitivas.

El prototipo para resolver este problema, es definir las instrucciones predefinidas en un programa, estas pasarlas a una base de datos a un servidor *Apache*; y *Arduino* sea quien se comunique con el servidor para que al momento de ser interrumpido por una instrucción maestra (dispositivo móvil, *Kinect*) este pueda enviar al servidor en que registro del proceso fue interrumpido para que el robot (esclavo) ejecute las nuevas instrucciones remotas; y al terminar el servidor continúe con las instrucciones anteriores.

Es decir este algoritmo funcionaria como un modelo maestro-esclavo (cliente-servidor), don- de tendríamos un administrador de procesos quien es independiente de las instrucciones dadas vía remota. Es en esta etapa donde se utilizaría *Raspberry Pi* [21], para que haga las funciones de servidor servidor y sea quien se comunique con *Arduino*.

CONCLUSIONES

El desarrollar sistemas controlados por medios inalámbricos está dando pasos agigantados en la actualidad. En esta investigación el uso de la cámara *Kinect* nos facilitó el control de la guarda de

manera sencilla, gracias a que el desarrollador Microsoft dio libre acceso a las bibliotecas y un entono para su manipulación; aun se continua el desarrollo del mismo, y cuenta con una gran gamma de bibliotecas de acceso libre, su entorno es muy amigable y fácil de usar , tiene varias limitaciones y no permite el máximo uso de las funciones del *Kinect*. Investigando aún más profundo encontramos que para controlar la cámara, se liberó otra plataforma de control llamada *Processing* [18] el cual maneja un lenguaje de programación Java [22] y de código abierto, desarrollada por *Massachusetts Institute of Technology*.

Este proyecto tenía como objetivo la guarda de un control para sistemas mecánicos, con el fin de prevenir accidentes. Las pruebas se hicieron bajo un brazo robótico, simulando el brazo humano para desarrollar actividades (movimientos) que pudieran poner en riesgo a una persona; bajo cualquier manejo de maquinaria mecánica, químicos, etc. Una gran cantidad de desafíos nos enfrentamos al desarrollar este proyecto debido a que se mezclan diferentes disciplinas como lo son las áreas de sistemas, electrónica, mecánica, mecatrónica, eléctrica, cómputo móvil, sistemas en tiempo real etc.

En el caso de *Android*, fue Google quien diseño el software para que cualquier persona pudiera hacer sus propias aplicaciones a celular, de no existir este software se hubiera utilizado *Eclipse* [19], para programar dicha aplicación.

Con el diseño de este proyecto se abrieron nuevas problemáticas, las cuales son áreas de oportunidad para diseñar nuevas formas de administrar procesos y relacionar diferentes disciplinas. La meta de crear la guarda de control se cumplió, ya que el operador no se encuentra directamente expuesto con alguna herramienta o maquinaria; gracias a la cámara *Kinect* o la aplicación *App inventor*. Sin embargo nos pudimos percatar de que este sistema aplicaría para manejo de cualquier sistema mecánico siempre y cuando no sea con una actividad de visión fija, ya que sería necesario tener cámaras alrededor del proceso. Este sistema funciona correctamente para procesos donde el operador puede a la distancia visualizar que es lo que necesita pintar, cortar, prensar etc.

REFERENCIAS

(1) K. Ogata, Modern Control Engineering. Jakarta, Erlangga, 1997.
(2) Human-machine interfaces based on EMG and Kinect applied to teleoperation of a mobile humanoid robot, Baocheng Wang ; Chenguang Yang ; Qing Xie : Intelligent Control and Automation (WCICA), 2012 10th World Congress 2012 , Page 3903 – 3908
(3) C. John, Introduction to Robotics Mechanics & Control, Stanford
University, 1986.
(4) Teleoperation Humanoid Robot Control System Based on Kinect Sensor, Weibo Song ; Xianjiu Guo ; Fengjiao Jiang ; Song Yang ; Guoxing Jiang ; Yunfeng Shi ;Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference , Page(s): 264 – 267
(5) Kinect Open Source Programming Secrets: Hacking the Kinect with OpenNI, NITE, and Java, by Andrew Davison McGraw-Hill/TAB Electronics, ISBN-10: 0071783172
(6) Arduino and Kinect Projects: Design, Build, Blow Their Minds (Technology in Action),Enrique Ramos Melgar , Ciriaco Castro Diez ISBN-10: 1430241675
(7) Teleoperation of Differential Drive Mobile Robots. Oscar Martinez Palafox 2011, ISBN-10: 1243594128
(8) Programming Arduino Getting Started with Sketches, Simon Monk,2011: ISBN-13: 978-0071784221
(9) Home Automation with Arduino: Automate your Home using Open-Source Hardware ,Marco Schwartz: 2013, ISBN-13: 978-1491016824
(10) Kinect based gesture controlled Robotic arm: A research work at HuT Labs Megalingam, R.K. ; Saboo, N. ; Ajithkumar, N. ; Unny, S. ; Menon, D. Innovation and Technology in Education (MITE), 2013 IEEE International Conference in MOOC , 2013 , Page(s): 294 - 299
(11) Telemanipulation of an industrial robotic arm using gesture recognition with Kinect, Shirwalkar, S. ; Singh, A. ; Sharma, K. ; Singh, N. Control, Automation, Robotics and Embedded Systems (CARE), 2013 International Conference IEEE
(12) Dynamic analysis and control of robotic manipulator for chemically aggressive environments, Martin Goubej,Martin Švejda2013 IEEE International Conference IEEE

(13) Seguridad en maquinas, Diego Gonzalez M. ISBN-10: 8496743772

(14) Robotica Inustrial Fundamentos y aplicacaciones, Arantxa Renteria; ISBN-10: 8448128192

(15) Introduction to Android Application Development: Android Essentials (4th Edition) ; Joseph Annuzzi Jr. ,Lauren Darcey, Shane Conder , ISBN-10: 0321940261

(16) App Inventor: Create Your Own Android Apps, David Wolber, Hal Abelson , Ellen Spertus , Liz Looney ISBN-10: 1449397484

(17) Prof. Greg Kovacs, Department of Electrical Engineering Stanford University, http://web.stanford.edu/class/ee122/Handouts/2-Op-Amp_Concepts.pdf

(18)Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction, Daniel Shiffman; ISBN-10: 0123736021

(19)15 Minutes To Easy Eclipse Java Programming, DJ Slavens; 1 edition (2013)

(20)http://www.stps.gob.mx/bp/secciones/dgsst/estadisticas.htm

(21) Getting started with Raspberry Pi, Matt Richardson, ISBN-10: 1449344216.

(22)Java The complete reference, Herbert Schildt, 2014 ISBN-10: 0071808558

(23)http://www.nielseninsights.it/mobile/

