Comunicación entre RobotStudio–Leapmotion para el control de robot industrial

Pedro Andrey Cañón [1], Pedro Cárdenas [2], Oscar Rodríguez [3]

(1) pacanonj@unal.edu.co
Ingeniería Mecatrónica Universidad Nacional de Colombia
Bogotá, Colombia

(2) pfcardenash@unal.edu.co
Profesor Asociado, Universidad Nacional de Colombia
Bogotá, Colombia

(3) oscar.rodriguez@uptc.edu.co
Profesor, Universidad Pedagógica y Tecnológica de Colombia

Comunicación entre RobotStudio–Leapmotion para el control de robot industrial

Palabras clave:
Telerobótica; interacción, manipulador; LeapMotion; SDK ABB; Pythonfor Net

RESUMEN

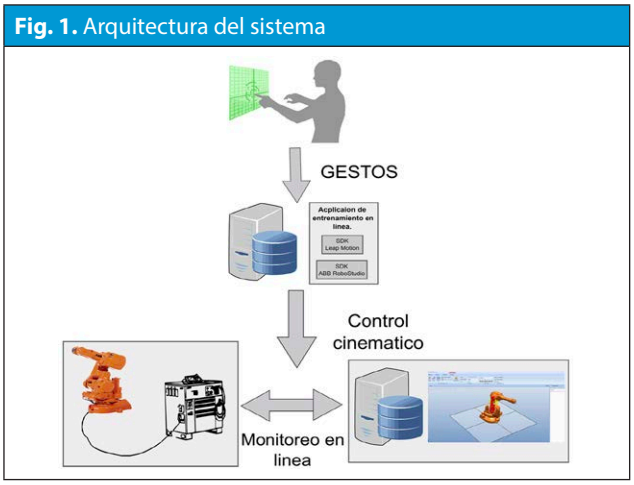
En este artículo se muestra el trabajo desarrollado para lograr la comunicación entre una aplicación desarrollada en Windows, el programa RobotStudio y el controlador IRC5 de la marca ABB. La aplicación desarrollada implementa dos kits de desarrollo de software (SDK); uno de estos permite obtener datos del dispositivo Leapmotion el cual detecta la posición de la mano y retorna una coordenada en el espacio, el otro kit de desarrollo permite la manipulación de datos RAPID (Lenguaje de programación implementado por ABB) y el envío de comandos de ejecución desde la aplicación al programa RobotStudio y al controlador IRC5.

I. INTRODUCCIÓN

En el ámbito de la robótica industrial está presente la necesidad de lograr procesos de manera sencilla e intuitiva; dado que estos actualmente requieren ya sea personal capacitado o el tiempo para capacitar a los mismos. El desarrollo de aplicaciones que permitan una interacción de manera eficiente entre el usuario y la máquina, sin llegar a necesitar conocimientos específicos, lograra que la implementación de los procesos sea mucho más sencilla en cuanto a requerimientos de programación disminuyendo así tiempos y costos de proceso.

Los métodos de programación intuitivos para sistemas robotizados nace del análisis de sistemas industriales, donde incluso observando que los sistemas robotizados en la industria se enfocan en tareas repetitivas ofreciendo mayor velocidad, precisión y perceptibilidad es evidente que sistema de producción robusto requiere de la capacidad de adaptar la configuración de una celda de trabajo a nuevas tareas. Debido a esto, los robots entes dinámicos capaces de adoptar nuevos roles dentro de las cadenas de producción optimizando los procesos [6].

Fig. 1. Arquitectura del sistema



El proceso convencional de programación de robot industriales contempla la planeamiento de trayectorias, análisis de configuración y solución numérica del problema cinemático inverso que requiere la dedicación de personal especializado en la configuración de los equipos [2]. En el diseño de un plan de trabajo robusto donde el robot sea capaz de adaptarse a nuevas tareas y configuraciones de su entorno no es una tarea trivial y requiere de especial atención por parte de los ingenieros de programación [3].

Se planteó la pregunta de cómo lograr las aplicaciones deseadas y que tan flexibles podrían llegar a ser estas y para esto se buscó desarrollar una aplicación que permitiera controlar el desplazamiento de un robot industrial de modo que el usuario solo necesitara mover la mano en un espacio determinado para lograr recorridos por el robot idénticos a los realizados. Finalmente como se presentan diversos tipos de procesos se plantean las bases necesarias para que cualquier persona pueda crear una aplicación correspondiente a su necesidad.

Con el objetivo de desarrollar nuevos métodos de programación de robots industriales que disminuyan el tiempo y la complejidad asociada al proceso de ingeniería se han estudiado recientemente nuevas tecnologías de interacción humano-máquina para servir como un método de programación mas natural [4] [5]. Los métodos naturales de comunicación como voz y gestos son de especial interés en esta área de investigación por el potencial que representa que una persona con poco entrenamiento pueda ser capaz de programar un sistema robotizado [2] [8].

Los gestos son expresiones visuales naturales de una persona que desea transmitir información al robot para que este realice un trajo especifico [7]; estos gestos pueden ser identificados utilizando elementos de captura de imágenes, analizando movimientos y gestos básicos que son traducidos al lenguaje de máquina y alimentados en tiempo real a un controlador robótico [3] .

Este trabajo se enfoca en la detección de la interacción humana en el espacio para controlar la cinemática de un brazo robotico industrial dentro del entorno de programacion offline de robots industriales. El método propuesto permite al usuario operar el robot controlando la trayectoria del efector final mediante gestos manuales haciendo uso de una aplicación desarrollada en torno a la adquisición de datos de un sensor de movimiento y su enlace al software de programación de robot industriales.

El artículo esta organizado como sigue, en las Sección II se describe la configuración del sistema, en la sección III se muestra la integración de los diferentes componentes como una sola aplicación. Finalmente en la sección V se presentan los resultados y conclusiones de la aplicación desarrolla.

II. ARQUITECTURA DEL SISTEMA

En la figura 1 se presenta el concepto total de la aplicación desarrollada. Se utiliza un sensor de movimiento Leap Motion para capturar ciertos movimientos que realiza la mano. Para interactuar con el sensor Leap Motion es necesario disponer del conjunto de librerías SDKLeapmotion. El sensor se conecta a un PC a través de el puerto UBS3. Adicional en el PC debe estar instalado el software de programación offline de robots ABB, denominado RobotStudio, al igual que el conjunto de librerías PC-SDK de ABB.

A. Sensor leap Motion

El software de Leap Motion corre como un servicio en Windows o como daemon en Mac o Linux este se encarga de conectar el controlador del dispositivo a través de puerto USB. Las aplicaciones habilitadas acceden al servicio de Leap Motion para recibir los datos de seguimiento pertenecientes al movimiento.

El SDK de Leap Motion provee dos variedades de API (Application Programming Interface) para obtener los datos de seguimiento desde el servicio Leap Motion: una interfaz propia y una interfaz WebSocket. La interfaz propia es una librería dinámica que uno puede usar para crear nuevas aplicaciones. La WebSocket interfaz permite la creación de aplicaciones web.

1) *Interfaz de Aplicaciones Propias*: Las aplicaciones de interfaz propias son provistas a través de una librería dinámica. Esta librería se conecta al servicio de Leap Motion y provee el seguimiento de datos a la aplicación desarrollada. Figura 2.

2) *Interfaz WebSocket*: El servicio de Leap Motion corre como un servicio WebSocket sobre el dominio de localhost en el puerto 6437. La interfaz WebSocket provee los datos de seguimiento en forma de mensajes JSON(JavaScript Object Notation). Un cliente JavaScript es habilitado para el uso de mensajes JSON y destina el seguimiento de datos como un objeto JavaScript. Figura 3.

Fig. 2: Arquitectura Interfaz Propia LeapMotion

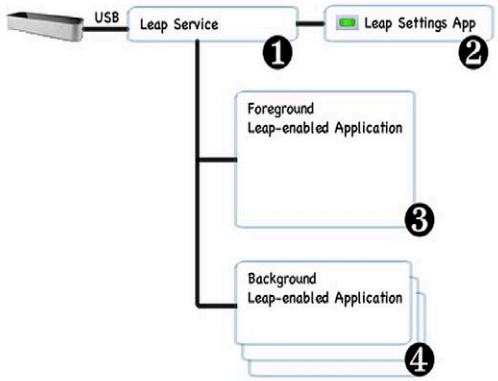
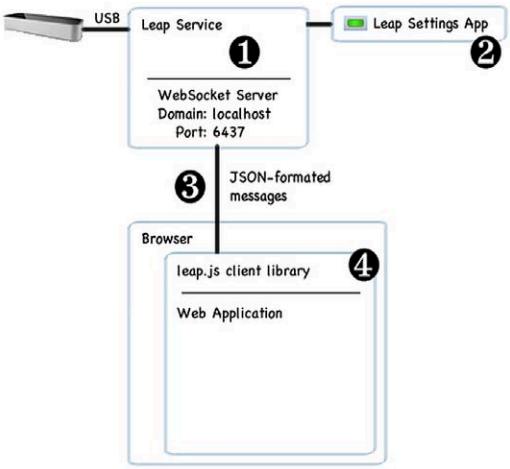


Fig. 3. Arquitectura Interfaz WebSocket



B. Software RobotStudio

RobotStudio es una aplicación de PC destinada al modelado, la programación fuera de línea y la simulación de células de robot.

RobotStudio le permite trabajar con un controlador fuera de línea, que constituye un controlador IRC5 virtual que se ejecuta localmente en su PC. Este controlador fuera de línea también se conoce como el controlador virtual (VC). RobotStudio también le permite trabajar con un controlador IRC5 físico real, que simplemente se conoce como el controlador real.

Cuando RobotStudio se utiliza con controladores reales, se conoce como el modo online. Al trabajar sin conexión a un controlador real o mientras está conectado a un controlador virtual, se dice que RobotStudio se encuentra en el modo fuera de línea.[9]

PC SDK permite a los integradores de sistemas, terceros o usuarios finales añadir sus propias interfaces de operador personalizadas para el controlador IRC5. Tales aplicaciones personalizadas se pueden realizar como aplicaciones de PC independientes, que se comunican con el controlador de robot sobre una red.

Se debe esperar un tiempo de respuesta mínimo para un controlador real en el orden de 10-100 milisegundos, lo que significa que duras exigencias de tiempo real no pueden ser satisfechas en cualquier plataforma. [10]

III. DESCRIPCIÓN DE SOFTWARE E IMPLEMENTACIÓN

A. Configuración

No fue necesaria ninguna configuración especifica para realizar las aplicaciones en C# ya que VisualStudio provee las herramientas necesarias para el desarrollo de las mismas. En el caso de las aplicaciones desarrolladas en Python posteriormente a la instalación se necesito descargar varios complementos que se listan a continuación:

• PyQt4 Permite el uso de interfaces desarrolladas en QtCreator

• Pythonfor.NET Permite el uso de librerías de C # en Python

B. Programas desarrollados

La metodología implementada para el desarrollo de la aplicación se realizó en dos lenguajes de programación distintos, C# y Python, en ambos lenguajes se creó una aplicación de consola y una aplicación de interfaz generada por VisualStudio para C#y QtCreator para Python.

El objetivo de la aplicación con interfaz es listar todos los controladores tanto virtuales como reales que este en línea con el host y mostrar las características de cada uno, para luego seleccionar el controlador deseado y poder asignar una posición (*x, y, z*) de forma manual, que corresponderá a la posición del efector final de robot en línea con el controlador.

El objetivo de la aplicación de consola es seleccionar de manera automática un controlador en linea y detectar la posición de la mano mediante el sensor Leap Motion. Las coordenadas obtenidas seran enviadas al controlador del robot de modo que este replique el desplazamiento de la mano.

Para ambas aplicaciones se sigue el diagrama de flujo mostrado en la figura 4. En la aplicación de interfaz la lectura de la coordenada se realiza de un dato ingresado por el usuario final y en la aplicación de consola este dato es provisto por el sensor Leap Motion de manera automática.

Cada aplicación se ejecuta en paralelo con un programa en lenguaje RAPID cargado previamente en el controlador, ya sea virtual o real, este programa debe tener definidas la variables de manera global que van estar sujetas a modificación posterior por parte de la aplicación. El programa RAPID se mantendrá en ejecución hasta que la aplicación le envíe una señal de stop. El diagrama correspondiente a este programa se puede observar en la figura 5.

C. Operación

El programa del Leap Motion se configura tipo interrupción de manera que cada vez que el usuario ingrese la mano en el campo de lectura del sensor se ejecute una rutina la cual es la encarga de leer el dato de la posición de la palma de la mano, transformar las coordenadas obtenidas al espacio de trabajo del robot y finalmente enviar estos valores al código RAPID.

El movimiento del robot se programa mediante la instrucción MoveL que desplaza el TCP con una orientación constante de manera lineal. Los parámetros necesarios para ejecutar la instrucción son un dato robtarget, que contiene la información del punto al que se desea ejecutar el desplazamiento con una orientación definida, un parámetro de velocidad y un dato que contiene la información de herramienta usada. Uno de los parámetros del dato robtarget es una variable de tipo pos; esta variable contiene

las tres coordenada del punto al cual se quiere desplazar. Desde la aplicación se accede vía Ethernet al controlador de robot y es cambiado el valor de la variable tipo pos, de modo que cada vez que se ingrese un nuevo valor el robot se desplace a este.

Fig. 4. Diagrama de flujo de la aplicación

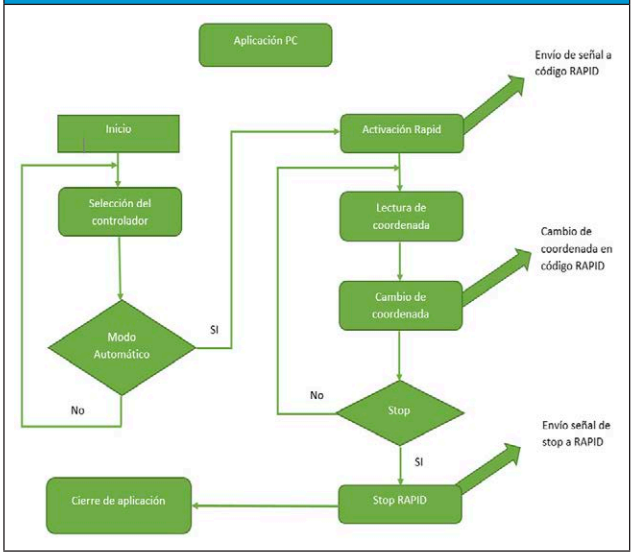
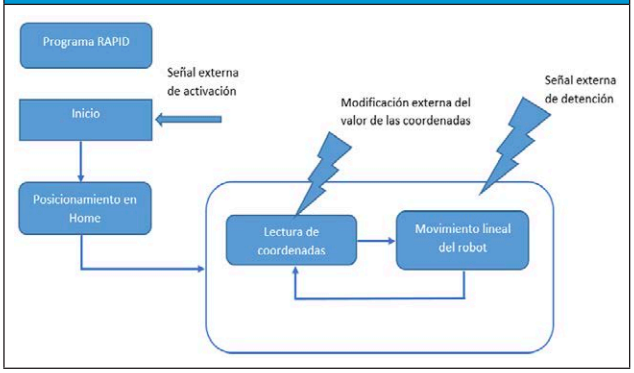


Fig. 5. Diagrama de flujo programa RAPID



IV. RESULTADOS

En el presente trabajo se describe la metodología para integrar el sensor de movimiento leap motion y el entorno de programación y simulación de robots industriales robot studio. Los resultados obtenidos de pruebas a partir de esta metodología permitieron desarrollar tiempos mínimos de ejecución del movimiento del robot. La prueba realizada se baso en enviar una coordenada desde la aplicación al código RAPID y medir el tiempo requerido para el desplazamiento del punto inicial a la coordenada enviada en el espacio operacional del robot. El tiempo mínimo se obtuvo enviando una coordenada muy cercana a la posición inicial, para esta prueba se aumento solo la coordenada *x* en 0.1 mm y se obtuvo un tiempo aproximado de 60 ms. Este tiempo hace referencia al tiempo de comunicación y al tiempo que toma ejecutar la instrucción de movimiento; dichas pruebas se realizaron variando el parámetro de velocidad del robot y se obtuvieron valores similares. El código implementado en RAPID se puede observar a continuación y la interfaz en la figura 6. Respecto a la relación entre el movimiento de la mano dentro del espacio de captura del sensor y el movimiento en el espacio operacional del robot es ajustada a través de escalización, sin embargo para determinar la precisión en los movimientos no se contó con elementos externos de medición por tanto hacer una relación entre el movimiento de la mano el extremo del efetor no tiene lugar.

```
ClkStart myclock;
// position variable tipo robottarget
modificada desde la aplicación.
MoveL position, vmax,fine, tool0;
ClkStop myclock; reg1:=ClkRead(myclock \HighRes);
```

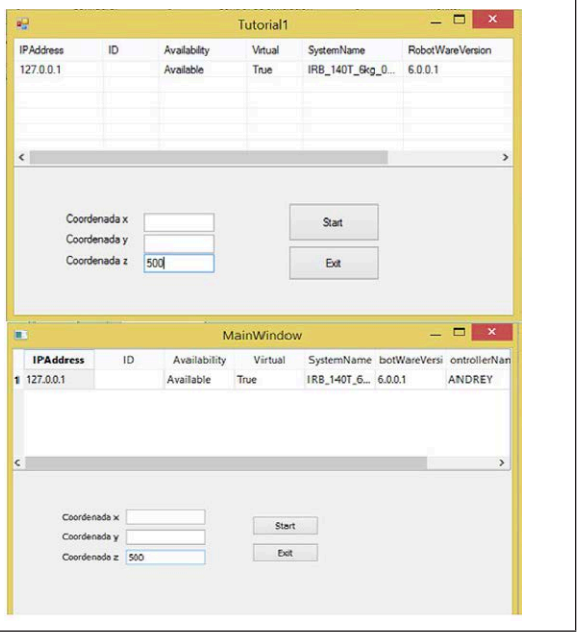
Las pruebas realizadas implementando el sensor Leap Motion estuvieron limitadas a diversos factores tanto físicos como de software. Físicamente se tuvo que limitar la velocidad del robot debido a que, a velocidades altas y por la cantidad de múltiples coordenadas provistas por el sensor, el robot presentaba vibraciones y sobreactuación en posiciones fijas; esto se soluciono limitando la velocidad del TCP del robot en un rango de 100mm/s a 200mm/s. Debido a que el sensor retorna una coordenada en un tiempo de 5ms y la ejecución del movimiento toma un tiempo mínimo de 60ms se perdió fidelidad en el seguimiento de la trayectoria para solucionar esto se planteó el uso de una pila de estructura fifo, que almacene las coordenadas provistas por el sensor Leap Motion, con un vaciado total de la pila cada cierto intervalo de tiempo; en este caso se mejora el seguimiento de la trayectoria pero se pierde la sensación de movimiento en "tiempo real" de la aplicación.

V. CONCLUSIONES

Es difícil realizar un seguimiento idéntico de la trayectoria de la mano por parte del robot debido a que los tiempos de ejecución del movimiento exceden los tiempos de detección por parte del sensor; lo que impide que se recorran todos los puntos detectados, para solucionar este inconveniente se sugiere que el seguimiento no se realice en tiempo real.

La realización de aplicaciones intuitivas se puede lograr limitando la funcionalidad de las mismas y considerando los posibles errores que se puedan presentar durante la ejecución, para esto el programador debe tener claro el proceso a realizar y las restricciones necesarias para evitar un caso de falla esto genera poca flexibilidad en las aplicaciones y hace necesario una aplicación específica para cada proceso.

Fig. 6. Interfaz realizada en VisualStudio y en QtCreator



AGRADECIMIENTOS

A la Universidad Nacional de Colombia a través de las DIB por el programa de Semilleros de investigación. A Colciencias por la beca de doctorado de Pedro Fabián Cárdenas.

REFERENCIAS

[1] Wen, R., Tay, W. L., Nguyen, B. P., Chng, C. B., & Chui, C. K. (2014). Hand gesture guided robot assisted surgery based on a direct augmented reality interface. Computer Methods and Programs in Biomedicine, 116(2), 68–80. http://doi.org/10.1016/j.cmpb.2013.12.018

[2] Ganapathyraju, S. (2013). Hand gesture recognition using convexity hull defects to control an industrial robot. 2013 3rd International Conference on Instrumentation Control and Automation (ICA), 63–67. http://doi.org/10.1109/ICA.2013.6734047

[3] Lambrecht, J., Kleinsorge, M., & Krüger, J. (2011). Markerless gesture based motion control and programming of industrial robots. IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, 1–4. http://doi.org/10.1109/ETFA.2011.6059226

[4] Kruse, D., Radke, R. J., & Wen, J. T. (2013). A Sensor based Dual Arm Tele Robotic Manipulation Platform. 2013 IEEE International Conference on Automation Science and Engineering (CASE), 350–355.

[5] Neto, P., Pereira, D., Pires, J. N., & Moreira, a P. (2013). Real Time and Continuous Hand Gesture Spotting : an Approach Based on Artificial Neural Networks.

[6] Cerlinca, T., & Vlad, V. (2012). A Human Robot Interaction based approach to automation in industrial manufacturing. EPE 2012 - Proceedings of the 2012 International Conference and Exposition on Electrical and Power Engineering, (Epe), 29–34. http://doi.org/10.1109/ICEPE.2012.6463602

[7] Lin, H., Liu, Y., & Lin, Y. (2013). Intuitive Kinematic Control of a Robot Arm via Human Motion. Procedia Engineering, 00(1st ICM), 1–6. http://doi.org/10.1016/j.proeng.2014.06.362

[8] Mohammad, Y., & Nishida, T. (2014). Learning interaction protocols by mimicking understanding and reproducing human interactive behavior. Pattern Recognition Letters, 000, 1–9. http://doi.org/10.1016/j.patrec.2014.11.010

[9] Manual del operador RobotStudio 6.01

[10] Application manual PC SDK RobotWare 5.14