# Control gestual de robot de 4 GDL con sensor Leap Motion

Oscar Justinico[1], Pedro F. Cárdenas[2], Jhoan Sebastián Rodríguez[3]

(1) oijustinicocj@unal.edu.co
Ingeniero Mecatrónico, Universidad Nacional de Colombia
Bogotá, Colombia

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

(3) jhsrodriguezro@unal.edu.co
Ingeniero Mecánico, Universidad Nacional de Colombia
Bogotá, Colombia

# Control gestual de robot de 4 GDL con sensor Leap Motion

*Palabras clave:*
***Robótica; interación; Leap Motion; control gestual; brazo robótico***

**RESUMEN**

Existe una fuerte tendencia de investigación el reconocimiento de gestos en la interacción hombre-computador. En robótica el reconocimiento de gestos es usado para controlar robots de manera simple con señales de mano. Básicamente un dispositivo con un sistema de visión de máquina captura las posiciones de la mano y asigna una acción a aquellos movimientos que presentan un patrón determinado. El control del robot de 4 grados de libertad (GDL) presentado en este artículo se basa en técnicas de reconocimiento de gestos simples y usa solo movimientos predeterminados de la mano derecha. Se diseña una serie movimientos gestuales para ejecutar funciones que de otra forma requerirían de dispositivos externos para operar sobre el robot. Se implementa el reconocimiento de patrones gestuales que permiten operaciones tales como encendido-apagado del robot, y manejo de la pinza.

## I. INTRODUCCIÓN

En los próximos años, se espera que el numero de personas en contacto con robots en la industria aumente convirtiéndose en la tecnología con mayor importancia en el desarrollo tecnológico; el estudio de la interacción de robots con humanos ha permitido establecer las condiciones ideales en las que un sistema robotizado puede formar un sistema eficiente en contacto con las personas, estos robots necesitarán ser capaces de adquirir suficiente información de su entorno, detectar personas y establecer un enlace de comunicación con el personal involucrado en el proceso para poder ser un elemento efectivo en sistema.

En la actualidad la capacidad de un sistema robotizado de adaptarse a una tarea o a un entorno depende en gran parte de la capacidad del operador quien programa el robot, limitando esta tarea a las personas que tengan la suficiente capacidad técnica y adecuado entrenamiento. Es evidente que si se desea avanzar en la implementación de estos sistemas en ambientes cotidianos se debe establecer un método de interacción más versátil que permita el control y programación de estos dispositivos en cualquier entorno. [4]

En recientes investigaciones se ha prestado especial atención a la interacción humano-máquina mediante la interpretación de gestos [5], creando un lenguaje de comunicación natural entre un operador o instructor y el robot que recibe e interpreta los mensajes [?]. Consecuentemente, el robot podría ser operado y a la vez aprender nuevas rutinas mediante el seguimiento de comandos de un instructor que gracias al sistema de gesto puede ser cualquier persona en contacto con el robot.

La comunicación humano-máquina mediante gestos es constantemente estudiada en la implementación de nuevas tecnologías de comunicación instantánea y entretenimiento [5], [6], en estos estudios se han establecido los procedimientos por los cuales se establece los gestos adecuados que hacen parte de un lenguaje específico [7]; con base en esto se puede relacionar las características mínimas de comandos que un manipulador robótico necesita obtener del guía con los gestos apropiados para su detección e interpretación por el sistema.

En el presente trabajo se hace un descripción del proceso de implementación de un sistema de reconocimiento gestual en el control de un robot serial de 4 grados de libertad (GDL) y el desarrollo de un sistema de interpretación de gestos manuales para el control en linea del dispositivo en la ejecución de tareas comunes, ver figura 1.

**Fig. 1:** Esquema General de la aplicación


El artículo está organizado como sigue, en la sección II se describen los dispositivos que componen la arquitectura fisica del sistema, en la sección III se desarrollan los modelos geométricos para calcular la cinemática directa e inversa, en la sección IV se describe la estructura de comunicación y la lógica del sistema. En la sección V se presenta la interfaz gráfica, en la sección VI se describen los resultados obtenidos. El artículo finaliza con las conclusiones de la aplicación desarrollada, los agradecimientos y las referencias consultadas.

## II. COMPONENTES DEL SISTEMA

En la figura 1 se observa el esquema general de la aplicación. La integración del sensores de movimiento como el robot se hace bajo la plataforma de windows y Matlab. Se utilizan cada una de las herramientas de software proveídas por los fabricantes.

### *A. Sensor de movimiento Leap Motion*

El sensor Leap Motion es un dispositivo que rastrea puntos de las manos usando cámaras infrarrojas en un campo de visión de 150°, (fig. 2). Este dispositivo obtiene la posición, orientación y velocidad del objeto rastreado a una velocidad de hasta 200 cuadros por segundo.

**Fig. 2:** Dispositivo Leap Motion

El SDK del Leap Motion permite acceder a las diferentes funciones que retornan los datos obtenidos por el sensor. El SDK contiene librerías para varias plataformas tales como Python, C++, y *C#* en diferentes versiones de .Net Framework. La forma de obtener los datos con el SDK es muy simple, basta con crear un objeto de tipo *Controller* el cual permite acceder al último cuadro capturado. Este cuadro es un objeto de tipo *Frame* que contiene toda la información de los puntos rastreables de la mano. Entre los datos obtenidos se tiene la posición de los dedos, palma y muñeca, así como también la orientación de la mano.

### *B. Brazo robótico*

El robot usado como modelo es el PhantomX Pincher AX-12 mostrado en la figura 3. Es un brazo robótico de 4 DOF compuesto de actuadores Dynamixel AX-12A. Las propiedades principales se muestran en la Tabla I.

**Fig. 3.** PhantomX Pincher Robot Arm

**TABLA I.** PROPIEDADES ROBOT PINCHER

| *Propiedad* | *Valor* |
|---|---|
| Alcance Vertical | 35 cm |
| Alcance horizontal | 31 cm |
| Alimentación | 12 V |
| Peso | 550 g |
| Capacidad de Carga a 25 cm | 40 g |
| Capacidad de Carga a 20 cm | 70 g |
| Capacidad de Carga a 15 cm | 100 g |

### *C. Actuadores Dynamixel AX-12A*

Los motores *Dynamixel* son actuadores programables de alto rendimiento con función de retroalimentacion, (fig. 4). Tienen un ID único para controlarlos por medio de un bus de comunicación. El control implementado es de tipo PID. La configuración y programación del Dynamixel puede ser hecha fácilmente usando el software *RoboPlus*. Las características principales se muestran en la tabla II.

**Fig. 4.** Actuador Dynamixel AX-12A


**TABLA II.** CARACTERÍSTICAS DE DYNAMIXEL AX-12A

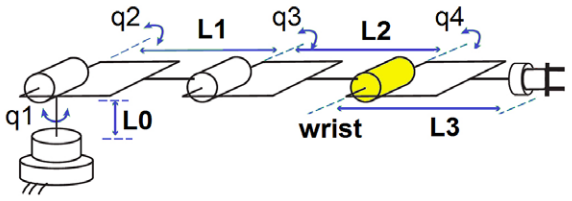
| *Característica* | *Valor* |
|---|---|
| Peso | 54.6 g |
| Dimensión | 32x50x40 (mm) |
| Relación de Transmisión | 254:1 |
| Interfaz de Red | TTL |
| Sensor de Posición (Resolución) | Potenciómetro (300o/1024) |
| Motor | Motor de núcleo |
| Voltaje de Operación | 9 12V |
| Velocidad sin carga | 59 RPM |

Estos actuadores cuentan con un SDK que contiene la librería de programación estándar en lenguaje C para el desarrollo de aplicaciones. Para controlar el Dynamixel directamente usando el PC se usa el dispositivo USB2Dynamixel. Cada Dynamixel posse una memoria de tipo EEPROM y RAM que almacena el estado y operación del actuador en la Tabla de Control. Por medio de funciones provistas en el SDK el usuario puede cambiar los datos en la Tabla de Control y así controlar el Dynamixel.

**TABLA III.** DIMENSIONES DE LOS ESLABONES DEL BRAZO ROBÓTICO

| *Eslabón* | *Parámetro* | *Dimensión* |
|---|---|---|
| 1 | $L_0$ | 0 *cm* |
| 2 | $L_1$ | 10.3 *cm* |
| *3* | $L_2$ | 10.3 *cm* |
| *4* | $L_3$ | 10.5 *cm* |

**Fig. 5.** Configuración del Robot




## III. CINEMÁTICA DEL BRAZO ROBÓTICO

### *A. Modelo geométrico directo*

El modelo geométrico directo consiste en determinar la posición del efector final, en particular el TCP (Tool Central Point), a partir de la información de los actuadores. Los actuadores son ubicados en cada una de la articulaciones del robot. Las variables articulares son denotadas por $q_n$ donde $n$ = 1,2,...,4. ver figura 5.

Se ha utilizado la convención de DH modificada para determinar el modelo directo, ver Tabla IV. Para facilitar el cálculo se establece $L_0 = 0$, de esta forma el origen del sistema de coordenadas para la primera y segunda articulación son el mismo.

**TABLA IV.** PARÁMETROS DH MODIFICADA

| i | $\theta_i$ | $d_i$ | $a_{i-1}$ | $alpha_{i-1}$ |
|---|---|---|---|---|
| 1 | $q_1$ | 0 | 0 | 0 |
| 2 | $q_2$ | 0 | 0 | 90º |
| 3 | $q_3$ | 0 | L1 | 0 |
| 4 | $q_4$ | 0 | L2 | 0 |

Se necesitan 6 parámetros para especificar totalmente la posición y orientación del efector final como un conjunto de coordenadas generalizadas de la forma $X = [x_p, x_O]^T$.

El vector $x_p = [x, y, z]$ corresponde al vector posición de la muñeca, donde $x, y, z$ son funciones dependientes de los ángulos $q_1$, $q_2$, y $q_3$. Para la configuración de la figura 5 estos parámetros se calculan de la siguiente forma.

$$r = L_1 \cos(q_2) + L_2 \cos(q2 + q3)$$

$$x = r \cos(q_1) \quad (1)$$

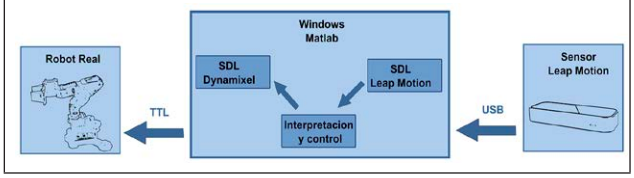
$$y = r \sin(q_1)$$

$$z = L_1 \sin(q_2) + L_2 \sin(q_2 + q_3)$$

Por otra parte, el vector $x_O = [\alpha, \beta, \gamma]$ corresponde al vector orientación del efector final, donde $\alpha, \beta, \gamma$ son funciones que dependen de los ángulos $q_1, q_2, q_3$ y $q_4$.

Sin embargo es posible representar la orientación mediante una matriz de rotación 3x3 la cual es útil como operador.

**Fig. 6.** Arquitectura del Sistema




Dicha forma matricial al multiplicarse por un vector realiza la operación de rotación. Para el robot de la figura 5 la representación matricial de la orientación del efector final con respecto al sistema coordenado de la base viene dada de la siguiente forma.

$$R_w = \begin{bmatrix} C_{234} * C_1 & -S_{234} * C_1 & S_1 \\ C_{234} * S_1 & -S_{234} * S_1 & -C_1 \\ S_{234} & C_{234} & 0 \end{bmatrix} \quad (4)$$

Donde $C_{234} = \cos(q2 + q3 + q4)$; $S_{234} = \sin(q2 + q3 + q4)$; $C_1 = \cos(q1)$; $S_1 = \sin(q1)$.

### *B. Modelo geométrico inverso*

La cinemática inversa consiste en obtener la configuración del brazo dadas las posiciones *x, y, z* de la muñeca (wrist). Una forma de lograrlo consiste en usar el método geométrico a fin de obtener los valores de los a ángulos $q1$, $q2$, y $q3$ en función de las coordenadas *x, y, z*.

El ángulo $q_4$ es manipulado de forma interactiva por el usuario, por lo que dicho parámetro no se considera en el cálculo.

$$
\begin{aligned}
q_1 &= atan2(y, x) \\
r &= \sqrt{x^2 + y^2} \\
c_3 &= (r^2 + z^2 - L_1^2 - L_2^2)/(2 * L_1 * L_2) \\
s_3 &= \sqrt{(1 - c_3^2)} \\
q_3 &= atan2(-s_3, c_3) \ [codo\ arriba] \\
q_2 &= atan2(z, r) - atan2(L_2 \sin(q_3), L_1 + L_2 \cos(q_3))
\end{aligned}
\tag{2}
$$

## V. DESARROLLO DEL MODELO

### *A. Estructura general del sistema*

El sistema se desarrolla en un computador x64 con sistema operativo *Windows*. En la figura 6 se observa un esquema desde el punto de vista del software y hardware. Los motores del brazo robótico se conectan al adaptador *USB2Dynamixel* usando el protocolo *TTL-level RS232*. Tanto el dispositivo Leap Motion como el USB2Dynamixel se conectan al computador por medio de puertos USB.

### *B. Interfaz en Matlab*

La integración funcional del sistema se realiza en el entorno Matlab, esto permite hacer uso del toolbox de robótica de Peter Corke para facilitar la visualización en tres dimensiones de la configuración actual del brazo robótico.

La interfaz de Matlab permite cargar librerías externas en memoria y acceder a las funciones de dichas librerías. Sin embargo, esta característica tiene la desventaja de funcionar solo con librerías desarrolladas en Lenguaje C. Debido a que las librerías del SDK del Leap Motion fueron desarrolladas en C++, es necesario crear un archivo propio de Matlab de tipo MEX-FILE que sí permite invocar las funciones de dicho SDK en Matlab.

La interfaz MEX de Matlab contiene solo las funciones necesarias para leer la posición de la muñeca y punta de los dedos, además de los ángulos *pitch*, *yaw* y *roll* de la palma de la mano.

### *C. Comandos gestuales*

El sistema reconoce tres comandos gestuales diferentes los cuales activan y desactivan el brazo robótico, y abren y cierran el gripper. Para ello se utiliza la distancia entre cada uno de los dedos y la distancia de la palma de la mano, de forma que cuando la mano está abierta las puntas de los dedos se encuentran lejos del centro de la palma lo que inicia el comando de encendido del robot. De la misma manera si la mano está cerrada, la distancia entre los dedos es pequeña lo que inicia la secuencia de apagado.

Por otra parte si la mano está extendida con los dedos separados unos de otros, se envía la señal para que el gripper abra, en caso contrario si los dedos se encuentran juntos se cierra el gripper.

### *D. Secuencia de Comandos*

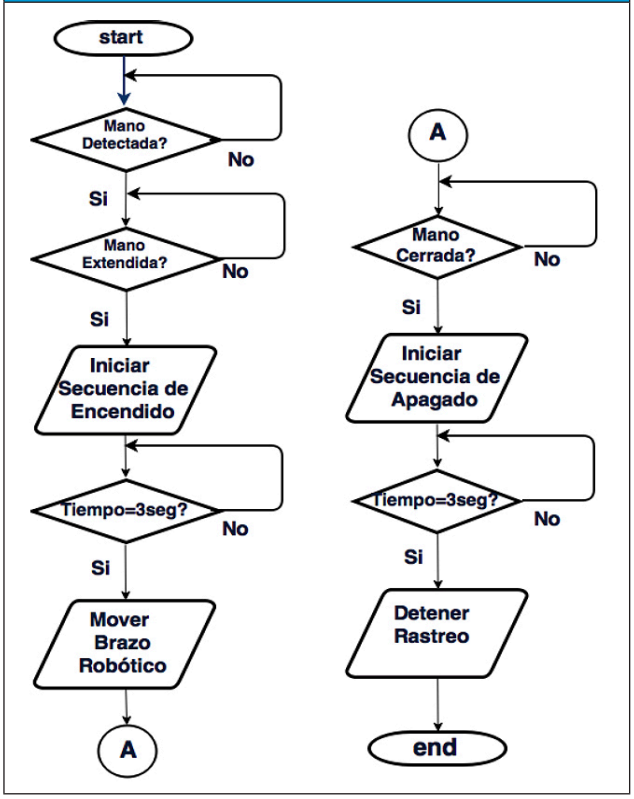
Al ejecutar la interfaz gráfica tiene lugar la secuencia de comandos mostrada en la figura 7. El sistema inicia comprobando si se detecta la mano, en caso afirmativo y si la mano se encuentra extendida se activa el temporizador de 3 segundos. Si el usuario cierra la mano antes de los 3 segundos, el temporizador reinicia desde cero. En estado de encendido, el sistema rastrea la posición de la muñeca del usuario y la convierte en posición de la muñeca del robot. Se calcula la cinemática inversa para que los actuadores adopten la configuración necesaria de forma que el robot llegue a dicha posición. Finalmente si el usuario cierra la mano, se inicia la secuencia de apagado y después de 3 segundos el sistema desactiva el rastreo y detiene el robot.

### *E. Configuración del brazo robótico*

Por cuestiones de seguridad y de rendimiento algunos parámetros de los motores Dynamixel son configurados antes de enviar los comandos de posicionamiento al robot. Inicialmente se limitan los ángulos que pueden tomar cada una de las articulaciones, y así evitar que cada eslabón choque con el eslabón inmediatamente anterior. Luego se

limita la velocidad y torque máximo a la que pueden funcionar los actuadores con el fin de generar movimientos más suavizados y evitar que movimientos bruscos generen oscilaciones indeseables en el brazo robótico.

**Fig. 7.** Flujo del Programa




De igual forma también se establece un espacio de trabajo en el cual el robot puede moverse libremente sin peligro de colisiones consigo mismo o con el entorno. En caso tal que el usuario trate de posicionarlo fuera de esta zona, la interfaz gráfica muestra una alerta visual y detiene el robot.

### *F. Módulos desarrollados*

Los módulos y funciones desarrolladas que conforman el sistema de control gestual del brazo robótico son los siguientes.

1) *ReadLeap:* Módulo simplificado y personalizado basado en la interfaz MEX de Matlab *MatLeap* [8]. Permite acceder a las funciones del SDK del Leap Motion.

2) *Rastreo de objetos:* Hace el seguimiento de los puntos de la mano, y controla las banderas correspondientes que indican la activación de comando gestuales o alertas de funcionamiento.

3) *Cinemática inversa del brazo robótico:* El módulo recibe como parámetros la posición objetivo del robot, y calcula los ángulos de posición de cada uno de las articulaciones para que el robot alcance la configuración requerida.

4) *Accionamiento del brazo robótico:* Este módulo tiene como parámetros de entrada los valores de posición de las articulaciones y del gripper. Se encarga de enviar los comandos necesarios a los actuadores Dynamixel.

5) *Visualización 3D:* El módulo de visualización 3D utiliza el *toolbox de robótica de Peter Corke* para mostrar la configuración de posición que se le está enviando al robot.

**Fig. 8.** Interfaz gráfica

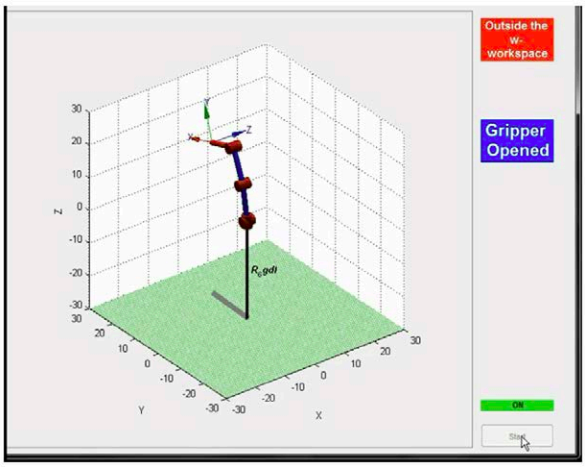
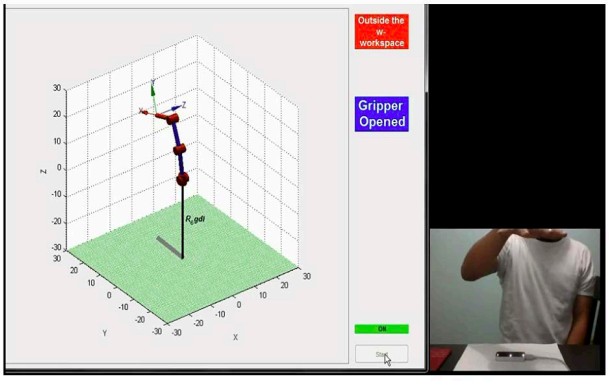



**Fig. 9.** Alerta - Fuera del espacio de trabajo

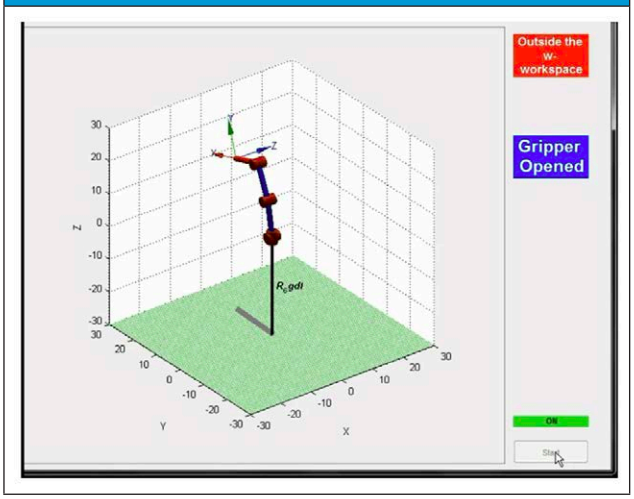


## V. INTERFAZ GRÁFICA Y OPERACIÓN

La ventana principal de la interfaz gráfica se compone de dos secciones bien definidas. A la izquierda se muestra la simulación 3D del brazo robótico, y en la derecha se muestran las alertas de funcionamiento. Ver figura 8.

Existen dos alertas en color rojo, la primera indica si el usuario trató de llevar el robot a una posición fuera del espacio de trabajo (fig. 9), mientras que la segunda indica si la mano se encuentra en el campo de visión del sensor Leap Motion. Una tercera alerta informa sobre el estado del gripper, de esta forma si es azul el gripper se encuentra abierto, y si es de color negro el gripper se encuentra cerrado. Una cuarta alerta informa sobre el estado de encendido o apagado del robot (fig. 10), es de color rojo si el robot esta en reposo, amarilla si se ha iniciado el proceso de encendido o apagado, y verde cuando el robot está en función de rastreo.

**Fig. 10.** Comando gestual de encendido




## VI. RESULTADOS

Implementaciones con el robot real muestran que el sistema reconoce satisfactoriamente los comandos gestuales y ejecuta las acciones correspondientes de manera apropiada.

En lo que respecta al movimiento del robot, se observa que la velocidad limite configurada inicialmente en los actuadores del robot es clave en el funcionamiento óptimo del posicionamiento del robot. Pruebas realizadas con diferentes valores de velocidad límite muestran que velocidades de más de 10% de la velocidad máxima del actuador generaron oscilaciones por la misma dinámica del robot que hace que la tarea se lleve a cabo de forma no suavizada. Se estableció que que la velocidad ideal para posicionar el robot de un punto a otro corresponde al 5% de la velocidad máxima del Dynamixel.

Al igual que con la velocidad límite, también se desarrollaron pruebas con la configuración del torque límite del Dynamixel. Dichas pruebas mostraron que valores de torque límite superiores al 40% del torque máximo del Dynamixel producen oscilaciones indeseables en el brazo robótico. Se evidenció que el mejor rendimiento se consigue cuando los actuadores están configurados al 30% de su torque máximo.

## VII. CONCLUSIONES

No se puede 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.

## AGRADECIMIENTOS

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

## REFERENCES

[1] Xie, Q., Liang, G., Tang, C., Wu, X, Robust Gesture Based Interaction System for Manipulating Service Robot, 2013.Third International Conference on Information Science and Technology. 658–662.

[2] Panwar, M. (n.d.), Hand Gesture Recognition based on Shape Parameters., 2013.Centre for Development of Advanced Computing 658–662.

[3] Segen, J., Kumar, S., Laboratories, B. (n.d.), Fast and Accurate 3D Gesture Recognition Interface, 2013.Bell Laboratories, Holmdel.

[4] M. Gad-el-Hak, The MEMS Handbook: MEMS Design and Fabrication, 2nd ed. Boca Raton, FL: CRC Press, 2006.

[5] M. Gad-el-Hak, The MEMS Handbook: MEMS Design and Fabrication. Alemania: ScienceDirect, 2006.

[6] M. Gad-el-Hak, The MEMS Handbook: MEMS Design and Fabrication. Springer Science & Business Media, 2012.

[7] M. Gad-el-Hak, The MEMS Handbook: MEMS Design and Fabrication. Sandia National Laboratories, Albuquerque, New Mexico, USA: sandia Report, 2004.

[8] J. Perry, Matleap MEX Interface for the Leap Motion Sensor, 2014. URL: https://github.com/jeffsp/matleap [acceso: 2015- 03-15].