DISEÑO E IMPLEMENTACIÓN DE UN DISPOSITIVO DETECTOR DE PROFUNDIDAD DE APOYO PARA PERSONAS INVIDENTES

TESIS

Para obtener el grado de

INGENIERO EN COMPUTACIÓN

Presenta

José Francisco Mercado Miramontes

M.C. José Ortíz Bejar

Director de Tesis

M.C. Garibaldi Pineda García

Co-Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ingeniería Eléctrica

Morelia, Michoacán – Agosto 2013

Resumen

La tesis presenta el diseño e implementación de un dispositivo detector de profundidad de apoyo para personas invidentes, el dispositivo tiene como función principal, obtener las diferentes distancias de los objetos en el ángulo de visión del sensor y enviar una señal de pulsos vibrantes, mediante motores para indicar la ubicación y la distancia de dichos objetos. El dispositivo es de apoyo porque no tiene un ángulo de visión completo, por lo que requiere de un bastón comercial para invidentes. Utilizando un sensor llamado Kinect, que originalmente fue diseñado para ser un mando de videojuegos, se toman los datos de profundidad de la imagen, mediante el método de escaneo 3D Luz Estructurada.

Para procesar los datos se utiliza una computadora de tamaño reducido, llamada BeagleBone. Se procesan secciones de la imagen de profundidad captada por el sensor Kinect y se transforman estas distancias en vibraciones, generadas por unos micro motores, la frecuencia de los pulsos nos indica la distancia de los objetos al usuario(a mayor frecuencia mas próximo).

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos

D	Desplazamiento de un punto conocido a un punto en el plano de la imagen.
Zo	Distancia predeterminada de Kinect.
$Z1$	Distancia de profundidad del objeto.
Xo	Distancia entre proyector y cámara infrarroja de Kinect.
$X1$	Distancia entre cámara infrarroja y el punto objeto.
f	Distancia focal de la cámara de infrarrojo.
d	Disparadidad correspondiente a un punto.
t_{on}	Tiempo de encendido.
t_{off}	Tiempo de apagado.

Capítulo 1

Introducción

1.1. Justificación

En la actualidad existen dispositivos de bajo costo y buena capacidad de procesamiento para realizar actividades de visión computacional móvil. Uno de los problemas más antiguos es el de poder brindar o sustituir el sentido de la vista en personas con debilidad visual avanzada. La razón de esta tesis es aplicar la visión computacional para el desarrollo de un dispositivo de ayuda para el sistema visual humano, ya que hay poca investigación de este tipo de dispositivos[Manduchi12]. Gracias a la tecnología actual, se pueden aprovechar aquellos dispositivos comerciales, y aplicando el conocimiento obtenido en la carrera e investigando, crear un dispositivo de bajo costo que ayude a las personas con debilidad visual.

1.2. Objetivo

El objetivo de la tesis es desarrollar un dispositivo de apoyo para las personas invidentes, el cual, detecta la distancia de los objetos en un rango de visión. Dependiendo de la cercanía de los objetos al dispositivo, emitirá una serie de pulsos vibratorios que se incrementarán a medida que los objetos se acerquen al dispositivo.

1.3. Metodología Utilizada

El dispositivo a desarrollar consta de un sensor, ya que se necesita estimar la distancia de los objetos a la persona, una computadora o un dispositivo que permitierá procesar esta información y un medio de salida de ésta, en este caso motores vibrantes.

El sensor utilizado se llama Kinect®(Figura 1.1), fue creado para ser un dispositivo de mando para los juegos de video de la consola Xbox 360®[1]. Éste cuenta con una cámara RGB (color, rojo-verde-azul, por sus siglas en inglés), un sensor de profundidad por infrarrojos compuesto por un emisor y proyector de rayos infrarrojos, un motor que realiza una inclinación mecánica para cambiar el ángulo de visión y cuatro micrófonos.

La forma en que el sensor calcula la profundidad es con un método de escaneo en 3D llamado "Luz Estructurada"[Khoshelham], es un método rápido y efectivo donde se obtienen muestras suficientes para aproximar la profundidad de los elementos en la "imagen".

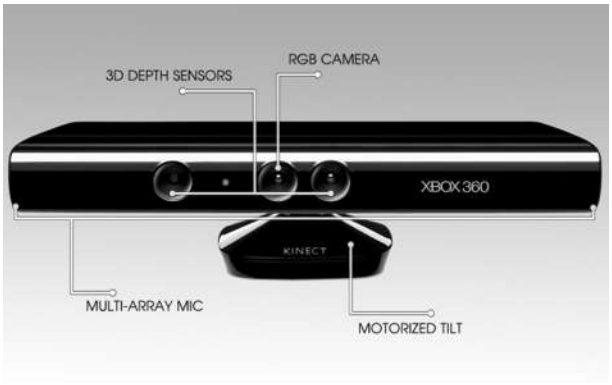


Figura 1.1. Dispositivo Kinect y sus componentes.

El dispositivo se pudo conectar a la computadora gracias a que la comunidad llamada OpenKinect desarrolló el driver llamado Freenect para poder controlar el sensor. Por otro lado la compañía creadora de la tecnología Kinect, PrimeSense®lanza OpenNI, que son las librerías de Kinect para uso público. Utilizando estas librerías se puede trabajar en

[1]Consola de videojuegos creada por Microsoft.

múltiples lenguajes de programación, dado que ya se tenía un conocimiento previo del lenguaje, se eligió el lenguaje C.

Como el sensor debía estar en movimiento y conectado a una computadora, se buscó un dispositivo compacto y económico, la BeagleBone®fue el hardware elegido para procesar los datos del Kinect.

BeagleBone(Figura 1.2) es una barebone[2] de bajo consumo de energía electrica, con salidas de propósito general (GPIO)[3], USB (Universal Serial Bus) y Fast Ethernet, cuenta con procesador ARM Cortex-A8, modelo AM3359, 256 MB de RAM y una ranura SD.

El propósito de esta computadora es que, a través de ella los datos obtenidos del Kinect sean procesados y sean enviados a los motores vibrantes mediante las GPIO.

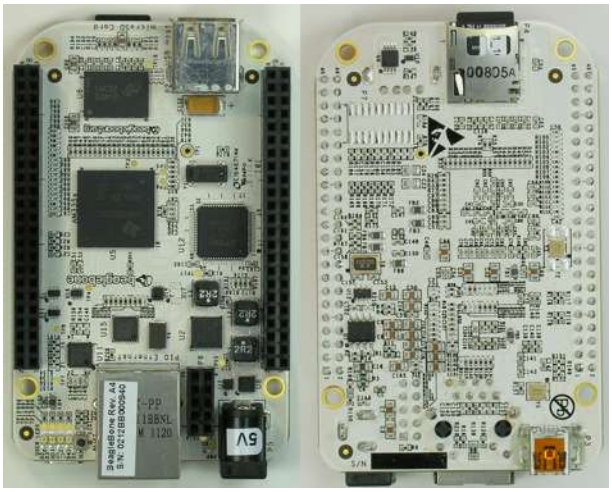


Figura 1.2. Barebone BeagleBone.

[2]Computadora semi ensamblada, reducida en dimensiones fisicas que viene con placa y RAM específica.

[3](General Purpose Input/Output) Es un tipo genérico de pin que se encuentra en un procesador y puede ser programado durante la ejecución, se utiliza para tanto la entrada como salida de datos.

1.4. Descripción de los Capítulos

En el segundo capítulo se presentan los componentes y técnicas utilizadas para hacer este dispositivo, se especifica con detalle cada uno de ellos.

En el tercer capítulo se explica cómo se conjuntan los elementos antes mencionados, para poder ensamblar el dispositivo, se explica brevemente la lógica del programa creado para este propósito.

En el cuarto capítulo se explican las pruebas a las que se sometió el dispositivo.

En el quinto capítulo se exponen las conclusiones a las que se llegaron, así como de la mejorías que se pueden hacer en un futuro.

Capítulo 2

Componentes

2.1. Sensor Kinect

Kinect es un dispositivo de detección de movimiento de Microsoft para la consola de videojuegos Xbox 360, que permite interactuar con los juegos y la consola sin la necesidad de tocarles y funciona a través de gestos y comandos de voz. Fue exhibido por primera vez el 1 de junio del 2009 en la Electronic Entertainment Expo (E3) bajo el nombre de Project Natal. El 13 de junio del 2010 fue lanzado al mercado bajo el nombre oficial Kinect, que es un acrónimo de Kinetic y Connect (cinético y conexión) que son los aspectos fundamentales del dispositivo.

Se basa en la tecnología de software desarrollado por Rare®, una subsidiaria de Microsoft Game Studio®que trabajó en esa parte del periférico de Kinect, y en la tecnología de cámaras por la empresa israelí PrimeSense, la cual desarrolló un sistema que puede interpretar gestos específicos y así controlar dispositivos electrónicos mediante el uso de un proyector de infrarrojos, una cámara y un microchip especial para seguir el movimiento de las personas.

El sensor Kinect (Figura 2.1) cuenta con las siguientes medidas 28cm de largo, 6cm de ancho, tiene una cámara y proyector infrarrojos, una cámara de color RGB, cuatro micrófonos y mecanismo de inclinación (tilt mécanico).

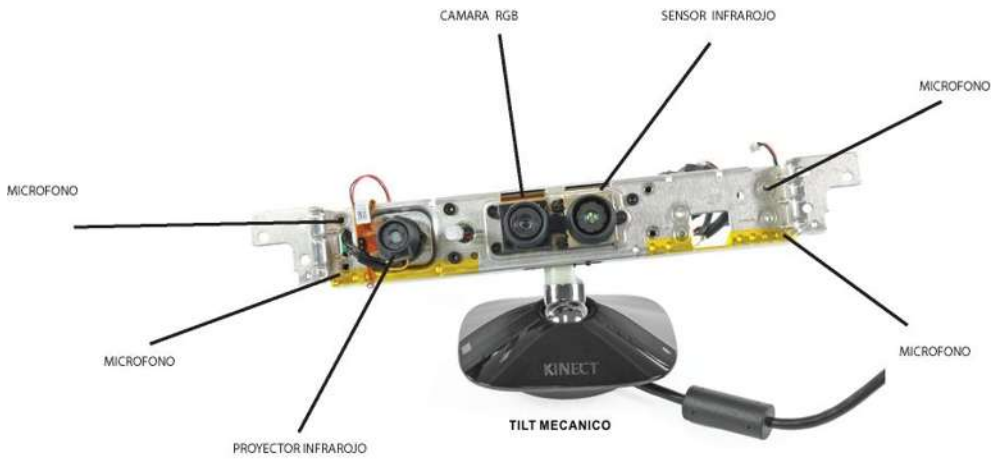


Figura 2.1. Lugar de los componentes de Kinect.

La frecuencia de refresco de la imagen es 30Hz, el video RGB utiliza 32 bits en total y la detección de profundidad de 11 bits. La cámara RGB tiene una resolución de VGA (640×480 pixeles) al igual que la cámara de profundidad. El sensor tiene un campo de visión de 57º horizontal y 43º vertical, con el inclinador mécanico se puede inclinar hasta 27º de arriba hacia abajo y viceversa. La distancia mínima de visión horizontal es de 87cm y la vertical es de 63cm. Tiene cuatro micrófonos en las esquinas del sensor que operan con 16 bits de audio y una velocidad de 16 Khz. Debido a que el Kinect necesita mas energía eléctrica de la que puede aportar un puerto USB, se requirió de un conector que permite la combinación del dispositivo con alimentación adicional[Robotics].

El límite práctico (Figura 2.2) va de 1.2m a 3.5m, cuando se detecta y rastrea el movimiento de una persona (Skeletal Tracking). El área requerida para utilizar el Kinect es de $6m^2$ aunque puede mantener el seguimiento a través de una gama extendida de 6m a 8m de profundidad. Este dispositivo está diseñado para interiores, es decir, lugares con poca radiación infrarroja.

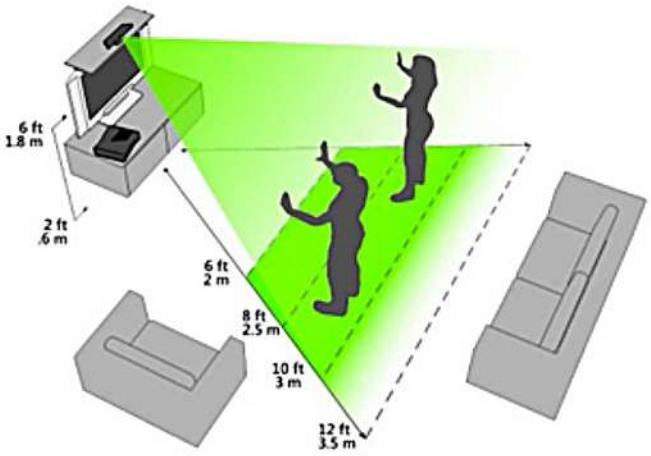


Figura 2.2. Área óptima para detectar el movimiento.

Para poder procesar todas estas acciones, el Kinect utiliza un circuito integrado PS1080 (Figura 2.3)[Ladislao Mathe] con el propósito de controlar todos los componentes del dispositivo y obtener la imagen de profundidad, imagen de color, procesar el audio y el movimiento del tilt mecánico, para luego enviar los datos procesados por el puerto USB.

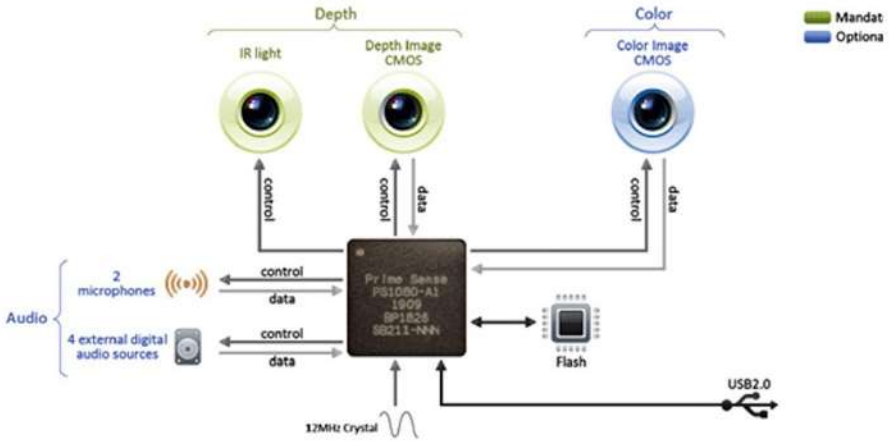


Figura 2.3. Diagrama del chip PS1080.

2.2. Luz Estructurada

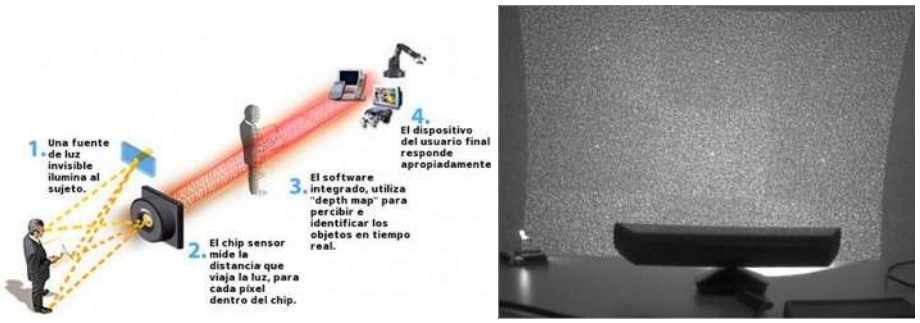
Los escáneres 3D son dispositivos que analizan un objeto para reunir información de su forma, y construir modelos digitales. Existen dos tipos de escáner 3D que son: de contacto y sin contacto.

Los escáner 3D de contacto son aquellos que examinan un objeto utilizando un elemento de medición, que tiene que estar en contacto con el objeto. Lo más tradicional es un palpador, que es una barra con punta redonda que transmite un movimiento al contacto con la pieza, éste envía datos a un sensor y escanea el objeto tocándolo. La principal desventaja de este método es que el objeto a escanear puede ser dañado, además éste método de escaneo es muy lento en comparación con otros.

Los escáner 3D sin contacto son aquellos en los que no se tiene un contacto físico con el objeto a escanear. Esta clasificación, a su vez, se divide en pasivos y activos. Los activos son aquellos que emiten varios haces de luz, que se utilizan para calcular la distancia del escáner al objeto. Los escáneres pasivos utilizan la luz natural para realizar dicho proceso. Entre los distintos modelos de escáneres activos se encuentran algunos que utilizan "luz estructurada".

La Luz Estructurada (Figura 2.4)[Ladislao Mathe] es un método donde se proyectan rayos infrarrojos, los cuales no son visibles al ojo humano, en un patrón de puntos preestablecidos(Figura 2.4(b)). Este es el método de estimación de profundidad que utiliza el Kinect, el patrón antes mencionado se genera aleatoriamente y se fija al momento de fabricar el dispositivo. En la fabricación del Kinect se calibra la imagen patrón para que cada píxel donde ocurra una intersección del patrón con la superficie de prueba tenga una distancia conocida. En este trabajo a este patrón de referencia se le llamará "primera imagen". Con los datos obtenidos en tiempo real se generará una imagen infrarroja que, en este trabajo, se conocerá como "segunda imagen". Posteriormente, se compara la primera y segunda imágen, y para cada píxel donde ocurrió una diferencia en la localización del pixel de referencia, se realiza una operación para estimar la profundidad del píxel comparado. Es

importante notar que el Kinect realiza estas operaciones 30 veces por segundo.



(a) **(b)**

Figura 2.4. Pasos del sensor Kinect para detectar la profundidad.

Kinect es un aparato "cerrado", es decir, no se conocen al 100 % las funciones de sus circuitos y no existe documentación disponible al público general que indique tales atribuciones. Hasta donde se pudo investigar, se utiliza el método de triangulación. La triangulación es un método para determinar la profundidad de un pixel patrón dado en un punto en el espacio en 3D a partir de dos imágenes que son: una distancia conocida y la que se obtiene a partir del rebote del rayo infrarrojo con el objeto, como se observa en la Figura 2.5[Khoshelham]. En la opinión del autor, ésta es la explicación más probable dado que no hay información oficial disponible, y éste método es el que obtuvo mayor consenso de la información disponible en los grupos en línea como OpenKinect.

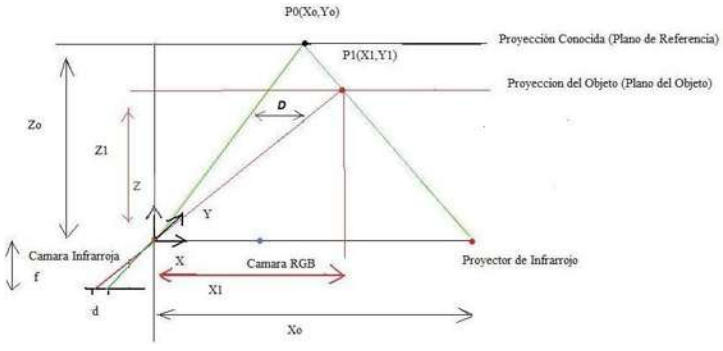


Figura 2.5. Triangulación del sensor Kinect.

A continuación se presenta una posible explicación de la estimación de profundidad hecha por el Kinect. Se tiene el plano de referencia, que es una proyección conocida con el origen en el centro de la cámara a una distancia conocida *Zo*. También existe una distancia conocida entre proyector y cámara infrarrojos, que es la distancia base *Xo*, la distancia focal de la cámara de infrarrojo *f*. La distancia entre el punto patrón y el que se supone es su correspondiente en la imagen en tiempo real *d* se estima cada que se obtiene un nuevo fotograma.

El punto *P1(X1,Y1)* es aquel cuya profundidad se desea estimar, este se desplaza hacia la izquierda o derecha del punto de referencia *P0(Xo,Yo)*, dependiendo de la profundidad, esto se crea una distancia *D* que es el desplazamiento del punto *P1* al punto *P0* en el plano de la imagen, *Z1* es la distancia de profundidad del objeto en el punto *P1*, de la semejanza de triángulos se obtienen las siguientes ecuaciones:

$$\frac{D}{Xo} = \frac{Zo - Z1}{Zo} \tag{2.1}$$

$$\frac{d}{f} = \frac{D}{Z1} \tag{2.2}$$

Al sustituir la ecuación (2.2) en la ecuación (2.1) y despejando *Z1* se obtiene (ver [Khoshelham]):

$$Z1 = \frac{Zo}{1 + \frac{Zo}{fXo}d} \tag{2.3}$$

De la ecuación (2.3) se obtiene la distancia de profundidad, donde Kinect ya cuenta con los siguientes datos conocidos que son: la profundidad inicial *Zo*, la distancia focal *f*, y la distancia base *Xo*, lo que Kinect necesita es la disparadidad correspondiente *d* que se obtiene de la imagen patrón y la recién adquirida. No conocemos la distancia predeterminada exacta que tiene el Kinect ya que no existe una documentación oficial donde se especifiquen estos datos y el funcionamiento exacto, pero la triangulación es la explicación más viable.

Como es necesario procesar los datos obtenidos por el Kinect en movimiento, se utilizó una computadora llamada BeagleBone.

2.3. BeagleBone

BeagleBone es una barebone de alto desempeño que trabaja con bajos consumo de voltaje y corriente. Fue inicialmente producida y distribuida por Digi-Key en asociación con trabajadores de Texas Instruments, actualmente Digi-key ya no es el distribuidor exclusivo, por lo que, se puede adquirir por otras vías.

BeagleBone (Figura 2.6) cuenta con las siguientes características técnicas: tiene un procesador ARM Cortex-A8, modelo AM3359, trabaja a 720MHz, memoria RAM 256MB, puerto Fast Ethernet, entrada para microSD, 4 LEDs, dos puertos USB 2.0, uno de cliente y otro de host, 2 conexiones de expansión de 46 pines con diferente función cada uno. Las partes que se utilizaron son los puertos USB cliente y host, y las ranuras de expansión. El USB cliente se utilizó para alimentar al BeagleBone, el USB host se utilizó para conectar el Kinect y las ranuras de expansión para entregar la señal de activación a los motores.

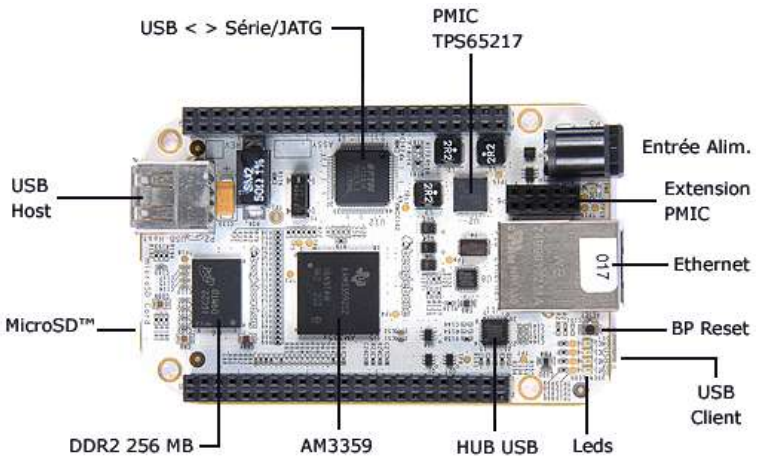


Figura 2.6. Componentes del BeagleBone.

2.4. Hub (Concentrador) USB con Alimentación Eléctrica Externa

El puerto USB del BeagleBone no entrega la suficiente potencia para poder hacer funcionar cualquier dispositivo USB, por lo que se utilizó un Hub USB con alimentación eléctrica externa (Figura 2.7). Este componente es capaz de concentrar la conexión de varios dispositivos USB a través de un solo puerto. El puerto es una versión USB 2.0, al igual que el USB del BeagleBone. Éste puede soportar dispositivos de alta velocidad, de hasta 480 Mb/s y tiene un límite teórico de 127 puertos simultáneos, pero la mayoría tiene de 4 a 7 puertos.

Figura 2.7. Hub USB 2.0 con alimentación eléctrica externa de 7 puertos.

2.5. Router y Acceso Remoto

Se utilizó para poder acceder y ejecutar algunas acciones desde un equipo local al BeagleBone, a través del protocolo SSH (Secure SHell, por sus siglas en inglés), a la BeagleBone, gracias a la tarjeta de red del con que cuenta el BeagleBone se creó una red local, utilizando un enrutador (Figura 2.8). Se hizo uso de una computadora de escritorio para acceder a la BeagleBone y así poder instalar los componentes necesarios para controlar el Kinect. Se empleó este mismo tipo de acceso para crear, compilar y ejecutar el programa desarrollado directamente en el BeagleBone.

Figura 2.8. Router Linksys WRT54G.

2.6. Micro Motores

Las salidas del dispositivo desarrollado son vibraciones, para realizarlas se utilizarón micro motores (Figura 2.9), ya que son dispositivos muy pequeños y consumen poca energía. Además éstos permiten aumentar la frecuencia de pulsos mientras más cerca se encuentra el usuario a los objetos y disminuirla a medida que se alejan. Las personas al perder un sentido o al nunca haberlo tenido, desvían la entrada de las neuronas no estimuladas, por lo que los otros sentidos tienden a agudizarse[Amedi03]. La hipótesis sobre el uso del dispositivo desarrollado es que, la persona que lo utilice podrá sentir las vibraciones sin problemas. Si se utiliza otro medio de salida, como el sonido, podría obstaculizar la audición de la persona y sería un riesgo en algunos ambientes.

Las ventajas de este tipo de motores es que son: discretos, económicos y trabajan con poca potencia. Los motores se obtuvieron de diversos celulares, no se encontraron hojas de datos para los motores, por lo que se recurrió a hojas de datos de motores con caracteristícas físicas similares. De estos documentos se pudo concluir que los motores funcionan óptimamente a un voltaje de 3.3 Volts y una corriente de 100 Miliamperes.

Figura 2.9. Micro motores.

2.7. Conclusiones

En este capítulo se describieron las características relevantes de los componentes utilizados, así como sus especificaciones. Éstos dispositivos destacan por ser bastante versátiles y económicos en comparación con otros productos utilizados en la investigación.

Kinect es el primer escáner de aplicación comercial que utiliza la técnica de luz estructurada, destaca por ser un escaner rápido, económico, compacto y accesible.

La BeagleBone en comparación con otras barebones comerciales, es una de las más baratas, pequeñas y completas, pues cuenta con un gran número de entradas y salidas de propósito general.

Para la realización del dispositivo, fue necesario buscar aquellos componentes que tuvieran la funcionalidad deseada, que fueran económicos y de bajo consumo de energía.

En el siguiente capítulo se abordan a detalle los pasos de la creación del dispositivo con los componentes descritos en este capítulo.

Capítulo 3

Desarrollo del Dispositivo

3.1. Conexión de los Componentes

La conexión del dispositivo (Figura 3.1) se hizo de la siguiente manera:

- El puerto USB host del BeagleBone se conectó a un Hub USB con alimentación eléctrica independiente.
- El Kinect se conectó a uno de los puertos del Hub para que éste le diera la energía suficiente para interactuar con el BeagleBone.
- Se conectaron los motores a los pines del BeagleBone.

Figura 3.1. Conexión de los componentes.

Para hacer funcionar el sensor Kinect fue necesario un controlador[4] y librerías[5] obtenidas de *openkinect.org*.

OpenKinect es una comunidad de personas cuyo propósito es crear y mejorar la interacción del Kinect con la PC. Esta comunidad lanzó su controlador, llamado Freenect, para poder acceder al dispositivo y así poder interactuar con las librerías más ampliamente. Sabiendo esto, se instalaron las librerías de Kinect llamadas libfreenect en la PC y en el BeagleBone.

3.2. Sensor Kinect y Detección de Profundidad

Kinect toma la profundidad de la imagen, que es la distancia de los objetos al sensor en la imagen y lo guarda en una matriz de 640×480 pixeles, cada pixel tiene 11 Bits de profundidad, es decir, un valor que va de 0 a 2047. Kinect admite dos rangos de distancia modo por defecto y el modo cerca (Figura 3.2)[Ladislao Mathe], el modo por defecto se utiliza cuando se juega en la consola y el modo de cerca que se utiliza va desde un mínimo de 0.4 metros hasta un máximo de 8 metros.

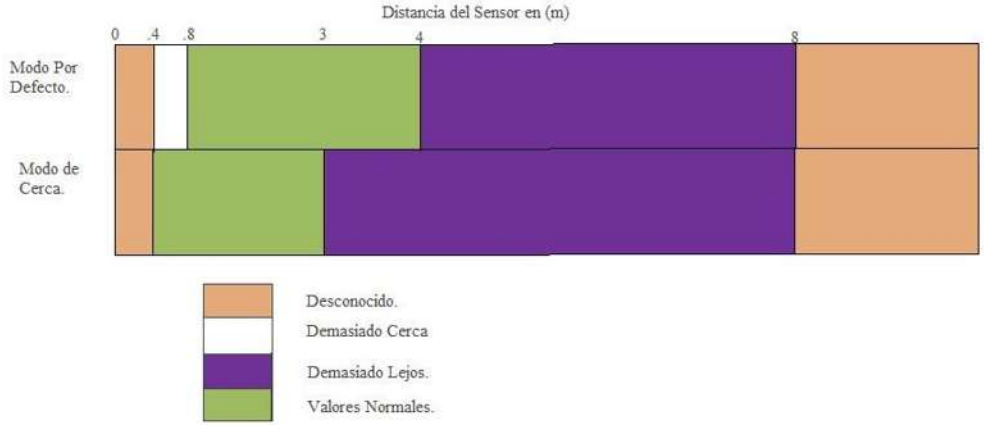


Figura 3.2. Rango de profundidad del sensor a los objetos.

[4]Programa que permite al sistema operativo comunicarse con un dispositivo de hardware para que éste pueda ser controlado.

[5]Conjunto de funciones, utilizadas para desarrollar programas.

En los foros de OpenKinect, Stéphane Magnenat[OpenKinect] propuso una fórmula obtenida en base a prueba y error, donde se asigna un valor de distancia a cada nivel de profundidad, se guardaron estos valores en un vector de 0 a 2047 que en esta tesis se llamará "gamma".

La función "freenect_sync_get_depth" de las librerías, se encuentra una variable llamada "depth"que es una matriz de 640×480 pixeles, en esta matriz está el nível de profundidad de la imagen de cada píxel, un valor entre 0 a 2047. Posteriormente los pixeles se comparan con la posición del vector gamma correspondiente y entrega la distancia en centímetros.

Con la fórmula (3.1) se obtuvo una mejor aproximación al valor de la profundidad, por lo que se utilizó para estimar la distancia.

$$DistanciaEnCentimetros = 100(0.1236[tan(\frac{i}{2842.5 + 1.1863})]) \tag{3.1}$$

Donde *i* es la posición en el vector, un valor del conjunto [0, 2047].

Durante las pruebas iniciales a la fórmula (3.1) se obtuvieron las distancias del Kinect hasta objetos a diferentes profundidades. Para filtrar las distancias que no eran de utilidad se optó por encontrar la distancia mínima en pantalla.

Posteriormente, para el funcionamiento del aparato desarrollado, la pantalla se dividió imaginariamente en una cuadrícula de 6 secciones, para obtener 6 diferentes lecturas de profundidad, correspondientes a cada uno de los motores. Se obtuvieron las distancias de cada una de las secciones, entonces al conocerlas fue posible enviar la información a los motores vibrantes para que indiquen su cercanía. La Figura 3.3 muestra las coordenadas de los píxeles centrales que estos son: (x0=107 ,y0=120), (x1=319,y1=120), (x2=533,y2=120), (x3=107,y3=360), (x4=319,y4=360), (x5=533,y5=360).

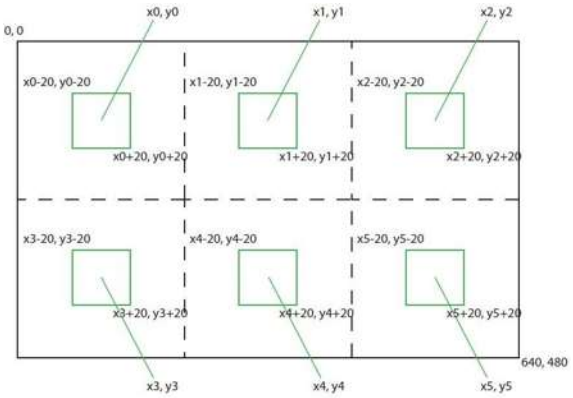


Figura 3.3. Secciones de la pantalla y píxeles centrales.

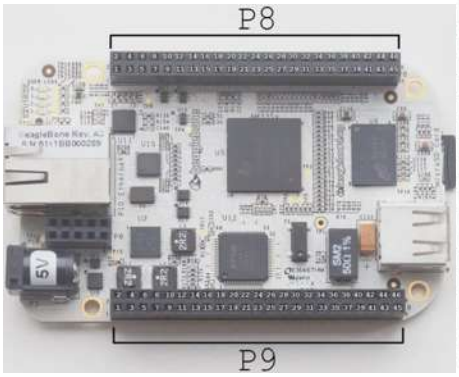
La manera de enviar la señal vibrante de salida es a través de los pines[6] de la BeagleBone. Ésta cuenta con dos cabeceras de expansión llamadas P8 y P9 (Figura 3.4)[Richardson] de 46 pines cada uno.

La tabla de la Figura 3.4 muestra la configuración de las cabeceras de expansión del BeagleBone[BeagleBone]. A continuación se lista el significado de cada columna:

- **SIGNAL NAME**: Es el nombre de la señal del pin.
- **PROC**: Es el número del pin en el procesador.
- **CONN**: Es el número del pin en la cabecera de expansión.

Las cabeceras tienen 4 clases de pines: GPIO entrada/salida de propósito general(General Purpose Input/Output), UARTS puertos seriales, TIMER salidas de temporizadores y PWM salidas PWM(Pulse Width Modulation).

[6]Contacto metálico de un componente para la transmisión de electricidad o información.



Expansion Header P8 Pinout

SIGNAL NAME	PROC	CONN		PROC	SIGNAL NAME
	GND	1	2	GND	
GPIO1_6	R9	3	4	T9	GPIO1_7
GPIO1_2	R8	5	6	T8	GPIO1_3
TIMER4	R7	7	8	T7	TIMER7
TIMER5	T6	9	10	U6	TIMER6
GPIO1_13	R12	11	12	T12	GPIO1_12
EHRPWM2B	T10	13	14	T11	GPIO0_26
GPIO1_15	U13	15	16	V13	GPIO1_14
GPIO0_27	U12	17	18	V12	GPIO2_1
EHRPWM2A	U10	19	20	V9	GPIO1_31
GPIO1_30	U9	21	22	V8	GPIO1_5
GPIO1_4	U8	23	24	V7	GPIO1_1
GPIO1_0	U7	25	26	V6	GPIO1_29
GPIO2_22	U5	27	28	V5	GPIO2_24
GPIO2_23	R5	29	30	R6	GPIO2_25
UART5_CTSN	V4	31	32	T5	UART5_RTSN
UART4_RTSN	V3	33	34	U4	UART3_RTSN
UART4_CTSN	V2	35	36	U3	UART3_CTSN
UART5_TXD	U1	37	38	U2	UART5_RXD
GPIO2_12	T3	39	40	T4	GPIO2_13
GPIO2_10	T1	41	42	T2	GPIO2_11
GPIO2_8	R3	43	44	R4	GPIO2_9
GPIO2_6	R1	45	46	R2	GPIO2_7

Figura 3.4. BeagleBone número de pin y hoja de cabecera de expansión P8.

Se utilizaron, los pines GPIO con etiqueta PROC R. Para poder activarlos, primero se debe conocer el número de pin, para eso se reviso la tabla de los pines del manual de referencia de BeagleBone y se obtuvo el número de activación del pin con la fórmula (3.2).

$$NumeroDeActivacion = GPION\_M = (N * 32) + M \tag{3.2}$$

En la fórmula (3.2) el número 32 es una constante y las variables son los números que contiene el Signal Name que son: número de pin (M) y número de chip (N).

Ya que se obtuvo el número de activación del pin es necesario seguir ciertos pasos para poder utilizarlo. El diagrama de flujo para activar y desactivar el pin se muestra en la Figura 3.5.

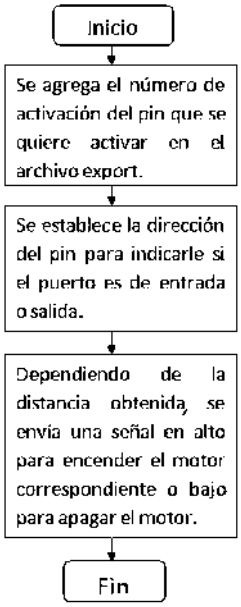


Figura 3.5. Diagrama de flujo para activar y desactivar los pines.

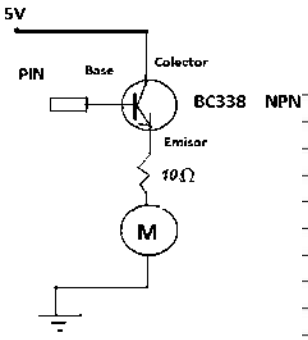
A través de la terminal, teniendo los privilegios de administrador, se utilizan las instrucciones de la Tabla (3.1) para activar o desactivar el pin y así poder controlar la vibración de los motores.

Instrucción.	Acción.
echo 38 >/sys/class/gpio/export	Se indica el pin a activar, en este caso el 38.
echo out >/sys/class/gpio/gpio38/direction	Indica que el puerto es de salida.
echo in >/sys/class/gpio/gpio38/direction	Indica que el puerto es de entrada.
echo 1 >/sys/class/gpio/gpio38/value	Indica que el estado de salida es alto 3.3V
echo 0 >/sys/class/gpio/gpio38/value	Indica que el estado de salida es bajo 0V
echo 38 >/sys/class/gpio/unexport	Se indica el pin a desactivar, en este caso el 38.

Tabla 3.1: Tabla de ejemplo de instrucciones necesarias para activar o desactivar el pin 38.

Como la BeagleBone no puede suministrar la corriente necesaria para el correcto funcionamiento de los motores, se utilizó un transistor para permitir el paso de corriente de una fuente externa. El circuito utilizado se muestra en la Figura 3.6. Cada circuito consta de un transistor BC338, una resistencia de 10 Ω y trabaja con una fuente de alimentación de 5 volts.

Cada motor está asociado a una sección de la pantalla y a un pin del BeagleBone. Los pines utilizados (Figura 3.6)[BeagleBone] fueron:29, 30, 43, 44, 45, 46 de la cabecera de expansión P8, se obtuvó su número de activación de pin de la información del fabricante.



GPIO2_23	R5	29	30	R6	GPIO2_25
UART5_CTSN	V4	31	32	T5	UART5_RTSN
UART4_RTSN	V3	33	34	U4	UART3_RTSN
UART4_CTSN	V2	35	36	U3	UART3_CTSN
UART5_TXD	U1	37	38	U2	UART5_RXD
GPIO2_12	T3	39	40	T4	GPIO2_13
GPIO2_10	T1	41	42	T2	GPIO2_11
GPIO2_8	R3	43	44	R4	GPIO2_9
GPIO2_6	R1	45	46	R2	GPIO2_7

Figura 3.6. Circuito propuesto para cada motor y hoja de cabecera de expansión P8.

Las secciones de pantalla se asignaron a los pines como se muestra en la Tabla (3.2).

Sección en la Imagen.	Pin del BeagleBone.	No Pin Activación
1	29	87
2	30	89
3	43	72
4	44	73
5	45	70
6	46	71

Tabla 3.2: Tabla de asignación de los motores.

3.3. Creación del Programa

El código fuente (publicado en los anexos) que se empleó para este dispositivo está escrito en lenguaje C, la codificación se hizo lo más simple posible para que cumpliera con lo necesario para el funcionamiento del dispositivo.

El diagrama de flujo de la Figura 3.7 presenta la lógica del programa; se comenzó por establecer los níveles de valores gama con la fórmula (3.1) de la distancia. Se crea un

vector llamado “gamma”, cada uno de sus elementos liga un nivel de profundidad a la distancia en centímetros. Se obtiene la matriz de profundidad, para posteriormente obtener la distancia de cada una de las secciones de la pantalla. Después de obtenerla, se envía la información de vibración a los pines, los cuales, se prenderán o se apagarán de acuerdo a la distancia obtenida. Finalmente, se vuelve a iniciar el ciclo al adquirir una nueva matriz de profundidad.

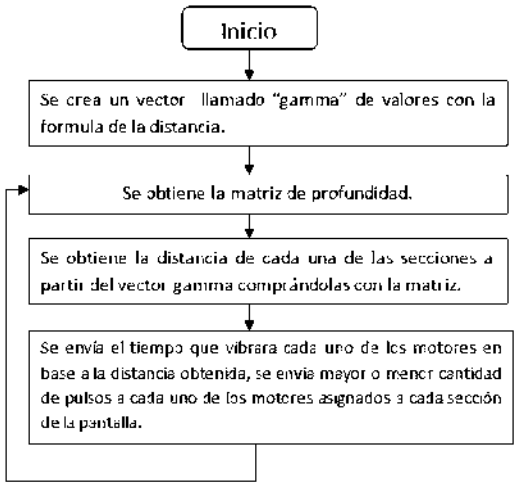


Figura 3.7. Diagrama de flujo general.

La forma de indicar la distancia de los objetos es por pulsaciones y el tiempo que va a permanecer apagado un motor. Mientras más corta sea la distancia el motor estará apagado menos tiempo y mientras más alejado se encuentre éste se mantendrá apagado más tiempo.

El rango de distancia útil (Figura 3.8) es a partir de 1 metro, ya que el usuario irá con un bastón cuyo rango de medición será de 1 metro aproximadamente, antes de ése valor el motor estará apagado.

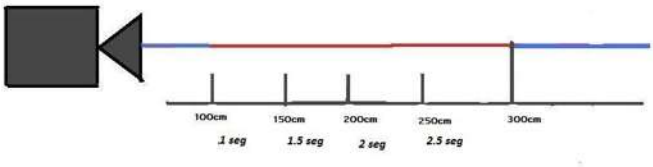


Figura 3.8. Diagrama de distancia y tiempo de apagado.

El valor máximo de distancia que se usará es 3 metros después de esa distancia el motor estará apagado, ya que como se escribió anteriormente (sección 3.2), esta distancia se encuentra en el rango óptimo donde el sensor puede estimar más precisamente la profundidad de los objetos. En la Tabla (3.3) se muestra los 4 intervalos de tiempo utilizados para determinar el tiempo de apagado de cada motor.

Distancia(m).	Tiempo Apagado(s).
1 a 1.5	1
1.5 a 2	1.5
2 a 2.5	2
2.5 a 3	2.5
Menor que 1 o mayor que 3	Se apaga el motor.

Tabla 3.3: Tabla de intervalos de distancia y tiempo de apagado.

Se mantiene encendido el motor un tiempo (t_{on}), dependiendo de la distancia de los objetos, éste se apagará en un lapso de tiempo (t_{off}), como se puede observar en la Figura 3.9. Los rangos de apagado son más grandes cuando el objeto se encuentra alejado y al acercarse se reducen.

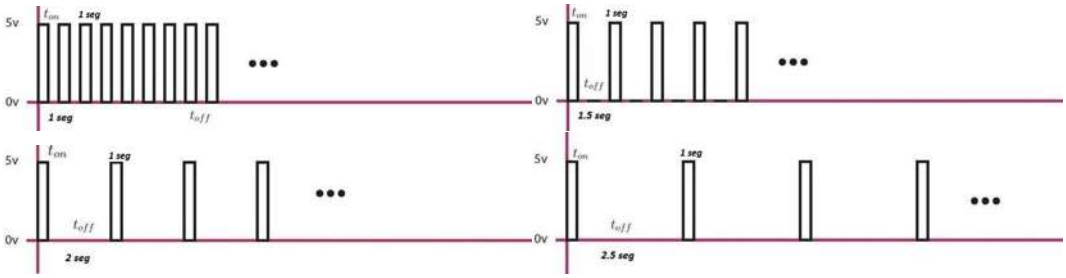


Figura 3.9. Rango de tiempo de apagado de los motores.

El tiempo de encendido y apagado de cada motor es determinado, como lo indica la Figura 3.10, en cada instante se determina si hay un objeto en el rango de visión si lo hay, se determina el tiempo de apagado apropiado como se indica en la Tabla(3.3) y se pone a vibrar el motor. Si no hay un objeto dentro del rango el motor se apaga (ver Tabla 3.3).

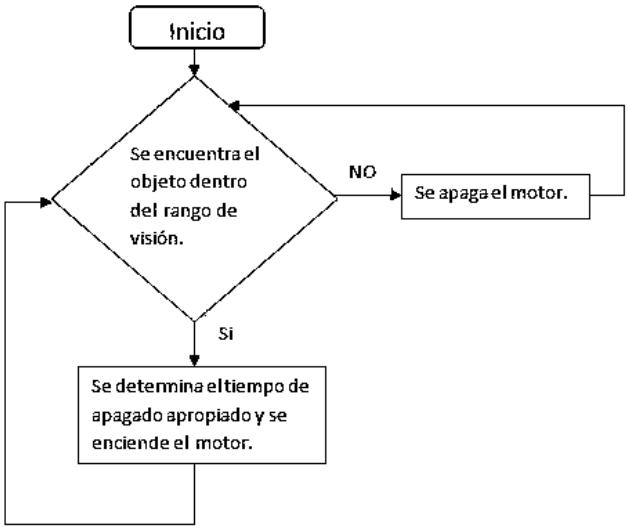


Figura 3.10. Diagrama de flujo de las condicionales para controlar las pulsaciones.

3.4. Conclusiones

En este capítulo se analizaron las funciones de profundidad del Kinect y como se aplicarán para obtener la distancia y la ubicación del objeto más próximo, así como las partes más importantes del programa para que éste pudiera interactuar con todos los componentes mencionados en capítulos previos.

En el próximo capítulo se muestran las pruebas a las que se sometió este sensor, así como las correcciones hechas para obtener una funcionalidad más efectiva.

Capítulo 4

Pruebas

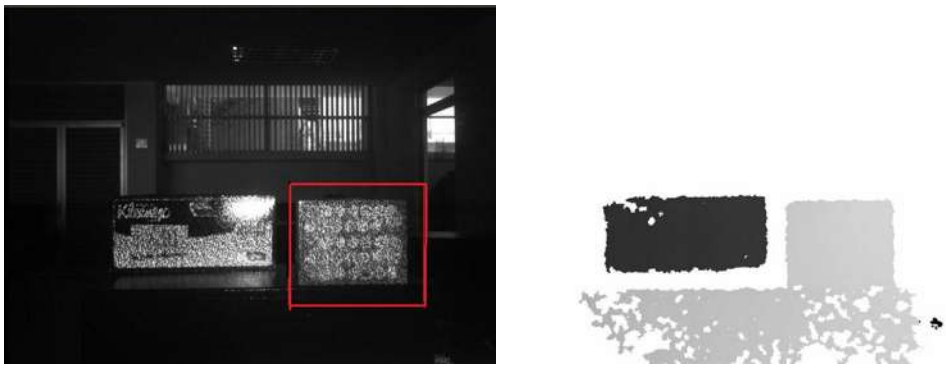
4.1. Pruebas Previas

La primera prueba a la que se sometió el dispositivo fue la detección de profundidad, ésto con el fin de comprobar si la distancia medida fuera igual a la real. Se utilizó un rango de valores de cada uno de los níveles de profundidad, mediante la fórmula (3.1) se obtienen los valores de cada uno de los niveles, es decir, las diferentes distancias. Se realizaron pruebas con objetos a diferentes distancias y con ello se comprobó tanto la eficacia del sensor como de la fórmula (3.1).

Usando dos objetos en el centro de visión del Kinect (Figura 4.1), se midió la distancia real de los objetos al sensor (Figura 4.2) la distancia más cercana es de 52cm, el objeto más alejado está a 63cm. Al hacer el programa de prueba, fue necesario conocer la distancia del objeto más próximo, por lo que se obtuvó la matriz de profundidad a partir de la imagen de profundidad (Figura 4.3). Se hizo un barrido de datos para obtener la distancia mínima y se comparó con la distancia real (Figura 4.4), finalmente se obtuvó una distancia medida de 52cm, el resultado esperado. Ésta prueba para tomar la distancia más cercana fue una de varias pruebas que se realizaron para comprobar las distancias reales.

Figura 4.1. Imagen de profundidad con gama de colores e imagen RGB.

Figura 4.2. La diferencia de distancia de un objeto a otro y la distancia al objeto más próximo.



(a) **(b)**

Figura 4.3. La imagen de infrarrojo y la imagen de profundidad.

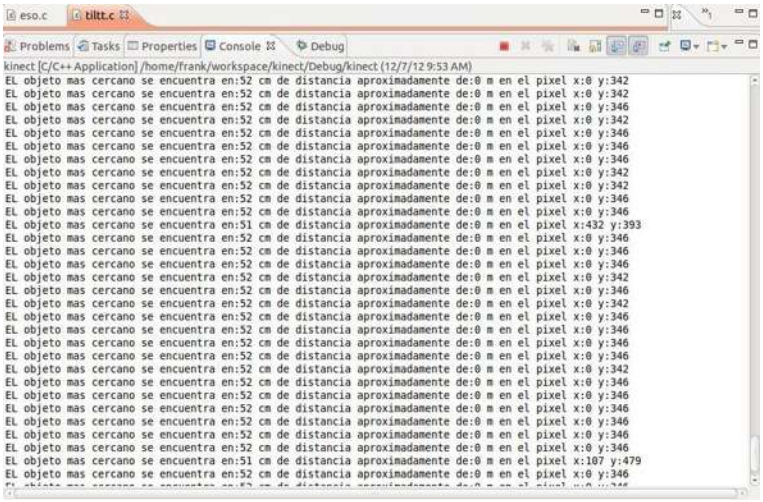


Figura 4.4. La distancia del objeto más cercano obtenido por el sensor Kinect en su ángulo de visión(El objeto del lado derecho de la Figura 4.3(a)).

La segunda prueba fue el funcionamiento del código para la función de activar y desactivar los pines. Ésta prueba se realizó primeramente activar un pin y posteriormente con múltiples pines, utilizando en un comienzo LEDs (diodo emisor de luz, por sus siglas en inglés) (Figura 4.5). Finalmente se realizaron pruebas de encendido y apagado de motores.

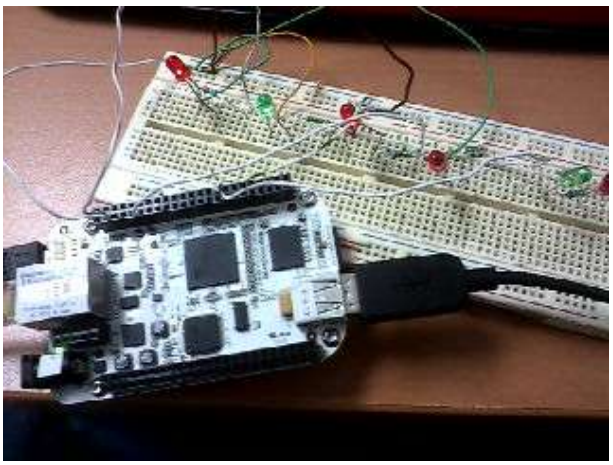


Figura 4.5. Prueba de habilitación correcta de pines mediante encendido de LED's.

Se determino que el circuito óptimo para que el motor trabajara sin riesgos consistía de una resistencia fija de 10Ω y un transistor NPN BC338(Figura 4.6).

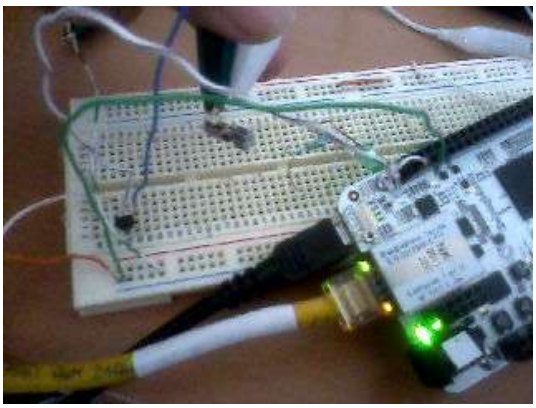
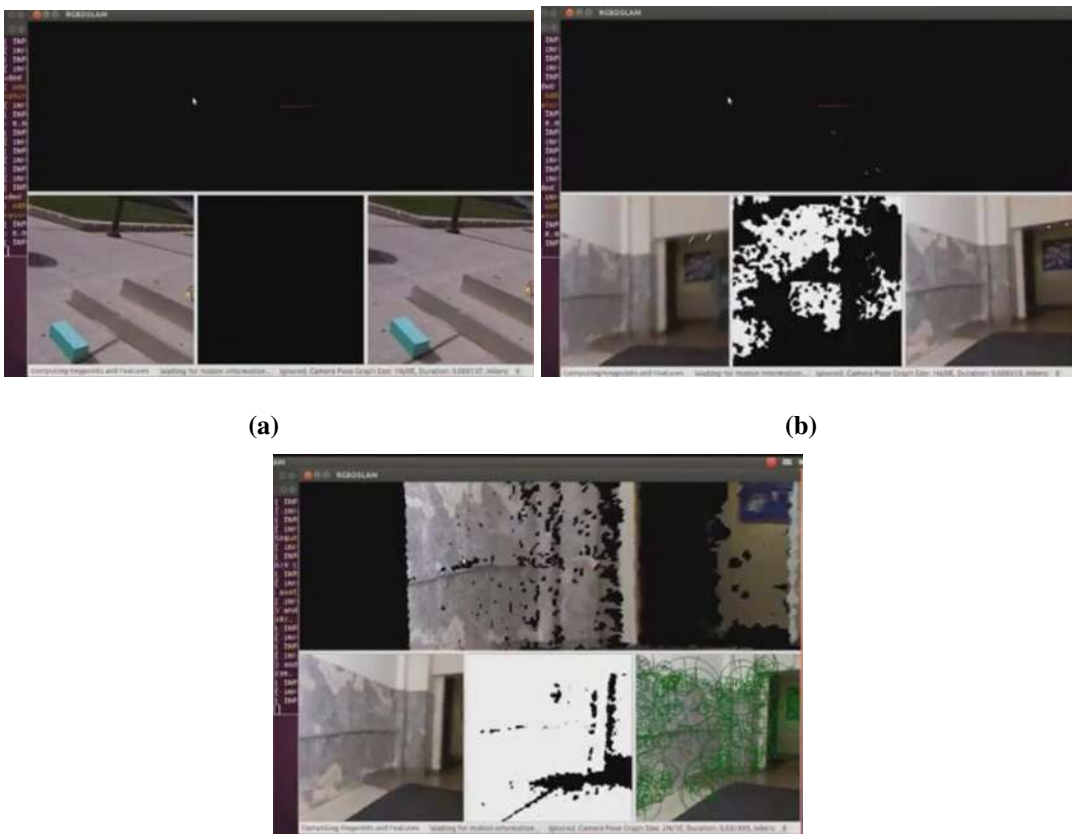


Figura 4.6. Circuito final para el motor.

4.2. Prueba de Saturación del Sensor Kinect

El comportamiento del sensor Kinect en un ambiente donde hay luz solar, da como resultado que la imagen de profundidad se sature, debido a la cantidad de radiación infrarroja en el ambiente. Los rayos infrarrojos del sol interfieren con los el sensor Kinect.

Como se observa en la Figura 4.7(a) en las imágenes RGB (lados) y la imagen de profundidad (centro). Dado que la cámara infrarroja del sensor capta toda la radiación infrarroja del la luz solar, se produce un efecto de saturación, lo que provoca que el sensor entregue información que se traduce a una imagen totalmente negra. Al momento de ingresar a un ambiente interior (menor radiación infrarroja), el sensor comienza a trabajar de forma normal como se observa en la Figura 4.7(c).



(a) **(b)**

(c)

Figura 4.7. Prueba de saturación de la imagen de profundidad.

4.3. Versión Final Del Dispositivo

El dispositivo se adaptó al cuerpo de la persona de tal manera que ésta pudiera caminar sin problemas y no obstruyera la comunicación entre los componentes.

Se decidió que el sensor Kinect estuviera encima de la cabeza para facilitar su movimiento, por ello se optó por utilizar un casco (Figura 4.8) para montar el sensor. Como se puede observar, el sensor queda en la parte alta del casco y el ángulo de inclinación es de 25º hacia abajo aproximadamente, para que se puedan captar los objetos al frente de la persona. Se utilizó una cinta de Velcro®para ajustar el casco y estabilizarlo al ser utilizado.

Figura 4.8. Casco y el sensor Kinect.

Al inclinar el tilt mecánico (mecanismo de inclinación) 25º, se pueden detectar los objetos delante del usuario (Figura 4.9). En la imagen que toma el sensor, la distancia mínima para detectar profundidades se encuentra a 1.60 metros. El bastón de ayuda a débiles visuales comercial tiene una longitud de 1.20 metros, por lo que al usarlo se pueden tocar objetos a una distancia horizontal de 90 centímetros, esto se obtuvo midiendo la distancia del primer objeto, tomado en la imagen capturada por el sensor en el ángulo de inclinación de 25º.

Figura 4.9. Visión a 90º y visión a 25º.

Los motores se soldaron al circuito de control, para que pudiera ser más fácil colocarlos en el dispositivo (Figura 4.10).

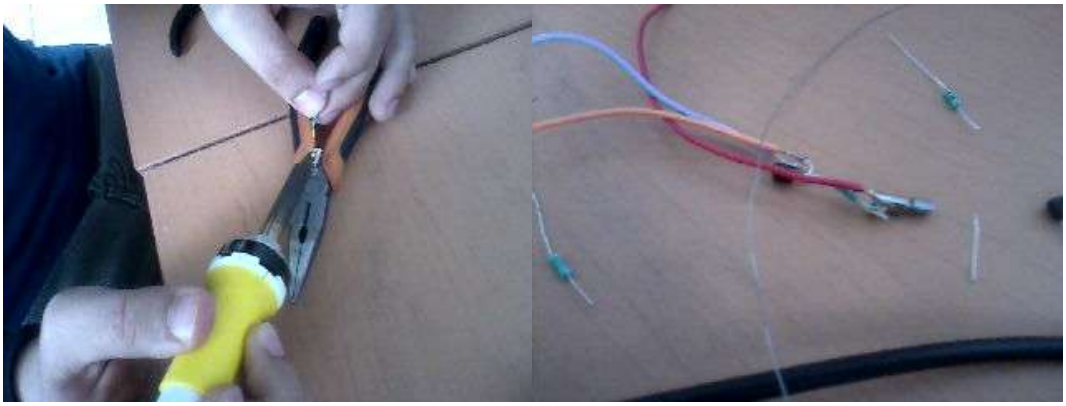


Figura 4.10. Motores soldados con el circuito.

Después de haber sido soldados al circuito, fue necesario cubrir la cabeza vibrante del motor para que no se atascara con la ropa del usuario al estar en operación. Para ello se recubrió cada uno con tubos de plástico y se les pusó pegamento para que quedaran fijos (Figura 4.11).

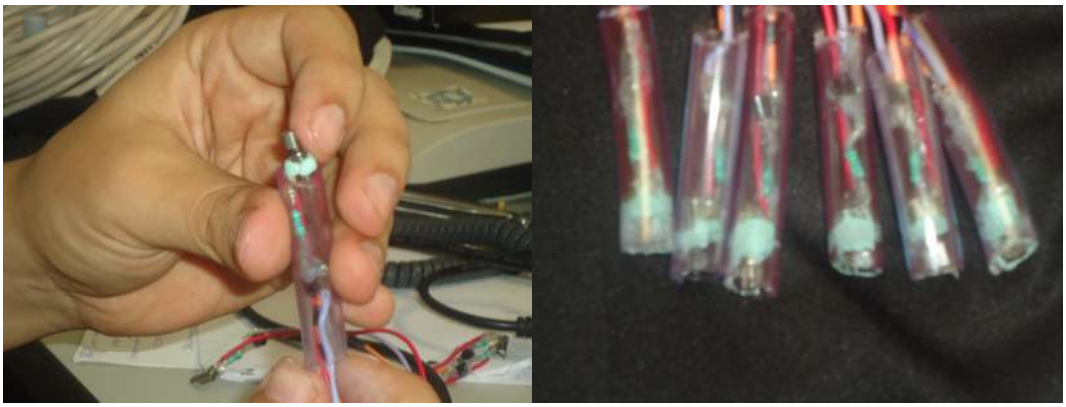


Figura 4.11. Motores en tubos de plastico.

Para el funcionamiento del dispositivo fue necesaria una fuente de alimentación de 5 volts para los motores y el Beaglebone, por lo que se fijó a una placa un adaptador de

corriente de un disco duro externo (Figura 4.12).

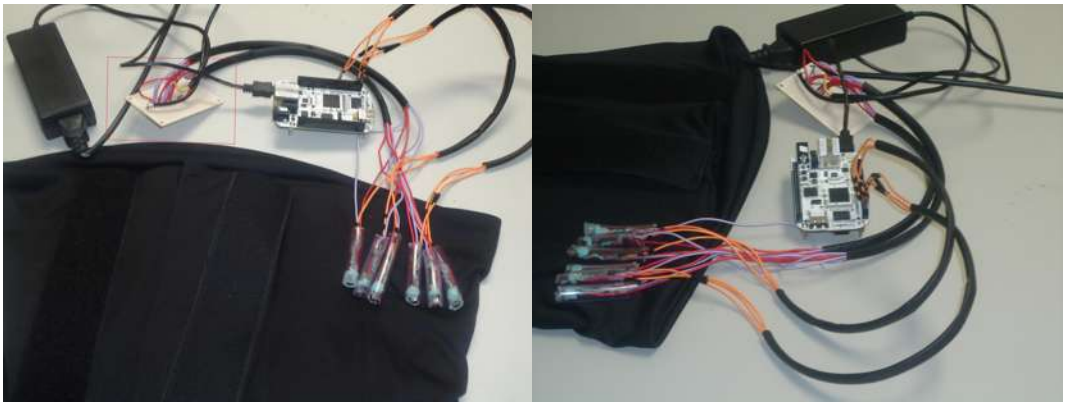


Figura 4.12. Faja, motores y adaptador de corriente.

Se realizaron diversas pruebas de campo donde se somete al dispositivo en toda su funcionalidad para probar su efectividad, en interiores y en exteriores, de día, noche y tarde en lugares comunes.

4.4. Pruebas de Campo

La primera de las pruebas se hizo en una casa, como el dispositivo tiene que estar conectado a un tomacorriente fue necesario utilizar una mochila para poder llevar un multicontacto para que fuera más fácil conectar todos los componentes. Se optó por una mochila escolar para que ésta no estorbara el movimiento del usuario. Los componentes más importantes fueron montados en la faja que son: los motores, el BeagleBone protegida con una caja, un adaptador de 5 volts para alimentar el BeagleBone y los motores. Se provee al sujeto de pruebas de un bastón para invidentes comercial, como se puede ver en la Figura 4.13.

Figura 4.13. Mochila utilizada con el tomacorriente, la faja con los componentes necesarios y el bastón comercial.

Al caminar por los pasillos y habitaciones de la casa (Figura 4.14), se observó que determinaba de manera correcta la distancia de los objetos dentro del rango de visión. Los motores se encienden si el sensor detecta objetos en la zona correspondiente a cada motor (ver Figura 3.3).

Figura 4.14. Prueba previa con el dispositivo terminado.

4.5. Casos de Estudio

Caso de prueba 1 en interior: la prueba se realizo en un laboratorio de cómputo, el cual tiene un tamaño aproximado de 5x6 metros, donde el sujeto de prueba estuvó en movimiento por el área. En base a los comentarios del sujeto de prueba, se comprobó que el dispositivo detectaba los objetos a su alrededor, cuando estaban dentro del rango de distancia establecido, en el caso de la Figura 4.15(c) se encendieron los motores del lado izquierdo del dispositivo. También se verificó que cuando se encuentra un objeto a menos de un metro de distancia como en el caso de la Figura 4.15(b) los motores se detuvieron tal como se esperaba. De esta prueba se concluyó que el dispositivo es capaz de detectar los objetos en el rango de visión del sensor que es de 0.4 metros a 3 metros, sin embargo no detecta adecuadamente los objetos que están a poca altura del piso, a unos 30cm o menos aproximadamente. El sujeto sugirió separar un poco más los motores en la faja para que se perciban mejor los distintos motores.

(a) **(b)** **(c)**

Figura 4.15. Prueba en un laboratorio de cómputo.

Caso de prueba 2 en interior: la prueba se realizó en una casa, el sujeto de prueba explor el área y en base a sus comentarios, se comprobó que la detección de los objetos, cuando este llegaba a detectar un objeto que se encuentraba a uana distancia menor de un metro como en la Figura 4.16(b) los sensores apagaron todos los motores, tal y como se esperaba. De esta prueba se concluyo que los motores, respondieron eficientemente a diferentes distancias obtenidas de los objetos.

(a) **(b)** **(c)**

(d)

Figura 4.16. Prueba en una casa.

Caso de prueba 3 en exterior: la prueba se realizó en la entrada de un edificio de un canal de televisión, el sujeto de prueba estuvo utilizando el dispositivo en el exterior y

luego se cambio al interior. Como se pude ver en la Figura 4.17(b) subió escaleras y esto causo un conflicto ya que el ángulo de visión del sensor no llega a detectar, objetos muy cercanos al piso. Mientras se encontraba en el exterior el dispositivo se saturaba y por no se comportaba de la forma esperada.. Una vez en el interior (ver Figura 4.17(c)) el sujeto de prueba estuvó en movimiento por el área mostrando resultados similares a los casos 1 y 2. Con este caso de estudio se muestra que el dispositivo no opera de forma óptima en exteriores, debido al problema de saturación que se explicó en la sección 4.2.

(a) **(b)** **(c)**

Figura 4.17. Prueba en un edificio.

4.6. Conclusiones

Como se pudo observar en este capítulo, las pruebas en las que fueron sometidos los componentes individualmente para corregir los errores y al último en conjunto, en un día normal, en interior y en exterior. Se observó la reacción de los usuarios y el dispositivo en los diferentes ambientes en los que está acostumbrada una persona.

En el próximo capítulo se hablará de las conclusiones de estos experimentos y el funcionamiento del dispositivo, así como las mejorías y trabajos futuros que se pueden realizar.

Capítulo 5

Conclusiones

Después de varios experimentos realizados en los diferentes ambientes, se vio que el dispositivo indica dónde se encuentran los objetos y a que distancia. Debido a que el sensor fue diseñado para interiores, no funciona en exteriores ya que se satura de radiación infrarroja. Este dispositivo es de apoyo ya que es necesario llevar un bastón porque el sensor solo toma los datos en un rango de visión limitado. Esto porque se colocó sobre la cabeza y con una inclinación, el sensor detecta solamente los objetos a una distancia mayor de 1.60 metros (para una persona de estatura de 1.70 m). La distancia mínima de detección de distancia del dispositivo cambiará dependiendo de la estatura del usuario.

Este proyecto se enfocó a utilizar componentes de bajo costo para solucionar un problema que aqueja a sectores vulnerables de la población. Kinect es un dispositivo innovador ya que es uno de los primero escáneres 3D comerciales. La idea de utilizar luz estructurada es muy efectiva ya que permite conocer la distancia de los objetos al sensor en poco tiempo. BeagleBone en comparación con una computadora "estándar" es muy efectiva en este tipo de proyectos, ya que ofrece las bondades de la computadora, en un tamaño reducido y sus diferentes periféricos. Otra ventaja es que envía o recibe datos a través de sus pines, cosa que es muy útil para proyectos de investigación donde se combina electrónia y computación. Por otro lado, no puede procesar y almacenar muchos datos como una computadora debido a sus limitantes físicas.

En este trabajo se realizaron pruebas de uso del dispositivo por un tiempo pequeño. Es probable que después de un entrenamiento con el aparato, el usuario identifique de mejor manera tanto la distancia como la zona en que se encuentran los obstáculos, debido a la plasticidad del cerebro.

5.1. Trabajos Futuros

Este dispositivo se puede mejorar tanto en el procesamiento de datos, como en la forma en que se indica la profundidad y el tamaño. Kinect fue el pionero en usar un escáner 3D aplicado a un propósito comercial, sin embargo la teconología de sensores sigue evolucionando. Actualmente la compañía ASUS®lanzó Xtion®(Figura 5.1) que es una sensor similar al Kinect, pero para controlar las acciones de una PC. Xtion es más compacto, y no necesita alimentación eléctrica externa, sólo la recibida por el puerto USB, la cámara RGB tiene una resolución de 1280×1024 pixeles y la cámara de profundidad 640×480 píxeles, sin embargo, su precio es mayor al del Kinect. Se debe investigar si los motores utilizados en el dispositivo pueden ser útiles a esta velocidad de refresco o se debe cambiar a otro método de comunicación con el cuerpo humano.

Figura 5.1. Xtion y su diferencia en tamaño con Kinect.

Otra mejoría es el cambio de la barebone BeagleBone, por su nueva presentación Bea-

gleBone Black®(Figura 5.2), la nueva BeagleBone a comparación con la utilizada en el proyecto, da mayor velocidad en el procesamiento ya que aumentó la frecuencia del procesador a 1GHz y la memoria RAM a 512MB. Una mejora significativa es el hecho de tener 2GB de ROM interno. Este contiene los mismos periféricos de entrada y salida del BeagleBone tanto el numero de pines, la misma arquitectura del procesador y consume poca potencia eléctrica. El costo de esta versión es menor, lo que resulta más conveniente.

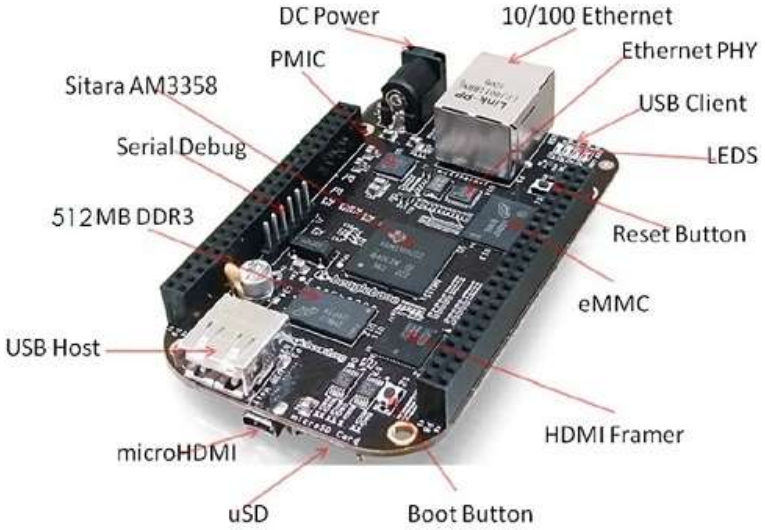


Figura 5.2. Componentes del BeagleBone Black.

Una variante para la entrada del dispositivo puede ser utilizar un sensor que calcule la profundidad mediante luz en el rango de visibilidad humana. Se propone la utilización de otras técnicas de visión computacional en lugar de luz estructurada como son, la profundidad por desenfoque o la profundidad por estereo[Schechner00]. Existe también la posibilidad de agregar sensores mediante ultrasonido. Ambas sugerencias ayudarían para que se pueda utilizar el dispositivo tanto en interiores como en exteriores.

En las pruebas se hicieron varias sugerencias como: la utilización de una caja de baterías para alimentar el dispositivo, cambiar la posición del sensor en la parte central de la persona, para que este tomara la perspectiva céntrica y se pudiera ver los objetos a la altura

de la rodilla y un poco más alto de la barbilla. Sin embargo, la idea de utilizar el sensor en la cabeza tiene su origen en la mayor movilidad de la misma, en comparación con el torso.

El trabajo tiene más vertientes y es posible realizar mejorías, pero para ello se necesita mayor tiempo de investigación y hacer pruebas utilizando diversos componentes. Lamentablemente muchos de estos componentes son costosos y, en algunos casos, son difíciles de conseguir.

Referencias

[Amedi03] Amedi, A., Raz, N., Pianka, P., Malach, R., y Zohary, E. Early 'visual' cortex activation correlates with superior verbal memory performance in the blind. *Nature Neuroscience*, 6:758–766, 2003. Investigación acerca de la agudeza de los sentidos en las personas invidentes.

[BeagleBone] BeagleBone. Beaglebone system reference manual rev a6. Manual de referencia de BeagleBone.
URL `http://beagleboard.org/static/beaglebone/latest/Docs/Hardware/BONE_SRM.pdf`

[ijuanjojose] ijuanjojose. Kinect en linux primeros pasos - instalación - libfreenect. Tutorial donde se muestran los pasos necesarios para poder conectar el Kinect al PC y ver algunos demos en Linux.
URL `http://www.youtube.com/watch?v=LgY2zf5qKnA`

[Khoshelham] Khoshelham, K. Accuracy analysis of kinect depth data. Investigación acerca de la calidad geométrica de los datos de profundidad de Kinect.
URL `https://wiki.rit.edu/download/attachments/52806003/ls2011_submission_40.pdf`

[Ladislao Mathe] Ladislao Mathe, D. S. y Gomez., G. Estudio del funcionamiento del sensor kinect y aplicaciones para bioingeniería. Proyecto de brazo robótico para laparoscopia con Kinect.

URL http://www.computacion.efn.uncor.edu/sites/default/files/123.pdf

[Manduchi12] Manduchi, R. y Coughlan, J. (computer) vision without sight. *Commun. ACM*, 55(1):96–104, ene. 2012. ISSN 0001-0782. doi: 10.1145/2063176.2063200. Articulo acerca de las nuevas tecnologías e investigaciones implementadas para el apoyo visual humano.
URL http://doi.acm.org/10.1145/2063176.2063200

[Molloy] Molloy, D. Beaglebone: Gpio programming on arm embedded linux. Introducción al BeagleBone y desarrollo de aplicaciones con Linux.
URL http://www.youtube.com/watch?v=SaIpz00lE84

[Nagai] Nagai, A. Como programar la beaglebone con c/c++. Página japonesa donde presentan proyectos utilizando el BeagleBone.
URL http://www.cs.gunma-u.ac.jp/ nagai/wiki/index.php?BeagleBone%20%A4%CE%BB%C8%A4%A4%CA%FD

[OpenKinect] OpenKinect. Openkinect. Comunidad libre de personas interesadas en dar uso a la cámara Kinect.
URL http://openkinect.org/wiki/Main_Page

[Richardson] Richardson, M. How-to: Get started with the beaglebone. Muestra cómo empezar a trabajar en el BeagleBone haciendo parpadear un LED.
URL http://www.youtube.com/watch?v=Y0uqRVxismQ

[Robotics] Robotics, M. Kinect sensor. Especificaciones proporcionadas por los fabricantes de Kinect.
URL http://msdn.microsoft.com/en-us/library/hh438998.aspx

[Schechner00] Schechner, Y. Y. y Kiryati, N. Depth from defocus vs. stereo: How different really are they? *Int. J. Comput. Vision*, 39(2):141–162, sep. 2000. ISSN 0920-5691. doi:10.1023/A:1008175127327. Muestra la diferencia y similitudes entre dos técnicas para tomar la profundidad la de desenfoque y la estéreo.
URL `http://dx.doi.org/10.1023/A:1008175127327`

Glosario

Acceso Remoto Programas que permiten acceder y ejecutar algunas acciones desde un equipo local a un equipo remoto.

Arquitectura ó ISA(Instruction Set Architecture) Son las instrucciones que una CPU, puede entender y ejecutar.

Barebone Computadora semi ensamblada reducida en dimensiones fisicas que viene con placa y RAM específica.

Controlador Programa que permite al sistema operativo comunicarse con un dispositivo de hardware para que éste pueda ser controlado.

GPIO(General Purpose Input/Output) Es un tipo genérico de pin que se encuentra en un procesador y puede ser programado durante la ejecución, se utiliza para tanto la entrada como salida de datos.

Librerias Conjunto de funciones, utilizadas para desarrollar programas.

Pin Patilla metálica de un conector, se utiliza para transferir datos o energía sin la necesidad de soldarlo.

Visión Computacional Sub campo de la inteligencia artificial que busca entender las características de una imagen.

Anexos

Código Fuente

Código fuente escrito en C llamado profundidad.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>
#include <assert.h>
#include <pthread.h>
#include <libfreenect.h>
#include <libfreenect_sync.h>

////Prototipos de las funciones.////
int ExportarPin(char [],FILE *,char []);
int EncenderPin(char [],FILE *);
int ApagarPin(char [],FILE *);
int InexportarPin(char [],FILE *);
int Intencidad(int [],int [],int [],int [],double [],double [],
   int []);
void Calculo_de_Intencidad(int [],int []);
```

```
18 void no_kinect_quit(void);
19 int Distancia_mas_Cerca(int,int,int,int []);
20
21 ////Exporta el pin.////
22 int ExportarPin(char GPIODirection[],FILE *LEDHandle,char
      GPIOString[]){
23 char setValue[4];
24 if((LEDHandle = fopen("/sys/class/gpio/export", "ab")) == NULL){
25     printf("No se puede exportar GPIO pin.\n");
26     return 1;
27     }
28     strcpy(setValue, GPIOString);
29     fwrite(&setValue, sizeof(char), 2, LEDHandle);
30     fclose(LEDHandle);
31
32 //Establecer la dirección del pin
33 if((LEDHandle = fopen(GPIODirection, "rb+")) == NULL){
34     printf("No se puede abrir la dirección handle.\n");
35     return 1;
36     }
37     strcpy(setValue,"out");
38     fwrite(&setValue, sizeof(char), 3, LEDHandle);
39     fclose(LEDHandle);
40     return 0;
41 }
42
43 ////Encender el pin.////
44 int EncenderPin(char GPIOValue[],FILE *LEDHandle){
45 char setValue[4];
```

```
46 if((LEDHandle = fopen(GPIOValue, "rb+")) == NULL){
47     printf("No se puede abrir la dirección handle.\n");
48     return 1;
49     }
50     strcpy(setValue,"1");
51     fwrite(&setValue, sizeof(char), 1, LEDHandle);
52     fclose(LEDHandle);
53     return 0;
54 }
55
56 ////Apagar el pin.////
57 int ApagarPin(char GPIOValue[],FILE *LEDHandle){
58 char setValue[4];
59 if((LEDHandle = fopen(GPIOValue, "rb+")) == NULL){
60     printf("No se puede abrir la dirección handle.\n");
61     return 1;
62     }
63     strcpy(setValue,"0");
64     fwrite(&setValue, sizeof(char), 1, LEDHandle);
65     fclose(LEDHandle);
66     return 0;
67 }
68 ////Inexportar el pin.////
69 int InexportarPin(char GPIOString[],FILE *LEDHandle){
70 char setValue[4];
71 if((LEDHandle = fopen("/sys/class/gpio/unexport", "ab")) == NULL)
    {
72     printf("No se puede exportar GPIO pin.\n");
73     return 1;
```

```
74      }
75      strcpy(setValue, GPIOString);
76      fwrite(&setValue, sizeof(char), 2, LEDHandle);
77      fclose(LEDHandle);
78      return 0;
79  }
80
81  ////Función de intensidad del motor.////
82  int Intencidad(int area[],int estado[],int areadistancia[],int
       distanciaanterior[],double tiempoarea[],double tiempoapagado
       [],int tiempo[]){
83  double diferenciatiempo;
84  int GPIOPin;
85  int h;
86  int Area;
87  clock_t Tiempoinicial;
88  FILE *LEDHandle = NULL;
89  char setValue[4], GPIOString[4], GPIOValue[64], GPIODirection
       [64];
90  for(h=0;h<6;h++){
91       GPIOPin=area[h];
92       Area=areadistancia[h];
93       Tiempoinicial=tiempoapagado[h];
94       sprintf(GPIOString, "%d", GPIOPin);
95       sprintf(GPIOValue, "/sys/class/gpio/gpio%d/value", GPIOPin);
96       sprintf(GPIODirection, "/sys/class/gpio/gpio%d/direction",
            GPIOPin);
97  ///Vemos el estado.
98       if(estado[h]==0 && areadistancia[h]!=4 && tiempo[h]==0){
```

```
99          ExportarPin(GPIODirection,LEDHandle,GPIOString);
100         EncenderPin(GPIOValue,LEDHandle);
101         estado[h]=1;
102         distanciaanterior[h]=areadistancia[h];
103      }
104      if(estado[h]==1 && areadistancia[h]==4){
105         ExportarPin(GPIODirection,LEDHandle,GPIOString);
106         ApagarPin(GPIOValue,LEDHandle);
107         InexportarPin(GPIOString,LEDHandle);
108         estado[h]=0;
109         tiempo[h]=0;
110         distanciaanterior[h]=areadistancia[h];
111      }
112      if(estado[h]==1 && areadistancia[h]!=4){
113         ExportarPin(GPIODirection,LEDHandle,GPIOString);
114         ApagarPin(GPIOValue,LEDHandle);
115         InexportarPin(GPIOString,LEDHandle);
116         tiempoapagado[h]=(double)clock();
117         estado[h]=0;
118         tiempo[h]=1;
119         distanciaanterior[h]=areadistancia[h];
120      }
121       if(estado[h]==0 && areadistancia[h]<distanciaanterior[h] &&
              areadistancia[h]!=4){
122         ExportarPin(GPIODirection,LEDHandle,GPIOString);
123         EncenderPin(GPIOValue,LEDHandle);
124         estado[h]=1;
125         tiempo[h]=0;
126         distanciaanterior[h]=areadistancia[h];
```

```
127       }
128        if(estado[h]==0 && tiempo[h]==1){
129           diferenciatiempo=(double)(clock()-Tiempoinicial)/
                 CLOCKS_PER_SEC;
130          if(diferenciatiempo>=tiempoarea[Area]){
131             ExportarPin(GPIODirection,LEDHandle,GPIOString);
132             EncenderPin(GPIOValue,LEDHandle);
133             estado[h]=1;
134             tiempo[h]=0;
135             tiempoapagado[h]=0.0;
136             distanciaanterior[h]=areadistancia[h];
137           }
138       }
139    }
140 }
141 ///Pondremos condicionales para cada uno de las seis áreas
       necesarias.
142 ///Primera área de la imagen para diferentes distancias.
143 void Calculo_de_Intensidad(int distancia[],int areadistancia[]){
144 int h,Distancia;
145   for(h=0;h<6;h++){
146        Distancia=distancia[h];
147           if(Distancia>=100 && Distancia<150){areadistancia[h]=0;
148     }else if(Distancia>=150 && Distancia<200){areadistancia[h]=1;
149     }else if(Distancia>=200 && Distancia<250){areadistancia[h]=2;
150     }else if(Distancia>=250 && Distancia<=300){areadistancia[h]=3;
151     }else if(Distancia>300 && Distancia<=400){areadistancia[h]=4;
152     }else if(Distancia>=0 && Distancia<100){areadistancia[h]=4;
153    }
```

```
154   }
155  }
156  ///Función que devuelve un mensaje cuando el sensor kinect
157  ///no se encuentra conectado.
158  void no_kinect_quit(void){
159          fprintf(stderr, "Error: Kinect no conectado?\n");
160          exit(1);
161  }
162  ///Función que permite colocar la distancia más cerca
163  ///en la sección correspondiente.
164  int Distancia_mas_Cerca(int x,int y,int d,int distancia[]){
165   if(x<=213 && y<=240){
166           distancia[0]=d;
167           return 1;
168   }
169   if(x<=426 && y<=240 && x>213){
170           distancia[1]=d;
171           return 1;
172   }
173   if(x<=640 && y<=240 && x>426){
174           distancia[2]=d;
175           return 1;
176   }
177   if(x<=213 && y<=480 && y>240){
178           distancia[3]=d;
179           return 1;
180   }
181   if(x<=426 && y<=480 && y>240 && x>213){
182           distancia[4]=d;
```

```
183             return 1;
184      }
185      if(x<=640 && y<=480 && y>240 && x>426){
186             distancia[5]=d;
187             return 1;
188      }
189     }
190
191     int main(void)
192     {
193         short *depth = 0;
194         int d,x,y,i,h;
195         int valor;
196         int area[6];
197         int areadistancia[6];
198         int distanciaanterior[6];
199         int distancia[6];
200         int estado[6];
201         int tiempo[6];
202         double tiempoarea[6];
203         double tiempoapagado[6];
204         uint32_t ts;
205         int ret;
206
207     ///Profundidad del pixel.
208           int t_gamma[2048];
209       for(i=0;i<2048;i++){
210           float k1 = 1.1863;
211           float k2 = 2842.5;
```

```
212         float k3 = 0.1236;
213         float profundidad =100*(k3*tan(i/k2+k1));///El valor esta
                en metros lo convertimos a cm.
214         t_gamma[i] =(int)profundidad;///La gamma la ponemos en cm
                enteros.
215       }
216
217 for(h=0;h<6;h++){
218   area[h]=0;///Indica el valor de cada sección en la imagen
          contiene.
219   tiempo[h]=0;///Nos dice si el valor tiene un tiempo por la
          distancia o no.
220   estado[h]=0;///Indica si esta encendido o apagado.
221   distancia[h]=0;///Es la distancia a la que se encuentra el
          objeto captado.
222   tiempoarea[h]=0.0;///Es el tiempo de acuerdo al área en el que
          se encuentra.
223   areadistancia[h]=0;///Valor de la sección de la distancia.
224   distanciaanterior[h]=0;///Valor anterior de la sección de la
          distancia.
225   tiempoapagado[h]=0.0;///Valor del tiempo de apagado de cada
          sección.
226 }
227 ///Establecemos cada una de las secciones de la pantalla.
228 area[0]=87;
229 area[1]=89;
230 area[2]=72;
231 area[3]=73;
232 area[4]=70;
```

```
233 area[5]=71;
234
235 ///Establecemos cada uno de los tiempos a la distancia.
236 tiempoarea[0]=1.0;
237 tiempoarea[1]=1.5;
238 tiempoarea[2]=2.0;
239 tiempoarea[3]=2.5;
240
241 inicio:
242 ///Tomamos la matriz de profundidad con la variable "depth".
243     ret = freenect_sync_get_depth((void**)&depth, &ts, 0,
        FREENECT_DEPTH_11BIT);
244           if (ret < 0)
245                 no_kinect_quit();
246
247  for(h=0;h<640;h++){///Anchura
248           for(i=0;i<480;i++){///Altura
249
250 ///Detectamos el punto más cerca de distancia hacia un objeto.
251    if((int)depth[i*480+h]!=2047){
252     valor=(int)t_gamma[(int)depth[i*480+h]];
253 ///Distancia en el centro del pixel.
254       if(h==107 && i==120){distancia[0]=valor;
255 }else if(h==319 && i==120){distancia[1]=valor;
256 }else if(h==533 && i==120){distancia[2]=valor;
257 }else if(h==107 && i==360){distancia[3]=valor;
258 }else if(h==319 && i==360){distancia[4]=valor;
259 }else if(h==533 && i==360){distancia[5]=valor;
260 }
```

```
261       if(d==0){
262         d=valor;
263         x=h;
264         y=i;
265       }
266       if(valor<d){
267         d=valor;
268         x=h;
269         y=i;
270         }
271       }
272     }
273   }
274   ///Llamamos a las funciones y de los datos obtenidos
        transformarlos a vibraciones.
275     Distancia_mas_Cerca(x,y,d,distancia);
276     Calculo_de_Intensidad(distancia,areadistancia);
277     Intencidad(area,estado,areadistancia,distanciaanterior,
          tiempoarea,tiempoapagado,tiempo);
278     d=0;
279     sleep(1);
280     goto inicio;
281   }
```

Compilación usando `gcc`:

```
gcc profundidad.c -o profundidad -lm -lfreenect_sync
```

Instalación de Freenect

La instalación se realizó en Ubuntu con los siguientes comandos.

Se agregaron los repositores de Freenect.

```
sudo add-apt-repository ppa:arne-alamut/freenect
```

Se actualizaron los repositorios.

```
sudo apt-get update
```

Se instalaron los repositores.

```
sudo apt-get install freenect
```

Se agregó el usuario al grupo[7] video, para accesar a los dispositivos de captura de video.

```
sudo adduser ubuntu video
```

Se accesó a la carpeta, para compilar se utilizó.

```
cmake ..
make
```

Se instalaron las librerías y se actualizaron las liberías utilizadas por el sistema.

```
sudo make install
sudo ldconfig /usr/local/lib/
```

Con esto quedaron instalados los demos y las librerías de Kinect, para poder así controlar el dispositivo.

[7] Un grupo en Linux/UNIX permite asignar permiso de accesos a archivos y dispositivos.