UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO INGENIERÍA ELÉCTRICA

SISTEMA INTERACTIVO POR VOZ PARA ROBOT PR2

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

SEBASTIÁN ADOLFO GUERRERO DÍAZ

PROFESOR GUÍA:
NÉSTOR BECERRA YOMA

MIEMBROS DE LA COMISIÓN:
JOSÉ NOVOA ILIC
CÉSAR AZURDIA MEZA

SANTIAGO DE CHILE
2015

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERIA CIVIL ELÉCTRICA
POR: SEBASTIÁN GUERRERO DÍAZ
FECHA: AGOSTO DE 2015
PROF. GUÍA: NÉSTOR BECERRA YOMA

SISTEMA INTERACTIVO POR VOZ PARA ROBOT PR2

En el marco de la línea de investigación: Interacción Humano Robot, el lenguaje es un candidato natural para la interacción con máquinas y robots. En particular las tecnologías de procesamiento y transmisión voz juegan un rol fundamental en lo que respecta al reconocimiento automático de la voz o *Automatic Speech Recognition* (ASR), ya que como concepto entrega la habilidad de escuchar al momento de interaccionar con un robot.

El objetivo principal de esta memoria es implementar una aplicación para comandar a un *Personal Robo*t 2 (PR2) a través de instrucciones por voz, utilizando arquitectura cliente-servidor mediante un micrófono como elemento para muestrear la señal de audio o comando enunciado por el usuario.

En el desarollo de la aplicación se crea una interfaz de interacción con el usuario que permite grabar audio usando HTML5/CSS/JS. Para el funcionamiento de la aplicación la comunicación con todos los servidores se realiza a través de *socket* TCP/IP y el procesamiento de la señal de audio se realiza utilizando KALDI, estado del arte en ASR. Luego de obtener el resultado de la transcripción de la elocución grabada por el usuario, es analizada por el *Robot PR2* y entrega la respuesta del comando de voz mediante el *Text to Speech* (TTS) Festival y ejecuta el movimiento correspondiente al requerimiento del usuario.

A partir esta investigación y el desarrollo de la aplicación se concluye que es posible comandar el *Robot PR2* de forma exitosa mediante la interfaz de interacción con el usuario. Dicha aplicación puede ser instalada en cualquier versión del sistema operativo Ubuntu. Utilizando un modo de conexión local para los servidores: ASR y WEB, se cumple el objetivo de realizar procesamiento en tiempo real. El sistema de reconocimiento de voz obtiene un 10 % de *Word Error Rate* (WER) cuando es probado con el mismo micrófono con el que se construye la base de datos, que corresponde a un conjunto de 4991 elocuciones de base de datos Latino-4k y 200 elocuciones grabadas en el Laboratorio de Procesamiento y Transmisión de Voz LPTV, todas a una frecuencia de 16 kHz y 16 bit.

El trabajo futuro considera la realización de una base de datos independiente del hablante y género. Considerar el efecto del canal en el ASR: ambiente y micrófonos. Debido a la forma de construcción de la aplicación, ésta es escalable ,por lo que es factible complementar el diccionario utilizado en el ASR y la personalización de los movimientos.

Dedicado a
mi familia

Agradecimientos

A mi Padre, Madre y Hermana que han sido fuente de apoyo contanste e incondicional desde los inicios de mi proceso educativo y más aún en éstos últimos años, ya que sin su ayuda esta meta no habría sido cumplida.

Al Profesor Néstor Becerra Yoma por permitirme trabajar en un proyecto de memoria único, innovador y desafiante. También agradecer los consejos y el apoyo a mis compañeros del Laboratorio de Procesamiento y Transmisión de Voz.

Al Profesor César Azurdia y José Novoa por formar parte del proyecto y asistirme en la construcción de este documento formal.

Tabla de contenido

Índice de tablas

Índice de ilustraciones

Capítulo 1

Introducción

1.1. Motivación

Sin duda en un mundo donde el desarrollo de la tecnología se sustenta en la base de asistir al hombre en sus quehaceres tanto cotidianos como exclusivos de una actividad particular, la creación de máquinas que realicen actividades específicas toma relevancia.

En particular, máquinas inteligentes que de alguna manera intentan emular el comportamiento humano al realizar una tarea. Siguiendo esta idea, aquí es dónde la robótica establece su nicho de investigación, en particular: brazos mecánicos articulados, robots humanoides, robots bípedos, por mencionar algunos ejemplos.

PR2 es un robot diseñado por Willow Garage, una compañía y laboratorio de investigación robótica, ubicada en la ciudad de Menlo Park en el estado de California en Estados Unidos de América, que se dedica a la creación de software y aplicaciones de código abierto con licencia BSD para distintos tipos de robots, entre ellos el *Robot PR2*.

El objetivo de la compañía al desarrollar el robot se enfoca en investigar aplicaciones que le permitan a las personas crear un ambiente mas productivo en el trabajo y en el hogar, ya que el *Robot PR2* tiene la movilidad de navegación (traslado), el tamaño promedio, la precisión y los grados de libertad de las articulaciones humanas. Como ya se ha mencionado, la ventaja de trabajar con una licencia de código abierto es que permite la cooperación internacional e multidisciplinaria con otros usuarios de el *Robot PR2*, además de la posibilidad de realizar cambios profundos a la programación en las distintas versiones de su sistema operativo *Robot Operating System* (ROS), en caso de ser necesario.

En líneas generales la construcción de el *Robot PR2*, es de material robusto, que permite la operación y experimentación dentro de condiciones de laboratorio. La aplicabilidad de la

investigación es amplia y abarca tareas tales como:

- Rescates en accidentes.
- Accesos remotos donde el organismo humano no es capaz de habitar.
- Tareas rutinarias de precisión.
- Aplicaciones en el campo de investigación social e interacción humana.
- Educación y entretenimiento.

1.2. Objetivos

1.2.1. Objetivo general

Implementar una aplicación prototipo para comandar a un *Robot PR2* a través de instrucciones por voz, utilizando arquitectura cliente-servidor utilizando un micrófono como elemento para muestrear la señal de audio o comando enunciado por el usuario.

1.2.2. Objetivos específicos

Como objetivos específicos se tiene lo siguiente:

- Estudio de ROS aplicado en *Robot PR2*.
- Elaboración de una interfaz WEB, que permita la grabación de audio para que el usuario pueda interactuar con el *Robot PR2*.
- Creación de base de datos, modelo de lenguaje y transcripción para los comandos que serán reconocidos en el sistema.
- Utilización de *toolkit* KALDI para ASR.
- Implementación de movimientos del cuerpo de el *Robot PR2*.
- Utilización de sensor Kinect para función de imitación.
- Demostración en tiempo real de la aplicación mostrando el funcionamiento coherente de la interfaz de interacción con el usuario, el sistema de reconocimiento automático de voz y movimientos de el *Robot PR2*.

1.3. Estructura de la memoria

El capítulo 2 tiene por objetivo interiorizar al lector en las tecnologías de procesamiento de voz, en una forma que no requiera ningún tipo de conocimiento previo para entender los subsecuentes capitulos. Además se muestra como funciona ROS a través de el *Robot PR2*.

El capítulo 3 tiene por objetivo mostrar al lector el desarrollo de la implementación de la aplicación que permite comandar al *Robot PR2*. a través de la voz.

El capítulo 4 y 5 muestra los resultados, conclusiones y trabajo futuro a realizar en base al prototipo propuesto en esta memoria.

Capítulo 2

Marco teórico

A continuación se presentan los fundamentos teóricos necesarios para comprender el desarrollo del *Sistema Interactivo por Voz para Robot PR2*. Se enfatiza en todo lo relacionado con el diseño de movimientos orientados al manejo del robot mediante *software* y el procesamiento de la voz.

Dentro de este campo de investigación se ha realizado desarrollo en el control del movimiento de un brazo articulado mediante la voz [6] [7]. Así como también un brazo articulado con la habilidad de escribir luego de reconocer lo que el locutor ha expresado [8]. Todo con el objetivo de realizar un aporte en la calidad de vida de personas que están inhabilitadas para escribir. Continuando en esta senda de robots de servicio, se desarrolló un brazo articulado que puede ser añadido a una silla de rudas que asiste a los usuarios en sus actividades cotidianas [9] [10].

Específicamente con respecto al *Robot PR2*, su manipulación ha estado enfocada a: la gestión del movimiento de sus brazos articulados; la interacción con el entorno; visión y detección de objetos; realización de actividades repetitivas [11] [12] [13] [14]. Esto deja una línea de investigación actualmente poco explorada, que corresponde a la integración de las técnicas de procesamiento de voz y la manipulación del *Robot PR2* mediante la voz.

2.1. Procesamiento de voz

La comunicación en todas su formas, es una acción necesaria para todos los humanos. Desde el contexto de *Human Robot Interaction* o Interacción Humano Robot, establecer la conexión entre un emisor - receptor de un hablante humano y un robot, es un hito en el cual la voz es la forma como se transmite del mensaje, y por lo tanto foco central de esta investigación.

En esta sección se muestran los contenidos referentes al proceso de la emisión, producción, técnicas de análisis y procesamiento de voz, que permiten realizar la transcripción de una señal de audio a una cadena de texto.

2.1.1. Producción y emisión de la voz

Haciendo referencia a lo anteriormente enunciado, la voz es esencial en el proceso de la comunicación, es por medio de éste que se construye el discurso, proceso clave en la Interacción Humano Robot.

Desde una perspectiva biológica la producción de la voz es el resultado de una serie de mecanismos que incluyen diferentes órganos del aparato respiratorio del cuerpo humano.

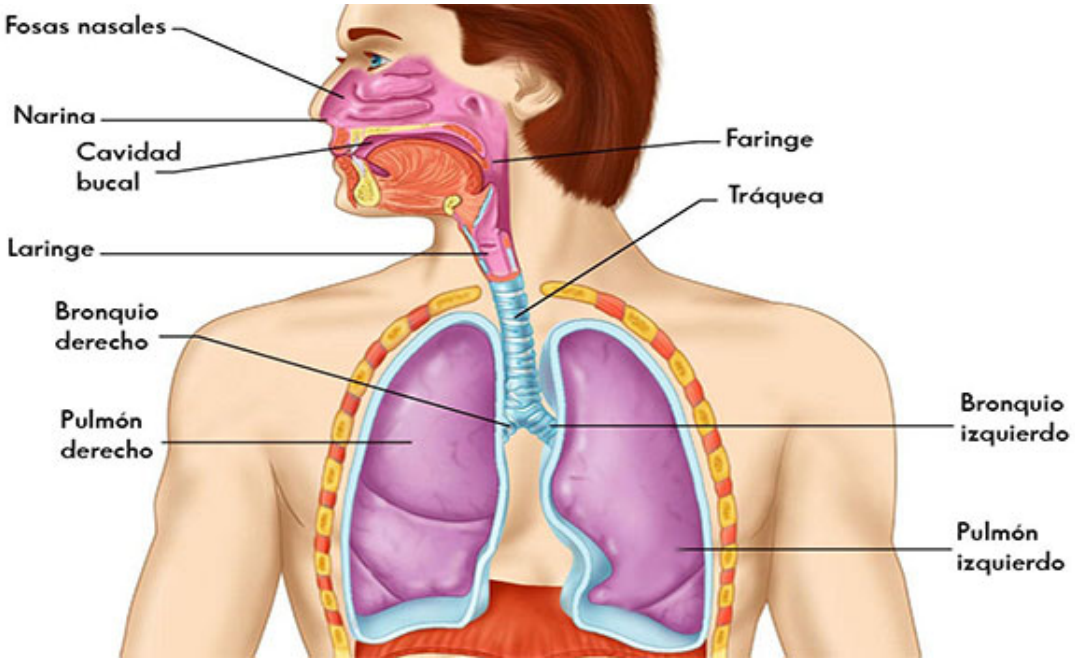

Figura 2.1: Aparato fonador humano [2].

El aparato fonador del cuerpo humano, como se observa en la Figura 2.1, es el encargado de la producción voluntaria de la voz, el cuál incluye varios conjuntos de órganos del cuerpo: respiratorios (pulmones, bronquios y tráquea); de fonación (cavidad glotal: laringe y cuerdas vocales, resonadores nasal y buco-faríngeo); y de articulación (paladar, lengua, dientes, labios, velo y mandíbula).

La laringe es el órgano principal del aparato fonador, ya que en ella se encuentran situadas las cuerdas vocales o pliegues vocales, ligamentos elásticos que se encuentran entre las paredes de la laringe, como se observa en la Figura 2.2.

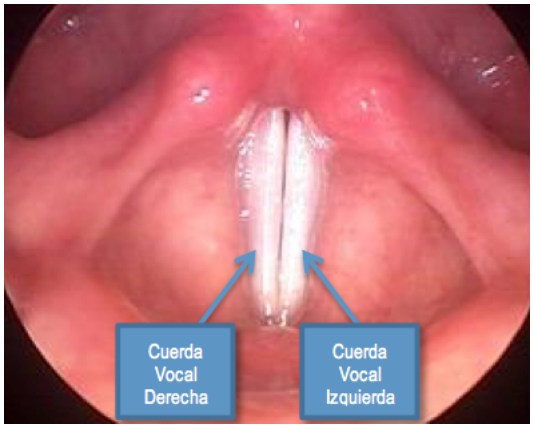

Figura 2.2: Cuerdas vocales cerradas [3].

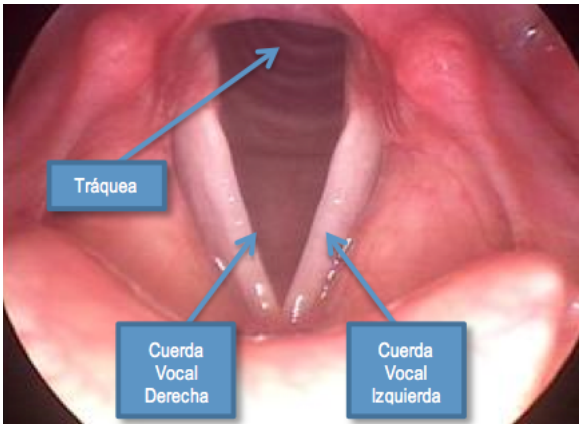

Figura 2.3: Cuerdas vocales abiertas [3].

Las imágenes 2.2, 2.3 fueron capturadas en un procedimiento conocido como *Laringoscopía*, que corresponde a un examen visual para evualar el estado de salud de las cuerdas vocales.

Finalmente la emisión del sonido por el aparato fonador, puede ser segmentado de forma general en dos etapas que ocurren de forma cíclica que se explican a continuación:

2.1.1.1. Inspiración

Es el proceso en el cuál ingresa el aire, específicamente oxígeno desde el medio exterior hacia los pulmones. Este mecanismo es producido principalmente por la el diafragma , músculos abdominales y de la caja torácica (Figura 2.1).

Cuando el diafragma se contrae, se mueve hacia abajo, los músculos pectorales menores y los intercostales presionan las costillas hacia fuera provocando un aumento en el volumen en la caja torácica, con lo cual ingresa el aire a través de la cavidad nasal y/o bucal atravesando el aparato fonador (con las cuerdas vocales abiertas, como en Figura 2.3), luego hacia la tráquea llegando a los bronquios y finalmente a los pulmones.

2.1.1.2. Exhalación

Ocurre cuando el diafragma se relaja, volviendo su posición normal, provocando un efecto opuesto al de la inspiración. Es así que la corriente de aire emanada desde los pulmones atraviesa los bronquios y la tráquea, pasando por las cuerdas vocales (Figura 2.2), las cuales vibran de forma elástica con éste flujo de aire.

Finalmente dicha columna de aire es amplificada y modificada por los resonadores nasales y las cavidades buco-faríngea, para luego ser moldeada por los órganos articulares y así emitir un sonido.

2.1.2. Sonido articulado

Los fonemas son la articulación mínima de un sonido vocálico y consonántico. Éstos no son sonidos con entidad física, sino abstracciones mentales o formales de los sonidos del habla. En este sentido, un fonema puede ser representado por una familia o clase de equivalencia de sonidos, técnicamente denominados fonos, que los hablantes asocian a un sonido específico durante la producción o la percepción del habla [15] [16].

Desde un punto de vista estructural, el fonema pertenece a la lengua, mientras que el sonido pertenece al habla. La palabra $< casa >$, por ejemplo, consta de cuatro fonemas (/k/, /a/, /s/, /a/). A esta misma palabra también corresponden en el habla, cuatro sonidos, a los que la fonología denomina alófonos, y estos últimos pueden variar según el sujeto que lo pronuncie, mientras que el fonema se considera invariante.

Existe una detallada clasificación de los fonemas, con un respectivo ejemplo que se muestra en la Tabla 2.1.

Punto articulación	
Bilabial	/*p*/
Labiodental	/*f*/
Dental	/*t*/
Interdental	/Θ/
Alveolar	/*l*/
Palatal	/*y*/
Velar	/*g*/
Modo articulación	
Oclusivo	/*p*/
Fricativo	/*f*/
Africado	/*ch*/
Lateral	/*l*/
Vibrante	/*r*/
Cuerdas vocales	
Sonoro	/*b*/
Sordo	/*p*/
Columna aire	
Nasal	/*n*/,/*m*/,/ñ/
Oral	Todos excepto fonemas nasales

Tabla 2.1: Clasificación fonemas según distintos modos de producción [1].

2.1.3. Reconocimiento automático de voz

El progreso realizado en cuanto al desarrollo de tecnologías de reconocimiento de voz o ASR y el entendimiento de la naturalidad del lenguaje, ha estado presente alrededor de cuatro décadas [17]. En sus inicios aproximadamente en los años 1960 ya era posible reconocer pequeños vocabularios (del orden de 10-100 palabras), basados simplemente en las propiedades acústicas y fonéticas de los sonidos hablados.

La clave de estas tecnologías fue el desarrollo durante esta etapa del análisis utilizando bancos de filtros, simples métodos de normalizar y el comienzo de la programación dinámica. Ya en la década de 1970 era posible reconocer vocabularios de tamaño medio (del orden de 100-1000 palabras) usando métodos de reconocimiento de patrones. La clave en esta época fueron los desarrollos de modelos de reconocimiento de patrones, el concepto de métodos LPC para representación espectral y la introducción de métodos de programación dinámica.

En la década de 1980 se comienza a manipular problemas con vocabularios de gran tamaño (mayor a 1000 palabras), la clave del éxito fue la introducción de los modelos HMM (*Hidden Markov Model*) [18] [19] y los modelos de lenguaje estocásticos, los cuales en conjunto permiten la utilización de nuevos métodos para resolver el problema de reconocimiento

automático de voz de forma continua, con gran desempeño y confianza.

Alrededor de los años 1990 era posible construir grandes sistemas de vocabulario sin restricciones de modelo de lenguaje basados en el desarrollo de métodos estocásticos para el entendimiento del lenguaje, modelos acústicos y de lenguaje basados en conceptos estadísticos y la introducción de FST (*Finite State Transducer*). Finalmente en los últimos años se ha desarrollado sistemas con grandes vocabularios con modelos semánticos completos, integrados a sistemas de síntesis de texto (TTS), que incluyen las técnicas de *Machine Learning*, para mejorar el rendimiento de la comprensión del discurso.

2.1.4. Arquitectura reconocedor rasado en modelo HMM

En la Figura 2.4 se muestran los componentes principales de un reconocedor de voz con un vocabulario de gran tamaño (mayor a 1000 palabras).

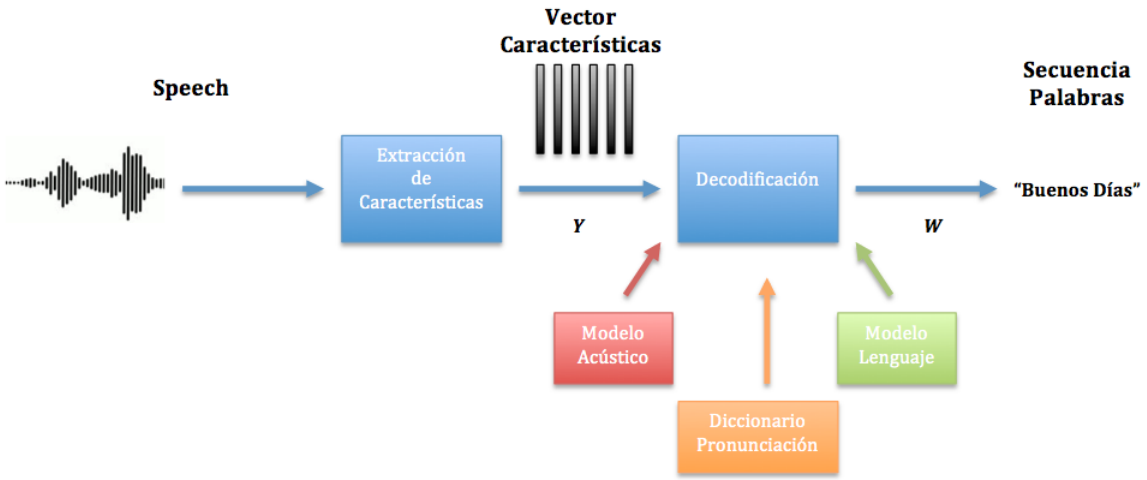

Figura 2.4: Arquitectura sistema basado en modelo HMM.

La entrada al sistema es una señal de audio, que es convertida en una secuencia de tamaño fijo de vectores $Y = [y_1 ... y_T]$, luego de la etapa de extracción de características. La etapa de decodificación busca encontrar la secuencia de palabras $W = [w_1 ... w_T]$ que mas se parece a la secuencia de generada por Y. Formalmente, la etapa de decodifcación se define como:

$$\hat{W} = \arg\max_{w}[p(W|Y)] \tag{2.1}$$

Sin embargo modelar directamente la probabilidad $p(W|Y)$ no es sencillo, aunque utilizando el teorema de probabilidad de Bayes, la ecuación 2.1 queda definida en:

$$\hat{W} = \arg\max_{w}[p(Y|W)p(W)] \tag{2.2}$$

En donde $p(Y|W)$ corresponde a la probabilidad calculada a través de las características acústicas dado una secuencia de palabras, quedando determinada por el modelo acústico en la etapa de decodificación, mientras que $p(W)$ es determinado por el modelo de lenguaje.

En la práctica el modelo acústico no es normalizado y el modelo de lenguaje es en escalado por una constante empírica y una constante que penaliza la inserción de palabras w_i en una determinada secuencia W. La verosimilitud en el dominio logarítmico se calcula mediante:

$$\Lambda(\hat{W}) = \log p(Y|W) + \alpha p(W) + \beta|W| \tag{2.3}$$

Típicamente los valores de constantes se encuentran en el rango:

- $\alpha = [8,20]$
- $\beta = [0,20]$

Es así que dada una secuencia de palabras W, el correspondiente modelo acústico es sintetizado mediante la concatenación de fonemas utilizados para construir palabras presentes en el modelo diccionario de pronunciación. El modelo de lenguaje de una secuencia W, es generalmente un modelo N-Gram, en el cual la probabilidad de cada palabra w_i está condicionada solo a sus $N-1$ predecesores.

Finalmente la decodificación opera realizando una búsqueda a través de todas las posibles secuencias de palabras, eliminando las palabras w_i poco probables. Cuando la última elocución es alcanzada, la secuencia de palabras W con mayor probabilidad es retornada.

2.1.5. Extracción de características

Para realizar procesar la señal en una primera etapa es necesario dividir en *frames*, para ello se utiliza una ventana de *Hamming* [20], con el fin de evitar discontinuidades al inicio y final de cada segmento. Continuando con el análisis espectral se aplica una transformada de *Fourier*. Dado que el oído humano no es capaz de percibir frecuencias puntuales, sino que intervalos, se utilizan filtros en una escala representativa: Mel.

$$Mel(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.4}$$

Los filtros corresponden a funciones triangulares con ganancia unitaria en la frecuencia central, con un traslape de 50 % y un ancho de banda fijo en escala Mel.

Luego de que la señal ha sido filtrada se obtienen los *Mel Frecuency Cepstral Coeficient* (MFCC) [21] [22], con lo que de forma acústica la señal queda representada por un vector de coeficientes con los parámetros que lo identifican de forma única.

2.1.6. WER : Word Error Rate

El WER es una forma de medir la calidad de un sistema de reconocimiento automático de voz. Compara la referencia con la hipotésis original de la siguiente forma:

$$WER = \frac{S + D + I}{N} \tag{2.5}$$

En donde:

- S es el número de sustituciones.
- D es el número de eliminaciones.
- I es el número de inserciones.
- N es el número total de palabras en la hipótesis.

El cálculo del WER no es más que la distancia de para palabras [23] y su rango de valores es siempre positivo. Es exactamente cero cuando la hipotesis y la referencia son idénticas. En la Tabla 2.2 se muestran ejemplos prácticos para las aseveraciones anteriores.

Hipótesis	**Referencia**	**S**	**D**	**I**
Hola como estas	Hola estas	0	1	0
Hola como estas	Hola como tu estas	0	0	1
Hola como estas	Hola quien estas	1	0	0

Tabla 2.2: Muestra ejemplo sencillo de computo de ineserciones, eliminaciones y susticiones en el cálculo del WER.

2.2. Robot PR2

Sin duda en un mundo donde el desarrollo de la tecnología, se sustenta en la base de asistir al hombre en sus quehaceres tanto cotidianos como exclusivos de una actividad particular, la creación de máquinas que realicen actividades específicas toma relevancia.

En particular, máquinas inteligentes que de alguna manera intentan emular el comportamiento humano al realizar una tarea. Siguiendo esta idea, es dónde la robótica establece su nicho de investigación, en particular:

- Brazos mecánicos articulados.
- Robot bípedo (desplazamiento a través de dos pies).
- Robot humanoide (brazos, torso y piernas articuladas).

El objetivo de la compañía Willow Garage al desarrollar el *Robot PR2*, se enfoca a investigar aplicaciones que le permitan a las personas crear un ambiente mas productivo en el trabajo y en el hogar, ya que posee la movilidad de navegación (traslado), el tamaño promedio, la precisión y los grados de libertad de las articulaciones humanas.

A nivel de software la ventaja de trabajar con una licencia de código abierto, es que permite la cooperación internacional y multidisciplinaria con otros usuarios de el *Robot PR2*, además de la posibilidad de realizar cambios profundos a la programación de su sistema operativo ROS, en caso de ser necesario. En líneas generales la construcción de el *Robot PR2*, es de material robusto, lo cual permite la operación y experimentación dentro de condiciones de laboratorio.

2.2.1. Hardware de Robot PR2

Dotado de gran cantidad de grados de libertad en sus brazos, torso y base, el *Robot PR2* por lo demás cuenta con múltiples sensores (entre ellos un sensor Microsoft Kinect) y cámaras que permiten obtener las condiciones del entorno en el que se encuentra.

A continuación se describen detalles técnicos respecto del *hardware* de el *Robot PR2*. Cuyos brazos poseen una gran extensión y capacidad de rotación como se puede ver en la Figura 2.5. Los grados de libertad del brazo derecho e izquierdo son idénticos y se enlistan a continuación:

- Grado de libertad brazo **(A)** : 4.

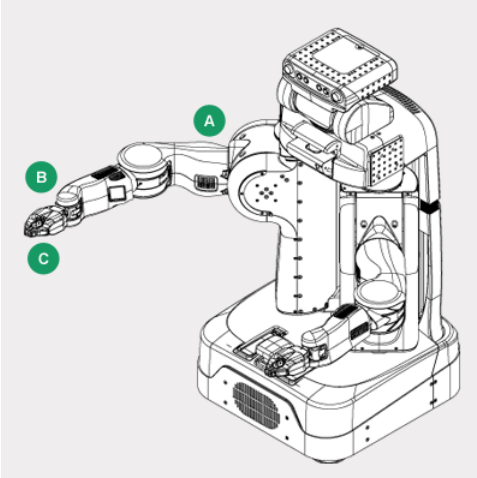

Figura 2.5: Brazo derecho. Libertad de movmiento [4].

- Grado de libertad muñeca **(B)** : 3.
- Grado de libertad pinza **(C)** : 1.

Cada brazo tiene una extensión de 400 mm, el antebrazo de 321 mm y la superficie útil de la pinza varía entre 120 mm y 240 mm. El rango de movimiento de cada una de las articulaciones del brazo es bastante amplia y se detalla a continuación:

- Inclinación hombro: 170^o.
- Movimiento horizontal hombro: 115^o.
- Flexión codo: 114^o.
- Movimiento muñeca : 130^o.
- Giro de la muñeca : Continuo 360^o.
- Giro del antebrazo : Continuo 360^o.

Respecto de la Figura 2.6, la base del robot **(C)** posee cuatro ruedas que tienen la capacidad de girar en 360^o y se encuentran en una base cuadrada de 689 mm por lado.

Es importante destacar en la base **(D)** se encuentra todo el *hardware* computacional. Dicho equipamiento incluye:

- Dos procesadores Intel Quad-Core $i7$ Xeon de ocho núcleos físicos cada uno.
- $24Gb$ de memoria RAM.
- Un disco duro de 1.5 TB.

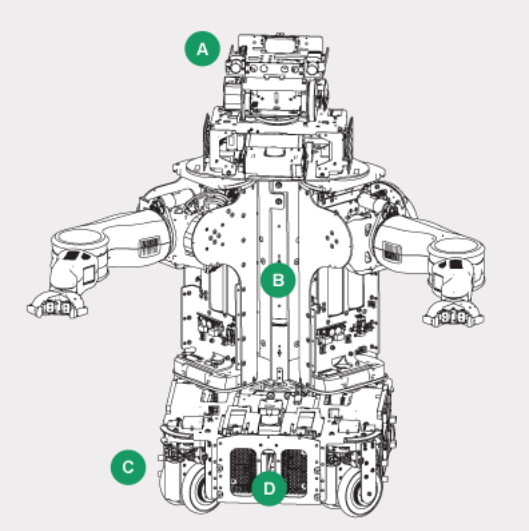

Figura 2.6: Cabeza, torso y base de Robot PR2 [4].

- Un disco duro de 500 GB.

La cabeza del robot **(A)**, puede moverse libremente en 350^o horizontalmente y en 115^o verticalmente. El torso **(B)** es extensible, es decir, la altura del robot varía desde 1330 mm a 1645 mm, medidos desde la base hasta la cabeza.

En el torso además se encuentra todas las conexiones de red del robot. Éstas incluyen:

- Cisco 32 Gigabit Switch.
- Wifi 802.11b/g.
- Punto de Acceso Bluetooth.
- Red EtherCAT.
    - Control de motores.
    - Cámaras de antebrazo y cámaras frontales estéreo.

El PR2 es un robot que funciona con energía eléctrica, por lo que debe ser conectado a un red eléctrica domiciliaria AC 220 V y 50 Hz para su operación y/o carga de baterías.

En la sección posterior de la base **(D)** se encuentra un banco de baterías de Litio con una capacidad de almacenamiento de 1.3 kWh, energía que le permite una autonomía de aproximadamente dos horas de funcionamiento continuo.

Al ser un robot eléctrico, su autonomía nominal de dos horas de funcionamiento continuo varía dependiendo de las actividades realizadas como: esfuerzo en los motores de brazos, movimientos de base en caminos planos o inclinados, gran procesamiento de datos, el tiempo de funcionamiento de los sensores y cámaras.

Haciendo referencia a lo anteriormente expresado, el *Robot PR2* cuenta con una gran diversidad de sensores que le permiten obtener información acerca de la condiciones del entorno. Dichos sensores se ubican distribuidos a lo largo de todo el cuerpo del robot: brazos, torso, cabeza, base. Comos se aprecia en la Figura 2.7.

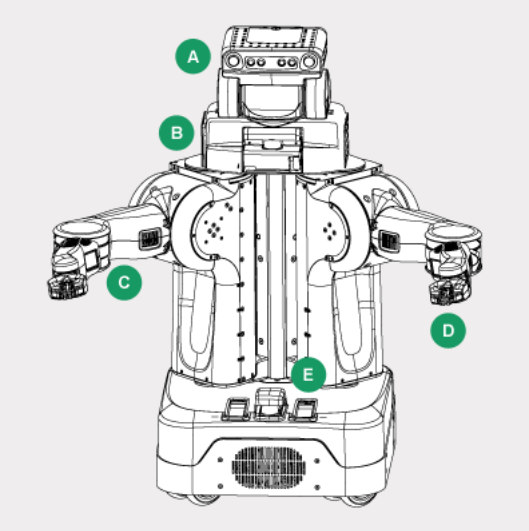

Figura 2.7: Distribución de sensores [4].

En la cabeza **(A)**, se encuentran todos los dispositivos que se utilizan en vision computacional y procesamiento de imágenes:

- Microsoft Kinect Xbox 360.
- Cámara Color 5 Mega Pixel.
- Cámara Estéreo *Wide-Angle*.
- Cámara Estéreo *Narrow-Angle*.

Sobre los hombros del robot **(B)** se encuentra un dispositivo láser Hokuyo UTM-30LX y un giroscopio Microstrain 3DM-GX2 IMU. En la base **(E)**, se utiliza el mismo dispositivo que se encuentra en los hombros.

En el antebrazo *(C)* se encuentra localizada una cámara color. Finalmente en las pinzas **(D)** se encuentra un acelerómetro de tres ejes, y en las puntas de las pinzas un arreglo de sensores de presión.

2.2.2. Sensor Kinect

Sobre la cabeza del robot PR2 se encuentra instalada un Sensor Microsoft Kinect Xbox 360 [24] [25] [26] , idéntico al de la Figura 2.8.

Figura 2.8: Sensor Kinect sin cubierta de plástico [5].

Éste sensor en líneas generales posee:

- Arreglo de cuatro micrófonos.
- Cámara color resolución 640x480.
- Proyector y receptor infrarrojo.

Los cuales se encuentras distribuidos espacialmente como se muestra en la Figura 2.9.

En donde **(1)** corresponde a un láser infrarrojo de 830ñm y 60 mW de potencia. Mientras **(3)** es un sensor CMOS infrarrojo 1280×1024 y 640×480 a 30fps en vídeo, con un rango de operación entre: 0.8 m - 3.5 m. En conjunto ambos dispositivos trabajan para obtener la

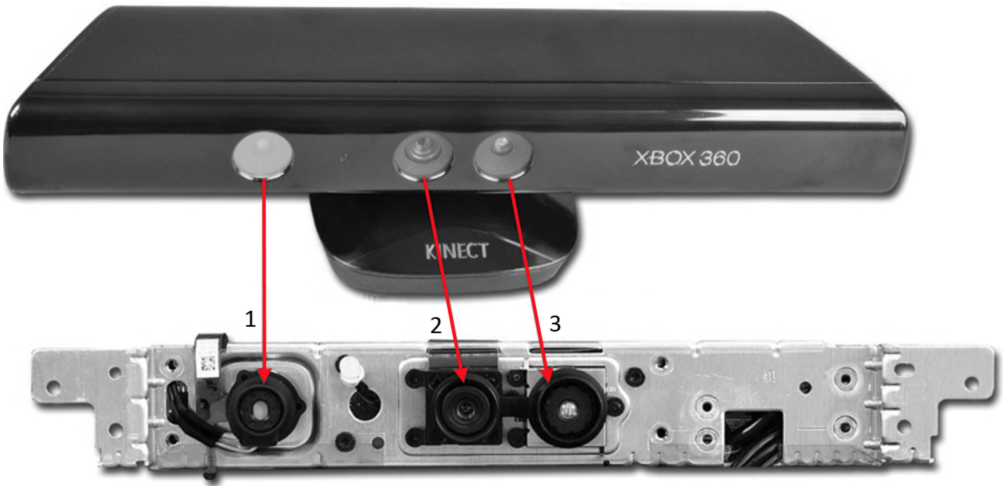

Figura 2.9: Sensor Kinect y componentes principales [5].

profundidad (distancia entre el sensor y el usuario).

El emisor infrarrojo proyecta un patrón irregular de puntos (una grilla) que varía en intensidad. Así el sensor CMOS infrarrojo detecta la distorsión en la grilla para así reconstruir la imagen y calcular la profundidad.

Finalmente **(2)** corresponde a una cámara color con resolución 640×480 en imagen y 640×480 a 30fps en vídeo.

Respecto del audio, el sensor Kinect trae integrado cuatro micrófonos distribuidos en toda la extensión de su carcasa, como se observa en la Figura 2.10.

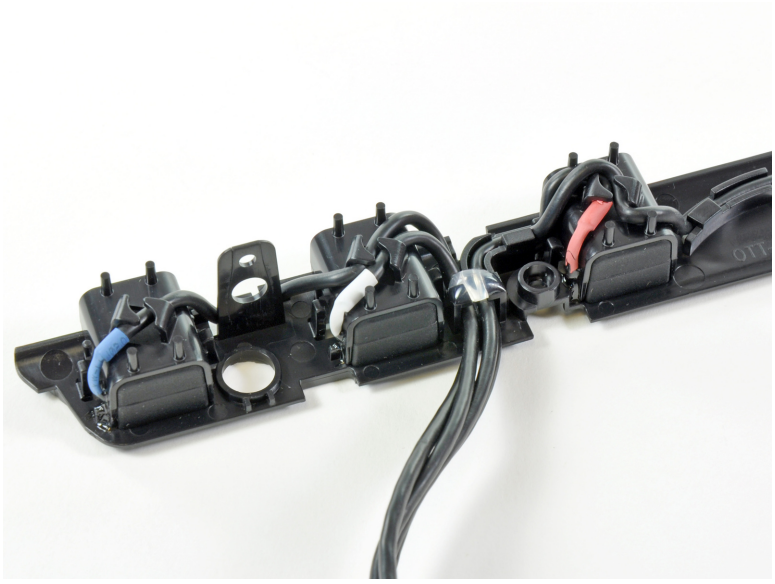

Figura 2.10: Micrófonos Kinect [5].

Por requerimientos de diseño, 3 micrófonos del sensor Kinect se encuentran ubicados en el lado derecho y uno en el izquierdo, respecto de la Figura 2.8.

La ventaja del uso de un arreglo de micrófonos orientados especialmente, es que permite detectar la fuente de emisión de sonido y segmentar elocuciones de diferentes fuentes.

2.2.3. ROS : El sistema operativo para robots

ROS es una plataforma de desarrollo de código abierto, para robots. Como sistema operativo ofrece una serie de librerías que se enfocan en el desarrollo de aplicación para la manipulación sencilla de robots.

Desde su creación, ROS ha sido diseñado para facilitar el intercambio de códigos y aplicaciones entre los investigadores del área. Gracias a su licencia de código abierto (BSD/APACHE) al momento de crear una aplicación se pueden emplear lenguajes de programación como Python y C++.

Éste sistema operativo solo puede ser ejecutado por sistemas operativos basados en UNIX, tales como:

- Ubuntu.
- OSX.
- Fedora.
- Gentoo.
- Debian.
- ArchLinux.

En sistemas operativos como Windows, no es posible ejecutar de forma nativa ROS, y solo es posible su ejecución a través de una maquina virtual que emule un sistema operativo basado en UNIX.

ROS es el primer sistema operativo orientado a robots, creado por Willow Garage, ubicada en la ciudad de Menlo Park en el estado de California en Estados Unidos de América, una compañía y laboratorio de investigación robótica que se dedica a la creación de software de código abierto con licencia BSD. El sistema operativo está en constante actualización, y como proyecto participan una gran cantidad de colaboradores [1] que desarollan y reparan librerías.

Es por esto que ROS tiene como principal objetivo ser desarrollado en un entorno de colaboración. Aunque todo aporte debe estar en la misma dirección, es decir ROS debe ser:

- Sencillo e integrable, para que así quien esté interesado pueda realizar su aporte.

[1] http://www.ros.org/contributors/

- Libertad en la utilización de lenguajes de programación, punto importante ya que no todos los investigadores del área son expertos en el mismo lenguaje de programación. Así el esfuerzo se concentra en el objetivo del programa (acción robot, detección de patrones, movimientos en un espacio confinado) y no en el lenguaje de programación.
- Escalable para que un conjunto de aplicaciones pequeñas sean capaces de funcionar en conjunto en un gran sistema estable.
- Mantener el sistema de archivos para facilitar el orden.

Figura 2.11: Estructura de ROS.

La estructura del sistema operativo ROS funciona similar al esquema de la Figura 2.11, de forma jerárquica y colaborativa, como un sistema de engranajes. Dentro de los componentes principales, se identifican:

- Pila (*Stack*).
- Paquete (*Package*).
- Nodo *(Node)*.
- Servicios y Mensajes *(Services)*.

Es así como los *Stack* o Pilas corresponden al conjunto de *Packages* o Paquetes que cumplen una determinada función. Los paquetes son la unidad básica y principal del sistema operativo, cada uno de ellos puede contener: conjuntos de datos, archivos de configuración, librerías, dependencias a otros paquetes y uno o varios nodos.

Los nodos corresponden a los ejecutables, es decir, los procesos que se realizan en el robot. Gracias al diseño de ROS y su estructura modular un paquete puede contener muchos nodos, escritos en diferente lenguajes de programación (Python o C++), por ejemplo: En el *Robot PR2* existe $pr2\_bringup$ que es uno de los primeros paquetes que se ejecuta cuando se inicia el robot, contiene varios nodos, entre ellos $recalibrate.launch$ que precisa las instrucciones necesarias ara realizar una re-calibración de los sensores y motores cuando se estime conveniente. Los mensajes son la forma de comunicación de los nodos, mientras que los servicios establecen la comunicación entre los requerimientos de ROS.

2.2.4. Instalación de ROS

La instalación de ROS, se realizó en un *laptop* con el sistema operativo Ubuntu 12.04, para eso primero se debe actualizar `sources.list` ejecutando en un terminal la instrucción:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main"
            > /etc/apt/sources.list.d/ros-latest.list"
```

Y luego al termino de la instrucción anterior ejecutar la siguiente linea de comando en el terminal:

```
wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.key
-O - | sudo apt-key add -
```

Luego basta solo actualizar la lista de los repositorios e instalar la versión completa de ROS.

```
sudo apt-get update
sudo apt-get install ros-hydro-desktop-full
```

Al finalizar la descarga es necesario inicializar `rosdep`, que habilita de forma simple la instalacion de dependencias de sistema para compilar y también para módulos de ROS necesarios para su ejecución

```
sudo rosdep init
rosdep update
```

Finalmente basta configurar las variables de entorno para habilitar la ejecución de los programas de ROS en cada sesión del terminal:

```
echo "source /opt/ros/hydro/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Con esto finalizan las instrucciones para instalar ROS, y para probar que todo haya sido instalado de forma correcta solo basta ejecutar `roscore` en un terminal.

2.2.5. Creación de paquetes en ROS

Existen dos formas para crear paquetes nuevos en ROS. Una hace uso de `roscpack` que crea los paquetes en el directorio donde está instalado ROS que usualmente es `/opt/ros/`.

Debido a que está carpeta no se puede editar sin permisos de `root` y hacer cambios a nivel de sistema en versiones de prueba puede inducir a errores mayores, es recomendable utilizar lo que se conoce como `catkin_workspace`, un carpeta de trabajo personal que no requiere permisos de `root`.

Para crear una carpeta de trabajo personal, primero se elige la ubicación en donde se creará, en un emulador de terminal se escribe lo siguiente:

```
cd
mkdir catkin_ws_SGD
```

Luego de que la carpeta haya sido creada es necesario inicializar el espacio de trabajo, creando un vinculo entre los directorios donde está ROS y la carpeta de trabajo. Para ello dentro de `catkin_ws_SGD` se crea la carpeta `src` y se ejecuta la instrucción de inicialización:

```
cd ~/catkin_ws_SGD
mkdir src
cd src
catkin_init_workspace
```

Con esto es factible crear paquetes en el espacio de trabajo personal sin interferir con los archivos críticos de sistema ubicados en la carpeta `/opt/ros/hydro`. Todos los paquetes nuevos se crean en la carpeta `src`, de la siguiente forma:

```
cd ~/catkin_ws_SGD/src
catkin_create_pkg <nombre_paquete> <dep_1> .... <dep_n>
```

En donde para crear el paquete se necesita primero un nombre y luego todas las dependencias. A modo de ejemplo la creación de un paquete para el movimiento del brazo se ejecuta con la siguiente sentencia:

```
cd ~/catkin_ws_SGD/src
catkin_create_pkg sgd_motion actionlib roscpp pr2_controllers_msgs
```

Así el paquete `sgd_motion`, depende de `actionlib`, `roscpp` y `pr2_controller_msgs`. El código fuente que se compilará en este paquete debe ir en la ruta:

```
/home/lptv/Desktop/catkin_ws_SGD/src/sgd_motion/src
```

Para compilar es necesario editar manualmente `CMakeList` que contiene las instrucciones para realizar la compilación, debido a que `catkin_make` utiliza `cmake`. A modo de ejemplo se muestra el contenido del archivo `CMakeList` uno de los que ha sido construido para realizar la aplicación.

Código Fuente – Ejemplo construcción CMakeList

```
make_minimum_required(VERSION 2.8.3)
    project(sgd_motion)
    #   Sebastian Guerrero Diaz
    #   seguerre at gmail dot com
    #   Otono 2015
    find_package(catkin REQUIRED COMPONENTS
    actionlib
    pr2_controllers_msgs
    roscpp
    )
    generate_messages(
    DEPENDENCIES
    pr2_controllers_msgs
    )
    catkin_package()
    include_directories( ${catkin_INCLUDE_DIRS})
    add_executable(arm_init      src/arm_init.cpp)
    add_dependencies(arm_init      sgd_motion_generate_messages_cpp)
    target_link_libraries(arm_init      ${catkin_LIBRARIES} )
```

Una vez que esto haya sido exitosamente creado basta compilar los paquetes creados de la siguiente forma:

```
cd ~/catkin_ws_SGD
catkin_make
```

2.2.6. TTS - Festival

Festival es un sintetizador de voz desarrollado en CSTR (*Centre for Speech Technology Research*). Ofrece dentro de sus librerías la traducción de texto a voz soportando una gran variedad de lenguajes, entre ellos Español.

Con la distribución Ubuntu 12.04 viene instalada una versión de Festival, pero existe la posibilidad de personalizar completamente el sintetizador. Entre las mas importantes está la personalización de la voz y la velocidad con que se enuncian las palabras.

La ejecución de festival se simplifica mediante el uso de *script* BASH, que permiten realizar tareas de forma regular, por ejemplo:

```
#!/bin/bash
echo "Sistema Interactivo por Voz PR2" | festival --tts
```

Capítulo 3

Implementación del sistema PR2SPK

En el presente capítulo se detalla el proceso de desarrollo del **"Sistema Interactivo por Voz para Robot PR2"** o como se mencionará desde ahora **PR2SPK**. En líneas generales el sistema implementado está constituido en tres bloques: Interacción con el usuario; Procesamiento señales; Accionamiento.

3.1. Equipamiento

A continuación se presenta en detalle el equipamiento utilizado en el desarrollo de la aplicación **PR2SPK** profundizando en el detalle de: micrófonos de adquisición, descripción del servidor APACHE-PHP y la conexión entre la aplicación y el *Robot PR2*.

3.1.1. Micrófono de adquisición

El micrófono utilizado para la adquisición de los datos es uno externo que posee un pedestal en su base. La Figura 3.1 muestra físicamente como esta confeccionado el dispositivo, dentro de las características que posee se encuentran:

- Modelo: C-200.
- Cobertura: Omnidireccional.
- Impedancia: 2,2 kΩ.
- Sensibilidad: -48dB +/- 3dB.
- Frecuencia de respuesta: 100 Hz a 10 kHz.
- SNR: 40dB.
- Dimensiones: 100 mm x 280 mm x 100 mm.

- Conexión: Jack de 3,5 mm .
- Extensión cable : 1.5 m.

Figura 3.1: Micrófono utilizado en el sitema **PR2SPK**.

La elección de un micrófono común tiene por objetivo que el sistema de reconocimiento de voz sea entrenado en condiciones pedestres, que sean accesibles a todas las personas y que un micrófono no sea una barrera en la interacción con **PR2SPK**.

3.1.2. Servidor ASR y APACHE-PHP

El servidor ASR y APACHE-PHP se encuentran instalados en un mismo *host*, aunque el sistema está diseñado para que ambos servidores se comuniquen remotamente si estuvieren instalados en *host* físicamente diferentes. La elección de unificación tuvo por objetivo la portabilidad del sistema.

El equipo que sustenta ambos servidores es un computador portátil **HP ProBook 440 G2** que posee las siguientes características físicas:

- Procesador Intel Core i7-5500U con gráficos Intel HD 5500.
- 16 GB de SDRAM DDR3L-1600.
- 1 TB SATA (de 5400 rpm).
- Interfaz de red Realtek Ethernet (10/100/1000).
- Combinación de Realtek 802.11b/g/n (1x1) y Bluetooth 4.0.

El equipo utiliza un sistema operativo OpenSource Ubuntu 14.04 Trusty Tahr, versión LTS que tiene un soporte de cinco años.

3.1.3. Conexión entre servidor ASR y PR2

La comunicación entre el *host* en el que se encuentra instalado el servidor ASR y el servidor APACHE-PHP con el *Robot PR2* se establace a través de una estructura que se conoce como *socket* que utilizan el protocolo TCP/IP como se observa en la Figura 3.2.

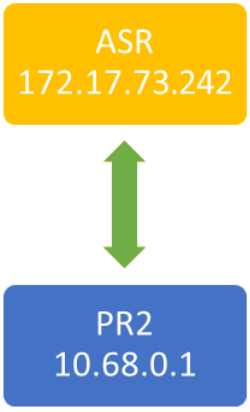

Figura 3.2: *Socket* Servidor ASR al *Robot PR2*.

El programa `pr2-server.py` se ejecuta en el *Robot PR2* y contiene el servidor que se mantiene escuchando el puerto `8889` en todo instante, hasta que el cliente `client-pr2.py` que se ejecuta en el computador portátil ó servidor ASR envía la transcripción al *Robot PR2* del audio que ha sido grabado con la aplicación **PR2SPK**. Detalles acerca de cómo se obtiene la transcripción del audio en el servidor ASR será ampliamente detallado en la sección 3.2.2.

3.2. Arquitectura del sistema PR2SPK

El sistema desarrollado se muestra en la Figura 3.3, en donde se detalla cada bloque que permite la interacción con el *Robot PR2* a través de la voz de un usuario.

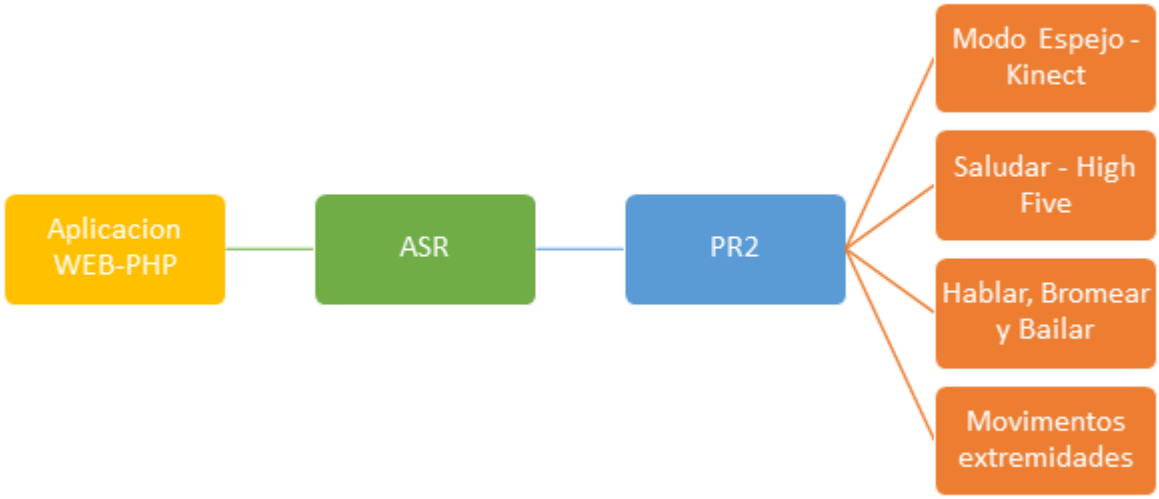

Figura 3.3: Descripción del sistema implementado.

Para acceder al sistema **PR2SPK** el usuario debe grabar el comando a que desea enviar al *Robot PR2* mediante la interfaz de usuario, que permite grabación a través del micrófono de un *laptop*. Luego de que el archivo de audio ha sido creado, la interfaz de usuario internamente envía los datos al servidor de ASR para realizar el procesamiento.

Como resultado de ASR se obtiene una cadena de texto que contiene la transcripción del audio grabado por el usuario, que es analizada por un programa que contiene el diccionario (*ver sección* 3.2.1) diseñado para el *Robot PR2* y que retornará la señal de control y accionamiento hacia el robot. Cabe destacar que dentro de los accionamientos y respuestas del *Robot PR2* que incluyen:

Movimiento

Movimientos de brazos y torso a voluntad del usuario.

Diálogo

Interacción en base a preguntas enunciadas por el usuario y respuestas generadas por el *Robot PR2* que incluyen además movimientos naturales que intentan emular una conversación (*ver sección* 3.2.3).

Otros

Acciones lúdicas que tienen por objetivo entretener al usuario, entre las cuales destacan: Chiste; Bailar; Posición Defensiva (imitando pose de arte marcial).

3.2.1. Interfaz interacción usuario

La interfaz de usuario está programada mediante lenguaje llamado HTML5. El objetivo principal al desarrollar la aplicación es ofrecer facilidad en el acceso y que no requiera conocimientos previos al momento de ser utilizada.

Para acceder a la aplicación es necesario utilizar como navegador *Google Chrome*, actualizado en su última versión, escribiendo en la ruta la dirección IP del servidor WEB, en éste caso `172.17.73.242` .La aplicación **PR2SPK** se muestra en la Figura 3.4.

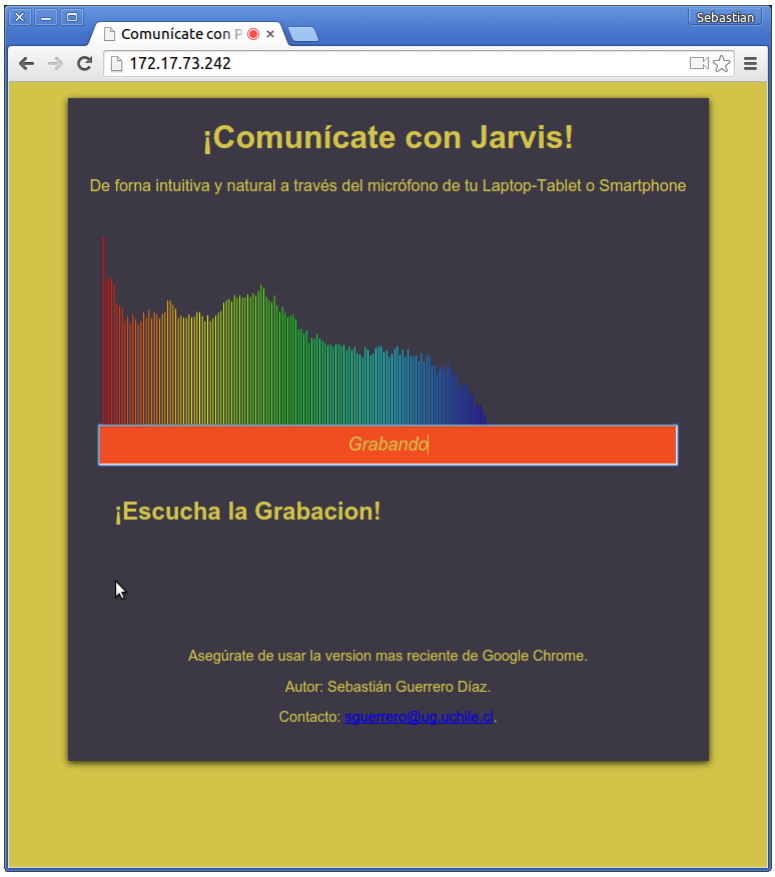

Figura 3.4: Interfaz **PR2SPK** grabando audio.

Al iniciar la aplicación, por motivos de seguridad, se pide acceso a la toma de muestras del micrófono instalado por defecto en el sistema. Por lo que es necesario hacer clic en *Allow* para permitir el acceso de la aplicación al micrófono, como se muestra en la siguiente figura 3.5:

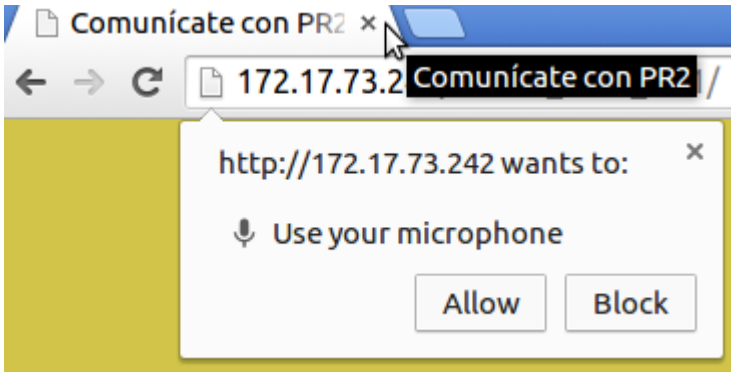

Figura 3.5: Interfaz **PR2SPK** acceso a micrófono

Luego de que se ha permitido el acceso a la toma de datos por medio del micrófono, se comienza a calcular en tiempo real el espectro de frecuencias de tamaño 2048, que corresponden a los datos que se capturan provenientes de un *buffer* con tamaño 4096.

La grabación del audio sólo se activa cuando la tecla `Enter` es presionada. Más aún, esta programado con el concepto de **Pulsar Para Hablar (PPH)**, hace referencia a que mientras esta presionada la tecla `Enter` se capturan los datos. De este modo se obtiene una señal de audio segmentada (*ver sección 4*).

La captura de audio se realiza mediante el micrófono descrito anteriormente (*ver sección 3.1.1*) y a través de la aplicación WEB se genera un archivo de audio muestreado a una frecuencia de 16 kHz y a 16 bit.

En forma de diagrama en bloques se muestra de forma detallada en la Figura 3.6 el funcionamiento y los lenguajes de programación involucrados al momento de crear la interfaz.

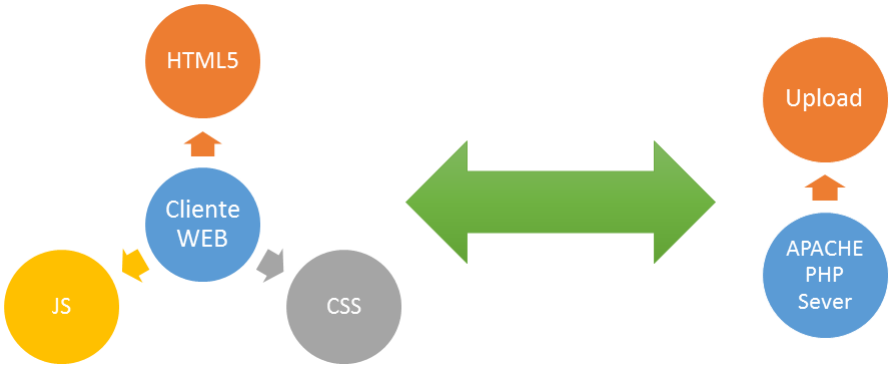

Figura 3.6: Programas involucrados en interfaz **PR2SPK**

Considerado lo mostrado anteriormente en la Figura 3.6, se establece formalmente una

comunicación bilateral entre el *Browser - Google Chrome* y el servidor WEB APACHE-PHP, que permiten que la aplicación funcione de forma adecuada a través de:

HTML5

La primera página que se carga al ingresar la IP en el navegador es `index.html`. Está escrita en código HTML5 y en ella se cargan todos los elementos visuales que se despliegan en la pantalla del usuario, por ejemplo: cuadros de texto, *canvas*, fondos e imágenes.

CSS

A través del uso de CSS se crea `style.css`, un código que contiene todas las instrucciones referentes a la presentación de los objetos que se cargan en `index.html`, dichas instrucciones engloban: posición, tamaño, color, sombra, letra.

JS

El uso de JS permite que la aplicación WEB sea dinámica y que pueda interaccionar con el usuario a través de:

- `recorder.js` código que define el tamaño del *buffer* de datos, el numero de canales del audio grabado, la frecuencia de muestreo y el manejo de la codificación de los datos a formato WAV.
- `recorder_client.js` este código contiene funciones para iniciar y detener la grabación de audio, detectar los eventos de interacción con el teclado, enviar las muestras al servidor PPH y actualizar el *canvas* con que se dibuja el espectro de frecuencias utilizando `audiodisplay.js`.
- `audiodisplay.js` este código que contiene las instrucciones para dibujar un *buffer* de datos.

Dichos códigos siempre se ejecutan en el lado del cliente (*Browser*), mientras que en el lado del servidor APACHE-PHP se ejecuta código PHP.En particular:

PHP

El código `save.php` cuya función es capturar todos los datos del *buffer* que se envían al momento de que el usuario mantiene presionado `Enter`. Cuando el usuario deja de presionar la tecla del PPH se envía la señal a `save.php` para que guarde el archivo de audio en extensión WAV en una ruta determinada en el servidor APACHE-PHP.

Para finalizar luego de que todo el proceso de interacción del usuario con la interfaz ha terminado, el archivo de audio se guarda con una estampa de tiempo como identificador.

3.2.2. Servidor de reconocimiento de voz

El servidor de reconocimiento de voz tiene por objetivo obtener la transcripción del audio que ha sido grabado mediante la aplicación **PR2SPK** utilizando KALDI. Para llevar a cabo la tarea del ASR se requieren varias etapas previas a la decodificación del sistema que se detallan a continuación:

Selección base de datos

Debido a que se requiere que el ASR sea en español, se utiliza para entrenar el sistema la base de datos "Latino 4k", que corresponden a conversaciones telefónicas espontáneas en 4992 audios con una frecuencia de 16 kHz y a 16 bit por muestra. En esta base de datos hay un total de 32 hablantes de diferentes nacionalidades de habla hispana y de diferentes géneros (masculino y femenino). Es necesario destacar que las señales de audio no tienen ningún tipo de ruido adicionado, y se encuentran en lo que se conoce como *clean*.

Grabación de base de datos DB_PR2

Además de la base de datos anteriormente mencionada se adiciona un conjunto de audios que corresponden a 5 hablantes de género masculino y con un total de 200 grabaciones (50 audios por cada hablante). Para obtener los audios se utilizó la aplicación **PR2SPK** utilizando además un micrófono como el que se muestra en la Figura 3.1. Todo esto con el objetivo de que al entrenar el sistema se consideren las condiciones ambientales y diferentes tipos de micrófono, lo que implica modificaciones en el modelo de canal de comunicación. Esta base de datos es nombrada DB_PR2.

Creación de diccionario

El diccionario contiene todas las palabras que se utilizarán al momento de realizar el ASR. Para la creación de este archivo se necesitan definir los fonemas que se reconocerán, en particular se utilizan solo los fonemas presentes en español, y se escriben en un archivo llamado `phones` en donde cada fonema está escrito con su correspondiente grafema. Luego de que este archivo está creado se procede a construir el diccionario, utilizando como base el archivo `phones`, el cual se escribe en dos columnas: palabras y su separación en grafemas, de la siguiente forma en un archivo de nombre `latino.dic`:

```
ASCENSO A S S E N S O
ASCIENDEN A S S I E N D E N
CARGADO K A RR G A D O
GERENTE J E R E N T E
GERMAN J E RR M A N
ZOOLOGICO S O O L O J I K O
```

Creación modelos de lenguaje

Parte importante en el ASR es la construcción de el modelo de lenguaje, en este caso se crea uno con el formato ARPA-MIT, que contiene la información referente a los unigramas, bigramas y trigramas tomando como base la transcripción de los audios de entrenamientos y el diccionario mencionado en la sección anterior. Utilizando el *software* de código libre que provee la Universidad de Cambridge llamado: "The CMU-Cambridge Statistical Language Modeling Toolkit v2".

Entrenamiento modelo para ASR

Al momento de realizar el entrenamiento del sistema KALDI requiere los siguientes archivos:

- Diccionario con el listados de palabras que el sistema reconocerá *(ver apéndices A y B)* .
- Modelo de lenguaje en formato ARPA-MIT construido a partir del diccionario.
- Selección del conjunto de datos para entrenamiento y para *test*, se realizó de forma aleatoria de la siguiente forma: debido a que se utilizó dos bases de datos para evitar condiciones de redundancia en el entrenamiento del del 100% de las elocuciones de la base de datos "Latino-4k"se utilizó solo el 90% del total para el entrenamiento y el 10% restante para el conjunto de *test*. De forma análoga se realizó para la base de datos DB_PR2.
- El archivo `wav.scp` contiene el nombre de los archivos y su ubicación en le sistema. De esta forma KALDI es capaz de ubicar cada archivo para realizar el procesamiento de ellos.

```
...
T3001 /home/lptv/DB_PR2/T3001.wav
T3002 /home/lptv/DB_PR2/T3002.wav
T3003 /home/lptv/DB_PR2/T3003.wav
T3004 /home/lptv/DB_PR2/T3004.wav
T3005 /home/lptv/DB_PR2/T3005.wav
T3006 /home/lptv/DB_PR2/T3006.wav
...
```

- El archivo `utt2spk` contiene las referencias a los hablantes involucrados en cada una de las elocuciones de la base de datos.

```
...
T3001 spk1
T3002 spk1
```

```
T3003 spk1
T3004 spk2
T3005 spk2
T3006 spk2
...
```

- El archivo `text` contiene la identificación de las transcripciones de cada audio que utiliza en el entrenamiento del sistema.

```
...
T3001 VOLVEREMOS A TRANSMITIRLOS CON EL FORMATO CORRECTO
T3002 EN EL PARRAFO CATORCE FIGURAN LOS COSTOS RESPECTIVOS
T3003 PERDIMOS CONTACTO DE RADIO CON ELLOS DIJO ARCE
T3004 OTROS PAISES TAMBIEN ESTAN TRATANDO EL ASUNTO
T3005 YO NO CRUCE POR EL MEDIO DE NINGUNA HACIENDA
T3006 SOY OPTIMISTA Y TENGO CONFIANZA DECLARO A LA PRENSA
...
```

- El archivo `spk2utt` contiene visto desde otra forma cuantas elocuciones existen por cada hablante, de la siguiente forma:

```
...
spk1 [T3001 T3002 T3003]
spk2 [T3004 T3005 T3006]
...
```

Luego de todos los archivos necesarios son construidos satisfactoriamente se procede a realizar el entrenamiento del sistema, proceso que tomó alrededor de 4 horas, y que consiste principalmente en: Extracción de características de tipo MFCC; Transformación modelo de lenguajes en formato *Finite State Transducer* (FST); Entrenamiento del sistema usando *Hidden Markov Model* (HMM) para mono-fonema;Alineamiento y eliminación de redundancias en modelo de lenguaje; Entrenamiento del sistema usando tri-fonema basados en el entrenamiento de mono-fonema.

Decodificación

Finalmente luego de que estas etapas están terminadas, la decodificación del sistema, es decir obtener la transcripción del audio se realiza de la siguiente forma: A partir del audio que se encuentra en el servidor APACHE-PHP se le extraen las características de tipo MFCC y luego son ingresadas al sistema entrenado, que retorna la transcripción del audio en cuestión.

3.2.3. Diseño movimientos en Robot PR2 usando ROS

Los movimientos que PR2 puede ejecutar se crearon utilizando ROS y el lenguaje de programación C++. Existen múltiples rutinas que se implementan con distintas clases como se muestra a continuación en las Tablas 3.1, 3.2, 3.4 y 3.3.

Clase Robot Arm	
arm_init	calibrate
byebye_1	right_arm_up
byebye_2	left_arm_up
conversation	right
defensa	left
ironman	tuck_r
dance	tuck_l

Tabla 3.1: Rutina clase RobotArm.

Clase Robot Head
afirmation
negation

Tabla 3.2: Rutina clase RobotHead.

Clase Torso
torso_up
torso_down
torso_up_down

Tabla 3.3: Rutina clase RobotTorso.

Clase Mirror Kinect
follow_left
follow_both
follow_both_ind
follow_both_elbow

Tabla 3.4: Rutina clase Mirro-Kinect.

A en los siguientes apartados se muestra de forma general como se realiza el control de las distintas partes del cuerpo del *Robot PR2*. No se analiza completamente en detalle el código debido a la gran extensión de cada uno de ellos, sino que se enfocan la forma como han sido construidos.

3.2.3.1. Clase RobotArm

La clase RobotArm se crea con el fin de controlar los brazos a voluntad a través del modulo `JointTrayectoryAction` y la librería de ROS `actionlib`. Dado que el brazo tiene 7 grados de libertad, y se definen de la siguiente forma para el brazo derecho:

Código Fuente – Definición articulaciones para el brazo derecho.

```
goalr.trajectory.joint_names.push_back("r_shoulder_pan_joint");
goalr.trajectory.joint_names.push_back("r_shoulder_lift_joint");
goalr.trajectory.joint_names.push_back("r_upper_arm_roll_joint");
goalr.trajectory.joint_names.push_back("r_elbow_flex_joint");
goalr.trajectory.joint_names.push_back("r_forearm_roll_joint");
goalr.trajectory.joint_names.push_back("r_wrist_flex_joint");
goalr.trajectory.joint_names.push_back("r_wrist_roll_joint");
```

Y para el brazo izquierdo:

Código Fuente – Definición articulaciones para el brazo izquierdo.

```
goall.trajectory.joint_names.push_back("l_shoulder_pan_joint");
goall.trajectory.joint_names.push_back("l_shoulder_lift_joint");
goall.trajectory.joint_names.push_back("l_upper_arm_roll_joint");
goall.trajectory.joint_names.push_back("l_elbow_flex_joint");
goall.trajectory.joint_names.push_back("l_forearm_roll_joint");
goall.trajectory.joint_names.push_back("l_wrist_flex_joint");
goall.trajectory.joint_names.push_back("l_wrist_roll_joint");
```

Cada una de las articulaciones queda claramente identificada a partir de la Figura **??**.

- shoulder_pan_joint : 1.
- shoulder_lift_joint : 2.
- upper_arm_roll_joint : 3.
- forearm_roll_joint : 4.
- elbow_flex_joint : 5.
- wrist_flex_joint : 6.
- wrist_roll_joint : 7.

Es así que cada movimiento que se muestra en la Tabla 3.1 fue diseñado definiendo cada ángulo de las articulaciones involucradas en ambos brazos.

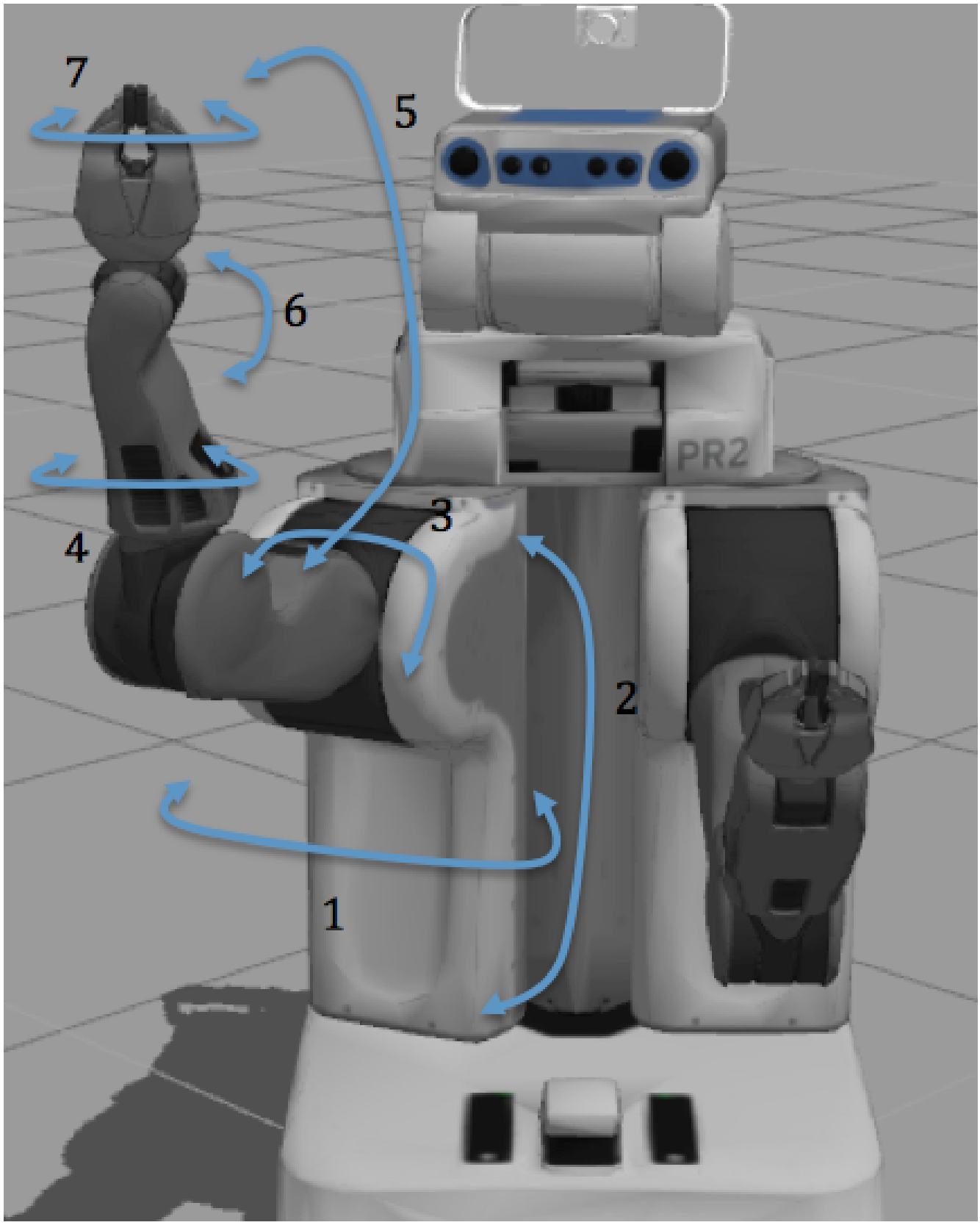

Figura 3.7: Articulaciones brazo robot PR2

3.2.3.2. Clase RobotHead y RobotTorso

La clase RobotHead y RobotTorso están escritas en `C++` y hacen uso de las librerías de `actionlib`, específicamente `PointHeadAction` para el control de la cabeza y en el control del torso `SingleJointPositionAction` .

Lo central del clase RobotHead corresponde a las siguientes instrucciones:

Código Fuente – Punto cartesiano en donde apuntará la cabeza del Robot PR2.

```
#############################################################
pr2_controllers_msgs::PointHeadGoal goal;
geometry_msgs::PointStamped point;
point.header.frame_id = "base_link";
#############################################################
point.point.x = 5;
point.point.y = 0;
point.point.z = 1.2;
goal.target = point;
```

Básicamente se define una coordenada en cartesianas $(x, y, z) = (5, 0, 1{,}2)$ en donde las distancias están en metros a la cuál apuntará la cabeza del robot. Es importante recordar que el sistema de medida de todas unidades en ROS es MKS y todos los ejes coordenados corresponden a sistemas diestros (siguen la regla de la mano derecha). De forma similar se realiza el control del torso, en donde el función para levantar el torso en la clase se define como:

Código Fuente – Función levantar torso del Robot PR2.

```
void up(){
      ###########################################################
      pr2_controllers_msgs::SingleJointPositionGoal up;
      up.position = 0.195;
      up.min_duration = ros::Duration(2.0);
      up.max_velocity = 1.0;
      ###########################################################
      ROS_INFO("Sending up goal");
      torso_client_->sendGoal(up);
      torso_client_->waitForResult();
}
```

De forma analógica para bajar el torso:

Código Fuente – Función bajar torso del Robot PR2.

```
void down(){
    ////////////////////////////////////////////////////////
    pr2_controllers_msgs::SingleJointPositionGoal down;
    down.position = 0.0;
    down.min_duration = ros::Duration(2.0);
    down.max_velocity = 1.0;
    ////////////////////////////////////////////////////////
    ROS_INFO("Sending down goal");
    torso_client_->sendGoal(down);
    torso_client_->waitForResult();
}
```

Se identifican parámetros importantes en el objeto `up` y `down` como: posición; duración del movimiento; velocidad del movimiento medida en m/s. La posición mínima es 0.0 y la máxima es 2.0.

3.2.3.3. Clase Mirror-Kinect

La clase Mirror Kinect hace uso de dos librerías principalmente disponibles en ROS y de la clase RobotArm. El objetivo de esta clase es realizar una función de imitación con los datos obtenidos a través del sensor Kinect disponible en la cabeza del robot PR2, para que los brazos imiten el moviento de quien está en frente de la cámara.

Para esto se utiliza `Openni_Tracker` que permite realizar un seguimiento de las articulaciones principales mediante el sensor Kinect. Dichas articulaciones se muestran en la Figura 3.8.

En donde:

- Cabeza (*Head*).
- Cuello (*Neck*).
- Hombro (*Shoulder*).
- Codo (*Elbow*).
- Mano (*Hand*).
- Pecho (*Torso*).

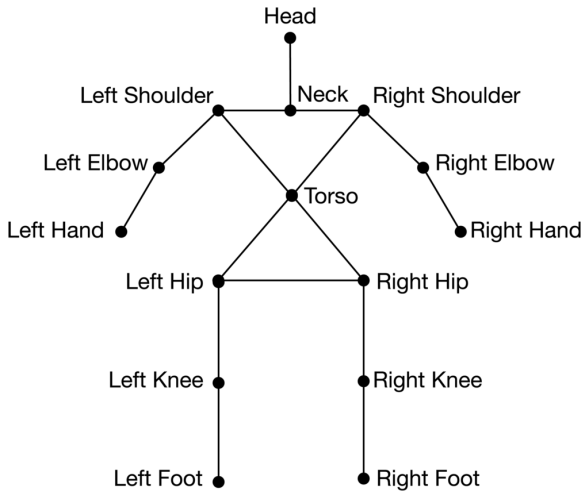

Figura 3.8: Kinect *Skeleton*.

- Cadera (*Hip*).
- Rodilla (*Knee*).
- Pie (*Foot*).

Por medio de la utilización de la librería `tf` disponible en ROS se obtiene una matriz de traslación (x, y, z) referencia al origen que es el centro del sensor Kinect. Esto permite tener la ubicación en todo momento del usuario en frente de la cámara. Con lo que al unir con la clase RobotArm es posible calcular los ángulos para:

- shoulder_pan_joint.
- shoulder_lift_joint.
- elbow_flex_joint.

Utilizando simple geometría euclidiana y traslaciones de ejes, debido a que se debe trasladar el origen coordenado al que utiliza el *Robot PR2* que está en su base, mientras que el de las mediciones provenientes del sensor se encuentran en el centro de la cámara.

3.2.4. Ejecución de rutinas

Para ejecutar las rutinas creadas en el *Robot PR2* es necesario realizar el siguiente procedimiento en un emulador de terminal para el sistema operativo Ubuntu:

1. Realizar la conexión de forma remota a PR2 a través de ssh utilizando el puerto 22, utilizando como usuario `pr2admin` y contraseña `LPTV-RyCh`.

```
ssh pr2admin@10.68.0.1
```

2. Ingresar al espacio de trabajo personal que se encuentra en el *root* de la cuenta.

```
cd ~/catkin_ws_SGD
```

3. En ocasiones es necesario volver a re-compilar el codigo fuente, para esto es necesario borrar compilaciones previas y luego compilar a través de ROS, por medio de `catkin`.

```
cd ~/catkin_ws_SGD
rm -rf /build
rm -rf /devel
catkin_make
```

4. Al terminar la compilación del código fuente, se debe hacer *source* para actualizar las variables de entorno y habilitar la ejecución de lo que se encuentra compilado en `catkin_ws_SGD` en el terminal.

```
cd ~/catkin_ws_SGD
source ./devel/setup.bash
```

5. Para ejecutar las rutinas identificando el nombre del *stack* o paquete y el nodo (ejecutable) correspondiente, de la siguiente forma:

```
rosrun <stack> <nodo>
...
rosrun sgd_motion right_arm_up
```

3.2.5. Diccionario de interacción en Robot PR2

Dado que uno de los objetivos del sistema es desarrollar un sistema que fomente la interacción humano-robot, el *Robot PR2* tiene un diccionario creado con la intención de establecer una comunicación oral con un usuario.

El diccionario creado en el *Robot PR2* contiene 40 preguntas y respuestas, y a cada una de ellas se encuentra asociado una rutina de movimiento o conjunto de rutinas de movimientos.

La estructura del diccionario es un archivo de datos tipo `csv`, que corresponde a columnas separadas por coma, como se muestra a continuación:

```
PREGUNTA,RESPUESTA
...
HOLA ROBOT,HOLA AMIGO HUMANO. QUIERES CONVERSAR CONMIGO
YARVIS LEVANTA BRAZO DERECHO,MOVIENDO MI BRAZO DERECHO
YARVIS LEVANTA BRAZO IZQUIERDO,MOVIENDO MI BRAZO IZQUIERDO
YARVIS CAMARA UNO,ACCEDIENDO A CAMARA UNO
YARVIS CAMARA DOS,ACCEDIENDO A CAMARA DOS
YARVIS TELEOPERACION,ACTIVANDO MODO TELEOPERACION
...
PREGUNTA,RESPUESTA
```

Al momento de enviar la transcripción del audio al *Robot PR2* la cadena de texto es analizada y así tomar una decisión respecto de la rutina que se ejecutará como se muestra en la Figura 3.9

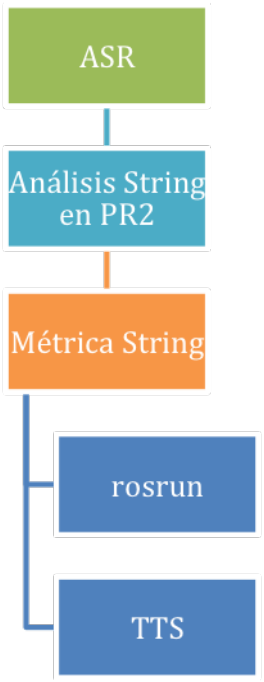

Figura 3.9: Diccionario y análisis transcripción audio.

Cuando llega cadena proveniente del ASR se procede a buscar dentro del diccionario la pregunta de mayor coincidencia con lo que proviene del ASR. Se utiliza un métrica de comparación de cadenas de texto o *string*, en donde el método implementado retorna una medida normalizada que entre $[0,1]$ de la similaridad de las cadenas de texto a comparar.

Sea T el número total de elementos en las secuencias S_1 y S_2, con M el número de aciertos, se tiene que:

$$r = 2 \cdot \frac{M}{T} \tag{3.1}$$

Si $S_1 = aabc$, $S_2 = aa$, entonces $T = 6$,$M = 2$, lo que se traduce en en un 66% de coincidencia acorde a Ecuación 3.2.

$$r = 2 \cdot \frac{M}{T} = 2 \cdot \frac{2}{6} = 0{,}66 \tag{3.2}$$

Se define un umbral de comparación fijo en 0,8 como mínimo para aceptar una coincidencia entre la pregunta que viene del ASR y la que esta en el diccionario en el *Robot PR2*, siempre se buscará por la comparación que presente la mayor coincidencia.

Cada par ordenado de pregunta y respuesta tiene asociado un BASH *script* con las instrucciones para ejecutar el movimiento adecuado a la pregunta y para utilizar Festival con el fin de que la respuesta presente en el diccionario se sintetizada y reproducida por los parlantes del *Robot PR2*, tal como se observa en la Figura 3.9.

3.2.6. Ciclo funcionamiento sistema

El ciclo de la generalidad del funcionamiento del sistema se muestra en la Figura 3.10

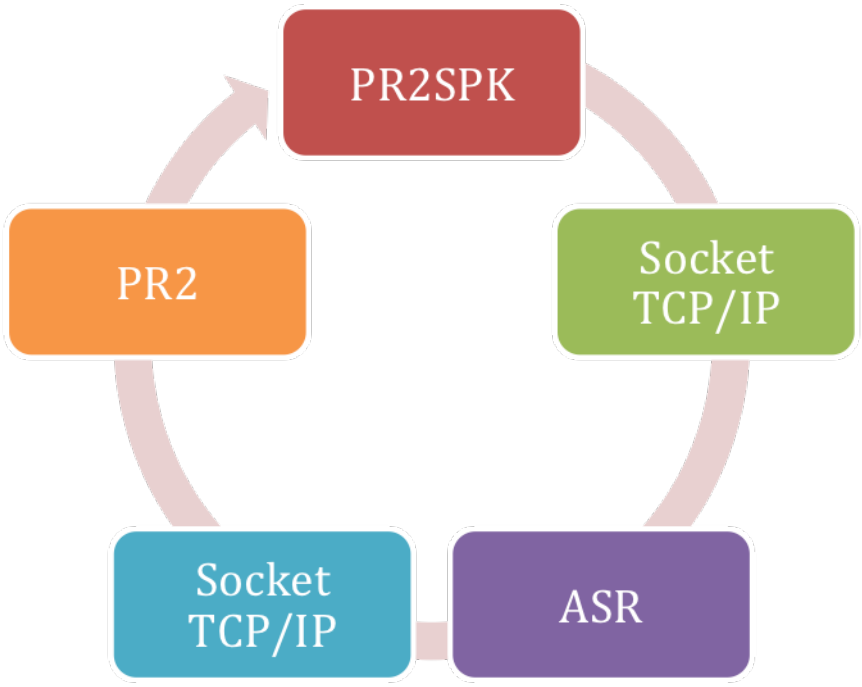

Figura 3.10: Ciclo de funcionamiento sistema.

Todo comienza al ejecutar la aplicación **PR2SPK**, aplicación WEB que permite la interacción con el usuario y la grabación del audio con el comando que reconocerá PR2, luego de que el audio haya sido exitosamente grabado se envía una señal a través de un *socket* TCP/IP al servidor ASR para realizar el análisis de la elocución grabada a 16 kHz y 16 bit. Cuando el proceso de análisis termina, a través de otro *socket* TCP/IP se envía la cadena de texto correspondiente al resultado del ASR. Ésta cadena de texto es procesada en PR2 en donde se busca asociar la máxima coincidencia al diccionario para obtener la respuesta y la ejecución de movimientos correspondientes al requerimiento inicial en la aplicación **PR2SPK**.

Capítulo 4

Resultados

En las siguientes secciones se evaluará el desempeño de los dos bloques centrales de la aplicación: ASR y movimientos del *Robot PR2*, enfocándose en como influye su desempeño en tiempo real al ejecutar la aplicación en diferentes contextos.

4.1. Evaluación del ASR

A continuación se evalúa la calidad del ASR realizando un énfasis en como afectan los ambientes ruidosos, cambios en los micrófonos y la influencia en el tiempo de repuesta en el caso de utilizar un sistema de ASR distribuido.

4.1.1. Influencia en cambios del micrófono de captura de audio

La influencia en los cambios de micrófono es muy importante en el análisis de la calidad del ASR, puesto que es el instrumento que realiza la transducción de la señal de voz, además se utiliza como supuesto en el entrenamiento del sistema, que el modelo de canal considera las perturbaciones ambientales y el micrófono en un conjunto.

Para esta prueba se utiliza la métrica del WER para evaluar la calidad del ASR, que es el error de palabras en una oración, que considera en el cálculo de todas las sustituciones, eliminación e inserciones de palabras.

Considerando las 40 frases que se grabaron como anexo a la base de datos "Latino-4k"(*ver Apéndice A y B*), se realizó el test utilizando diferentes micrófonos lo que muestra el siguiente resultado en la Figura 4.1.

En donde cada micrófono corresponde a:

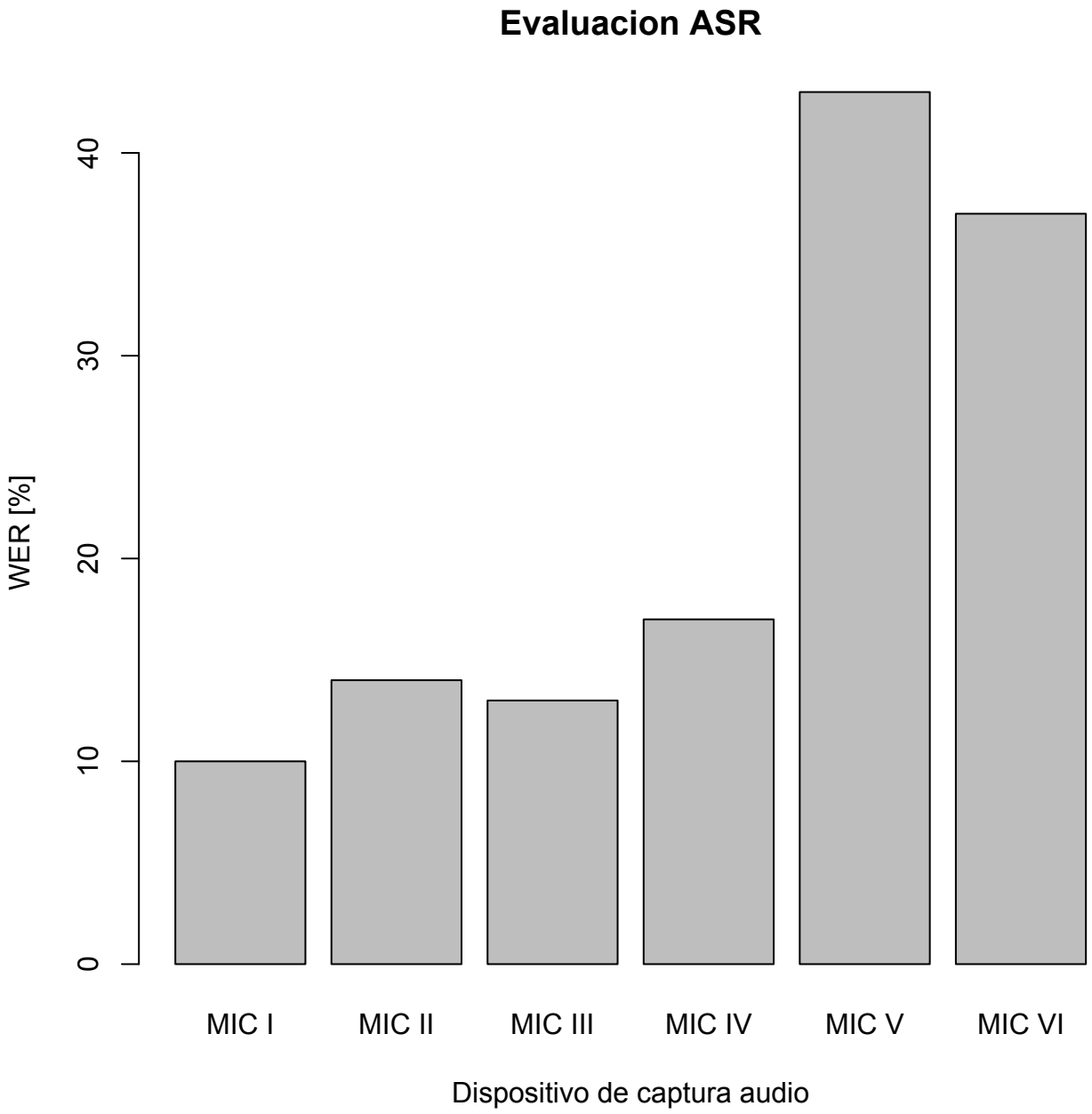

Figura 4.1: Gráfico que muestra el efecto del cambio de micrófono en la calidad del ASR.

- MIC I : micrófono utilizado para la creación de la base de datos anexa a "Latino-4k".
- MIC II : micrófono de *laptop* HP ProBook 440 G2.
- MIC III : micrófono de *laptop* MacBook Pro 2012.
- MIC VI : micrófono de *headset* Maxell.
- MIC V : micrófono de *smartphone* iPhone5.

- MIC VI : micrófono de *smartphone* Samsung Galaxy A5.

Era lógico esperar que el mejor resultado se obtiene con la utilización del micrófono utilizado en la creación de la base de datos. Sin embargo ambos *latpop* presentan una *performance* similar, lo que indica que en cuánto a *hardware* son similares o bien su construcción es similar. Sorprendente resulta el hecho de que ambos *smartphone* influyan en la calidad del reconocimiento, esto puede ser debido a múltiples razones de conexión, ya que el audio es grabado remotamente en el *smartphone* y dichas muestras son enviadas al servidor ASR a través de la red, considerando el estado y eventual pérdida de paquetes.

4.1.2. Influencia de la red

En el caso de utilizar un sistema distribuido para la ejecución de la aplicación, es decir, que el servidor ASR y el servidor WEB-PHP no se encuentren físicamente en el mismo *host* induce un aumento en el tiempo del proceso de obtener la transcripción a partir de la elocución grabada.

El experimento se realizó en la Facultad de Ciencias Físicas y Matemáticas en el Laboratorio de Procesamiento y Transmisión de Voz del departamento de Ingeniería Eléctrica, lo que mostró el siguiente resultado para los distintos casos en la figura 4.2.

Se utilizó un audio grabado mediante la aplicación de una duración de 6 s a una frecuencia de 16 kHz y 16 bit analizándose cuatro casos de estudio considerando un sistema de red distribuido, entre ellos:

- Caso I : corresponde a ambos servidores físicamente instalados en el mismo *host*.
- Caso II : corresponde al servidor ASR y servidor WEB ambos pertenecientes al Laboratorio de Transmisión y Procesamiento de Voz del departamento de Ingeniería Eléctrica.
- Caso III : corresponde a una prueba en donde el servidor WEB se encuentra instalado en un *laptop* conectado a través de una VPN a la red de la Facultad de Ciencias Físicas y Matemáticas.
- Caso IV : corresponde a una prueba en donde tanto el servidor WEB y el servidor ASR están conectados a la red de la Facultad de Ciencias Físicas y Matemáticas a través de una VPN.

El resultado de esta prueba es totalmente esperable, lo mejor se obtiene mientras ambos servidores se encuentren físicamente en el mismo *host*, ya que al mantenerlos separados

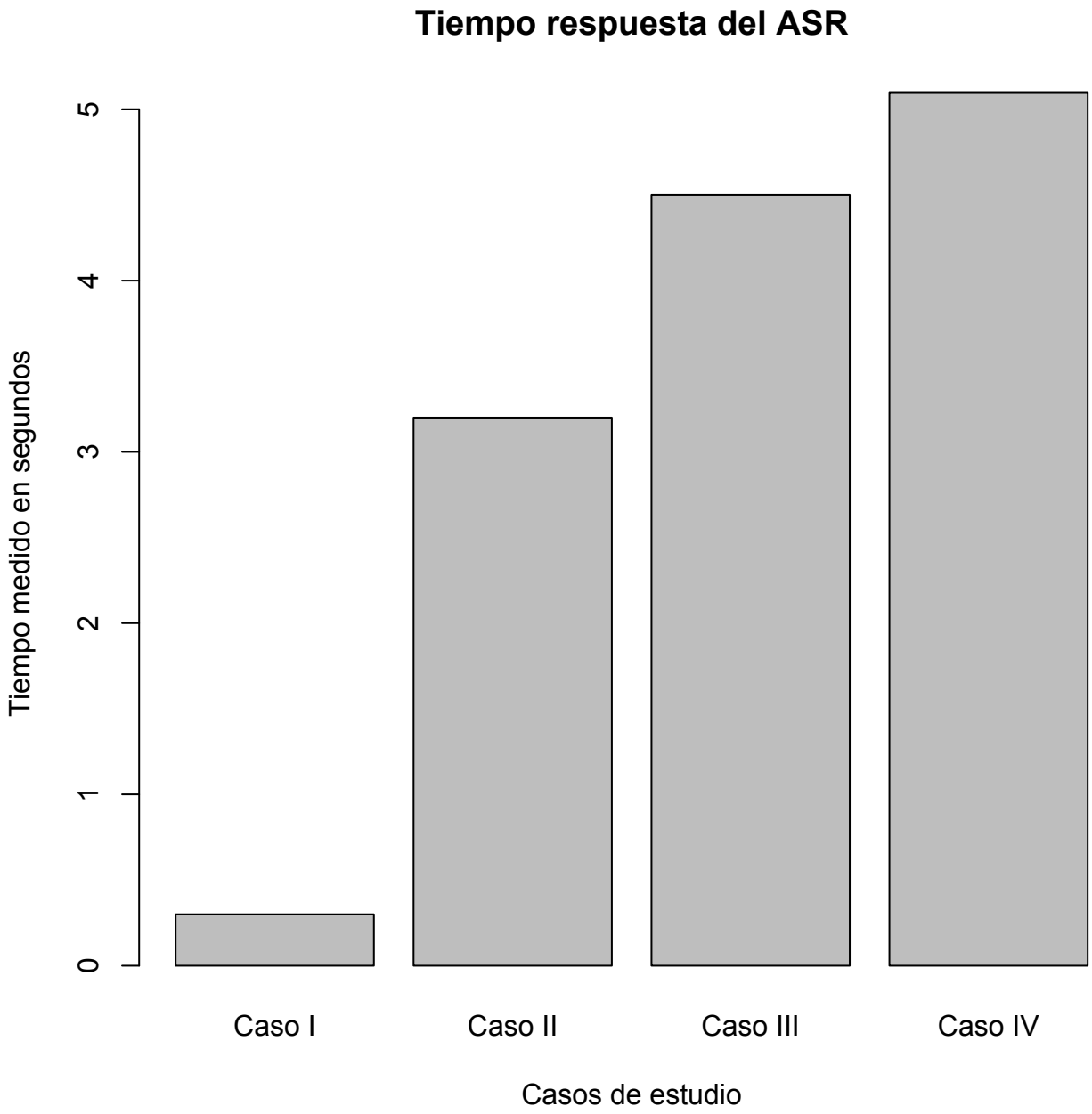

Figura 4.2: Gráfico que muestra el tiempo de respuesta del ASR al utilizar un sistema distribuido.

existe un tiempo en la descarga y subida de archivos a los distintos servidores para su posterior análisis, considerando además que el uso de la red universitaria es alto y la congestión en horarios punta ocasiona latencia desde que se graba el archivo con la aplicación hasta que se obtiene la respuesta del ASR.

4.2. Evaluación de Movimientos del Robot PR2

A continuación se realiza una evaluación de los movimientos diseñados al ejecutar la aplicación en su conjunto como se explicó en la Sección 3.2.6.

4.2.1. Calculo de coordenadas

El cálculo de las coordenadas para los accionamientos de los brazos y cabeza, es una medida que se calcula en tiempo real. El sensor Kinect a través de ROS está configurado para publicar las posiciones de las articulaciones a una frecuencia de 10 hz, como se muestra en la Figura 4.3 y **??**.

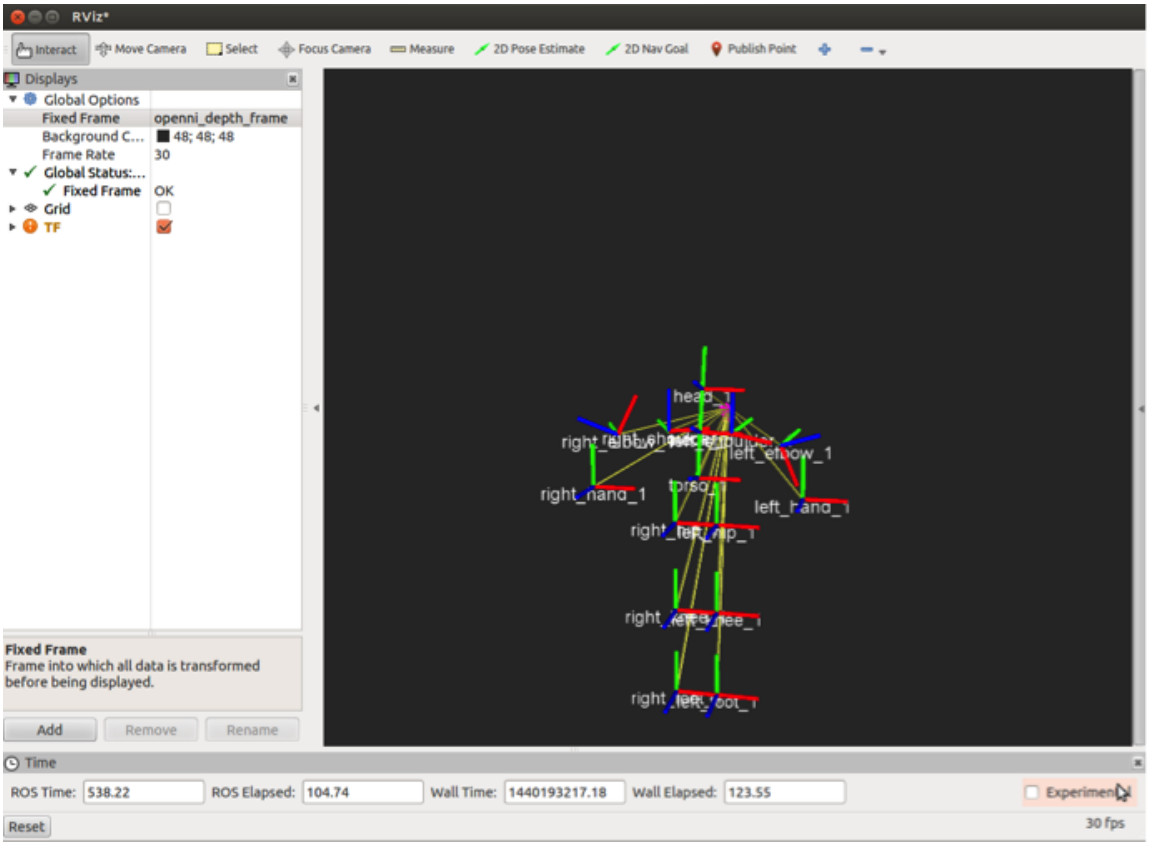

Figura 4.3: Articulaciones desde Kinect en rviz.

Esto se traduce en que cada muestra de las posiciones de las articulaciones llega cada 0.1 s. Favorablemente el calculo de las coordenadas del hombro y codo son funciones trigonométricas que no presentan mayor carga al calculo del procesador del *Robot PR2*. Sin embargo el ajuste de dicha frecuencia de publicación de datos fue principalmente a través de ensayo y error, ya que como es lógico una frecuencia muy baja en la publicación de las posiciones se traduce en movimientos poco naturales de los brazos debido a la distancia temporal entre muestras, caso opuesto ocurre cuando se aumenta sobre los 10 hz, que al llegar

una rápidamente una gran cantidad de muestras los accionamientos mecánicos del robot no son capaces de realizar los movimientos. Un punto importante ocurre a favor del aumento de frecuencia es que es posible detectar pequeñas variaciones de movimientos del cuerpo del usuario que se encuentra frente al sensor Kinect.

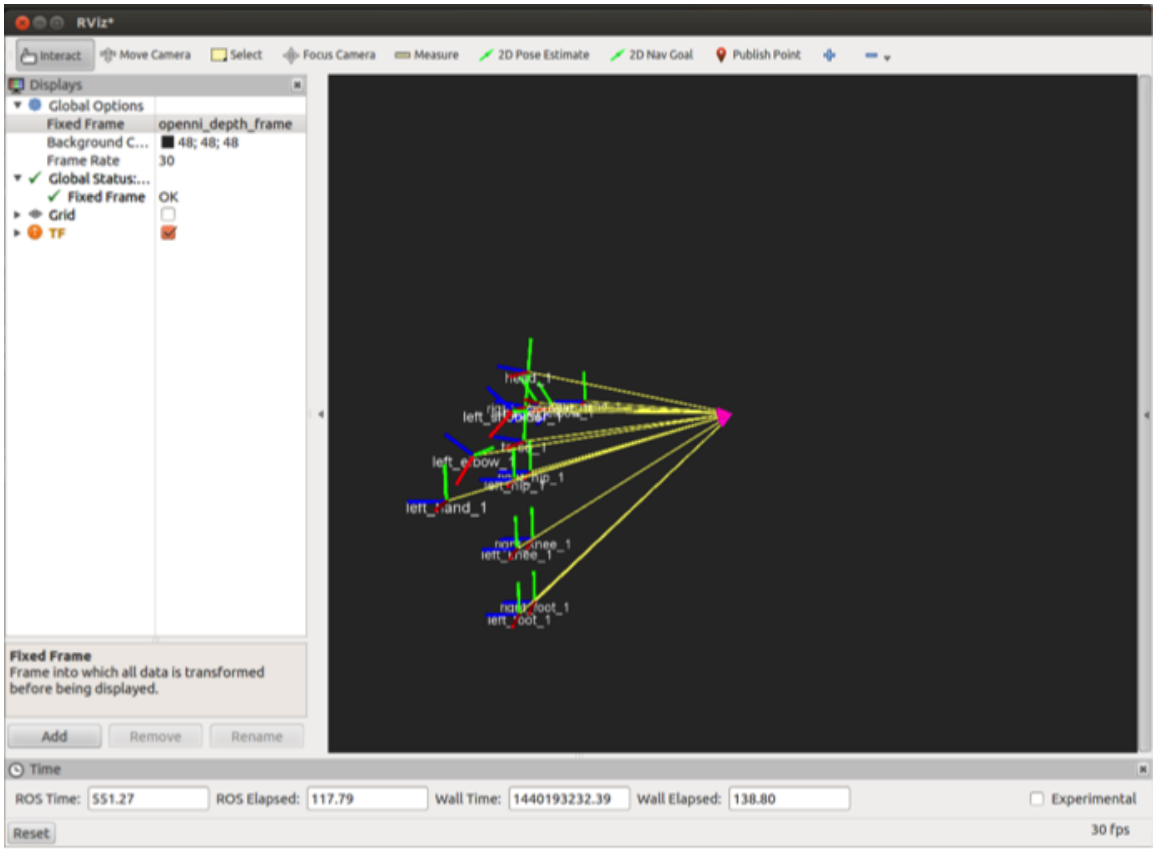

Figura 4.4: Origen sistema coordenado articulaciones desde Kinect en rviz

Si se observa detenidamente la Figura **??**, que corresponde a una visualización en tiempo real de la publicación de las coordenadas del sensor Kinect a través de `Openni_tracker` de ROS, todas las medidas se encuentran con referencia a un origen común (el punto de color magenta). Acá también fue necesario realizar un ajuste en la traslación de las coordenadas, para que estuviera con referencia al origen coordenado en el *Robot PR2* y no desde el sensor Kinect.

El efecto de mover sin cuidado el origen, es nocivo pues influye de manera negativa en el calculo de los ángulos, pues quedan con referencia a un origen que no corresponde y por lo tanto el movimiento de los brazos en la función *Mirror* se vuelve impredecible. Dado que las medidas se encuentran en sistema MKS, se realizó una traslación $(x, y, z) = (0, 0, -h)$, en donde $h = 1{,}2$ corresponde a la altura en metros del *Robot PR2* sin extender el torso.

4.2.2. Accionamientos de rutinas en Robot PR2

Por motivos prácticos se presentaran imágenes correspondientes de las rutinas más representativas del total programado que se muestra en la Tabla 3.1, 3.2, 3.3 y 3.4.

Figura 4.5: Rutina *arm init* de RobotArm.

Figura 4.6: Rutina *conversation* de RobotArm.

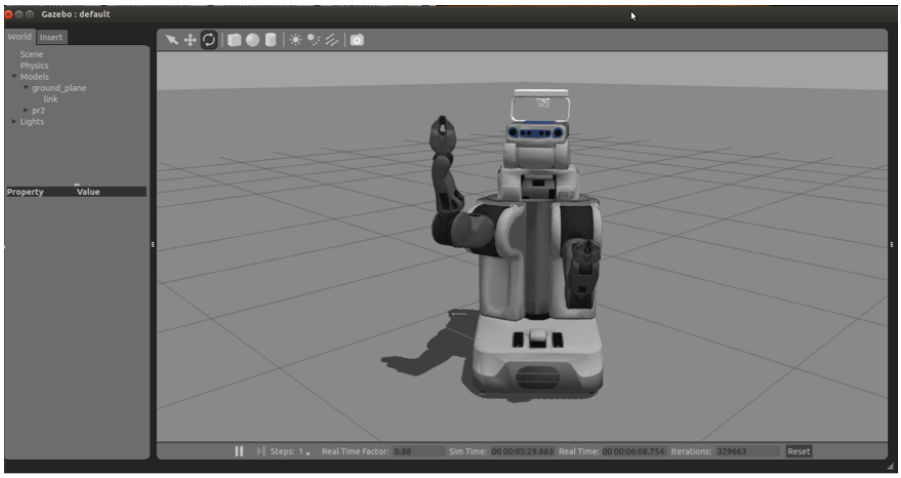

Figura 4.7: Rutina *left arm up* de RobotArm.

Figura 4.8: Rutina *gripper pound* de RobotArm.

Figura 4.9: Rutina *init* de RobotArm.

Figura 4.10: Rutina *IronMan* de RobotArm.

Figura 4.11: Rutina *Torso* de RobotArm.

Figura 4.12: Rutina *Mirror* de Mirror-Kinect.

Figura 4.13: Rutina *Mirror* de Mirror-Kinect.

A continuación se muestra un detalle de los tiempo de ejecución de cada una de las rutinas que se muestra en cada una de las imágenes anteriores:

- Tiempo de ejecución de la rutina Figura 4.5 : 1.5 s.
- Tiempo de ejecución de la rutina Figura 4.6 : 3.0 s.
- Tiempo de ejecución de la rutina Figura 4.7 : 2.0 s.
- Tiempo de ejecución de la rutina Figura 4.8 : 2.5 s.
- Tiempo de ejecución de la rutina Figura 4.9 : 1.2 s.
- Tiempo de ejecución de la rutina Figura 4.10 : 2.2 s.
- Tiempo de ejecución de la rutina Figura 4.11 : 5 s.

Considerando los tiempo de ejecución existe otro efecto contraproducente al momento de iniciar un accionamiento, dado que existe un tiempo definido para completar la trayectoria de movimientos de los brazos, una posición inicial distinta al de la Figura 4.9 o bien muy alejada de ella como en la Figura 4.10 induce un mayor tiempo en completar el movimiento pues la trayectoria a recorrer es mas larga, debido a este motivo, luego de realizar cualquier tipo de accionamiento siempre se vuelve a la posición inicial en los brazos.

Finalmente cuando se lanzan simultáneamente dos rutinas de accionamientos del cuerpo del *Robot PR2*, se establece una cola de prioridad en donde las rutinas van siendo ejecutadas en el orden en que van llegando las instrucciones al `ActionServer`.

Capítulo 5

Conclusiones

A continuación se presentan la conclusiones itemizadas del trabajo de investigación e implementación:

1. A partir de la investigación y el desarrollo de la aplicación se concluye de que forma exitosa es posible comandar el *Robot PR2* mediante la interfaz de interacción con el usuario.

2. Utilizando un modo de conexión local para los servidores: ASR y WEB, se cumple el objetivo de realizar procesamiento de la señal de audio en tiempo real.

3. La interfaz WEB diseñada utilizando HTML5/CSS/JS permite un entorno amigable de interacción con el usuario, está diseñado de forma intuitiva, no se requiere ningún conocimiento previo para utilizarla.

4. Debido a la necesidad de segmentar el inicio y fin de la señal de audio, la implementación del sistema PPH funciona como metodo de control natural de error a la insersión de palabras de inicio y fin.

5. El sistema de reconocimiento de voz se realiza mediante el *toolkit* KALDI, que corresponde al estado del arte en cuanto a ASR incorporando la construcción de modelos FST con información acústica y léxica.

6. Respecto de la calidad del ASR, como era esperable el mejor resultado se obtiene cuando el sistema se prueba con el mismo micrófono con el que la base de datos ha sido grabada, obteniendo un WER de 10 %.

7. Los micrófonos de *laptop* muestran una variación de un 4 % respecto del máximo del micrófono original (con el que la base de datos de entrenamiento fue grabada), lo que genera un mayor peligro de error en el ASR.

8. Respecto de los *smartphone* el error en el ASR crece sustancialmente por lo que la calidad en el reconocimiento de las palabras empeora.

9. El diseño de movimientos cumple con el objetivo, con bajos tiempos de repuesta (alrededor de 2 s) para completar la trayectoria del movimiento, y control absoluto de los brazos, torso y cabeza.

10. A pesar de que el sensor Kinect es sensible frente a las grandes intensidades luminosas, lo que genera errores en publicación de las posiciones de las articulaciones, su funcionamiento en interiores ha sido exitoso, logrando copiar los movimientos de las articulaciones del hombro y el codo del usuario que este en frente.

11. El tiempo entre que el audio es grabado hasta que se obtiene la cadena con la transcripción, en un sistema local, es de 0.5 s que le toma a KALDI procesar la señal de audio.

5.1. Trabajo futuro

A continuación se presenta el trabajo futuro luego del desarrollo e implementación de la memoria de título:

1. Considera la realización de una base de datos independiente del hablante y género.

2. Incluir el efecto del canal de comunicación en el ASR: ambiente y micrófonos.

3. Gracias a la construcción modular y orientada a objetos complementar el diccionario de preguntas y respuestas utilizado en el ASR.

4. Similarmente agregar nuevos movimientos personalizados al *Robot PR2*.

Bibliografía

[1] L. D. King, L. D. King, and M. Suñer, *Gramática española: Análisis y práctica*. McGraw-Hill Humanities Social, 2004.

[2] R. Molineros. (2015) El aparato fonador humano. 20:15 horas, 19 de Noviembre de 2015. [Online]. Available: https://www.tes.com/lessons/KiNWkEnV1WhjZw/el-aparato-fonador

[3] M. B. F. Canal, "Acercamiento a la voz humana, su fisiología y rehabilitación," *Revista de Logopedia, Foniatría y Audiología*, vol. 33, no. 1, 2013.

[4] W. Garage. (2015) Especificaciones técnicas de robot pr2. 22:05 horas, 19 de Noviembre de 2015. [Online]. Available: https://www.willowgarage.com/pages/pr2/specs

[5] D. North. (2010) Fotos internas de sensor microsoft kinect. 18:05 horas, 13 de Junio de 2015. [Online]. Available: http://www.destructoid.com/kinect-busted-open-tech-explained-180724.phtml

[6] S.-H. Salleh, H. K. Sze, and T. Swee, "Design and development of speech-control robotic manipulator arm," in *Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on*, vol. 1, Dec 2002, pp. 459–464 vol.1.

[7] I. El-Emary, M. Fezari, and H. Abbassi, "Speech as a high level control for teleoperated manipulator arm," in *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, vol. 2, March 2010, pp. 657–662.

[8] M. Balaganesh, E. Logashanmugam, C. Aadhitya, and R. Manikandan, "Robotic arm showing writing skills by speech recognition," in *Emerging Trends in Robotics and Communication Technologies (INTERACT), 2010 International Conference on*, Dec 2010, pp. 12–15.

[9] T. Tremblay and T. Padir, "Modular robot arm design for physical human-robot interaction," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, Oct 2013, pp. 4482–4487.

[10] J. Bilmes, J. Malkin, X. Li, S. Harada, K. Kilanski, K. Kirchhoff, R. Wright, A. Subramanya, J. Landay, P. Dowden, and H. Chizeck, "The vocal joystick," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 1, May 2006, pp. I–I.

[11] S. Cremer, I. Ranatunga, and D. Popa, "Robotic waiter with physical co-manipulation capabilities," in *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, Aug 2014, pp. 1153–1158.

[12] J. Bohren, R. Rusu, E. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mosenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the pr2," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 5568–5575.

[13] D. Nyga, F. Balint-Benczedi, and M. Beetz, "Pr2 looking at things; ensemble learning for unstructured information processing with markov logic networks," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 3916–3923.

[14] J. Mainprice, M. Gharbi, T. Simeon, and R. Alami, "Sharing effort in planning human-robot handover tasks," in *RO-MAN, 2012 IEEE*, Sept 2012, pp. 764–770.

[15] S. Sadowsky and G. F. S. Gutiérrez, "El inventario fonético del español de chile: principios orientadores, inventario provisorio de consonantes y sistema de representación (afi-cl)," *Onomázein: Revista de lingüística, filología y traducción de la Pontificia Universidad Católica de Chile*, no. 24, pp. 61–84, 2011.

[16] H. E. Pérez, "Frecuencia de fonemas1," 2003.

[17] J. Benesty, *Springer handbook of speech processing*. Springer Science & Business Media, 2008.

[18] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[19] L. R. Rabiner, B.-H. Juang, S. Levinson, and M. Sondhi, "Recognition of isolated digits using hidden markov models with continuous mixture densities," *AT&T technical journal*, vol. 64, no. 6, pp. 1211–1234, 1985.

[20] J. Picone, "Signal modeling techniques in speech recognition," *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215–1247, Sep 1993.

[21] M. Hossan, S. Memon, and M. Gregory, "A novel approach for mfcc feature extraction," in *Signal Processing and Communication Systems (ICSPCS), 2010 4th International Conference on*, Dec 2010, pp. 1–5.

[22] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, no. 4, pp. 357–366, 1980.

[23] S. Konstantinidis, "Computing the levenshtein distance of a regular language," in *Information Theory Workshop, 2005 IEEE*, Aug 2005, pp. 4 pp.–.

[24] Z. Zhang, "Microsoft kinect sensor and its effect," *MultiMedia, IEEE*, vol. 19, no. 2, pp. 4–10, 2012.

[25] B. Guo, J. Sun, Y.-C. Wei, C. Meekhof, and T. Leyvand, "Kinect identity: Technology and experience," *Computer*, no. 4, pp. 94–96, 2011.

[26] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *Cybernetics, IEEE Transactions on*, vol. 43, no. 5, pp. 1318–1334, 2013.

[27] L. Xia, C.-C. Chen, and J. K. Aggarwal, "Human detection using depth information by kinect," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*. IEEE, 2011, pp. 15–22.

[28] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[29] N. Villaroman, D. Rowe, and B. Swan, "Teaching natural user interaction using openni and the microsoft kinect sensor," in *Proceedings of the 2011 conference on Information technology education*. ACM, 2011, pp. 227–232.

[30] L. Rabiner and B.-H. Juang, "Fundamentals of speech recognition," 1993.

[31] M. T. V. de Ramírez, *El español de América: Pronunciación*. Arco libros, 1996, vol. 1.

Glosario

ASR *Automatic Speech Recognition*.

CSS *Cascading Style Sheets*. Es un lenguaje de programación usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML.

FST *Finite State Transducer*.

HMM *Hidden Markov Model*.

HTML5 *HyperText Markup Language* versión 5. Lenguaje de programación no compilado, que se ejecuta en el cliente enfocado al diseño de páginas WEB y todos sus elementos que lo contienen.

IP Etiqueta numérica que identifica, de manera lógica y jerárquica, a una interfaz de un dispositivo dentro de una red que utilice el protocolo IP.

JS *JavaScript*. Es un lenguaje de programación interpretado, orientado a objetos y se utiliza principalmente en el lado del cliente , implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

KALDI Conjunto de herramientas para reconocimiento de voz. Escrito en **C++**, bajo licencia APACHE v2.0.

MFCC *Mel Frecuency Cepstral Coeficient*.

PHP *PHP Hypertext Preprocessor*. Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico..

PPH Pulsar Para Hablar.

PR2 *Personal Robo*t 2.

ROS *Robot Operating System*.

TTS *Text to Speech*.

WER *Word Error Rate*.

Apéndice A

Listado Preguntas

1	YARVIS BAILA CONMIGO
2	YARVIS PUEDES BAILAR
3	YARVIS LEVANTA BRAZO DERECHO
4	YARVIS LEVANTA BRAZO IZQUIERDO
5	YARVIS CAMARA UNO
6	YARVIS CAMARA DOS
7	YARVIS CUENTA UN CHISTE
8	YARVIS SABES OTRO CHISTE
9	YARVIS QUE TE GUSTA HACER
10	YARVIS COMO LLEGASTE ACA
11	YARVIS CUANDO LLEGASTE A CHILE
12	YARVIS QUE TE GUSTA COMER
13	YARVIS A QUE TE DEDICAS
14	YARVIS QUE EQUIPO TE GUSTA
15	YARVIS QUE CANCION TE GUSTA
16	YARVIS QUE MUSICA TE GUSTA
17	YARVIS COMO TE MUEVES
18	YARVIS QUE PELICULA TE GUSTA
19	COMO TE LLAMAS
20	YARVIS QUIENES SON TUS PADRES
21	YARVIS DONDE NACISTE
22	YARVIS QUE EDAD TIENES
23	YARVIS DONDE TRABAJAS
24	YARVIS DE DONDE VIENES
25	YARVIS COMO VIAJASTE HASTA ACA
26	HOLA COMO ESTAS

27	YARVIS SALUDA
28	YARVIS DEFENSA
29	YARVIS TELEOPERACION
30	YARVIS PUEDES HACER ALGO DIVERTIDO
31	HOLA ROBOT
32	YARVIS MANIQUI
33	PE ERRE DOS
34	YARVIS PRESENTATE
35	YARVIS PRESENTATE EN INGLES
36	YARVIS RECALIBRATE
37	YARVIS TIENES ALGUN HOBBY
38	YARVIS TE GUSTA LEER
39	YARVIS TE GUSTA VER TELEVISION
40	YARVIS TE GUSTA EL FUTBOL
41	YARVIS TIENES POLOLA
42	YARVIS TIENES SENTIMIENTOS
43	YARVIS PUEDES LEER
44	YARVIS CONOCES ALGUN SUPERHEROE

Apéndice B

Listado Respuestas

1	ME ENCANTA BAILAR CON BUENA MUSICA
2	ME GUSTA MUCHO BAILAR Y CON BUENA MUSICA
3	MOVIENDO MI BRAZO DERECHO
4	MOVIENDO MI BRAZO IZQUIERDO
5	ACCEDIENDO A CAMARA UNO
6	ACCEDIENDO A CAMARA DOS
7	JA JA. HABIA UNA VEZ UNA OLLA QUE NO QUERIA SER OLLA Y FUE OLLA A PRESION
8	MAMA EN EL COLEGIO ME DICEN CAMPANA. AY TATAN TU TAN TONTIN
9	DE TODO. SOBRE TODO COSAS DE ROBOT
10	LLEGUE CON MIS PIES JA JA JA
11	LLEGUE EN MARZO DEL AGNO DOS MIL CATORCE
12	ME ALIMENTO DE ELECTRICIDAD A DOSCIENTOS VEINTE VOLTS
13	ME DEDICO A APRENDER DE LOS HUMANOS Y SU ENTORNO
14	ME GUSTA EL EQUIPO DE FUTBOL DEL LABORATORIO EN EL QUE TRABAJO
15	ME GUSTA LA MUSICA ELECTRONICA DE DAFT PANK
16	ME GUSTAN TODOS LOS ESTILO DE MUSICA SOBRE TODO EL ROC Y LA ELECTRONICA
17	ME TRASLADO CON MIS CUATRO PIES QUE TENGO EN LA BASE
18	MI FAVORITA ES YO ROBOT. JA JA JA
19	MI NOMBRE ES YARVIS
20	MIS PADRES SON UNAS MAQUINAS DE LA FABRICA EN DONDE ME CREARON
21	NACI EN UNA FABRICA EN ESTADOS UNIDOS
22	TENGO UN AGNO
23	TRABAJO EN EL LABORATORIO DE PROCESAMIENTO Y TRANSMISION DE VOZ
24	VENGO DE WILLOW GARAGE

25	VIAJE DESDE ESTADOS UNIDOS EN BARCO A EL PUERTO DE VALPARAISO
26	YO ME ENCUENTRO MUY BIEN AMIGO HUMANO
27	HOLA AMIGO HUMANO. ESTRECHA MI MANO
28	ACTIVANDO MODO DEFENSIVO. LANZANDO COMBINACION MORTAL DE BOKCEO
29	ACTIVANDO MODO TELEOPERACION. DEBES ESTAR FRENTE A MI CAMARA PARA QUE PUEDA RECONOCERTE
30	TODO LO QUE YO HAGO ES DIVERTIDO. JA JA JA. PARATE FRENTE A MI Y SEGUIRE TU BRAZO
31	HOLA AMIGO HUMANO. QUIERES CONVERSAR CONMIGO
32	AHORA PUEDES CONTROLARME A LIBERTAD
33	MI NOMBRE NO ES PR2. ES YARVIS
34	MI NOMBRE ES YARVIS. SOY ROBOT DE SERVICIO Y ME GUSTA INTERACTUAR CON HUMANOS
35	MY NAME IS YARVIS. I AM A SERVICE AND PERSONAL ROBOT. I REALLY LIKE TALK TO A HUMAN BEEN
36	RECALIBRANDO. EN TRES DOS UNO
37	ME GUSTA APRENDER IDIOMAS Y PRACTICAR KUNG FU.
38	ME GUSTA LEER LIBROS DE PROGRAMACION. JA JA
39	ME GUSTA VER LOS PROGRAMAS DE TECNOLOGIA
40	SI SOY FANATICO DE LA SELECCION DE CHILE. LA ROJA. Y SIEMPRE SIGO LOS PARTIDOS DE LA UNIVERSIDAD DE CHILE
41	SI, EL DEL CONTROL AMARILLO ES MI POLOLA
42	SI, TENGO SENTIMIENTOS POR EL QUE MANEJA EL CONTROL AMARILLO
43	SI. PUEDO LEER. ME GUSTAN LOS LIBROS EN PE DE EFE JA JA
44	MI FAVORITO ES AIRON MAN. Y TENGO SU SUPER FUERZA