Sistema de reconocimiento de voz aislada utilizando *Arduino Micro* y *Bitvoicer Server* para control domótico simulado

Luis Carlos Rúa Sánchez

Proyecto de grado presentado como requisito parcial para optar por el título de Ingeniero Electricista

Director:

Julián David Echeverry Correa, PhD.

Universidad Tecnológica de Pereira

Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación

Programa de Ingeniería Eléctrica

Pereira

2016

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

1. INTRODUCCIÓN

El agitado ritmo de vida y los consecuentes niveles de estrés, han llevado a los seres humanos a idear y construir dispositivos capaces de simplificar, e incluso ejecutar directamente tareas domésticas, conocidos como sistemas domóticos [4]. Estos dispositivos están diseñados para operar en respuesta a órdenes de los usuarios, transmitidas a través de un canal y mediante alguna forma de comunicación.

Aunque es posible emplear múltiples canales para la transmisión de órdenes desde el usuario hacia el sistema domótico, la forma más típica es el uso de aplicativos, usualmente disponibles para *smartphones* y *tablets*. En los que mediante controles contextuales, disponibles en la pantalla del dispositivo, tiene lugar la interacción entre ambos; este tipo de control, puede estar disponible para manejo local o remoto a través de intranet o internet.

La proliferación de herramientas como el *Kinect* y el *Wiimote* ha promovido la ampliación del espectro de posibilidades, en cuanto a formas de control respecta, permitiendo operar mediante gestos corporales o incluso señas el sistema domótico. Sin embargo, entre las posibles formas conocidas a la fecha, es el lenguaje hablado, la forma más natural e intuitiva de comunicación, considerando su amplia difusión y uso alrededor del mundo, así como su fácil transmisión en el aire.

Estas características, hacen de la voz una señal idónea para transmitir información. No obstante, sólo hasta el incremento de la velocidad de cálculo de los sistemas de cómputo, la aparición de los sistemas multinúcleo, los avances en procesamiento de señales y la optimización de algoritmos como la transformada de Fourier, fue posible concebir la idea de utilizar señales como la voz para comandar dispositivos electrónicos.

Inicialmente se desarrollaron aplicaciones de escritorio principalmente pensadas para reemplazar periféricos como el mouse y el teclado, así como facilitar la navegación del usuario en el equipo, consiguiendo una interacción hombre – máquina más rápida e intuitiva que la obtenida mediante los periféricos tradicionales.

La introducción de aplicaciones de reconocimiento de habla en sistemas operativos móviles, permitió a los usuarios, controlar funciones propias de los *smartphones*, sin presuponer conocimiento de las rutas de acceso a las

funciones, recibiendo respuestas distintas para cada una de las acciones solicitadas mediante órdenes.

En este proyecto de grado, se implementa un sistema de reconocimiento de habla, robusto, escalable, de bajo costo y de fiabilidad suficiente, para responder ante comandos aislados, empleando hardware y software modernos, de fácil manejo y adquisición permitiendo el máximo grado de interoperabilidad posible, entre las tecnologías involucradas, verificando su eficiencia mediante un ambiente domótico simulado, en un navegador web.

Una ventaja de emplear tecnologías web para el desarrollo de esta tesis, es la posibilidad de operar el ambiente domótico simulado, de forma remota. Este potencial podría ser aprovechado en tesis derivadas.

2. PLANTEAMIENTO DEL PROBLEMA

Son pocos los sistemas domóticos ofertados, con el potencial de ser operados mediante voz, siendo la forma más popular de interacción entre este y el usuario, una pantalla táctil. Es así como el usuario debe memorizar secuencias de acciones, y rutas de acceso a menús contextuales, para delegar tareas al dispositivo. Este proceso resulta en una operación poco intuitiva y lenta, típicamente realizada a través de una pequeña pantalla de *smartphone*, como resultado de la popularidad que han adquirido estos dispositivos, sobre las *tablets*, dificultando la maniobrabilidad, y bloqueando cualquier acción en curso por parte del usuario, distinta, al manejo del propio dispositivo, al ocupar sus manos y su vista en dicho proceso.
Algunas empresas estadounidenses, han enfrentado este problema, mediante el diseño y elaboración de sistemas embebidos como el *VoicePod*[1] y el *Ubi*[2], pensados para capturar las órdenes emitidas, y procesarlas. En este proceso, se seleccionan las acciones que deberán realizarse, en relación a los comandos de voz detectados.

VoicePod, cuenta con un aplicativo para móviles. Este permite emplear el dispositivo donde es instalado como interfaz de captura de habla y preprocesamiento, para su envío a un servidor web dedicado. Es allí donde son identificados los comandos y enviados al *VoicePod*, encargado de distribuirlos en

[1] http://www.voicepod.com/
[2] http://www.ucic.io/

los respectivos periféricos según corresponda. El aplicativo está disponible para las plataformas *Android* y *iOS*, y puede ser descargada desde las tiendas *App Store* y *Google Play*. Estos dispositivos sólo son compatibles con sistemas de automatización profesionales como *Control4* o *Creston*, según se evidencia en la página oficial del fabricante, limitando la elección del usuario, en cuanto a los sistemas domóticos que puede utilizar. Existen otros dispositivos comerciales como el *Echo* de *Amazon*, acondicionados al procesamiento de habla natural. Este dispositivo trabaja conjuntamente con *Amazon Web Services*, permitiendo la realización de consultas online por parte del usuario. No obstante, sus funciones se limitan a la respuesta a preguntas del usuario y a la reproducción de música. Esta función es similar a la desempeñada por el software *Siri* de *Apple*, encargado de interpretar el habla del usuario y ejecutar tareas en el hardware en sus dispositivos, en relación a estos comandos.

Los productos referidos, están preparados para controlar ambientes domóticos, a través de mecanismos tan naturales para el usuario como el habla. Sin embargo, son desarrollos cerrados y netamente comerciales, impidiendo por vías legales, su explotación, adaptación y usos particulares que impliquen su modificación parcial, ya sea para extender su grado de interoperabilidad con otros periféricos, distintos a los permitidos, o la adecuación de algunas de sus funciones, que deriven en operaciones distintas a las diseñadas. El resultado es un buen sistema de reconocimiento, pero inhábil ante la interconexión de sensores y actuadores, distintos a los direccionados por el fabricante.

En este trabajo de grado, se implementa un sistema de reconocimiento de habla aislada de bajo costo, empleando el dispositivo electrónico programable *Arduino*, como puente entre los distintos periféricos y una computadora portátil. De esta manera, se consigue extender el potencial de interoperación entre diversos dispositivos electrónicos conectados al *Arduino* con la computadora. Entre sensores, actuadores y elementos de presentación.

Este trabajo de grado propone un procedimiento de integración de hardware y software, involucrando el habla humana, como elemento de mando, a través de aplicaciones y dispositivos *open source* en su mayoría, abriendo paso a Ingenieros y entusiastas, en la construcción y diseño de este tipo de sistemas comandados por voz.

3. JUSTIFICACIÓN

Los sistemas de reconocimiento de habla comerciales, están protegidos por patentes y secretos empresariales, que impiden la reproducción, adaptación y extensión de sus características, acorde a las necesidades puntuales del desarrollador o investigador, bien sea para uso académico o comercial. De este modo se limita el número de desarrollos en el área del reconocimiento de voz, aplicado al control de ambientes en la universidad, así como su adaptabilidad a sistemas domóticos distintos a los permitidos por el fabricante. Así mismo, la falta de información de calidad en esta área, y de una metodología clara, que describa la forma en que pueden construirse este tipo de sistemas, donde se involucran tecnologías y marcas tan heterogéneas, dificulta la creación de estándares en este sentido.

Esta investigación, deriva en la implementación de un sistema de reconocimiento de voz, para el control de ambientes domóticos simulados, empleando tecnologías de fácil acceso, aportando un procedimiento para la integración de las tecnologías involucradas, consiguiendo la integración entre un sistema de reconocimiento de voz y un ambiente domótico. El desarrollo resultante será en su mayoría código abierto, lo que permitirá su modificación y estudio, por parte de las personas interesadas, así como de las empresas que deseen hacer compatibles sus desarrollos domóticos con este sistema en particular.

Dentro del perfil ocupacional del Ingeniero Electricista de la Universidad Tecnológica de Pereira, se encuentra el diseño e implementación de software y hardware mediante las nuevas tecnologías para la sistematización y control de procesos, siendo estas áreas afines al problema planteado, por lo que se cuenta con las competencias para el desarrollo de una solución.

4. OBJETIVOS

A continuación se presentan los objetivos perseguidos en este trabajo

4.1. OBJETIVOS GENERAL

Implementar un sistema de reconocimiento de voz utilizando *Arduino Micro* y *Bitvoicer Server* para el control de un ambiente domótico simulado mediante una aplicación en *JavaScript* y *HTML5* y *CSS3*.

4.2. OBJETIVOS ESPECÍFICOS

- Desarrollar una herramienta basada en *Arduino Micro* para la implementación de un sistema de reconocimiento automático de voz

- Programar el Arduino para operar conjuntamente con el *Bitvoicer Server* y la interfaz gráfica

- Desarrollar una interfaz gráfica que simule un entorno domótico donde se puedan validar los resultados del sistema de reconocimiento de voz

5. MARCO DE REFERENCIA

Este trabajo de grado involucra una plétora de tecnologías muy diversas, por lo que se vio conveniente, organizar cada uno de sus marcos de referencia en forma separada, para luego unirlos con un marco común al conjunto.

5.1. MARCO DE REFERENCIA MICROCONTROLADORES

En esta sección se describe brevemente la historia, teoría y principales conceptos, en los que se involucran desde etapas tempranas del desarrollo de los microcontroladores, hasta acercarse a lo que hoy conocemos como *Arduino Micro*. Esta mención a los microcontroladores es pertinente, debido a que en este trabajo de grado, se emplea un microcontrolador con conexión *USB*, para realizar la captura y digitalización de una señal de habla, para su procesamiento e

interpretación en una computadora conectada a través del puerto *USB*, en apoyo al microcontrolador.

5.1.1. MARCO HISTÓRICO

En 1969, un equipo de ingenieros japoneses de la compañía *BUSICOM*, llegó a Estados Unidos para solicitar a Intel, diseñar un conjunto de circuitos integrados para una nueva línea de calculadoras electrónicas programables.
De esta manera, se estimuló la invención del primer microprocesador para ser comercializado. Sin embargo, este tipo de circuito integrado, no incluía memorias, ni periféricos de entrada y salida en el mismo chip, teniendo que ser adicionados externamente, aumentando el área ocupada.

No pasó mucho tiempo para que Gary Boone de *Texas Instruments*, experto en el desarrollo de calculadoras, desarrollara en 1971, en un circuito integrado de un solo chip, que pudiera contener los circuitos esenciales para formar una calculadora; el *display* y el teclado, no fueron incorporados. Este desarrollo se conoce como el primer microcontrolador, y recibió el nombre de TMS1802NC [1].

Este proceso, fue continuado por *Intel*, construyendo en 1976 el MCS – 48, ampliando el espectro de uso de estos dispositivos a televisores, controles remotos, juguetes, entre etc.

En 1977, *Intel* empezó a comercializar un microcontrolador, especialmente pensado para aplicaciones de control, el *Intel* 8048, provisto de una memoria *RAM* y *ROM* en el mismo chip. Este chip fue ampliamente utilizado en teclados compatibles con *IBM PC*.

En 1993, se lanzan al mercado los primeros microcontroladores con memoria *EEPROM*, iniciando con el microcontrolador *PIC16F84*, permitiendo sobrescribir el contenido de su memoria sin necesidad de contar costosos equipos. Ese mismo año *Atmel*, lanza el primer microcontrolador con memoria *flash*.

Estos avances, acercaron a entusiastas y conocedores de Electrónica y Computación, a desarrollos más rápidos y flexibles. Sin embargo, sólo hasta 2005, que personas ajenas a este campo, tuvieron la posibilidad de desarrollar proyectos microcontrolados, con la introducción de *The Wiring Microncontroller*, y proyectos derivados como *Arduino*.

Es así como ese mismo año, nace *Arduino*, una placa constituida por un microcontrolador marca *Atmel*, integrada con una conexión a puertos *USB*¸ y facilidad de acceso a sus terminales. Con características como la usabilidad y la simplicidad, permitiendo a no programadores usarlo.

Según *Máximo Banzi*, cofundador de *Arduino Inc.,* son 4 las razones del éxito de *Arduino*.

1. No es una plataforma costosa
2. Cuenta con una interfaz de desarrollo propia
3. Es programable a través del puerto *USB*
4. Cuenta con una amplia comunidad y soporte

El éxito de *Arduino*, ha llevado a que se contruyan muchas variantes y subvariantes de la versión original, donde cada una cuenta con distintas características de hardware, que van desde el número de bits de la arquitectura, hasta la posibilidad de conectarse a internet [2].

5.1.2. MARCO TEÓRICO

Un microcontrolador es un circuito integrado programable, preparado para ejecutar las instrucciones grabadas en su memoria *ROM*. Está compuesto de varios módulos, encargados de labores específicas. Estos integran una unidad lógica aritmética, de capacidad mucho inferior al de un microprocesador, una memoria *RAM*, y una memoria *ROM* y periféricos de entrada y salida.

La mayor parte de los microcontroladores emplean palabras de 8 bits y funcionan a velocidad de reloj con frecuencias no superiores a los 48MHz. Los microcontroladores, al igual que los microprocesadores, cuentan con fuentes de interrupción, es decir, eventos ante los que debe responder, en el momento en que se presenten, cuando dicha fuente se encuentre habilitada por software. Algunas fuentes de interrupción son: cambio de estado de una entrada, un temporizador interno, el desbordamiento de algún registro de propósito especial, etc.

Para programar un microcontrolador, se debe lograr grabar en su memoria interna, un código hexadecimal, respetando el tamaño de su palabra, las instrucciones disponibles, así como el espacio con el que cuenta la memoria interna. Esta programación normalmente se realiza mediante pines específicos del

microcontrolador, típicamente 5, 2 de polarización, el *RESET*, el *CLOCK*, y el recibo de datos desde el programador.

El diseño del programa a cargar al microcontrolador, normalmente se realiza en una computadora de escritorio, en un software conocido como compilador, encargado de procesar las instrucciones, ingresadas por el usuario, y convertirlas en un código hexadecimal útil, para su transferencia al dispositivo. Esta transferencia se realiza comúnmente a través del puerto serie de la computadora, o a través de un conversor *USB/*Serie. El programa, normalmente está estructurado en varias filas, donde cada línea, inicia con un símbolo, típicamente ':', aunque puede variar según la tecnología, seguido de un código hexadecimal donde se condensan, la cantidad de datos de la línea, la dirección de 16 *bits* del primer registro en la *ROM*, el tipo de registro, los datos, y un *checksum*, obtenido como el complemento a 2 de todos los datos de la línea, salvo el ':'.

El formato citado en el párrafo anterior, se conoce como *INTEL HEX*, y presenta 3 sub-formatos principalmente:

1. *INTEL* 8
2. *INTEL* 16
3. *INTEL* 32

A continuación se presenta la Tabla 1, donde se explica la forma correcta de interpretar el formato *INTEL* 8.

RECORD MARK	RECLEN	LOAD OFFSET	RECTYP	DATA	CHKSUM
1 - byte	1 - byte	2 – byte	1 - byte	n - byte	1 - byte

Tabla 1. FORMATO HEXADECIMAL INTEL 8

Así mismo se presenta un ejemplo de interpretación del formato INTEL 8, tomando como referencia este formato[3], y un código hexadecimal arbitrario. Cada línea además, cuenta con un pequeño comentario, expresado después del punto y coma, como suele hacerse en este formato.

:03800000028100FA; En esta línea se encuentran 3 *bytes* de información

[3] http://microsym.com/editor/assets/intelhex.pdf, página 8

:00000001FF; Esta línea, demarca el fin del código hexadecimal

En la primera fila, encontramos el primer *byte*, que nos indica la cantidad de datos que encontraremos en la fila, en este caso 3, correspondientes a 02 81 00. Los siguientes 2 *bytes*, tal como lo indica el formato, corresponden a una dirección de memoria de 16 *bits*, 80 00. El siguiente *byte*, corresponde al formato, particularmente 00, este indica que existen datos en la fila. Si la línea contiene 01 en este *byte* como el caso de la siguiente línea, no contiene datos y corresponde a la última. El último *byte*, en cada fila, corresponde al *checksum*, calculado como el complemento a 2 de la suma de los números de la fila. En caso de que alguna de las sumas genere acarreo, más allá del *byte*, este debe ser descartado antes de obtener el complemento. Así se verifica fácilmente que la fila uno tiene un *checksum* FA, y la 2 FF.

Son pocas las ocasiones, en que un programador, debe enfrentarse al código hexadecimal, debido a que programas conocidos como compiladores, se encargan de convertir comandos de más fácil aprendizaje y estructura más intuitiva, en este tipo de código. Algunos de los lenguajes compilados más comunes son por ejemplo *C* o *C++*.

Los continuos avances en hardware, han hecho que las computadoras empiecen a prescindir del puerto serie, empleando en su lugar, múltiples puertos *USB*. Esto promovió la aparición de microcontroladores programables desde el puerto *USB*, sin necesidad de hardware externo adicional. Este tipo de microcontroladores, es capaz de sobrescribir su propia memoria de programa a través de una conexión *USB*, y un software de control. La región de memoria donde se lleva a cabo la escritura, se conoce como región de booteo, mientras que el código de máquina cargado, se conoce como *bootloader*; para que un microcontrolador con conexión *USB*, y una región de booteo habilitada de fábrica pueda operar en modo *bootloader*. Primero debe cargarse por los métodos ya descritos el *bootloader*, garantizando que quede cargado en la región habilitada para esta función, en caso contrario, en el momento en que trate de reprogramarse, el microcontrolador, perderá el programa, y será necesario reprogramar el *bootloader*. No obstante, este problema se resuelve cuando el fabricante carga un *bootloader*, en el microcontrolador, antes de comercializarlo. Este es el caso de las placas *Arduino*.

Este tipo de microcontroladores, cuentan con una interfaz de desarrollo, de fácil manejo, compatible con todos sus productos, permitiendo la programación de estos dispositivos desde un lenguaje de programación similar a *processing*, de fácil aprendizaje, incluso si no se cuenta con conocimientos previos en

microcontroladores. Esta interfaz, está basada en *Java*, por lo que requiere la instalación previa de la máquina virtual *Java* para su funcionamiento.

5.2. MARCO DE REFERENCIA OBJETOS

En esta sección se describe brevemente la historia, teoría y principales conceptos, en los que se vio involucrada desde etapas tempranas la programación orientada a objetos.

Los objetos desde el punto de vista computacional, son relevantes para este trabajo, debido a que cada una de las estructuras desarrolladas, tanto en la interfaz gráfica web, como la implementación del *Bitvoicer Server*, implican la manipulación de elementos, que no pueden ser descritos, mediante una única propiedad, como el caso de una simple variable, sino que están conformados por conjuntos de ellas, lo que hace pertinente la descripción de los objetos como apoyo a este trabajo.

5.2.1. MARCO HISTÓRICO

Muchas personas acogen los inicios de 1960, como la fecha en que se introdujo el concepto de objeto, con el lenguaje de programación *Simula* [3].

Este software, fue creado para apoyar simulaciones de situaciones reales, donde están involucradas cientos de partes interactuando entre ellas, permitiendo dar un tratamiento individual a cada pieza, así como definir su comportamiento. La aparición de este lenguaje, popularizó el uso de términos como clases, objetos, instancias, herencia y polimorfismo.

No pasó mucho tiempo para la aparición de un segundo lenguaje orientado a objetos. Con una interfaz de desarrollo basada en una máquina virtual, una gran biblioteca de objetos reutilizables y tipado dinámico, nace *Smalltalk*, un lenguaje de programación orientado a objetos desarrollado por un grupo investigativo de *Xerox's*. Una de sus particularidades, es la posibilidad de enviar mensajes entre los distintos objetos creados.

En 1983, nace *C* ++, un lenguaje de programación basado en *C* y en *Simula*, con amplia aceptación, aunque no es un lenguaje puramente orientado a objetos, debido a que combina técnicas comunes de la programación estructurada, en procura de su compatibilidad con C.

En 1991, James Gosling de *Sun Microsystems*, inicia una investigación acerca de la posibilidad de crear un lenguaje de programación independiente de la

plataforma de hardware donde se ejecutara, intentando inicialmente con *C++*, pero abandona pronto esta idea, iniciando la creación de un nuevo lenguaje, que inicialmente llamó *Oak*, que posteriormente fue renombrado por la empresa a *Java*.

Java se popularizó rápidamente, permitiendo escribir un único código, que luego podía ser ejecutado en múltiples sistemas operativos, sin requerir compilación. Convirtiéndose en un lenguaje interpretado. Motivo de esta popularidad, la empresa *Microsoft*, respondió con la creación de un lenguaje de programación denominado *C#*, así como la plataforma *.NET Framework*, que permitía desarrollar en varios lenguajes de *Microsoft*, cada uno con características diferentes, para después integrar los resultados. Sin embargo este *Framework*, únicamente funcionaba en sistema operativo *Windows*, lo que suponía una gran limitación.

5.2.2. MARCO TEÓRICO

Un objeto es una unidad definida dentro de un programa de computadora, descrita a partir de propiedades y comportamientos. Estas propiedades y comportamientos son heredados desde unidades de referencia denominadas clases, ya sea a través de instanciamiento o extensión. El instanciamiento se refiere a la creación de un objeto, a partir de una unidad de referencia, tomando, todas las características y comportamientos de la clase padre, mientras que la extensión corresponde al uso, de sólo algunos de estos elementos de las clases desde las que se extiende.

No todas las propiedades de un objeto pueden ser vistas o modificadas, desde otras clases. Esta característica se conoce como visibilidad, y limita el acceso entre objetos que no guarden relación unos con otros. Esta condición, se presenta típicamente con la palabra *public* o *private* antepuesta al elemento correspondiente.

5.3. MARCO DE REFERENCIA *BITVOICER SERVER*

En esta sección se describe brevemente la historia, principales conceptos y operación de los principales desarrollos de software de reconocimiento de habla. Cabe resaltar que este software, estructura su funcionamiento en el paradigma de objetos, de modo que la mayor parte de los elementos relacionados al software, se auxiliaran en este paradigma para su definición y uso.

5.3.1. MARCO HISTÓRICO

El primer software de reconocimiento de habla conocido, hace su aparición en 1952. AUDREY, fue construido por Davis, Biddulph, y Blashek, en los laboratorios *Bell*, a partir de circuitería análoga. Este sistema era capaz de reconocer dígitos separados por pausas, con una efectividad de entre el 97 y el 99%, cuando se ajustaba a las condiciones de voz del usuario [6]. Su principal función era reemplazar el marcado de números telefónicos de forma manual, sustituyéndolo por la voz del usuario.

A continuación se presenta una línea de tiempo, con los avances más relevantes en materia de reconocimiento de habla, hasta la fecha:

- En 1976 la universidad de *Carnagie Mellon*, construyó *HARPY*, un sistema de reconocimiento basado en un enfoque de búsqueda de las posibles palabras más eficiente, pudiendo reconocer hasta 1011 palabras [7]
- En 1990, la empresa *Dragon Systems*, comercializa el primer software de reconocimiento de voz abierto al público, aunque demasiado costoso, denominado *Dragon Dictate*. En este software, las palabras debían ser pronunciadas una a la vez, debido a que no estaba preparado para reconocimiento continuo de habla
- En 1997, *Dragon Systems*, mejora su software de reconocimiento, de tal forma que el usuario pudiera interactuar de forma natural con el software, sin tener que hacer pausas, en cada palabra, denominándolo *Dragon NaturallySpeaking*
- En 1980, gracias a nuevas técnicas para comprender el lenguaje humano, el reconocimiento de vocabulario, pasó de unas pocas cientas de palabras a unas pocas miles, con el potencial de reconocer un número ilimitado de palabras. Una de las razones principales, fue un nuevo método estadístico, conocido como *hidden Markov model*.
- En 2007, el grupo *SRI* desarrolló una aplicación de procesamiento de audio natural para la respuesta a preguntas y la realización de acciones. Esta aplicación puede encontrarse en los teléfonos *iPhone*.

- En 2002, Google libera *Google Voice Search*, un producto que permite realizar búsquedas en este navegador
- En 2012, la empresa brasileña *Bitsophia*, desarrolla un servidor diseñado para el reconocimiento y síntesis de voz automática con el potencial de proveer estos recursos a un microcontrolador de 8 *bits*. Este software conocido como *Bitvoicer*, permite establecer las palabras a las que se desea que el sistema responda, y las acciones a ejecutar según el caso.

5.3.2. MARCO TEÓRICO *BITVOICER SERVER*

Bitvoicer Server es una software desarrollado en *Microsoft .Net Framework*, lo que confina su uso al sistema operativo *Windows*.

Este servidor, interactúa con el sistema operativo y sus aplicaciones, mediante *Windows Communication Foundation services*, a través de *Web Services*, permitiendo la transmisión de los datos generados en el servidor, a otras aplicaciones para su aprovechamiento. Estos datos, se generan a partir de las diferentes funciones desarrolladas por el software por defecto, o los asociados a la captura de datos desde cada uno de los *Devices* conectados.

Los *Devices*, son grupos de hardware o software compuestos de uno o más microcontroladores, con el potencial de acceder a los servicios del servidor a través de la interfaz de comunicación. Al interior del programa, reciben un tratamiento de objetos, por lo que son definidos a partir de sus propiedades y comportamientos.

Existen 3 tipos de dispositivos admitidos por el *Bitvoicer Server, input*, *output* y *mixed*.

Este último fue el empleado en este trabajo, debido a que admite las características de los 2 primeros tipos, pudiendo comportarse como entrada y salida de audio.

5.3.2.1. FILTROS

El *Bitvoicer Server*, recepciona las peticiones de transcripción enviadas por cada uno de los *Devices* conectados. Sin embargo, no todas estas peticiones derivan en transcripciones válidas, debido a que algunos de los comandos no son

identificados correctamente o simplemente corresponden a ruido en el entorno donde se encuentre ubicado el micrófono.

El reconocimiento de cada uno de las cadenas de audio de cada uno de los *Devices* que llega al *Bitvoicer*, es procesado y convertido a texto, mediante el proceso *Speech Recognition Engine (SRE)*, ejecutado para cada cliente.

Este proceso no sólo identifica las palabras en cada conjunto de datos, sino que se encarga de validarlas, enviando únicamente las que se consideran válidas, al aplicar 5 filtros por software.

En la Tabla 2, se presentan estos filtros

Filtro	**Función**
Minimun Audio Level	Está compuesto por el nivel de audio promedio medido en las últimas capturas más el valor definido en este filtro. Para ser considerado válido, el nivel de audio de los datos enviados, debe alcanzar o superar este valor por lo menos una vez
Audio-level-ativated Period	Este periodo es medido en milisegundos, empieza cuando el mínimo nivel de audio es alcanzado
Latency Period	Este periodo es medido en milisegundos, empieza justo después de que se realiza un reconocimiento válido. Los reconocimientos que tengan lugar al interior de este periodo son rechazados
Activation Word	Palabra o frase que apertura la posibilidad de realizar más reconocimientos
Activation-word-activated Period	Periodo de vigencia de la palabra de activación. Sólo los reconocimientos realizados dentro de este periodo son considerados válidos

Tabla 2. FILTROS IMPLEMENTADOS POR EL *SER*

5.3.2.2. ESTRUCTURA DE UN SISTEMA DE RECONOCIMIENTO DE VOZ

Un sistema de reconocimiento de voz, es un software con la habilidad de una de identificar palabras o frases en lenguaje hablado, para traducirlas a un formato interpretable por la máquina. Este proceso de convertir una señal de voz en secuencias de palabras, se consigue mediante la implementación de algoritmos específicos. El diseño de un sistema de reconocimiento de voz, depende de varias componentes: una categorización de los tipos de habla reconocida a través de clases, la forma en la que se representan las señales preprocesadas, las técnicas de extracción de características disponibles, los clasificadores usados, y los recursos como plantillas y bases de datos necesarios para la identificación de las frases [9]. En la Figura 1, puede verse una estructura típica de un sistema de reconocimiento de voz, similar a la descrita.

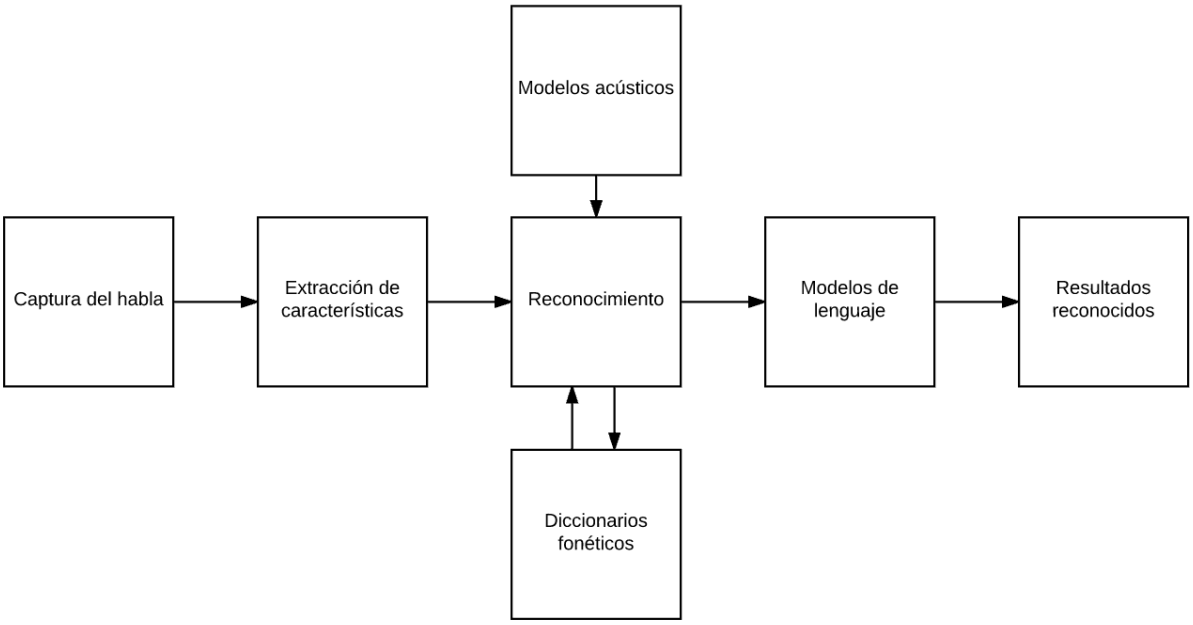


Figura 1. ESTRUCTURA DE UN SISTEMA DE RECONOCIMIENTO DE VOZ

Las categorías de reconocimiento de voz son: reconocimiento de palabras aisladas, reconocimiento de palabras conectadas, reconocimiento de habla continuo, y reconocimiento de habla espontáneo. Las dos primeras categorías mencionadas, no requieren la ejecución de cálculos tan complejos como el caso

de la tercera y cuarta categoría. En este trabajo de grado no se desarrolla un sistema de reconocimiento de habla. Sin embargo, se implementa uno capaz de reconocer palabras conectadas, conocido como *Bitvoicer Server*, este tipo de sistemas de reconocimiento difiere de los de habla aislada, en que la pausa entre palabras emitidas, es mínima, en relación a los de habla aislada. De esta manera se consigue expresar frases completas de una sola vez, sin el deber de hacer una pausa significativa entre palabras.

El *Bitvoicer Server*, además de reconocer frases predeterminadas por el usuario, es independiente al interlocutor. Estas características, son similares a las descritas en la sección *Conclusion* de [10], por tal motivo, se estima que este software, basa su funcionamiento en los modelos ocultos de *Markov*.

5.4. MARCO DE REFERENCIA DESARROLLO WEB

En esta sección se describe brevemente la historia y principales conceptos involucrados en el desarrollo web. Así mismo, se describen los múltiples papeles, que desempeña el desarrollo web, en este trabajo. Uno de ellos es el de dar vida a la interfaz de usuario, que representa el ambiente domótico simulado, permitiendo no sólo la construcción de su estructura básica, sino también, su posicionamiento y estilizado, a través de tecnologías como *HTML5* y *CSS3*.

5.4.1. MARCO HISTÓRICO

La historia del desarrollo web, se remonta a 1980, cuando el físico Tim Berners-Lee, trabajador del *CERN*, propuso un nuevo sistema de “hipertexto” para compartir documentos [8].

Los sistemas de “hipertexto” habían sido desarrollados años antes, permitiendo que los usuarios accedieran a la información relacionada con los documentos electrónicos que estaban visualizando. De cierta manera, los primitivos sistemas de “hipertexto” podrían asimilarse a los enlaces de las páginas web actuales.

Tras finalizar el desarrollo de su sistema de “hipertexto”, Tim Berners-Lee lo presentó a una convocatoria organizada para desarrollar un sistema de “hipertexto” para internet. Después de unir sus fuerzas con el ingeniero en sistemas Robert Cailliau, presentaron la propuesta ganadora llamada *Word Wide Web (W3)*.

El primer documento formal con la descripción de *HTML* se publicó en 1991 bajo el nombre de *HTML Tags* y todavía hoy, puede ser consultado online a modo de reliquia informática.

La primera propuesta oficial para convertir *HTML* en un estándar se realizó en 1993 por parte del organismo IETF (*Internet Engineering Task Force*). Aunque se consiguieron avances significativos (en esta época se definieron las etiquetas para imágenes, tablas y formularios) ninguna de las dos propuestas de estándar, llamadas *HTML* y *HTML+* consiguieron convertirse en estándar oficial.

En 1995, el organismo IETF organiza un grupo de trabajo de *HTML* y consigue publicar, el 22 de septiembre de ese mismo año, el estándar *HTML 2.0*. A pesar de su nombre, *HTML 2.0* es el primer estándar oficial de *HTML*.

A partir de 1996, los estándares de *HTML* los publica otro organismo de estandarización llamado *W3C* (*World Wide Web Consortium*). La versión HTML 3.2 se publicó el 14 de Enero de 1997 y es la primera recomendación de HTML publicada por el *W3C*. Esta revisión incorpora los últimos avances de las páginas web desarrolladas hasta 1996, como *applets* de *Java* y texto que fluye alrededor de las imágenes.

HTML 4.0 se publicó el 24 de Abril de 1998 (siendo una versión corregida de la publicación original del 18 de Diciembre de 1997) y supone un gran salto desde las versiones anteriores. Entre sus novedades más destacadas se encuentran las hojas de estilos *CSS*, la posibilidad de incluir pequeños programas o *scripts* en las páginas web, mejora de la accesibilidad de las páginas diseñadas, tablas complejas y mejoras en los formularios.

La última especificación oficial de *HTML* se publicó el 24 de diciembre de 1999 y se denomina *HTML 4.01*. Se trata de una revisión y actualización de la versión *HTML 4.0*, por lo que no incluye novedades significativas.

Desde la publicación de *HTML 4.01*, la actividad de estandarización de *HTML* se detuvo y el *W3C* se centró en el desarrollo del estándar *XHTML*. Por este motivo, en el año 2004, las empresas *Apple*, *Mozilla* y *Opera* mostraron su preocupación por la falta de interés del *W3C* en *HTML* y decidieron organizarse en una nueva asociación llamada *WHATWG* (*Web Hypertext Application Technology Working Group*).

La actividad actual del *WHATWG* se centra en el futuro estándar *HTML* 5, cuyo primer borrador oficial se publicó el 22 de enero de 2008. Debido a la fuerza

de las empresas que forman el grupo *WHATWG* y a la publicación de los borradores de *HTML* 5.0, en marzo de 2007 el *W3C* decidió retomar el proceso de estandarización de *HTML*.

La empresa *Netscape*, después de hacer compatibles sus navegadores con las *applets Java*, comenzó a desarrollar pequeños programas en las páginas y que fuese mucho más sencillo de usar, sin demandar la instalación de software adicional, como el caso de la máquina virtual de *Java*. El desarrollo de este nuevo lenguaje de programación se realizó de la mano de la empresa *Sun Microsystems*, de modo que el resultado fuera de fácil de usar, y empleable únicamente al interior de una página web.

Netscape 2.0 fue el primer navegador que interpretaba *Javascript* y su estela fue seguida por otros navegadores web como *Internet Explorer* a partir de la versión 3.0. Sin embargo, la compañía *Microsoft* nombró a este lenguaje como *JScript* y tenía ligeras diferencias con respecto a *Javascript*, algunas de las cuales perduran hasta el día de hoy.

5.4.2. MARCO TEÓRICO

A continuación se presentan algunos conceptos importantes intrínsecos al diseño web, empleados de forma directa, en el desarrollo de la interfaz gráfica.

5.4.2.1. DOCUMENT OBJECT MODEL

Cuando se definió el lenguaje *XML*, surgió la necesidad de procesar y manipular el contenido de este tipo de archivos mediante los lenguajes de programación tradicionales. *XML* es un lenguaje sencillo de escribir pero complejo para procesar y manipular de forma eficiente. Por este motivo, surgieron algunas técnicas entre las que se encuentra *DOM*.

DOM es un conjunto de utilidades específicamente diseñadas para manipular documentos *XML*. Por extensión, también puede utilizarse para manipular documentos *XHTML* y *HTML*. Técnicamente, *DOM* es una *API* de funciones que se pueden utilizar para manipular las páginas *XHTML* de forma rápida y eficiente.

Antes de poder utilizar sus funciones, *DOM* transforma internamente el archivo *XML* original en una estructura más fácil de manejar, formada por una jerarquía de nodos. De esta forma, *DOM* transforma el código *XML*, en una serie de nodos interconectados en forma de árbol, representando los contenidos del archivo original y las relaciones entre ellos.

El *DOM*, es de suma importancia en este trabajo, el motivo principal es que los elementos que se visualizan en la interfaz gráfica, una vez carga el navegador, no están expresamente definidos en el código *HTML*, sino que son adicionados dinámicamente en tiempo de ejecución al *DOM.*

5.4.2.2. PREPROCESADORES DE *CSS*

Un preprocesador de *CSS*, es una herramienta que permite escribir código similar al *CSS*, pero sin las limitaciones típicas que este lenguaje presenta, tales como variables, funciones, *mixins*, anidaciones o modularidad.

El uso de preprocesadores genera código más manejable y mantenible. En este trabajo de grado se eligió *SASS*, como preprocesador de *CSS*, permitiendo simplificar la creación directa de código *CSS*, de 3600 a 250 líneas en las hojas de estilos aproximadamente, aprovechando la mayoría de propiedades de lenguajes como la herencia y el encapsulamiento, tan útiles a la hora de reutilizar el código. Esta tarea resultaría muy tediosa en construyendo el código desde la perspectiva de *CSS* puro, resultado en un código poco administrable, y concentrando la detección de errores de código, y no en la lógica del problema como se pretende.

5.4.2.3. EVENT HANDLING

En una computadora, un evento es la ocurrencia de algún tipo de interacción no necesariamente periódica entre el usuario y el sistema operativo, usualmente a través de los periféricos. Esta interacción es procesada directamente por un servicio de manejo de interrupciones, encargado de asignar las acciones más apropiadas en relación a la naturaleza de la interrupción. Fuentes típicas de interrupción son por ejemplo el *mouse* y el teclado, donde el movimiento del primero, se corresponde con el movimiento del puntero en la pantalla de la computadora, y la pulsación de teclas del segundo se refleja en acciones muy variadas en la interfaz gráfica.

Existen diversos programas con los que pueden capturarse eventos, y ejecutar funciones en relación a los mismos. Un navegador web, es uno de estos ejemplos. Sin embargo esta función la consigue con el apoyo *JavaScript* en conjunción con la *API* para el manejo del DOM, del propio *JavaScript*.

De esta manera *JavaScript*, definido como un lenguaje de programación orientado a objetos, de tipado dinámico, admite manejo de eventos, lo que lo hace

suficientemente flexible para la creación y control de ambientes simulados, pues estos están conformados por objetos y demandan interacción con el usuario.

5.4.2.4. JQUERY

jQuery es una librería de *JavaScript*, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos *HTML,* manipular el árbol del *DOM*, manejar eventos, desarrollar animaciones y agregar interacción con la técnica *AJAX* a páginas web. *jQuery* es la librería de *JavaScript* más utilizada.

Una de las grandes potencialidades de *jQuery* empleadas en este trabajo de grado es la adición dinámica de elementos *HTML*. Esta habilidad, se utilizó conjuntamente con un ciclo for, de 42 iteraciones, para ubicar estructuralmente los bloques al *DOM*. Los estilos asociados a cada bloque, así como sus posiciones al interior de la interfaz, fueron asignados mediante el preprocesador de *CSS SASS.*

6. METODOLOGÍA

Este trabajo busca implementar un sistema de reconocimiento de voz, empleando un dispositivo de captura de datos a través de un microcontrolador, empleando, una computadora para procesamiento de datos y un ambiente domótico simulado.
Sin embargo, son muchas las tecnologías empleadas en este trabajo, lo que hace poco intuitiva, la comprensión de cada una de las piezas como un todo, sin antes, detallar la forma en la que se procedió a diseñar y programar cada uno de los elementos implicados. Finalmente, se presenta la forma en la que se integraron los sistemas vinculados.

En la Figura 2, se presenta un diagrama de flujo, donde se muestran la manera en que fluye la información entre los 3 sistemas involucrados en este trabajo de grado

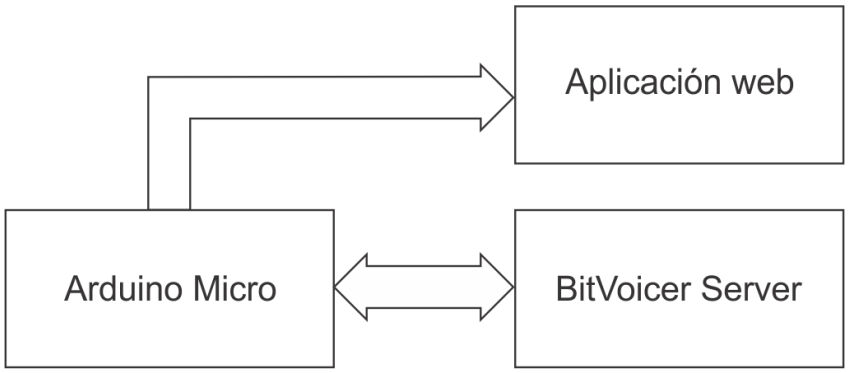


Figura 2. FLUJO DE DATOS ENTRE LOS MÒDULOS DEL SISTEMA DE RECONOCIMIENTO

6.1. METODOLOGÍA ASOCIADA AL *ARDUINO MICRO*

En esta etapa, se escogió el hardware más adecuado para cubrir la mayor cantidad de necesidades del trabajo, y empleando la menor cantidad de hardware posible. Por esta razón, se escogió el *Arduino Micro*, dado su potencial como interfaz de adquisición de señales con la resolución adecuada, y la posibilidad de operar como puente entre las tecnologías involucradas, pasando a la selección del micrófono.

Para la elección del micrófono junto con su acondicionamiento de señal, se consideraron las siguientes condiciones:

1. Operación con fuente simple
2. Polarización con un nivel de tensión no superior al del puerto *USB*
3. Pequeña presentación
4. Fácil adaptabilidad, a la placa *Arduino*

El micrófono seleccionado, es de tipo *electret*, y viene soldado a una placa base, en conjunto con el acondicionamiento de señal, como se muestra en la Figura 3.

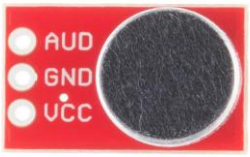


Figura 3. MICROFONO *ELECTRET* EMPLEADO

Esta placa, se comercializa bajo la referencia BOB-09964, donde reside el acondicionamiento de señal, conformado por un amplificador operacional de precisión opa344, necesario para llevar la señal del orden de mV a un valor máximo de 5.5v. De esta manera, se logró prescindir del diseño de un circuito de acondicionamiento de señal y de la utilización de cableado innecesario. En esta etapa, se conecta la terminal *AVD*, al conversor análogo digital de *Arduino,* en la terminal A0, de la placa. Una vez la señal es capturada, puede ser digitalizada por el *Arduino*.

La segunda etapa, consistió en la digitalización y el procesamiento de la señal capturada. Sin embargo, el *Arduino Micro*, no cuenta con la capacidad de cálculo requerida por esta última tarea, por lo que debe delegar esta labor, a la computadora a la que se conecta a través del puerto *USB*, en este caso mediante la clase *HID*, característica de los teclados y mouse. Permitiendo a la computadora, identificar al *Arduino Micro*, una vez programado y conectado a la computadora, como un periférico tradicional. Este último potencial es especialmente importante, debido a que permitió establecer una comunicación bidireccional, entre el *Arduino Micro* y la computadora, a través del protocolo *USB*, empleando la clase *HID*, con la ayuda de la librería *Keyboard*.

A continuación se presenta a través de un diagrama de bloques, el algoritmo que describe la interacción entre el programa en el *Arduino Micro*, el *Bitvoicer Server*, y el ambiente domótico, proveyéndole el potencial de operar como un teclado virtual. Se anexa un diagrama de bloques en la Figura 4.

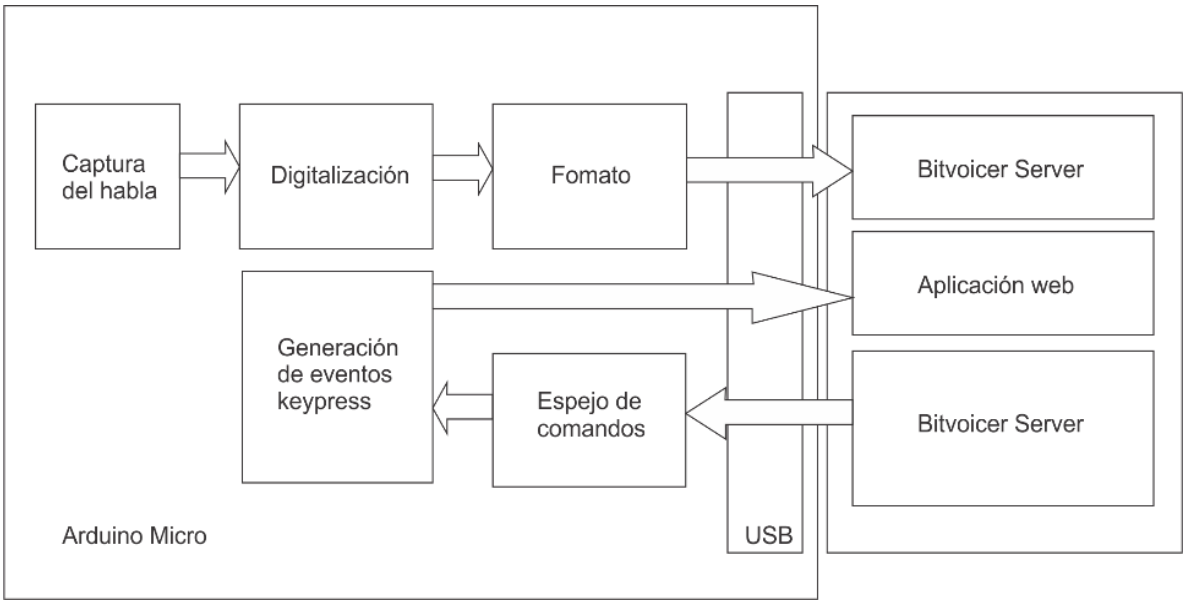


Figura 4. FUNCIONES PRINCIPALES ARDUINO MICRO

El usuario emite un comando mediante su voz, este puede estar compuesto por palabras separadas mediante una pequeña pausa, e incluso palabras individuales. Estos comandos, llegan al micrófono *electret*, quien se encarga, a través de su acondicionamiento de señal, de llevarlas a un nivel de tensión adecuado para su uso en el *Arduino* conectado, mediante uno de sus terminales de adquisición *AD*. De esta manera, la señal sonora es digitalizada y almacenada en el *Arduino*. El proceso continúa, cuando el programa en el *Arduino Micro*, da formato, a la señal digitalizada.

Uno de las principales funciones desempeñados por el *Arduino Micro*, es la de recibir los comandos enviados desde el *Bitvoicer Server*, una vez identifica alguna de los comandos emitidos por el usuario. De este modo los comandos llegan desde el *Bitvoicer Server,* al *Arduino*, a través del puerto *USB*, y son procesados mediante una subrutina. Esta subrutina, se encarga de enviar a través del puerto *USB* de la computadora, el mismo comando recibido, pero en lugar de enviarlo al software *Bitvoicer Server*, este llega directamente al sistema operativo de la computadora directamente mediante el puerto *USB*, disparando un evento tipo *Keypress*[4], adquiriendo el potencial de gobernar cualquier aplicación interna, que no cuente con restricciones de este tipo.

[4] Evento accionado por el navegador, sobre la detección de una tecla pulsada

Esta propiedad fue empleado, para integrar el *Arduino* y el hardware conectado con una aplicación web, accesible desde el navegador.

6.2. METODOLOGÍA ASOCIADA AL USO DEL BITVOICER SERVER

Uno de los primeros debates llevados a cabo, alrededor del desarrollo de este trabajo de grado, fue la elección del software de reconocimiento de voz. Esta fue favorable para el *Bitvoicer Server*, atendiendo la recomendación del Director del trabajo, entre las posibles alternativas para computadoras, considerando bajo costo y compatibilidad con el protocolo *USB* v2. Por esta razón y bajo la justificación descrita, se resolvió adquirir una licencia tipo *per-computer per-input device basis*[5].

A continuación se presenta un diagrama del tipo *UML*, que describe el sistema de reconocimiento de voz implementado en el *Bitvoicer Server*, mediante *Solution Objects*[6], algo así como las librerías de clases disponibles en un lenguaje de programación orientado a objetos. En la Figura 5, se encuentran 7 bloques. Cada uno de ellos, representa uno de los 7 objetos ingresados al software. Así mismo cada bloque, contiene los respectivos atributos, asociados al objeto. Los objetos ubicados debajo de cada objeto, son en realidad objetos al interior del objeto ubicado un nivel más arriba, debido a que provienen de clases internas.

[5] Licencia distribuida por la empresa brasileña *Bitsophia*, válida para la activación de un único computador con un sólo dispositivo de captura de habla a la vez

[6] Concepto propio de la empresa *Bitsophia*, para referirse a un objeto computacional

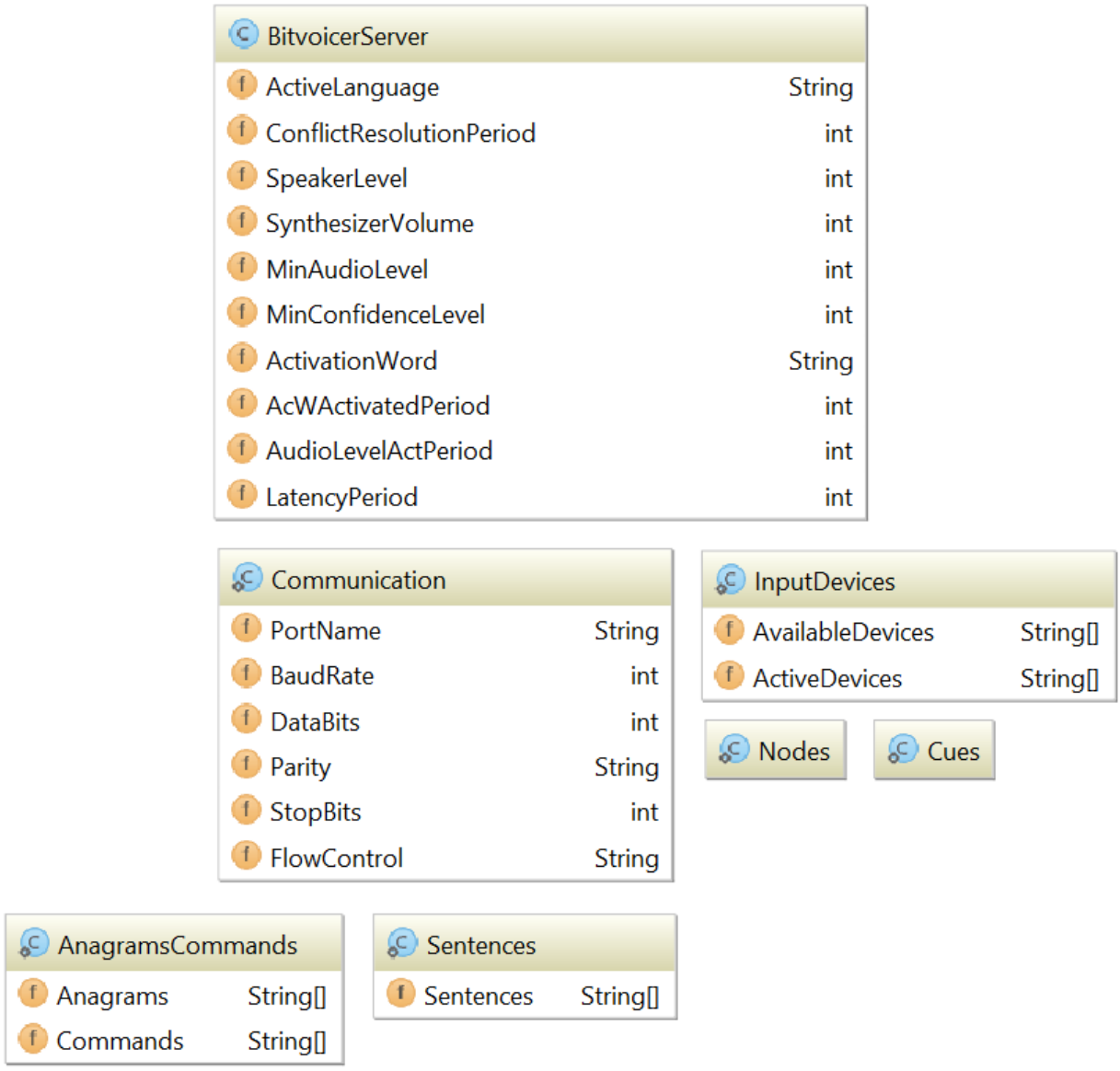


Figura 5. DIAGRAMA UML DEL SISTEMA DE RECONOCIMIENTO IMPLEMENTADO EN EL BITVOICER

En la Tabla 3, Se presentan las propiedades públicas presentadadas en el diagrama, así como los valores que les fueron asignados en este trabajo.

Propiedad	Valor
ActiveLanguage	Español (España)
ConflictResolutionPeriod	500
SynthesizerVolume	100
MinAudioLevel	35
MinConfidenceLevel	75
ActivationWord	Hola Arduino Micro
AcWActivatedPeriod	120s
AudioLevelActPeriod	5000
LatencyPeriod	1s
BaudRate	115200
DataBits	8
Parity	None
StopBits	1
FlowControl	None

Tabla 3. VALORES ELEGIDOS PARA LAS PROPIEDADES DEL SISTEMA DE RECONOCIMIENTO

6.3. METODOLOGÍA ASOCIADA AL DESARROLLO DE LA INTERFAZ GRÁFICA

Este trabajo de grado, implicó el desarrollo de una aplicación web basada en los estándares *HTML5*, *JavaScript* y *CSS3*, consiguiendo un desarrollo de vanguardia con el potencial de ser operado de forma remota desde una interconexión a internet.

Un ambiente domótico, está conformado por gran variedad de dispositivos, estructuras y elementos de mando. Estos últimos se emplean para controlar los dispositivos, presentes en la estructura del ambiente. Sin embargo, no es indispensable contar con un ambiente domótico real, para verificar su funcionamiento, debido a que existen diversas formas de modelar y simular ambientes. Por este motivo, a partir del desarrollo web, y considerando las facilidades que ofrece para el desarrollo de aplicaciones en la actualidad, se optó por construir una interfaz gráfica, donde pudiera estimarse la forma en la que operaría un sistema domótico operado por voz, sin la necesidad de adquirir costosos dispositivos, o disponer de algún ambiente.

Previa iniciación del desarrollo de la aplicación, se planeó la forma en que se construiría cada uno de los elementos del ambiente simulado, considerando un modelo por bloques, bastante adecuado para construir los elementos tanto interiores como exteriores, procurando que todos los elementos necesarios, pudieran ser modelados como paralelepípedos, donde el color, la textura, el volumen, y el área de las caras, generara la distinción entre una puerta, un andén, un cuarto oscuro y una simple pared. Sin embargo, algunos elementos tridimensionales como las bombillas, debieron ser simulados mediante imágenes, por simplicidad, respetando la perspectiva. En la Figura 6, se muestra la interfaz desarrollada.

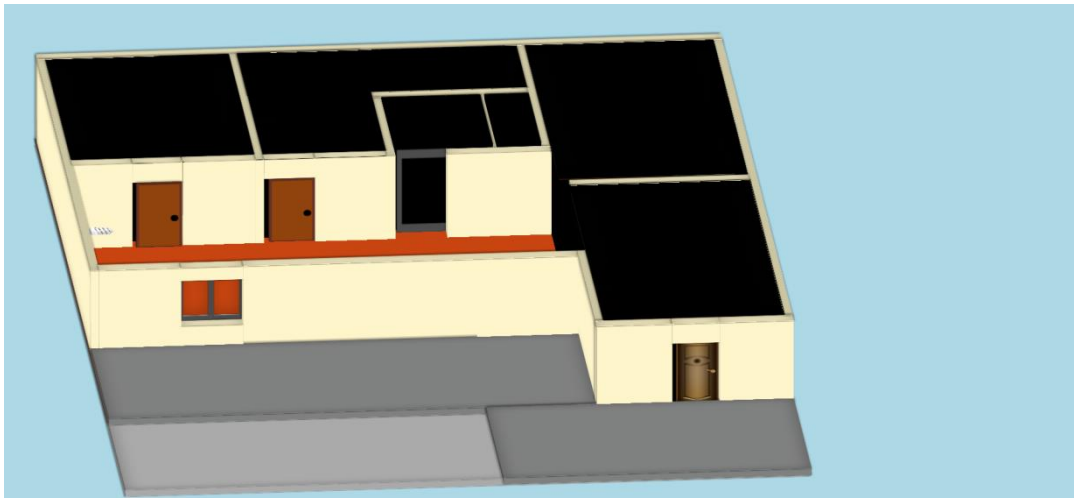


Figura 6. APLICACIÓN WEB DESARROLLADA

Cada uno de los 47 elementos tipo bloque al interior de la aplicación web, exceptuando los bombillos, están conformado por un paralelepípedo, lo que facilita la adecuación y expansión del ambiente. Sin embargo, ni *HTML5*, cuenta de forma nativa con etiquetas de este tipo, ni *CSS3,* con las funciones necesarias para estilizar tantas etiquetas de forma sistemática, de modo que debía construirse, empleando los elementos a disposición. En ese sentido, se decidió construir un paralelepípedo a partir de etiquetas simples denominadas *divs*. Es así como se construyó un paralelepípedo, a partir de 6 *divs*, donde cada uno representaba una cara diferente de las 6 necesarias para su elaboración, variando su posición, ángulo de orientación en cada eje y dimensiones. El bloque resultante se estilizó mediante una hoja *CSS*.

Si bien fue posible construir un bloque a partir de etiquetas más simples, permitiendo la elaboración de formas más complejas, como paredes huecas, construidas a partir de bloques, el ambiente demandaba muchas formas más incluso la posibilidad de adicionar más formas, en futuros trabajos, logrando escalabilidad, en la medida en que creciera la infraestructura del ambiente. Es ahí donde se empleó por primera vez *JavaScript*, al interior del desarrollo, cumpliendo como función primaria, la inserción de etiquetas o grupos de etiquetas predeterminadas de *forma dinámica*[7] al interior de la aplicación. No obstante, los nuevos bloques insertados, carecían de estilo alguno, que les diera, las dimensiones, posiciones y formas indicadas, según su ubicación real en el ambiente simulado, debido a que los estilos, debían ser asignados de forma independiente, mediante un archivo de tipo *CSS*, donde además se garantizara la compatibilidad con navegadores distintos al *Google Chrome*, empleado en este trabajo. De esta manera se hizo un proceso de validación del estado del arte, con el fin de conocer y aprender algunas herramientas, que lograran acelerar el proceso de adición de estilos al ambiente, para cada uno de los bloques presentes, sin limitarse por las restricciones propias de las reglas *CSS*, citadas en el marco teórico asociado al desarrollo web. En condiciones normales, y empleando únicamente *CSS* en este desarrollo, hubiera resultado en 3560 líneas de código, una a una. Sin embargo, en este trabajo de grado, se empleó *SASS*, una extensión de *CSS*, resultando en la escritura, de sólo 200.

Una vez terminada la estructura del ambiente, así como definidos los estilos para cada uno de los elementos. Se diseñó la lógica de operación del ambiente mediante el lenguaje *JavaScript*, proveyendo al ambiente el potencial de responder ante eventos sobre él.

Son muchos los eventos, ante los que puede responder *JavaScript*, sin embargo, debía buscarse uno que fuese compatible con los emitidos desde el puerto *USB* con el *Arduino Micro*. Es así como se escogió trabajar con el evento *Keypress*, disponible en *JavaScript*, correspondiendo con el tipo de evento emitido por el *Arduino Micro,* en el momento en que recibe comandos desde el *Bitvoicer Server*. Sin embargo, previa realización de las funciones delegadas para este fin, se simuló la operación del *Arduino Micro*, como emisor de eventos, mediante la pulsación de teclas.

La estructura de comandos empleados, mediante el teclado es de la forma *ab*, donde el primer carácter, representa el objeto sobre el que se desea realizar alguna acción, y el segundo su magnitud.

[7] Cambio en la definición de algún contenido *HTML* en tiempo de ejecución

Los elementos móviles sobre la interfaz, son las puertas y ventanas. Estos objetos, fueron designados con números del 0 al 4. Mientras que las luminarias, fueron designadas con los códigos del conjunto: {5,6,7,8,9,a}.

La implementación de este trabajo, implicó la elaboración de un autómata de estados finitos mediante código *JavaScript*.

El autómata, está conformado por la 5-tupla $(Q, \Sigma, q_0, \delta, A)$ donde:

- Q es el conjunto finito de estados
- Σ se corresponde con el alfabeto empleado
- q_0 es el estado inicial, y está incluido en el conjunto Q
- δ es una función de transición sobre el conjunto de entrada y el alfabeto
- A representa los estados finales o de aceptación

La Tabla 4, muestra el conjunto de estados Q, de la máquina.

Estado	Significado
s0	cerrada
s1	muy poco abierta
s2	un poco abierta
s3	a la mitad
s4	bastante abierta
s5	por completo abierta

Tabla 4. CONJUNTO DE ESTADOS DE LA MÁQUINA DE ESTADOS PUERTAS Y VENTANAS

Para alcanzar los estados de la Tabla 4, la máquina debe ser estimulada mediante el conjunto de entradas Σ. Estas entradas se muestran en la Tabla 5.

Entrada	Significado
p0	cerrar
p20	abrir muy poco
p40	abrir un poco
p60	abrir a la mitad
p80	abrir bastante
p100	abrir por completo

Tabla 5. ESTIMULOS DE ENTRADA A LA MÁQUINA DE ESTADOS

En la Tabla 6, se muestra la tabla de transiciones, empleada para el diseño del autómata de estados finitos, encargado de definir la operación de cada una de las puertas y ventanas del ambiente. Esta tabla presenta los estados alcanzados por la máquina, en el momento en que es estimulada con las entradas del conjunto Σ.

	Entrada					
Estado	**p0**	**p20**	**p40**	**p60**	**p80**	**p100**
s0	s0	s1	s2	s3	s4	s5
s1	s0	s1	s2	s3	s4	s5
s2	s0	s1	s2	s3	s4	s5
s3	s0	s1	s2	s3	s4	s5
s4	s0	s1	s2	s3	s4	s5
s5	s0	s1	s2	s3	s4	s5

Tabla 6. TABLA DE TRANSICIONES PUERTAS Y VENTANAS

De forma similar, se elaboró la Tabla 7, donde se muestran las salidas de la máquina, como función del conjunto de entrada Σ, y el estado actual.

	Salidas					
Estado	**p0**	**p20**	**p40**	**p60**	**p80**	**p100**
s0	0	1	2	3	4	5
s1	0	1	2	3	4	5
s2	0	1	2	3	4	5
s3	0	1	2	3	4	5
s4	0	1	2	3	4	5
s5	0	1	2	3	4	5

Tabla 7. TABLA DE SALIDAS PUERTAS Y VENTANAS

Cada salida se corresponde con la acción requerida, en el momento en que se estimula la entrada. En la Tabla 8, se muestra el significado de cada una de las salidas.

Salida	Significado
0	se encuentra cerrada
1	se encuentra muy poco abierta
2	se encuentra un poco abierta
3	se encuentra abierta a la mitad
4	se encuentra bastante abierta
5	se encuentra abierta por completo

Tabla 8. SALIDAS POSIBLES DE LA MÁQUINA DE ESTADOS

La Tabla 8, y La Tabla 9, presentan el caso análogo para el caso de las luminarias.

	Entrada	
Estado	**on**	**off**
s0	s0	s1
s1	**s0**	**s1**

Tabla 9. TABLA DE TRANSICIONES LUMINARIAS

	Salida	
Estado	**on**	**off**
s0	s0	s1
s1	s0	s1

Tabla 10. TABLA DE SALIDAS LUMINARIAS

7. INTEGRACIÓN DE LOS COMPONENTES DEL SISTEMA DOMÓTICO

En la etapa final del proceso, se integran, los subsistemas implementados en las etapas previas, consiguiendo establecer una interoperación armónica, entre los elementos implicados, reemplazando el uso del teclado físico del computador, por la función delegada al *Arduino Micro*, conectado a través del puerto *USB* de la computadora.

El siguiente diagrama de bloques, representa el papel que desempeñan cada uno de los elementos del sistema integrado, así como la direccionalidad de la comunicación con su periferia, ver Figura 6.

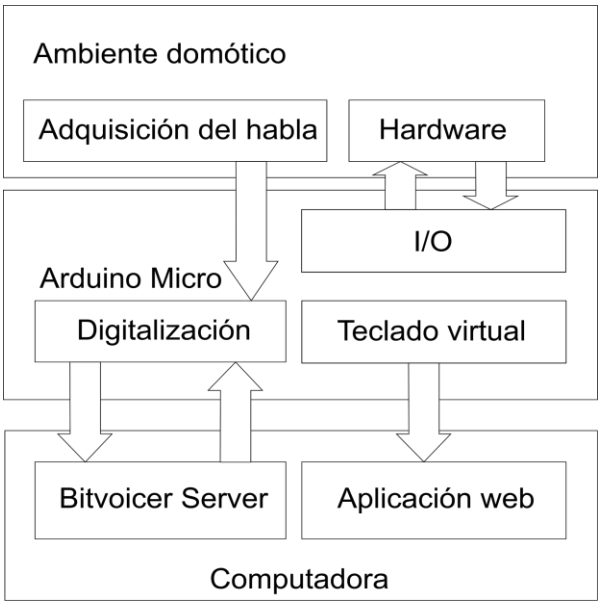


Figura 7. FUNCIÓN DESEMPEÑADA POR CADA MÓDULO

En la Figura 6, se presentan 3 bloques principales. En el primer bloque se representa el ambiente domótico, donde se ubican los elementos que representan, la estructura, así como los actuadores. En este primer bloque, es donde se pretende el usuario exprese las ordenes, que desea sean ejecutadas por el sistema. En el segundo bloque se encuentra el *Arduino Micro*, el sistema encargado de capturar el habla emitida desde el ambiente domótico, así como permitir la interacción bidireccional entre la computadora y el ambiente. Al tratarse de un microcontrolador, *Arduino* permite la interconexión a través de sus puertos de diferentes dispositivos electrónicos, como sensores y actuadores, ampliando el espectro de posibles acciones sobre el ambiente.

8. RESULTADOS

A continuación se presentan cada uno de los resultados obtenidos en este trabajo de grado, en relación a los objetivos perseguidos en el mismo.

8.1. RESULTADOS OBJETIVO 1

Se implementó un sistema de reconocimiento de habla aislada, empleando una voz femenina y una masculina, validando la injerencia del sexo en el resultado del reconocimiento, e identificando que el resultado es independiente a este factor. Así mismo se empleó música suave de fondo (*Jazz*) identificando que el resultado del reconocimiento no se ve afectado significativamente, siempre que el ruido de fondo no supere el 25% del volumen de la voz del usuario.

8.2. RESULTADOS OBJETIVO 2

Se desarrolló una rutina para el *Arduino Micro*, capaz de capturar, digitalizar y formatear adecuadamente la señal del proveniente del micrófono *electret*, para su envío al *Bitvoicer Server*, a través del puerto *USB*. Esta rutina, también es la encargada de recibir los comandos enviados desde el *Bitvoicer*, para su utilización. Este resultado es especialmente importante debido a que los resultados que llegan al *Arduino*, pueden ser empleados en hardware externo, o enviados devuelta para su uso en el sistema operativo, como en este caso, donde fueron empleados en el ambiente domótico simulado.

8.3. RESULTADOS OBJETIVO 3

Uno de los objetivos principales de este trabajo era el desarrollo de una interfaz gráfica empleando lenguajes web, en representación de un ambiente domótico simulado, con las condiciones mínimas comunes a estos ambientes. De este modo, el ambiente debía contar con puertas, ventanas y luminarias, por lo que se agregaron estos elementos. La aplicación resultante, responde ante

eventos *keypress* habilitados mediante comandos de voz enviados al *Arduino*, mediante el *Bitvoicer Server*.

A continuación se presentan 2 tablas en las que se presenta el estado actual del ambiente simulado, una acción de entrada, el estado final del ambiente, y la salida ante dicha entrada. En la Tabla 11, se muestra este proceso para la acción de encendido de la luz de la sala.

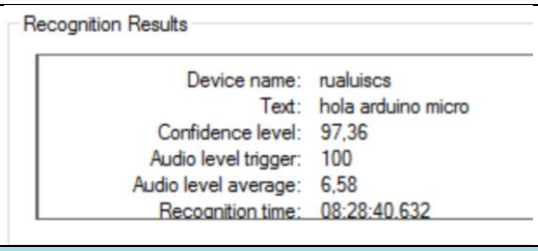
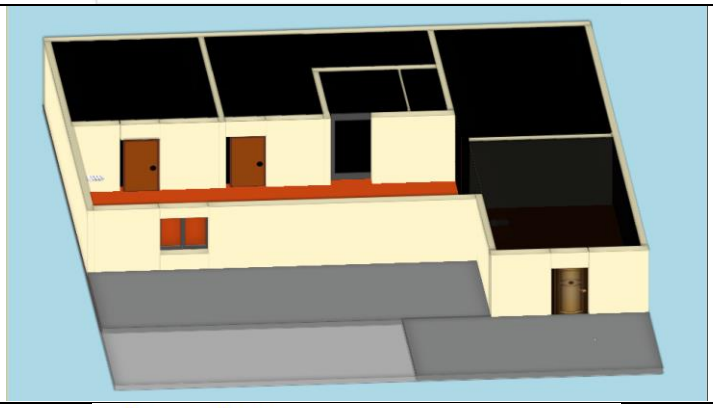
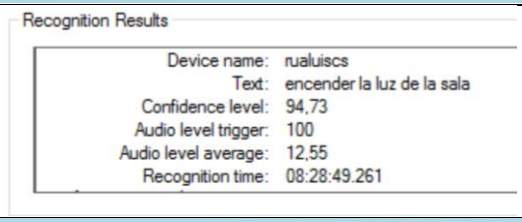
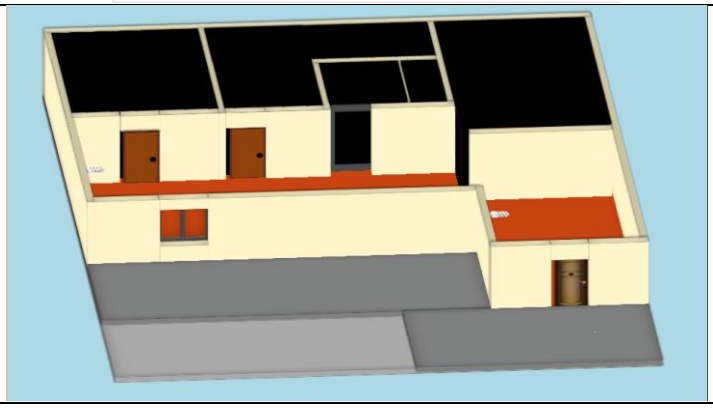
	Comando emitido
	Estado actual
	Comando emitido
	Estado final

Tabla 11. ENCENDIDO DE UNA LUZ DEL AMBIENTE DOMÓTICO

En la Tabla 11, se mostró el proceso general para encender una luz del ambiente domótico, así como los niveles de confianza típicos alcanzados por el sistema en una habitación cerrada. Cabe resaltar, que antes de iniciar el proceso de emisión de órdenes, debe habilitarse el sistema de reconocimiento, mediante el comando de activación, en este caso particular se escogió la frase: “Hola *Arduino Micro*”.

En la Tabla 12, se muestra el proceso para apagar una luz del ambiente domótico. Se destaca que en esta oportunidad, no fue necesario emitir el comando de activación, debido a que el sistema ya se encontraba habilitado, es decir, la habilitación del sistema seguía vigente, desde la emisión del último comando de activación. En este caso particular, el tiempo de vigencia del comando de activación, se seleccionó en 120s.

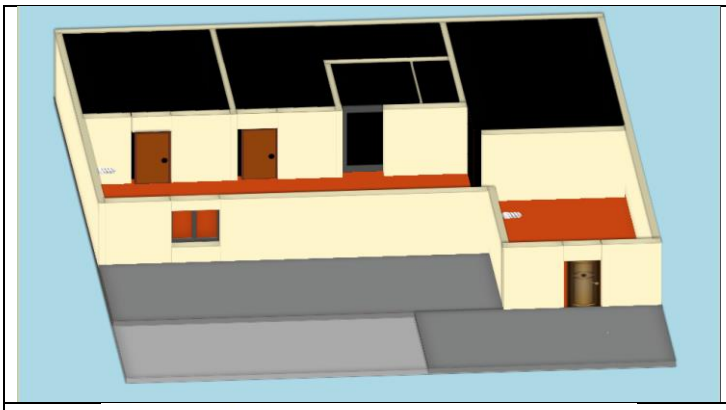
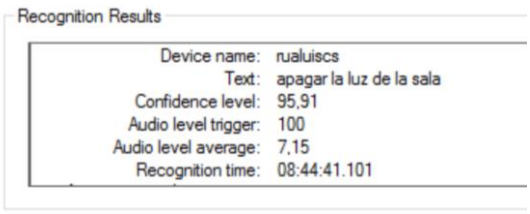
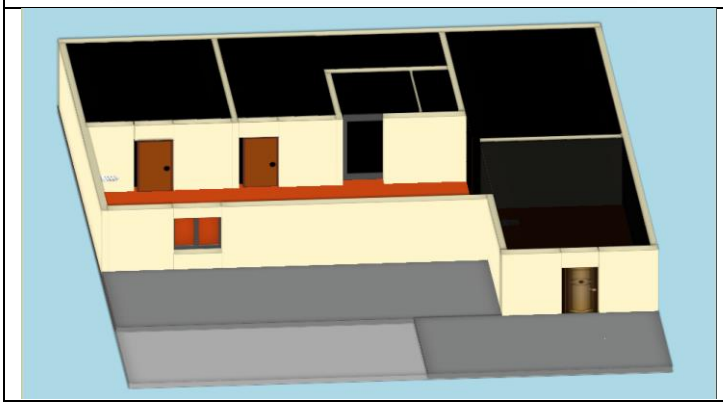
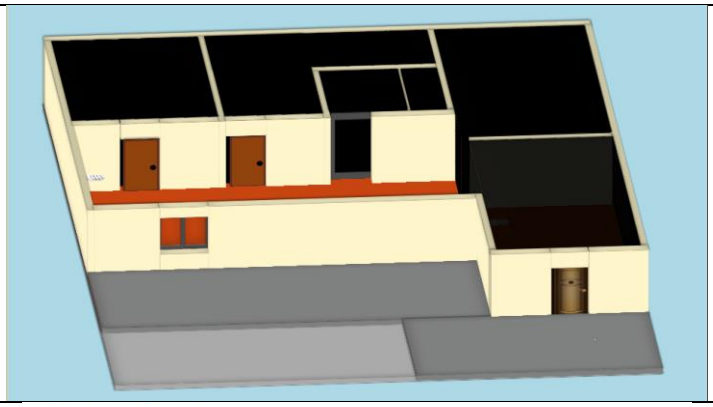
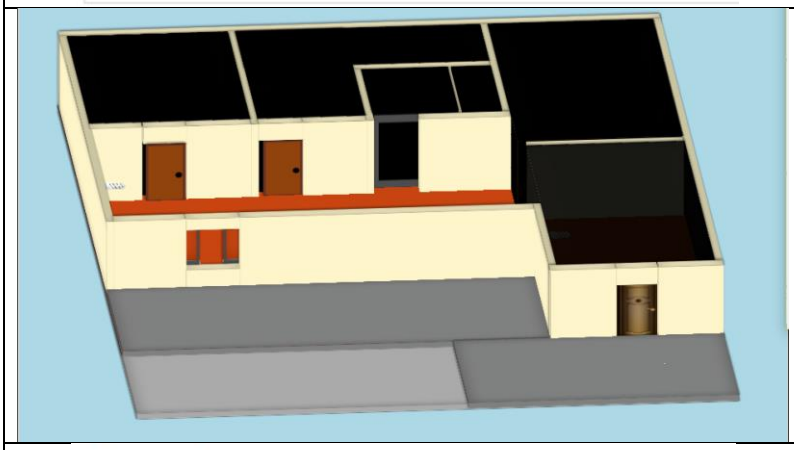
	Estado actual
	Comando emitido
	Estado final

Tabla 12. APAGADO DE UNA LUZ DEL AMBIENTE DOMÓTICO

En la Tabla 13, se muestra el procedimiento general, para cambiar el grado de apertura de una puerta o ventana, al interior del sistema domótico. Se asume que el comando de activación se encuentra vigente.

	Estado actual
Recognition Results Device name: rualuiscs Text: abrir un poco la ventana del corredor Confidence level: 95,91 Audio level trigger: 71 Audio level average: 18,73 Recognition time: 08:47:48.625	Comando emitido
	Estado p40
Recognition Results Device name: rualuiscs Text: abrir bastante la ventana del corredor Confidence level: 94,91 Audio level trigger: 100 Audio level average: 3,52 Recognition time: 08:49:18.698	Comando emitido

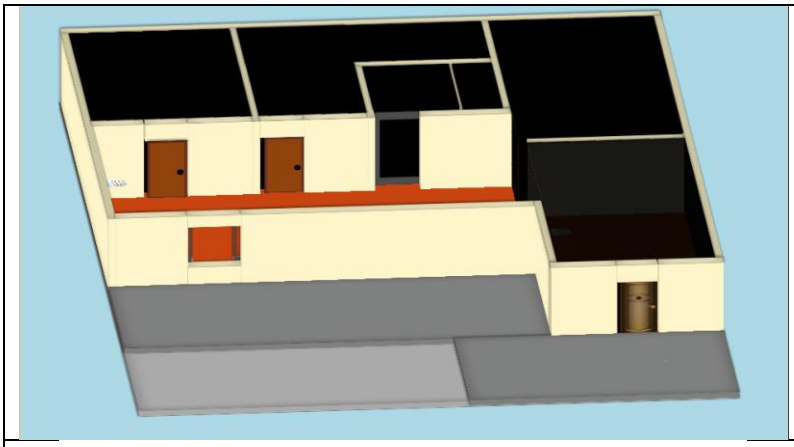
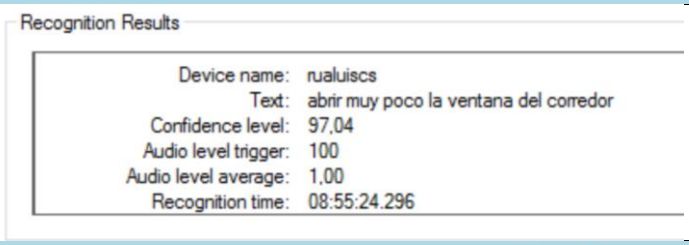
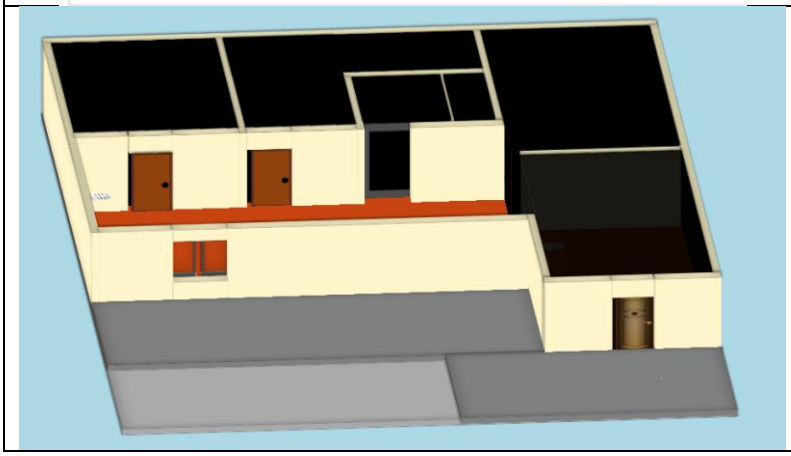
	Estado p80
	Comando emitido
	Estado p20

Tabla 13. DISTINTOS GRADOS DE APERTURA ALCANZADOS POR UNA VENTANA

La Tabla 13, al igual que las 3 tablas anteriores, presentan la evolución que presenta el ambiente domótico, en la medida en que se emite alguna de las entradas mostradas. Cabe resaltar que cuando el sistema se encuentra en algún estado dado, y este mismo estado es solicitado mediante una entrada, se emite una frase por parte del ambiente domótico simulado, indicado que el estado solicitado, es el actual, sin ejecutar acción distinta a esta.

La interfaz resultante no sólo puede ser controlada desde comandos de voz, gracias a la integración conseguida con *Arduino Micro* y el *Bitvoicer Server*, sino también a través del teclado de la computadora.

9. CONCLUSIONES

- Fue posible implementar un sistema de reconocimiento de habla aislada, donde la captura del habla, estaba a cargo del microcontrolador *Arduino Micro*

- Fue posible integrar las funciones desarrolladas por el *Arduino Micro*, con el *Bitvoicer Server*

- Se logró desarrollar una interfaz gráfica, donde pudieran verse las acciones tomadas por el sistema domótico, en el momento en que el interlocutor emitía una orden

- Herramientas como *Arduino Micro*, hacen transparente al programador protocolos como el *USB*, permitiendo al desarrollador enfocarse en la lógica del problema y su implementación

- Este trabajo de grado emplea el *Bitvoicer Server*, como sistema de reconocimiento de habla, sin embargo, podría emplearse algún otro software de reconocimiento de habla aislada para reemplazarlo, siempre y cuando contara con un protocolo basado en *USB* y compatible con *Arduino*, obteniendo resultados similares.

- El diseño de la interfaz gráfica, implicó el aprendizaje e implementación del preprocesador de *CSS SASS*, haciendo posible la reutilización de código, y consiguiendo desarrollar la interfaz, a partir de bloques

- Una forma fácil de integrar dos o más sistemas, es mediante eventos, de modo que mientras uno de los dos pueda emitirlos, los demás puedan capturarlos. Esta característica se aprovechó para enlazar el microcontrolador *Arduino Micro*, con el sistema operativo de la computadora mediante el navegador.

- El software *Bitvoicer Server*, permite definir tantos comandos como lo permita el almacenamiento de la computadora donde se instala, lo que permite definir cuantos comandos se desee. En este caso particular se comprobó su respuesta ante 25 comandos distintos.

- El procesamiento de habla del software *Bitvoicer Server*, se basa en el algoritmo de modelos ocultos de *Markov*

- 8 *bits* de resolución, son suficientes para representar una señal de voz, cuando la discriminación del interlocutor, no es un parámetro importante en un sistema de reconocimiento de voz de habla aislada, siempre y cuando se conozcan de antemano las posibles frases

10. TRABAJO FUTURO

Los resultados obtenidos en este trabajo de grado abren la puerta, a la posibilidad de controlar el sistema domótico de forma remota, debido a que los lenguajes empleados para el desarrollo de la interfaz gráfica son en su totalidad web. Así mismo le interfaz desarrollada puede representar el ambiente a controlar, visto desde la computadora del usuario, pudiendo validar el estado actual de sus electrodomésticos, y en el estado de puertas y luminarias.

Desde el punto de vista del sistema de reconocimiento, el *Bitvoicer Server*, podría reemplazarse por un sistema de reconocimiento distinto, siempre y cuando el software de reconocimiento empleado cuente con comunicación *USB*, con el *Arduino Micro*.

10.1. TRABAJO FUTURO *ARDUINO MICRO*

En este trabajo de grado, se llevó a cabo la simulación de un sistema domótico, mediante una interfaz desarrollada en *HTML5, CSS3* y *JavaScript*, de modo que el control de actuadores, la captura de señales desde sensores, y la presentación de información en pantallas y otros sistemas, no fue necesaria, ya que la simulación web, obvia todos estos elementos, permitiendo el movimiento de los elementos de la interfaz gráfica. Sin embargo, si lo que se desea es llevar esta implementación a

un sistema domótico real, bastaría con conectar el *Arduino*, con las interfaces necesarias, para comunicar las órdenes provenientes desde la computadora, hacia los periféricos correspondientes. Si bien bajo estas condiciones, la computadora, no ejecutaría la aplicación que simula el ambiente domótico, sigue siendo imprescindible su uso, debido a que en ella se ejecutaría el software de reconocimiento de voz, sea que se trate del *Bitvoicer* o alguna alternativa.

10.2. TRABAJO FUTURO SISTEMA DE RECONOCIMIENTO DE VOZ

Los sistemas de reconocimiento de habla aislada, presumen que el usuario, debe prestablecer los comandos que desea utilizar para operar un ambiente domótico determinado. El software empleado en este trabajo de grado, no es la excepción a esta norma. Esta situación impide la ejecución de comandos nuevos de forma espontánea, situación típica cuando se mueven elementos al interior del ambiente, cuando hay cambios en la infraestructura del mismo, o simplemente cuando el usuario cambia involuntariamente la forma en la que expresa el comando. Estas situaciones, pueden presentarse con frecuencia, limitando la utilidad del sistema de reconocimiento.

En el caso de que se desee obtener mejores resultados, y contemplar las problemáticas ya mencionadas, podría buscarse o desarrollarse una solución alternativa en software, capaz de realizar reconocimiento de habla continuo.

10.3. TRABAJO FUTURO INTERFAZ GRÁFICA

La interfaz gráfica desarrollada en este trabajo de grado, puede ser escalada a ambientes con un mayor número de habitaciones, plantas, y elementos interiores.

Esta situación depende netamente del ambiente que se desee simular. En este caso particular, y con el fin de comprobar su funcionamiento, se optó por una opción de una sola planta, 2 habitaciones, sala y comedor.

La mayor parte de los elementos ubicados al interior de la interfaz, se encuentran en perspectiva y están modelados en 3D, salvo las luminarias, que se encuentran modeladas a partir de una simple imagen bidimensional. Esto sugiere el potencial de mejorar los elementos al interior de la interfaz, a través de

modelado 3D, con tecnologías como *WebGL*, o directamente con *Tree.js*, tecnologías disponibles para la creación de contenidos tridimensionales. De esta manera podría conseguirse extender el número de posibles objetos a presentar al interior de la interfaz.

Una ventaja adicional de emplear diseño web para el desarrollo de este trabajo de grado, es la posibilidad de elaborar una *API* para la comunicación entre la interfaz gráfica de usuario y una aplicación móvil. Notificando al usuario, a través del mismo, acerca de algunas situaciones típicas, como la dejación de las luminarias encendidas, o de algunas de las puertas o llaves abierto.

11. GLOSARIO

AJAX	*Asynchronous JavasScript And XML*
API	*Application Programming Interface*
APP	*Application*
CERN	*Conseil Européen pour la Recherche Nucléaire*
CHKSUM	*Checksum*
DOM	*Document Object Model*
EEPROM	*Electrically Erasable Programmable Read – Only Memory*
HID	*Human Interface Device*
HTML	*HyperText Markup Language*
IETF	*Internet Engineering Task Force*
RAM	*Random Access Memory*
ROM	*Read – Only Memory*
PC	*Personal Computer*
RECLEN	*Record Length*
RECTYP	*Record Type*
SASS	*Syntactically Awesome Style Sheets*
UML	*Unified Modeling Language*
Ubi	*Ubiquitous computer*
USB	*Universal Serial Port*
W3	*World Wide Web*
WebGL	*Web Graphics Library*
XHTML	*Extensible HyperText Markup Language*

12. BIBLIOGRAFÍA

[1] D. Morton, “An interview Conducted by David Morton, Center of Electrical Engineering”, Available in the Center for the History of Electrical Engineering, june 22, 1996.

[2] A. Gibb,”NEW MEDIA ART, DESIGN, AND THE ARDUINO MICROCONTROLLER: A MALLEABLE TOOL”, M.S. thesis, Criticism and History of Art, Design and Architecture, Pratt Institute, 2010.

[3] M. Weisfeld. (2005). “The Evolution of Object-Orient Languages” [Online]. Available: http://www.developer.com/java/other/article.php/3493761/The-Evolution-of-Object-Oriented-Languages.htm

[4] S. Lorente, “Key issues regarding domotic applications”, Telecommunication Engineering School, Universidad Politécnica de Madrid, Madrid España, 0-7803-8482-2/04, 2004.

[5] T. Rastogi, “Home automation system design: the basics” [online] http://www.embedded.com/design/connectivity/4431025/Home-automation-system-design--the-basics

[6] R. Pieraccini, “From AUDREY to Siri” [Online].
Available: http://www.icsi.berkeley.edu/pubs/speech/audreytosiri12.pdf

[7] A. Newell, “HARPY, PRODUCTION SYSTEMS AND HUMAN COGNITION”, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1978.

[8] J. Eguiluz, “Introducción a XHTML” [Online].
Available: http://librosweb.es/libro/xhtml/, 2016

[9] M. Sharma, “Soft - Computational Techniques and Spectro - Temporal Features for TelephonicSpeech Recognition: ”, 2016 © IGI Global, 10.4018/978-1-4666-9474-3.ch006.

[10] J. Wilpon and L. Rabiner, “Application of Hidden Markov Models for Recognition of a Limited Set of Words in Unconstrained Speech”, AT&T Bell Laboratories, New Jersey.