Universidad Autónoma Metropolitana
Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Sistema para expresar las necesidades básicas en personas con trastornos del habla.

Modalidad: Proyecto Tecnológico

Trimestre 2017 – Primavera

Rebollar Gómez Alejandro
2113000241
arebollar24@gmail.com

Asesor

Dr. José Alejandro Reyes Ortiz
Profesor Asociado
jaro@correo.azc.uam.mx
Departamento de Sistemas

31 de Agosto de 2017

Declaratoria

Yo, Dr. José Alejandro Reyes Ortiz, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, Alejandro Rebollar Gómez, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Resumen

En este reporte se explicará la forma en que fue realizado un sistema para expresar las necesidades básicas en personas con trastornos del habla, el cual busca servir como apoyo de comunicación para este tipo de personas.

Serán descritos los procesos de diseño e implementación, tanto de hardware como de software, para el desarrollo del proyecto, así como las herramientas utilizadas y las pruebas realizadas.

Se incluyen todas las etapas del desarrollo desde el diseño hasta la construcción final, también se explica el funcionamiento del dispositivo desde la perspectiva del usuario final y se analizan los resultados obtenidos.

Tabla de contenido

Tabla de Figuras

1. Introducción

En la historia de la humanidad siempre han existido personas con trastornos del habla como la disfonía, el paladar hendido o el tartamudeo, las cuales encuentran dificultad con la comunicación verbal. Para poder superar estos obstáculos se han creado métodos de comunicación para estas personas como el lenguaje a señas, sin embargo, no es una forma de comunicación conocida por el resto de las personas. Esta problemática ha sido (y será) un tema relevante porque la comunicación es un elemento crucial para las relaciones humanas. En este aspecto, existe una necesidad de contar con dispositivos y herramientas que apoyen la comunicación idónea en personas con este tipo de trastornos.

Los dispositivos y/o aplicaciones, necesarios para la carencia descrita arriba, deben hacer uso de la tecnología actual para apoyar a las personas con trastornos del habla a comunicarse con cualquier persona, conozca o no el lenguaje a señas, aprovechando al máximo sus beneficios tecnológicos, tales como comunicación remota, procesamiento de datos, entre otras.

Por lo tanto, en este proyecto se propone crear un sistema que pueda ayudar a la comunidad de personas con trastornos del habla a tener una forma de comunicación rápida, sencilla de aprender y que todas las personas puedan entender. La solución que se propone es crear un sistema para expresión de necesidades básicas y frases comunes con el que el usuario interactúe de una manera que no requiera el uso de la voz y que el sistema reproduzca esta interacción en forma de voz.

2. Antecedentes

Proyectos terminales o de integración:

1.- Estudio y manejo de una pantalla Gráfica "Touchscreen" con Arduino y su aplicación a un teclado musical de una octava [1]: Este proyecto busca desarrollar un teclado de una octava en una pantalla táctil como interfaz de entrada; de manera similar, con este sistema se busca usar Arduino en conjunto con la interfaz de usuario, sin embargo, se usarán botones de membrana en lugar de una pantalla táctil.

2.- Dispositivo de iluminación LED para habitaciones con incorporación de electrónica digital y control desde Android por Bluetooth [2]: En este proyecto se usa una aplicación de Android para controlar una lámpara LED a distancia. De manera similar, se usará una aplicación de Android para comunicarse de manera inalámbrica usando tecnología Bluetooth, sin embargo, a diferencia de este proyecto que usa la aplicación como control a distancia, la aplicación solamente procesará el código enviado por el módulo codificador y fungirá como módulo reconocedor de patrones y sintetizador de voz.

Hardware y software comercial:

1.- Touch Voice App [3]: Esta es una aplicación que cuenta con botones con frases comunes para permitir que las personas con problemas del habla puedan comunicarse. Es similar a mi sistema porque tiene el mismo propósito, sin embargo, este sistema involucra un dispositivo físico con botones además de una aplicación que fungirá como módulo reconocedor de patrones y sintetizador de voz para procesar el código enviado por el módulo codificador.

2.-Google Text-to-Speech [4]: Esta aplicación desarrollada por Google cuenta con un algoritmo de síntesis de texto en voz que ha sido incluido en diversas aplicaciones de Android y en el traductor de Google. La API de Google Text-to-Speech será incluida en mi sistema para hacer la síntesis de voz requerida.

Artículos o proyectos de otras instituciones:

1.- B-Pack: un dispositivo *wearable* de detección basado en Bluetooth para reconocimiento de actividades de asistencia a pacientes [5]: "B-Pack" es un dispositivo que captura las actividades de cuidado de las enfermeras en hospitales reales. Este proyecto es similar a mi sistema porque utiliza la comunicación Bluetooth para comunicarse con un dispositivo maestro y es un dispositivo *wearable*.

2.- Hacía la medición del estrés con Smartphones y dispositivos *wearable*s durante las jornadas laborales y el sueño [6]: Este artículo habla sobre el uso de dispositivos *wearable* para medir los niveles de estrés de las personas usando micrófonos y sensores para entregar una estadística de salud y poder mejorar su calidad de vida. Con mi sistema se plantea el mismo fin: mejorar la calidad de vida de las personas que usen este dispositivo *wearable,* en este caso personas con trastornos del habla.

3. Justificación

El porcentaje de personas que saben interpretar el lenguaje a señas en México es muy reducido y las personas que lo utilizan, como medio de comunicación principal, son las que, normalmente, padecen un trastorno; ellas se encuentran expuestas a una situación de estrés al no poderse comunicar con otras personas que no entienden este lenguaje. [7]. Por otro lado, para las personas que no padecen estos trastornos también resulta una situación complicada, ya que no pueden comunicarse por el desconocimiento del lenguaje de señas, esto genera una comunicación errónea o inexistente. Por estos motivos, es necesaria la existencia de un método de comunicación no verbal alternativo que puedan usar las personas con trastornos del habla y que el resto de las personas puedan entender sin necesidad de aprender otro lenguaje.

En este trabajo se propone un sistema para expresión de necesidades básicas y frases comunes pensado para personas con trastornos del habla como la disfonía, el paladar hendido o el tartamudeo con la finalidad de facilitar su comunicación con el resto de la gente.

Con este sistema se obtendrá una interfaz humano-computadora física con sensores de presión y contará con un esquema de patrones de toques únicos que serán interpretados en textos y posteriormente, emitidos en forma de voz para facilitar el poder comunicarse. La curva de aprendizaje de los patrones no será muy marcada, con el fin de que el usuario final pueda comunicarse rápidamente sin tener que consultar la relación de patrones y mensajes.

4. Objetivos

4.1. Objetivo general

Desarrollar un sistema que permita expresar las necesidades básicas en personas con trastornos del habla utilizando una interfaz táctil.

4.2. Objetivos particulares

- Diseñar e implementar un módulo sensorial (interfaz táctil) para permitir la entrada de datos del usuario.
- Desarrollar un módulo codificador que procese la información recibida del módulo sensorial (interfaz táctil).
- Reconocer un conjunto de patrones textuales para cada necesidad a partir de la información recibida.
- Implementar un módulo sintetizador de voz que reproduzca el mensaje deseado a partir de los textos.

5. Marco Teórico

La electrónica y la computación han llegado para quedarse; y con ellas los circuitos electrónicos que son la base de la mayoría de las tecnologías en el mundo La electrónica y la computación son parte de nuestras vidas, por ello es necesario apostar por la creación de este tipo de herramientas para el futuro, pero para ello debemos conocer los conceptos, técnicas y leyes que rigen el mundo de la electrónica y la informática.

5.1. Electrónica

La electricidad se puede comprender como un tipo de energía la cual, mediante una manipulación correcta, puede generar efectos mecánicos, luminosos, caloríficos y químicos. La manipulación de esta energía mediante dispositivos que son capaces de recibir, procesar o enviar información para el uso de la industria se le conoce como **electrónica.**

5.2. Circuito eléctrico

Existen múltiples componentes capaces de manipular la electricidad, éstos se organizan dentro de un sistema para lograr un objetivo general. A este sistema se le conoce como **circuito eléctrico**.

Un circuito eléctrico básico consta de un mínimo de cuatro elementos que permiten el flujo y la utilización de los electrones:

1. La fuente de energía.
2. Los conductores
3. La carga.
4. El dispositivo de control

5.3. La corriente, el voltaje y la resistencia

Para entender cómo se realiza la manipulación de la energía, en específico los electrones se deben comprender algunas variables de interés, por ejemplo: el voltaje es la fuerza electromotriz que permite que los electrones se muevan a través de un circuito eléctrico, se mide en volts. La corriente es la cantidad de electrones que se mueven en un segundo dentro de un circuito eléctrico, el movimiento de electrones se mide en amperes. Un ampere corresponde al movimiento de 6.28x10^18 electrones en un segundo. Y la resistencia es la oposición al flujo de corriente que se genera por los choques entre electrones, esto reduce el número de electrones que se mueven a través un conductor y su unidad de medida es el ohm (Ω).

5.4. Tipos de corriente y frecuencia

La corriente (flujo de electrones) puede ser de dos tipos: corriente continua y corriente alterna. La corriente continua se produce cuando un circuito tiene una fuente de voltaje constante, es decir, las cargas en los polos negativo y positivo no cambian a través de tiempo. Y la corriente alterna se produce cuando un circuito tiene una fuente de voltaje cuya polaridad de las cargas en los polos negativo y positivo se invierten en un determinado tiempo. Al número de veces que un ciclo de corriente alterna puede completarse en un segundo se le llama frecuencia y se mide en Hertz (Hz).

5.5. Ley de ohm

Esta ley describe formalmente la relación entre las variables voltaje, corriente, energía y potencia. Dentro de los circuitos eléctricos, cuando los electrones han salido de la fuente, la única oposición al flujo de estos es la resistencia, además, este flujo tiene un determinado voltaje y corriente, por ello existen relaciones entre ellas. Estas relaciones, según la ley de ohm, se determinan de la siguiente forma:

1. El voltaje necesario para establecer cierta intensidad de corriente a través de un circuito es igual al producto de la corriente y de la resistencia del circuito como se muestra en la ecuación:

$V = I \times R$

Donde:
V= voltaje, I= corriente y R= resistencia.

2. La intensidad de corriente de un circuito es igual al voltaje que se aplica al circuito, dividido entre la resistencia del circuito como se muestra en la ecuación:

$I = V / R$

3. La resistencia de un circuito es igual al voltaje que se aplica al circuito, dividido entre la cantidad de corriente del circuito como se muestra en la ecuación:

$R = V / I$

4.- La potencia de un circuito es igual al voltaje que se aplica al circuito, multiplicado por la cantidad de corriente del circuito como se muestra en la ecuación:

$P = V \times I$

El flujo de la corriente se puede dar de dos formas, dependiendo del tipo de circuito: circuito en serie o circuito en paralelo.

5.6. Circuito en serie y paralelo

En el **circuito en serie**, los elementos se conectan uno después de otro. De esta forma sólo existe una ruta por la que los electrones pueden circular. Se utiliza para controlar y proteger sistemas eléctricos. En un circuito de este tipo la suma de las caídas de voltaje en el circuito es igual al voltaje aplicado, este hecho se conoce como la **ley de voltaje de Kirchhoff**.

En un **circuito en paralelo**, las cargas se conectan de forma ramificada, si una de las ramas se desconecta o abre las ramas restantes continuaran operando.

5.7. Componentes de circuitos eléctricos

Dentro de estos circuitos podemos encontrar algunos componentes electrónicos que determinan su funcionamiento, por ejemplo:

El diodo rectificador. Es un dispositivo semiconductor que permite el paso de la corriente eléctrica en una única dirección.

El diodo emisor de Luz (LED). Es un dispositivo semiconductor que transforma la energía eléctrica en luz de un espectro determinado cuando se polariza de forma correcta.

El resistor. Es un componente eléctrico diseñado para introducir una resistencia eléctrica determinada entre dos puntos de un circuito eléctrico. Comúnmente se conocen como resistencias.

El transistor. Es un dispositivo electrónico especialmente diseñado para entregar una señal de salida en respuesta a una señal de entrada, puede cumplir las funciones de conmutador, oscilador o amplificador.

Estos componentes han revolucionado la industria de la electrónica y han dado lugar a la aparición de otras disciplinas como la electrónica digital.

5.8. Electrónica Digital

La electrónica digital es la rama más moderna de la electrónica y que evoluciona más rápidamente. Su aplicación es la construcción de dispositivos electrónicos en los que la información está codificada en estados discretos. Los enormes avances tecnológicos han logrado que se desarrollen componentes electrónicos mucho más avanzados, capaces de realizar tareas complejas y formar parte de sistemas robustos que, a su vez, logran más avances tecnológicos. Algunos componentes recientemente creados con electrónica digital son, por ejemplo:

5.8.1. Los microcontroladores

Son circuitos integrados programables, capaces de ejecutar las ordenes grabadas en su memoria. Contienen en su interior tres unidades funcionales:

Unidad central de procesamiento.

Memoria.

Periféricos de entrada y salida.

5.8.2. Los transmisores

Son instrumentos que captan las señales y son capaces de transmitirlas a distancia hacia un receptor para la transferencia de información. Pueden ser de varios tipos e implementar distintas tecnologías como:

Wifi.

Bluetooth.

Radio frecuencia determinada.

Zigbee.

Lifi.

5.8.3. Los módulos digitales

Son dispositivos que se encargan de aprovechar la unión de varios dispositivos electrónicos y realizar tareas específicas. Su objetivo puede ser de distintos tipos:

Módulos de control: **El módulo de control Arduino Nano** por ejemplo, es una plataforma de prototipos electrónicos basados en hardware y software de código abierto. Integra un microcontrolador de la marca ATMEL atmega328 a una velocidad de 16 MHz. [8]

Módulos de transmisión: **El módulo bluetooth HC-06** por ejemplo, es un transmisor de tecnología bluetooth V2.0 diseñado para trabajar como servidor configurable mediante comandos AT (parecidos a los comandos de una terminal “tonta”) y con una frecuencia de 2.4 GHz, con un alcance de 10 metros a un voltaje de 3.3v.

Algunos de los sistemas electrónicos más complejos creados hasta el momento son:

Las computadoras: Son máquinas electrónicas capaces de almacenar información y tratarla automáticamente mediante operaciones matemáticas y lógicas controladas por programas informáticos.

Los teléfonos inteligentes o Smartphone: Son dispositivos semejantes a minicomputadoras y con una mayor conectividad que un teléfono móvil convencional, el término ”inteligente” se utiliza con fines comerciales y hace referencia a la capacidad de usarse como un computador personal.

5.9. Hardware

Dentro de un sistema digital o informático, es el conjunto de elementos tangibles o físicos (procesador, dispositivos de almacenamiento permanente, memoria principal, etc.).

5.9.1. Desarrollo de hardware

Para cada uno de los componentes electrónicos que utilicemos, podemos usar internet para encontrar su documentación (datasheet) que el fabricante pone a disposición publica y así saber los detalles pertinentes acerca de la especificación técnica de determinado componente. Esto facilita la utilización de un componente dentro de un circuito electrónico y reduce el tiempo de diseño ya que no es necesario probar los componentes uno a uno para saber acerca de su funcionamiento.

Para el diseño de circuitos electrónicos existe software que nos permiten desarrollar diagramas, simular el funcionamiento del circuito, detectar errores y diseñar un circuito impreso. Esto permite crear circuitos más confiables y que se pueden probar de forma virtual antes de fabricarlos. EasyEda por ejemplo, es un entorno web para el desarrollo de circuitos electrónicos, placas de circuito impreso y simulación basado en la web y la nube.

Existen distintas normas que indican como se debe realizar el desarrollo de circuitos impresos y como se pueden evitar errores que no son predecibles mediante los sistemas de diseño de circuitos, además de incluir metodologías para la realización de circuitos impresos ya sea de forma rudimentaria o de forma industrial.

5.10. Software

Dentro de un sistema digital o informático, hace referencia a los códigos o programas que permiten el uso y administración de los dispositivos de hardware. Es la parte intangible de un computador o Smartphone, como los sistemas operativos. Android, por ejemplo, es un sistema operativo basado en el núcleo Linux, diseñado para dispositivos móviles con pantalla táctil; este sistema operativo es propiedad de Google inc.

Los sistemas informáticos actuales son complejos, por ello han surgido metodologías, normas, estándares, técnicas, documentos y herramientas que nos permiten sobrellevar la gran carga de trabajo y abstracción que la construcción de un sistema electrónico avanzado requiere.

5.10.1. Lenguajes de programación

Para el desarrollo de programas existen lenguajes de programación que permiten, mediante un lenguaje formal, describir procedimientos, actividades o acciones que un sistema informático o digital debe realizar. La mayoría de lenguajes de programación integran un conjunto de instrucciones cuyo léxico es similar al lenguaje común para hacerlo más entendible para el programador y al conjunto de instrucciones se le suele llamar código fuente. Algunos de los más conocidos son:

Java: es un lenguaje de programación de propósito general y orientado a objetos; los programas generados con este lenguaje requieren de una máquina virtual lo que permite que el mismo programa pueda correr sobre diferentes dispositivos.

C: es un lenguaje de programación estructurado que a pesar de ser de alto nivel permite usar instrucciones de muy bajo nivel. Este lenguaje de programación está orientado a la implementación de sistemas operativos.

5.10.2. Entornos de desarrollo

Existen entornos de desarrollo integrados (IDE) que son aplicaciones que proporcionan servicios para facilitar el desarrollo de software. Por ejemplo:

IDE Arduino: es un entorno de desarrollo de sistemas para microcontroladores bajo la plataforma de hardware libre Arduino y permite, mediante el uso de un compilador y un depurador, la programación en un lenguaje propio de Arduino basado en Wiring2 (pequeña arquitectura de hardware parecida a la de un computador personal) y el lenguaje de programación C.

APP Inventor 2: es un entorno de desarrollo basado en programación orientado a eventos, hace uso de la librería Open Blocks de Java y permite la creación de aplicaciones para dispositivos con el sistema operativo Android.

Se han creado metodologías de desarrollo de software que ayudan a identificar los elementos clave que deben incluirse en el desarrollo de un determinado programa además se han descrito las reglas necesarias para el trabajo en equipo y brindar soporte para la generación de la documentación necesaria para el desarrollo de proyectos. Por ejemplo:

5.10.3. **Los diagramas de flujo**

Nos permite abstraer de forma gráfica los elementos de un proceso o algoritmo para poder ser comprendido y posteriormente programado. LucidChart por ejemplo, es un entorno web que permite utilizar una versión de prueba (30 días) o de pago para realizar diagramas de flujo que permitan abstraer un procedimiento antes de programarlo.

5.10.4. Los casos de uso de texto

Es una metodología enfocada al descubrimiento de los requerimientos de un sistema. En específico proveen el entendimiento de las funciones de un escenario principal de éxito y el análisis de los flujos alternativos, es decir, nos permite saber lo que realmente se necesita en un sistema complejo.

5.10.5. Paradigmas de programación

Por otra parte, han surgido paradigmas de programación que son distintos enfoques o filosofías acerca de cómo generar soluciones de software. Nos brindan una forma específica de abstraer los elementos del problema, así como la forma de relacionarlos e integrarlos en una solución. Un paradigma de programación puede determinar la estructura y abstracción de uno o varios lenguajes de programación, por ejemplo:

IDE de Arduino hace uso de programación estructurada que considera una ejecución secuencia de varias subrutinas lo cual permite mejorar la calidad, claridad y tiempo de desarrollo.

IDE App Inventor 2 hace uso de programación orientada a eventos en la que el flujo de ejecución del programa está determinado por los sucesos que ocurran en el sistema y definidos por el usuario.

5.11. Autómatas finitos

Los autómatas finitos son modelos computacionales que permiten realizar cómputos en forma automática sobre una entrada dada para producir una salida [9]. Estos autómatas son, formalmente, una 5-tupla de la siguiente forma:

(Q, Σ, q0, δ, F) de donde,

Q es un conjunto de estados finito.
Σ es un alfabeto formado por un conjunto finito de caracteres.
q0 es el estado inicial, que es el estado a partir del cual el autómata comienza leer la entrada.
δ es la función de transición:
F es el conjunto de estados finales o de aceptación del autómata.

La entrada: es una cadena de caracteres pertenecientes al alfabeto.

5.11.1. Autómatas finitos deterministas (AFD)

Estos autómatas son autómatas finitos con la característica de que para cada estado $q \in Q$ en que se encuentre el autómata, y con cualquier símbolo $a \in \Sigma$ del

alfabeto leído, existe siempre a lo más una transición posible δ(*q*,*a*).

En un autómata de este tipo no pueden darse ninguno de los siguientes casos:

- Que existan dos transiciones en donde una misma entrada lleve a estados diferentes (δ(q,a)=q1 y δ(q,a)=q2, siendo q1 ≠ q2)
- Que existan transiciones nulas (δ(*q*, ε)); es decir, que hayan transiciones que no produzcan entrada alguna.

6. Desarrollo del proyecto

Se tendrán dos dispositivos: un dispositivo cliente y un servidor.

El cliente será un dispositivo con sistema operativo Android que contendrá el módulo reconocedor de patrones textuales y el módulo sintetizador de voz. El servidor contendrá los módulos sensorial y codificador y estará conformado por 5 botones, la banda del usuario, una tarjeta Arduino Nano y un módulo Bluetooth.

De forma general, la interacción entre ambos dispositivos será la siguiente:

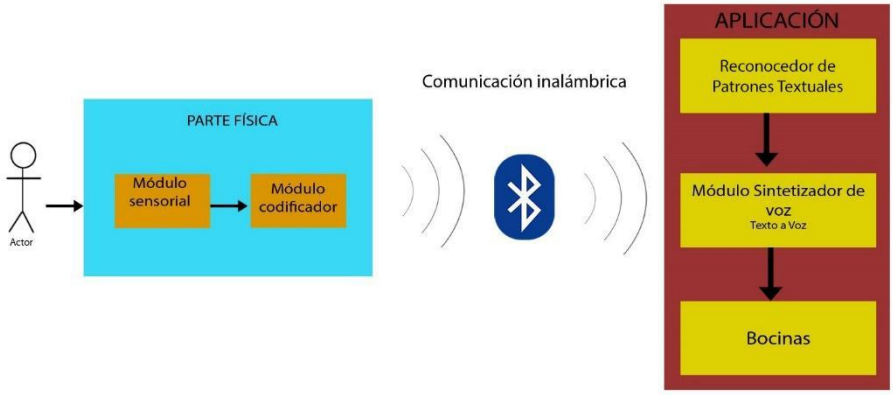


Figura 1. Interacción cliente-servidor

El módulo sensorial genera señales digitales mediante la presión de los botones incluidos en él mismo y las envía al módulo codificador, el cual procesa estas señales y forma un patrón textual. Este patrón es enviado vía Bluetooth al módulo reconocedor de patrones textuales ubicado en el dispositivo cliente. Este módulo cuenta con una tabla de asociación patrón-frase con la cual compara el patrón recibido y obtiene la frase asociada a él. Esta frase es después enviada al módulo sintetizador de voz, el cual, mediante los algoritmos incluidos en la API *Google Text-To-Speech,* procesa la frase y genera un archivo de audio que "habla" dicha frase.

De manera específica, el proceso completo se describe enseguida:

1. El módulo codificador ubicado en el servidor se queda a la espera de la entrada del usuario (presión de botones).
2. Si el usuario presiona un botón y lo mantiene presionado el módulo codificador comienza a formar el patrón con una letra (A, B, C, D o E) que indica una categoría y pasa al siguiente estado (explicado a detalle en la sección de diseño del módulo codificador). En caso contrario (el usuario no presiona nada, el usuario suelta el botón), el módulo codificador sigue en espera de la entrada.
3. En el estado actual, presionar uno o más botones forma un código patrón que es enviado al módulo intérprete de patrones textuales vía Bluetooth, el cual se encontraba

en espera de dicho código.

4. Si el módulo intérprete reconoce ese código dentro de su tabla de asociaciones, busca la frase asociado a dicho código y lo envía al módulo sintetizador de voz.
5. El módulo sintetizador de voz mediante los algoritmos incluidos en la API *Google Text-To-Speech,* procesa la frase y genera un archivo de audio que reproduce dicha frase por las bocinas del dispositivo cliente.

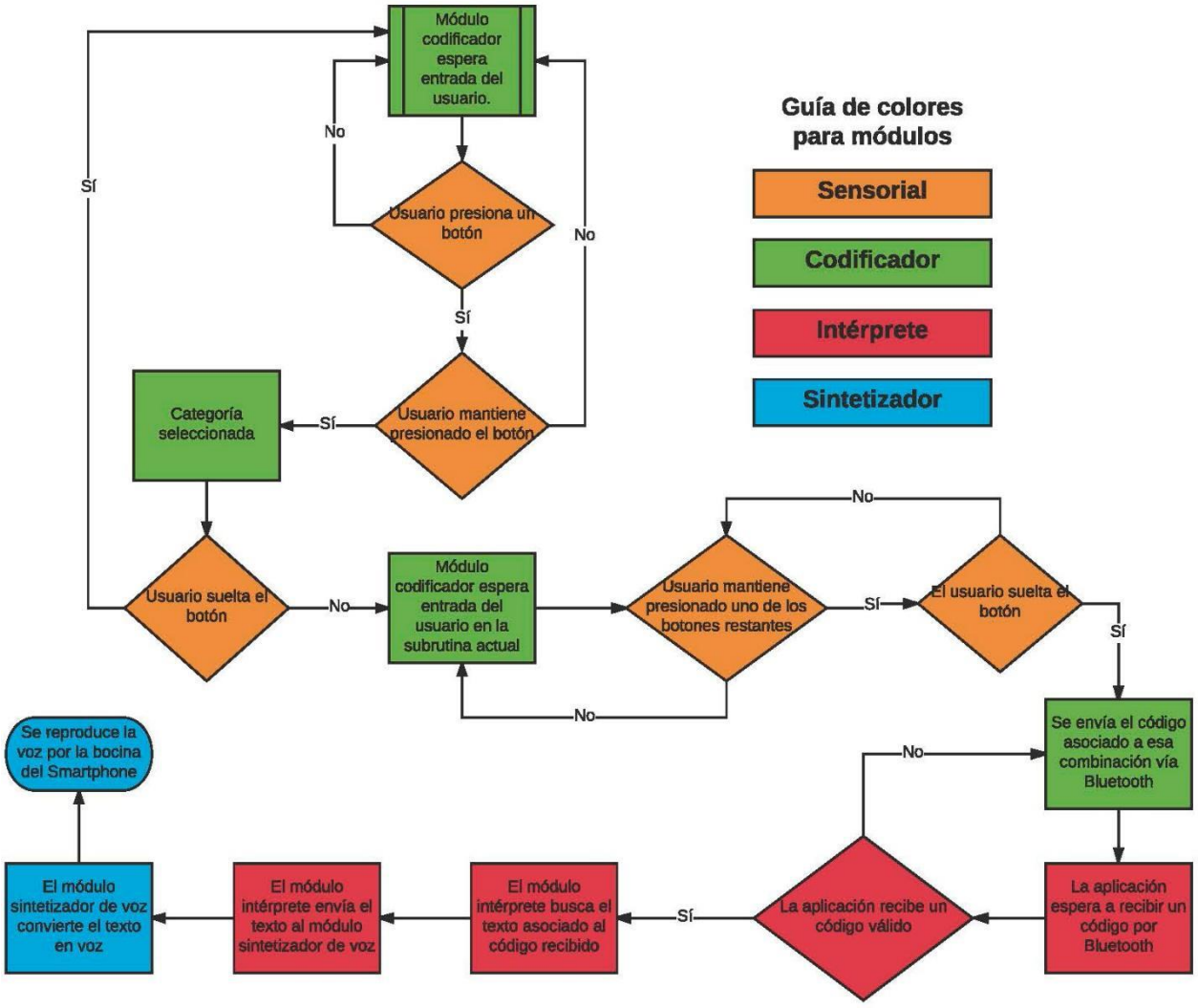


Figura 2 Diagrama de flujo del dispositive.

A continuación, se describe el diseño y la implementación de los dispositivos cliente y servidor.

6.1. El servidor

El servidor está formado por la banda de botones, la tarjeta de desarrollo Arduino Nano y el transmisor Bluetooth HC-06. Consta de dos módulos: el módulo sensorial y el módulo codificador. Su diseño e implementación se describen a continuación.

6.1.1. El módulo sensorial

Este módulo es el encargado de enviar una señal digital al módulo codificador mediante la presión de una combinación de botones. Está conformado por la banda táctil y los botones contenidos en ella. Se tomó una correa ajustable al tamaño de la mano y se adaptaron 5 botones de tal forma que pudieran ser presionados sin mayor esfuerzo por el portador.

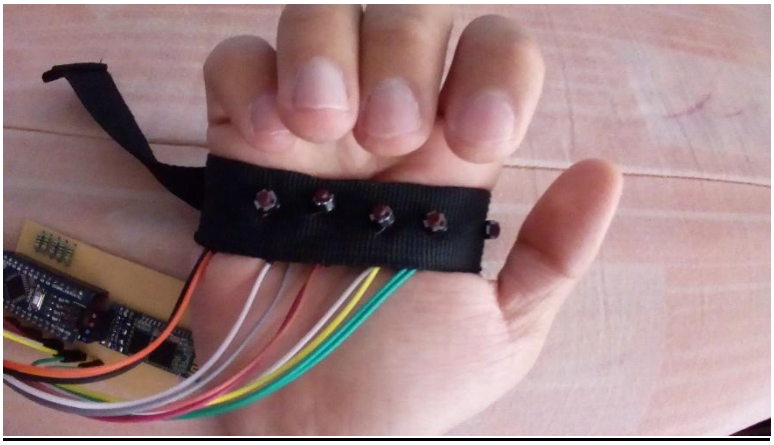


Figura 3 Módulo sensorial (sin presionar).

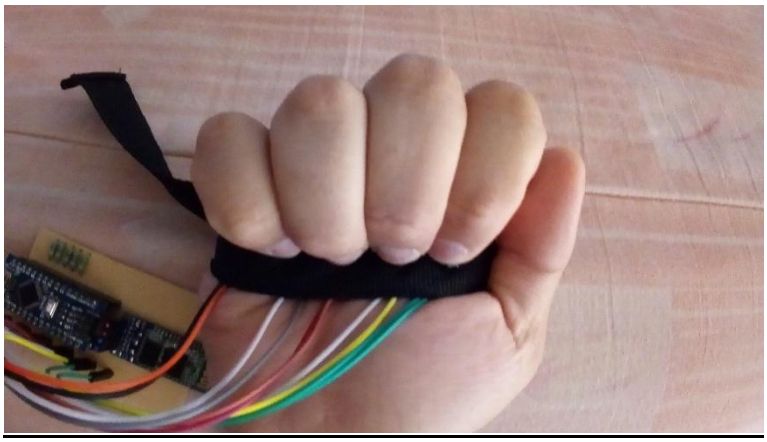


Figura 4. Módulo sensorial (presionado)

6.1.2. El módulo codificador

Este módulo es el encargado de recibir las señales digitales enviadas por el módulo sensorial y hacer uso de ellas para crear un patrón y enviarlo vía bluetooth al módulo reconocedor de patrones textuales en el cliente. Este módulo se encuentra conformado por la tarjeta de desarrollo Arduino Nano y el transmisor inalámbrico Bluetooth HC-06.

Figura 5. Arduino Nano cargado con el código del módulo codificador

6.1.2.1. Diseño

Se diseñó un autómata finito determinista para ser implementado como máquina de estados en código Arduino C. Se tiene el estado inicial S0, el cuál necesita una entrada para generar una transición hacia el estado S1. En este caso, mantener presionado uno de los 5 botones, genera una entrada del tipo A, B, C, D o E según la siguiente tabla:

Entrada	Botón (dedo)
A	1 (Pulgar)
B	2 (Índice)
C	3 (Medio)
D	4 (Anular)
E	5 (Meñique)

En caso de soltar el botón, se inicializa el autómata regresando al estado S0. Al pasar al estado S1, se hace una reasignación de botones, dejando de contar el botón presionado.

Estando en el estado S1 se produce una transición a S2 al presionar, o no, el botón 1 de acuerdo a la siguiente tabla:

Si la entrada inicial fue	Se asigna el botón	Al dedo
A	1	Índice
B	1	Pulgar
C	1	Pulgar
D	1	Pulgar
E	1	Pulgar

Estando en el estado S2 se produce una transición a S3 al presionar, o no, el botón 2 de acuerdo a la siguiente tabla:

Si la entrada inicial fue	Se asigna el botón	Al dedo
A	2	Medio
B	2	Medio
C	2	Índice
D	2	Índice
E	2	Índice

Estando en el estado S3 se produce una transición a S4 al presionar, o no, el botón 3 de acuerdo a la siguiente tabla:

Si la entrada inicial fue	Se asigna el botón	Al dedo
A	3	Anular
B	3	Anular
C	3	Anular
D	3	Medio
E	3	Medio

Finalmente, estando en el estado S4 se produce una transición a SF al presionar, o no, el botón 4 de acuerdo a la siguiente tabla:

Si la entrada inicial fue	Se asigna el botón	Al dedo
A	4	Meñique
B	4	Meñique
C	4	Meñique
D	4	Meñique
E	4	Anular

A continuación, se muestra esquematizado el autómata descrito previamente:

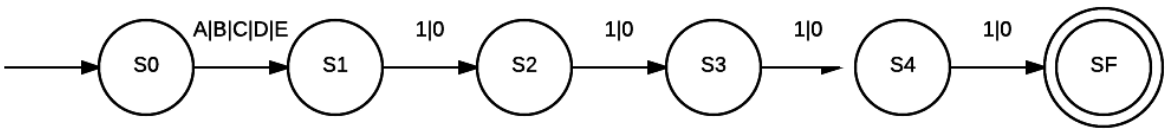


Figura 6. AFD para el módulo codificador

Este autómata puede generar salidas del tipo Xnnnn, donde:

X = A|B|C|D|E
n = 0|1

6.1.2.2. *Implementación*

Se generó un código de testeo para realizar pruebas utilizando únicamente el servidor y el dispositivo Android. Haciendo uso de una protoboard, se implementó el servidor añadiendo 5 LEDs que representaban la presión de cada botón y el transmisor bluetooth, junto con la aplicación “Serial Bluetooth Terminal” para Android, con la que podíamos captar los mensajes enviados de manera serial por el transmisor Bluetooth del servidor.

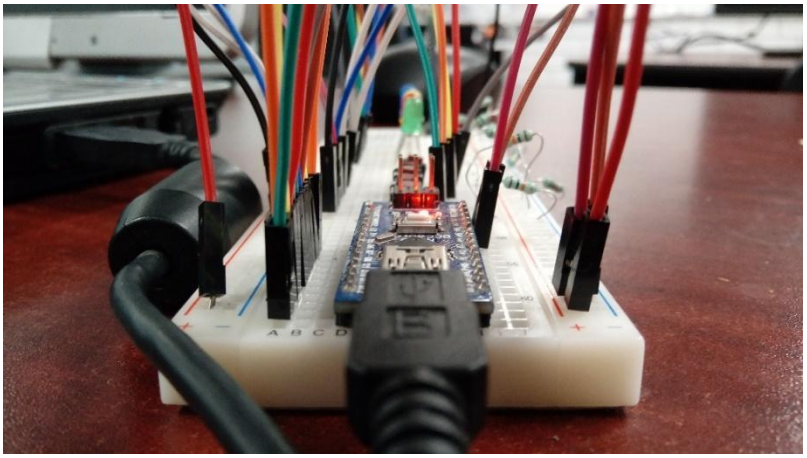


Figura 7. Servidor con el código de prueba.

El código de prueba es similar al código final, con la diferencia de que las señales se mandan a una salida digital correspondiente a cada LED además de al pin TX. Se añadió, además, la función “Test” para probar que se creaban y enviaban correctamente los patrones. Se probaron, en un principio, solamente dos patrones: A0001 y B1010.

Primero, se hace una comparación del patrón recibido con el patrón de prueba (en este caso, A0001). En caso de ser iguales, se manda a llamar a la función flashLED con los siguientes parámetros:

flashLED (int n, int LEDPIN)

Donde n es el número de veces que parpadea el LED y LEDPIN es el número de pin asociado al LED que queremos encender.

Después se manda llamar a la función printMacro con los siguientes parámetros:

printMacro(String string)

Donde string es el patrón que se va a enviar vía Bluetooth a la aplicación “Serial Bluetooth Terminal”. Esto nos permitirá saber si el patrón enviado tiene basura, se manda cortado, o si efectivamente se manda correctamente.

Después de inicializa la variable str con el valor “00000”, el cual no está asociado a ningún patrón. Inmediatamente se vuelve a enviar str con ceros, con el fin de validar el inicio y el fin del patrón en la terminal serial en Android.

En caso contrario de que el patrón no sea igual a A0001, se hace la comparación con B0101 y,

en caso de ser igual, se repite el proceso, pero con un LED diferente.

En el caso de que no coincida con ninguno de ambos patrones, se envía el código tal cual y se enciende un tercer LED que indica que no corresponde a ninguno de los patrones anteriores.

A continuación, se ejemplifica el código mencionado previamente:

```
void Test(){
  if(str == "A0001"){
    flashLED(1,ledPin1);
    printMacro(str);
    str = "00000";
    printMacro(str);
  }

  else if(str == "B1010"){
    flashLED(1,ledPin2);
    printMacro(str);
    str = "00000";
    printMacro(str);
  }

  else{
    flashLED(1,ledPin4);
    printMacro(str);
    str = "00000";
    printMacro(str);
  }
}

void flashLED (int n, int LEDPIN) {
  for (int i=1; i<=n; i++) {
    digitalWrite (LEDPIN, HIGH);
    delay (FLASH_RATE);
    digitalWrite (LEDPIN, LOW);
    delay(FLASH_RATE);
  };
  return;
}

void printMacro(String string){
  if(Serial){
    Serial.print(string);
    }
}
```

Estas pruebas permitieron encontrar errores como entradas “sucias” (con símbolos “basura” que entorpecen el procesamiento del patrón por parte del cliente), patrones incompletos o de tamaños diferentes al establecido.
El código final se encuentra en el apéndice, al final de este documento.

6.2. El cliente

El cliente está compuesto por dos módulos: reconocedor de patrones textuales y sintetizador de voz. Ambos módulos viven en una aplicación de Android llamada “ProtoPattern”, creada en App Inventor 2 exclusivamente para este proyecto. La aplicación, los módulos, su diseño e implementación serán explicados a detalle a continuación.

6.2.1. El módulo reconocedor de patrones textuales

Este módulo reside en la aplicación de Android “ProtoPattern”. Es el módulo encargado de reconocer el patrón enviado vía bluetooth por el servidor al cliente, interpretarlo y enviar un texto asociado a este patrón al módulo sintetizador de voz de manera interna.

Este módulo cuenta con una tabla de asociación patrón-texto para poder hacer su trabajo. En la Figura 8 se muestra un fragmento de esta tabla.

Figura 8. Tabla de asociación patrón-texto "in-code".

6.2.2. El módulo sintetizador de voz

Este módulo reside en la aplicación de Android “ProtoPattern”. Es el encargado de recibir el texto enviado por el módulo reconocedor de patrones textuales y convertirlo a voz mediante el uso de la API Text-to-Speech de Google.

En la Figura 9 podemos observar una pieza de código en donde se manda a llamar el objeto “TextToSpeech1” para ejecutar la función “Speak message” con la variable global “A11111”, la cual tiene asociado el texto “No sé”. Esto reproducirá dicho texto en forma de voz.

Figura 9. Uso de la API Text-To-Speech.

6.2.3. Diseño e implementación

Para crear la aplicación “ProtoPattern” que contiene ambos módulos del cliente, se utilizó el entorno de desarrollo App Inventor 2 que usa el paradigma de programación visual orientado a eventos mediante código de bloques. Este paradigma nos permite utilizar objetos con funciones específicas y crear eventos que, al desencadenarse, nos permiten usar una o varias funciones de los objetos involucrados en dicho evento.

En la figura 10 podemos ver el entorno de desarrollo de App Inventor 2.

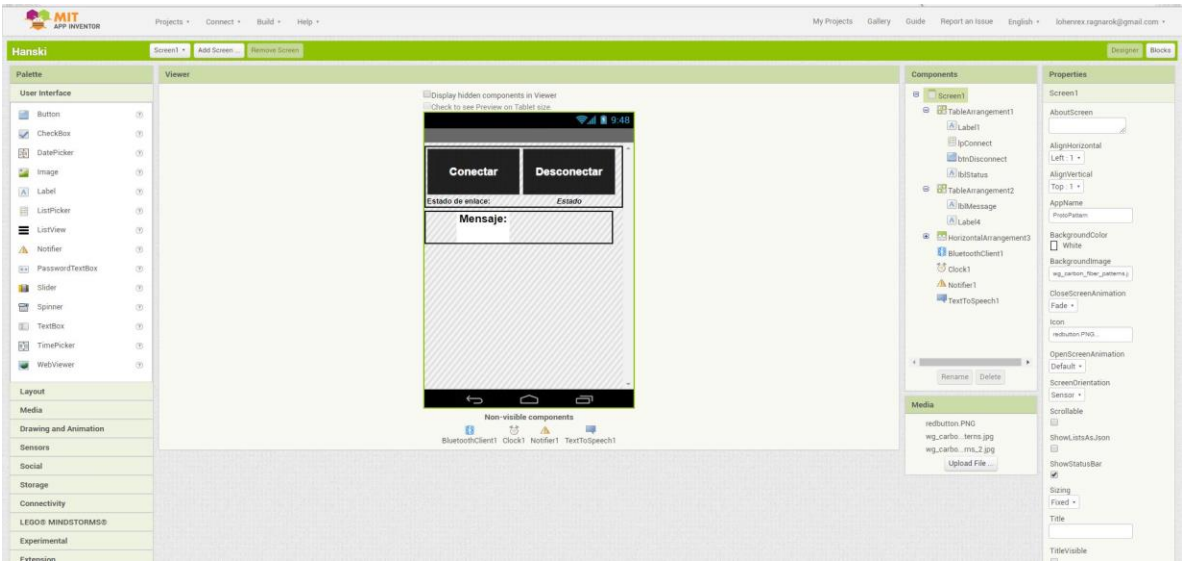


Figura 10. Entorno de desarrollo de App Inventor 2.

Como pudimos ver anteriormente en la Figura 8, la tabla de asociación patrón-texto es simplemente una asociación de una cadena de caracteres a una variable global de tipo implícito String, la cual lleva por nombre el patrón asociado. El hecho de que sea una variable de tipo “implícito” quiere decir que App Inventor 2 para “facilitar” el desarrollo, es flexible en cuanto al tipo de variable a usar. De la misma manera que en Python o PHP, estas variables no tienen un tipo fijo.

El código de bloques de la aplicación se puede encontrar en el apéndice 2. En él podemos observar que hay una sección de código que se repite constantemente y es la parte “nuclear” de la aplicación. En la Figura 11 podemos ver dicha pieza de código, la cual será explicada a continuación.

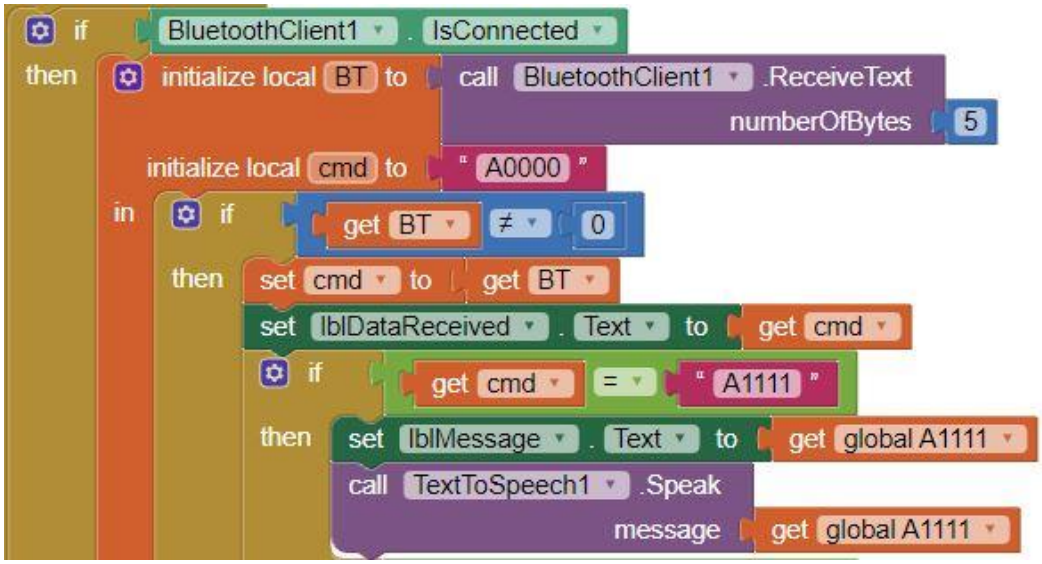


Figura 11. Pieza de código principal del cliente.

Primero se revisa que el objeto de bluetooth “BluetoothClient1” se encuentre encendido. Esto es, que este activado el bluetooth en el Smartphone. Si es así, se inicializa la variable local “BT”, la cual va a almacenar los datos recibidos por bluetooth en forma de texto, limitados a 5 bytes. Esto es el patrón.

Se inicializa la variable local “cmd” con la cadena “A0000” ya que no corresponde a ningún patrón. Después se hace una comparación: Si la variable “BT”, que está fungiendo como buffer de datos recibidos vía Bluetooth, “tiene algo” (es diferente de cero), entonces se asigna el contenido de “BT” a “cmd”.

Después, de introduce una comparación extensa entre la variable “cmd” y los 75 patrones posibles ((2`4 - 1) * 5 patrones). Si coincide con alguno, se manda a llamar la variable “TextToSpeech1”, la cual hace uso de la API de Google “Text-to-Speech” para procesar la variable asociada al patrón coincidente y reproduce el contenido de dicha variable en forma de voz.

Finalmente, se vuelve a inicializar la variable “cmd” y se repite el proceso de manera infinita hasta cerrar la aplicación.

6.3. El circuito impreso

6.3.1. Diseño

Se utilizó el entorno de desarrollo EasyEda (software para el diseño de circuitos) para realizar el diagrama del circuito electrónico.

El procedimiento realizado es el siguiente:

1. Se colocan los componentes electrónicos en el área de diseño, se les asigna su orientación y posición
2. Se unen las entradas y salidas de los componentes electrónicos con la herramienta de alambrado manual.
3. Se exporta el diseño final a un formato de imagen o pdf, ya que es una herramienta de abstracción del circuito y se debe poder revisar en cualquier momento sin necesidad de abrir el entorno de desarrollo.

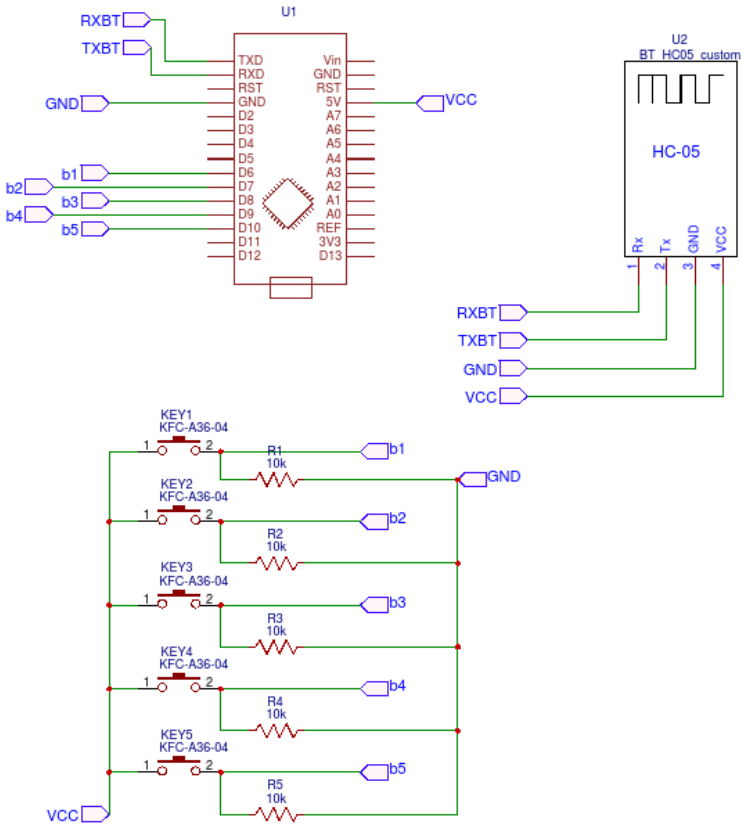


Figura 12. Diseño final del circuito.

6.3.2. ***Construcción***

Se utilizó el entorno de desarrollo de EasyEda, este entorno es capaz de convertir un diagrama de circuito electrónico en un circuito impreso. El proceso es:

1. Convertimos los componentes y rutas de alambrado en un diseño de circuito impreso con la herramienta de conversión de EasyEda.
2. Vamos colocando los componentes del circuito en la posición y orientación deseada para el circuito impreso.
3. Al terminar de colocar los componentes dentro del área se procede a crear las rutas (pistas) que conformarán el circuito impreso, todas se deben crear de forma manual.
4. Al terminar el proceso de creación de rutas se procede a la creación de orificios donde se conectarán cada uno de los componentes electrónicos. La separación y la posición de los puntos son calculados automáticamente por el entorno de desarrollo.

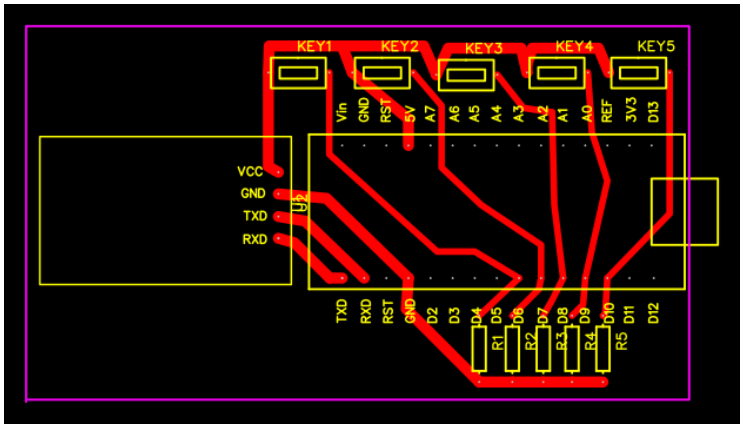


Figura 13. Circuito para imprimir.

Para la elaboración del circuito se pagó un servicio de fresado con un router CNC, al cual solamente se le proporcionó la imagen PCB del circuito sin las pistas.

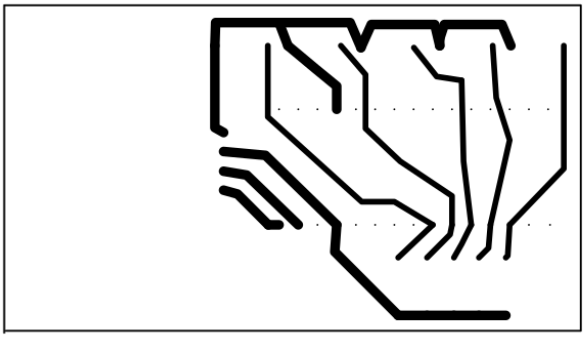


Figura 14. Imagen proporcionada para el fresado en CNC.

Finalmente, se soldaron todos los componentes de acuerdo a circuito diseñado previamente (figura 8) según el PCB mostrado en la figura 9.

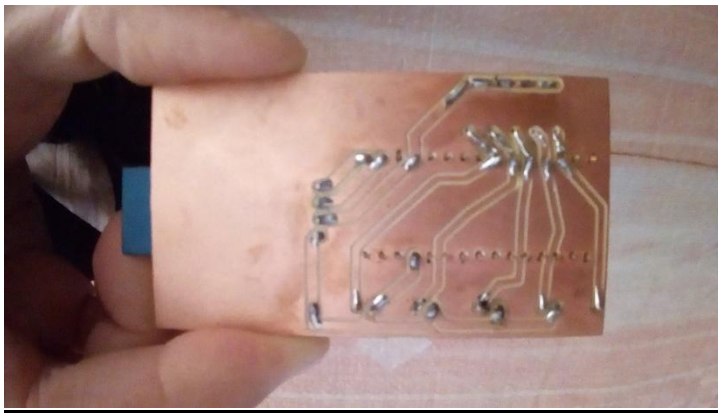


Figura 15. Circuito del servidor terminado.

7. Resultados

Al término de este proyecto, se obtuvo un sistema con el modelo cliente-servidor que permite reproducir un mensaje de voz con la simple presión de, al menos, dos botones. La manera de operar del sistema es la siguiente:

Asegurándose de que tanto cliente como servidor se encuentren encendidos, se activa el bluetooth en el smartphone que tiene la aplicación cliente “ProtoPattern”. Después, se empareja el servidor (en este caso se le asignó el nombre de prueba “HANSKI_01”) con el cliente.

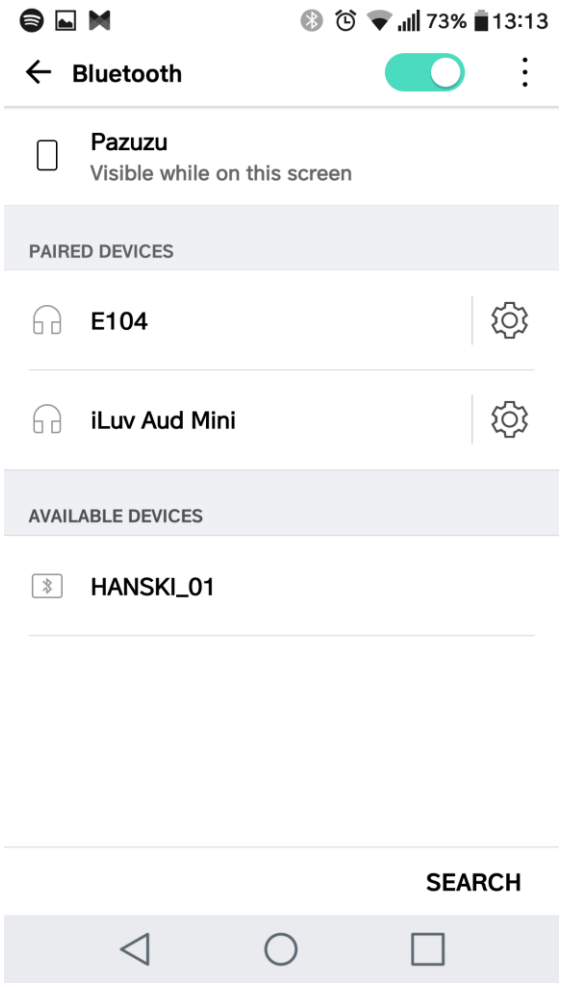


Figura 16. Emparejamiento del servidor con el cliente.

Después, se ejecuta la aplicación “ProtoPattern”, la cual nos pide conectarla al dispositivo servidor.

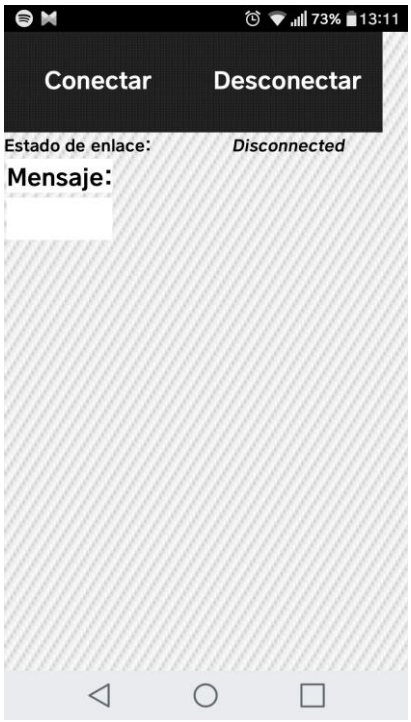


Figura 17. Interfaz de la aplicación "ProtoPattern".

Al presionar el botón “Conectar” nos pedirá que escojamos el dispositivo a conectar. en este caso, “HANSKI_01” es nuestro servidor.

Figura 18. Conexión del servidor con la aplicación.

Al estar conectado, podemos proceder a presionar una combinación de botones para generar un “texto hablado”. En la figura 19 se presiona pulgar + índice, lo que es equivalente a A1000.

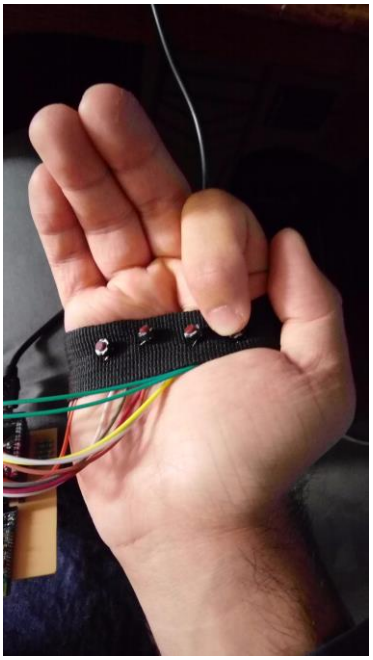


Figura 19. Ejemplo del envío del mensaje “Hasta Luego”.

Este código está asociado al mensaje “Hasta luego”, el cual se despliega en pantalla y es reproducido por las bocinas del smartphone.

Figura 20. Resultado del envío del mensaje "Hasta Luego".

Para ver la lista completa de patrones y mensajes, refiérase al apartado “Entregables” al final de este documento, en donde se incluyen las 5 tablas de cada categoría.
Para analizar los resultados podemos abordar las siguientes condiciones que determinan cuando el proyecto se puede dar por terminado.

El sistema general puede realizar todas las funcionalidades propuestas.

- Permite introducir los 75 mensajes acordados en la propuesta.
- Mediante un sistema de combinación de botones, permite manejar 5 categorías de mensajes como se había descrito en la propuesta.
- El sistema hace uso de un cliente receptor y procesador de los patrones recibidos y un servidor que recibe y procesa las señales digitales enviadas por los botones y envía un código vía bluetooth al cliente como se había propuesto.
- El sistema procesa los mensajes textuales y los convierte a voz mediante la API “Text-to-Speech” como se había planteado.

El sistema general no presenta alguna anomalía física como calentamiento de los circuitos electrónicos.

- Se probó el dispositivo con todos los patrones por dos horas continuas y no presentó ningún tipo de calentamiento o anomalía.

El sistema general no presenta errores de código ni inconsistencias en las categorías y mensajes.

- El dispositivo ejecuta todas las instrucciones de forma inmediata y la experiencia de uso es buena. No presenta retardos ni envía mensajes incorrectos.

8. Conclusiones

Un proyecto de este tipo, si bien no es un sustituto del lenguaje hablado o del lenguaje a señas, hace posible la comunicación entre personas que viven en realidades diferentes: la de padecer un trastorno del habla que dificulta la comunicación eficiente y la gente que no presenta estas limitantes.

El público objetivo de este proyecto son las personas con trastornos del habla, sin embargo, puede ser usado por cualquier persona con fines de comunicación. El tener una tabla de asociación patrón-mensaje hace que sea muy fácil utilizarlo, consultar el mensaje que se quiere enviar e incluso aprenderse los patrones que el usuario desee usar constantemente.

Para proyectos posteriores, se podría diseñar e implementar un sistema de personalización de mensajes y creación de perfiles para poder aumentar la gama de mensajes que se pueden enviar con este dispositivo. También se podría pensar en crear un dispositivo más compacto y unificado en un sólo gadget “wearable” con bocina integrada y que únicamente use la aplicación para su configuración.

9. Referencias

[1] A. Lovera Monroy, *Estudio y manejo de una pantalla Gráfica "Touchscreen" con Arduino y su aplicación a un teclado musical de una octava*, 1st ed. Ciudad de México, 2013.

[2] M. Hernández Alves, *Dispositivo de iluminación LED para habitaciones con incorporación de electrónica digital y control desde Android por bluetooth*, 1st ed. Ciudad de México, 2016.

[3] "Touch Voice | The Medical Speaking App", *Touch-voice.com*, 2017. [Online]. Available: https://touch-voice.com/marketing/. [Acceso: 24- feb- 2017].

[4] E. Bidelman, "Web apps that talk - Introduction to the Speech Synthesis API | Web | Google Developers", *Google Developers*, 2014. [Online]. Available: https://developers.google.com/web/updates/2014/01/Web-apps-that-talk-Introduction-to-the-Speech-Synthesis-API. [Acceso: 27- feb- 2017].

[5] Ohmura, R., Naya, F. and Noma, H. (2006). *2006 1st International Symposium on Wireless Pervasive Computing (ISWPC)*. 1st ed. Piscataway, NJ: IEEE Operations Center.

[6] Muaremi, A., Arnrich, B. and Tröster, G. (2013). *Towards Measuring Stress with Smartphones and Wearable Devices During Workday and Sleep*. 1st ed. BioNanoScience, p.172.

[7] "Los sordos son 'invisibles' en México", *Sipse.com*, 2017. [Online]. Available: http://sipse.com/mexico/sordos-discapacidad-gobierno-mexico-224324.html. [Acceso: 28- mar- 2017].

[8]"Arduino Nano", Store.arduino.cc, 2017. [Online]. Available: https://store.arduino.cc/usa/arduino-nano. [Accessed: 17- Aug- 2017].

[9]Brena, Ramón (2003). *Autómatas y Lenguajes. Un enfoque de diseño*. Tecnológico de Monterrey, México. [Accessed: 17- Aug- 2017].

10. Apéndices

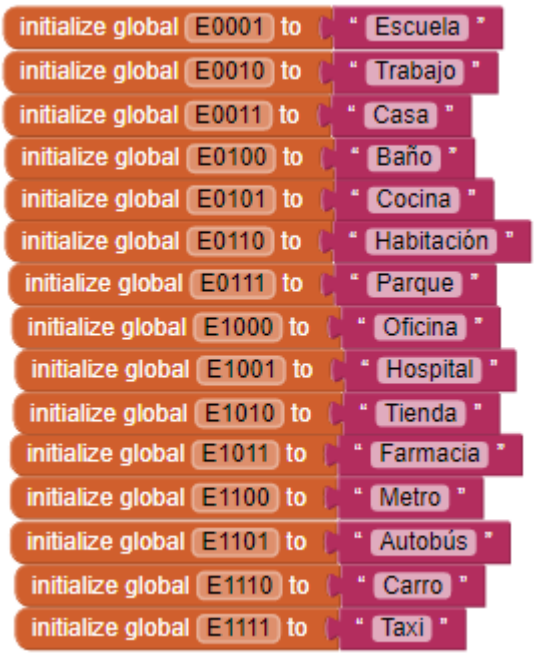
10.1. Código Arduino C para el servidor.

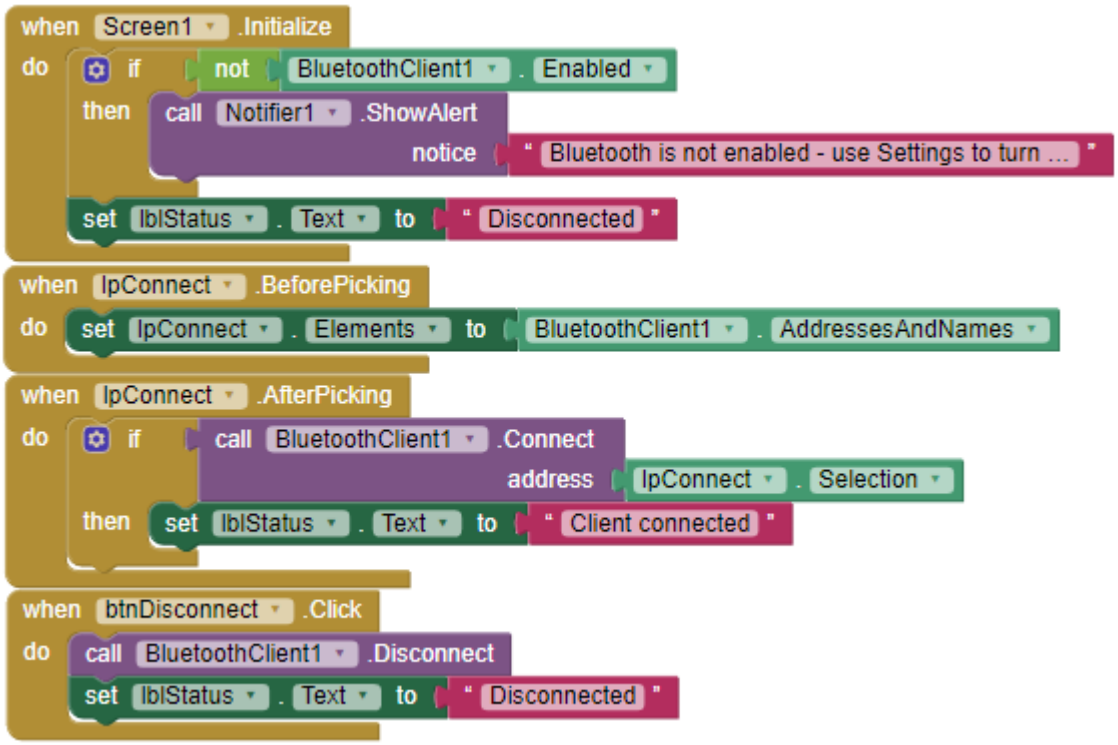
```
const int buttonPin1 = 6;
const int buttonPin2 = 7;
const int buttonPin3 = 8;
const int buttonPin4 = 9;
const int buttonPin5 = 10;

int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int buttonState4 = 0;
int buttonState5 = 0;

char pattern[5] = {0, 0, 0, 0, 0}; //contiene el patrón generado por los botones
String str = "A0000";
String str2 = "xxxxx";

/*
pattern[0]  /   dedo      /   buttonState

   A        /   PULGAR    /       1
   B        /   INDICE    /       2
   C        /   MEDIO     /       3
   D        /   ANULAR    /       4
   E        /   ME~NIQUE  /       5

*/

void setup() {

  Serial.begin(9600);

  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
  pinMode(buttonPin3, INPUT);
  pinMode(buttonPin4, INPUT);
  pinMode(buttonPin5, INPUT);

}

void loop() {
  buttonState1 = digitalRead(buttonPin1);
  buttonState2 = digitalRead(buttonPin2);
  buttonState3 = digitalRead(buttonPin3);
  buttonState4 = digitalRead(buttonPin4);
  buttonState5 = digitalRead(buttonPin5);

  if(Serial){
  s0();
```

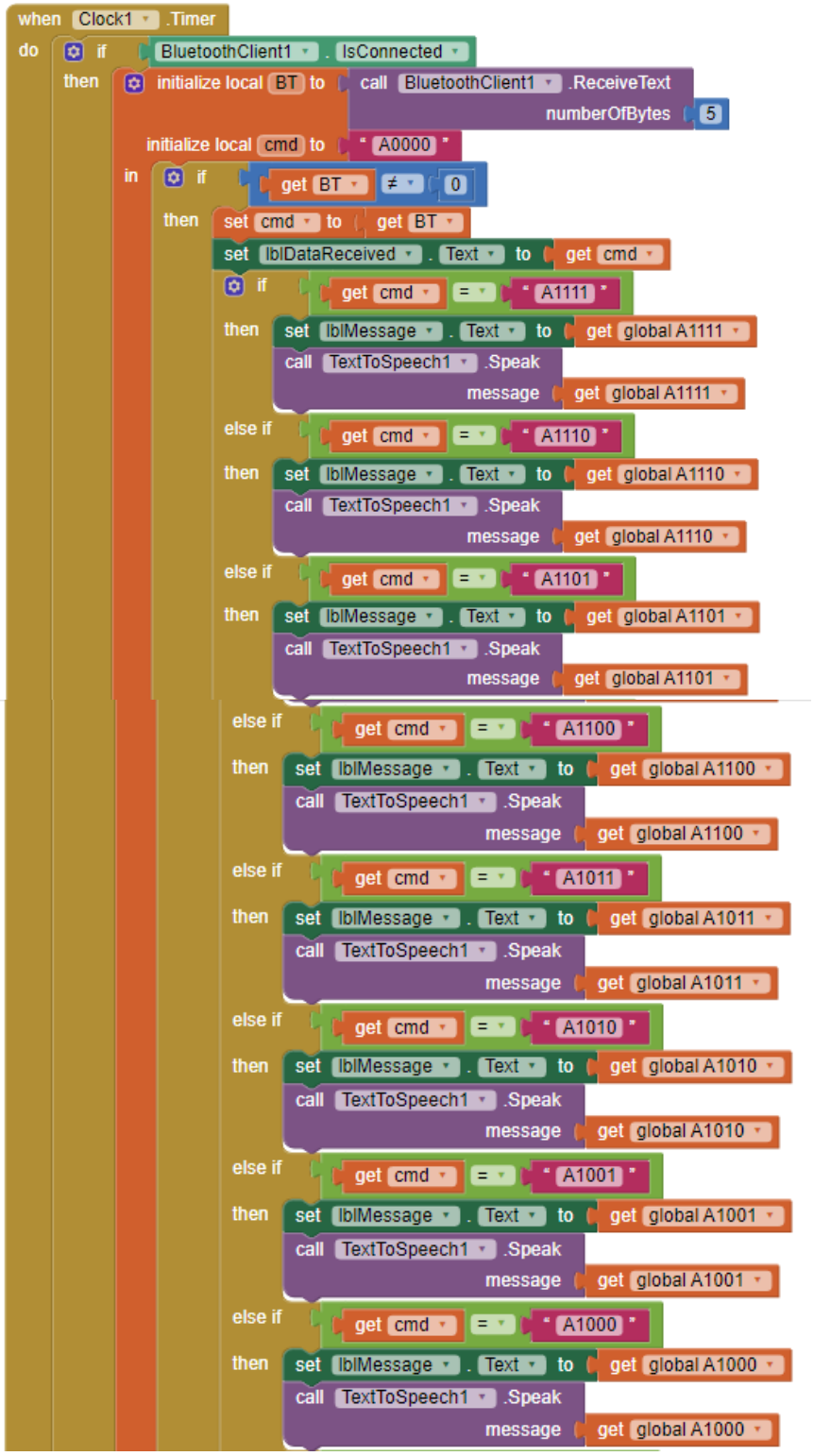
```
  }
}


//Funciones para la máquina de estados

void s0(){

  for(int i = 0; i <= 1000; i++){

    if (buttonState1 == HIGH && buttonState2 == LOW && buttonState3 == LOW &&
buttonState4 == LOW && buttonState5 == LOW) pattern[0] = 'A';
    if (buttonState1 == LOW && buttonState2 == HIGH && buttonState3 == LOW &&
buttonState4 == LOW && buttonState5 == LOW) pattern[0] = 'B';
    if (buttonState1 == LOW && buttonState2 == LOW && buttonState3 == HIGH &&
buttonState4 == LOW && buttonState5 == LOW) pattern[0] = 'C';
    if (buttonState1 == LOW && buttonState2 == LOW && buttonState3 == LOW &&
buttonState4 == HIGH && buttonState5 == LOW) pattern[0] = 'D';
    if (buttonState1 == LOW && buttonState2 == LOW && buttonState3 == LOW &&
buttonState4 == LOW && buttonState5 == HIGH) pattern[0] = 'E';

  }
  s1();
}

void s1(){

  for(int i = 0; i <= 1000; i++){
    if(pattern[0] == 'A'){
      if(buttonState2 == HIGH) pattern[1] = '1';
      else if(buttonState2 == LOW) pattern[1] = '0';
    }
    if(pattern[0] == 'B' || pattern[0] == 'C' || pattern[0] == 'D' || pattern[0] ==
'E'){
      if(buttonState1 == HIGH) pattern[1] = '1';
      else if(buttonState1 == LOW) pattern[1] = '0';
    }
  }
  s2();
}

void s2(){

  for(int i = 0; i <= 1000; i++){
    if(pattern[0] == 'A' || pattern[0] == 'B'){
      if(buttonState3 == HIGH) pattern[2] = '1';
      else if(buttonState3 == LOW) pattern[2] = '0';
    }
    if(pattern[0] == 'C' || pattern[0] == 'D' || pattern[0] == 'E'){
      if(buttonState2 == HIGH) pattern[2] = '1';
      else if(buttonState2 == LOW) pattern[2] = '0';

    }
  }
```

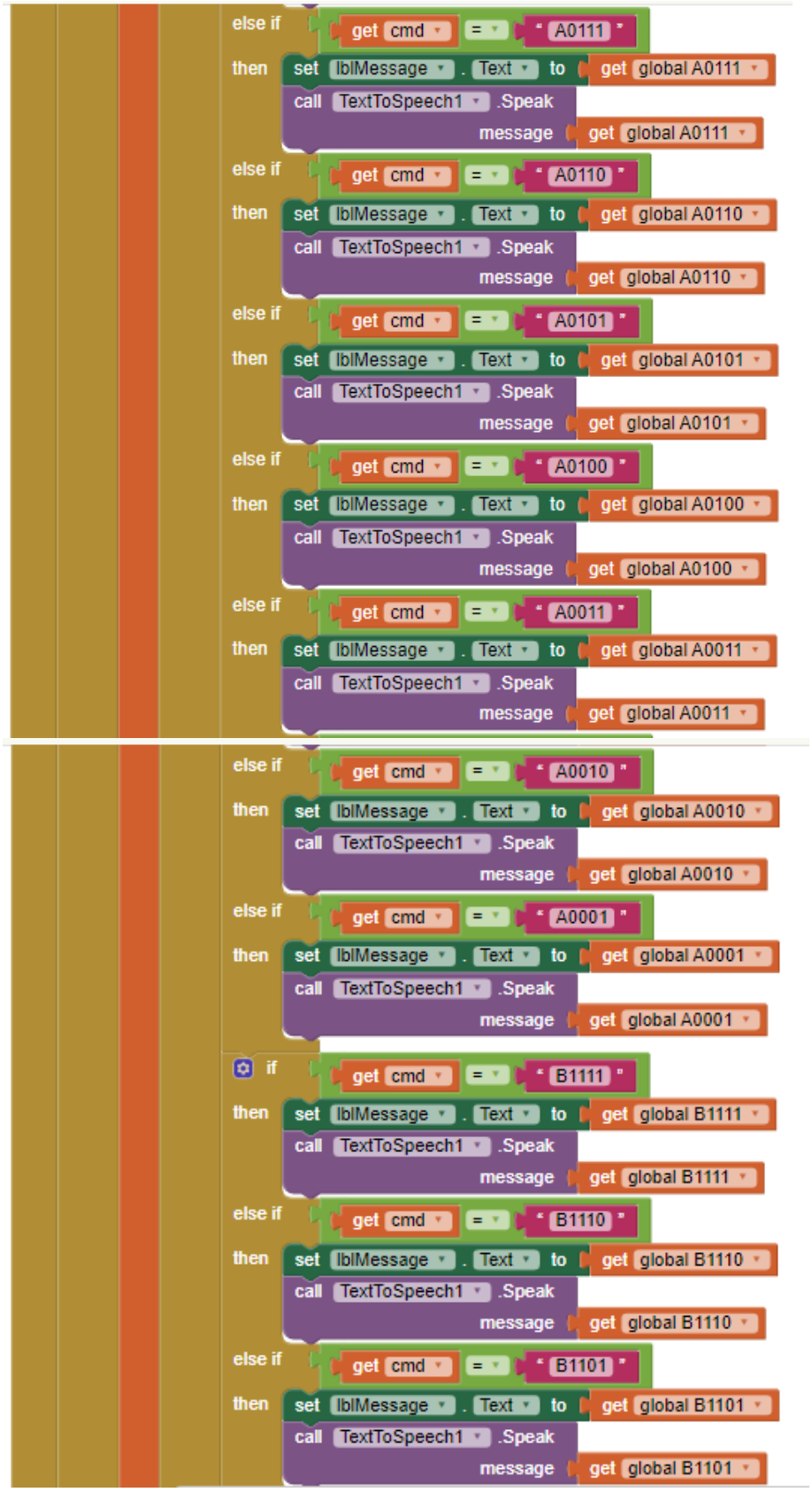
```
  s3();

}

void s3(){

  for(int i = 0; i <= 1000; i++){
    if(pattern[0] == 'A' || pattern[0] == 'B' || pattern[0] == 'C'){
      if(buttonState4 == HIGH) pattern[3] = '1';
      else if(buttonState4 == LOW) pattern[3] = '0';
    }
    if(pattern[0] == 'D' || pattern[0] == 'E'){
      if(buttonState3 == HIGH) pattern[3] = '1';
      else if(buttonState3 == LOW) pattern[3] = '0';
    }
  }
    s4();
}

void s4(){

  for(int i = 0; i <= 1000; i++){
    if(pattern[0] == 'A' || pattern[0] == 'B' || pattern[0] == 'C' || pattern[0] ==
'D'){
      if(buttonState5 == HIGH) pattern[4] = '1';
      else if(buttonState5 == LOW) pattern[4] = '0';
    }

    if( pattern[0] == 'E'){
      if(buttonState4 == HIGH) pattern[4] = '1';
      else if(buttonState4 == LOW) pattern[4] = '0';
    }
  }
  sF();
}

void sF(){
  str = pattern;
  str = str.substring(0, 5);
  if(str != "A0000" && str != "B0000" && str != "C0000" && str != "D0000" && str !=
"E0000" && str != ""){
    printMacro();
    //delay(500);
  }
 }


 void printMacro(){
    if (str != str2){
      Serial.print(str);
    }
    str2 = str;
 }
```

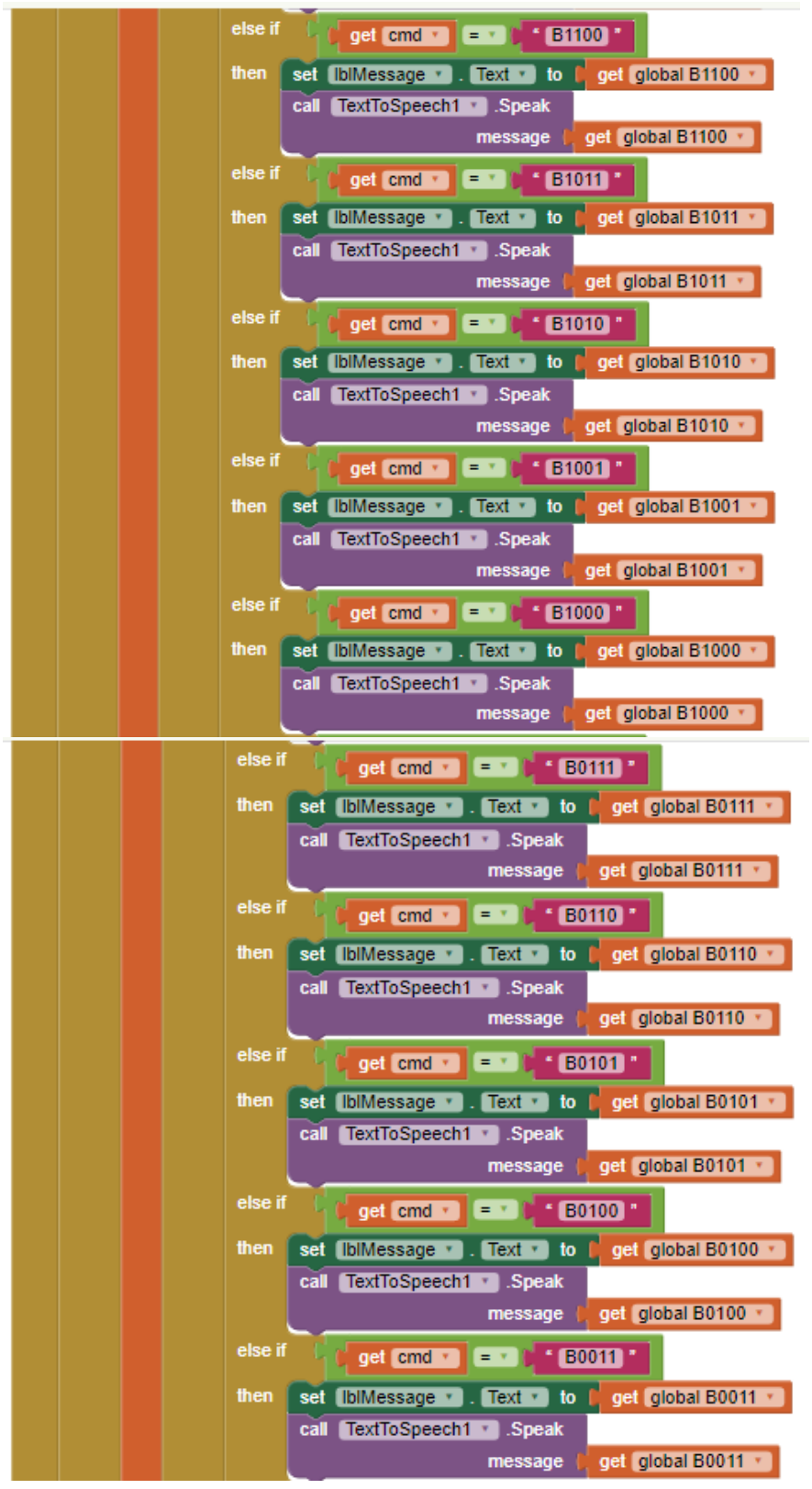
10.2. Código de bloques de App Inventor 2 para el cliente.

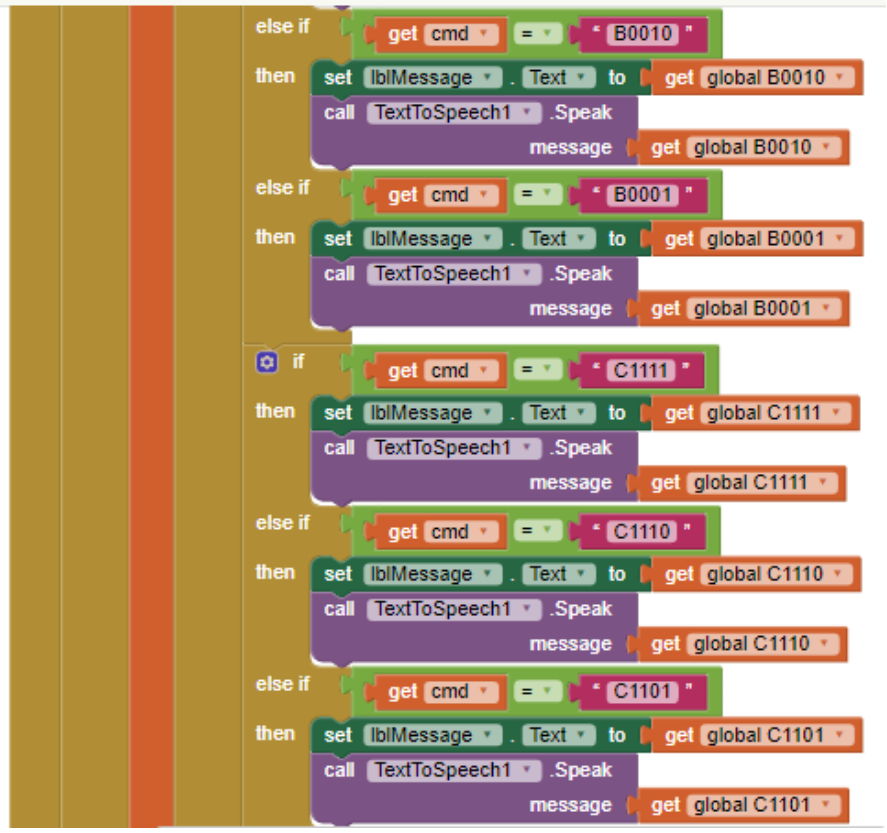

initialize global E0001 to " Escuela "
initialize global E0010 to " Trabajo "
initialize global E0011 to " Casa "
initialize global E0100 to " Baño "
initialize global E0101 to " Cocina "
initialize global E0110 to " Habitación "
initialize global E0111 to " Parque "
initialize global E1000 to " Oficina "
initialize global E1001 to " Hospital "
initialize global E1010 to " Tienda "
initialize global E1011 to " Farmacia "
initialize global E1100 to " Metro "
initialize global E1101 to " Autobús "
initialize global E1110 to " Carro "
initialize global E1111 to " Taxi "

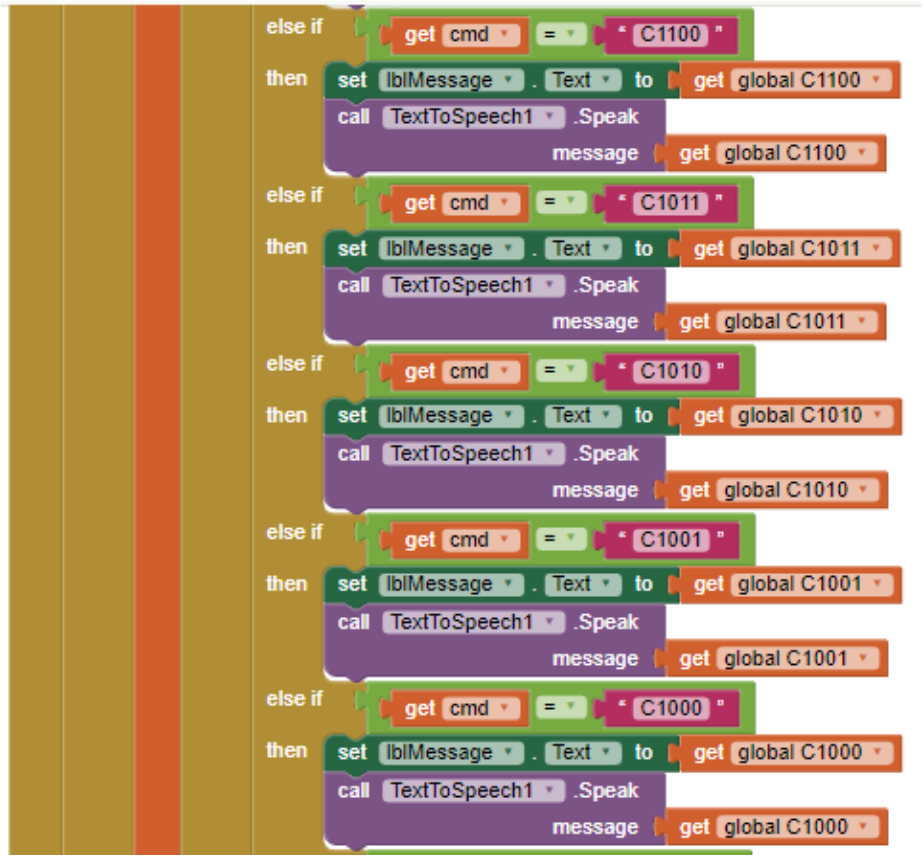

when Screen1 .Initialize
do if not BluetoothClient1 . Enabled
then call Notifier1 .ShowAlert
notice " Bluetooth is not enabled - use Settings to turn ... "
set lblStatus . Text to " Disconnected "
when lpConnect .BeforePicking
do set lpConnect . Elements to BluetoothClient1 . AddressesAndNames
when lpConnect .AfterPicking
do if call BluetoothClient1 .Connect
address lpConnect . Selection
then set lblStatus . Text to " Client connected "
when btnDisconnect .Click
do call BluetoothClient1 .Disconnect
set lblStatus . Text to " Disconnected "

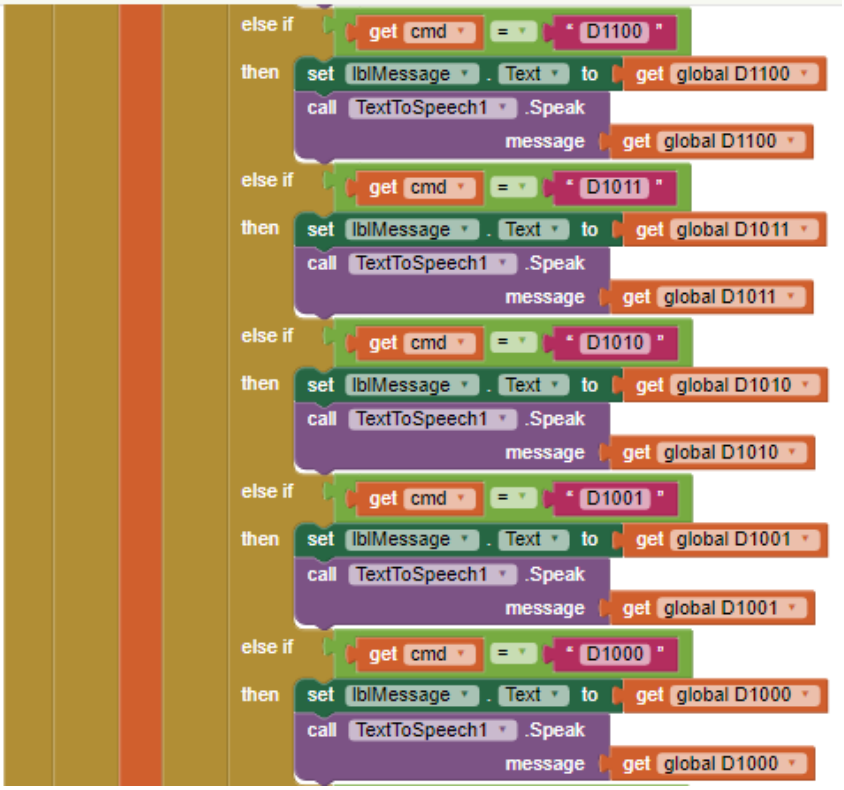

when Clock1 .Timer
do if BluetoothClient1 . IsConnected
then initialize local BT to call BluetoothClient1 .ReceiveText
numberOfBytes 5
initialize local cmd to " A0000 "
in if get BT ≠ 0
then set cmd to get BT
set lblDataReceived . Text to get cmd
if get cmd = " A1111 "
then set lblMessage . Text to get global A1111
call TextToSpeech1 .Speak
message get global A1111
else if get cmd = " A1110 "
then set lblMessage . Text to get global A1110
call TextToSpeech1 .Speak
message get global A1110
else if get cmd = " A1101 "
then set lblMessage . Text to get global A1101
call TextToSpeech1 .Speak
message get global A1101
else if get cmd = " A1100 "
then set lblMessage . Text to get global A1100
call TextToSpeech1 .Speak
message get global A1100
else if get cmd = " A1011 "
then set lblMessage . Text to get global A1011
call TextToSpeech1 .Speak
message get global A1011
else if get cmd = " A1010 "
then set lblMessage . Text to get global A1010
call TextToSpeech1 .Speak
message get global A1010
else if get cmd = " A1001 "
then set lblMessage . Text to get global A1001
call TextToSpeech1 .Speak
message get global A1001
else if get cmd = " A1000 "
then set lblMessage . Text to get global A1000
call TextToSpeech1 .Speak
message get global A1000

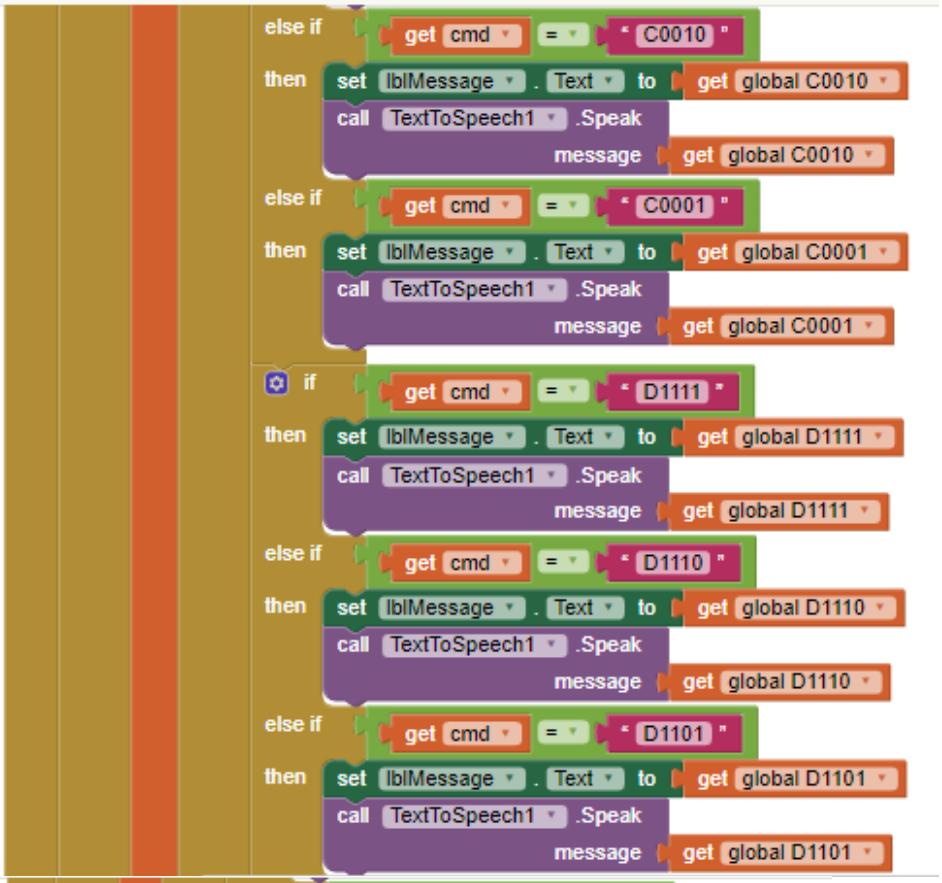

else if get cmd = " A0111 "
then set lblMessage . Text to get global A0111
call TextToSpeech1 .Speak
message get global A0111
else if get cmd = " A0110 "
then set lblMessage . Text to get global A0110
call TextToSpeech1 .Speak
message get global A0110
else if get cmd = " A0101 "
then set lblMessage . Text to get global A0101
call TextToSpeech1 .Speak
message get global A0101
else if get cmd = " A0100 "
then set lblMessage . Text to get global A0100
call TextToSpeech1 .Speak
message get global A0100
else if get cmd = " A0011 "
then set lblMessage . Text to get global A0011
call TextToSpeech1 .Speak
message get global A0011
else if get cmd = " A0010 "
then set lblMessage . Text to get global A0010
call TextToSpeech1 .Speak
message get global A0010
else if get cmd = " A0001 "
then set lblMessage . Text to get global A0001
call TextToSpeech1 .Speak
message get global A0001
if get cmd = " B1111 "
then set lblMessage . Text to get global B1111
call TextToSpeech1 .Speak
message get global B1111
else if get cmd = " B1110 "
then set lblMessage . Text to get global B1110
call TextToSpeech1 .Speak
message get global B1110
else if get cmd = " B1101 "
then set lblMessage . Text to get global B1101
call TextToSpeech1 .Speak
message get global B1101

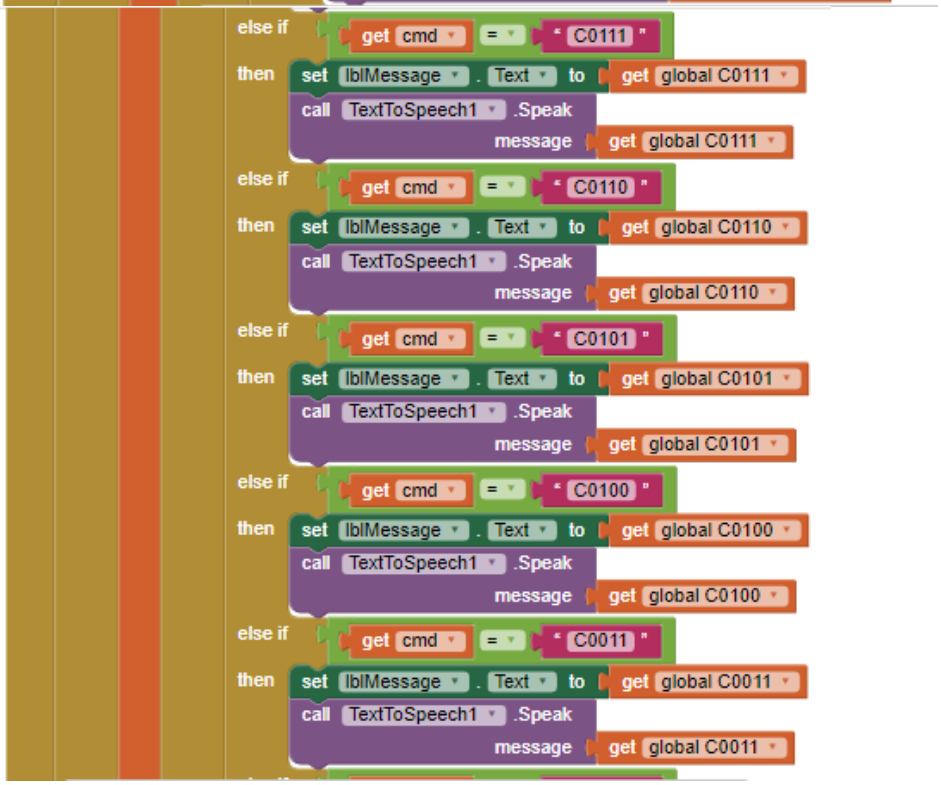

else if get cmd = " B1100 "
then set lblMessage . Text to get global B1100
call TextToSpeech1 .Speak
message get global B1100
else if get cmd = " B1011 "
then set lblMessage . Text to get global B1011
call TextToSpeech1 .Speak
message get global B1011
else if get cmd = " B1010 "
then set lblMessage . Text to get global B1010
call TextToSpeech1 .Speak
message get global B1010
else if get cmd = " B1001 "
then set lblMessage . Text to get global B1001
call TextToSpeech1 .Speak
message get global B1001
else if get cmd = " B1000 "
then set lblMessage . Text to get global B1000
call TextToSpeech1 .Speak
message get global B1000
else if get cmd = " B0111 "
then set lblMessage . Text to get global B0111
call TextToSpeech1 .Speak
message get global B0111
else if get cmd = " B0110 "
then set lblMessage . Text to get global B0110
call TextToSpeech1 .Speak
message get global B0110
else if get cmd = " B0101 "
then set lblMessage . Text to get global B0101
call TextToSpeech1 .Speak
message get global B0101
else if get cmd = " B0100 "
then set lblMessage . Text to get global B0100
call TextToSpeech1 .Speak
message get global B0100
else if get cmd = " B0011 "
then set lblMessage . Text to get global B0011
call TextToSpeech1 .Speak
message get global B0011

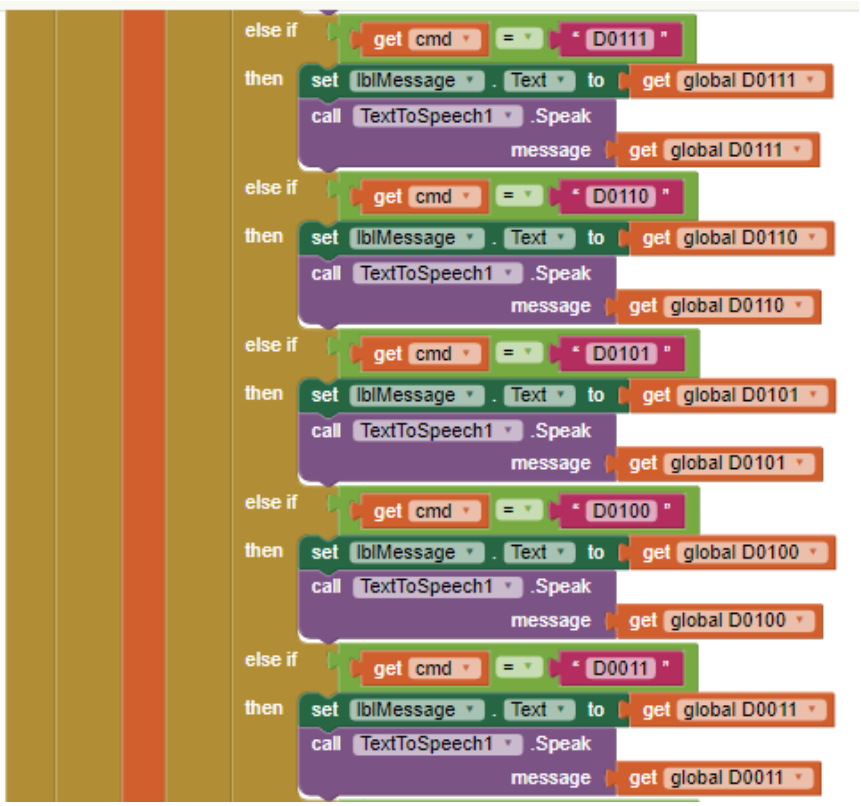

else if
get cmd = " B0010 "
then
set lblMessage . Text to get global B0010
call TextToSpeech1 .Speak
message get global B0010
else if
get cmd = " B0001 "
then
set lblMessage . Text to get global B0001
call TextToSpeech1 .Speak
message get global B0001
if
get cmd = " C1111 "
then
set lblMessage . Text to get global C1111
call TextToSpeech1 .Speak
message get global C1111
else if
get cmd = " C1110 "
then
set lblMessage . Text to get global C1110
call TextToSpeech1 .Speak
message get global C1110
else if
get cmd = " C1101 "
then
set lblMessage . Text to get global C1101
call TextToSpeech1 .Speak
message get global C1101

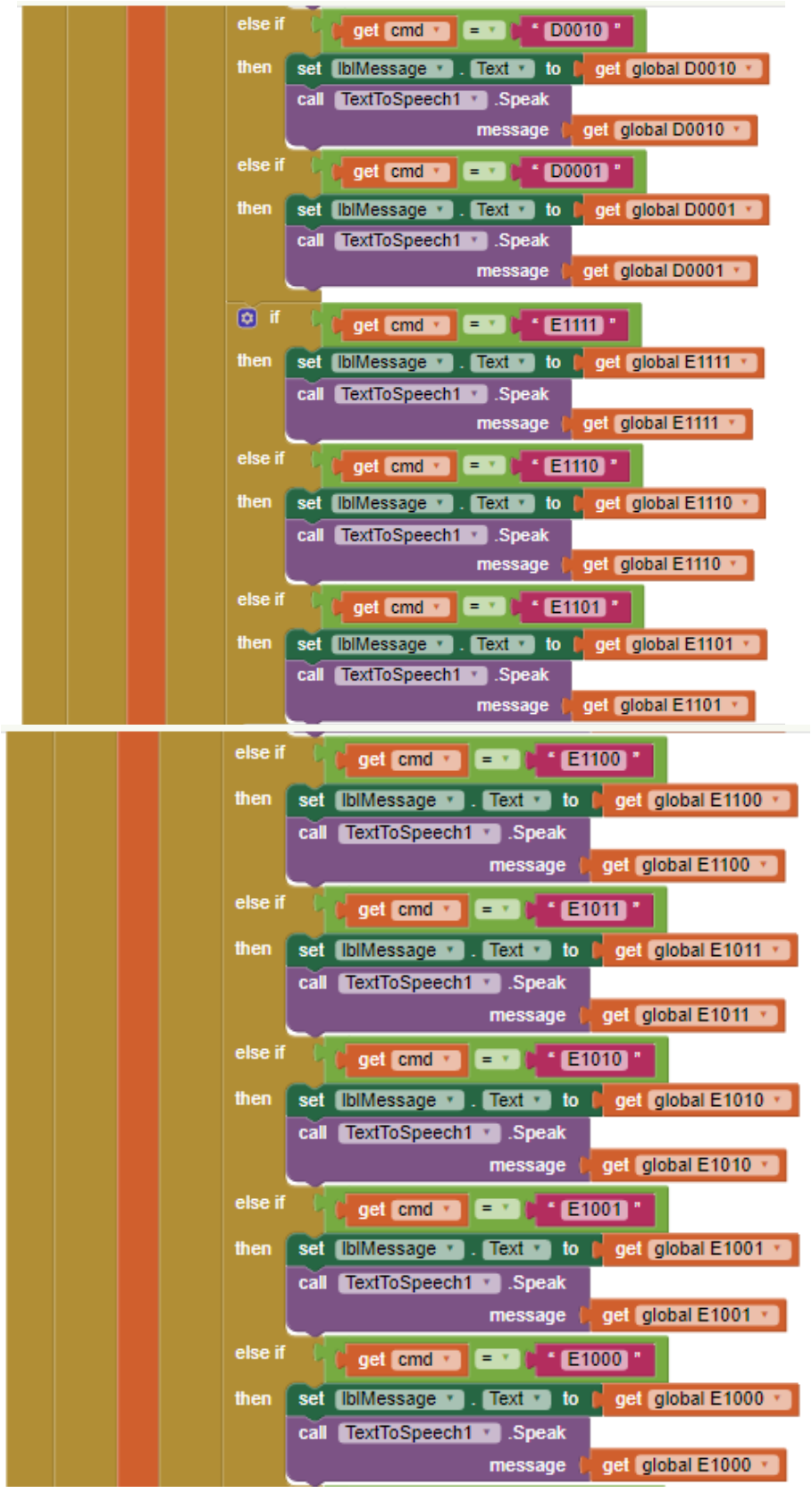

else if get cmd = " C1100 "
then set lblMessage . Text to get global C1100
call TextToSpeech1 .Speak
message get global C1100
else if get cmd = " C1011 "
then set lblMessage . Text to get global C1011
call TextToSpeech1 .Speak
message get global C1011
else if get cmd = " C1010 "
then set lblMessage . Text to get global C1010
call TextToSpeech1 .Speak
message get global C1010
else if get cmd = " C1001 "
then set lblMessage . Text to get global C1001
call TextToSpeech1 .Speak
message get global C1001
else if get cmd = " C1000 "
then set lblMessage . Text to get global C1000
call TextToSpeech1 .Speak
message get global C1000

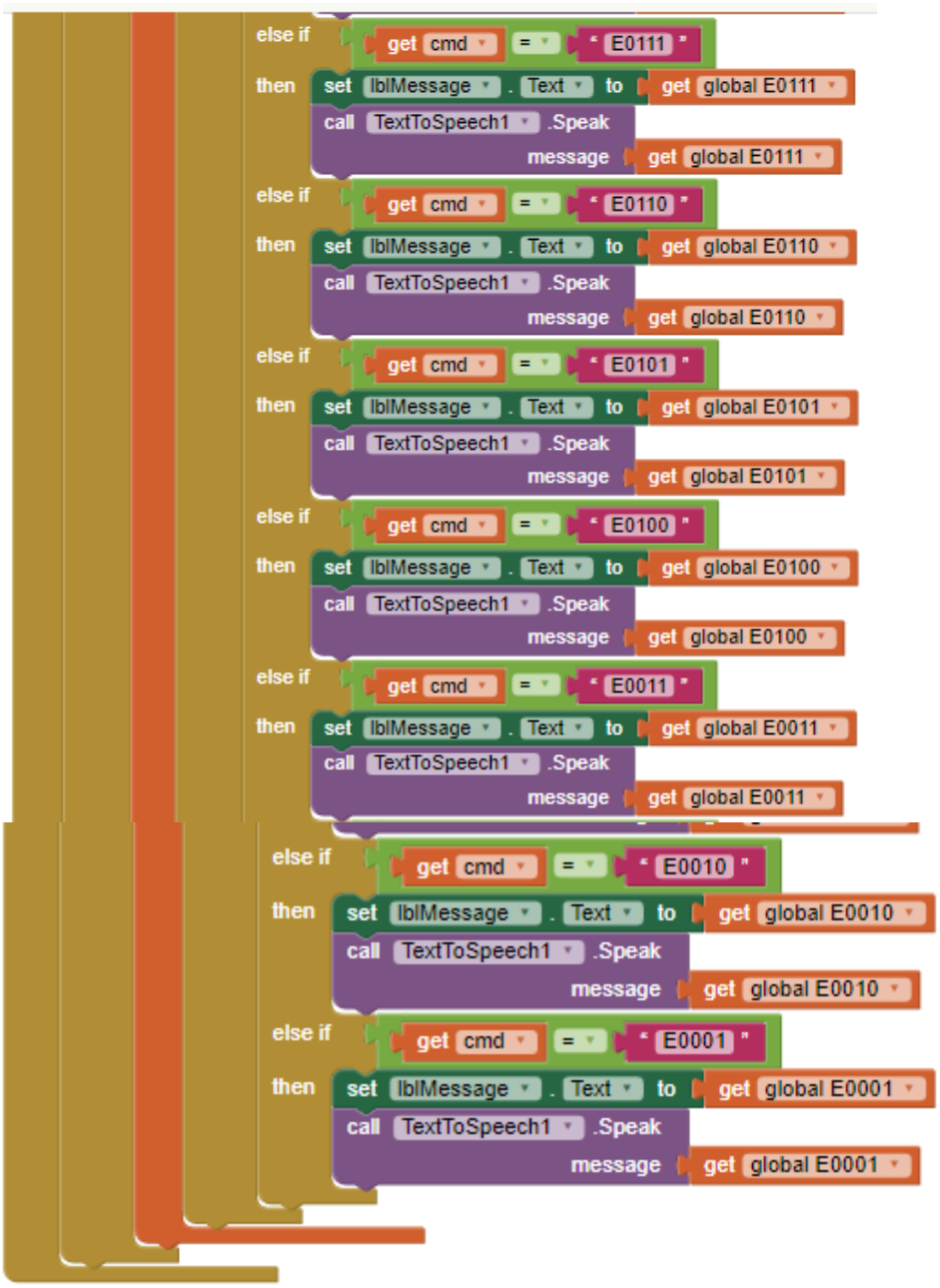

else if get cmd = " D1100 "
then set lblMessage . Text to get global D1100
call TextToSpeech1 .Speak
message get global D1100
else if get cmd = " D1011 "
then set lblMessage . Text to get global D1011
call TextToSpeech1 .Speak
message get global D1011
else if get cmd = " D1010 "
then set lblMessage . Text to get global D1010
call TextToSpeech1 .Speak
message get global D1010
else if get cmd = " D1001 "
then set lblMessage . Text to get global D1001
call TextToSpeech1 .Speak
message get global D1001
else if get cmd = " D1000 "
then set lblMessage . Text to get global D1000
call TextToSpeech1 .Speak
message get global D1000


else if get cmd = " C0010 "
then set lblMessage . Text to get global C0010
call TextToSpeech1 .Speak
message get global C0010
else if get cmd = " C0001 "
then set lblMessage . Text to get global C0001
call TextToSpeech1 .Speak
message get global C0001
if get cmd = " D1111 "
then set lblMessage . Text to get global D1111
call TextToSpeech1 .Speak
message get global D1111
else if get cmd = " D1110 "
then set lblMessage . Text to get global D1110
call TextToSpeech1 .Speak
message get global D1110
else if get cmd = " D1101 "
then set lblMessage . Text to get global D1101
call TextToSpeech1 .Speak
message get global D1101


else if get cmd = " C0111 "
then set lblMessage . Text to get global C0111
call TextToSpeech1 .Speak
message get global C0111
else if get cmd = " C0110 "
then set lblMessage . Text to get global C0110
call TextToSpeech1 .Speak
message get global C0110
else if get cmd = " C0101 "
then set lblMessage . Text to get global C0101
call TextToSpeech1 .Speak
message get global C0101
else if get cmd = " C0100 "
then set lblMessage . Text to get global C0100
call TextToSpeech1 .Speak
message get global C0100
else if get cmd = " C0011 "
then set lblMessage . Text to get global C0011
call TextToSpeech1 .Speak
message get global C0011


else if
get cmd = " D0111 "
then
set lblMessage . Text to get global D0111
call TextToSpeech1 .Speak
message get global D0111
else if
get cmd = " D0110 "
then
set lblMessage . Text to get global D0110
call TextToSpeech1 .Speak
message get global D0110
else if
get cmd = " D0101 "
then
set lblMessage . Text to get global D0101
call TextToSpeech1 .Speak
message get global D0101
else if
get cmd = " D0100 "
then
set lblMessage . Text to get global D0100
call TextToSpeech1 .Speak
message get global D0100
else if
get cmd = " D0011 "
then
set lblMessage . Text to get global D0011
call TextToSpeech1 .Speak
message get global D0011


else if get cmd = " D0010 "
then set lblMessage . Text to get global D0010
call TextToSpeech1 .Speak
message get global D0010
else if get cmd = " D0001 "
then set lblMessage . Text to get global D0001
call TextToSpeech1 .Speak
message get global D0001
if get cmd = " E1111 "
then set lblMessage . Text to get global E1111
call TextToSpeech1 .Speak
message get global E1111
else if get cmd = " E1110 "
then set lblMessage . Text to get global E1110
call TextToSpeech1 .Speak
message get global E1110
else if get cmd = " E1101 "
then set lblMessage . Text to get global E1101
call TextToSpeech1 .Speak
message get global E1101
else if get cmd = " E1100 "
then set lblMessage . Text to get global E1100
call TextToSpeech1 .Speak
message get global E1100
else if get cmd = " E1011 "
then set lblMessage . Text to get global E1011
call TextToSpeech1 .Speak
message get global E1011
else if get cmd = " E1010 "
then set lblMessage . Text to get global E1010
call TextToSpeech1 .Speak
message get global E1010
else if get cmd = " E1001 "
then set lblMessage . Text to get global E1001
call TextToSpeech1 .Speak
message get global E1001
else if get cmd = " E1000 "
then set lblMessage . Text to get global E1000
call TextToSpeech1 .Speak
message get global E1000


else if get cmd = " E0111 "
then set lblMessage . Text to get global E0111
call TextToSpeech1 .Speak
message get global E0111
else if get cmd = " E0110 "
then set lblMessage . Text to get global E0110
call TextToSpeech1 .Speak
message get global E0110
else if get cmd = " E0101 "
then set lblMessage . Text to get global E0101
call TextToSpeech1 .Speak
message get global E0101
else if get cmd = " E0100 "
then set lblMessage . Text to get global E0100
call TextToSpeech1 .Speak
message get global E0100
else if get cmd = " E0011 "
then set lblMessage . Text to get global E0011
call TextToSpeech1 .Speak
message get global E0011
else if get cmd = " E0010 "
then set lblMessage . Text to get global E0010
call TextToSpeech1 .Speak
message get global E0010
else if get cmd = " E0001 "
then set lblMessage . Text to get global E0001
call TextToSpeech1 .Speak
message get global E0001

11. Entregables

11.1. Tablas de patrones