UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

FACULTAD DE ESTUDIOS SUPERIORES

ARAGÓN

INGENIERÍA ELÉCTRICA ELECTRÓNICA

Desarrollo de un sistema computacional para la comunicación verbal, que brinda apoyo a personas mudas con escasa movilidad mediante una interfaz visual y síntesis de voz.

PARA OBTENER EL GRADO

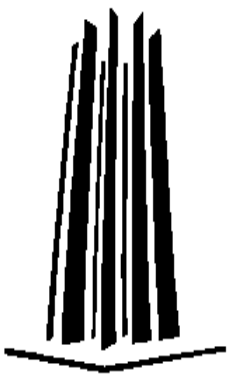
INGENIERO ELÉCTRICO ELECTRÓNICO

PRESENTA

GIOVANNI BELLI GÓMEZ

ASESOR DE TESIS

DR. ISMAEL DÍAZ RANGEL



Estado de México, noviembre 2017

Índice.

Capítulo 1. Introducción.

Aunque no es una enfermedad tan común, tampoco es rara, la Esclerosis Lateral Amiotrófica (ELA) en México afecta a cerca de 20 mil pacientes mientras que en el mundo se alcanzan los 2.3 millones de personas afectadas por esta enfermedad (Puig, 2017).

A pesar de que los afectados por esta enfermedad representan una minoría con respecto a la población mundial, los casos de esta enfermedad siguen apareciendo, no hay un diagnóstico de lo que causa la enfermedad, por lo que no hay forma de prevenirla, entre los factores influyen tenemos a la genética, la posición geográfica o un componente viral. (Rivera, 2017)

La ELA es una enfermedad difícil de diagnosticar, al punto de que los primeros síntomas pueden confundirse con depresión, incluso con pereza, la ELA resulta del daño a la mielina, la cual es una sustancia grasa que recubre las fibras nerviosas del cerebro, la mielina tiene una función parecida a la del aislante de un cable, cuando la mielina se daña, los impulsos nerviosos que van al cerebro se interrumpen o distorsionan, afectando la comunicación entre el cerebro y el cuerpo.

La causa de la enfermedad es desconocida hasta hoy, la ELA no es contagiosa, ni hereditaria y si bien no tiene cura por ahora, si hay medicación para controlarla y atenuar o espaciar los brotes o remisiones. (Milenio, 2017)

Un paciente con ELA va perdiendo gradualmente la capacidad de mover su cuerpo, pero el sistema nervioso no es afectado, por lo que sienten todo el tiempo a pesar de su incapacidad de expresarse.

Este trabajo consiste en desarrollar un sistema de comunicación para personas con discapacidad de comunicación verbal, escrita y gestual, que les permitirá expresarse sin necesidad de contar con un intérprete en todo momento; el mayor número de enfermos en estas condiciones son los afectados por ELA; sin embargo, no se limita a personas con dicha enfermedad.

El sistema de comunicación incorporará una microcomputadora Raspberry Pi con pantalla que mostrará el alfabeto, y mediante un sensor, el usuario podrá seleccionar

letras para formar frases, que serán reproducidas con un sintetizador de voz. Todo implementado a bajo costo.

1.1. Objetivo general.

Desarrollar un sistema de comunicación que incorpore una microcomputadora con pantalla, la cual mostrará el alfabeto, y mediante un sensor el usuario fácilmente formará expresiones, que se reproducirán con un sintetizador de voz.

1.2. Objetivos particulares.

- Investigar las características de la enfermedad ELA (Esclerosis Lateral Amiotrófica).
- Identificar las condiciones en las que se encuentra una persona con ELA.
- Detallar las técnicas actuales con las que un paciente con ELA se comunica.
- Definir un sistema de cómputo adecuado para desarrollo de la propuesta.
- Seleccionar un sistema operativo sobre el cual se ejecutará el sistema de comunicación.
- Desarrollar una interfaz gráfica de usuario.
- Determinar un método de entrada asequible a personas con ELA.
- Establecer un método de síntesis de voz.
- Desarrollar algoritmos computacionales que integren entrada de texto y síntesis de voz.

1.3. Motivación.

La ELA es una enfermedad que afecta hasta 20 personas por cada 100 mil habitantes en México, según la Federación de Esclerosis Múltiple en México (Femmex, 2017), y que debido a su condición física y económica no pueden contar siempre con un una persona que cuide de ellos en todo momento, ya que algunos métodos para establecer la comunicación con estos pacientes depende de un intérprete, y los dispositivos electrónicos que pueden auxiliar y brindar algo de autonomía a los enfermos tiene un costo elevado; es debido a esto, que se decidió desarrollar un sistema a bajo costo, que permita a pacientes con esta enfermedad, o a personas que se encuentren en condiciones físicas similares, comunicarse a través de un sistema de cómputo.

1.4. Descripción del capitulado.

Capítulo 2. Se trata el marco teórico que sustenta el desarrollo de esta propuesta, el cual menciona: ELA (Esclerosis Lateral Amiotrófica), técnicas de comunicación de pacientes con ELA, sistemas de cómputo, sistemas operativos para Raspberry Pi, lenguajes de programación para Raspberry Pi, desarrollo de interfaces gráficas, métodos de entrada para el sistema de cómputo y sintetizadores de voz.

Capítulo 3. Se menciona el desarrollo experimental del sistema de comunicación propuesto a lo largo de este trabajo, los obstáculos encontrados a lo largo de la implementación y sus soluciones, así como de los procesos de programación utilizados.

Capítulo 4. Se presentan las pruebas realizadas al sistema propuesto y resultados obtenidos.

Finalmente se hará mención de las conclusiones, así como del trabajo futuro que se propone desarrollar para complementar esta propuesta.

Capítulo 2 Marco Teórico.

2.1. ELA (Esclerosis Lateral Amiotrófica).

“La esclerosis lateral amiotrófica (ELA) es una enfermedad neurológica progresiva, que ataca a las células nerviosas (neuronas) encargadas de controlar los músculos voluntarios. Las neuronas motoras están localizadas en el tallo del cerebro y la médula espinal, y sirven como unidades de control y enlaces de comunicación vital entre el sistema nervioso y los músculos voluntarios del cuerpo.

Los primeros datos sobre la enfermedad son de 1874, cuando un neurólogo francés, Jean-Martin Charcot, publicó el primer artículo completo llamado “Lecciones sobre enfermedades del sistema nervioso”. Es conocida también como enfermedad de Charcot, de Lou Gehrig o enfermedad de la neurona motora.

La ELA afecta a personas adultas, de cualquier raza y etnia, y el riesgo de desarrollarla aumenta al envejecer. Generalmente se da en personas de 40-60 años de edad. Los hombres la sufren con mayor frecuencia que las mujeres. La prevalencia de ELA es de 3-6 personas por cada 100.000 habitantes. En un 90-95% de todos los casos la enfermedad ocurre aparentemente de forma aleatoria, sin ningún factor de riesgo claramente asociado. Se desconoce por qué las neuronas motoras mueren en los pacientes con ELA, aunque se han considerado factores genéticos, ambientales y factores relacionados con la edad. Los pacientes no tienen una historia familiar de la enfermedad (sólo 5-10% de todos los casos son heredados). La forma familiar de ELA generalmente resulta de un patrón hereditario, que requiere que uno de los padres lleve el gen responsable. En la ELA tanto las neuronas motoras superiores como las inferiores se degeneran o mueren y dejan de enviar mensajes a los músculos, que quedan imposibilitados para funcionar; se van debilitando gradualmente, se gastan (atrofia) y se contraen (fasciculaciones). Finalmente desaparece la capacidad cerebral para controlar el movimiento voluntario y los pacientes pierden su fuerza y la capacidad de mover sus extremidades y al cabo todo su cuerpo. Cuando fallan los músculos del diafragma y de la pared torácica, se pierde la capacidad de respirar sin un ventilador o respirador artificial.

El inicio puede ser sutil y los síntomas pasan desapercibidos; pero la progresión más habitual es la siguiente: dificultad para caminar que conlleva el uso de ayudas técnicas hasta quedar en silla de ruedas, debilidad en los brazos y las manos, pérdida de la habilidad para escribir, teclear y alimentarse, al tiempo que se comienzan a presentar las dificultades para hablar y deglutir. Típicamente el paciente con ELA tiene un decremento gradual en la capacidad vital pulmonar por el debilitamiento de los músculos del pecho y el diafragma. Existen síntomas tempranos del empeoramiento de la capacidad respiratoria, como sentir falta de aire cuando están acostados o cuando realizan tareas sencillas como vestirse o comer" (Lerones & Rodriguéz, 2010).

2.2. Técnicas que ayudan a la comunicación de personas con ELA.

La ELA no afecta a los cinco sentidos y normalmente tampoco a la mente, los músculos de los ojos, el corazón, vejiga, intestino, o músculos sexuales, por tal motivo un paciente puede ver durante todo el proceso de su enfermedad, al igual que pueden sentir, oler, tener frio, calor, sed, ganas de ir al baño y presentan una gran dificultad al poder expresar sus necesidades, por tal motivo se han creado algunas ayudas técnicas que ayuden a la comunicación de la persona afectada, algunas de estas técnicas se tratarán a continuación al igual que la propuesta para este trabajo.

2.2.1 Tableros.

Una de las técnicas que se desarrollaron para la comunicación de personas con esta enfermedad que su condición ya no permite el habla, es la implementación de tableros con gráficos, como letras o imágenes, que permitan al paciente hacer referencia a lo que desea comunicar, recordando que la ELA es una enfermedad progresiva, el paciente puede encontrarse en distintas etapas de esta enfermedad, por lo tanto, podrían aun tener control motor del cuello o incluso de alguna extremidad, con esto la posibilidad de desplazar algún apuntador, como una linterna o un láser, y seleccionar con más precisión el grafico que deseen.

Dado que la enfermedad no afecta la vista, aunque el paciente se encuentre sin movilidad motora los ojos ayudan a la comunicación, para esto se han desarrollado

algunos tableros que con la ayuda del movimiento de los ojos permiten la interpretación de lo que el enfermo desea expresar, a continuación, hablaremos de dos de los tableros antes mencionados.

Cuadro ETRAN

Este es un tablero de acrílico que facilita a comunicación cara a cara mediante la mirada, el uso de este tablero consiste en distribuir pictogramas, imágenes o el alfabeto en bloques en las esquinas del tablero, a modo de que el usuario mire hacia el pictograma que desea seleccionar y el intérprete observe la dirección de los ojos con el propósito de que se establezca la comunicación, permitiéndole al usuario expresarse.

Ilustración 2.1 Cuadro ETRAN. [1]

En la Ilustración 2.1 se puede apreciar la distribución de un cuadro ETRAN, esta distribución permite que el usuario pueda elegir con una primera mirada el bloque donde se encuentra la letra que desea utilizar para formar sus palabras, el intérprete menciona en voz alta el color del bloque, se deben establecer previamente gestos de referencia para error y correcto, comúnmente un giño para correcto y cerrar ambos ojos para error, si el bloque que el intérprete menciono es el correcto el usuario procede a realizar una segunda mirada, esta vez dirigiendo la vista hacia el color de la letra que se encuentra en el bloque previamente seleccionado, y el intérprete debe mencionar en voz alta la letra. Un ejemplo utilizando la distribución del tablero de la Ilustración 2.1, para elegir la letra “G”, el usuario en su primera mirada debe dirigir la vista al bloque azul, que es donde se encuentra la letra y en

[1] Recuperada de https://sites.google.com/site/vicortega2/etran [Diciembre, 2016]

la segunda mirada deberá dirigir la vista al bloque rojo, ya que la letra “G” es de color rojo dentro de su correspondiente bloque. (Garcia, 2016)

Tablero Corradine

Este es otro tablero que utiliza una técnica muy similar a la del cuadro ETRAN, con la diferencia de que en este tablero no se utilizan colores, de igual manera el usuario selecciona la letra deseada con la mirada y un intérprete debe ir diciéndolas en voz alta hasta formar la palabra completa.

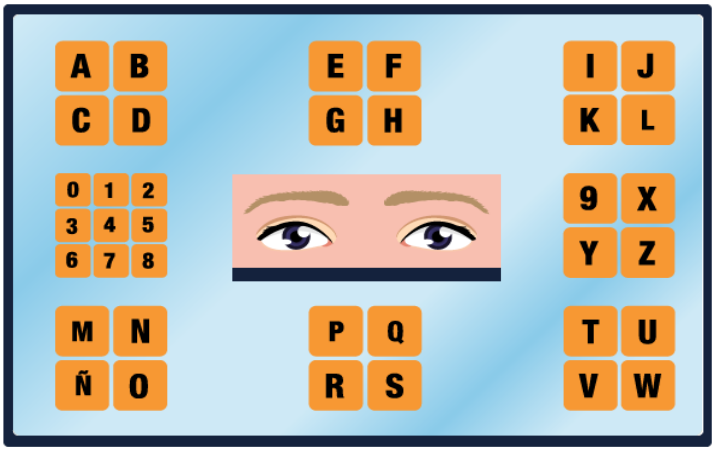


Ilustración 2.2 Tablero Corradine. [2]

Como se puede ver en la Ilustración 2.2 la distribución es de igual manera que en el cuadro ETRAN a bloques pero en esta distribución las letras se agrupan en conjuntos de cuatro, con la primer mirada el usuario selecciona el bloque en el que se encuentra la letra y también se deben establecer gestos para saber si el intérprete está entendiendo correctamente o no, para la segunda mirada el usuario únicamente debe ver hacia alguna de las cuatro esquinas del tablero, con esto el intérprete deberá saber que letra es la que se desea y decirla en voz alta, para los números, el usuario deberá indicar con el primer vistazo el bloque donde se encuentran y el intérprete deberá posicionar su dedo sobre cada uno de los números hasta que el usuario cierre los ojos o realice el gesto previamente establecido.

Como se puede notar con el cuadro ETRAN y con el tablero Corradina se requiere de un intérprete quien es el que va descifrando las palabras que el usuario quiere expresa y anotándolas en un papel, esto puede ser un tanto tedioso o no siempre se

[2] Recuperada de https://sites.google.com/site/vicortega2/etran [Diciembre, 2016]

puede contar con un intérprete a todas horas, por tanto, se han desarrollado otro tipo de técnicas que incluyen distintas tecnologías (Ruiz, 2016).

2.2.2 Iriscom.

Iriscom es una fórmula de comunicación para personas que han perdido los recursos que, digamos la gente utiliza en la vida corriente para comunicarse. Podemos decir, pues que es un recurso de comunicación para personas que tienen una discapacidad máxima; una persona que pierde el habla puede utilizar las manos para comunicarse y cuando pierdes la movilidad en las manos puedes utilizar la cabeza, comunicarte a través de un pequeño joystick, un montón de soluciones aplicadas al nivel de discapacidad que se tenga, pero cuando una persona pierde el control total del cuerpo y sólo controla el ojo, puede ser una gran solución; un sistema que permite comunicarse a través del control del ojo (Arrazola & Pedro, 2017).

Ilustración 2.3 Iriscom. [3]

Iriscom es un dispositivo que se conecta mediante un puerto USB a una computadora normal con sistema operativo Windows, este dispositivo está conformado por un emisor de haces infrarrojos y una cámara; los haces infrarrojos van dirigidos al ojo y la cámara va siguiendo el punto de encuentro de estos dos haces y lo transforman en movimientos del cursor de la computadora. Este sistema no es agresivo ni invasivo lo único que el usuario recibe es un haz de luz de 5 voltios que sería unas 20 veces menos que la luz de una lámpara incandescente.

[3] Recuperada de http://www.ite.educacion.es/formacion/materiales/146/cd/m4_dificultad_motora/acceso_con_la_mirada.html [Enero, 2017]

Como se puede ver en la Ilustración 2.3, el dispositivo Iriscom es muy sencillo, solo pesa 350 gramos, con lo que se vuelve un sistema muy manejable, sin embargo, este dispositivo tiene un costo bastante alto, alrededor de los 9 mil euros, al tipo de cambio en el 2017 equivale a unos 207 mil pesos mexicanos, haciendo este dispositivo si bien una buena opción para la comunicación de personas con ELA o incluso con alguna otra enfermedad que les impida el control motor de su cuerpo, no muy accesible por el costo que tiene.

2.2.3 Irisbond Primma.

Irisbond Primma es un sistema informático que permite el control de la computadora con el movimiento de los ojos, ofrece funciones de clic derecho, doble clic, arrastrar, lupa, modo lectura, y compatibilidad con la mayoría de programas que se ejecutan sobre el sistema operativo Windows.

Este sistema se compone de un dispositivo que realiza la captación del movimiento del ojo a través de luz infrarroja (Ilustración 2.4), de un programa llamado Prima que es la aplicación principal que sirve para la calibración y permite configurar el dispositivo, también incluye SmartPlaphoons que es una aplicación basada en el comunicador Plaphoons, esta aplicación incluye pictogramas con los que el usuario puede comunicarse de manera más sencilla, permitiendo la adaptación de pictogramas personalizados, así como un teclado para la escritura en páginas web y mensajería instantánea, el sistema se complementa con EyeLearn que es un software para la educación especial desarrollado por pedagogos y logopedas, este software facilita la iniciación al control con la mirada, contiene varios juegos a las necesidades de niños y personas con una afección cognitiva.

Este sistema además incluye opcionalmente un dispositivo extra y una aplicación domótica para el control del entorno con el cual se puede encender la televisión, apagar la luz, subir persianas, abrir una puerta, entre otras funciones. (IRISBOND, 2016).

Ilustración 2.4 Irisbond Primma. [4]

Este es quizá el sistema más popular para el control y comunicación de personas con escasa movilidad y si bien es un sistema muy completo, se debe mencionar que el sistema tiene un costo elevado entre los 16 y 20 mil euros correspondiendo a los distintos paquetes que ofrecen, que al tipo de cambio en el 2017 equivale entre unos 37 y 46 mil pesos mexicanos, y aunque comparándolo con el sistema antes mencionado (Iriscom), el costo es mucho más bajo pero aun así sigue siendo un sistema de difícil acceso para todos los usuarios es por eso que se decidió crear la propuesta de este trabajo.

2.2.4 Sistema de comunicación propuesto.

Como se puede apreciar, a pesar de que existen técnicas y sistemas de comunicación para un usuario con discapacidad motora, estos pueden depender de un intérprete, como en el caso de los tableros, del cual muchas veces se carece o estos sistemas que si bien son muy eficientes y que no solo permiten la comunicación sino que también permiten el control del entorno, suelen ser muy costosos, por tal motivo se ha pensado en la propuesta de sistema de comunicación a bajo costo que se menciona a lo largo de este trabajo, para conocer un poco más del funcionamiento y la implementación de esta propuesta a continuación se presentaran los componentes que se usaron para desarrollar este sistema.

[4] Recuperada de https://i.ytimg.com/vi/x2pVs9IeOrg/maxresdefault.jpg [Diciembre, 2016]

2.3. Sistemas de cómputo.

Un sistema de cómputo es un conjunto de elementos electrónicos que interactúan entre sí, para procesar y almacenar información de acuerdo a una serie de instrucciones.

Al interior de un sistema de cómputo pueden encontrarse elementos físicos denominados Hardware como una unidad de disco duro, una tarjeta de video, una tarjeta de sonido, entre otros, y el conjunto de instrucciones que manipulan estos elementos físicos se denomina Software.

Existe una gran variedad en el tamaño, costo y desempeño de los sistemas de cómputo, para el desarrollo de esta propuesta se realizó estudio de una clasificación especial de estos sistemas, las microcomputadoras, que son equipos de cómputo de tamaño pequeño y recursos limitados, algunos de los cuales se mencionan a continuación.

2.3.1. BeagleBone Black.

BeagleBone Black (Ilustración 2.5) es una de las microcomputadoras de Texas Instruments, se menciona está en particular por ser su computadora más popular, una de las más completas y que se consideró también para el desarrollo de esta propuesta.

Esta microcomputadora es de software y hardware libre, dicho en otras palabras, esta computadora tiene el código de programación así como los planos de su elaboración al alcance de todos, permitiendo con esto que cualquier usuario pueda modificar el código o incluso comprar los componentes y crear su propia computadora desde cero, es una computadora de bajo costo, alrededor de 60 dólares, que al tipo de cambio en el 2017 equivale a unos 1,300 pesos mexicanos, de bajo consumo ya que la placa requiere poco más de 2 W y se alimenta mediante una fuente externa de 5V, tiene un procesador Sitara ARM Cortex-A8 a 1 GHz de velocidad, memoria RAM de 512 MB, 4 GB de memoria flash, 2 conectores de expansión para agregar sensores y actuadores, 1 puerto Ethernet para la conexión a internet, 1 ranura microSD para expansión de memoria, 1 puerto micro HDMI para conectarla a cualquier televisor o monitor con ese puerto que es uno de los más

comunes y cuenta con 1 puerto USB multipropósito con protocolo OTG. (eLinux, 2016) (Coley, 2016) (beagleboard.org, 2016).

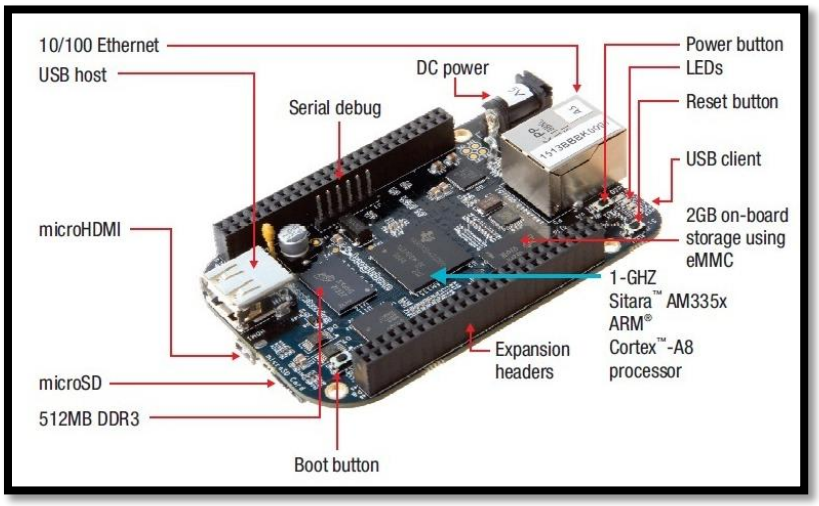


Ilustración 2.5 Microcomputadora Beagle Bone Black. [5]

Esta computadora además se entrega con el sistema operativo Debian, que es una distribución de Linux, ya instalado en su memoria interna con la opción de remplazarlo.

2.3.2. PCDuino 2.

PCDuino 2 es una microcomputadora del fabricante Allwinner, de igual manera Allwinner tiene a la venta distintas placas de desarrollo y microcomputadoras, pero se hace mención de su microcomputadora más popular que es PCDuino, cabe señalar que esta microcomputadora no es fabricada ni vendida por Arduino, solo lleva este nombre por la compatibilidad que tiene con las placas Arduino y los sensores y actuadores que estas usan.

La PCDuino 2 (Ilustración 2.6) dispone de un procesador Allwinner 10 AMR Cortex-A8 Dual-Core con una velocidad de 1Ghz, 1 GPU Mali 400, 1 GB en memoria RAM, 1 memoria flash interna de 4 GB, 1 puerto microSD para expandir memoria, 1 puerto de salida de video HDMI, es de bajo consumo ya que solo requiere 5V de alimentación y 2A de corriente, 1 puerto de Ethernet para la conexión a internet, tiene un tamaño de 13cm x 5cm lo que la hace muy compacta y de fácil manejo, cuenta con puertos de expansión para colocar sensores y actuadores, 1 puerto USB multipropósito con protocolo OTG y tiene un costo

[5] Recuperada de http://pds.exblog.jp/pds/1/201310/19/10/b0133710_19572731.jpg [Diciembre, 2016]

alrededor de los 90 dólares que al tipo de cambio en 2017 equivale a unos 2 mil pesos mexicanos (LinkSprite, 2016).

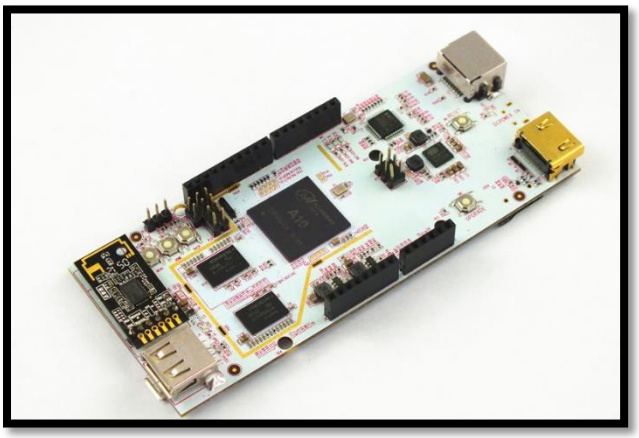


Ilustración 2.6 Pc Duino 2. [6]

Esta microcomputadora tiene el sistema operativo Ubuntu 12.04 preinstalado reemplazable, soporta otras distribuciones de Linux, así como Android ICS 4.0, habrá que mencionar que soporta los lenguajes de programación C, C++, Java y Python.

2.3.3. Raspberry Pi 2.

Raspberry Pi 2 es la microcomputadora más popular del fabricante Raspberry Pi Fundation, fue creada con el objetivo de incitar la programación y la educación en las ciencias de la computación, tienen en el mercado más modelos de microcomputadoras, incluso un modelo llamado Raspberry Pi Zero, con un costo de 5 dólares, que para tener ese costo y las capacidades que ofrece es una herramienta bastante atractiva.

La microcomputadora Raspberry Pi 2 (Ilustración 2.7) cuenta un procesador Broadcom ARM Cortex-A7 Quad-Core con una velocidad de 900 Hz, con la posibilidad de hacer overclock hasta 1GHz sin perder garantía, este procesador tienen una estructura RISC (Reduced Instruction Set Computer, Computadora con Conjunto Reducido de Instrucciones) generalmente utilizado en las microcomputadoras, una GPU VideoCore IV, cuenta con 1GB de memoria RAM, 40 GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General), que son los puertos en los cuales se pueden agregar sensores y actuadores, cuenta con 4

[6] Recuperada de: http://www.linksprite.com/linksprite-pcduino2/ [Diciembre, 2016]

puertos USB con protocolo OTG, un puerto de salida de video en alta definición HDMI con lo que permite conectarse a cualquier televisor o monitor ya que actualmente es el puerto más popular para señal de video, un puerto Ethernet para la conexión a internet, salida de audio Jack de 3.5mm, que es la salida de audio estándar al cual se le pueden conectar bocinas o audífonos convencionales, los modelos de Raspberry Pi no cuentan con memoria flash interna sin embargo tienen un puerto microSD que soporta memorias de hasta 32GB, es una microcomputadora de bajo consumo ya que se alimenta con una fuente externa de 5V y 2A de corriente, tiene dimensiones de 8.5cm x 5.4cm, además cuenta con un puerto de comunicación serial para una cámara y uno para un display, esta microcomputadora tiene un costo alrededor de 40 dólares que al tipo de cambio actual sería equivalente a unos 875 pesos mexicanos, con ella se promueve principalmente el lenguaje de programación Python, sin embargo se puede programar también en C, C++, Java, Tyni BASIC, Ruby, Pearl, entre otros (RASPBERRY PI FOUNDATION, 2016).

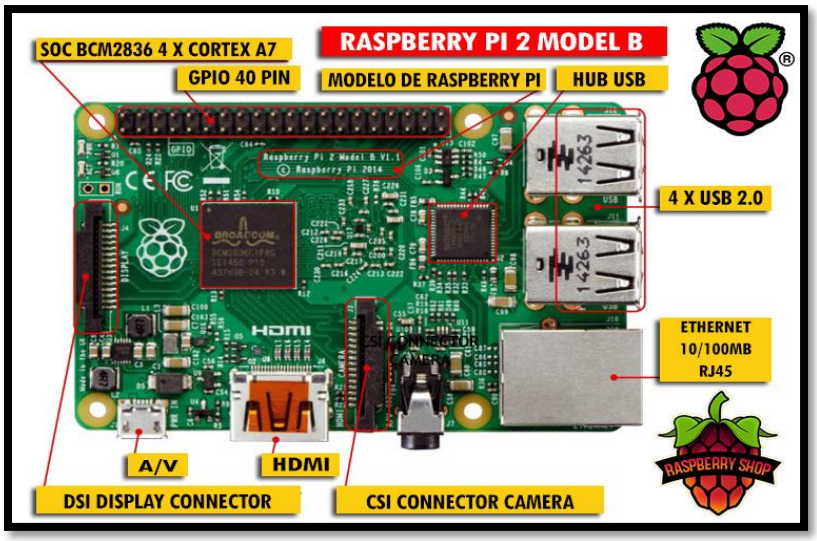


Ilustración 2.7 Raspberry Pi 2. [7]

La Raspberry Pi 2 puede ejecutar el sistema operativo Rasbian que es una versión de Debian, distribución de Linux, adaptada a la microcomputadora, soporta Ubuntu MATE, Pidora (Fedora), Web OS, Risc OS, una versión de Windows 10 para desarrolladores llamada Windows 10 IoT Core, Firefox OS, Google Chromium OS, Android, Unix, inclusive esta microcomputadora puede usarse como centro multimedia ya que soporta la reproducción de video en HD y existe también una versión de sistema operativo llamado RetroPie que te permite ejecutar emuladores

[7] Recuperada de: https://www.raspberryshop.es/hardware-raspberry-pi.php [Diciembre, 2016]

de juegos árcade de antaño, hasta el de la consola PSOne de Sony, debido a que no tiene memoria flash interna la instalación del sistema operativo se realiza sobre la memoria microSD externa, por lo cual te permite tener distintas memorias con sistemas operativos diferentes y cambiar de una a otra en el momento que se requiera, convirtiéndola en una herramienta muy atractiva. (DOUTEL, 2016) (Upton & Gareth, 2014)

Esta es la microcomputadora que se eligió para desarrollar la propuesta de sistema de comunicación que se mencionará a lo largo de este trabajo de tesis, se eligió principalmente porque al comparar con los modelos anteriores se encontró que tiene un costo más accesible y características muy similares, además de que se ha convertido en una herramienta bastante popular y se puede encontrar mucha documentación relacionada, así como proyectos, si bien es cierto que la PCDuino 2 era también muy buena opción se descartó debido a que no cuenta con una salida de audio y para esta aplicación era fundamental contar con una para hacer uso del sintetizador de voz.

También se debe señalar que al investigar sobre la Raspberry Pi 2 se encontró mucho recurso en Linux sobre síntesis de voz y en programación Python, y como esta microcomputadora está muy orientada a esa programación la hizo la ideal para esta aplicación.

2.4. Sistemas operativos para Raspberry Pi 2.

Como ya se mencionó las distintas microcomputadoras soportan distintos sistemas operativos, pero cada uno es diferente o están optimizados para una cosa en particular, es por eso que en este punto se hará mención de algunos de los sistemas más importantes que hay para la Raspberry Pi 2 que es la herramienta que se utiliza para el desarrollo de esta propuesta.

2.4.1. Windows 10 IoT Core.

Esta es una versión de Windows 10 que esta optimizada para dispositivos pequeños que pueden o no tener una pantalla, se puede ejecutar sobre la Raspberry Pi 2, la Raspberry Pi 3, sobre la Arrow DragonBoard 410c y sobre MinnowBoard MAX, este sistema operativo aporta el poder de Windows a estos dispositivos y facilita la

integración de elementos importantes para cualquier proyecto, como lo son las interfaces de usuario, la búsqueda en la red y servicios basados en la nube.

Hay que señalar que esta no es una versión de Windows 10 como la que se puede observar en una PC o laptop, simplemente por qué es una versión muy limitada que es creada por desarrolladores para desarrolladores con el fin de crear e inventar soluciones y proyectos adaptadas al internet de las cosas, esta no es una plataforma que te permita abrir documentos, utilizar un navegador, o ejecutar alguna de las aplicaciones a las que se está acostumbrado en una computadora portátil, se debe estar muy consciente de eso, de que este sistema nos va a permitir controlar actuadores y adquirir datos de sensores que se pueden conectar mediante los puertos de propósito general que lleguen a tener las distintas microcomputadoras antes mencionadas. La instalación de este sistema operativo es relativamente sencilla, ya que solo basta con que se tenga una memoria microSD, una computadora con el sistema operativo Windows 10 y con un programa, que en la sección para desarrolladores de la página oficial de Windows se puede encontrar, se carga la imagen del sistema a la tarjeta microSD, Windows ofrece una guía completa para la instalación y el primer arranque de este sistema, sin embargo, eso no lo veremos dentro de este trabajo ya que este no será el sistema que se usará para el desarrollo de esta propuesta aunque se tuvo en cuenta para hacerlo. (Microsoft, 2016) (Pastor, 2016)

2.4.2. Risc OS.

El sistema operativo Risc OS fue creado para la plataforma Acorn, una serie de ordenadores del reino Unido, en dichos países que conforman el reino unido y en parte de Alemania albergó, un creciente éxito.

El gran auge de esta plataforma fue contemporáneo al uso mayoritario de ordenadores ATARI y Amiga.

RISC OS es un sistema operativo nuevo y diferente para Raspberry Pi. No es Linux, no es Unix, no está basado en ningún otro sistema operativo. Es el primer ARM OS, iniciado en 1987.

Es pequeño y rápido. RISC OS es un sistema operativo de escritorio completo, donde el sistema central, incluido el sistema de ventanas y algunas aplicaciones, encaja dentro de 6 MB. Se desarrolló en un momento en que la computadora de

escritorio más rápida era un 8MHz ARM2 con 512KB de RAM. Esto significa que es rápido y sensible en hardware moderno.

RISC OS es un SO de usuario único, lo que significa que hay muy poca seguridad, no es ideal para la banca por Internet, pero muy útil cuando se quiere hacer uso y programar los componentes internos del sistema operativo.

Como SO de escritorio completo, también hay un montón de software de escritorio tradicional disponible como programas de dibujo y editores de escritorio. RISC OS era grande en la educación del Reino Unido en los años 90, y hay un catálogo grande detrás del software educativo (theom, 2017) (Revill, 2017).

2.4.3. Raspbian.

Raspbian es una versión no oficial de Debian Wheezy armhf con ajustes de compilación ajustados para producir código optimizado que se ejecutará en el Raspberry Pi. Esto proporciona un rendimiento significativamente más rápido para aplicaciones que hacen un uso intensivo de operaciones aritméticas. Todas las demás aplicaciones también obtendrán cierto rendimiento mediante el uso de instrucciones avanzadas de la CPU ARMv6 en Raspberry Pi.

Aunque Raspbian es principalmente los esfuerzos de Mike Thompson (mpthompson) y Peter Green (plugwash), también se ha beneficiado enormemente del apoyo entusiasta de los miembros de la comunidad de Raspberry Pi que desean obtener el máximo rendimiento de su dispositivo.

Raspbian intenta mantenerse tan cerca de Debian como sea razonablemente posible. Debian es utilizado por millones de usuarios en todo el mundo sobre una base diaria y hay una gran cantidad de conocimientos y documentación sobre el uso de Debian en toda la web. Cualquier información que encuentres que se aplique a Debian casi seguramente se aplicará a la misma versión de Raspbian.

El objetivo de Raspbian es convertirse en el sistema operativo líder de elección para todos los usuarios de la Raspberry Pi. Este objetivo se ha logrado en gran medida. (Raspbian, 2017)

2.4.4. Ubuntu MATE.

Ubuntu es uno de los sistemas operativos de escritorio basados en Linux más grandes en el mundo, si no es el más grande. Linux está en el corazón de Ubuntu y hace posible crear sistemas operativos seguros, potentes y versátiles, como Ubuntu y Android. Android está ahora en manos de miles de millones de personas en todo el mundo y también es alimentado por Linux.

Ubuntu está disponible en una serie de diferentes versiones, cada una viene con su propio entorno de escritorio. Ubuntu MATE toma el sistema operativo base de Ubuntu y agrega el escritorio MATE.

MATE es la implementación de un entorno de escritorio e incluye un administrador de archivos que puede conectarlo a sus archivos locales y en red, un editor de texto, una calculadora, un visor de imágenes, un visor de documentos, un monitor del sistema y un terminal. Todos los cuales son altamente personalizables y gestionados a través de un centro de control, proporciona un entorno de escritorio intuitivo y atractivo utilizando componentes tradicionales, lo que significa que, si alguna vez has utilizado Microsoft Windows o Apple Mac OS, se sentirá muy familiar.

El escritorio MATE tiene una rica historia y es la continuación del escritorio GNOME2, que era el entorno de escritorio predeterminado en muchos sistemas operativos Linux y Unix durante más de una década. Esto significa que el escritorio MATE es sencillo y muy fiable.

Lo que Linux, Ubuntu y el escritorio MATE tienen en común es que son de código abierto. El software de código abierto es un software que puede ser libremente utilizado, cambiado y compartido por cualquier persona. En pocas palabras Ubuntu MATE es libre, en el verdadero sentido de la palabra.

Aunque se pueden encontrar algunas versiones (distribuciones) de Linux para la compra, la gran mayoría se proporcionan de forma gratuita, como Ubuntu MATE. Software de código abierto tiene licencia de una manera que permite a cualquiera darlo de forma gratuita, sin restricciones. Por ejemplo, la licencia da a cualquier miembro de la comunidad de usuarios la libertad de usar Linux para cualquier propósito, distribuir, modificar, redistribuir o incluso vender el sistema operativo. Si modifica y luego redistribuye Linux con sus modificaciones, la licencia le exige

que envíe sus modificaciones para su posible inclusión en futuras versiones. No hay ninguna garantía de que esto ocurra nunca, pero si se hace es mejor, entonces los cambios podrían ser incluidos en la próxima versión de Ubuntu MATE.

Así es como Ubuntu MATE se puede mejorar y hacer crecer continuamente sin tener que cobrar dinero a los usuarios. Muchos de los usuarios de Linux son corporaciones que utilizan el sistema operativo para ejecutar sus negocios o incluirlo en sus productos. Muchas de estas corporaciones proporcionan arreglos y nuevas características para Linux mientras usan el software para sus negocios. Estas mejoras se devuelven a la distribución de Linux y el software mejora como resultado.

A diferencia de Windows y OSX, Linux no es creado y soportado por una sola compañía. Es compatible con Intel, Redhat, Linaro, Samsung, IBM, SUSE, Texas Instruments, Google, Canonical, Oracle, AMD y Microsoft. Más de 4.000 desarrolladores contribuyeron a Linux en los últimos 15 años.

Ubuntu MATE es un sistema operativo estable y fácil de usar con un entorno de escritorio configurable. Es ideal para aquellos que quieren más de sus computadoras y prefieren componentes de escritorio tradicional. Con modestos requisitos de hardware es adecuado para estaciones de trabajo modernas, microcomputadoras y hardware más antiguo por igual. Ubuntu MATE hace que las computadoras sean rápidas aun teniendo hardware básico o viejo (Ubuntu MATE, 2017).

Ubuntu MATE es el sistema operativo que se utilizó para la implementación del sistema de comunicación, se eligió por su gran popularidad y por qué está basado en una de las distribuciones de Linux más usadas (Ubuntu), además por que inicialmente la pc en la que se desarrollaron las pruebas contaba con Ubuntu, es por eso que se buscó un sistema operativo similar, convirtiendo a Ubuntu MATE la mejor opción, además, de que es un sistema optimizado para microcomputadoras y que es muy estable ejecutándose sobre la Raspberry Pi 2 y hay mucha documentación en la red, sin embargo, se debe aclarar que el sistema se podría ejecutar sobre otro sistema operativo como los mencionados en este tema.

2.5. Lenguajes de programación para Raspberry Pi 2.

Los diferentes sistemas operativos que se pueden ejecutar sobre la microcomputadora Raspberry Pi 2 soportan distintos lenguajes de programación, es por eso que a continuación se hará mención de algunos que se consideraron para este caso.

2.5.1. Lenguaje C.

Lenguaje de programación C. También conocido como “Lenguaje de programación de sistemas” desarrollado en el año 1972 por Dennis Ritchie para UNIX un sistema operativo multiplataforma. El lenguaje C es del tipo lenguaje estructurado como son Pascal, Fortran, Basic. Sus instrucciones son muy parecidas a otros lenguajes incluyendo sentencias como if, else, for, do y while.

Aunque C es un lenguaje de alto nivel (puesto que es estructurado y posee sentencias y funciones que simplifican su funcionamiento) tenemos la posibilidad de programar a bajo nivel (como en el Assembler tocando los registros, memoria etc.). Para simplificar el funcionamiento del lenguaje C tiene incluidas librerías de funciones que pueden ser incluidas haciendo referencia la librería que las incluye, es decir que si queremos usar una función para borrar la pantalla tendremos que incluir en nuestro programa la librería que tiene la función para borrar la pantalla.

La programación en C tiene una gran facilidad para escribir código compacto y sencillo a la vez. En el lenguaje C no tenemos procedimientos como en otros lenguajes solamente tenemos funciones los procedimientos los simula y esta terminante mente prohibido escribir funcione. Los archivos en C se escriben en texto puro de ASCII del Dos si se escribe en WORD por ejemplo el mismo incluye muchos códigos no entendidos por el compilador y generara errores; una vez escrito se debe pasar a compilar el archivo; los archivos tienen 2 Extensiones archivo.C que es el archivo a compilar el que contiene todos los procedimientos funciones y código de nuestro programa y archivo.h que son las librerías que contienen las funciones de nuestro programa. Cada instrucción que pasemos a poner en C va seguida de un punto y coma para decirle al compilador que hasta ahí llega la

instrucción simula un Enter del teclado. Ejemplo: clrscr(); /* borra la pantalla */ (EcuRed, 2017).

2.5.2. Java.

El lenguaje para la programación en Java, es un lenguaje orientado a objeto, de una plataforma independiente, este lenguaje fue desarrollado por la compañía Sun Microsystems, con la idea original de usarlo para la creación de páginas WEB.

Con la programación en Java, se pueden realizar distintos aplicativos, como son applets, que son aplicaciones especiales, que se ejecutan dentro de un navegador al ser cargada una página HTML en un servidor WEB, Por lo general los applets son programas pequeños y de propósitos específicos.

Otra de las utilidades de la programación en Java es el desarrollo de aplicaciones, que son programas que se ejecutan en forma independiente, es decir con la programación Java, se pueden realizar aplicaciones como un procesador de palabras, una hoja que sirva para cálculos, una aplicación gráfica, etc. en resumen cualquier tipo de aplicación se puede realizar con ella. Java permite la modularidad por lo que se pueden hacer rutinas individuales que sean usadas por más de una aplicación, por ejemplo, tenemos una rutina de impresión que puede servir para el procesador de palabras, como para la hoja de cálculo.

La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente Servidor, como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente, y de esta forma logra distribuir el trabajo a realizar (Anónimo, 2017).

2.5.3. Python.

Python es un lenguaje de programación de propósito general, orientado a objetos, que también puede utilizarse para el desarrollo web.

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo,

lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

Este lenguaje de programación es muy popular gracias a la gran cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero, gracias también a la sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C, también a la cantidad de plataformas en las que se puede desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros, Además, Python es gratuito, incluso para propósitos empresariales.

El creador del lenguaje es un europeo llamado Guido Van Rossum. Hace ya más de una década que diseño Python, ayudado y motivado por su experiencia en la creación de otro lenguaje llamado ABC. El objetivo de Guido era cubrir la necesidad de un lenguaje orientado a objetos de sencillo uso que sirviese para tratar diversas tareas dentro de la programación que habitualmente se hacía en Unix usando C.

El desarrollo de Python duró varios años, durante los que trabajó en diversas compañías de Estados Unidos. En el 2000 ya disponía de un producto bastante completo y un equipo de desarrollo con el que se había asociado incluso en proyectos empresariales. Actualmente trabaja en Zope, una plataforma de gestión de contenidos y servidor de aplicaciones para el web, por supuesto, programada por completo en Python.

Es un lenguaje de propósito general ya que se pueden crear todo tipo de programas no es un lenguaje creado específicamente para la web, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas.

Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip.

Python está en movimiento y en pleno desarrollo, pero ya es una realidad y una interesante opción para realizar todo tipo de programas que se ejecuten en cualquier máquina (Alvarez, 2017).

Python puede ejecutarse desde la terminal del sistema operativo o en un IDLE a través de instrucciones, una instrucción es una orden que se le da a un programa de computadora que actúa como intérprete del mismo, para así realizar una tarea específica.

Este es el lenguaje que se eligió para la creación de la propuesta de este trabajo se eligió principalmente porque es un lenguaje de programación sencillo, muy utilizado en Linux y además porque su entorno de programación Python IDLE ya se encuentra instalado al cargar el sistema operativo.

2.6. Interfaz gráfica para Raspberry Pi 2.

Para la implementación de esta propuesta se requiere crear una interfaz gráfica y ya con el lenguaje de programación elegido se proceden a analizar los distintos programas que pueden servir para la creación de dicha interfaz.

2.6.1. PyQT.

PyQt es un módulo de la biblioteca gráfica Qt para el lenguaje de programación Python. La biblioteca está desarrollada por la firma británica Riverbank Computing y está disponible para Windows, GNU/Linux y Mac OS X bajo diferentes licencias.

PyQt tiene un conjunto de enlaces Python para el conjunto de herramientas Qt y está disponible para todas las plataformas soportadas por Qt, incluyendo Windows, Linux, UNIX, Mac OS / X y sistemas embebidos, tales como el Sharp Zaurus y el iPAQ de Compaq. Se han probado en las versiones de Qt 1.43 a 3.3.4, Qt no-comerciales, Qtopia 1.5.0 y las versiones de Python 1.5 a la versión 2.4.1. Qt / Embedde.

PyQt está disponible bajo la licencia GPL para su uso con la versión GPL de Qt, la licencia “aa” comerciales para su uso con la versión comercial de Qt, una licencia no comercial para su uso con la versión no comercial de Qt v2, y una licencia para la educación de uso en la versión educativa de Qt.

También hay una versión de evaluación de PyQt para Windows. Esto se debe utilizar con la versión de evaluación correspondiente de Qt.

PyQt es construido usando SIP (una herramienta para la generación de módulos de Python extensión de C ++ bibliotecas de clases). SIP v4.2 o posterior debe estar instalado con el fin de crear y ejecutar esta versión de PyQt. (Ecu Red, 2017)

2.6.2. PyCharm IDE.

Pycharm es un entorno de desarrollo integrado, IDE por sus siglas en inglés, muy completo, creado por JetBrains. Este IDE es profesional y viene en dos modalidades: una edición Free y otra muy completa privada que apunta a empresas de desarrollo de software. La popularidad del IDE Pycharm se puede medir a partir de que grandes empresas como Twitter, Groupon, Spotify, ebay y telefónica, han utilizado éste para su trabajo.

La mayoría de sus características están disponibles en la versión gratuita, se integra con IPython, soporta Anaconda, así como otros paquetes científicos como matplotlib y NumPy.

Características como desarrollo remoto, soporte de bases de datos, soporte de frameworks de desarrollo web, etc, están disponibles solo para la edición profesional de PyCharm.

Algo muy útil de Pycharm es su compatibilidad con múltiples marcos de desarrollo web de terceros como Django, Pyramid, web2py, motor de aplicaciones Google y Flask, lo que lo convierte en un competo IDE de desarrollo de aplicaciones rápidas.

PyCharm proporciona análisis de código, un depurador gráfico, un probador de unidad integrado, integración con sistemas de control de versiones (VCSes), y soporta el desarrollo web con Django.

Es multiplataforma, con versiones de Windows, macOS y Linux. La edición de la comunidad se libera bajo licencia de Apache, y hay también edición profesional lanzada bajo licencia propietaria y esta tiene características adicionales (pythondiario, 2017).

2.6.3. Tkinter (Tk).

Tkinter es un módulo de la biblioteca gráfica Tcl/Tk para el lenguaje de programación Python. Se considera un estándar para la interfaz gráfica de usuario (GUI) para Python y es el que viene por defecto con la instalación del IDLE Python.

Tk es un módulo de python que nos dota de unas funciones para el desarrollo de Interfaces de usuario, está orientado a diseñar interfaces gráficas para aplicaciones de escritorio a través del lenguaje de programación Python, sirve para crear las ventanas a nivel visual que se pueden observar en el escritorio de una computadora, facilitando la comunicación de la computadora con el usuario ya que crea un entorno grafico de fácil identificación.

Tkinter demuestra la sencillez y potencia del lenguaje Python ya que sin necesidad de salir del mismo o complicarse usando extensiones como Glade o GTK3 que implican una multitud de librerías y volver más engorroso el proceso de desarrollo de software.

Aunque Glade y GTK son útiles por que se utiliza una herramienta de desarrollo visual (IDE RAD = Entorno de Desarrollo integrado, Desarrollo de aplicaciones rápidas) y eso facilita su uso, se recomienda el uso de Tkinter porque te enseña sin salir de Python y en el mismo editor como crear y como funciona una aplicación visual de escritorio. Después se puede migrar a otros entornos de desarrollo más visuales, pero es necesario comprender como funciona una aplicación desde la base y Tkinter es perfecto para esto es por su sencillez y el beneficio de programar la interfaz desde el mismo lenguaje que el algoritmo principal que se eligió este módulo para desarrollar la interfaz gráfica.

2.7. Métodos de entrada para Raspberry Pi.

Con la parte de programación ya definida, se procede a elegir el sensor que recibirá la señal del paciente que permita elegir las letras en la interfaz para formar las frases, para esto se analizarán los métodos de entrada que pueden ser usados.

2.7.1. Joystick.

Dispositivo que se conecta con una computadora o videoconsola para controlar de forma manual un software, consiste en una palanca que se pivotea sobre una base y reporta el ángulo de dirección al equipo al que está conectado, especialmente juegos o programas de simulación. En español llamado también palanca de mando. El nombre proviene del inglés joy que significa alegría y stick que es palo (ALEGSA, 2017).

En esta investigación solo se analizaron los joysticks relacionados a las computadoras, pero se pueden encontrar también para controlar máquinas como sillas de rueda, cámaras de vigilancia, grúas, etc.

Un joystick suele estar formado por dos potenciómetros a 90º que transforman el movimiento en X e Y del mando en una señal eléctrica proporcional a su posición y que además suele incluir un botón pulsador en el eje Z, así pues, suelen tener 5 pines: X, Y, botón y 5V más GND (Ilustración 2.8).

Ilustración 2.8 Modulo Joystick 5v.[8]

2.7.2. Sensor mioeléctrico.

Este sensor de señales mioeléctricas es de la marca MyoWare y mide, filtra, rectifica y amplifica la actividad eléctrica de un músculo y produce una señal análoga que puede ser fácilmente leída por un microcontrolador haciendo que se pueda controlar un software con los músculos.

[8] Recuperada de: http://www.alegsa.com.ar/Dic/joystick.php [Junio, 2017]

Tienen la capacidad de detectar el impulso eléctrico de un músculo, y por medio de un acondicionamiento de señal, amplificación y filtrado se interpreta una orden. En un movimiento natural siempre hay un músculo dominante, midiendo su impulso eléctrico se puede indicar al software lo que se desea hacer.

Este es el sensor muscular MyoWare, es un sensor de electromiografía todo-en-uno (EMG) de Advancer Technologies. La placa MyoWare actúa mediante la medición de la actividad eléctrica filtrada y rectificada de un músculo; la salida de 0-Vs dependiendo la cantidad de actividad en el músculo seleccionado, donde Vs representa el voltaje de la fuente de alimentación.

En otras palabras: pega unos electrodos, lee la tensión y flexión de algunos músculos.

El sensor muscular MyoWare cuenta con un diseño portátil que le permite conectar paneles sensores biomédicos directamente a la propia placa para deshacerse de los molestos cables. Esta placa también incluye una serie de otras características, incluyendo, voltaje de un solo suministro de 3.1V + a + 5V, la salida RAW EMG, pines de alimentación de polaridad protegida, indicadores LED y un interruptor de encendido/apagado.

La medición de la actividad muscular mediante la detección de su potencial eléctrico, se hace referencia como la electromiografía (EMG), se ha utilizado tradicionalmente para la investigación médica. Sin embargo, con la llegada de microcontroladores y circuitos integrados cada vez más reducidos aún más potentes, circuitos y sensores EMG han encontrado su camino en todo tipo de sistemas de control (tdrobotica, 2017).

Características:

- Diseño portátil
- Fuente de alimentación Sencilla + 2.9V a + 5.7V
- Protección contra inversión de polaridad
- Indicadores LED
- Especialmente diseñado para microcontroladores
- Ganancia ajustable
- Dimensiones: 0,82" x 2,06"

Ilustración 2.9 Sensor miográfico.[9]

Mediante este sensor se puede detectar el movimiento de algunos músculos y como se estudió en este trabajo algunos pacientes pueden tener el control de ciertos músculos, es por esto que este sensor podría ser una alternativa para el control de este sistema, pero va a depender de la condición de cada paciente.

2.7.3. Sensor Capacitivo Touch.

El sensor capacitivo es simplemente un capacitor que detecta cuando una persona lo pulsa. El circuito integrado sobre el que viene montado dispone de todos los componentes para un funcionamiento correcto.

El sensor detecta cuando se le toca con la mano (o cualquier parte del cuerpo) generalmente este sensor detecta una determinada capacidad al pulsarlo nuestro cuerpo crea un efecto capacitivo sobre tierra suficiente para que el sensor salte a estado alto (Ilustración 2.10).

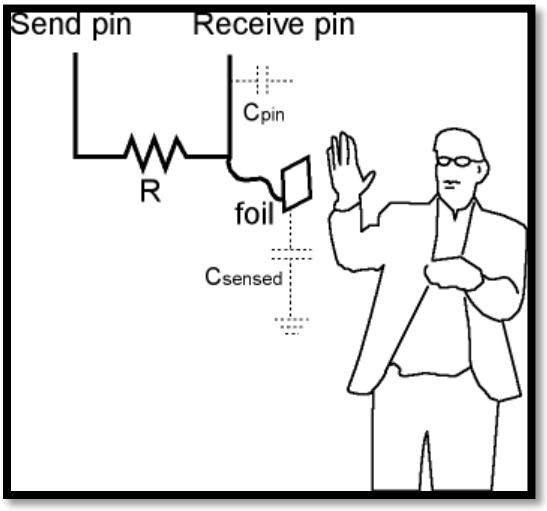


Ilustración 2.10 Diagrama de funcionamiento sensor Touch.[10]

[9] Recuperada de: http://tdrobotica.co/sensor-de-musculo-mioelectrico/397.html [Agosto, 2017]
[10] Recuperada de: https://www.drouiz.com/blog/2016/03/14/sensor-touch-arduino/ [Agosto, 2017]

El sensor se puede encontrar comercialmente en un módulo que utiliza el circuito integrado TTP223B que es el sensor Touch capacitivo (Ilustración 2.11). En su estado normal la salida del módulo se encuentra en cero lógico y baja consumo de corriente, cuando un dedo toca la posición correspondiente la salido del módulo se activa en uno lógico, sino se toca el módulo en 13 segundos vuelve a modo de bajo consumo (Drouiz, 2017).

Características:

- Voltaje de entrada: 2 ~ 5.5 VCD
- Salida alta: 0.8 VCD
- Salida baja: 0.3 VCD
- Tiempo de respuesta: 60 ms
- Dimensiones: 24 x 24 x 7.2 mm
- Peso: 2 g

Ilustración 2.11 Sensor Touch Capacitivo.[11]

Este es el sensor que se utilizará en el desarrollo de esta propuesta, debido a que es un sensor muy simple, de fácil configuración, de una respuesta rápida y precisa y que nos permite detectar la pulsación de un dedo o cualquier parte del cuerpo de forma adecuada.

2.8. Sintetizador de Voz.

Un sintetizador de voz es un software de computadora que puede transformar el texto a voz.

En una computadora el sonido lo generan los altavoces haciendo vibrar una membrana que es movida por un imán. La intensidad y la velocidad con la que se

[11] Recuperada de: http://img.dxcdn.com/productimages/sku_288078_1.jpg [Agosto, 2017]

mueve el imán, viene dada por la señal eléctrica que le llega al altavoz, así que lo que hay que hacer es enviar a los altavoces la señal adecuada. Para eso, primero debemos saber que sonido tenemos que producir, convirtiendo un texto en fonemas. Después, para cada fonema, generamos una señal periódica si queremos generar una vocal. Si lo que queremos generar es una consonante, tendremos que generar una señal de ruido. A continuación, se pasa por un modelo que imite las resonancias del tracto vocal y, por último, por otro modelo que imite el efecto que produce el medio en el que se expande la onda sonora. Cuando llegamos al siguiente fonema se intenta que el cambio entre uno y otro sea gradual, para que la voz sea más realista. Después de todo este procesamiento, la señal que hasta ahora sólo era información, se convierte en una señal eléctrica que se envía a los altavoces.

Para la creación de este sistema se requiere el uso de un sintetizador de voz que nos permita realizar las acciones antes mencionadas; a continuación, se describen alguno de los sintetizadores de voz considerados para el desarrollo del sistema de comunicación.

2.8.1. Gespeaker.

Gespeaker es un software que permite leer y grabar textos en formato wav. Su interfaz es sencilla y muy intuitiva (basta con escribir o pegar el texto en la ventana y pinchar reproducir). Por defecto, las voces que utiliza son excesivamente metálicas, pero es posible instalar las voces mbrola, algo menos robotizadas (Ilustración 2.12).

Ilustración 2.12 Interfaz Gespeaker.[12]

Dispone de una serie de controles para modificar el volumen, la velocidad de lectura, el tiempo de las pausas o la entonación que nos permiten "afinar" un poco la voz deseada. Junto a estos controles podemos elegir el idioma en el que será leído el texto escrito y el sexo de la voz (Ubumedia, 2017).

2.8.2. Festival.

Festival es un sistema de síntesis de voz, en su conjunto ofrece texto completo a voz a través de un gran número de APIs: desde el nivel consola, aunque un intérprete de instrucciones Scheme, como una biblioteca C ++, desde Java y una interfaz Emacs. Festival es multilingüe (actualmente inglés (británico y americano) y español) aunque el inglés es el más avanzado. Otros grupos liberan nuevos idiomas para el sistema.

El sistema está escrito en C ++ y utiliza la Edinburgh Speech Tools Library para la arquitectura de bajo nivel y tiene un esquema (SIOD) basado en intérprete de instrucciones para el control. La documentación se da en el formato FSF texinfo, formato que puede generar, un manual impreso, archivos de información y HTML.

Festival es software libre. Festival y las herramientas del habla se distribuyen bajo una licencia de tipo X11 que permite el uso comercial y no comercial sin restricciones por igual (CSTR, 2017).

[12] Recuperada de: https://ubumedia.wordpress.com/2009/08/31/gespeaker-texto-a-voz/ [Agosto,2017]

Características:

- Inglés (británico y americano), español y galés texto a voz.
- Módulos independientes de lenguaje configurable externamente:
- reglas de letra a sonido
- entonación y duración
- Sintetizadores de forma de onda:
- Motor de selección de unidades Multisyn
- Motor de síntesis paramétrico HTS
- Clustergen motor de síntesis paramétrica
- Motor de selección de unidades Clunits
- basados en difonos: LPC excitado residual (y PSOLA no para distribución)
- Soporte de base de datos MBROLA.
- distribuidos bajo licencia gratuita tipo X11
- Distribución portátil (Unix)
- Documentación on-line

2.8.3. eSpeak.

eSpeak utiliza un método de "síntesis de formantes". Esto permite que se proporcionen muchos idiomas en un tamaño pequeño. La voz es clara, y se puede utilizar a altas velocidades, pero no es tan natural o suave como los sintetizadores más grandes que se basan en grabaciones de voz humana, es compatible con Linux, Mac y Windows (eSpeak, 2017).

eSpeak está disponible como:

- Un programa de línea de instrucciones (Linux y Windows) para hablar el texto de un archivo o de stdin.
- Una versión de biblioteca compartida para uso de otros programas. (En Windows esto es un DLL).
- Una versión SAPI5 para Windows, por lo que se puede utilizar con lectores de pantalla y otros programas que admiten la interfaz SAPI5 de Windows.
- eSpeak ha sido portado a otras plataformas, incluyendo Android, Mac OSX y Solaris.

Características:

- Incluye voces diferentes, cuyos parámetros pueden ser alteradas.
- Puede producir salida de voz como un archivo WAV.
- SSML (Speech Synthesis Markup Language) es compatible (no completo), y también HTML.
- Tamaño compacto. El programa y sus datos, incluyendo muchos idiomas, totalizan aproximadamente 2 Mbytes.
- Puede utilizarse como front-end para las voces de diphone MBROLA.
- eSpeak convierte texto en fonemas con información de tono y longitud.

- Puede traducir texto en códigos de fonemas, por lo que podría ser adaptado como front end para otro motor de síntesis de voz.
- Potencial para otros idiomas.
- Las herramientas de desarrollo están disponibles para producir y afinar datos de fonemas.
- Escrito en C.

Idiomas:

El sintetizador de voz eSpeak soporta varios idiomas; sin embargo, en muchos casos estos son borradores iniciales y necesitan más trabajo para mejorarlos.

eSpeak hace síntesis de texto a voz para los siguientes idiomas, algunos mejores que otros.

Africano, albanés, aragonés, armenio, búlgaro, cantonés, catalán, croata, checo, danés, holandés, inglés, esperanto, estonio, farsi, finlandés, francés, georgiano, griego, hindú, húngaro, islandés, indonesio, irlandés, italiano, kurdo, letón, lituano, lojban, macedonio, malayo, malayalam, mandarín, nepalés, noruego, polaco, portugués, punjabí, rumano, ruso, serbio, eslovaco, español, swahili, sueco, tamil y galés.

Fonemas:

En general, se puede definir un conjunto diferente de fonemas para cada idioma, aunque en la mayoría de los casos, diferentes lenguas heredan el mismo conjunto básico de consonantes. Pueden agregar a estos o modificarlos según sea necesario.

Los mnemónicos del fonema se basan en el esquema de Kirshenbaum que representa los símbolos del alfabeto fonético internacional usando caracteres ASCII.

Los archivos de voz tienen distintos parámetros que pueden ser configurados individualmente, para modificar la voz como el usuario lo desee.

Parámetros de Voz:

- Tono
- Formante
- Eco
- Rugosidad
- Voces
- Consonantes
- Aliento

- Respiración
- Velocidad

Parámetros del lenguaje:

- Fonemas
- Diccionario
- Longitudes
- Amplitudes
- Dialecto

Este fue el programa que se utilizó para el desarrollo de este sistema de comunicación principalmente porque es compatible con Linux, el sistema operativo que se eligió para la Raspberry Pi, además de su tamaño compacto y su variedad de idiomas, así como la posibilidad de invocarlo a través de Python, el lenguaje de programación elegido para el desarrollo de esta propuesta.

Capítulo 3 Desarrollo Experimental.

En este capítulo se dará una descripción detallada de todos los componentes del sistema, interconexión y de algoritmos implementados.

3.1. Diagrama de Bloques.

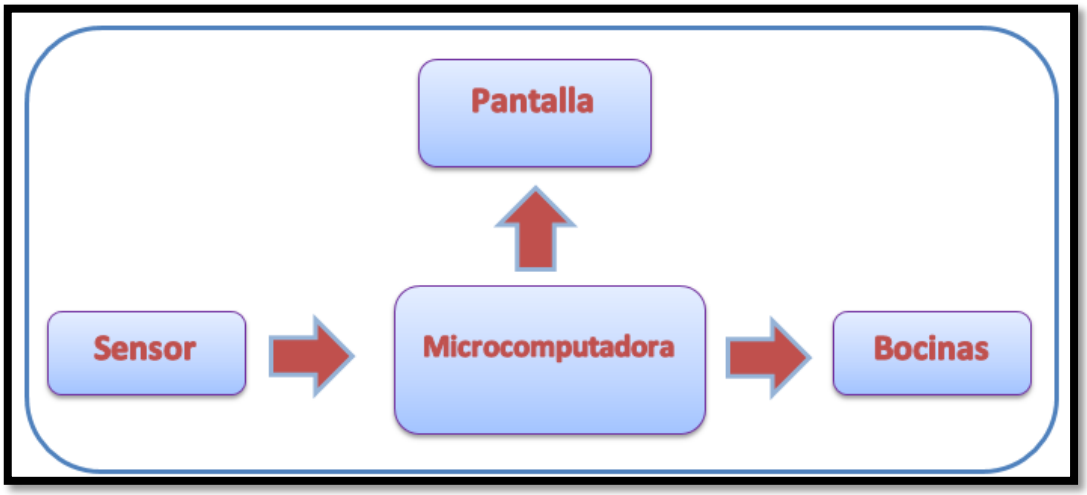


Ilustración 3.1 Diagrama a bloques.[13]

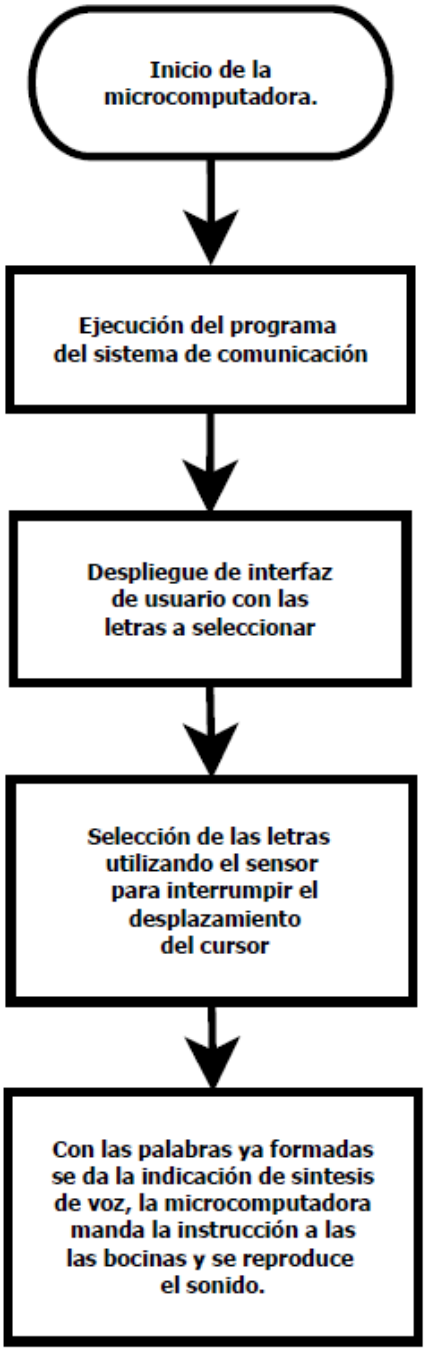
Dónde:

- **Microcomputadora**: es la Raspberry Pi 2 que se utilizó como núcleo del sistema.
- **Pantalla**: es la pantalla del monitor o de la tv a la que se conectó la Raspberry Pi 2.
- **Sensor**: es el sensor touch capacitivo.
- **Bocinas**: son las bocinas que se conectan mediante el puerto Jack 3.5 a la Raspberry Pi 2.

3.2. Diagrama de Flujo.

El diagrama de flujo es una forma esquemática para representa algoritmos e ideas o conceptos, el siguiente diagrama de flujo muestra las funciones a realizar por el sistema de comunicación mediante la microcomputadora y los componentes que lo conforman.

[13] Recuperada de: Autoría propia Belli G. [Agosto, 2017]



A continuación, se analiza a detalle el funcionamiento, así como la codificación implementada para cada uno de los componentes.

3.3. Implementación en Raspberry Pi.

Inicialmente el prototipo de esta propuesta se había implementado en una tarjeta de desarrollo Arduino Mega, la cual funcionaba bastante bien, sin embargo, a medida que se pensaba en más cosas para este sistema se descubrió que el rendimiento de esa tarjeta no sería suficiente, además de que desarrollarlo con esta herramienta

incluía la implementación de pantallas y hardware adicional que debía ser compatible, de manera que se buscaron alternativas para la creación de esta propuesta, y se encontró que existen algunas microcomputadoras en el mercado que aunque comparándolas por si solas con la tarjeta de desarrollo Arduino Mega son más costosas, presentan la ventaja de que no se requiere hardware adicional especial para estas, ya que se pueden conectar a internet de manera sencilla, se conectan a cualquier televisor de pantalla plana con conexión HDMI, que cuentan con salidas de audio estándar a los cuales se pueden conectar bocinas convencionales o incluso audífonos, también se debe considerar al tratarse de una microcomputadora se le pueden dar más usos como el acceso a internet incluso se podrían implementar sensores y actuadores, con lo cual lograr un control del entorno, sin embargo, eso no se desarrollará durante este trabajo ya que esta propuesta se concentra en crear un sistema de comunicación solamente.

Gracias al estudio que se desarrolló sobre este tipo de microcomputadoras se determinó que la Raspberry Pi 2, sería la elegida para el desarrollo de esta propuesta, la cual se seleccionó por su bajo costo y porque tiene la capacidad de cómputo suficiente para poder ejecutar todas las tareas computacionales de este sistema, se procede a instalar el sistema operativo que esta microcomputadora requiere, el sistema operativo que se seleccionó para instalar es una distribución de Linux llamada Ubuntu MATE, la cual se eligió por ser software libre y de bajo requerimiento de recursos.

Como primer paso, se adquirió el archivo imagen del sistema operativo con el que se arrancó la Raspberry Pi 2[14], se tuvo cuidado en seleccionar la versión para Raspberry Pi que es la que indica la Figura 3.2.

Una vez se tenga el archivo imagen, este se descomprimió y para esto el desarrollador del sistema operativo recomienda “7-Zip”[15] que es un programa que sirvió para realizar dicha descompresión, con el archivo imagen ya descomprimido (debe quedar un archivo con extensión .img), se le dio formato a la memoria microSD para después instalar el sistema operativo en ella, para hacer el formateo correspondiente el desarrollador de Ubuntu MATE recomienda el programa “SD

[14] El archivo imagen se encuentra en el sitio web del desarrollador https://ubuntu-mate.org/download/

[15] Este programa puede encontrarse en el sitio web del fabricante http://www.7-zip.org/

Formatter”.[16] Para instalar el sistema operativo en la memoria microSD se hizó uso de un último programa llamado “Win32 Disk Imager” [17] que sirvió para instalar el archivo imagen en la memoria microSD y con esto quedó listo el sistema operativo para iniciar la Raspberry Pi 2.

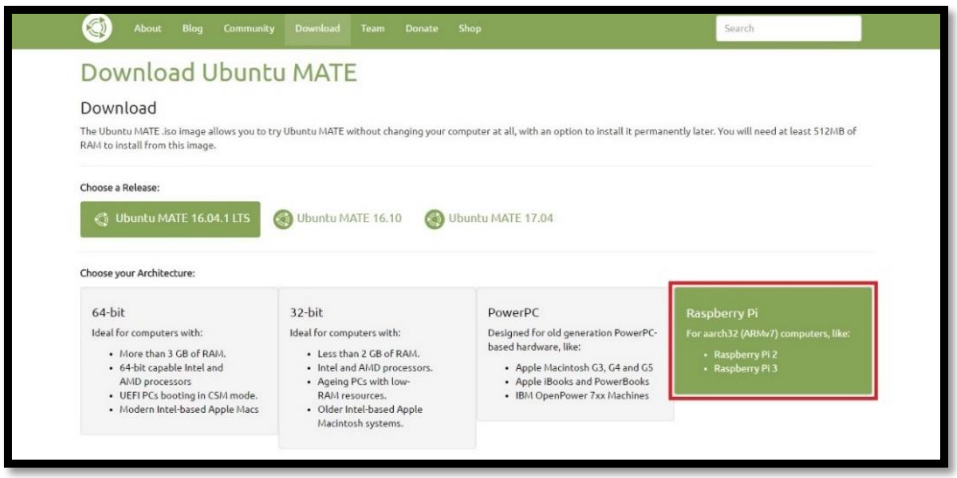


Ilustración 3.2 Captura de pantalla del sitio web del desarrollador de Ubuntu MATE que señala la versión de sistema operativo que se debe descargar.18

En la interfaz del programa Win32 Disk Imager se debe seleccionar la ruta donde se encuentra el archivo imagen (con extensión “.img”), para esto bastó con hacer clic en el icono de la carpeta señalado en la Ilustración 3.3, y ubicar el archivo.

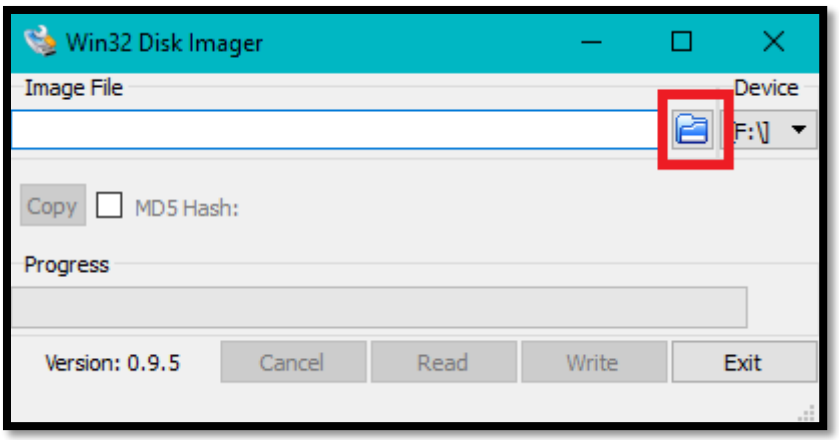


Ilustración 3.3 Interfaz del programa Win32 Disk Imager.19

En cuanto se selecciona la dirección del archivo correspondiente, se hizo clic en el botón “Write” inmediatamente salto una ventana de confirmación a la que se hizo

[16] Este programa se puede encontrar en el sitio web del fabricante https://www.sdcard.org/downloads/formatter_4/eula_windows/index.html
[17] Este programa puede encontrarse en el sitio web del desarrollador thttps://sourceforge.net/projects/win32diskimager/
[18] Para más detalle acerca de la instalación del sistema operativo consultar Anexo A. Recuperada de: https://ubuntu-mate.org/download/ [Diciembre, 2016]
[19] Recuperada de Win 32 Disk Imager, [Diciembre, 2016]

clic en “Yes” y la barra “Progress” de la interfaz comenzó a mostrar el estado de la escritura, en cuanto terminó apareció una ventana indicando que la escritura se había completado con lo que quedó correctamente instalado el sistema operativo en la microSD.

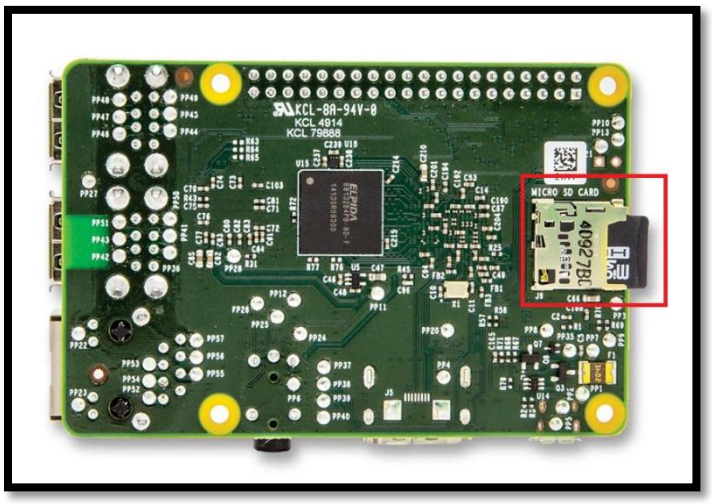


Ilustración 3.4 Puerto donde se debe colocar la memoria en la Raspberry Pi 2. [20]

Ya con el sistema operativo en la memoria microSD, esta se colocó dentro de la microcomputadora en el puerto de la memoria como se indica en la Ilustración 3.4.

Ilustración 3.5 Conexiones de la Raspberry Pi 2.[21]

Después se colocó el mouse y el teclado en los puertos USB, también se debió conectar el cable HDMI tanto a la pantalla como a la microcomputadora, así como el cable Ethernet tanto a la Raspberry Pi 2 como al modem para la conexión a

[20] Ilustración recuperada de: https://electronilab.co/tienda/raspberry-pi-2-modelo-b-armv7-1g-ram-8gb-microsd-noobs/ [Diciembre 2016].
[21] Recuperada de: Autoría propia, Belli G. [Agosto, 2016]

internet y una vez todo estuvo conectado, se procedió a conectar la microcomputadora a la fuente de alimentación como se muestra en la Ilustración 3.5.

La Raspberry Pi 2 comenzó con el primer arranque y lo primero que se mostró en pantalla fue el logotipo de Raspberry Pi seguido de lineas de codigo en fondo negro, como se puede apreciar en la ilustración 3.6.

Ilustración 3.6 Primera ventana de arranque inicial de Ubuntu MATE.[22]

3.4. Funcionamiento del programa del sistema de comunicación.

Hecho lo anterior, la microcomputadora quedó lista para la creación del programa que hace las funciones del sistema de comunicación, y para crearlo se hizo uso del entorno de desarrollo IDLE Python 3.5[23] que ya se encuentra incluido en el sistema operativo Ubuntu MATE, y el editor de texto que la IDLE incluye (Ilustración 3.7).

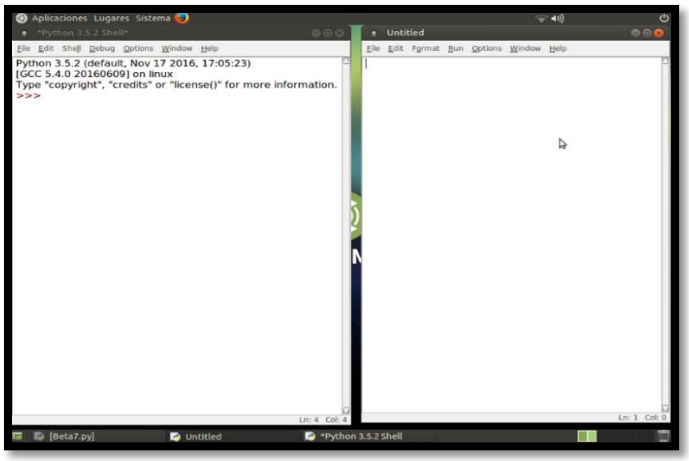


Ilustración 3.7 Intérprete y editor de textos de IDLE para Python 3.5.[24]

[22] Ilustración recuperada de: https://www.youtube.com/watch?v=5hSIZf7CmIU [Diciembre 2016]
[23] Para más detalle acerca del inicio de Python 3.5 consultar Anexo A.
[24] Recuperada de: Autoría Propia G. Belli [Diciembre, 2016]

El sistema de comunicación necesito de una interfaz gráfica que muestre el alfabeto para poder seleccionar las letras mediante el sensor de entrada, para la creación de esta interfaz se hizo uso de la biblioteca Tkinter ya incluida en Python, a continuación, se hará mención de la codificación necesaria para crear dicha interfaz.

El diagrama de flujo de la Ilustración 3.8 corresponde a la secuencia que debe seguir el cursor en las dos primeras líneas de la interfaz, servirá de ejemplo para mostrar la codificación para la interfaz gráfica.

La interfaz gráfica quedó como se muestra en la Ilustración 3.9, el diagrama de flujo de la Ilustración 3.8 servirá para explicar el funcionamiento del cursor, para poder consultar el diagrama de flujo completo, así como el resto de la codificación del sistema se recomienda ver el Anexo B.

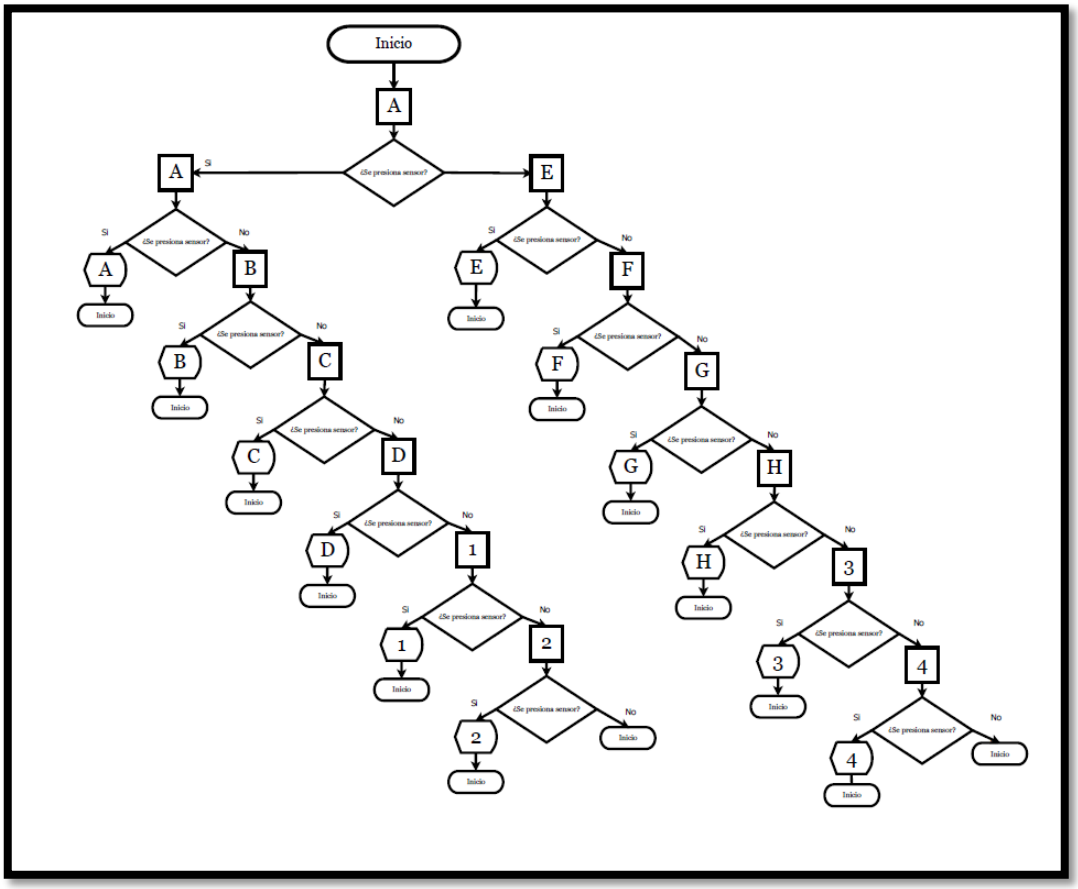


Ilustración 3.8 Diagrama de flujo para interfaz gráfica.[25]

[25] Recuperada de: Autoría propia, G. Belli [Junio, 2017]

Ilustración 3.9 Interfaz de la propuesta de Sistema de comunicación.[26]

3.5. Código de operación de la interfaz gráfica.

A continuación, se analizará la codificación que sirvió para hacer desplazar al cursor a través de las columnas de las 2 primeras filas, es decir a la fila que corresponde a los caracteres “A”, “B”, “C”, “D”, “1” y “2” y la fila con los caracteres “E”, “F”, “G”, “H”, “3” y “4”.

Primero se creó una función que muestra el cursor en cada una de las letras por ejemplo para la letra A y la letra E el código de cada función es el siguiente:

```
def CA():
    sv.set('''

      »A       B       C       D       1       2

       E       F       G       H       3       4
       ''')

def CE():
    sv.set('''

       A       B       C       D       1       2

      »E       F       G       H       3       4
       ''')
```

[26] Recuperada de: Autoría propia, G. Belli [Diciembre,2016]

Dónde:

- Def CA(): → es el parámetro que le da nombre a la función en este caso se llama CA (columna A).
- sv.set(´´´ Texto ´´´) → es el parámetro que le indica al programa que a la variable "set" (que se definirá más adelante) va a contener el texto que se encuentre entre las comillas.

Una vez se definió la función para cada una de las letras se definió una función que repetía las letras infinitamente, a la cual se le nombró Loop y se muestra a continuación en el código:

```
1.  def Loop():                         #Se define y se da nombre a la función (Loop).[27]
2.
3.      CA()                            #Se manda a llamar a la función "CA" definida con
4.                                      #anterioridad.
5.      delay()                         #Se llama a la función "delay", función que realiza el
6.                                      #retardo de tiempo en el cual se mostrará el cursor
7.  entre                               #una letra y otra.
8.      if (GPIO.input(22)):            #Función que pregunta si se ha presionado el sensor,
9.                                      #se explicará más adelante.
10.         LA()                        #Se manda a llamar a la función "LA" que es la que recorre
11.                                     #el cursor por las filas.
12.         delay()
13.         if (GPIO.input(22)):        #Se vuelve a preguntar si se ha presionado el sensor.
14.             x[i] = 'A'              #Si el sensor ha sido presionado se selecciona la letra "A"
15.             t()                     #Variable de apoyo que aumente en uno la variable i.
16.             print('A')              #Se imprime la letra "A" en pantalla.
17.             Loop()                  #Se invoca nuevamente a la función "Loop" para que se repita
18.                                     #el proceso.
19.         else:                       #Si no se presionó el sensor desplaza a la siguiente letra.
20.             LB()
21.             delay()
22.             if (GPIO.input(22)):
23.                 x[i]='B'
24.                 t()
25.                 print('B')
26.                 Loop()
27.             else:
28.                 LC()
29.                 delay()
30.                 if (GPIO.input(22)):
31.                     x[i]='C'
32.                     t()
33.                     print('C')
34.                     Loop()
35.                 else:
36.                     LD()
37.                     delay()
38.                     if (GPIO.input(22)):
39.                         x[i]='D'
40.                         t()
41.                         print('D')
42.                         Loop()
43.                     else:
44.                         N1()
45.                         delay()
46.                         if (GPIO.input(22)):
47.                             x[i] = '1'
48.                             t()
49.                             print('1')
50.                             Loop()
51.                         else:
52.                             N2()
53.                             delay()
```

[27] Esta es la manera de hacer comentarios en lenguaje Python, todo lo que se encuentre escrito después del símbolo "#" será ignorado por la programación y se considera un comentario, para poder identificar las funciones y haces anotaciones.

```
                                if (GPIO.input(22)):
                                    x[i] = '2'
                                    t()
                                    print('2')
                                    Loop()
                                else:
                                    Loop()

    CE()
    delay()
    if (GPIO.input(22)):
        LE()
        delay()
        if (GPIO.input(22)):
            x[i] = 'E'
            t()
            print('E')
            Loop()
        else:
            LF()
            delay()
            if (GPIO.input(22)):
                x[i] = 'F'
                t()
                print('F')
                Loop()
            else:
                LG()
                delay()
                if (GPIO.input(22)):
                    x[i] = 'G'
                    t()
                    print('G')
                    Loop()
                else:
                    LH()
                    delay()
                    if (GPIO.input(22)):
                        x[i] = 'H'
                        t()
                        print('H')
                        Loop()
                    else:
                        N3()
                        delay()
                        if (GPIO.input(22)):
                            x[i] = '3'
                            t()
                            print('3')
                            Loop()
                        else:
                            N4()
                            delay()
                            if (GPIO.input(22)):
                                x[i] = '4'
                                t()
                                print('4')
                                Loop()
                            else:
                                Loop()

        Loop()

v0 = Tk()                                   #Define una ventana nueva llamada "v0" con la
                                            #biblioteca "Tk"

v0.title("Sistema de Comunicación")         #Se le asigna como nombre a la ventana "Sistema de
                                            #Comunicación"

v0.geometry("800x600")                      #Se le da un tamaño a la ventana de 800 x 600
                                            #pixeles

sv = StringVar()                            #Se define una variable de tipo cadena, la cual
                                            #almacenará todos los caracteres seleccionados.


et = Label(v0, textvariable = sv, font = ( "Helvetica", 15 )) #Se define el widget "label" con

                                            #el nombre "et" con el tipo "text varible" que será
                                            #el objeto que irá cambiando su texto dependiendo de
                                            #la letra a mostrar.

et.pack()                                   #esta instrucción le indica al programa que el
                                            #widget deber ser incluido en la interfaz y
                                            #actualizarlo.

```

3.5.1. Código de operación de las funciones especiales.

A la interfaz gráfica se le agregaron funciones especiales para seleccionar la opción del espacio en la escritura, la de borrar un carácter, la de limpiar toda la pantalla para volver a escribir y la síntesis de voz, así que se analizará la parte del código de la última fila de la interfaz que es la que incluye estos caracteres especiales.

```
1.  def Limp():
2.
3.      x [0]=""                            #La función Limp se define para limpiar
4.      x [1]=""                            #los caracteres de la pantalla, para esto
5.      x [2]=""                            #se le asignan a todos los índices de la
6.      x [3]=""                            #variable cadena que incluye los caracteres
7.      x [4]=""                            #seleccionados un carácter vacío.
8.      x [5]=""
9.      x [6]=""
10.     x [7]=""
11.     x [8]=""
12.     x [9]=""
13.     x [10]=""
14.     x [11]=""
15.     x [12]=""
16.     x [13]=""
17.     x [14]=""
18.     x [15]=""
19.     x [16]=""
20.     x [17]=""
21.     x [18]=""
22.     x [19]=""
23.     x [20]=""
24.     x [21]=""
25.     x [22]=""
26.     x [23]=""
27.     x [24]=""
28.     x [25]=""
29.     x [26]=""
30.     x [27]=""
31.     x [28]=""
32.     x [29]=""
33.     x [30]=""
34.
35. CS()
36. delay()
37. if (GPIO.input(22)):
38.     RS()
39.     delay()
40.     if (GPIO.input(22)):
41.         x[i] = ' '                      #Para agregar el espacio basta con colocar
42.                                         #un carácter vacío en medio de las comillas
43.
44.         t()
45.         print(' ')
46.         Loop()
47.     else:
48.          RB()
49.          delay()
50.          if (GPIO.input(22)):
51.              x[i-1]  =''                #Para borrar un carácter se retrocede un
52.                                         #índice en la variable cadena y se
53.                                         #sustituye por un carácter vació.
54.              tm()
55.              print('_')
56.              Loop()
57.          else:
58.              RL()
59.              delay()
60.              if (GPIO.input(22)):
61.                  Limp()
62.                  t()
63.                  print('Limpiar')       #Para limpiar la pantalla se llama a la
64.                                         #función Limp.
65.                  Loop()
66.              else:
67.                    RV()
68.                    delay()
```

```
69.                     if (GPIO.input(22)):
70.                         g = x[0]+x[1]+x[2]+x[3]+x[4]+x[5]+x[6]+x[7]+
71.                          x[8] + x[9]+x[10]+x[11]+x[12]+x[13]+x[14]+
72.                          x[15]+x[16]+x[17]+x[18]+x[19]+x[20]+x[21]+
73.                 x[22]+x[23]+x[24]+x[25]+x[26]+x[27]+x[28]+
74.                 x[29]+x[30]
75.                         subprocess.call('espeak -v es+m1 -s130'.format(g),shell=True)
76.                                     #Esta es la línea de código que habilita
77.                                     #la extensión del sintetizador de voz.
78.                     else:
79.                          Loop()
```

El campo especial llamado “Voz”, activa al sintetizador de voz, con lo cual se puede escuchar lo escrito en la interfaz a través de las bocinas, su código es:

```
1.  subprocess.call('espeak -v es+m1 -s130'.format(g),shell=True)
```

Dónde:

- subprocess.call→ es la función sirve para invocar a un proceso nativo de Linux.
- espeak → este parámetro invoca al programa de Linux “eSpeak” que es el sintetizador de voz.
- -v → este parámetro le indica al programa “eSpeak” que tipo de voz es la que deberá usar de entre las que incluye el programa.
- es+m1 → son los parámetros que se le asignan a la instrucción que invoca al sintetizador de voz y le indican al programa que debe usar una voz en idioma español (es) y que es la voz masculina 1 (m1).
- -s130 → este parámetro le indica a “eSpeak” a qué velocidad de palabras por minuto deberá reproducir el sintetizador de voz.

3.6. Código de operación del sistema de comunicación.

Ya con la interfaz gráfica implementada, hizo falta crear el código principal del funcionamiento del sistema de comunicación, el cual se analizará a continuación.

```
1.  import subprocess                   #Esta línea invoca la librería subprocess al programa.
2.  from tkinter import *               #Esta línea invoca la librería tkinter al programa.
3.  import RPi.GPIO as GPIO             #Esta línea invoca la librería GPIO al programa.
4.  import time                         #Esla línea invoca la librería time al programa que será
5.                                      #la librería con la cual se harán los retardos de tiempo.
6.
7.  GPIO.setmode(GPIO.BCM)              #Esta línea activa los GPIO de la Raspberry.
8.
9.  v0 = Tk()                           #Se crea una ventana con las propiedades de Tkinter con el
10.                                     #nombre v0 (ventana 0).
11.
12. v0.title("Sistema de Comunicación") #Le pone título a la ventana v0.
13.
14. v0.geometry("800x600")              #Asigna las dimensiones de la ventana v0 800x600 pixeles.
15.
16. sv = StringVar()                    #Declara la variable con nombre sv de tipo cadena de
17.                                     #caracteres variables.
```

```
var = StringVar()

et = Label(v0, textvariable = sv, font = ( "Helvetica", 15 )) #Declara la variable et de tipo
                                                               #etiqueta de texto.

et.pack()                          #Esta línea le indica al programa que debe incluir la
                                   #variable et en la interfaz y hacerla visible
                                   #(empaquetarla a la interfaz)

lt = Label(v0, textvariable = var, font = ( " Helvetica", 17 ))

lt.pack()

x = ["","","","","","","","","","","","","","","","","","","","","","","","","",""] #se declara
                                   #la variable x, con caracteres vacíos, es donde se
                                   #almacenaran los caracteres que se irán seleccionando.

GPIO.setup(22, GPIO.IN)            #Esta línea declara el pin 22 de los GPIO de la raspberry
                                   como un pin de entrada.

Loop()                             #Esta línea invoca a la función Loop que es la que se creó
                                   #para mostrar la interfaz gráfica.

v0.mainloop()                      #Esta línea cierra el programa y mantiene en primer plano
                                   #la interfaz gráfica.
```

El código completo del sistema de comunicación puede encontrarse en el anexo C.

3.7. Diagrama de conexiones.

La Raspberry Pi se conectó a una fuente de alimentación la cual fue un eliminador de 5v a 2A de corriente, también se conectó mediante el cable HDMI a la pantalla, se conectó mediante el cable Ethernet a internet, y tanto el mouse como el teclado se conectaron mediante los puertos USB como se muestra en la Ilustración 3.10.

Ilustración 3.10 Diagrama de Conexión Raspberry Pi[28]

Adicionalmente conectó el sensor que se utiliza para interrumpir las letras en la interfaz gráfica, para la creación de este sistema se utilizó un sensor Touch, pero se debe señalar que este sensor podrá ser remplazado por alguno más adecuado para el paciente sin problema.

[28] Recuperada de: Autoría propia G. Belli [Agosto, 2016]

El diagrama que corresponde a la conexión del sensor Touch se puede apreciar en la Ilustración 3.11.

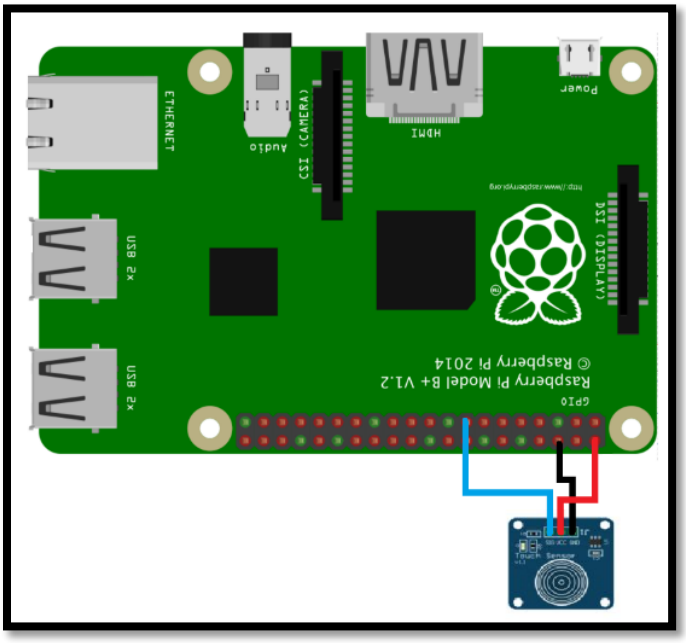


Ilustración 3.11 Diagrama de conexión del sensor Touch.[29]

El sensor Touch se alimenta de los pines de voltaje y GND con los que cuenta la Raspberry Pi, en este caso se está haciendo uso del pin 2 (5v) y el pin 6 (GND) de la Raspberry Pi, el pin que recibe la señal de que el sensor fue presionado es el pin físico número 15 que corresponde al pin GPIO 22 según la distribución de los pines de la Raspberry Pi (Ilustración 3.12).

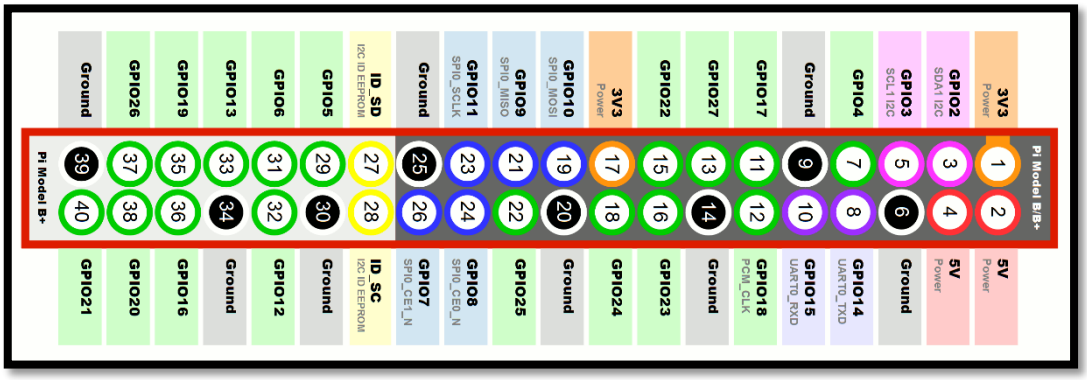


Ilustración 3.12 Distribución de pines GPIO de la Raspberry Pi [30]

[29] Recuperada de: Autoría propia, G. Belli. [Marzo, 2017]
[30] Ilustración recuperada de: http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/

Capítulo 4 Pruebas y Resultados.

4.1. Prueba de la Interfaz gráfica.

El desarrollo del sistema de comunicación comenzó por la creación de la interfaz gráfica, la cual se desarrolló sobre el sistema operativo Linux; a esta interfaz gráfica se le implementó el cursor, el cual se desplaza de manera automática y para lo cual se realizaron pruebas de la velocidad de transición de este cursor por cada letra, para esto se pidió a distintas personas que probaran la interfaz escribiendo frases.

Como resultado se pudo observar que al estar familiarizados con el alfabeto podían ubicar fácilmente la posición de las letras pero no atinaban a seleccionar la correcta ya que el cursor se desplazaba rápidamente, pero conforme iban utilizando el sistema iban adaptándose a la velocidad del cursor, es por esto que para pruebas iniciales se utilizaba un retardo de 0.8 segundos entre letra y letra pero conforme se utilizaba más el sistema este retardo podría disminuirse a la mitad en las mejores pruebas, por tal motivo se seleccionó un valor intermedio con lo que el sistema tiene un retardo de 0.6 segundos; sin embargo, este valor es modificable en la codificación ya que este retardo deberá ser adecuado a la condición del paciente así como la habilidad que se vaya adquiriendo para el uso de la interfaz.

La línea principal del programa del sistema es la que ajusta el retardo del cursor

```
1. retardo = 0.6  #Línea de código que permite ajustar el
2.                #retardo del cursor
```

Con lo cual basta remplazar el 0.6 para reducir o aumentar el tiempo de retardo en segundos.

4.2. Pruebas de Síntesis de voz.

El programa que se utilizó para realizar la síntesis de voz es eSpeak, este programa tiene distintos parámetros configurables, para los cuales se debieron realizar distintas pruebas y combinación de parámetros para hacer que la voz sonara inteligible, encontrando la siguiente configuración de parámetros, la cual se analizará a detalle.

```
1. espeak -v es+m1 -s130 "Texto a voz"
```

Lo primero que se realizó fue la configuración del idioma del sintetizador para esto existe un parámetro llamado ***"voice",*** para poder elegir el idioma basta con escribir en la consola de instrucciones la orden: ***"espeak --voices"*** con lo que se desplegará un listado (Ilustración 4.1) con todos los idiomas disponibles de este programa y su método de invocación, así para el caso de esta propuesta se seleccionó el idioma español ***"es"*** pero se tiene la opción de elegir el idioma que se requiera.

```
$ espeak --voices
Pty Language Age/Gender VoiceName          File          Other Languages
 5  af             M  afrikaans           other/af
 5  an             M  aragonese           europe/an
 5  bg             -  bulgarian           europe/bg
 5  bs             M  bosnian             europe/bs
 5  ca             M  catalan             europe/ca
 5  cs             M  czech               europe/cs
 5  cy             M  welsh               europe/cy
 5  da             M  danish              europe/da
 5  de             M  german              de
 5  el             M  greek               europe/el
 5  en             M  default             default
 2  en-gb          M  english             en            (en-uk 2)(en 2)
 5  en-sc          M  en-scottish         other/en-sc   (en 4)
 5  en-uk-north    M  english-north       other/en-n    (en-uk 3)(en 5)
 5  en-uk-rp       M  english_rp          other/en-rp   (en-uk 4)(en 5)
 5  en-uk-wmids    M  english_wmids       other/en-wm   (en-uk 9)(en 9)
 2  en-us          M  english-us          en-us         (en-r 5)(en 3)
 5  en-wi          M  en-westindies       other/en-wi   (en-uk 4)(en 10)
 5  eo             M  esperanto           other/eo
 5  es             M  spanish             europe/es
 5  es-la          M  spanish-latin-am    es-la         (es-mx 6)(es 6)
 5  et             -  estonian            europe/et
 5  fa             -  Farsi               asia/fa
 5  fa-pin         -  Farsi-Pinglish      asia/fa-pin
 5  fi             M  finnish             europe/fi
 5  fr-be          M  french-Belgium      europe/fr-be  (fr 8)
 5  fr-fr          M  french              fr            (fr 5)
 5  ga             -  irish-gaeilge       europe/ga
 5  grc            M  greek-ancient       other/grc
 5  hi             M  hindi               asia/hi
 5  hr             M  croatian            europe/hr     (hbs 5)
 5  hu             M  hungarian           europe/hu
 5  hy             M  armenian            asia/hy
 5  hy-west        M  armenian-west       asia/hy-west  (hy 8)
```

Ilustración 4.1 Listado de idiomas disponibles del programa eSpeak[31]

Para establecer la configuración del parámetro del idioma en la consola de instrucciones se sebe hacer mediante el parámetro ***"-v"*** seguido del identificador del idioma que se desea elegir, en este caso, el idioma español por tanto los parámetros quedan de la siguiente manera:

```
1. espeak -v es "Texto a voz"
```

Con lo que el texto que se desea sintetizar sonara en el idioma español, sin embargo esta configuración no basto para hacer que el sintetizador se escuchara de manera clara por lo que se hicieron más pruebas y en la documentación del desarrollador

[31] Recuperada de: Autoría propia G. Belli [Marzo, 2017]

del programa[32] se encontró que existen variantes de voces ya predefinidas a las que se le combinan distintos parámetros como el "tono", el "eco", la "frecuencia", "la entonación", todos estos parámetros son configurables individualmente, para esta aplicación se probaron las variantes propuestas por el desarrollador las cuales son: f1, f2, f3, f4, m1, m2, m3, m4, las variantes f tienen timbre de voces femeninas, y varían en lo grave del sonido; las variantes m se escuchan como voces masculinas, para lo que la variante m1, tiene una configuración de parámetros que sintetizan el texto con un timbre de voz masculino agudo, y m4 con un timbre masculino grave. Después de realizar las pruebas con cada uno, se identificó que la mejor síntesis de voz fue la variante m1; por lo cual, esta fue la que se seleccionó. Este parámetro se establece agregando el símbolo ***"+"*** a la función, seguido del nombre de la variante, justo después del parámetro del idioma, y la invocación queda de la siguiente manera:

```
1. espeak -v es+m1 "Texto a voz"
```

Con esto, el texto que se desee sintetizar se podrá escuchar en el idioma español con una voz masculina, esto hace que el sintetizador de voz se escuche más claramente, pero en las pruebas también se determinó que la voz era muy pausada; es decir, muy lenta entre la reproducción de una palabra y la siguiente, por lo que se determinó realizar la configuración de un parámetro adicional.

El parámetro que configura la velocidad de reproducción es ***"speed"***, que por defecto tiene establecido un valor del 100%, y puede ser aumentado o disminuido mediante la instrucción ***"-s"***, seguido del porcentaje al que se desea configurar la velocidad de la reproducción, esto se colocará después del parámetro del idioma, quedando la invocación completa de la siguiente manera:

```
1. espeak -v es+m1 -s130 "Texto a voz"
```

Dónde:

espeak → es la instrucción que invoca al programa que hace la síntesis de voz.

[32] En este link puede encontrarse la documentación de los parámetros configurables del programa eSpeak http://espeak.sourceforge.net/docindex.html

-v es+m1 → es el parámetro que modifica el lenguaje de voz al cual se le configura un idioma en español con una voz masculina con la variación del timbre de voz poco grave.

-s130 → es el parámetro que modifica la velocidad, 100% es el valor por defecto, y en este caso se hizo un aumento del 30%, por lo que se definió el valor 130.

"Texto a voz" → este parámetro corresponde a la cadena de texto que se sintetizo en voz.

Al realizar las configuraciones antes mencionadas, se determinó que la voz pasa de ser poco inteligible a tener mayor claridad; con lo cual, se eligió esa configuración; sin embargo, para el caso de pacientes femeninas, existe la opción de sintetizar con voz de mujer.

4.3. Pruebas de salida de Audio.

Para las pruebas de la salida de audio se tienen dos opciones: configurar la salida de audio a través del plug de audio de 3.5mm, al cual se le pueden conectar unos audífonos o unas bocinas, o el audio puede salir a través del cable HDMI y sonar a través de las bocinas de la pantalla que se tenga conectada, siempre y cuando la pantalla cuente con estas. Por defecto la salida de audio se encuentra configurada a través del cable HDMI, pero puede forzarse la salida a través del plug 3.5mm aún con este cable conectado, de esta manera podría tenerse el video en una pantalla sin bocinas, un monitor, por ejemplo, y la salida de audio a través del plug con unos audífonos o unas bocinas extra.

Para realizar el cambio entre una u otra configuración basta con ejecutar una línea de instrucción en la consola de Ubuntu MATE.

La línea de instrucción que permite configurar la salida de audio a través del plug de 3.5mm es la siguiente:

```
1. sudo amixer cset numid=3 1
```

Al ejecutar esta instrucción y sin necesidad de reiniciar la Raspberry Pi el audio comienza a salir a través del plug 3.5.

Si lo que se desea es volver a configurar la salida de audio a través del cable HDMI la línea de instrucción que deberá ejecutarse es esta:

```
1. sudo amixer cset numid=3 2
```

Y con eso se podrá configurar entre una u otra la salida del audio.

4.4. Pruebas de escritura.

Ya con el sistema funcionando, se realizaron pruebas con distintos compañeros de la universidad y familiares, en la mayoría de los casos, hacer uso del sistema era un poco complicado, porque no lograban atinar a detener el cursor a tiempo, y seleccionar la letra correcta. Pero no tardaban mucho en adecuarse a la velocidad del cursor, y comenzar a seleccionar las letras correctas para formar las frases deseadas.

Se hicieron pruebas con el sensor Touch y respondió de forma correcta, este sensor podría funcionar con pacientes que pueden mover solo un dedo o la mano, (Ilustración 4.2); sin embargo, el sistema está diseñado a modo de que pueda ser empleado otro tipo de sensor, remplazando el Touch de manera sencilla para adecuarlo a las condiciones particulares del paciente, de igual manera al reproducir el sonido ya con la configuración de los parámetros encontrada, entendían de manera correcta lo que se escribía en la interfaz.

Ilustración 4.2 Prueba del sistema de comunicación.[33]

[33] Autoría propia G. Belli [Diciembre, 2016]

Conclusiones y Trabajos Futuros.

Se realizó una indagación acerca de la Esclerosis Lateral Amiotrófica (ELA), sus síntomas, etapas de la enfermedad y condiciones de los enfermos, con lo que se tuvo conocimiento de que la enfermedad es progresiva y degenerativa, y que los pacientes pueden sentir normalmente, aunque no puedan moverse, por lo que deben estar en todo momento comunicados para de alguna manera expresar sus sensaciones y ser atendidos oportunamente.

Se identificaron los métodos y técnicas con las que un enfermo de ELA se comunica actualmente, encontrando que es a través de cuadros con letras colocadas estratégicamente, que requieren de un intérprete para su uso; se encontró también, que hay métodos más automatizados como sensores de retina, los cuales permiten detectar la posición de los ojos, y controlar sistemas de cómputo con la vista, pero tienen un elevado costo.

Se definió un sistema de cómputo sobre el cual desarrollar la propuesta, se identificaron algunas microcomputadoras en el mercado, y se eligió la Raspberry Pi, por ser una microcomputadora con la capacidad de proceso requerida y a bajo costo.

Esta microcomputadora tiene la particularidad de poder ejecutar distintos sistemas operativos, por lo cual se tuvo que realizar investigación sobre las diversas opciones; para este caso se eligió Ubuntu MATE como sistema operativo, ya que al ser una distribución de Linux sus requerimientos de hardware son mínimos, además del beneficio de que al ser un sistema operativo libre no se deben pagar licencias por su uso ni por los programas que se ejecutan sobre él.

Se desarrolló una interfaz gráfica con Tkinter, un módulo de programación para la creación de interfaces, que además viene incluido con el sistema operativo; se realizó una configuración del alfabeto que permitió la fácil ubicación de las letras y agilizar la escritura.

Se seleccionó un método de entrada que permitió a la microcomputadora interpretarlo como una interrupción por parte del paciente, y se determinó que se

debía hacer uso de un sensor muy general, ya que cada paciente se encuentra en condiciones distintas, así que se decidió usar el sensor Touch para la interacción del sistema con el usuario, pero no se limita únicamente de este sensor, con lo que se podría remplazar por un sensor adecuado a las necesidades de cada paciente.

Se buscó un método para la síntesis de voz, y se encontró eSpeak que esta inegrado al sistema operativo y no tiene costo por uso, y que cuenta con parámetros configurables para mejorar la inteligibilidad de la comunicación; se conectaron las bocinas al puerto incluido en la Raspberry Pi. Se configuró con parámetros para modificar las características de la voz sintética.

Se realizó un algoritmo computacional que uniera cada una de las partes del sistema de comunicación, anexando la función para invocar mediante la interfaz al sintetizador de voz nativo de Linux; permitiendo, mediante el sensor Touch, seleccionar las letras y escucharlas a través de las bocinas.

Con todo ello, se obtuvo un sistema de comunicación que le permitirá a los pacientes con ELA o incluso a pacientes con otro tipo de discapacidad que sean mudos y tengan una escasa movilidad, comunicarse sin necesidad de un intérprete, brindándoles la posibilidad de comunicarse con mayor facilidad, y todo con elementos de bajo costo.

Con lo antes mencionado, se puede decir que se cumple con lo establecido en el objetivo general de este trabajo, quedando con algunas características de funcionamiento como trabajo futuro, como lo son: la optimización de la interfaz gráfica, un reordenamiento del alfabeto, considerando las letras más usadas dependiendo del idioma a utilizar, implementar texto predictivo en el sistema de comunicación para evitar la escritura de todas las frases; desarrollo de la función para variar la velocidad del cursor y agilizar la comunicación, así como el desarrollo de pruebas con personas con ELA, y explorar la posibilidad de utilizar este sistema para la navegación web, ya que la microcomputadora tiene la posibilidad de conectarse a Internet, lo cual permitiría al paciente navegar en la red, e incluso aumentar el alcance de este sistema permitiendo al paciente controlar el entorno mediante aplicaciones domóticas, ya que la microcomputadora lo permite al contar con los GPIO, que son pines de propósito general y pueden recibir señales de sensores y enviar señales hacia actuadores que sean controladas por el paciente.

Bibliografía.

ALEGSA. (12 de 06 de 2017). *Alegsa*. Obtenido de http://www.alegsa.com.ar/Dic/joystick.php

Alvarez, M. A. (15 de 09 de 2017). *Desarrolloweb*. Obtenido de https://desarrolloweb.com/articulos/1325.php

Arrazola, J. M., & P. P. (15 de 01 de 2017). 'Iriscom', una manera de comunicarse a través del ojo.

beagleboard.org. (14 de 12 de 2016). Obtenido de https://beagleboard.org/black

Coley, G. (14 de 12 de 2016). *beagleboard.org*. Obtenido de https://www.icbanq.com/data/ICBShop/board/Biggle%20Bone%20Black%20DataSheet.pdf

CSTR. (30 de 07 de 2017). *Centre for Speech Technology Research*. Obtenido de http://www.cstr.ed.ac.uk/projects/festival/

DOUTEL, F. (16 de 12 de 2016). *xataka*. Obtenido de https://www.xatakahome.com/trucos-y-bricolaje-smart/probamos-la-nueva-raspberry-pi-2-a-fondo

Drouiz. (17 de 08 de 2017). *Drouiz*. Obtenido de https://www.drouiz.com/blog/2016/03/14/sensor-touch-arduino/

Ecu Red. (22 de 08 de 2017). *Ecu Red*. Obtenido de https://www.ecured.cu/PyQt

EcuRed. (10 de 01 de 2017). Obtenido de https://www.ecured.cu/Lenguaje_de_Programaci%C3%B3n_C

eLinux. (13 de 12 de 2016). Obtenido de http://elinux.org/Beagleboard:Main_Page

eSpeak. (29 de 08 de 2017). *eSpeak*. Obtenido de http://espeak.sourceforge.net/

Garcia, R. (13 de 12 de 2016). *Esclerosis Lateral Amiotrófica - Compendio de Información*. Obtenido de https://sites.google.com/site/vicortega2/etran

IRISBOND. (02 de 12 de 2016). Obtenido de http://www.irisbond.com/productos/irisbond-primma

Lenguajes de Programación. (11 de 01 de 2017). Obtenido de http://www.lenguajes-de-programacion.com/programacion-java.shtml

Lerones, M. F., & Rodriguéz, A. D. (06 de 2010). *Esclerosis lateral amiotrófica.* Obtenido de Medicina General y de Familia: http://www.mgyf.org/medicinageneral/revista_130/pdf/568-570.pdf

LinkSprite. (16 de 12 de 2016). Obtenido de http://linksprite.com/wiki/index.php5?title=Products-description

Microsoft. (26 de 12 de 2016). Obtenido de https://developer.microsoft.com/en-us/windows/iot/Explore/IoTCore

Milenio. (31 de 05 de 2017). *Milenio.* Obtenido de http://www.milenio.com/salud/dia_mundial_esclerosis_multiple-enfermedad-tratamiento-sintomas-milenio-noticias_0_966503590.html

Pastor, J. (26 de 12 de 2016). *xataka*.

Puig, C. (09 de 09 de 2017). *Milenio.* Obtenido de http://www.milenio.com/salud/esclerosis_multiple-enfermedad-cerebro-cuerpo-mielina-sintomas-milenio-noticias_0_1027097380.html

pythondiario. (16 de 07 de 2017). *Pythondiario*. Obtenido de http://www.pythondiario.com/2016/11/los-5-mejores-ide-para-python.html

RASPBERRY PI FOUNDATION. (16 de 12 de 2016). Obtenido de https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

Raspbian. (7 de 01 de 2017). Obtenido de https://www.raspbian.org/FrontPage

Revill, S. (4 de 01 de 2017). *RISC OS OPEN*. Obtenido de https://www.riscosopen.org/news/articles/2012/11/05/official-release-of-risc-os-for-raspberry-pi-updated

Rivera, S. (29 de 09 de 2017). *Xeo.* Obtenido de http://www.xeu.com.mx/nota.cfm?id=927377

Ruiz, O. (13 de 12 de 2016). *Esclerosis Lateral Amiotrófica - Compendio de Información*. Obtenido de https://sites.google.com/site/vicortega2/etran

tdrobotica. (14 de 08 de 2017). *tdrobotica*. Obtenido de http://tdrobotica.co/sensor-de-musculo-mioelectrico/397.html

theom. (4 de 01 de 2017). *raspberrypi.org*. Obtenido de https://www.raspberrypi.org/forums/viewtopic.php?f=55&t=22093

Ubumedia. (16 de 08 de 2017). *Ubumedia*. Obtenido de https://ubumedia.wordpress.com/2009/08/31/gespeaker-texto-a-voz/

Ubuntu MATE. (10 de 01 de 2017). Obtenido de https://ubuntu-mate.org/

Upton, E., & G. H. (2014). *Raspberry Pi User Guide.* John Wiley & Sons.

Índice de ilustraciones.

I. Anexo A.

I.1 Preparando el sistema operativo.

Para el desarrollo y la implementación de la propuesta de sistema de comunicación mencionado en este trabajo, se utiliza como elemento base la microcomputadora Raspberry Pi 2.

Para poder iniciar la microcomputadora se requiere de una fuente de alimentación que proporcione 5V y 2A con un puerto microUSB, un teclado con conexión USB, un mouse con conexión USB, un cable HDMI, una pantalla que cuente con entrada de video por HDMI, una tarjeta de memoria microSD de mínimo 8GB de capacidad y una computadora con sistema operativo Windows.

Como primer paso, se debe adquirir el archivo imagen del sistema operativo con el que se arrancará la Raspberry Pi 2, en este caso Ubuntu Mate,[34] se debe tener cuidado en seleccionar la versión para Raspberry Pi, que es una versión del sistema operativo diseñada para procesadores ARM con arquitectura RISC (Reduced Instruction Set Computer, Computadora con Conjunto Reducido de Instrucciones), como el que tiene la microcomputadora Raspberry Pi 2 (Ilustración I.1), y no las versiones de escritorio que están diseñadas para ser instaladas en computadoras con más recursos.

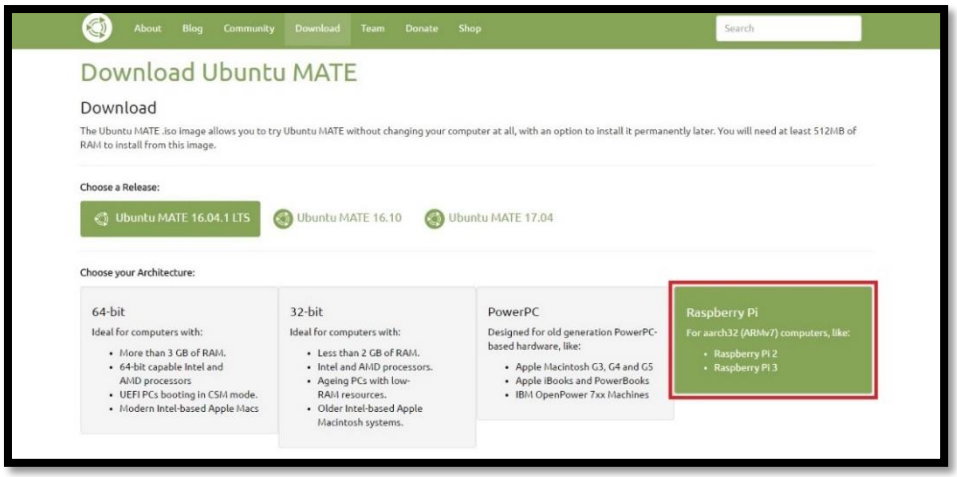


Ilustración I.1 Captura de pantalla del sitio web del desarrollador de Ubuntu MATE que señala la versión de sistema operativo que se debe descargar.35

[34] El archivo imagen se encuentra en el sitio web del desarrollador https://ubuntu-mate.org/download/

[35] Recuperada de: https://ubuntu-mate.org/download/ [Diciembre, 2016]

El archivo imagen se puede descargar a través de los servidores directos del sitio web del fabricante o utilizando un gestor de descargas como lo es “Torrent” y el tiempo de descarga va a depender de la velocidad de conexión que se tenga, se recomienda hacer uso de Torrent, ya que, durante el desarrollo de esta propuesta al descargar desde los servidores directos, la descarga tomaba más tiempo e incluso se interrumpía.

Una vez se tenga el archivo imagen, este se debe descomprimir y para esto el desarrollador del sistema operativo recomienda “7-Zip”[36] que es un programa que servirá para realizar la descompresión, se debe tener cuidado con la versión que se descarga de este programa ya que existe la versión de 32 y 64 bits, que dependen de la arquitectura de tu sistema operativo, pero en la página web del desarrollador se ofrecen ambas versiones, en cuanto se tenga la versión correcta, que es muy ligera, se procede a la instalación, para esto basta con encontrar el archivo descargado y hacer doble clic en él, con esto se abrirá el asistente de instalación (Ilustración I.2).

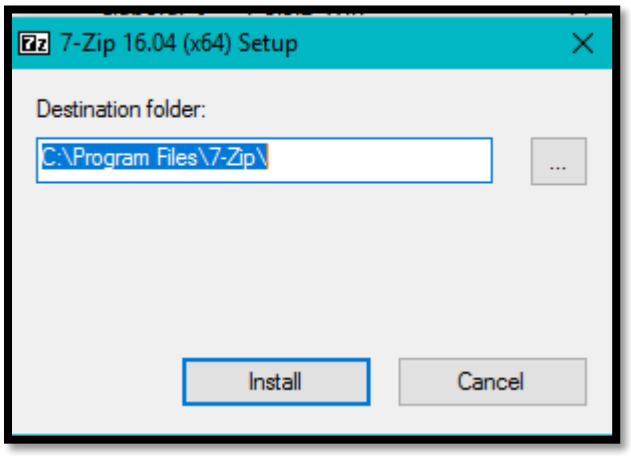


Ilustración I.2 Asistente de instalación 7-Zip.[37]

El asistente indica el directorio donde va a instalarse el programa, se puede modificar, aunque se recomienda que se deje tal cual esta y se presione el botón “Install”, el programa comenzará a instalarse y una vez haya concluido pedirá que

[36] Este programa puede encontrarse en el sitio web del fabricante http://www.7-zip.org/

[37] Recuperada de: 7-Zip [Diciembre, 2016]

se reinicie la computadora para agregar todas las características de este al sistema operativo, cosa que se recomienda hacer.

Ya con el 7-Zip instalado se puede descomprimir el archivo del sistema operativo Ubuntu MATE, para esto basta con abrir 7-zip, encontrar la ruta donde se encuentra guardado el comprimido de Ubuntu MATE y hacer clic en el botón “Extraer” como se muestra en la Ilustración I.3.

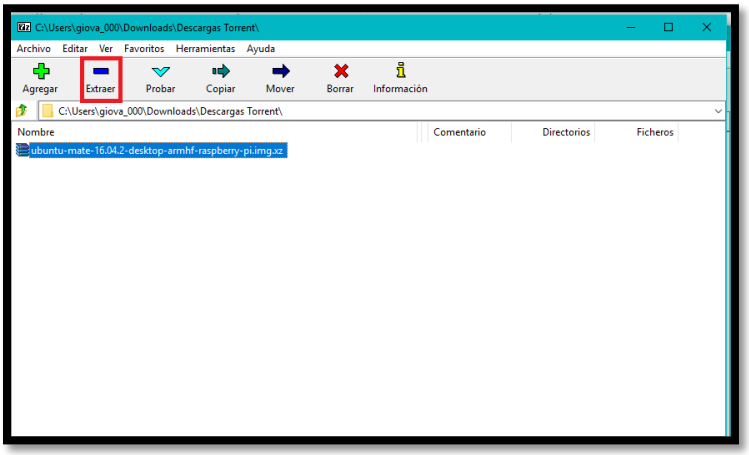


Ilustración I.3 Selección del archivo a descomprimir.[38]

Al realizar lo anterior se mostrará una nueva ventana de 7-Zip donde se muestran las opciones de descompresión, aquí debe indicarse la ruta donde el archivo será descomprimido, si se desea se puede establecer una nueva carpeta, si no, basta con hacer clic en el botón “Aceptar” con lo que se iniciará la descompresión y el nuevo archivo se guardará en el directorio del archivo comprimido, dentro de una carpeta con el mismo nombre (Ilustración I.4)

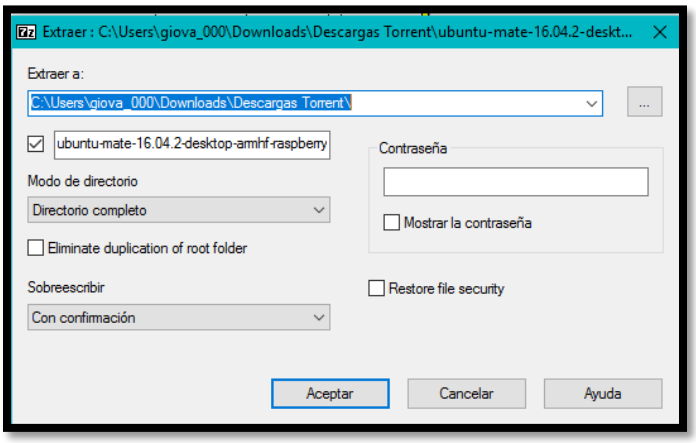


Ilustración I.4 Opciones de descompresión.[39]

[38] Recuperada de: 7-Zip [Diciembre, 2016]
[39] Recuperada de 7-Zip [Diciembre, 2016]

Con el archivo imagen ya descomprimido (debe quedar un archivo con extensión .img), se procederá a dar formato a la memoria microSD para después instalar el sistema operativo en ella, para hacer el formateo correspondiente el desarrollador de Ubuntu MATE recomienda el programa "SD Formatter".[40]

Al descargar el SD Formatter de la página oficial del fabricante, este se encontrará también comprimido, así que se usará nuevamente 7-Zip para poder extraerlo de la misma manera que con el archivo anterior.

Instalarlo es sencillo ya que una vez extraído el archivo resta solamente hacer doble clic sobre él y se ejecutará el asistente de instalación (Ilustración I.5).

Ilustración I.5 Asistente de instalación SD Formatter41.

En la primera pantalla del asistente de instalación se da la bienvenida y una advertencia de derechos de autor, se debe hacer clic en el botón "Next" y aparecerá la siguiente pantalla del asistente de instalación, en la cual se muestra la ruta donde será instalado el programa, se recomienda dejar todo como esta, pero si se desea cambiar se debe hacer clic en el botón "Change", si no, se debe continuar a la siguiente ventana del asistente haciendo clic en el botón "Next" (Ilustración I.6).

Al hacer esto aparecerá una nueva ventana la cual nos pide confirmar la instalación y a la que se debe hacer clic en el botón "Install" e inmediatamente comenzará la instalación del programa.

[40] Este programa se puede encontrar en el sitio web del fabricante: https://www.sdcard.org/downloads/formatter_4/eula_windows/index.html
[41] Recuperada de SD Formatter [Diciembre, 2016]

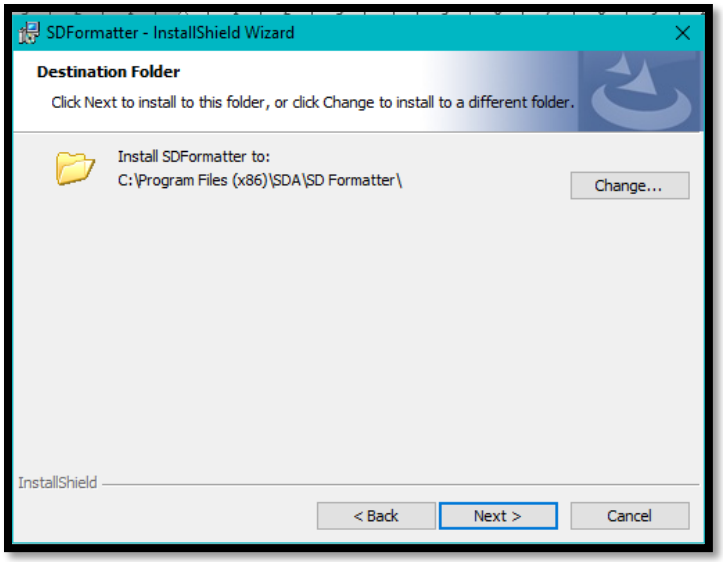


Ilustración I.6 Selección del directorio de instalación de SD Formatter.[42]

Finalmente, con el SD Formatter ya instalado daremos formato a la memoria microSD, para esto se debe abrir el SD Formatter desde el acceso directo que la instalación creará en el escritorio, o desde la lista de aplicaciones de Windows, al abrirse el programa se mostrará la interfaz, que es bastante sencilla pero muy útil, y una vez la interfaz se encuentre abierta debe insertarse la memoria microSD a la computadora, la interfaz detectará automáticamente la memoria insertada, si no fuera así se debe seleccionar la memoria y estar seguro que es la correcta en la opción “Drive” de la interfaz (Ilustración I.7), ya que este proceso borrará toda información de la tarjeta de memoria, se le puede asignar una nuevo nombre a la memoria en la casilla “Volumen Label” si así lo desea y se hace clic en el botón “Format” y aparecerá una advertencia que nos indica que no se debe retirar la memoria durante el proceso y que si se está seguro de realizar el formateo, a lo cual se confirma haciendo clic en el botón “Aceptar” (Ilustración I.8), con esto aparecerá una ventana mostrando el proceso de avance del formateo y finalmente una venta que indica las características de la memoria ya formateada (Ilustración I.9), se hace clic en el botón “Aceptar” y con esto la memoria ha quedado lista para instalar el sistema operativo en ella.

[42] Recuperada de SD Formatter [Diciembre,2016]

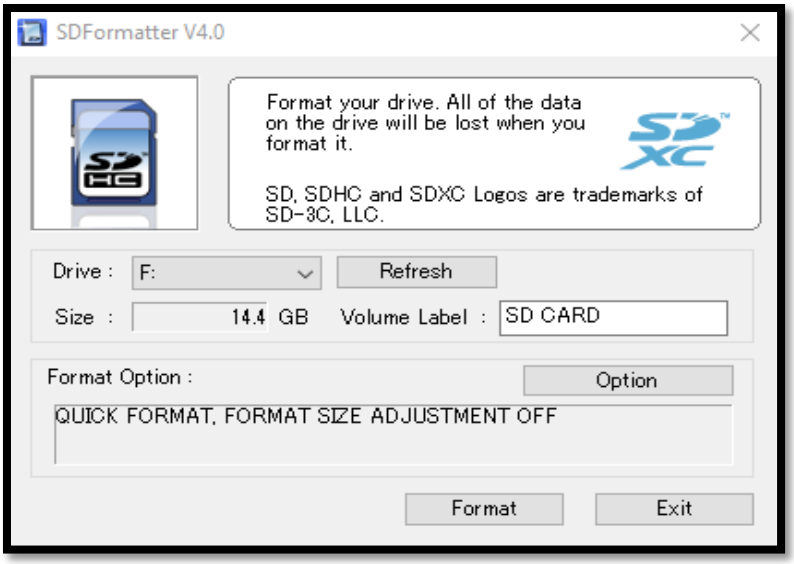


Ilustración I.7 Interfaz del programa SD Formatter.[43]

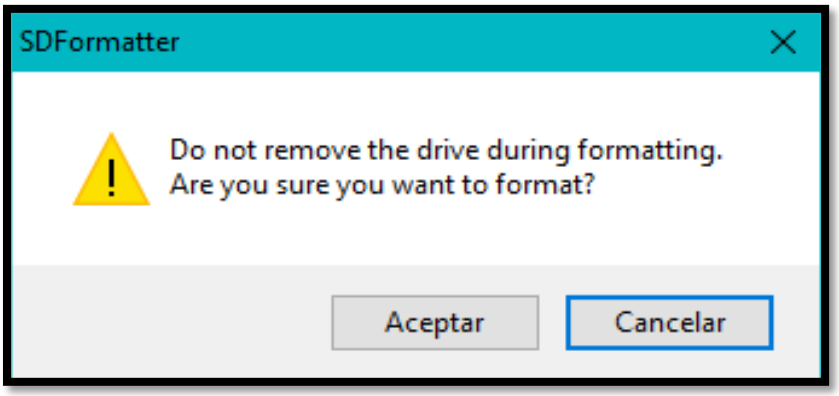


Ilustración I.8 Ventana de advertencia de formateo de SD Formatter.[44]

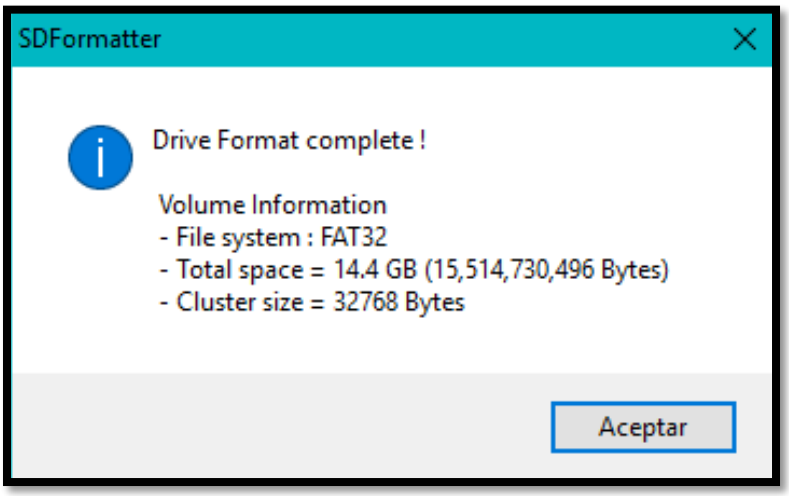


Ilustración I.9 Ventana final de SD Formatter que indica que el formateo se completó exitosamente.[45]

[43] Recuperada de: SD Formatter [Diciembre, 2016]
[44] Recuperada de: SD Formatter [Diciembre, 2016]
[45] Recuperada de: SD Formatter [Diciembre, 2016]

Para proceder a instalar el sistema operativo en la memoria microSD se hará uso de un último programa llamado “Win32 Disk Imager” [46] que servirá para instalar el archivo imagen en la memoria microSD y así tener listo el sistema operativo para iniciar la Raspberry Pi 2.

Para hacerse del programa solo hay que hacer clic en el botón “Download” señalado en la Ilustración I.10 en el sitio web oficial del desarrollador y con esto comenzará a descargarse de forma directa el programa en cuestión.

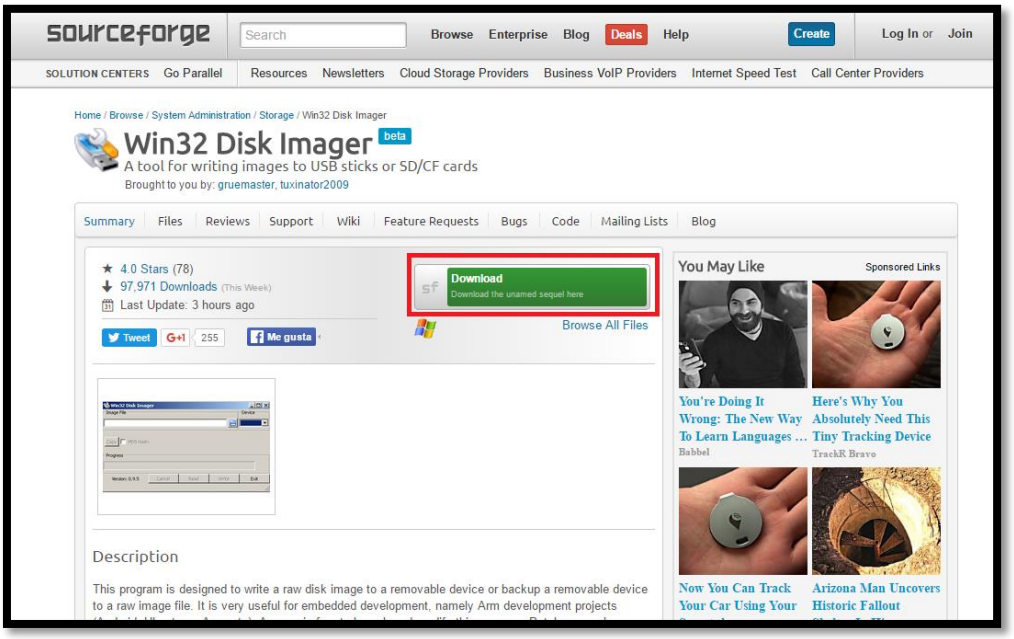


Ilustración I.10 Sitio web del desarrollador del programa Win32 Disk Imager. [47]

En cuanto se tenga ya el programa, se debe hacer doble clic sobre el archivo descargado con lo cual se abrirá el asistente de instalación, mostrando como primera ventana la bienvenida, así como la recomendación de cerrar todos los programas antes de continuar con la instalación (Ilustración I.11), se debe hacer clic en el botón “Next” , en la siguiente ventana se puede observar el acuerdo de licencia, y los derechos de autor, si se está de acuerdo se debe marcar la casilla “I accept the agreement” como se señala en la Ilustración I.12 y hacer clic nuevamente en el botón “Next”.

[46] Este programa puede encontrarse en el sitio web del desarrollador thttps://sourceforge.net/projects/win32diskimager/

[47] Ilustración recuperada de https://sourceforge.net/projects/win32diskimager/?source=typ_redirect

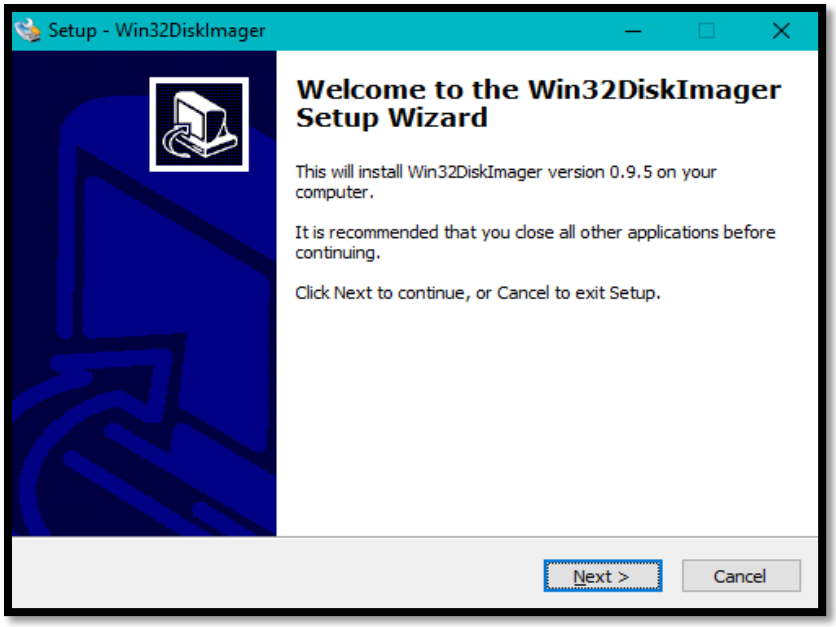


Ilustración I.11 Primera ventana de asistente de instalación de Win32 Disk Imager.[48]

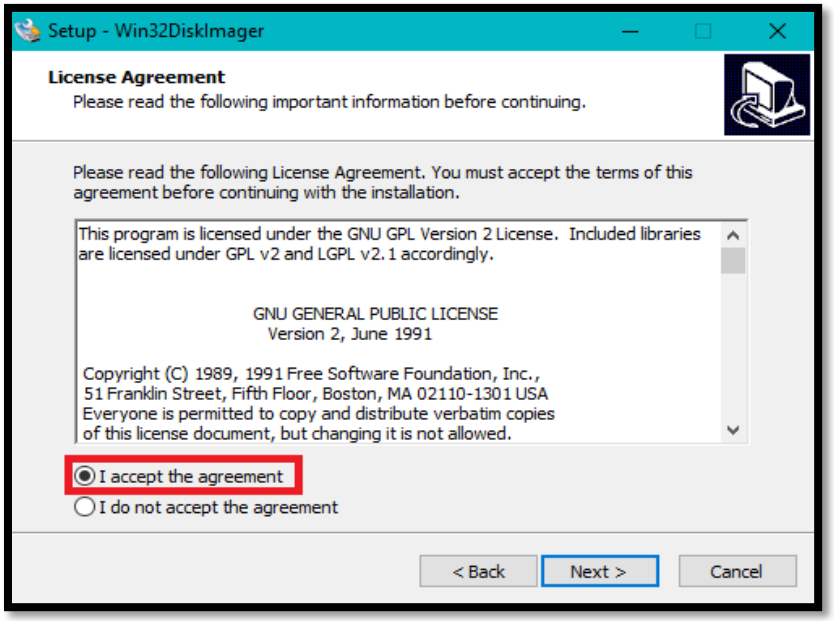


Ilustración I.12 Acuerdo de licencia y derechos de autor de Win32 Disk Imager.[49]

Después de presionar el botón “Next” la siguiente venta muestra la ruta donde será instalado el programa, se recomienda no modificar nada, pero si se desea se puede cambiar la ubicación de instalación haciendo clic en el botón “Browse” y seleccionando la nueva ruta (Ilustración I.13), si no basta con hacer clic en el botón “Next” para pasar a la siguiente ventana del asistente de instalación.

[48] Recuperada de Win 32 Disk Imager, [Diciembre, 2016]
[49] Recuperada de Win 32 Disk Imager, [Diciembre, 2016]

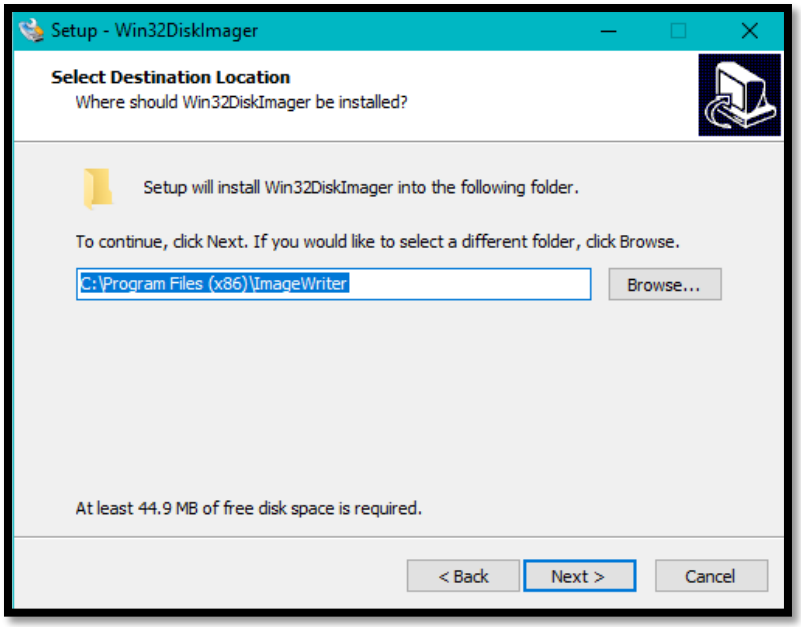


Ilustración I.13 Ventana que indica la ruta de instalación del programa.[50]

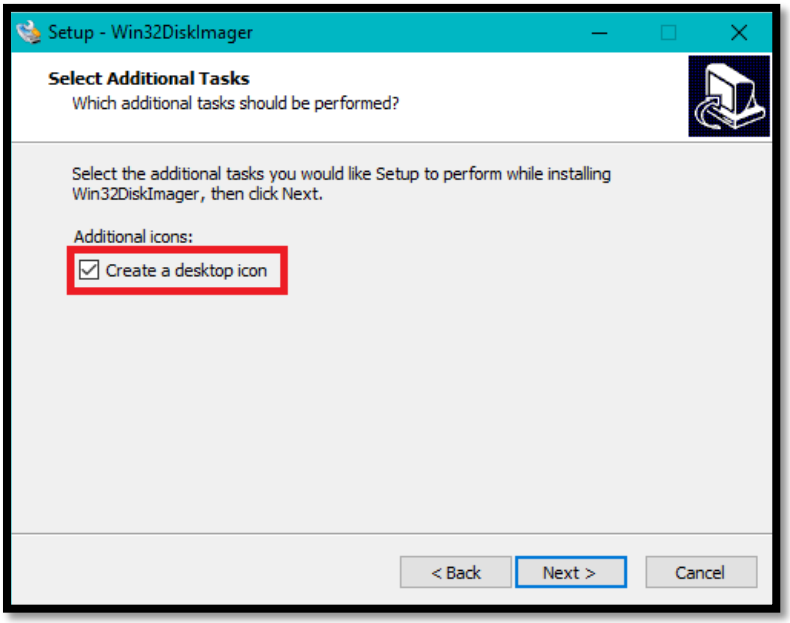


Ilustración I.14 Ventana que permite la creación de un icono de escritorio para el programa Win32 Disk Imager.[51]

En la siguiente ventana se presenta la opción de crear un icono en el escritorio de Windows esto con el fin de poder ejecutar el programa de forma más sencilla, cosa que se recomienda hacer y para lo cual solo basta con marcar la casilla “Create a desktop icon” indicada en la Ilustración I.14, después se debe presionar el botón “Next” y se abrirá la siguiente ventana del administrador que muestra un resumen de las configuraciones elegidas, así como la ruta de instalación y pregunta si esta

[50] Recuperada de Win 32 Disk Imager, [Diciembre, 2016]
[51] Recuperada de Win 32 Disk Imager, [Diciembre, 2016]

correcto y se desea continuar, a lo que se debe confirmar haciendo clic en el botón "Install" (Ilustración I.15) y posterior a esto se verá momentáneamente una ventana de progreso para después mostrar la última ventana del asistente de instalación (Ilustración I.16), la cual muestra la confirmación de instalación del programa a lo que basta con presionar el botón "Finish" para cerrar el asistente de instalación y hacer que se abra la interfaz del programa como acto seguido.

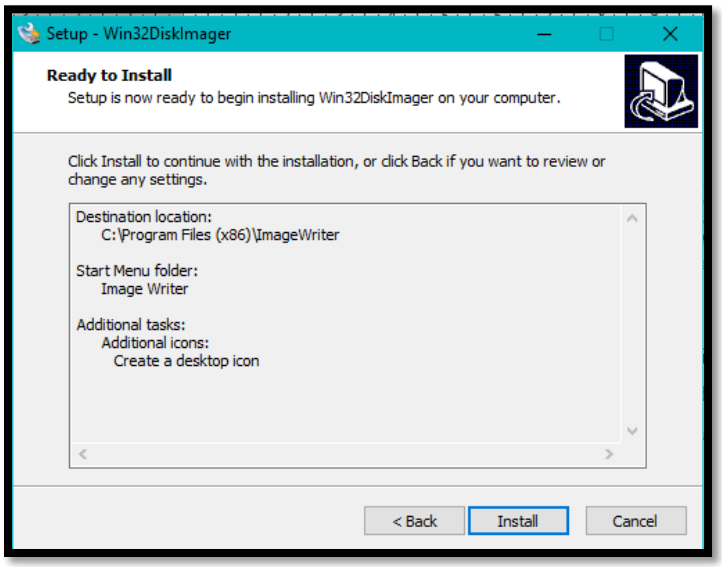


Ilustración I.15 Ventana de confirmación de instalación para el programa Win32 Disk Imager.[52]

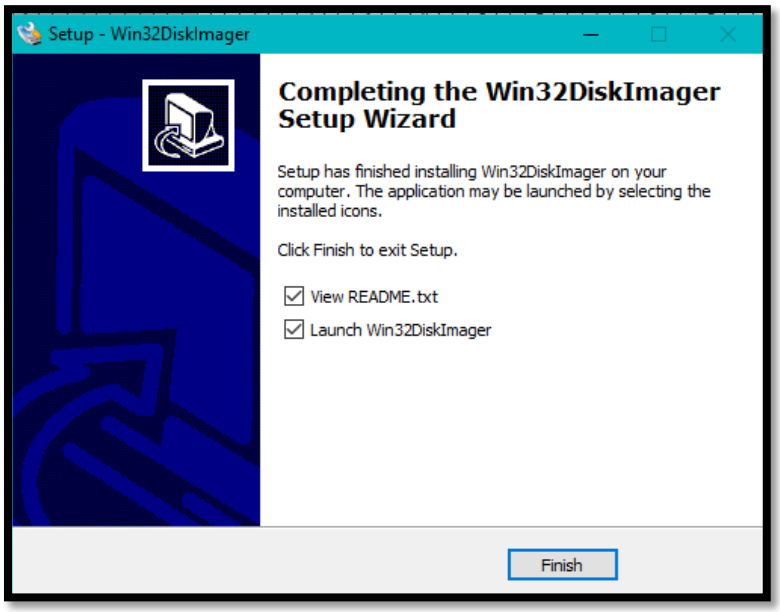


Ilustración I.16 Ventana de confirmación de instalación del programa Win32 Disk Imager.[53]

[52] Recuperada de Win 32 Disk Imager, [Diciembre, 2016]
[53] Recuperada de Win 32 Disk Imager, [Diciembre, 2016]

Ya en la interfaz del programa Win32 Disk Imager se debe seleccionar la ruta donde se encuentra el archivo imagen (con extensión .img), para esto basta con hacer clic en el icono de la carpeta señalado en la Ilustración I.17, y ubicar el archivo.

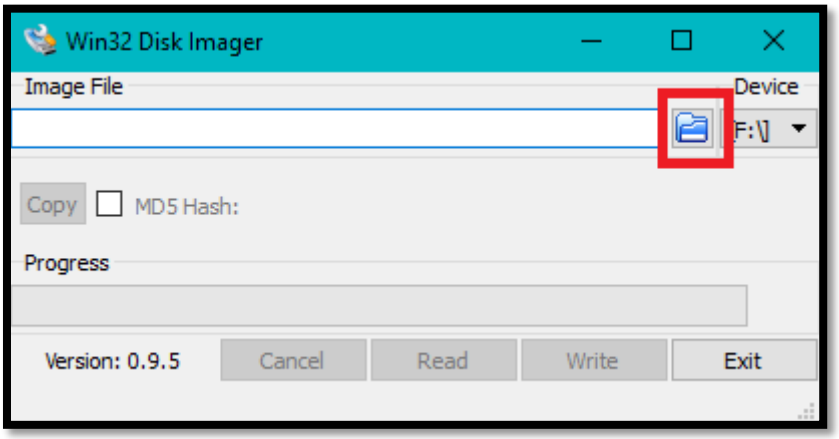


Ilustración I.17 Interfaz del programa Win32 Disk Imager.[54]

Antes de proceder a grabar el sistema operativo en la memoria se debe estar seguro que se esté seleccionando la memoria correcta, verificando que coincida el nombre del volumen de la tarjeta con el seleccionado en la interfaz del programa, para esto basta con cerciorarse que la letra que aparece en el apartado "Device" del interfaz señalado en la ilustración I.18 sea el que corresponde con la tarjeta de memoria.

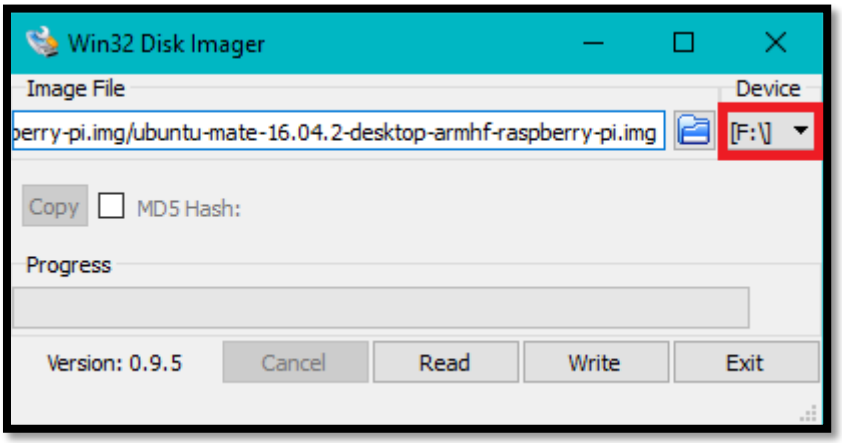


Ilustración I.18 Verificar que se selecciona la memoria correcta en la interfaz.[55]

Si todo es correcto se debe hacer clic en el botón "Write" inmediatamente saltara una ventana de confirmación a la que hay que hacer clic en "Yes" y la barra "Progress" de la interfaz comenzará a mostrar el estado de la escritura, en cuanto

[54] Recuperada de Win 32 Disk Imager, [Diciembre, 2016]
[55] Recuperada de Win 32 Disk Imager, [Diciembre, 2016]

termine aparecerá una ventana indicando que la escritura se completó y para terminar y cerrar el programa se debe presionar el botón “Exit”.

I.2 Iniciando la Raspberry Pi 2.

Ya con el sistema operativo en la memoria microSD se debe colocar dentro de la microcomputadora en el puerto de la memoria como se indica en la Ilustración I.19.

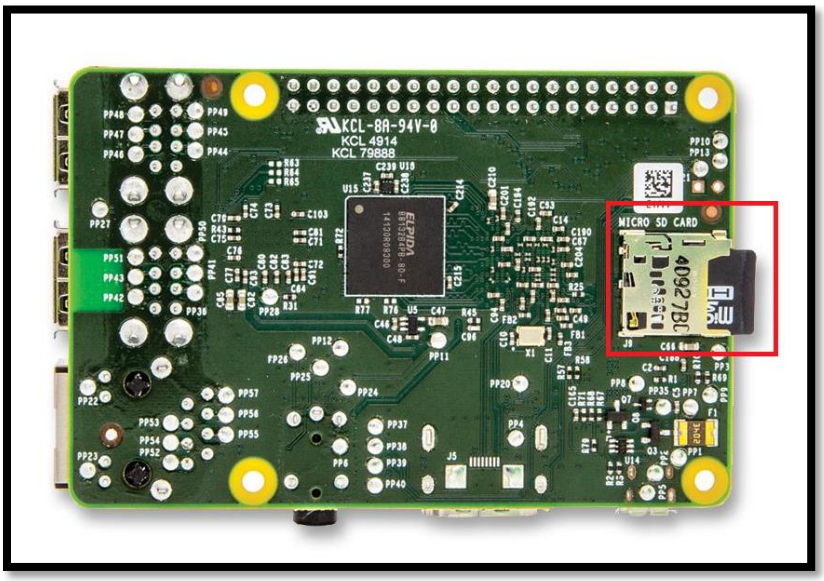


Ilustración I.19 Puerto donde se debe colocar la memoria en la Raspberry Pi 2. [56]

Después se debe colocar el mouse y el teclado en los puertos USB, también hará falta conectar el cable HDMI tanto a la pantalla como a la microcomputadora, así como el cable Ethernet tanto a la Raspberry Pi 2 como al modem para la conexión a internet y una vez todo esté conectado se debe conectar la fuente de alimentación como se muestra en la Ilustración I.20.

[56] Ilustración recuperada de: https://electronilab.co/tienda/raspberry-pi-2-modelo-b-armv7-1g-ram-8gb-microsd-noobs/

Ilustración I.20 Conexiones de la Raspberry Pi 2. [57]

Si todo se ha hecho de manera correcta hasta este momento, la Raspberry Pi 2 comenzará con el primer arranque y lo primero que se muestra en pantalla es el logotipo de Raspberry Pi seguido de lineas de codigo en fondo negro, como se puede apreciar en la ilustración I.21.

Ilustración I.21 Primera ventana de arranque inicial de Ubuntu MATE.[58]

Las lineas de texto solo se mostraran por un momento y en seguida se ejecutará el asistente de inicio, el cual mostrará en su primer ventana la opcion de configuración del idioma (Ilustración I.22) en la cual se debe elegir el idioma preferido y hacer clic en el boton “Continue”.

[57] Ilustración recuperada de: https://www.scirra.com/blog/ashley/23/how-to-get-webgl-on-the-raspberry-pi-2

[58] Ilustración recuperada de: https://www.youtube.com/watch?v=5hSIZf7CmIU

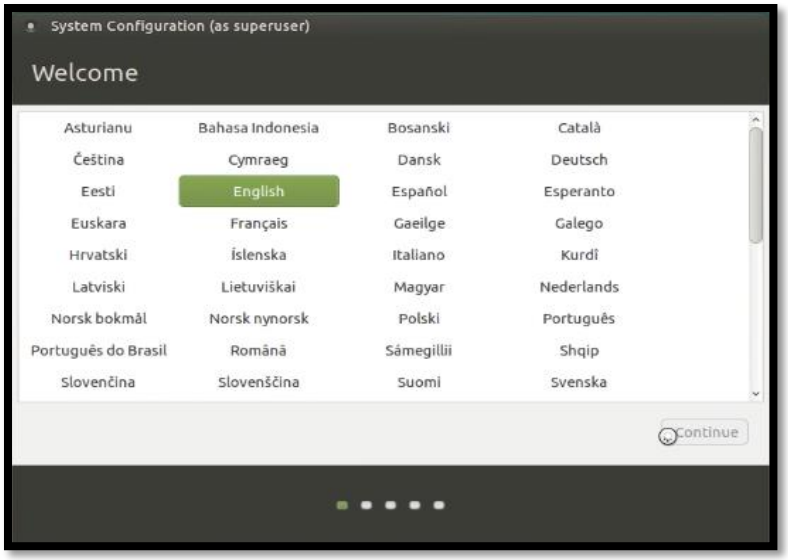


Ilustración I.22 Ventana de configuración de Idioma del Asistente de Inicio. [59]

En la siguiente ventana del asistente de inicio, se mostrará la configuración de región, para lo cual se puede escribir el nombre del país en la caja de texto o hacer clic sobre la ubicación en el mapa (Ilustración I.23) y hacer clic en el botón "Continue".

Ilustración I.23 Ventana de configuración de región del asistente de Inicio.[60]

La siguiente ventana en mostrar el asistente de inicio, será la de configuración del teclado, en la cual se podrá elegir la distribución del teclado que se halla colocado,

[59] Ilustración recuperada de: https://www.youtube.com/watch?v=5hSIZf7CmIU
[60] Ilustración recuperada de: https://www.youtube.com/watch?v=5hSIZf7CmIU

una vez elegido la configuración correcta para el teclado se debe hacer clic en el botón “Continue” del asistente (Ilustración I.24).

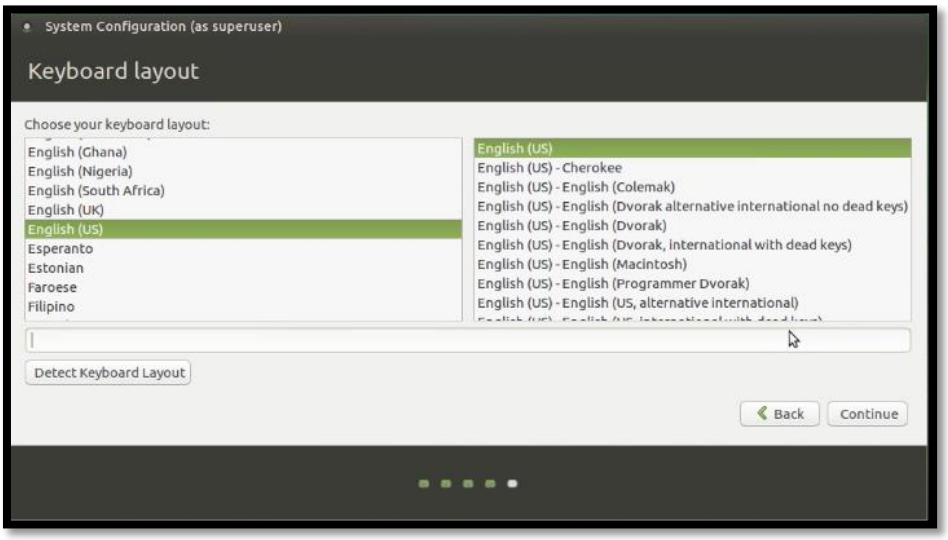


Ilustración I.24 Ventana de configuración del teclado del asistente de inicio. [61]

Enseguida se mostrará la ventana de configuración de usuario, en la que el asistente de inicio solicita que se agregue su nombre, un nombre de usuario, un nombre al equipo, así como una contraseña que servirá para la administración de los programas (Ilustración I.25), una vez se hayan colocado los datos solicitados, se debe hacer clic en el botón “Continue”.

Ilustración I.25 Ventana de configuración de usuario de asistente de inicio. [62]

Con esto se da por terminada la configuración inicial y el asistente se prepara para crear y mostrar el escritorio de Ubuntu MATE, mientras esto sucede, se mostrará

[61] Ilustración recuperada de: https://www.youtube.com/watch?v=5hSIZf7CmIU
[62] Ilustración recuperada de: https://www.youtube.com/watch?v=5hSIZf7CmIU

una ventana en la cual se muestra información sobre el sistema operativo, así como algunos consejos de uso, para finalmente tras un reinicio mostrar el escritorio y una ventana que da la bienvenida (Ilustración I.26).

Ilustración I.26 Escritorio Ubuntu MATE.[63]

I.3 Preparando Ubuntu MATE.

Una vez se tenga iniciado el sistema operativo sobre el cual se va a desarrollar la propuesta, es necesario instalar algunos programas adicionales, así como algunas bibliotecas, que nos permitirán desarrollar de manera correcta tanto la interfaz, como el programa que la ejecutará.

I.3.1 IDLE Python 3.5.

Para crear el programa de ejecución se utilizará el entorno de desarrollo IDLE Python 3.5, que ya viene incluido en el sistema operativo Ubuntu MATE y que se puede encontrar dentro de la pestaña “Aplicaciones”, en el submenú “Programando” (Ilustración I.27), es importante mencionar que se hará uso del entorno de programación para Python 3.5, ya que el sistema operativo tiene instalados dos entornos, uno que funciona con Python 2.7 y el otro con Python 3.5, durante el desarrollo de esta propuesta se hará uso de Python 3.X, esto porque

[63] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

algunas de las bibliotecas que se utilizarán solo se ejecutan sobre esta versión del lenguaje de programación, además de que los parámetros de programación cambian respecto a las versiones, así que para un óptimo funcionamiento de esta propuesta se debe estar seguro siempre que se trabaja sobre Python 3.X y no sobre Python 2.X.

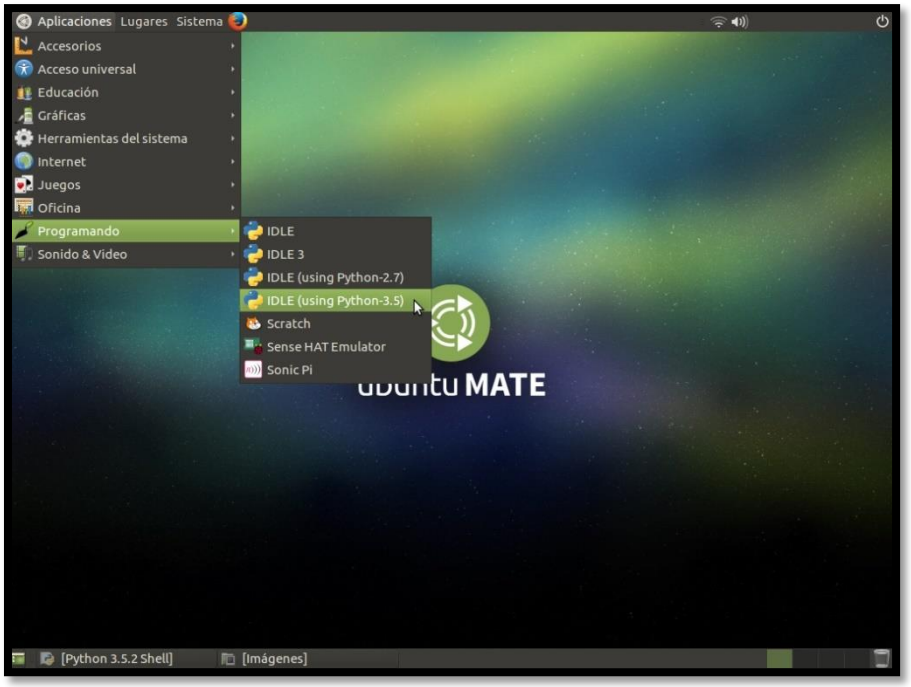


Ilustración I.27 Ubicación del IDLE para Python 3.5.[64]

Al abrir el programa se muestra la ventana del intérprete de Python, el cual muestra un cursor con el que se pueden ejecutar una por una las líneas de código y su resultado se mostrará en la línea inmediata, esta ventana mostrará los errores que se pueda generar durante la escritura del programa, y algunos resultados (Ilustración I.28).

[64] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

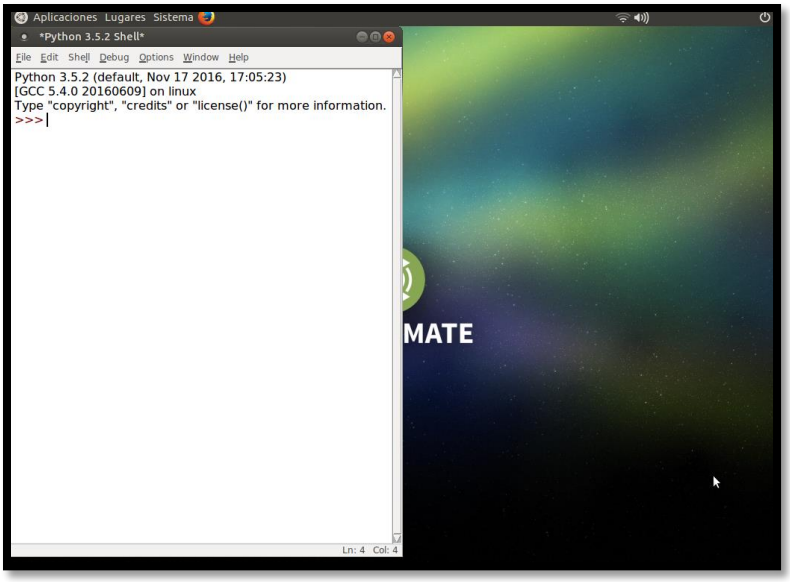


Ilustración I.28 Ventana del intérprete del IDLE Python 3.5.[65]

Para la creación del programa a ejecutar se usará un editor de texto, en este caso se recomienda utilizar el que ofrece el IDLE de Python, para hacer uso de él, se debe hacer clic en la pestaña “File” y seleccionar la opción “New File” como se muestra en la Ilustración I.29 o simplemente presionar la combinación de teclas Ctrl+N.

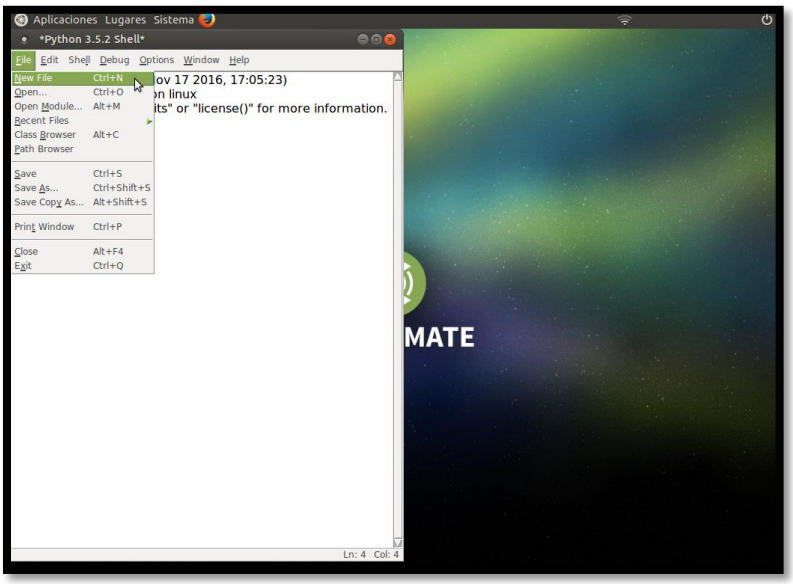


Ilustración I.29 Ruta para abrir editor de texto del IDLE.[66]

Con esto el editor de texto quedará disponible para comenzar a escribir el código del programa de un lado y del otro el intérprete interactivo del IDLE que mostrará

[65] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]
[66] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

los errores de programación si llegarán a suceder, así como los resultados de la ejecución (Ilustración I.30).

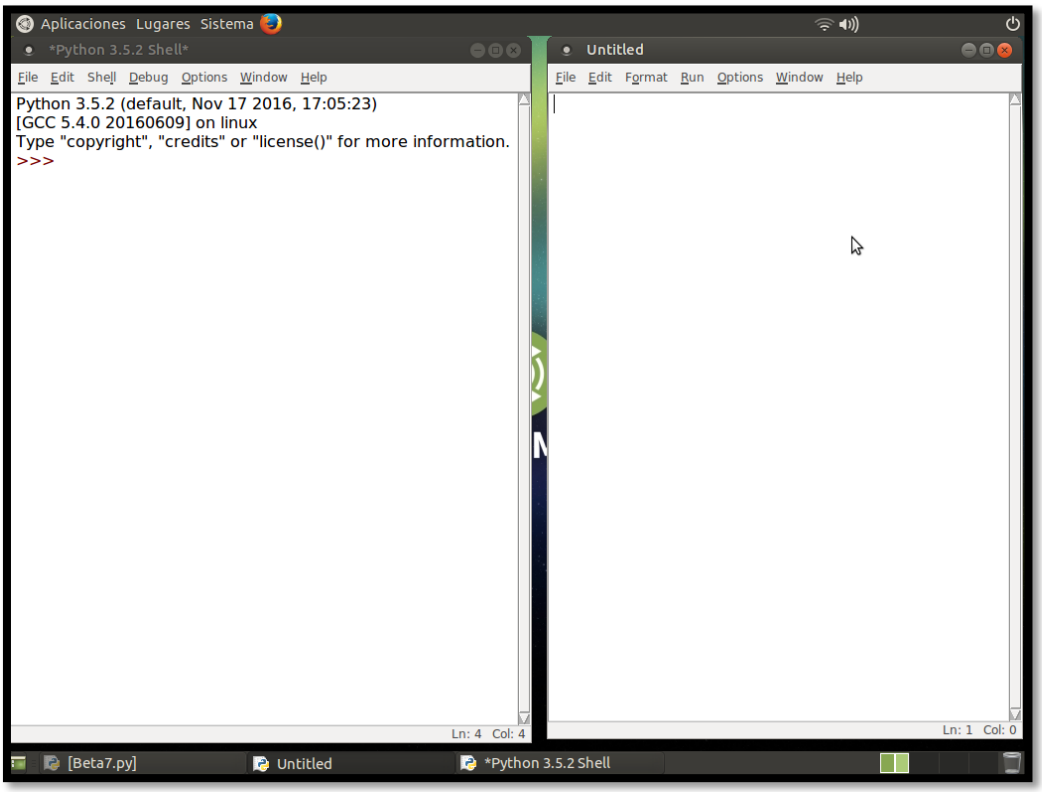


Ilustración I.30 Intérprete y editor de textos de IDLE para Python 3.5.[67]

I.3.2 Introducción a Tkinter.

La propuesta de este trabajo requiere de una interfaz gráfica que ayude al usuario a poder ver las letras del abecedario y poder seleccionar la deseada mediante el sensor de entrada y con esto ir creando las palabras o frases que el usuario desee.

Para realizar esta interfaz se hará uso de una biblioteca para la creación de interfaces gráficas que ofrece Python, Tkinter, que ya viene instalada junto a Python, aunque es muy sencilla, es suficiente para el propósito de esta propuesta.

La forma en la que se incluye la biblioteca a código del programa es con la sentencia “from tkinter import *” con lo que se le indica al entorno de programación que se incluyan todas las características que la librería Tkinter ofrece.

A modo de ejemplo se creará una ventana, para esto se debe colocar el texto del Código I.1 en el editor de textos del IDLE para Python 3.5 como se muestra en la Ilustración I.31.

[67] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

```
1  from tkinter import *
2
3  v0 = Tk()
4  v0.title("Sistema de comunicacion")
5  v0.geometry("600x280")
6
```

Código I.1 Creación de ventana ejemplo.

Donde:

- from tkinter import *: es la línea de código que invoca a librería tkinter a ser incluida en el programa.
- v0 = TK() : es la línea de código que indica que se creara una nueva ventana la cual tendrá por nombre “v0”.
- v0.title(“Sistema de comunicación”) : es la línea de código que indica que a la ventana v0 se le ponga como propiedad de título el texto “Sistema de comunicación.
- v0.geometry(“600x240”): es la línea de código que indica que la ventana debe tener un tamaño de 600 pixeles por ancho y 280 por alto.

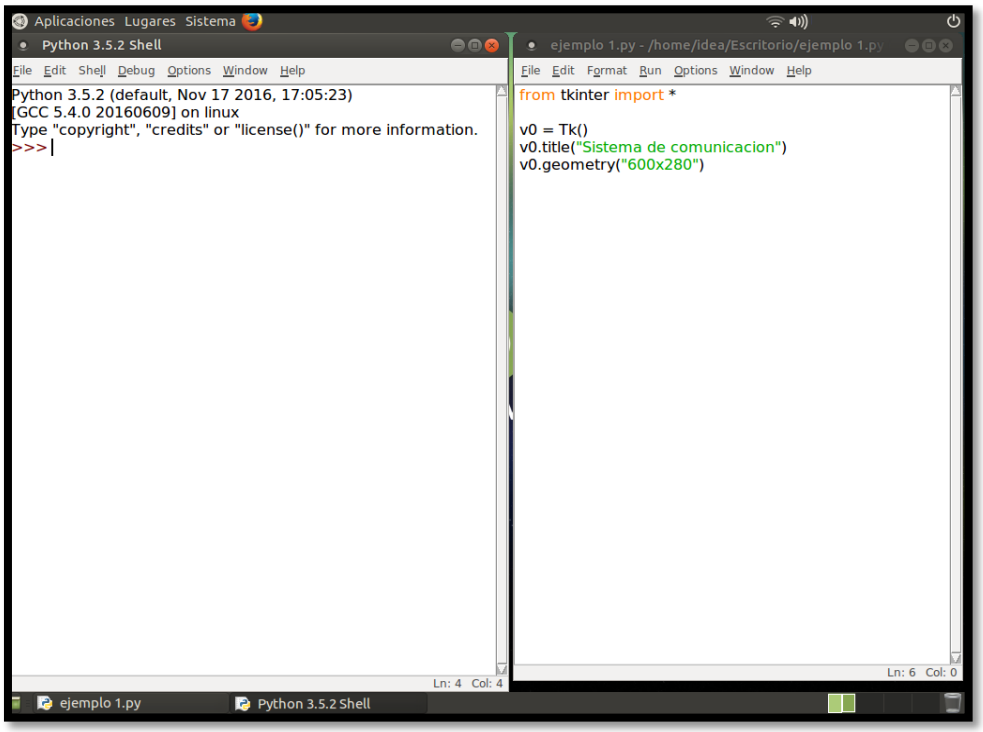


Ilustración I.31 Se captura la invocación en el editor de texto.[68]

[68] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

Una vez el texto haya sido capturado se debe ejecutar el programa para hacer esto en el editor de texto se debe hacer clic en la pestaña “Run” y después seleccionar la opción “Run Module” o simplemente presionar la tecla F5 (Ilustración I.32).

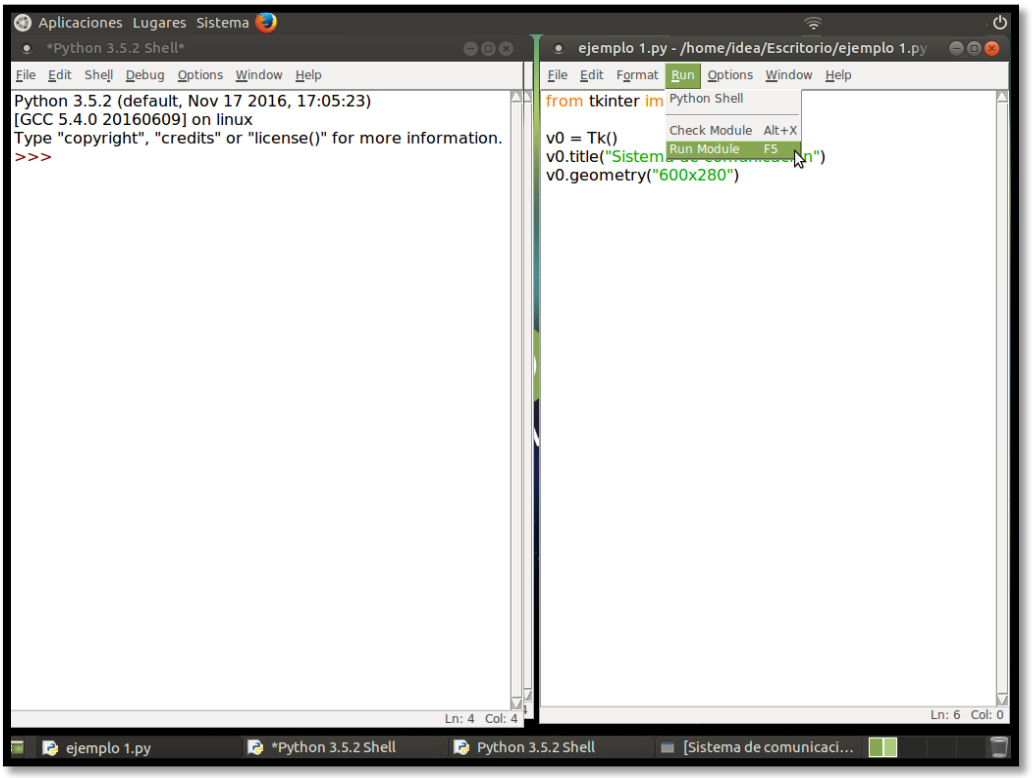


Ilustración I.32 Instrucciones para ejecutar un programa capturado en el editor de texto.[69]

Con esto el programa se ejecutará en el interpreté y hará la acción programada, en este caso, mostrar una ventana vacía con el título “Sistema de Comunicación” del tamaño configurado, ya que eso fue lo que se capturo en el editor, y si no tiene algún error de escritura se mostrará lo deseado (Ilustración 3.33).

[69] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

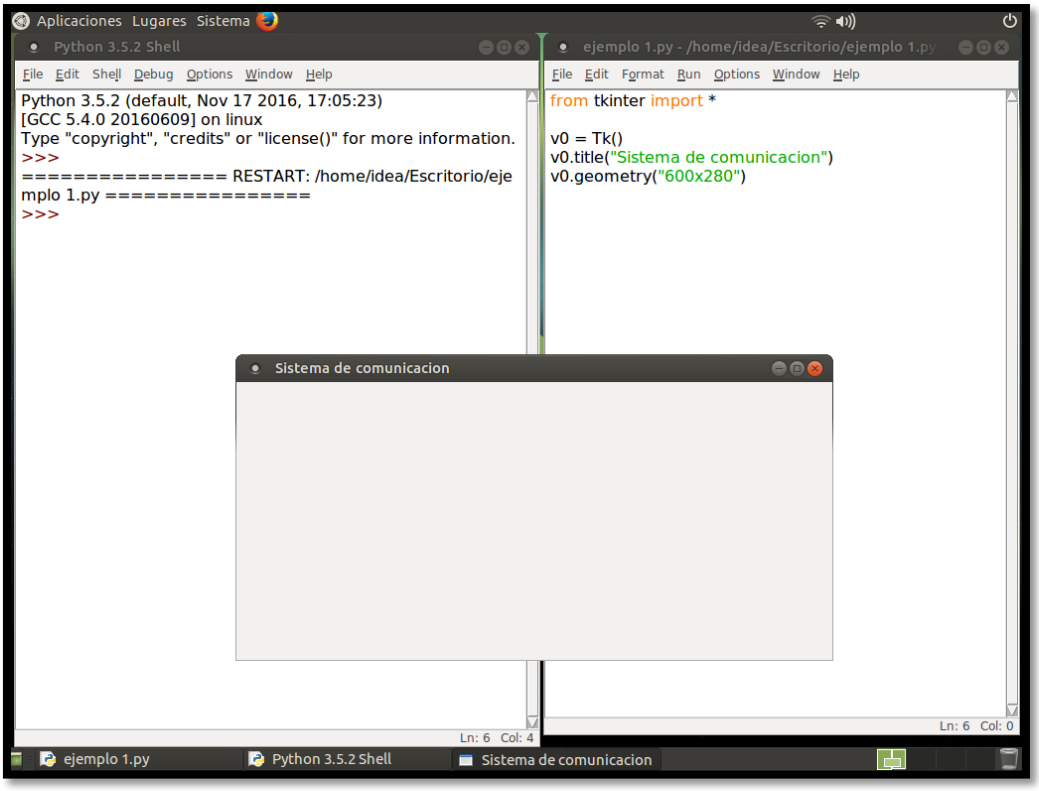


Ilustración I.33 Se muestra la ventana vacía que se programó.[70]

La librería de Tkinter tiene distintos widgets (que es como se le denomina a los elementos que se pueden incluir en la interfaz) y cada uno de ellos tiene su distintas propiedades que pueden ser programadas para ir incluyendo en la ventana o ventanas que se crean a la hora de hacer la interfaz, estos widgets no serán tratados durante esta propuesta ya que no es el fin de este trabajo, sin embargo hay mucha documentación en la red que sirve para entender mejor cada uno de los widgets y sus propiedades,[71] en el anexo de codificación de este trabajo se explica cómo fue creada a detalle la interfaz para esta propuesta, es algo sencilla pero funciona bastante bien, se puede apreciar en la Ilustración 3.34.

[70] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]
[71] En el siguiente enlace puede encontrarse una guía de introducción a Tkinter y sus widgets: http://effbot.org/tkinterbook/tkinter-index.htm

Ilustración I.34 Interfaz de la propuesta de Sistema de comunicación.[72]

La interfaz muestra el abecedario y un cursor que se mueve a través de él, desplazándose por la primera columna, y si se hace interrupción mediante el sensor de entrada, del cual se hará mención más adelante, el cursor comienza a desplazarse por las filas, es decir, el cursor todo el tiempo y en automático se desplaza por los caracteres: “A”, “E”, “I”, “O”, “U”, “5” y “_”, una vez llegue al último carácter que es el espacio (“_”), el cursor vuelve a comenzar en el carácter “A”, si por ejemplo se desea seleccionar la letra “H”, el cursos mediante el sensor, debe ser interrumpido cuando se encuentre en la letra “E” y comenzará a desplazarse ahora por filas, es decir por el carácter “F”, “G” y “H”, en cuanto llegue a este último, que es el deseado, se debe volver a interrumpir el cursor, con lo cual la letra “H” será seleccionada y aparecerá en el recuadro de abajo en la interfaz, de esta misma forma se deben seleccionar cada uno de los caracteres hasta que se genere la palabra, o en el caso de la Ilustración 3.34 la frase “HOLA MUNDO”, una vez se haya escrito lo que el usuario desea expresar, se utilizará el cursor para llegar a la opción “Voz”, mostrada en la interfaz, con la cual el texto pasará por un sintetizador de voz, del cual también se hará mención más adelante, para que se pueda escuchar la frase o palabra a través de las bocinas.

[72] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

I.3.3 Sensor de entrada.

Ya que cada uno de los pacientes que puedan llegar a ocupar este sistema de comunicación, tienen condiciones médicas distintas, se pensó en utilizar un método de entrada bastante general, esto debido a que, recordando, la ELA no afecta a todos de la misma manera y algún paciente puede tener control de la mano, pero otro no, alguno puede tener control de solo el pie, un musculo o solamente de los parpados, es por esto que el método de entrada que se utilizó para esta propuesta es de solamente dos estados, se activa, o no, con esto se garantiza que dependiendo de las condiciones del paciente se pueda remplazar de fácil manera y adaptarse a las condiciones necesarias, para la propuesta se utilizó un sensor touch del cual se hizo mención ya en el marco teórico, así que ahora se tratará el código utilizado para la implementación en el programa de ejecución.

Como ya se mencionó, también en el marco teórico, la Raspberry Pi 2 cuenta con 40 pines de propósito general (Ilustración 3.35), denominados (GPIO) y es donde podremos conectar el sensor de entrada.

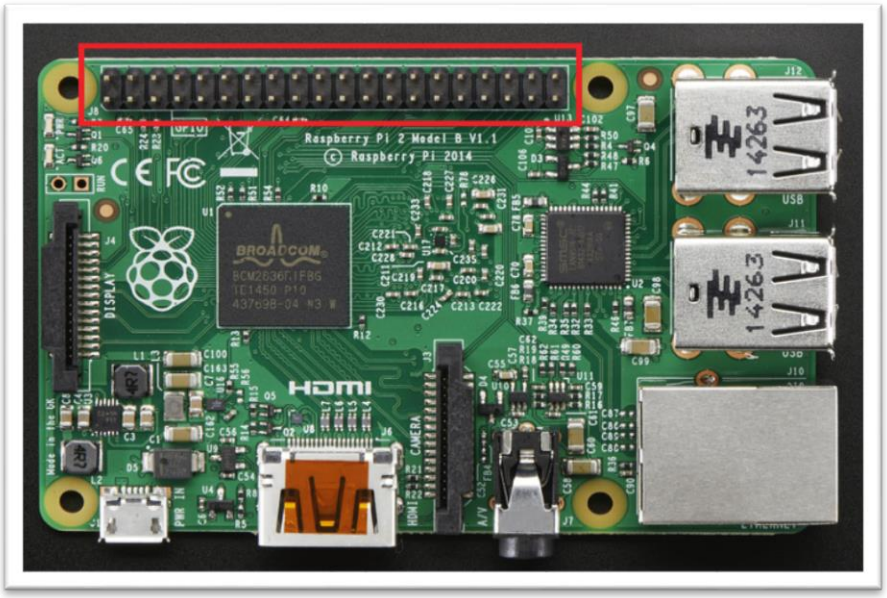


Ilustración I.35 Sección donde se encuentran los 40 pines GPIO. [73]

Estos pines cuentan con una distribución específica que se puede apreciar en la Ilustración 3.36 y como se mencionó previamente el sensor touch utilizado requiere de dos pines de alimentación y envía la señal solamente por un pin.

[73] Ilustración recuperada de: https://learn.adafruit.com/introducing-the-raspberry-pi-2-model b?view=all

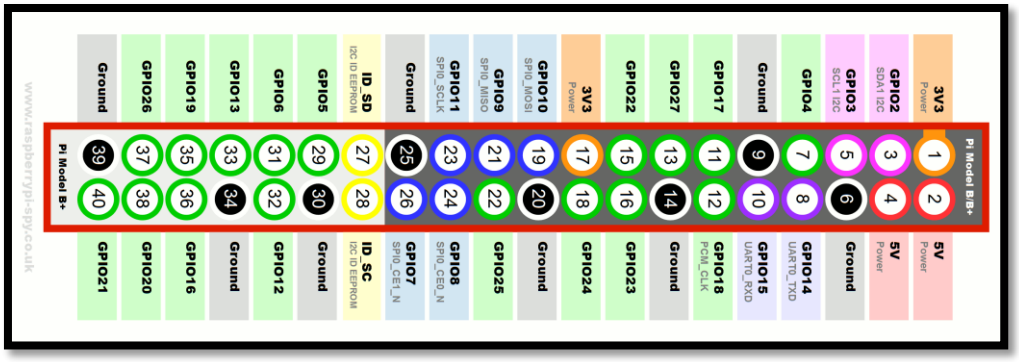


Ilustración I.36 Distribución de los pines GPIO. [74]

Por tanto la conexión para esta propuesta quedo de la siguiente manera, el pin del sensor que corresponde a “VCC” se conectó en el pin 4 de los GPIO, ya que este pin entrega 5v que es el voltaje con el cual opera el sensor, el pin del sensor correspondiente a “GND” se conectó al pin 6 de los GPIO, ya que este pin está conectado a tierra y es que el que cierra la alimentación del sensor, y el pin del sensor correspondiente a “SIG” que es el pin de señal se conectó en el pin 15 de los GPIO (Ilustración 3.37), ya que este es uno de los pines que puede programarse, aunque bien se pudo haber conectado en cualquier otro como el 7, 11 o 33 por ejemplo.

[74] Ilustración recuperada de: http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi gpio-header-and-pins/

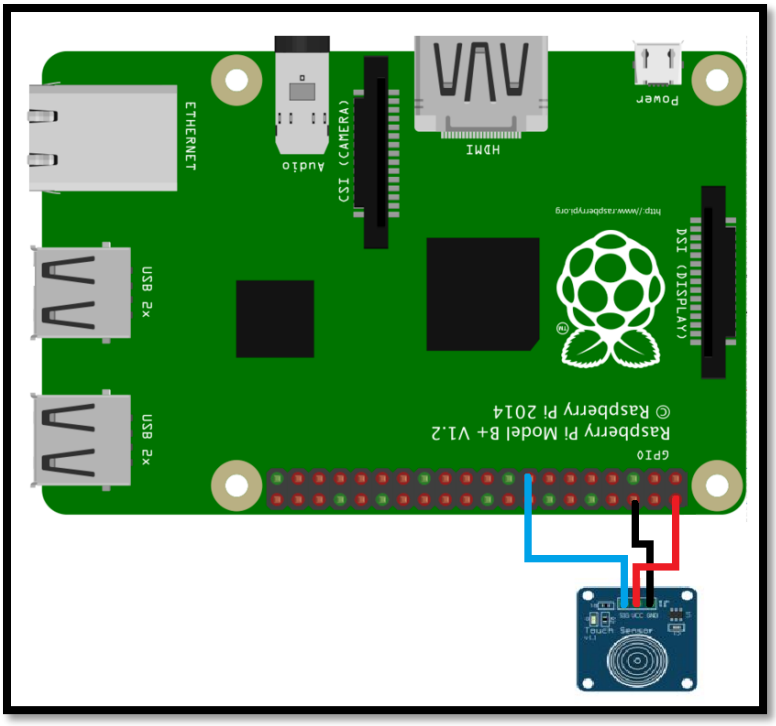


Ilustración I.37 Diagrama de conexión del sensor Touch.[75]

Ya que el sensor ha quedado conectado, se debe reconocer su comportamiento a través de la programación en Python, para lo cual se hará uso de la biblioteca "RPi.GPIO" que ya viene incluida en el sistema operativo y que se agrega al código de programación con la instrucción "import RPi.GPIO as GPIO".

A continuación, se verá un ejemplo del uso de esta librería y en primer lugar se deben agregar las Código I.2 al editor de texto del IDLE para Python 3.5.

```
1  import RPi.GPIO as GPIO
2
3  GPIO.setmode(GPIO.BCM)
4
5  GPIO.setup(22, GPIO.IN)
6
```

Código I.2 Instrucción que permite incluir la librería GPIO y configurar el pin 22 como entrada.

Donde:

[75] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

- Import RPi.GPIO as GPIO: es la instrucción que le indica al programa que incluya la librería GPIO al código.
- GPIO.setmode(GPIO.BMC): es la instrucción que habilita los pines GPIO para que estos puedan ser usados.
- GPIO.setup(22, GPIO.IN): es la instrucción que configura el pin 22 de los GPIO como entrada, ya que este pin es el que recibirá la señal del sensor.

Con la librería ya agregada al programa y los GPIO configurados se debe agregar las líneas restantes del Código I.3 al editor de texto.

```
1   import RPi.GPIO as GPIO
2
3   GPIO.setmode(GPIO.BCM)
4
5   GPIO.setup(22, GPIO.IN)
6
7   import time
8
9   while True:
10
11      if(GPIO.input(22)):
12
13          print("Presionado")
14      else:
15
16          print("Sin presionar")
17
18      time.sleep(1)
19
```

Código I.3 Código ejemplo del uso de los GPIO y el sensor.

Donde:

- import time: es la instrucción que le indica al programa que agregue la librería time.
- while True: es la instrucción que le indica al programa que se van a ejecutar el conjunto de instrucciones contenidas en esa función todo el tiempo una y otra vez.
- if (GPIO.input(22)): es la instrucción que le indica al programa que realice las instrucciones dentro de la función siempre y cuando el GPIO 22 reciba una señal.
- print (“Presionado”): es la instrucción que le indica al programa que si el GPIO 22 recibe una señal el interpreté muestre la palabra “Presionado”

- else: es la instrucción que le indica al programa que si la condición de la función if no se cumple realice las instrucciones contenidas dentro de la función else.
- print (“Sin presionar”): es la instrucción que le indica al programa que el intérprete muestre la frase “Sin presionar” si la condición de la función if no se cumple.
- time.sleep(1): es la instrucción que le indica al programa que haga un retardo de 1 segundo, esto con el fin de poder apreciar los cambios ya que de lo contrario al no agregar este retardo el intérprete mostraría de manera muy rápida las frases haciendo que no se aprecien de manera correcta.

Al ejecutar este programa se puede apreciar que el intérprete muestra la frase “Sin presionar” mientras que el sensor touch no se toca y que cada vez que hay contacto con el sensor touch muestra la palabra “Presionado”, comprobando cada segundo el estado, con lo que se verifica que se está recibiendo la señal de entrada (Ilustración 3.38).

En el anexo de codificación de este trabajo se encuentran detalladas las líneas de código correspondientes al uso de la librería GPIO, pero prácticamente es igual al del ejemplo ya que solo se necesita recibir la señal de entrada del sensor, a través de los GPIO.

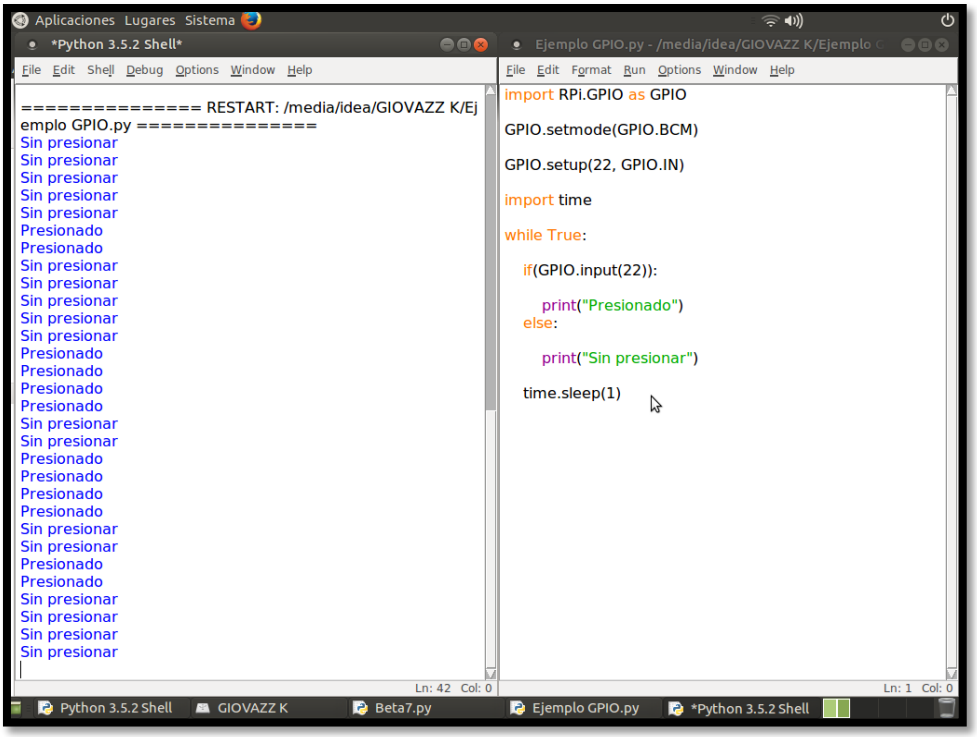


Ilustración I.38 Ejecución del programa que verifica la señal de entrada.[76]

I.3.4 eSpeak, Sintetizador de voz.

Para hacer que se puedan escuchar las palabras o frases que el usuario seleccione se hará uso de una biblioteca que tiene por nombre “eSpeak” que es la que contiene el sintetizador de voz que será utilizado para dicha función y que también se encuentra incluida en el sistema operativo.

Antes de usar eSpeak se debe hacer mención que la Raspberry Pi pude sacar audio tanto por el cable HDMI como por el plug 3.5 de audio, para elegir entre una u otra configuración hace falta escribir la siguiente línea de instrucción “sudo amixer cset numid=3 2” para poder escuchar el audio a través del HDMI y la siguiente línea “sudo amixer cset numid=3 1” si se desea escuchar el audio a través del plug 3.5 en una terminal del sistema operativo.

Para abrir una terminal hace falta hacer clic en la “pestaña aplicaciones”, seleccionar el submenú “Herramientas del sistema” y seleccionar “Terminal de MATE” como se muestra en la Ilustración 3.39, de inmediato se abrirá la terminal y en ella se de escribir la línea de código para activar la salida de audio de

[76] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

preferencia, para el caso de esta propuesta se utilizó la salida de audio a través del Jack 3.5 (Ilustración 3.40) debido a que se utilizan bocinas independientes.

Ilustración I.39 Ruta para abrir un Terminal de Ubuntu MATE.[77]

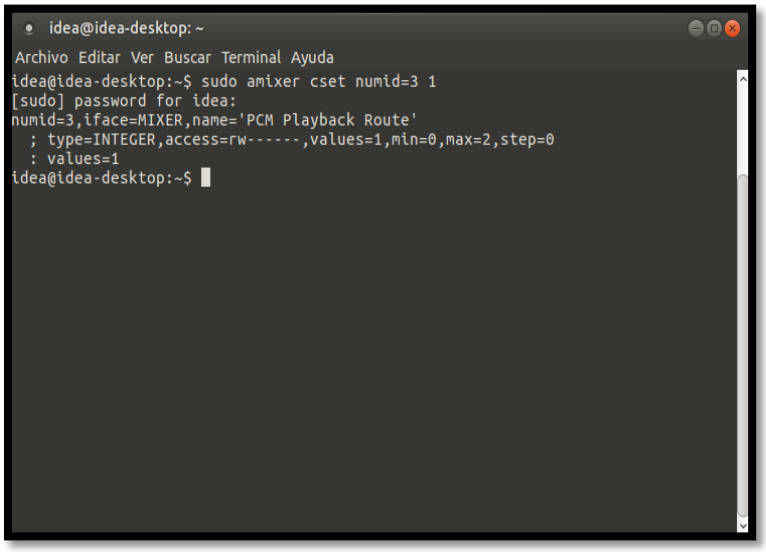


Ilustración I.40 Línea de instrucción para activar la salida de audio en el jack 3.5.[78]

[77] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]
[78] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

Ya con la salida de audio configurada se procede a probar el funcionamiento de eSpeak, para ello se abrirá una nueva terminal y se capturará la línea de instrucción del Código I.4 y se presionará enter, con lo que debería escucharse la síntesis de voz del texto entre comillas a través de la salida de audio (Ilustración I.41).

```
1  espeak “Hola, bienvenidos al sistema de comunicacion”
2
```

Código I.4 Línea de instruccion para invocar al sintetizador de voz eSpeak.

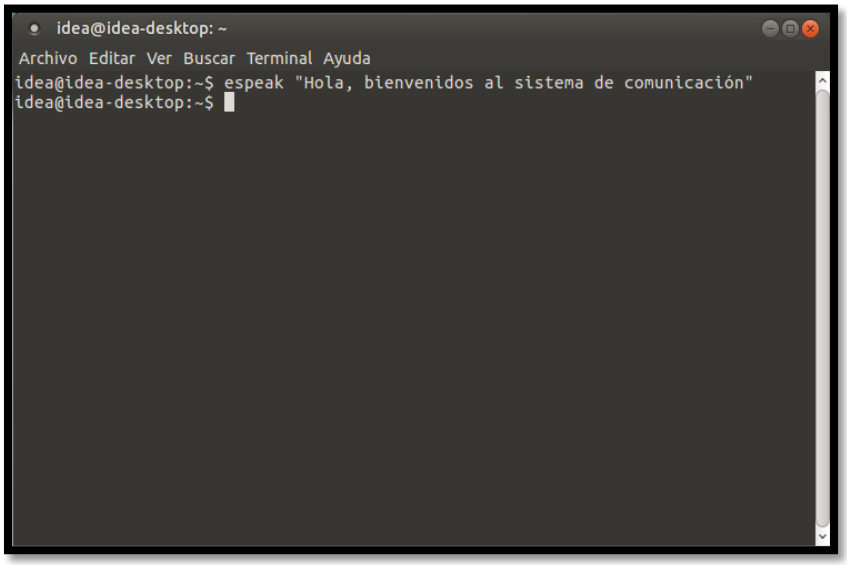


Ilustración I.41 Invocando al sintetizador de voz desde la Terminal MATE.[79]

La síntesis de voz del texto entre comillas se escucha con una mala pronunciación y esto es debido a que el idioma que tiene por defecto eSpeak es inglés y el texto escrito entre comillas se encuentra en español, eSpeak tiene distintas configuraciones como se mencionó en el marco teórico, así que para este casi haremos uso de la opción -ves para cambiar la voz al español, para lo cual la línea de instrucción que se debe capturar ahora en la terminal es la correspondiente a la Código I.5.

```
1  espeak-ves “Hola, bienvenidos al sistema de comunicacion”
2
```

Código I.5 Línea de instrucción que ejecuta el sintetizador de voz en idioma español.

El sintetizador de voz eSpeak se ejecuta desde la Terminal MATE sin embargo el programa de esta propuesta se encuentra escrito en un editor de texto para ejecutarse en Python, para lo que será necesario hacer uso de la biblioteca “os” que es la que

[79] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

servirá para pasar códigos escritos en Python a la terminal, para hacer demostración de ello, se deberá capturar las líneas de código de la Código I.6 en el editor de texto del IDLE para Python 3.5.

```
1  import os
2
3  os.system(espeak-ves “Hola, bienvenidos al sistema de comunicacion”)
4
```

Código I.6 Líneas de código que permiten invocar al sintetizador de voz eSpeak desde Python.

Donde:

- “import os”: es la instrucción que sirve para invocar a la biblioteca os al código del programa.
- “os.system(espeak-ves “Hola, bienvenidos al sistema de comunicación”)”: es la instrucción que le indica al programa que todo lo que este contenido entre paréntesis se ejecutará en una Terminal MATE y no en el interpreté de Python.

Al capturar estas líneas de código y ejecutarlas desde el editor de texto del IDLE para Python 3.5, el intérprete ejecutará el programa y pasará las instrucciones correspondientes a la Terminal MATE haciendo con esto que se invoque el sintetizador eSpeak desde Python y que a través de las bocinas se pueda escuchar la frase entre comillas (Ilustración 3.42).

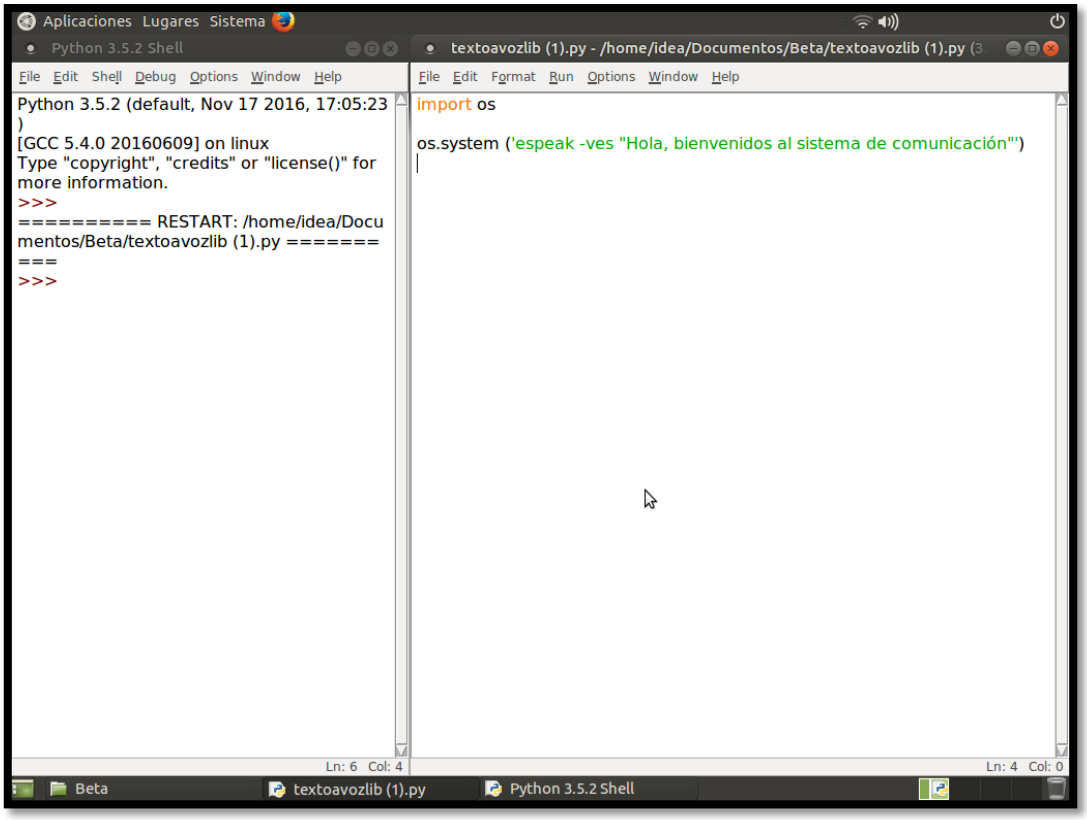


Ilustración I.42 Ejecución de las líneas de instrucciones que invocan al sintetizador de voz desde Python.[80]

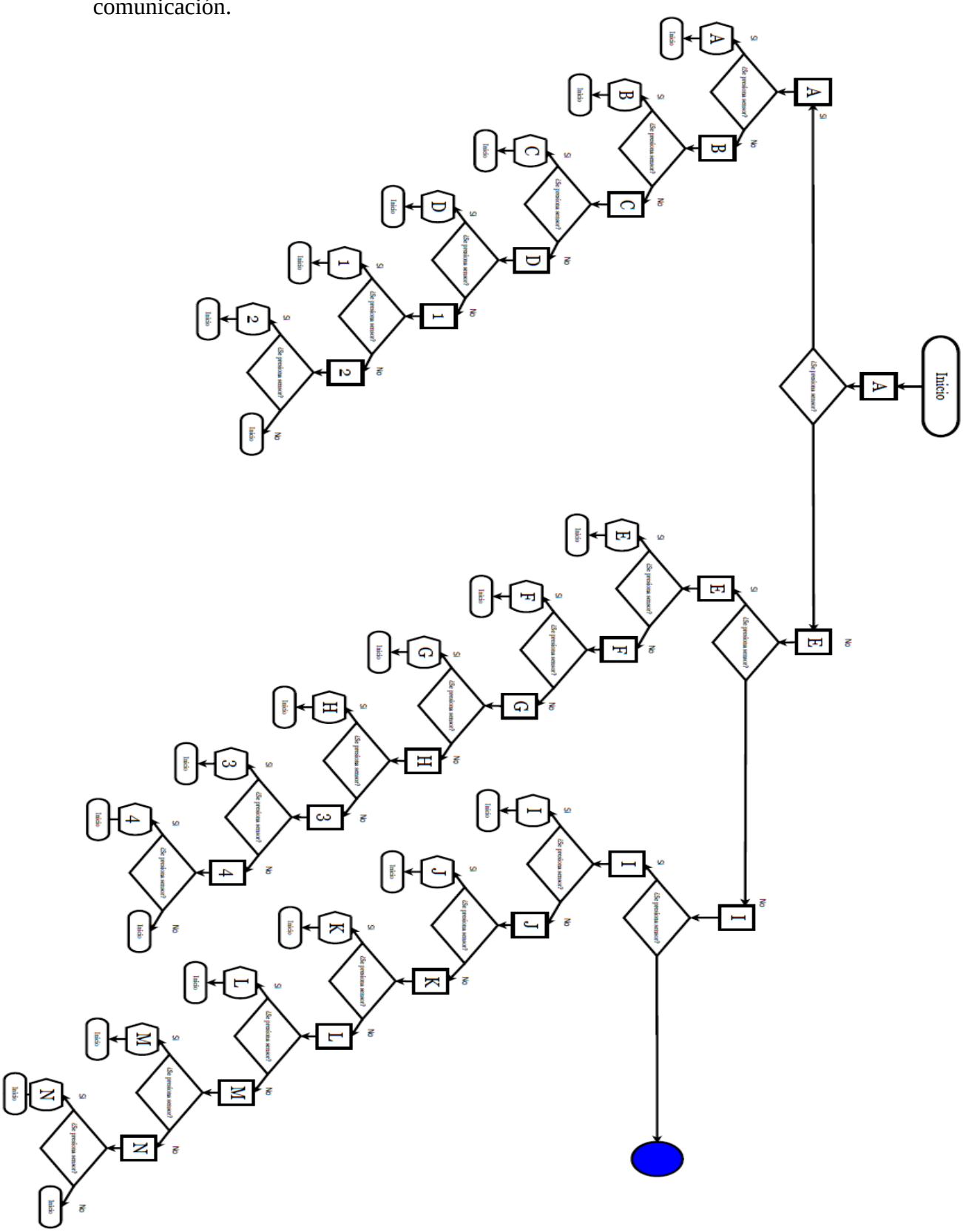
Así es como se invocará al sintetizador de voz eSpeak desde Python, en el anexo de codificación se explica que es lo que hace cada línea de instrucción referente al sintetizador de voz, pero básicamente es similar a lo que se muestra en el ejemplo anterior.

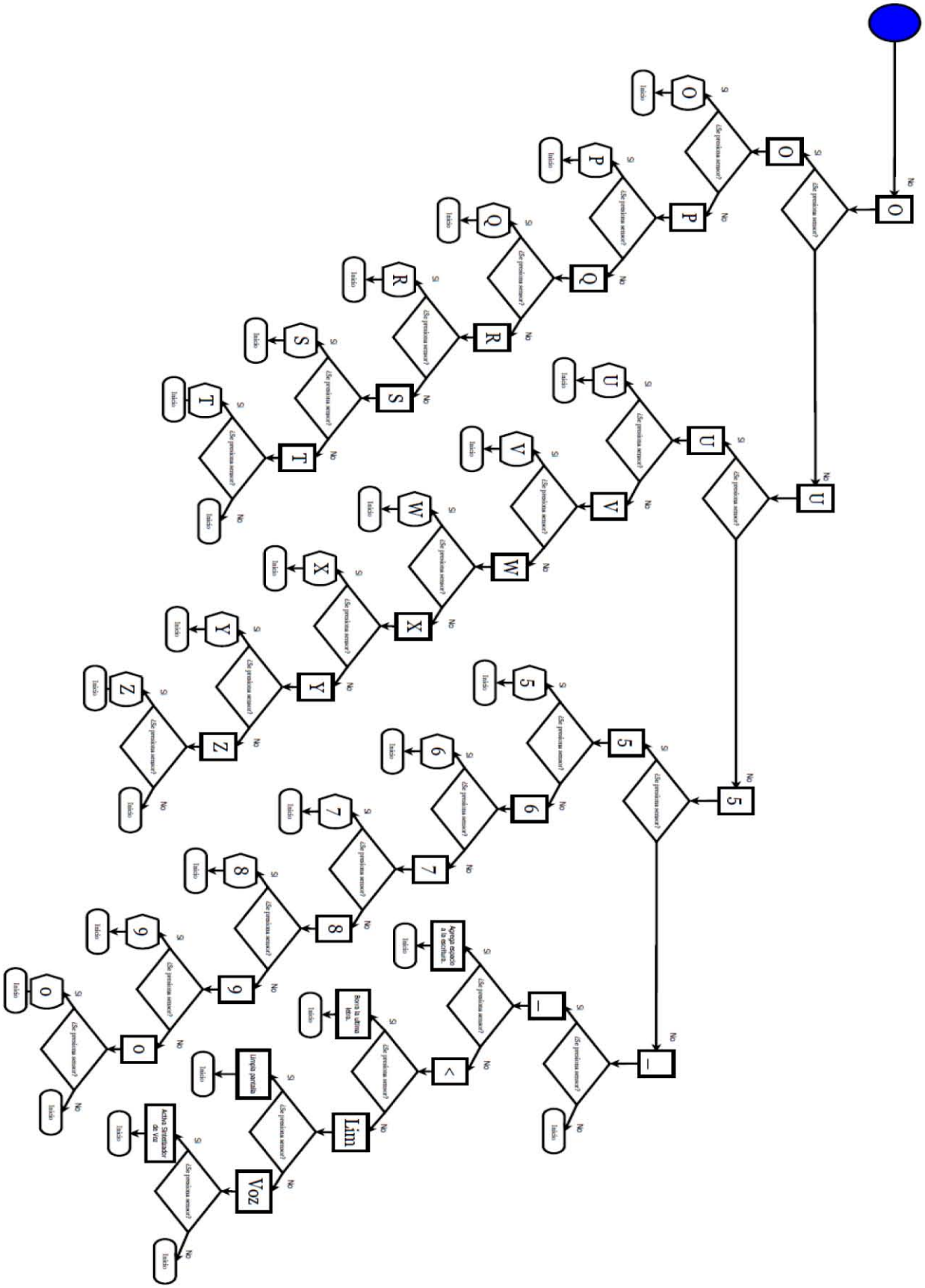
[80] Recuperada de: Autoría propia G. Belli [Diciembre, 2016]

II. Anexo B.

II.1 Diagrama de Flujo.

En este anexo se puede encontrar el diagrama de flujo completo del sistema de comunicación.




O
P
Q
R
S
T
U
V
W
X
Y
Z
5
6
7
8
9
0
_
<
Lim
Voz
Inicio
¿Se presiona acento?
Sí
No

III. Anexo C.

III.1 Código fuente del sistema de comunicación.

```
retardo = 0.6

i = 0


def CA():
    sv.set('''

»A        B       C        D        1        2

E        F       G        H        3        4

I        J       K        L        M        N

O        P       Q        R        S        T

U        V       W        X        Y        Z

5        6       7        8        9        0

_      <    Lim     Voz                        ''')


def CE():
    sv.set('''

A        B       C        D        1        2

»E        F       G        H        3        4

I        J       K        L        M        N

O        P       Q        R        S        T

U        V       W        X        Y        Z

5        6       7        8        9        0

_      <    Lim     Voz                        ''')

def CI():
    sv.set('''

A        B       C        D        1        2

E        F       G        H        3        4

»I        J       K        L        M        N

O        P       Q        R        S        T

U        V       W        X        Y        Z

```

```
5         6         7         8         9         0

_      <    Lim      Voz                          ''')

def CO():
    sv.set('''

A         B         C         D         1         2

E         F         G         H         3         4

I         J         K         L         M         N

»O         P         Q         R         S         T

U         V         W         X         Y         Z

5         6         7         8         9         0

_      <    Lim      Voz                          ''')


def CU():
    sv.set('''

A         B         C         D         1         2

E         F         G         H         3         4

I         J         K         L         M         N

O         P         Q         R         S         T

»U         V         W         X         Y         Z

5         6         7         8         9         0

_      <    Lim      Voz                          ''')

def C5():
    sv.set('''

A         B         C         D         1         2

E         F         G         H         3         4

 I         J         K         L         M         N

 O         P         Q         R         S         T

 U         V         W         X         Y         Z

 »5         6         7         8         9         0

 _      <    Lim      Voz                          ''')

 def CS():
     sv.set('''

 A         B         C         D         1         2
```

```
115.  E        F       G        H        3        4
116.
117.  I        J       K        L        M        N
118.
119.  O        P       Q        R        S        T
120.
121.  U        V       W        X        Y        Z
122.
123.  5        6       7         8        9         0
124.
125.  »_     <    Lim     Voz                        ''')
126.
127.  def LA():
128.      sv.set('''
129.
130.  >A        B       C        D        1        2
131.
132.  E        F       G        H        3        4
133.
134.  I        J       K        L        M        N
135.
136.  O        P       Q        R        S        T
137.
138.  U        V       W        X        Y        Z
139.
140.  5        6       7         8        9         0
141.
142.  _      <    Lim     Voz                        ''')
143.
144.  def LB():
145.      sv.set('''
146.
147.  A      >B       C        D        1        2
148.
149.  E        F       G        H        3        4
150.
151.  I        J       K        L        M        N
152.
153.  O        P       Q        R        S        T
154.
155.  U        V       W        X        Y        Z
156.
157.  5        6       7         8        9         0
158.
159.  _      <    Lim     Voz                        ''')
160.
161.  def LC():
162.      sv.set('''
163.
164.  A        B     >C        D        1        2
165.
166.  E        F       G        H        3        4
167.
168.  I        J       K        L        M        N
169.
170.  O        P       Q        R        S        T
171.
172.  U        V       W        X        Y        Z
173.
174.  5        6       7         8        9         0
175.
```

```
_      <    Lim     Voz                    ''')

def LD():
    sv.set('''

A        B       C     >D        1        2

E        F       G        H        3        4

I        J       K        L        M        N

O        P       Q        R        S        T

U        V       W        X        Y        Z

5        6       7         8        9         0

_      <    Lim     Voz                    ''')
def N1():
    sv.set('''

A        B       C        D    >1        2

E        F       G        H        3        4

I        J       K        L        M        N

O        P       Q        R        S        T

U        V       W        X        Y        Z

5        6       7         8        9         0

_      <    Lim     Voz                    ''')

def N2():
    sv.set('''

A        B       C        D        1       >2

E        F       G        H        3        4

I        J       K        L        M        N

O        P       Q        R        S        T

U        V       W        X        Y        Z

5        6       7         8        9         0

_      <    Lim     Voz                    ''')

def LE():
    sv.set('''

A        B       C        D        1        2

>E        F       G        H        3        4

I        J       K        L        M        N
```

```
237.
238.  O        P       Q        R        S        T
239.
240.  U        V       W        X        Y        Z
241.
242.  5        6       7         8        9         0
243.
244.  _      <    Lim     Voz                      ''')
245.
246.  def LF():
247.      sv.set('''
248.
249.  A        B       C        D        1        2
250.
251.  E      >F       G        H        3        4
252.
253.  I        J       K        L        M        N
254.
255.  O        P       Q        R        S        T
256.
257.  U        V       W        X        Y        Z
258.
259.  5        6       7         8        9         0
260.
261.  _      <    Lim     Voz                      ''')
262.
263.  def LG():
264.      sv.set('''
265.
266.  A        B       C        D        1        2
267.
268.  E        F     >G        H        3        4
269.
270.  I        J       K        L        M        N
271.
272.  O        P       Q        R        S        T
273.
274.  U        V       W        X        Y        Z
275.
276.  5        6       7         8        9         0
277.
278.  _      <    Lim     Voz                      ''')
279.
280.  def LH():
281.      sv.set('''
282.
283.  A        B       C        D        1        2
284.
285.  E        F       G     >H        3        4
286.
287.  I        J       K        L        M        N
288.
289.  O        P       Q        R        S        T
290.
291.  U        V       W        X        Y        Z
292.
293.  5        6       7         8        9         0
294.
295.  _      <    Lim     Voz                      ''')
296.
297.  def N3():
```

```
298.      sv.set('''
299.
300.  A         B         C         D         1         2
301.
302.  E         F         G         H     >3          4
303.
304.  I         J         K         L         M         N
305.
306.  O         P         Q         R         S         T
307.
308.  U         V         W         X         Y         Z
309.
310.  5         6         7          8         9          0
311.
312.  _       <     Lim      Voz                        ''')
313.
314.  def N4():
315.      sv.set('''
316.
317.  A         B         C         D         1         2
318.
319.  E         F         G         H         3       >4
320.
321.  I         J         K         L         M         N
322.
323.  O         P         Q         R         S         T
324.
325.  U         V         W         X         Y         Z
326.
327.  5         6         7          8         9          0
328.
329.  _       <     Lim      Voz                        ''')
330.
331.  def LI():
332.
333.      sv.set('''
334.
335.  A         B         C         D         1         2
336.
337.  E         F         G         H         3         4
338.
339.  >I         J         K         L         M         N
340.
341.  O         P         Q         R         S         T
342.
343.  U         V         W         X         Y         Z
344.
345.  5         6         7          8         9          0
346.
347.  _       <     Lim      Voz                        ''')
348.
349.  def LJ():
350.      sv.set('''
351.
352.  A         B         C         D         1         2
353.
354.  E         F         G         H         3         4
355.
356.  I       >J        K         L         M         N
357.
358.  O         P         Q         R         S         T
```

```
359.
360.  U        V       W        X        Y        Z
361.
362.  5        6       7         8        9         0
363.
364.  _      <    Lim     Voz                      ''')
365.
366.  def LK():
367.      sv.set('''
368.
369.  A        B       C        D        1        2
370.
371.  E        F       G        H        3        4
372.
373.  I        J     >K        L        M        N
374.
375.  O        P       Q        R        S        T
376.
377.  U        V       W        X        Y        Z
378.
379.  5        6       7         8        9         0
380.
381.  _      <    Lim     Voz                      ''')
382.
383.  def LL():
384.      sv.set('''
385.
386.  A        B       C        D        1        2
387.
388.  E        F       G        H        3        4
389.
390.  I        J       K     >L        M        N
391.
392.  O        P       Q        R        S        T
393.
394.  U        V       W        X        Y        Z
395.
396.  5        6       7         8        9         0
397.
398.  _      <    Lim     Voz                      ''')
399.
400.  def LM():
401.      sv.set('''
402.
403.  A        B       C        D        1        2
404.
405.  E        F       G        H        3        4
406.
407.  I        J       K        L    >M        N
408.
409.  O        P       Q        R        S        T
410.
411.  U        V       W        X        Y        Z
412.
413.  5        6       7         8        9         0
414.
415.  _      <    Lim     Voz                      ''')
416.
417.  def LN():
418.      sv.set('''
419.
```

```
A        B       C        D       1       2

E        F       G        H       3       4

I        J       K        L       M      >N

O        P       Q        R       S       T

U        V       W        X       Y       Z

5        6       7         8       9        0

_     <    Lim     Voz                    ''')

def LO():

    sv.set('''

A        B       C        D       1       2

E        F       G        H       3       4

I        J       K        L       M       N

>O        P       Q        R       S       T

U        V       W        X       Y       Z

5        6       7         8       9        0

_     <    Lim     Voz                    ''')

def LP():
    sv.set('''

A        B       C        D       1       2

E        F       G        H       3       4

I        J       K        L       M       N

O      >P       Q        R       S       T

U        V       W        X       Y       Z

5        6       7         8       9        0

_     <    Lim     Voz                    ''')

def LQ():
    sv.set('''

A        B       C        D       1       2

E        F       G        H       3       4

I        J       K        L       M       N

O        P     >Q        R       S       T

U        V       W        X       Y       Z
```

```
481.
482.  5        6       7         8        9         0
483.
484.       <    Lim     Voz                      ''')
485.
486.  def LR():
487.      sv.set('''
488.
489.  A        B       C        D        1        2
490.
491.  E        F       G        H        3        4
492.
493.  I        J       K        L        M        N
494.
495.  O        P       Q     >R         S        T
496.
497.  U        V       W        X        Y        Z
498.
499.  5        6       7         8        9         0
500.
501.  _      <    Lim     Voz                      ''')
502.
503.  def LS():
504.      sv.set('''
505.
506.  A        B       C        D        1        2
507.
508.  E        F       G        H        3        4
509.
510.  I        J       K        L        M        N
511.
512.  O        P       Q        R    >S        T
513.
514.  U        V       W        X        Y        Z
515.
516.  5        6       7         8        9         0
517.
518.  _      <    Lim     Voz                      ''')
519.
520.  def LT():
521.      sv.set('''
522.
523.  A        B       C        D        1        2
524.
525.  E        F       G        H        3        4
526.
527.  I        J       K        L        M        N
528.
529.  O        P       Q        R        S       >T
530.
531.  U        V       W        X        Y        Z
532.
533.  5        6       7         8        9         0
534.
535.  _      <    Lim     Voz                      ''')
536.
537.
538.  def LU():
539.
540.      sv.set('''
541.
```

```
542.  A         B        C         D         1         2
543.
544.  E         F        G         H         3         4
545.
546.  I         J        K         L         M         N
547.
548.  O         P        Q         R         S         T
549.
550.  >U         V        W         X         Y         Z
551.
552.  5         6        7         8         9          0
553.
554.  _      <     Lim      Voz                          ''')
555.
556.  def LV():
557.      sv.set('''
558.
559.  A         B        C         D         1         2
560.
561.  E         F        G         H         3         4
562.
563.  I         J        K         L         M         N
564.
565.  O         P        Q         R         S         T
566.
567.  U       >V        W         X         Y         Z
568.
569.  5         6        7         8         9          0
570.
571.  _      <     Lim      Voz                          ''')
572.
573.  def LW():
574.      sv.set('''
575.
576.  A         B        C         D         1         2
577.
578.  E         F        G         H         3         4
579.
580.  I         J        K         L         M         N
581.
582.  O         P        Q         R         S         T
583.
584.  U         V      >W         X         Y         Z
585.
586.  5         6        7         8         9          0
587.
588.  _      <     Lim      Voz                          ''')
589.
590.  def LX():
591.      sv.set('''
592.
593.  A         B        C         D         1         2
594.
595.  E         F        G         H         3         4
596.
597.  I         J        K         L         M         N
598.
599.  O         P        Q         R         S         T
600.
601.  U         V        W      >X         Y         Z
602.
```

```
5        6       7         8        9         0

_      <    Lim     Voz                    ''')

def LY():
    sv.set('''

A        B       C        D        1        2

E        F       G        H        3        4

I        J       K        L        M        N

O        P       Q        R        S        T

U        V       W        X    >Y        Z

5        6       7         8        9         0

_      <    Lim     Voz                    ''')

def LZ():
    sv.set('''

A        B       C        D        1        2

E        F       G        H        3        4

I        J       K        L        M        N

O        P       Q        R        S        T

U        V       W        X        Y       >Z

5        6       7         8        9         0

_      <    Lim     Voz                    ''')

def N5():

    sv.set('''

A        B       C        D        1        2

E        F       G        H        3        4

I        J       K        L        M        N

O        P       Q        R        S        T

U        V       W        X        Y        Z

>5        6       7         8        9         0

_      <    Lim     Voz                    ''')

def N6():
    sv.set('''

A        B       C        D        1        2

```

```
664.  E         F         G         H         3         4
665.
666.  I         J         K         L         M         N
667.
668.  O         P         Q         R         S         T
669.
670.  U         V         W         X         Y         Z
671.
672.  5       >6        7         8         9          0
673.
674.  _       <     Lim      Voz                        ''')
675.
676.  def N7():
677.      sv.set('''
678.
679.  A         B         C         D         1         2
680.
681.  E         F         G         H         3         4
682.
683.  I         J         K         L         M         N
684.
685.  O         P         Q         R         S         T
686.
687.  U         V         W         X         Y         Z
688.
689.  5         6      >7         8         9          0
690.
691.  _       <     Lim      Voz                        ''')
692.
693.  def N8():
694.      sv.set('''
695.
696.  A         B         C         D         1         2
697.
698.  E         F         G         H         3         4
699.
700.  I         J         K         L         M         N
701.
702.  O         P         Q         R         S         T
703.
704.  U         V         W         X         Y         Z
705.
706.  5         6         7      >8         9          0
707.
708.  _       <     Lim      Voz                        ''')
709.
710.  def N9():
711.      sv.set('''
712.
713.  A         B         C         D         1         2
714.
715.  E         F         G         H         3         4
716.
717.  I         J         K         L         M         N
718.
719.  O         P         Q         R         S         T
720.
721.  U         V         W         X         Y         Z
722.
723.  5         6         7          8        >9          0
724.
```

```
725.  _      <    Lim     Voz                      ''')
726.
727.  def N0():
728.      sv.set('''
729.
730.  A        B       C        D        1        2
731.
732.  E        F       G        H        3        4
733.
734.  I        J       K        L        M        N
735.
736.  O        P       Q        R        S        T
737.
738.  U        V       W        X        Y        Z
739.
740.  5        6       7         8        9      >0
741.
742.  _      <    Lim     Voz                      ''')
743.
744.
745.
746.  def RS():
747.      sv.set('''
748.
749.  A        B       C        D        1        2
750.
751.  E        F       G        H        3        4
752.
753.  I        J       K        L        M        N
754.
755.  O        P       Q        R        S        T
756.
757.  U        V       W        X        Y        Z
758.
759.  5        6       7         8        9         0
760.
761.  >_     <    Lim     Voz                      ''')
762.
763.
764.  def RB():
765.      sv.set('''
766.
767.  A        B       C        D        1        2
768.
769.  E        F       G        H        3        4
770.
771.  I        J       K        L        M        N
772.
773.  O        P       Q        R        S        T
774.
775.  U        V       W        X        Y        Z
776.
777.  5        6       7         8        9         0
778.
779.    _     ><    Lim     Voz                      ''')
780.
781.  def RL():
782.       sv.set('''
783.
784.  A        B       C        D        1        2
785.
```

```
786.  E        F       G        H        3        4
787.
788.  I        J       K        L        M        N
789.
790.  O        P       Q        R        S        T
791.
792.  U        V       W        X        Y        Z
793.
794.  5        6       7         8        9         0
795.
796.    _        <   >Lim     Voz                      ''')
797.
798.  def RV():
799.      sv.set('''
800.
801.  A        B       C        D        1        2
802.
803.  E        F       G        H        3        4
804.
805.  I        J       K        L        M        N
806.
807.  O        P       Q        R        S        T
808.
809.  U        V       W        X        Y        Z
810.
811.  5        6       7         8        9         0
812.
813.    _        <     Lim    >Voz                     ''')
814.
815.
816.
817.
818.
819.
820.  def delay():
821.
822.      var.set(x[0]+x[1]+x[2]+x[3]+x[4]+x[5]+x[6]+x[7]+x[8]
823.        +x[9]+x[10]+x[11]+x[12]+x[13]+x[14]+x[15]+x[16]+x[17]
824.        +x[18]+x[19]+x[20]+x[21]+x[22]+x[23]+x[24]+x[25]+x[26]
825.        +x[27]+x[28]+x[29]+x[30])
826.
827.
828.      v0.update()
829.      time.sleep(retardo)
830.
831.  def t():
832.      global i
833.      i = i + 1
834.
835.  def tm():
836.      global i
837.      i = i - 1
838.
839.
840.  def syn():
841.      global g
842.
843.  def Limp():
844.      x [0]=""
845.      x [1]=""
846.      x [2]=""
```

```
847.        x [3]=""
848.        x [4]=""
849.        x [5]=""
850.        x [6]=""
851.        x [7]=""
852.        x [8]=""
853.        x [9]=""
854.        x [10]=""
855.        x [11]=""
856.        x [12]=""
857.        x [13]=""
858.        x [14]=""
859.        x [15]=""
860.        x [16]=""
861.        x [17]=""
862.        x [18]=""
863.        x [19]=""
864.        x [20]=""
865.        x [21]=""
866.        x [22]=""
867.        x [23]=""
868.        x [24]=""
869.        x [25]=""
870.        x [26]=""
871.        x [27]=""
872.        x [28]=""
873.        x [29]=""
874.        x [30]=""
875.
876.
877.    def Loop():
878.
879.        CA()
880.        delay()
881.        if (GPIO.input(22)):
882.            LA()
883.            delay()
884.            if (GPIO.input(22)):
885.                x[i] = 'A'
886.                t()
887.                print('A')
888.                Loop()
889.            else:
890.                LB()
891.                delay()
892.                if (GPIO.input(22)):
893.                    x[i]='B'
894.                    t()
895.                    print('B')
896.                    Loop()
897.                else:
898.                    LC()
899.                    delay()
900.                    if (GPIO.input(22)):
901.                        x[i]='C'
902.                        t()
903.                        print('C')
904.                        Loop()
905.                    else:
906.                        LD()
907.                        delay()
```

```
                        if (GPIO.input(22)):
                            x[i]='D'
                            t()
                            print('D')
                            Loop()
                        else:
                            N1()
                            delay()
                            if (GPIO.input(22)):
                                x[i] = '1'
                                t()
                                print('1')
                                Loop()
                            else:
                                N2()
                                delay()
                                if (GPIO.input(22)):
                                    x[i] = '2'
                                    t()
                                    print('2')
                                    Loop()
                                else:
                                    Loop()

    CE()
    delay()
    if (GPIO.input(22)):
        LE()
        delay()
        if (GPIO.input(22)):
            x[i] = 'E'
            t()
            print('E')
            Loop()
        else:
            LF()
            delay()
            if (GPIO.input(22)):
                x[i] = 'F'
                t()
                print('F')
                Loop()
            else:
                LG()
                delay()
                if (GPIO.input(22)):
                    x[i] = 'G'
                    t()
                    print('G')
                    Loop()
                else:
                    LH()
                    delay()
                    if (GPIO.input(22)):
                        x[i] = 'H'
                        t()
                        print('H')
                        Loop()
                    else:
                        N3()
                        delay()
```

```
                            if (GPIO.input(22)):
                                x[i] = '3'
                                t()
                                print('3')
                                Loop()
                            else:
                                N4()
                                delay()
                                if (GPIO.input(22)):
                                    x[i] = '4'
                                    t()
                                    print('4')
                                    Loop()
                                else:
                                    Loop()

    CI()
    delay()
    if (GPIO.input(22)):
        LI()
        delay()
        if (GPIO.input(22)):
            x[i] = 'I'
            t()
            print('I')
            Loop()
        else:
            LJ()
            delay()
            if (GPIO.input(22)):
                x[i] = 'J'
                t()
                print('J')
                Loop()
            else:
                LK()
                delay()
                if (GPIO.input(22)):
                    x[i] = 'K'
                    t()
                    print('K')
                    Loop()
                else:
                    LL()
                    delay()
                    if (GPIO.input(22)):
                        x[i] = 'L'
                        t()
                        print('L')
                        Loop()
                    else:
                        LM()
                        delay()
                        if (GPIO.input(22)):
                            x[i] = 'M'
                            t()
                            print('M')
                            Loop()
                        else:
                            LN()
                            delay()
```

```
                                if (GPIO.input(22)):
                                    x[i] = 'N'
                                    t()
                                    print('N')
                                    Loop()
                                else:
                                    Loop()

    CO()
    delay()
    if (GPIO.input(22)):
        LO()
        delay()
        if (GPIO.input(22)):
            x[i] = 'O'
            t()
            print('O')
            Loop()
        else:
            LP()
            delay()
            if (GPIO.input(22)):
                x[i] = 'P'
                t()
                print('P')
                Loop()
            else:
                LQ()
                delay()
                if (GPIO.input(22)):
                    x[i]='Q'
                    t()
                    print('Q')
                    Loop()
                else:
                    LR()
                    delay()
                    if (GPIO.input(22)):
                        x[i] = 'R'
                        t()
                        print('R')
                        Loop()
                    else:
                        LS()
                        delay()
                        if (GPIO.input(22)):
                            x[i] = 'S'
                            t()
                            print('S')
                            Loop()
                        else:
                            LT()
                            delay()
                            if (GPIO.input(22)):
                                x[i] = 'T'
                                t()
                                print('T')
                                Loop()
                            else:
                                Loop()

```

```
    CU()
    delay()
    if (GPIO.input(22)):
        LU()
        delay()
        if (GPIO.input(22)):
            x[i] = 'U'
            t()
            print('U')
            Loop()
        else:
            LV()
            delay()
            if (GPIO.input(22)):
                x[i] = 'V'
                t()
                print('V')
                Loop()
            else:
                LW()
                delay()
                if (GPIO.input(22)):
                    x[i] = 'W'
                    t()
                    print('W')
                    Loop()
                else:
                    LX()
                    delay()
                    if (GPIO.input(22)):
                        x[i] = 'X'
                        t()
                        print('X')
                        Loop()
                    else:
                        LY()
                        delay()
                        if (GPIO.input(22)):
                            x[i] = 'Y'
                            t()
                            print('Y')
                            Loop()
                        else:
                            LZ()
                            delay()
                            if (GPIO.input(22)):
                                x[i] = 'Z'
                                t()
                                print('Z')
                                Loop()
                            else:
                                Loop()

    C5()
    delay()
    if (GPIO.input(22)):
        N5()
        delay()
        if (GPIO.input(22)):
            x[i] = '5'
            t()
```

```
1152.                print('5')
1153.                Loop()
1154.            else:
1155.                N6()
1156.                delay()
1157.                if (GPIO.input(22)):
1158.                    x[i] = '6'
1159.                    t()
1160.                    print('6')
1161.                    Loop()
1162.                else:
1163.                    N7()
1164.                    delay()
1165.                    if (GPIO.input(22)):
1166.                        x[i] = '7'
1167.                        t()
1168.                        print('7')
1169.                        Loop()
1170.                    else:
1171.                        N8()
1172.                        delay()
1173.                        if (GPIO.input(22)):
1174.                            x[i] = '8'
1175.                            t()
1176.                            print('8')
1177.                            Loop()
1178.                        else:
1179.                            N9()
1180.                            delay()
1181.                            if (GPIO.input(22)):
1182.                                x[i]  = '9'
1183.                                t()
1184.                                print('9')
1185.                                Loop()
1186.                            else:
1187.                                N0()
1188.                                delay()
1189.                                if (GPIO.input(22)):
1190.                                    x[i] = '0'
1191.                                    t()
1192.                                    print('0')
1193.                                    Loop()
1194.                                else:
1195.                                    Loop()
1196.
1197.    CS()
1198.    delay()
1199.    if (GPIO.input(22)):
1200.        RS()
1201.        delay()
1202.        if (GPIO.input(22)):
1203.            x[i] = ' '
1204.            t()
1205.            print(' ')
1206.            Loop()
1207.        else:
1208.             RB()
1209.             delay()
1210.             if (GPIO.input(22)):
1211.                 x[i-1]  =''
1212.                 tm()
```

```
                  print('_')
                  Loop()
              else:
                  RL()
                  delay()
                  if (GPIO.input(22)):
                      Limp()
                      t()
                      print('Limpiar')
                      Loop()
                  else:
                        RV()
                        delay()
                        if (GPIO.input(22)):
                         g = x[0]+x[1]+x[2]+x[3]+x[4]+x[5]
                        +x[6]+x[7]+x[8]+x[9]+x[10]+x[11]
                        +x[12]+x[13]+x[14]+x[15]+x[16]
                        +x[17]+x[18]+x[19]+x[20]+x[21]
                        +x[22]+x[23]+x[24]+x[25]+x[26]
                        +x[27]+x[28]+x[29]+x[30]
                         subprocess.call('espeak -v es+m1 -
                        s130 "{0}"'.format(g),shell=True)

                        else:
                             Loop()


     Loop()


import subprocess
from tkinter import *
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

v0 = Tk()

v0.title("Sistema de Comunicación")

v0.geometry("800x600")

sv = StringVar()

var = StringVar()

et = Label(v0, textvariable = sv, font = ( "Helvetica", 15 )
)

et.pack()

lt = Label(v0, textvariable = var, font = ( "
Helvetica", 17 ))

lt.pack()

x = ["","","","","","","","","","","","","","","","","","","",""
","","","","","","","","","","","",""]

GPIO.setup(22, GPIO.IN)
```

```
1274.
1275. syn()
1276.
1277. Loop()
1278.
1279. v0.mainloop()
```