UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

DIVISIÓN DE ELECTRÓNICA Y COMPUTACIÓN

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

INGENIERÍA EN COMUNICACIONES Y ELECTRÓNICA
INGENIERÍA ROBÓTICA
INGENIERÍA FOTÓNICA

PARA ACREDITAR LOS MÓDULOS

P R E S E N T A

OMAR BARUCH MORÓN LÓPEZ BRENDA ISABEL ESTRADA HERNÁNDEZ DAVID ALEJANDO RODRÍGUEZ PRECIADO

A S E S O R

JUAN CARLOS ALDAZ ROSAS

Guadalajara, Jalisco, jueves, 5 de diciembre de 2019

ÍNDICE

INTRODUCCIÓN

A continuación, se describe de forma detallada el desarrollo para la creación de un animatrónico con temática navideña, llamado “Snowy”, con el propósito de acreditar los proyectos modulares respectivos de la carrera de Ingeniería Electrónica y Comunicaciones (INCE), Ingeniería Fotónica (IGFO) e Ingeniería Robótica (INRO).

En el marco de “DIVEC” Innovación 2019, que es un programa completo para formar habilidades tanto de maestros como de estudiantes en torno a los conceptos de pensamiento innovador, trabajo en equipo, desarrollo multidisciplinar y nuevas fronteras de la tecnología, se llevó a cabo el reto de proyectos modulares, dicho reto tenía como requerimiento que fuera un animatrónico con una fuente de energía sustentable, demostrando que es eficiente en términos de consumo energético y circuitería mínima; con valores, actitudes y prácticas saludables, con alcance comunitario. Así mismo, que cuente con aplicaciones de iluminación controlada y con un mínimo de dos grados de movimiento, todo lo anterior con un límite de tiempo de una semana únicamente.

Cabe mencionar, que el proyecto “Snowy” cumple todos los requerimientos antes establecidos, pero también cuenta con funciones extras, aumentando el esfuerzo, pero mejorando sus resultados.

PLANTEAMIENTO DEL PROBLEMA

Un animatrónico navideño cumple con varios aspectos, todos son amigables estéticamente hablando, hacen alguna función animatrónica como cantar, hablar o bailar, las cuales son de mero entretenimiento y adorno, sin embargo, se decidió que aparte de dichas funciones y aspectos que comparten en común todos los animatrónicos, también tuviera un impacto social. Por lo que “Snowy”, el cuál es caracterizado como un mono de nieve, está dirigido al público en general, es decir, todos pueden gozar de las bondades que ofrece, pero hacemos un especial hincapié a las personas invidentes. Ya que ellas al no poder ser partícipes del diseño estético que cualquier animatrónico navideño tiene, si pueden ser partícipes de sus funciones de una manera simple y rápida, esto al alcance de su teléfono celular. Facilitando así su vida cotidiana.

Planteando esto, intervienen varios puntos a considerar:

En cuanto al animatrónico

- Conseguir un amplificador de audio.
- Desarrollar una aplicación móvil, la cual reconozca comandos de voz.
- Establecer una comunicación serial para comunicar la aplicación desde el teléfono, con el microcontrolador que se vaya a utilizar.
- Transformar comandos de voz, en código hexadecimal para que se reproduzcan.
- Encontrar un diseño práctico y económico para hacer la carcasa.

En cuanto a la conversión de energía

- Encontrar un panel solar o hacer un arreglo de paneles solares con el voltaje suficiente para que cargue una batería.
- Diseñar un regulador de voltaje.
- Diseñar un indicador de voltaje parametrizado, según las características de la batería.

JUSTIFICACIÓN

Según el INEGI en la zona metropolitana de Guadalajara, casi el 30% (70 mil personas) son invidentes, siendo esta la segunda discapacidad con más gente que la padece. (INEGI, 2010).

Sin embargo, una discapacidad no debe de ser motivo para que alguien este privado de la tecnología, al contrario, debe de ser una herramienta que facilite su vida. Es por eso que el desarrollo de Snowy ayuda a que dichas personas tengan una mejor interacción, a través de su celular. El cual, en la configuración predeterminada del mismo, con sólo presionar un botón la aplicación móvil, se abre y puedes preguntar, hora, fecha, días faltantes para navidad, el clima, llamar a un amigo o familiar, poner música (en este caso, es un villancico), tomar selfie, así como también hacer el control de iluminación mediante comandos de voz.

Es determinable resaltar el hecho de que consume energía sustentable por medio del sol, ya que todo lo obtiene de un panel solar. Convirtiendo a Snowy sustentable e inclusivo.

ANTECEDENTES ESTADO DE ARTE

En la actualidad se habla mucho de animatrónicos y robots, siendo lo más actual en tecnología y que se siguen innovando día tras día, ¿Quién no recuerda haber visto por primera vez un animatrónico en alguna película de ciencia ficción, o en algún juguete? Si bien, los primeros animatrónicos que existieron eran muy diferentes a los actuales, sin duda alguna causaron la misma emoción e interés en los espectadores y es que este fue uno de los mejores medios para involucrarnos como sociedad en la modernidad y en las tendencias tecnológicas. Existen muchos tipos con diferentes funciones, algunos apegados a necesidades de un público especifico, hasta aquellos que son por mero entretenimiento. Y aunque han deslumbrado a todo el mundo siendo todo un arte de ingeniería aplicada, todavía hay mucho que estudiar y desarrollar.

Primeramente, es necesario diferenciar entre un animatrónico y un robot y es que el primero consiste en que funciona con movimientos y sonidos pregrabados, en lugar de responder a estimulo externo. Además de que un robot como tal, tiene muchas más componentes, una de ellas y muy interesante es la inteligencia artificial, sólo por mencionar un ejemplo. Considerándolo desde este punto de vista se puede decir que un animatrónico es el antecesor de un robot. En el caso de “Snowy”, entra en la primera clasificación ya que el cerebro de todas sus funciones y su estructura es solamente un microcontrolador.

HIPÓTESIS

Los animatrónicos desde su aparición hasta la actualidad han estado inmersos en diferentes áreas desde su aplicación en relojes, atracciones, cine y televisión, juguetes y videojuegos hasta algunas de las principales empresas de animatrónica que son «Garner Holt Productions, Inc.» en San Bernardino, California; «UCFab International, LLC» en Apopka, Florida; «Sally Corporation» en Jacksonville, Florida; y «Lifeformations» en Bowling Green (Ohio). Todas ellas valoradas en millones de dólares. Y encargadas de crear ambientaciones siendo la industria cinematográfica su principal cliente. (García, 2017).

Dentro de este proyecto se buscará hacer un animatrónico totalmente sustentable e inclusivo, demostrando así que, no sólo cumple con la parte de decoración y entretenimiento como cualquier otro, sino también con la adaptación de la tecnología en una causa social.

METODOLOGÍA

Sin duda alguna, este proyecto se prestó, para poder poner en práctica los conocimientos adquiridos a lo largo de la carreara como ingenieros, y no solo eso, ya que como su nombre lo menciona es un "Reto" al considerar el corto tiempo para desarrollarlo, el proyecto sin duda alguna saca lo mejor de los estudiantes, forzando a investigar de forma autodidacta, ya que se hizo mucho trabajo de investigación por parte de los integrantes.

En el día 1, cuando se consolidó el equipo multidisciplinario, se realizó una revisión tanto del material que cada uno disponía, así como de los conocimientos que cada uno dominaba para poderlos integrar de la mejor manera posible. Apegándose a lo anterior se decidió realizar a "Snowy" ya que, al ser dos bolas de unicel decoradas, no demandaba mucho tiempo el diseño de la carcasa, lo cual fue una ventaja porque ese tiempo se invirtió en mejorar y eficientizar sus funciones. Además de que de este modo también cumple con la condición perfecta de ser un adorno navideño.

En el día 2, se comenzó por conceptualizar y diseñar tanto los circuitos de conversión de energía, para saber qué tanta corriente iba a demandar "Snowy", designar específicamente que funciones se iban a realizar desde el microcontrolador, así como empezar la programación de la aplicación.

En el día 3, se consiguió una celda sola, una batería y servomotores.

Los días 4 y 5, lo que más demandó tiempo a resolver fue la parte del audio, se consiguió un amplificador por internet, y conectado a un altavoz, se logró reproducir la música, aunque al principio se tuvo mucha interferencia de ruido. Así mismo se hicieron pruebas de los servos, se compró el material de decoración y una caja como base donde se guardó todos los circuitos, y el altavoz.

Simultáneamente se realizaron pruebas de los circuitos reguladores de voltaje e indicadores de voltaje, en la paca de pruebas, para decidir qué diseño cumplía mejor con la función, requería menos componentes electrónicos y consumía menos energía.

Y finalmente, en el día 6 del reto, se juntaron todas las partes y se hicieron la asignación de pines del microcontrolador, así como se hicieron las conexiones pertinentes y decoración.

De cada una de estas etapas se habla más adelante con mayor detenimiento.

DISEÑO Y FASE DE PROTOTIPADO DEL PROYECTO MODULAR

SISTEMA EMBEBIDO

El sistema embebido utilizado para la implementación del prototipo es la tarjeta ESP32 (Systems, 2019), dicha tarjeta es un sistema en chip o Soc (“System on a chip”) por sus siglas en inglés, lo cual quiere decir que en su fabricación se integra una gran parte de los módulos que componen a un sistema informático o electrónico en un solo circuito integrado. Esta tarjeta es fabricada por la compañía “Expressif Systems”, la cual es una versión mejorada de su predecesora ESP8266 la cual es ampliamente utilizada en proyectos IoT (Internet of Things) enfocados al internet de las cosas.

De acuerdo con su hoja de datos, la tarjeta ESP32 tiene integrados los módulos Wifi y Bluetooth los cuales debió a sus protocolos de comunicaciones pueden ser utilizados a la par ya que una señal no interfiere con la otra.

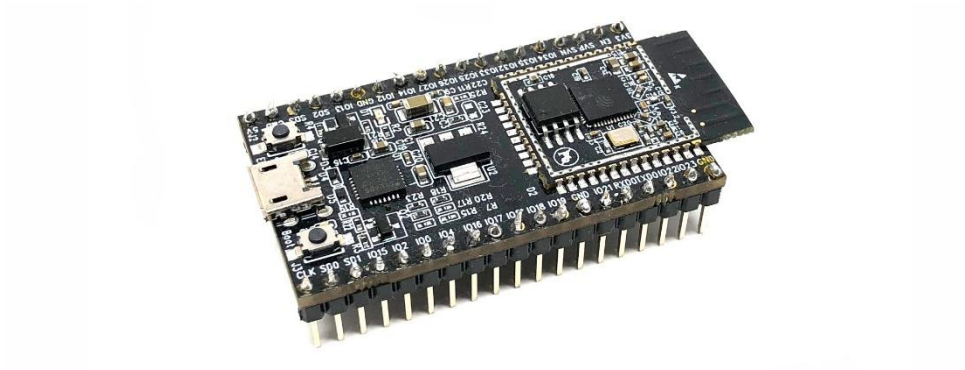


Ilustración 1: Placa embebida ESP32

JUSTIFICACIÓN DE ELECCIÓN

Se hace uso de este sistema embebido debido a algunas de sus características, útiles para el prototipo implementado, tales como son: la comunicación Wifi; debido a que por este protocolo de comunicaciones se realiza el envío de los comandos para controlar el prototipo, la resolución que tiene el conversor digital/análoga; necesaria para la reproducción de archivos de audio grabados en la memoria de la tarjeta ESP32, al igual que por sus salidas de control de modulación de pulso; útiles para el control de leds RGB (rojo, verde, azul) por sus siglas en inglés y el control de posicionamiento de los servomotores.

CARACTERÍSTICAS

La tarjeta cuenta con la capacidad de ser alimentada por medio de una entrada Micro-USB, puede ser alimentada por 5 Volts los cuales son regulados por la misma para ser modificados y energizarla, al igual que puede ser energizada por 3.3Volts, cuenta con un botón habilitador, esto por motivos de seguridad ya que restablece las configuraciones realizadas en la programación de la tarjeta, cuenta con seis pines analógicos los cuales realizan la medición de voltajes en el rango de 0 a 3.3Volts, además de un ADC de 12 bits con 18 canales de conversión analógica a digital, cuenta con dos conversores DAC (análogo a digital), además de un total de 39 pines que pueden ser utilizados como entradas o salidas, sin embargo, los pines 34 al 39 solo pueden ser utilizados como entradas. Cuenta también con otras funciones como comunicación VSPI, HSPI, I2C, comunicación serial, interrupciones externas, 16 canales independientes que pueden ser utilizados para modulación por ancho de pulso y un pin de referencia de voltaje externo.

El protocolo que utiliza para la comunicación Wifi es el 802.11 b/g/n y para la comunicación Bluetooth V4.2, su microprocesador es el "*Tensilica Xtensa LX6"* y su frecuencia de operación máxima es de 24MHz.

DISEÑO Y PROTOTIPADO ELECTRÓNICO

ADQUISICIÓN DE ENERGIA

Fundamentos

Las energías renovables cada vez se hacen más presentes en la vida cotidiana. Ya no se pueden prescindir de ellas. Y cada día se demanda más el hecho de tener que buscar soluciones alternativas al alto consumo de energía eléctrica.

La tecnología fotovoltaica que es la utilizada en este proyecto, consiste en la exposición de una celda solar, produciendo energía eléctrica. Aunque anteriormente este método era el más costoso para producir electricidad ha disminuido considerablemente sus costos en la última década, hasta llegar a convertirse en una inversión a largo plazo por el ahorro de energía que representa. El proceso de mejora ha enfatizado desarrollo de menos costo y más formas para producir celdas fotovoltaicas, las cuales implican películas de varios semiconductores que se aplican a la superficie de un material como el vidrio. Esto permite costos de fabricación significativamente reducidos, y la producción de unidades individuales mucho más grandes.

Para operar de manera efectiva, los paneles deben combinarse con otros equipos. Si el poder es requerido durante la noche, o cuando el sol no está brillante, se necesitan baterías. Y, si la energía es para ser suministrado a la red o a la corriente alterna local (AC), se necesitan inversores para convertir la potencia de corriente continua (CC) producida por paneles en corriente alterna. Los inversores son extremadamente importantes, y llegando a reflejar un creciente parte del costo total del sistema, las celdas mismas se vuelven menos costosas.

De la parte del inversor se muestra gráficamente más adelante.

Componentes

El panel solar que se utilizó es monocristalino de 30cm de altura por 18cm de ancho, brinda 18V como máximo en circunstancias ideales, esto es en contacto con la luz solar al mediodía. Brinda una potencia máxima de 50W.

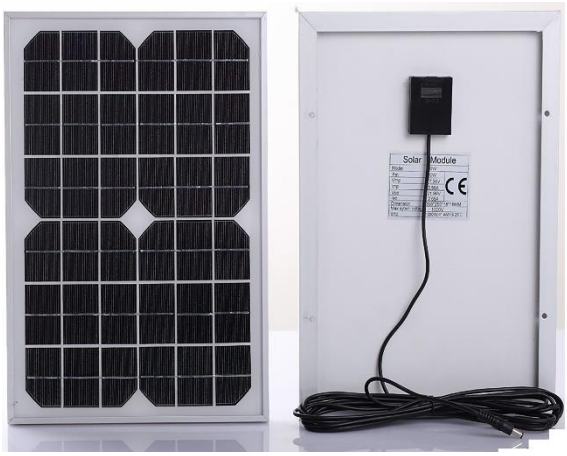


Ilustración 2: Panel Solar Utilizado.

Cálculos

Con un multímetro se realizaron las mediciones pertinentes del panel, tanto de voltaje siendo este 18V como máximo en un día soleado, y 8V cuando está expuesto a luz artificial, en una habitación iluminada, por ejemplo. Ya que el panel es utilizado para cargar una batería, las mediciones de voltaje son primordiales.

ALAMCENAMIENTO

Fundamentos

Es importante para el prototipo tener una forma de almacenamiento de energía, debido a que se enfoca al uso de energías limpias, se hace uso de una batería de ácido de 12V a 7A, debido a que estaba a nuestro alcance. El funcionamiento de este tipo de baterías está basado en la pila electroquímica. “Existen dos electrodos, uno positivo y otro negativo, que al conectarlos formando un circuito cerrado, generan una corriente eléctrica, es decir, los electrones fluyen de manera espontánea de un electrodo a otro. Las baterías están formadas por varios pares de electrodos que se sitúan en compartimentos independientes llamados celdas. En las celdas los electrodos están sumergidos en una disolución que recibe el nombre de electrolito.” (AutoSolar, 2015).

Reciben el nombre de baterías de plomo "ácido" porque utilizan como electrolito una disolución de ácido sulfúrico.

Si colocamos las celdas en serie, alternando positivo y negativo, podremos sumar las tensiones de cada una de ellas, y finalmente obtener un voltaje más alto (6 V, 12 V, 24 V, ...). Si, por el

contrario, colocamos las celdas en paralelo, positivos al lado de positivos, y negativos al lado de negativos, conseguiremos aumentar la intensidad de la batería. Este tipo de baterías manejan cuatro distintas etapas de carga las cuales son descritas a continuación. (AutoSolar, 2015)

1. Etapa Bulk:

En esta primera etapa se suministra corriente a la batería a intensidad máxima, de manera que el voltaje (tensión) aumenta rápidamente hasta llegar aproximadamente a 12,6 V, y después poco a poco hasta el primer límite de voltaje. Una vez alcanzado este límite la batería está cargada un 80-90%, a partir de este punto la absorción de corriente de carga se reduce rápidamente, estamos ahora a un potencial de 14,4-14,8 V según la batería. Si se desea cargar un banco de baterías el límite de voltaje se situaría entre un 10-20% de la intensidad nominal de la batería, es decir, entre 100-200 A para un banco de baterías de 1000 A/h. En esta etapa el regulador de carga que se sitúa entre el panel y el acumulador no juega ningún papel, pues la corriente se suministra a intensidad máxima, pero sin él la fase Bulk sería permanente y la corriente proveniente de los paneles solares podría destruir la batería por sobrecarga.

2. Etapa de Absorción:

En esta fase la corriente de carga disminuye lentamente hasta que la batería se carga al 100%. En esta etapa trabajamos al voltaje alcanzado al final de la etapa Bulk, denominado límite de absorción. Es importante conocer los valores de los voltajes utilizados con exactitud y siempre en conformidad a las indicaciones del fabricante. La finalidad de esta etapa es recuperar el electrolito, que puede haberse visto alterado en procesos de descarga profunda, así pues, en baterías que hayan sufrido una descarga profunda prolongada, la fase de absorción será más larga para asegurarnos de recuperar el electrolito por completo.

3. Etapa de Flotación:

en esta fase la batería ya está cargada al 100% y lo que se hace es proporcionar la corriente necesaria para compensar la auto-descarga, de manera que permanezca al 100%. Se trabaja a potenciales bajos y constantes. Si pretende almacenarse la batería el voltaje de flotación no puede variar más de un 1% respecto del recomendado por el fabricante. Para baterías liquidas se recomienda proporcionar voltajes entre 12,9-14 V, aunque no es recomendable la inutilidad de la batería durante periodos largos (meses). En cambio, las baterías de gel pueden ser dejadas en fase de flotación durante periodos largos sin problemas.

4. Etapa de Ecualización:

tiene como fin el ascenso del gas dentro del ácido (electrolito) haciendo que la disolución llegue a ser homogénea; por esto también se denomina etapa de gaseo. De esta forma evitamos que en la parte inferior no haya una densidad mayor que pueda provocar la sulfatación de las placas. Tras esta etapa conseguimos que todas las celdas tengan el mismo voltaje. El controlador puede

realizar esta etapa cada cierto periodo de tiempo, si se pretende hacer a mano conviene llevarla a cabo si se detecta disparidad de valores en la densidad del electrolito.

Componentes

Tabla 1: Componentes que integran el circuito

Componentes	Cantidad
Regulador LM317	1
TIP122	1
Diodo 1N4007	1
Capacitor 0.1uF/25V	1
Capacitor 220uF/25V	1
Resistencia Variable 4.5K	1
Resistencia 220 Ohms	1
Resistencia 1k	2
LED	1
Diodo de 5A	1
Diodo Zener 12V	1
Diodo Zener 15V	1
Capacitor 1000uf/25V	1

Diagrama

El diagrama del circuito implementado se muestra a continuación, se utiliza un regulador de voltaje LM317 programable, esto debido al control que se puede obtener no solo del voltaje sino también de la corriente; mediante un transistor con una resistencia en el borne de referencia para cuando la batería, limitando muy cercano a cero la corriente cuando el límite de voltaje superior se alcanza, al igual que se incluye un diodo LED el cual funciona como indicador cuando dicho voltaje del límite superior es alcanzado.

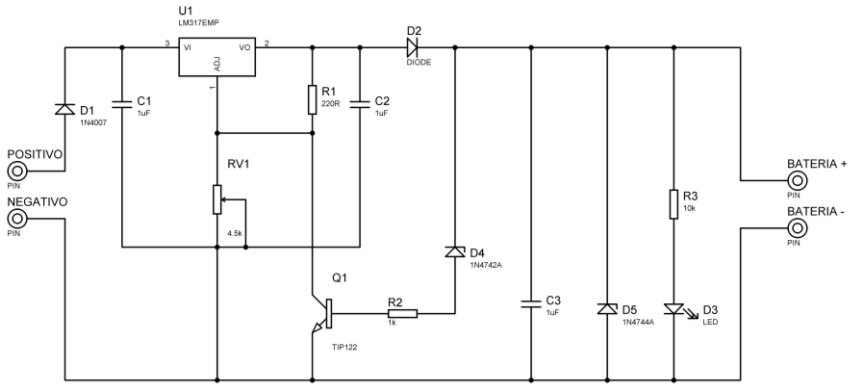


Ilustración 3: Diagrama eléctrico del almacenamiento de energía.

REGULACIÓN

Fundamentos

En el prototipo implementado se utilizan distintos voltajes de alimentación; 5 volts para alimentar el sistema embebido y 12 volts como salida de la batería, por ello se emplea un circuito de regulación para poder energizar el sistema embebido y los circuitos de manera adecuada.

Componentes

Tabla 2: Componentes que integral al circuito regulador.

Componentes	Cantidad
Capacitor 10 mF	1
Capacitor 0.1 mF	1
LM7805	1

Diagrama

El componente principal para realizar la regulación de voltaje es el LM7805 (Sparkfun, 2003), el cual disipa la energía restante, regulando a exactamente 5Volts de salida, los cuales alimentaran por medio de una conexión USB la tarjeta ESP32, el circuito implementado se muestra en la siguiente ilustración:

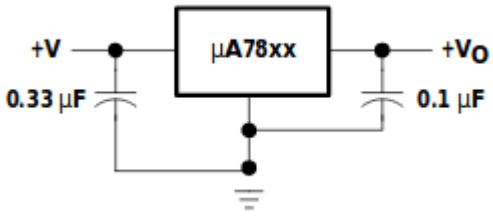


Ilustración 4: Diagrama electrónico del regulador de voltaje

Cálculos

Antes de realizar cualquier conexión con el regulador de voltaje, se verifica mediante multímetro que la salida del regulador con respecto al voltaje máximo de la batería, de forma que se puede comprobar que a la salida se tienen 5Volts.

DISEÑO DE LA APLICACIÓN

Para más versatilidad y completar los propósitos deseados, se implementó una aplicación móvil y así poder interactuar con el animatrónico.

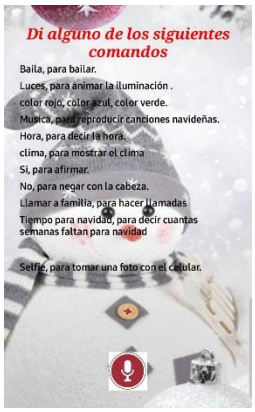


Ilustración 5:[Versión final de la aplicación móvil]

En la pantalla principal se muestra de forma visual y auditiva las funciones y comandos de voz que el animatrónico puede realizar, todo esto al inicializar la aplicación, de igual forma se encuentra el botón que activa los comandos por voz para interactuar con el animatrónico y con algunas funciones dentro aplicación. Su desarrollo y funciones se describen a continuación.

ENTORNO Y DESARROLLO

Enfoque (Android)

Se desarrolló la aplicación para el sistema operativo Android debido a la gran demanda que tiene en el mercado, por su gran facilidad de desarrollo y por qué según ¨StatCounter¨ durante el período comprendido entre enero de 2018 y enero de 2019, Android representó el 74,45 % del total del mercado, con iOS apenas un 22,85 %. (Casas, 2019)

Entorno de desarrollo

Para el entorno de desarrollo se utilizó el entorno de programación “MIT App Inventor” con licencia de uso “Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0)”. El cual es un entorno de programación visual e intuitivo que permite a todos, incluso a los niños, crear aplicaciones totalmente funcionales para teléfonos inteligentes y tabletas. (MIT, s.f.).

APK

Una vez desarrollada la aplicación en el entorno de “MIT App Inventor” se construye la aplicación para obtener su “APK” y poder transferirlo al celular y usar la aplicación.

INFORMACIÓN EXTRAÍDA DEL DISPOSITIVO

Para el desarrollo de algunas funciones de la aplicación se fue necesario tener acceso a cierta información del dispositivo, la cual es descrita a continuación y se exponen los fundamentos para su uso.

1. Micrófono del dispositivo para el reconocimiento de voz.
2. Contactos y llamadas del dispositivo para realizar llamadas por medio del comando de voz.
3. Reloj del dispositivo para obtener información de la hora y poder trasmitirla al animatrónico.
4. Comunicación WIFI para extraer información de servidores externos y comunicarnos como clientes del servidor al animatrónico
5. Coordenadas geográficas para poder consultar el clima correcto según la posición del usuario.

FUNCIONES DE LA APLICACIÓN Y COMANDOS DE VOZ

Las funciones que se pueden realizar en la aplicación móvil se pueden organizar en dos categorías.

1. Interactivas con la aplicación.
2. Interactivas con el animatrónico.

Interactivas con el animatrónico

En las funciones interactivas con el animatrónico encontramos todas aquellas funciones por comando de voz que ejecutan alguna acción con la animatrónico, las cuales son:

- Bailar: Al decir bailar y un numero puedes activar alguno de los bailes predeterminaos que están incluidos.
- Luces, Para activar la secuencia de luces de los ojos.
- Música, Para reproducir canciones navideñas.
- Si, para que el animatrónico asienta con la cabeza.
- No, para que el animatrónico niegue con la cabeza.

- Tiempo para navidad, para reproducir con audio precargado en el sistema las semanas que faltan para navidad.

Interactivas con la aplicación
Con los comandos de voz también se puede interactuar de forma interna para realizar funciones propias de la aplicación en las que el animatrónico no de involucrado como son:

- Selfie, para abrir la cámara frontal directamente y poder tomar una foto.
- Llamar a familia, para llamar a alguien de la lista de contactos.
- Clima, para mostrar el clima y reproducir con audio.
- Hora, para mostrar y reproducir con audio la hora.

CONEXIÓN WIFI
Para poder establecer conexión entre el animatrónico y la aplicación, se seleccionó la comunicación WIFI, para poder interactuar con el animatrónico desde cualquier lugar del hogar, así de esta forma los conocidos del usuario también puedan interactuar con el animatrónico y generar una vía de comunicación de entretenimiento entre el usuario y los conocidos del usuario dentro del hogar.

Fundamentos
La placa embebida “ESP32” cuenta con dos de entre las muchas tecnologías de comunicación inalámbrica, WIFI y Bluetooth, la decisión de tomar a WIFI como comunicación fue sobre las ventajas que tiene sobre el Bluetooth, como es su ancho de banda, y la distancia de comunicación posible.

Seguridad
Por motivos de seguridad y para que animatrónico no quede en control de un usuario no autorizado por medio de la aplicación móvil, se agregó un grado de seguridad al codificar las solicitudes al servidor con código JSON.

Codificación JSON
Para establecer comunicación, entre el servidor creado en la placa ¨ESP32¨ y la aplicación móvil de forma segura, se codificó la solicitud al servidor enviándolo por medio del formato JSON, de esta forma se resolvían dos problemas, tanto el de la seguridad como el del intercambio de información, como anteriormente se había mencionado, extraemos la hora del celular para enviarla al animatrónico y poder mostrarla, así como el numero de la secuencia de baile deseada.

Decodificación JSON-ASCII
De forma nativa al querer establecer comunicación, las solicitudes del cliente se mandan en forma de ¨URL¨ por lo que es necesario:

1. Retirar la información no requerida al llegar al servidor como es el http://IP/ del URL, por ejemplo~~: http://192.168.43.211/~~{"color":"1}

2. Los símbolos no alfanuméricos se envían en forma de código ASCII por lo que al recibirlos se tienen que decodificar del ASCII al símbolo dentro del servidor, por ejemplo:
    - Información enviada http://192.168.43.211/%7B"color":"1%7D
    - Información recibida y decodificada http://192.168.43.211/{"color":"1}
3. Obtener la información ligada a la solicitud, por ejemplo, en el caso recibir la instrucción http://192.168.43.211/{"color":"1}, establecer a la variable color, el entero 1. Por medio de la desencriptación del formato JSON.

Código de solicitud al servidor ¨MIT App Inventor¨

A continuación, se muestra un ejemplo del código desarrollado en el ¨MIT App inventor ¨ de la solicitud al servidor para establecer comunicación. Para más detalles ir a la sección de ANEXOS.

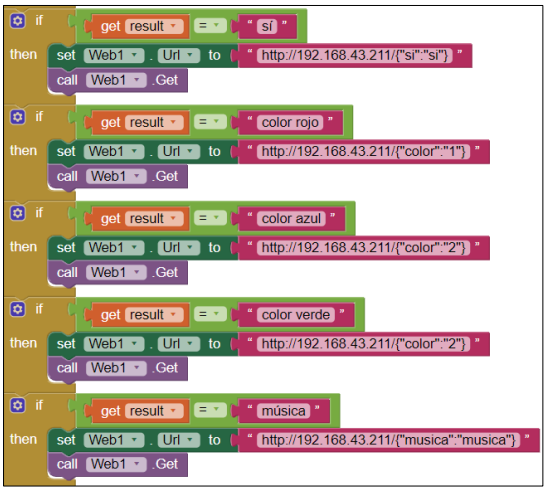


Ilustración 6: Fragmento del código de comunicación al servidor de la aplicación móvil.

Código de recepción de solicitud al servidor ¨Arduino IDE¨

A continuación, se muestra un fragmento de la decodificación de la comunicación con el servidor para interceptar la comunicación. Para más detalle ir a la sección de ANEXOS.

```
ClientRequest = (ReadIncomingRequest());
Serial.println(ClientRequest);
ClientRequest.remove(0, 5);
ClientRequest.remove(ClientRequest.length() - 9, 9);
String mensajeJSON = ClientRequest;
mensajeJSON.replace("%22", "\"");
mensajeJSON.replace("%7D", "}");
mensajeJSON.replace("%7B", "{");
Serial.println(mensajeJSON);
   StaticJsonBuffer<100> bufferJSON;
    //mensajeJSON es una variable String
    //bufferJSON es una variable string para la biblioteca ArduinoJSON
    //objetoJSON es una variable que si es valida, tiene el formato JSON
   JsonObject& objetoJSON = bufferJSON.parseObject(mensajeJSON);
   //Averiguar si el objeto JSON es valido
   if(objetoJSON.success()){
               String hora = objetoJSON["hora"];
               float clima = objetoJSON["clima"];
               String hola = objetoJSON["hola"];
               String luces = objetoJSON["luces"];
               String baila = objetoJSON["baila"];
               String no = objetoJSON["no"];
               String si = objetoJSON["si"];
               String musica = objetoJSON["musica"];
                String cuantofaltaparanavidad = objetoJSON["cuantofaltaparanavidad"];
               String selfie = objetoJSON["selfie"];
```

Ilustración 7: Fragmento del código de la decodificación de la solicitud de comunicación del cliente al servidor.

Adquisición de datos externos

Para poder adquirir la información sobre el clima se estableció comunicación con "https://openweathermap.org/" gracias a esta organización y su servicio gratuito de adquisición de datos sobre el clima a nivel mundial, fue posible obtener la información de clima por medio de su "API" por geolocalización, es por esto que como anteriormente se menciona, se requiere el permiso del usuario para obtener la posición geográfica del mismo por medio de su celular. De igual forma como en cualquier "API" al solicitar información al servidor este, la regresa en formato JSON y por medio de la aplicación ser puede, filtrar la información necesaria el hacer la decodificación de la información en formato JSON.

La información que el API ofrecía era la siguiente:

```
{"coord": { "lon": 139,"lat": 35},
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01n"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 289.92,
    "pressure": 1009,
    "humidity": 92,
    "temp_min": 288.71,
    "temp_max": 290.93
  },
  "wind": {
    "speed": 0.47,
    "deg": 107.538
  },
  "clouds": {
    "all": 2
  },
  "dt": 1560350192,
  "sys": {
    "type": 3,
    "id": 2019346,
    "message": 0.0065,
    "country": "JP",
    "sunrise": 1560281377,
    "sunset": 1560333478
  },
  "timezone": 32400,
  "id": 1851632,
  "name": "Shuzenji",
  "cod": 200
}
```


Ilustración 8: Ejemplo extraído de la página web oficial https://openweathermap.org/current.

Por lo que ahora se puede tener toda esta información a partir de la ubicación geográfica del usuario.

A continuación, se muestra la implementación por medio del entorno de desarrollo “MIT App inventor” para obtener esta información:

1. **Solicitar la información al “API” a aprtir de las coordeandas geograficas.**

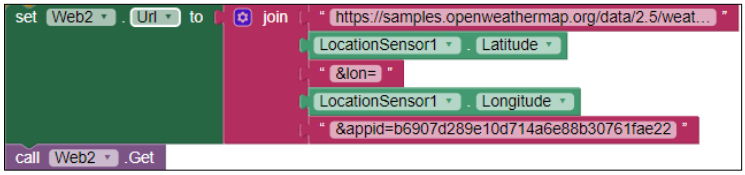


Ilustración 9: Solicitud de información.

2. **Decodificar JSON para la información que requerimos.**

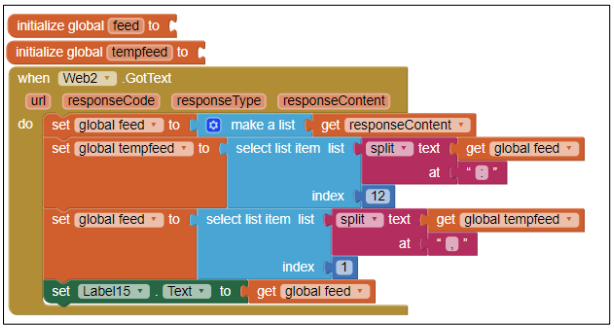


Ilustración 10:Decodificar Respuesta del API

3. **Transmitirla al animatrónico.**

Ilustración 11: Mandar los datos del clima al animatrónico

ACTUADORES

Para los propósitos de baile simples del animatrónico, se utilizaron 3 servos “9G SG90”, colocados en la siguiente configuración.

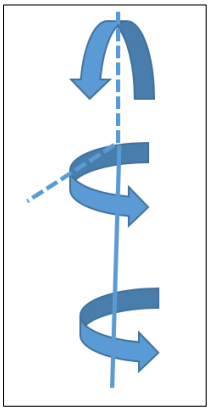


Ilustración 12:Configuración de movimiento de los servos.

Configuración en la ESP32

Para la correcta programación se utilizó la librería “#include <Servo.h>” y posteriormente para configurar los servos se les establecieron los siguientes 3 pines de salida.

```
static const int servoPin1 = 18;    ///BASE 0-90
static const int servoPin2 = 19;    ///0-110
static const int servoPin3 = 21;    ///
Servo servo1;
Servo servo2;
Servo servo3;
```

Ilustración 13: Configuración de los servos a pines de salida.

Programación de movimientos

Una vez teniendo la configuración y establecer comunicación con los servos, se puede programar para cualquier dirección de las permitidas, y así animar con movimientos al animatrónico, a continuación se muestra la programación de los movimientos del saludo, de la negación y de la afirmación. Para más detalle ir al final del documento al apartado de ANEXOS.

```
if (saludo == "saludo")
{
  ////////////////////////////////////////////////////////////////////////////////saludar
servo1.write(0);
delay(1000);
servo2.write(90);
delay(1000);

}
if (no == "no")
{
  ////////////////////////////////////////////////////////////////////////////////mover la cabeza
servo2.write(0);
delay(1000);
servo2.write(120);
delay(1000);
}
if (si == "si")
{
  ////////////////////////////////////////////////////////////////////////////////mover laa cabeza
servo1.write(0);
delay(1000);
servo1.write(120);
delay(1000);
```

Ilustración 14: Algunas configuraciones de movimiento.

ILUMINACIÓN

Para hacer más dinámico y visualmente estético al animatrónico y pensando en el público en general así mismo para el cumplimiento de los criterios de reto, se pusieron 2 leds “RGB” con control de iluminación como si fueran sus ojos.

Configuración

Se configuraron los 3 leds RGB para que se pudiera controlar el color de salida de cada led, los 2 leds están conectados en serie, por lo que lo que se manda en el pin de salía R, G, B provoca un cambio igual en los dos “Leds”.

```
const int redPin = 13;     // 13 corresponds to GPIO13
const int greenPin = 12;   // 12 corresponds to GPIO12
const int bluePin = 14;    // 14 corresponds to GPIO14
```

Ilustración 15: Configuración de pines de salida RGB

```
if (luces == "luces")
 {
     ledcWrite(redChannel, 0);
     ledcWrite(greenChannel,0);
     ledcWrite(blueChannel, 0);
     delay(500);
     ledcWrite(redChannel, 255);
     ledcWrite(greenChannel,255);
     ledcWrite(blueChannel, 255);
     delay(500);
     ledcWrite(redChannel, 0);
     ledcWrite(greenChannel,0);
     ledcWrite(blueChannel, 0);
     delay(500);
     ledcWrite(redChannel, 255);
     ledcWrite(greenChannel,255);
     ledcWrite(blueChannel, 255);
     delay(500);
     ledcWrite(redChannel, 100);
     ledcWrite(greenChannel,255);
     ledcWrite(blueChannel, 0);
     delay(500);
     ledcWrite(redChannel, 255);
     ledcWrite(greenChannel,100);
     ledcWrite(blueChannel, 0);
     delay(500);
```

Ilustración 16: Pequeña sección de una secuencia de luces

Secuencias

Se tienen diferentes animaciones programadas, una para cada ocasión entre ellas:

- Cunado de conecta por wifi a la aplicación, se ponen blancos los LEDS.
- Cunado saluda, Los LEDS hacen cambio de diferentes colores he intermitencia de prendido y apagado.
- Y para cada comando de voz con interacción con el animatrónico se genera una combinación diferente de secuencia y colores.

REPRODUCCIÓN DE AUDIO

El prototipo “Snowy” tiene dos modos de reproducción de audio, la primera local y la segunda en el dispositivo móvil, la primera por medio de la conversión de archivos de audio; bajando su resolución y cambiando los archivos a documentos de texto codificados en hexadecimal, lo cual será explicado más a fondo en el apartado “Trayectoria de archivo WAV a Hexadecimal”, dichos archivos son reproducidos por el DAC (conversor digital a análogo). Los audios fueron grabados un dispositivo móvil y guardados en librerías por cada uno de los mismos, de manera que se llamaron en el código cuando era necesario, algunos de los audios fueron las palabras: faltan, para, semanas, navidad, los números y algunas canciones navideñas. Al unir los archivos de audio se podía indicar cuantas semanas faltaban para navidad, lo cual fue una función añadida del prototipo.

Diagrama

A la salida del DAC se añade un capacitor para la eliminación del ruido, posteriormente se conecta al amplificador y a su vez a una bocina de 3W para su reproducción.

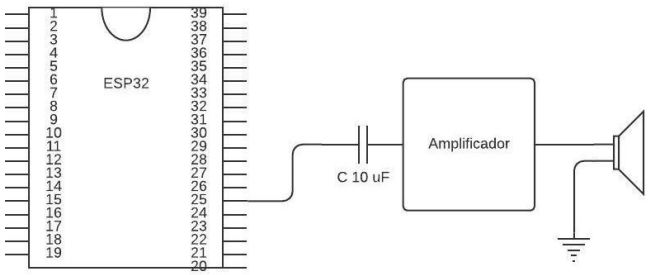


Ilustración 17:Diagrama de conexión del altavoz

CIRCUITO AMPLIFICADOR

Fundamentos

Es importante incluir un circuito amplificador debido a que el sistema embebido no es capaz de brindar la potencia necesaria para que la salida analógica pueda ser reproducida por la bocina,

para ello se considera la inclusión de un amplificador el cual tendrá una alimentación externa para no depender de la energía que brinda el sistema embebido.

Componentes

Componentes	Cantidad
Amplificador LM386	1
Resistencia variable 10K	1
Resistencia 10k	1
Capacitor 250 mF	1
Capacitor 0.05 mF	1

Diagrama

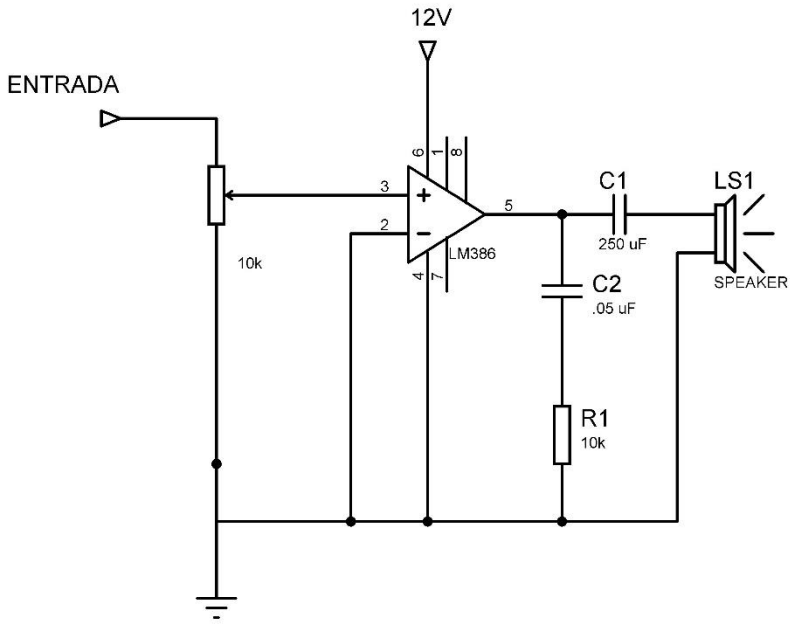


Ilustración 18: Diagrama del amplificador para el altavoz por la salida DAC de la ESP32.

TRAYECTORIA DE ARCHIVO WAV A HEXADECIMAL

Para poder reproducir el audio por el DAC se preparó el archivo para poder cargarlo a la memoria de la placa embebida y reproducirlo de forma correcta.

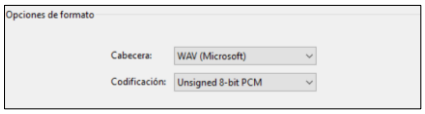
- Primero. Se edito el archivo de audio con especificaciones que se explicaran a continuación y se guardarlo en formato WAV.
- Segundo. Se crea una librería en donde en forma de funciones se guardan todos los sonidos para al querer ser reproducidos se mandan a llamar.
- Se programa la parte correspondiente de reproducción de audio.

Preparación de archivo de audio WAV (edición)

Características del audio

Una vez que ya se tienen los audios que se van a cargar a la memoria se prosigue a editarlos de la forma correcta, por medio del software ¨Audacity ¨ que es de uso gratuito.

1. Se cambia la frecuencia del audio a 8,000 Hz.
2. En caso de que el audio sea dual se monta para que quede en forma de mono audio.
3. Se Guarda como archivo WAV Unsigned 8-bit PCM.



Conversión a hexadecimal

Una vez que se tiene los archivos en el formato correcto se prosigue a obtener sus datos hexadecimales y la longitud.

Para esto se usa el comando: xxd -i archivo.wav archivoResultante.hex en Linux. (Weigert, 1997). Y muestra un resultado como este:

```
unsigned char sound_wav[] = {
  0x52, 0x49, 0x46, 0x46, 0xa0, 0x04, 0x00, 0x00, 0x57, 0x41, 0x56, 0x45,
  0x66, 0x6d, 0x74, 0x20, 0x32, 0x00, 0x00, 0x00, 0x02, 0x00, 0x02, 0x00,
  0x22, 0x56, 0x00, 0x00, 0x27, 0x57, 0x00, 0x00, 0x00, 0x04, 0x04, 0x00,
  0x20, 0x00, 0xf4, 0x03, 0x07, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x02,
  0x00, 0xff, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x40, 0x00, 0xf0, 0x00,
  0x00, 0x00, 0xcc, 0x01, 0x30, 0xff, 0x88, 0x01, 0x18, 0xff, 0x66, 0x61,
  0x63, 0x74, 0x04, 0x00, 0x00, 0x00, 0xa7, 0x02, 0x00, 0x00, 0x64, 0x61,
  0x74, 0x61, 0x00, 0x04, 0x00, 0x00, 0x02, 0x02, 0x28, 0x00, 0x37, 0x00,
  0x89, 0xff, 0x6f, 0x01, 0x3d, 0x01, 0x76, 0x00, 0x4e, 0x64, 0x05, 0xbf,
  0x1c, 0xc2, 0x0b, 0x70, 0x37, 0x80, 0x78, 0xe7, 0xfd, 0x50, 0xd4, 0xeb,
  0x2f, 0x22, 0x71, 0xf4, 0xed, 0xfd, 0x4e, 0x16, 0xf0, 0x0f, 0xd0, 0x4d,
  0x34, 0xc1, 0xfc, 0x20, 0x22, 0xf1, 0xcf, 0x0d, 0x20, 0x42, 0xe4, 0xed,
  0x2e, 0x02, 0x00, 0x21, 0x02, 0x2f, 0x92, 0x08, 0x00, 0xf0, 0x00, 0x00,

  0x34, 0x2e, 0x30, 0x00
};
unsigned int sound_wav_len = 1192;
```

Ilustración 19:Resultado del comando en Linux, longitud y archivo hexadecimal.

Para la creación adecuada de la librería con todo el sonido se aseguró que estuvieran en la forma que se muestra en la Ilustración 18. A continuación, se muestra un audio propio del sistema animatrónico.

```
const unsigned char navidad_wav[4684] = {
  0x52, 0x49, 0x46, 0x46, 0x44, 0x12, 0x00, 0x00, 0x57, 0x41, 0x56, 0x45,
  0x66, 0x6d, 0x74, 0x20, 0x10, 0x00, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00,
  0x40, 0x1f, 0x00, 0x00, 0x40, 0x1f, 0x00, 0x00, 0x01, 0x00, 0x08, 0x00,
  0x64, 0x61, 0x74, 0x61, 0x20, 0x12, 0x00, 0x00, 0x80, 0x80, 0x80, 0x81,
  0x81, 0x81, 0x81, 0x81, 0x81, 0x82, 0x82, 0x81, 0x81, 0x81, 0x81, 0x81,
  0x80, 0x7f, 0x7e, 0x7e, 0x7d, 0x7d, 0x7c, 0x7c, 0x7b, 0x7a, 0x7b, 0x7b,
  0x7a, 0x79, 0x78, 0x78, 0x79, 0x7a, 0x7c, 0x7d, 0x7e, 0x7e, 0x7f, 0x81,
  0x82, 0x83, 0x83, 0x83, 0x84, 0x85, 0x86, 0x86, 0x86, 0x87, 0x88, 0x88,
  0x88, 0x88, 0x88, 0x89, 0x88, 0x87, 0x85, 0x84, 0x83, 0x82, 0x81, 0x7e,
  0x7c, 0x7a, 0x78, 0x76, 0x73, 0x72, 0x71, 0x6f, 0x6e, 0x6e, 0x6d, 0x6d,
  0x6e, 0x70, 0x73, 0x76, 0x78, 0x7a, 0x7c, 0x7f, 0x83, 0x86, 0x88, 0x8b,
  0x8e, 0x8f, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x8f, 0x90, 0x91, 0x90,
  0x8f, 0x8e, 0x8c, 0x8b, 0x89, 0x86, 0x84, 0x81, 0x7f, 0x7c, 0x7a, 0x76,
  0x73, 0x6f, 0x6b, 0x68, 0x65, 0x64, 0x62, 0x62, 0x63, 0x64, 0x66, 0x68,
  0x6b, 0x6f, 0x74, 0x7a, 0x80, 0x87, 0x8d, 0x91, 0x94, 0x96, 0x98, 0x9a,
  0x9a, 0x99, 0x97, 0x95, 0x93, 0x91, 0x8e, 0x89, 0x86, 0x83, 0x82, 0x81,
  0x7f, 0x7d, 0x7d, 0x7d, 0x7d, 0x7d, 0x7b, 0x7a, 0x79, 0x79, 0x79, 0x79,
  0x79, 0x79, 0x79, 0x78, 0x76, 0x76, 0x77, 0x77, 0x78, 0x79, 0x7a, 0x7c,
  0x7d, 0x7f, 0x80, 0x81, 0x83, 0x84, 0x85, 0x86, 0x86, 0x87, 0x87, 0x86,
  0x85, 0x85, 0x84, 0x83, 0x82, 0x81, 0x80, 0x81, 0x80, 0x7e, 0x7c, 0x7b,
  0x7b, 0x7b, 0x7b, 0x7a, 0x7a, 0x7a, 0x7a, 0x7b, 0x7c, 0x7c, 0x7c, 0x7c,
```

Ilustración 20:Parte del archivo de audio de la canción navideña.

Donde se tiene que asegurar de que en el argumento de la función se escriba la longitud exacta de datos hexadecimales del audio.

PROGRAMACIÓN DE ARCHIVO DE AUDIO

Para la reproducción de los audios cargado en la librería de audio, se tuvo que implementar la librería #include "XT_DAC_Audio.h" la cual por medio de una función se creaban la variable para mandarla por el DAC y poder reproducirla en el altavoz.

Implementación

1. Clases que se crean para el funcionamiento adecuado de la reproducción de los archivos de audio:

```
XT_DAC_Audio_Class DacAudio(25,0);
XT_Wav_Class bellsD(bells_wav);       //b
XT_Wav_Class navidadD(navidad_wav);   //n
XT_Wav_Class faltanD(faltan_wav);     //f
XT_Wav_Class paraD(para_wav);         //p
XT_Wav_Class semanasD(semanas_wav);   //s
XT_Wav_Class unaD(una_wav);           //1
XT_Wav_Class dosD(dos_wav);           //2
XT_Wav_Class tresD(tres_wav);         //3
XT_Wav_Class cuatroD(cuatro_wav);     //4
XT_Wav_Class cincoD(cinco_wav);       //5
XT_Sequence_Class Sequence;
```

Ilustración 21:Clases para funcionamiento del audio

2. Se crea la función que creara las secuencias a partir de los audios convertidos a datos hexadecimales almacenados en la librería SoundData.h.

```
void PlayNumber(char const *Number)
{
  int NumChars=strlen(Number);
  Sequence.RemoveAllPlayItems();
  for(int i=0;i<NumChars;i++)
    AddNumberToSequence(Number[i]);
  DacAudio.Play(&Sequence);
  Serial.println(Number);
}
```

Ilustración 22: Función que reproduce los audios deseados.

3. Al iniciar el "Loop" se limpia el búfer con el comando "DacAudio.FillBuffer(); "
4. Se manda a llamar el audio de la lista de audio agregados a la secuencia del punto 2

```
PlayNumber("f");
```

CONCLUSION

En conclusión se pudieron verificar, probar e implementar, los circuitos requeridos por el concurso de proyectos modulares, para la creación del prototipo "Snowy", se incluyó una fuente de energía sustentable; panel solar, realizando la conversión de energía por medio de la circuitería mínima para alimentar el almacenamiento de la misma en una batería de ácido, se realizó un aplicación para Android con comunicación Wi-fi entre el animatrónico y la aplicación móvil, se manejó seguridad con encriptación JSON, se extrajeron datos de un servidor externo, se incluyeron funciones de control de iluminación para focos LED de tres canales (RGB), al igual que se tuvieron tres grados de movimiento por medio de dos servomotores y se incluyó reproducción de audio, todo ello enfocándose en las fortalezas de cada una de las carreras de los integrantes.

REFERENCIAS BIBLIOGRÁFICAS

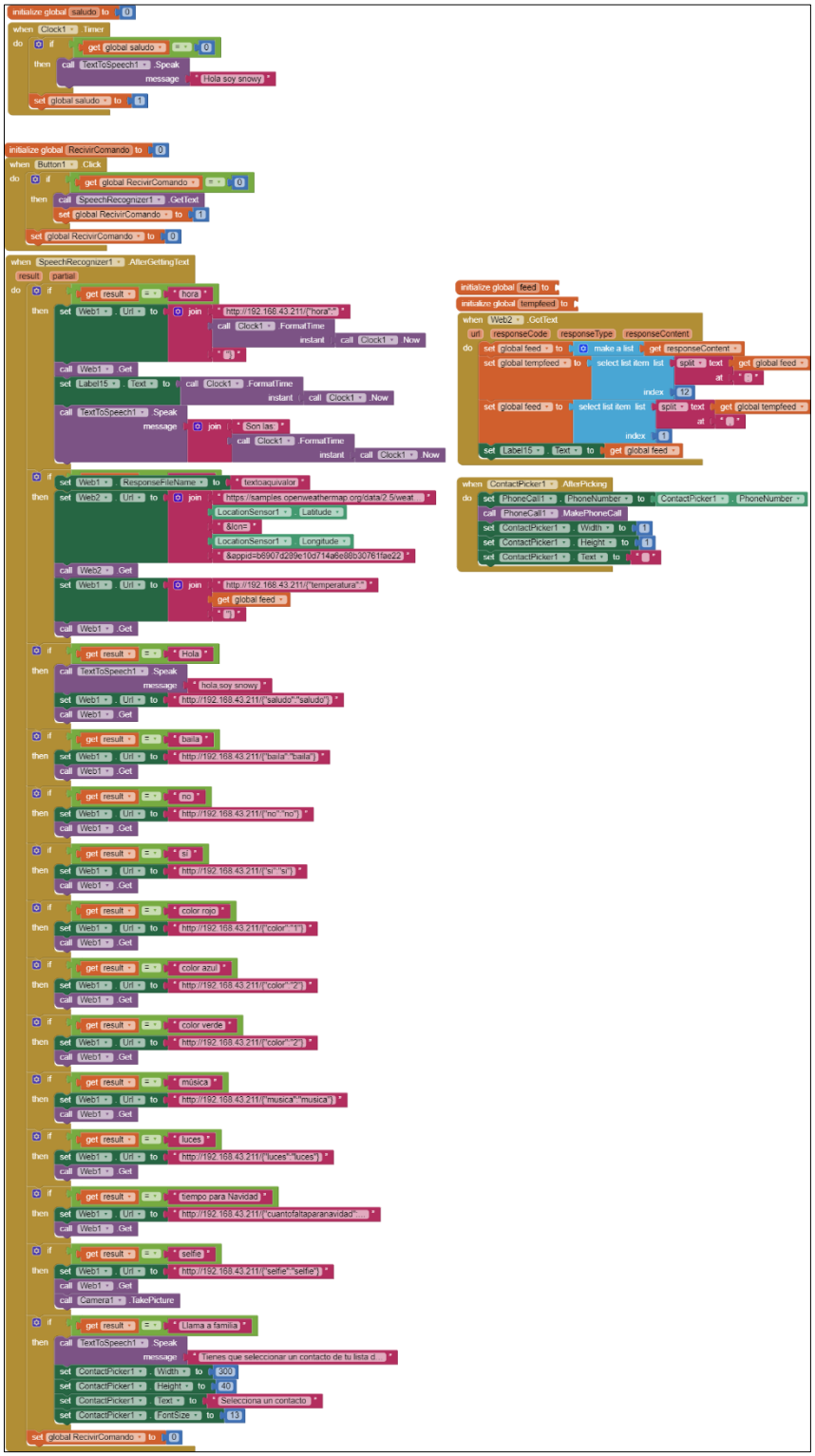
- *AutoSolar*. (19 de Mayo de 2015). Obtenido de Etapas de carga de una batería: https://autosolar.es/blog/aspectos-tecnicos/etapas-de-carga-de-una-bateria
- *AutoSolar*. (19 de Mayo de 2015). Obtenido de ¿Cómo funcionan las baterías de plomo ácido?: https://autosolar.es/blog/aspectos-tecnicos/como-funcionan-las-baterias-de-plomo-acido
- Barton, J. H. (2007). *Iprsonline.* Obtenido de Intellectual Property and Access to Clean Energy Technologies in Developing Countries: http://www.iprsonline.org/ictsd/docs/New%202009/CC%20Barton.pdf
- Casas, A. (25 de febrero de 2019). *iPhone vs Android: cuota de mercado.* Obtenido de PCWORLD DE IDG: https://www.pcworld.es/articulos/smartphones/iphone-vs-android-cuota-de-mercado-3692825/
- García, E. R. (6 de Agosto de 2017). *Omicrono* . Obtenido de La industria de la animatrónica y la Inteligencia Artificial, ¿un nuevo boom?:

https://www.elespanol.com/omicrono/tecnologia/20170806/industria-animatronica-inteligencia-artificial-nuevo-boom/236976893_0.html

- INEGI. (2010). *Discapasidad.* Guadalajara: INEGI.
- Intel. (2019). *Different Wi-Fi Protocols and Data Rates.* Intel .
- MIT. (s.f.). *MIT APP INVENTOR*. Obtenido de About Us: https://appinventor.mit.edu/explore/about-us.html
- Sparkfun. (2003). *POSITIVE-VOLTAGE REGULATORS LM7805.* Texas: Sparkfun.
- Systems, E. (2019). *ESP32 Series Datasheet.* Espressif Systems.
- Weigert, J. (1997). *TutorialsPoints*. Obtenido de xxd - Unix, Linux Command: https://www.tutorialspoint.com/unix_commands/xxd.htm

ANEXOS

Código de la aplicación en entorno “MIT App Inventor”



Código de la placa embebida ESP32 en entorno de desarrollo “Arduino IDE”

```
#include <Servo.h>
#include <WiFi.h>
#include "time.h"
#include<ArduinoJson.h>
#include "SoundData.h"
#include "XT_DAC_Audio.h"
#define ledVerde 23
// Red, green, and blue pins for PWM control
const int redPin = 13;     // 13 corresponds to GPIO13
const int greenPin = 12;   // 12 corresponds to GPIO12
const int bluePin = 14;    // 14 corresponds to GPIO14
////////////////////////////////////////////////
// Setting PWM frequency, channels and bit resolution
const int freq = 5000;
const int redChannel = 0;
const int greenChannel = 1;
const int blueChannel = 2;
// Bit resolution 2^8 = 256
const int resolution = 8;
String ClientRequest;
String mensajeJSON = "";
//////////////////////////////Audio Objetos
XT_DAC_Audio_Class DacAudio(25,0);
XT_Wav_Class bellsD(bells_wav);       //b
XT_Wav_Class navidadD(navidad_wav);   //n
XT_Wav_Class faltanD(faltan_wav);     //f
XT_Wav_Class paraD(para_wav);         //p
XT_Wav_Class semanasD(semanas_wav);   //s
XT_Wav_Class unaD(una_wav);           //1
XT_Wav_Class dosD(dos_wav);           //2
XT_Wav_Class tresD(tres_wav);         //3
XT_Wav_Class cuatroD(cuatro_wav);     //4
XT_Wav_Class cincoD(cinco_wav);       //5
XT_Sequence_Class Sequence;
//////////////////////////Imprimir la hora local//////////////////////
const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 3600;
const int   daylightOffset_sec = 3600;

void printLocalTime()
{
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
```

```
    return;
  }
//  int hora ;
//  hora = timeinfo.tm_hour;
//  byte horamodificada;
//  horamodificada = hora - 7;
//
  Serial.print(&timeinfo, "%A, %B %d %Y %H:%M:%S");
}
////////////////////////////////////////////////////////////////////////////////////////////////ESTABLECER IP
IPAddress staticIP(192, 168, 43, 211); //http://192.168.43.211
IPAddress gateway(192, 168, 3, 255);
IPAddress subnet(255, 255, 255, 0);
WiFiServer server(80);
WiFiClient client;
String myresultat;
////////////////////////////////////////////////////////////////////////////////////////////////VARIABLES DE
SERVOS
static const int servoPin1 = 18;    ///BASE 0-90
static const int servoPin2 = 19;    ///0-110
static const int servoPin3 = 21;    ///
Servo servo1;
Servo servo2;
Servo servo3;
////////////////////////////////////////////////////////////////////////////////////////////////FUNCIONES
String ReadIncomingRequest()
{
  while (client.available())
  {
    ClientRequest = (client.readStringUntil('\r'));
    if ((ClientRequest.indexOf("HTTP/1.1") > 0))
      myresultat = ClientRequest;
  }
  return myresultat;
}
void PlayNumber(char const *Number)
{
  int NumChars=strlen(Number);
  Sequence.RemoveAllPlayItems();
  for(int i=0;i<NumChars;i++)
    AddNumberToSequence(Number[i]);
  DacAudio.Play(&Sequence);
  Serial.println(Number);
}
void AddNumberToSequence(char TheNumber)
{
  switch(TheNumber)
```

```
  {
    case 'b' : Sequence.AddPlayItem(&bellsD);break;
    case 'n' : Sequence.AddPlayItem(&navidadD);break;
    case 'f' : Sequence.AddPlayItem(&faltanD);break;
    case 'p' : Sequence.AddPlayItem(&paraD);break;
    case 's' : Sequence.AddPlayItem(&semanasD);break;
    case '1' : Sequence.AddPlayItem(&unaD);break;
    case '2' : Sequence.AddPlayItem(&dosD);break;
    case '3' : Sequence.AddPlayItem(&tresD);break;
    case '4' : Sequence.AddPlayItem(&cuatroD);break;
    case '5' : Sequence.AddPlayItem(&cincoD);break;
  }
}
void setup()
{
  Serial.begin(115200);
  ClientRequest = "";
  // configure LED PWM functionalitites/////////////////////////////////////
  ledcSetup(redChannel, freq, resolution);
  ledcSetup(greenChannel, freq, resolution);
  ledcSetup(blueChannel, freq, resolution);
  // attach the channel to the GPIO to be controlled
  ledcAttachPin(redPin, redChannel);
  ledcAttachPin(greenPin, greenChannel);
  ledcAttachPin(bluePin, blueChannel);
  ////////////////////////////////////////////////////////////////////
  pinMode(ledVerde, OUTPUT);
  Serial.begin(115200);
  delay(10);
  mensajeJSON.reserve(100);
////////////////////////////////////////////////////////////////////////////////////////////////////CONECTAR A WIFI
  Serial.println("START");
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println("ATT_Internet_En_Casa_1937");
  WiFi.begin("virus", "123123123");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected");
//////////////////////init and get the time
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  printLocalTime();
 // WiFi.config(staticIP, gateway, subnet);
  Serial.println("Your IP is");
```

```
  Serial.println((WiFi.localIP()));
////////////////////////////////////////////////////////////////////////////////////////////////////INICIALIZAR
SERVIDOR
  server.begin();
////////////////////////////////////////////////////////////////////////////////////////////////////CONEXION DE
SERVOS
    servo1.attach(servoPin1);
    servo2.attach(servoPin2);
    servo3.attach(servoPin3);
}

void loop()
{
  DacAudio.FillBuffer();
  ledcWrite(redChannel, 255);
  ledcWrite(greenChannel,255);
  ledcWrite(blueChannel, 255);
////////////////////////////////////////////////////////////////////////////////////////////////////BUSCAR CLIENTE
EN EL SERVIDOR
  //obtém cliente
  client = server.available();
  if (!client)
    return;
  while (!client.available())
    delay(1);
////////////////////////////////////////////////////////////////////////////////////////////////////LEER URL
  ClientRequest = (ReadIncomingRequest());
  Serial.println(ClientRequest);
  ClientRequest.remove(0, 5);
  ClientRequest.remove(ClientRequest.length() - 9, 9);
  String mensajeJSON = ClientRequest;
  mensajeJSON.replace("%22", "\"");
  mensajeJSON.replace("%7D", "}");
  mensajeJSON.replace("%7B", "{");
  Serial.println(mensajeJSON);
     StaticJsonBuffer<100> bufferJSON;
      //mensajeJSON es una variable String
      //bufferJSON es una variable string para la biblioteca ArduinoJSON
      //objetoJSON es una variable que si es valida, tiene el formato JSON

     JsonObject& objetoJSON = bufferJSON.parseObject(mensajeJSON);

     //Averiguar si el objeto JSON es valido
     if(objetoJSON.success()){

                String hora = objetoJSON["hora"];
                float clima = objetoJSON["clima"];
```

```
    String saludo = objetoJSON["saludo"];
    String luces = objetoJSON["luces"];
    String baila = objetoJSON["baila"];
    String no = objetoJSON["no"];
    String si = objetoJSON["si"];
    String musica = objetoJSON["musica"];
    int color = objetoJSON["color"];
    String cuantofaltaparanavidad = objetoJSON["cuantofaltaparanavidad"];
    String selfie = objetoJSON["selfie"];
if (hora == "hora")
{
  //mostrar en lcd

}
if (selfie == "selfie")
{
  //girar360////////////////////////////////////////////////////////////////////////////////

}
if (clima >= 0 || clima <=0 )
{
  //mostrar en lcd
}
if (saludo == "saludo")
{
  ////////////////////////////////////////////////////////////////////////////////saludar
servo1.write(0);
delay(1000);
servo2.write(90);
delay(1000);
}
if (no == "no")
{
  ////////////////////////////////////////////////////////////////////////////////mover la cabeza
servo2.write(0);
delay(1000);
servo2.write(120);
delay(1000);
}
if (si == "si")
{
  ////////////////////////////////////////////////////////////////////////////////mover laa cabeza
servo1.write(0);
delay(1000);
servo1.write(120);
delay(1000);
}
```

```
if (luces == "luces")
 {
    ledcWrite(redChannel, 0);
    ledcWrite(greenChannel,0);
    ledcWrite(blueChannel, 0);
    delay(500);
    ledcWrite(redChannel, 255);
    ledcWrite(greenChannel,255);
    ledcWrite(blueChannel, 255);
    delay(500);
    ledcWrite(redChannel, 0);
    ledcWrite(greenChannel,0);
    ledcWrite(blueChannel, 0);
    delay(500);
    ledcWrite(redChannel, 255);
    ledcWrite(greenChannel,255);
    ledcWrite(blueChannel, 255);
    delay(500);
    ledcWrite(redChannel, 100);
    ledcWrite(greenChannel,255);
    ledcWrite(blueChannel, 0);
    delay(500);
    ledcWrite(redChannel, 255);
    ledcWrite(greenChannel,100);
    ledcWrite(blueChannel, 0);
    delay(500);
    ledcWrite(redChannel, 0);
    ledcWrite(greenChannel,255);
    ledcWrite(blueChannel, 100);
    delay(500);
    ledcWrite(redChannel, 50);
    ledcWrite(greenChannel,255);
    ledcWrite(blueChannel, 100);
    delay(500);
    ledcWrite(redChannel, 255);
    ledcWrite(greenChannel,255);
    ledcWrite(blueChannel, 255);
 }
 if (color <= 3 )
 {
  if(color==1){
    ledcWrite(redChannel, 255);
    ledcWrite(greenChannel,0);
    ledcWrite(blueChannel, 0);
    delay(1000);
    ledcWrite(redChannel, 255);
    ledcWrite(greenChannel,255);
```

```
            ledcWrite(blueChannel, 255);
          }else if(color ==2){
            ledcWrite(redChannel, 0);
            ledcWrite(greenChannel,255);
            ledcWrite(blueChannel, 0);
            delay(1000);
            ledcWrite(redChannel, 255);
            ledcWrite(greenChannel,255);
            ledcWrite(blueChannel, 255);
          }else if(color ==3){
            ledcWrite(redChannel, 0);
            ledcWrite(greenChannel,0);
            ledcWrite(blueChannel, 255);
            delay(1000);
            ledcWrite(redChannel, 255);
            ledcWrite(greenChannel,255);
            ledcWrite(blueChannel, 255);
            }
            else{
            ledcWrite(redChannel, 255);
            ledcWrite(greenChannel,255);
            ledcWrite(blueChannel, 255);
            }
        }
        if (musica == "musica")
        {
        PlayNumber("b");
        }
        if (cuantofaltaparanavidad== "cuantofaltaparanavidad")
        {
          PlayNumber("f");
          PlayNumber("4");
          PlayNumber("s");
          PlayNumber("p");
          PlayNumber("n");
        }
        if (baila == "baila")
        {
        }
}
  else
  {
  Serial.print("El comando no tiene el formato JSON");
  }
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("");
```

```
    client.println("");
    client.println("");
    client.println("OK");
    client.println("");
    client.flush();
    client.stop();
    delay(50);

  }
```

Fotografías del proceso y resultado final.