TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

REPORTE DE RESIDENCIA PROFESIONAL

INGENIERO EN ELECTRÓNICA

“SISTEMA DE REPRODUCCIÓN DE MÚSICA DE LAS CAMPANAS DE LA CATEDRAL DE SAN MARCOS TUXTLA GUTIÉRREZ”

PRESENTA

MARCELINO LÓPEZ HERNÁNDEZ

ASESOR

ING. ALVARO HERNÁNDEZ SOL

PERIODO DE REALIZACIÓN

AGOSTO-DICIEMBRE 2017

Tuxtla Gutiérrez, Chiapas.

ÍNDICE

CAPÍTULO I

INTRODUCCIÓN

1.1 ANTECEDENTES

Catedral de San Marcos.

El Edificio fundado a mediados del siglo XVI en homenaje a San Marcos, nombre al que también se le antepuso la aldea de Tuxhtla (tuxtla), habiendo quedado desde entonces como San Marcos Tuxhtla. Fue a partir de 1560 que se empezó a construir el templo dedicado a San Marcos Evangelista, santo que se representa simbólicamente con un “león alado”. Fundado como doctrina según refiere la tradición, por Gray Antonio de Pamplona, donde Tuxtla fue primera doctrina con sacerdote residente y después parroquia a mediados del siglo XVII.

Desde la construcción del templo para instruir a los indios zoques en la doctrina cristiana, el edificio ha sufrido varios cambios al transcurso de los siglos, el ábside es lo único que conserva de la época colonial. Le fue colocado en la torre del lado norte, el primer reloj público que tuvo Tuxtla en 1891. El 24 de julio de 1965 se erige la Diócesis de Tuxtla, la antigua parroquia adquirió el rango de Catedral de San Marcos, en (1982) siendo gobernador del Estado don Juan Sabines Gutiérrez fue cuando se construyó la torre campanario con reloj musical y se le coloca sobre el portón principal un león con alas, realizadas por el Arquitecto Ignacio Díaz Morales, Premio Nacional de Arte 1899, le dio su aspecto actual.

La gran Torre alberga un carrillón de fabricación alemana con 48 campanas que al cabo de cada hora tocan una canción para acompañar las figuras de los 12 apóstoles que desfilan sobre un carrusel. En 2004 se hizo la reparación y modernización del sistema eléctrico y electrónico del reloj de la catedral de San Marcos por residentes del Instituto Tecnológico de Tuxtla Gutiérrez, donde se le sustituyo la parte mecánica, del sistema de reproducción de música por un

digitalizador de canciones por medio de una PC, que contiene un software capaz de obtener un achivo fragmentado en acordes, códigos digitales de 48 bits (correspondiente a las 48 campanas) y 16 bits del tiempo entre acordes; este archivo se descarga en unas memorias regrabables por medio de un microcontrolador donde controla tanto su escritura como su reproducción. Mas adelante, también residente del instituto, en 2007 se modificó el sistema, usando iCatedral en etapa de software por la limitada armadura que soporta el sistema.

La torre campanario, ha sido importante para los ciudadanos tuxtlecos al igual que otras muchas poblaciones, su importancia desde mucho tiempo atrás. Su imagen para el municipio lo hace adquirir una representación única de la vida cotidiana para mayor influencia turística y beneficios socioeconómicos. Su importancia en la ciudad para la identidad de los tuxtlecos es única.

Figura 1.1. *Catedral de San Marcos, Tuxtla Gutiérrez.*

1.2 PLANTEAMIENTO DEL PROBLEMA

En las campanas de la catedral de San Marcos, el sistema actual elaborado desde hace 15 años, donde un circuito de microcontroladores recibe un archivo de 48 bits de una PC con software creada para dicho sistema, sin embargo, no se consiguen las partes fácilmente. A pesar de que sigue funcionando, ya es muy obsoleta, donde el circuito creado anteriormente no es capaz de almacenar más archivos, lo que causa que los ciudadanos estén escuchado la misma música todos los días, y puede llegar a fallar en cualquier momento, el cual se ve la necesidad de la actualización de un nuevo circuito, para la interfaz con la PC también software actualizada, donde el usuario pueda programar sus propias canciones.

1.3 DELIMITACIÓN DEL PROBLEMA

Se usara una tarjeta de arduino mega para la interfaz de la PC a las campanas, capaz de controlar la grabación de las canciones.

1.4 OBJETIVOS

1.4.1 General

Diseñar e implementación de un nuevo sistema electrónico, capaz de almacenar las canciones provenientes de la PC, y activar el sistema de reproducción en las campanas.

1.4.2 Específicos

- Comunicación serial entre iCatedral y el arduino.
- Poder reproducir los archivos de música.
- Guardar los archivos descargados desde iCatedral al SD.
- Poder instalar el arduino a la placa que abarca puertos de salida de la etapa de potencia.

1.5 JUSTIFICACIÓN

La razón por la que se requiere un nuevo hardware para el sistema de reproducción de música es, debido a que se fueron actualizando los sistemas operativos de Windows, usar el mismo sistema ya no es conveniente, debido a la antigüedad y tarde o temprano ya no se van a usar, aplicar el mismo programa para generar los archivos de las canciones es obsoleta para la tarjeta, ya que no cumple algunos requisitos, como la interfaz, el cableado, entre otras que se fueron revolucionado durante el transcurso de los años. Por otra parte el circuito electrónico diseñado anteriormente, únicamente puede guardar 16 canciones en la tarjeta de memorias EPROM y se requiere ampliarlo, para así programar más canciones, para beneficio de los ciudadanos que lleguen a escucharla.

Es aquello que se pretende actualizar a una nueva tarjeta de circuito electrónico, donde cumple con la descarga y reproducción de música.

CAPITULO II

FUNDAMENTO TEÓRICO

2.1 DECRIPCIÓN DEL FUNCIONAMIENTO DEL SISTEMA

A continuación se presenta una descripción del funcionamiento del sistema general con el subsistema electrónico:

- Alimentación
- Euroclock
- Órgano del carrillón
- Sistema electrónico

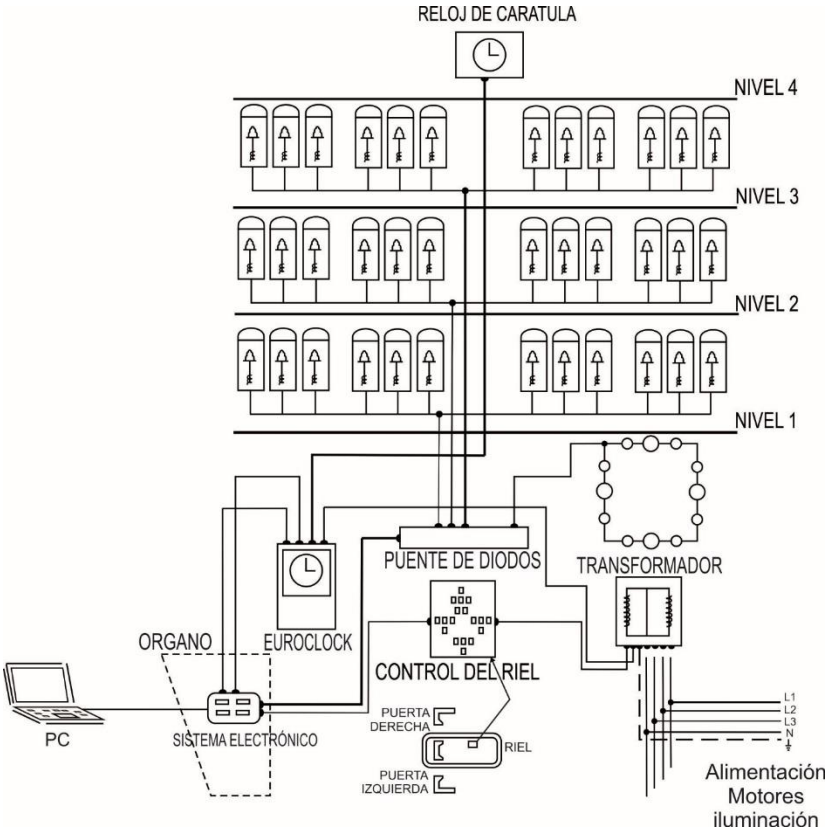


Figura 2.1 *Diagrama general del sistema.*

2.1.1 Alimentación

La alimentación general del sistema es trifásica, ya que posee dispositivos de este tipo, como motores y contactores.

Con el fin de aislar el sistema de control de los equipos de fuerza se utiliza un transformador; el cual se encarga de proporcionar alimentación a los subsistemas de control.

El diagrama de conexiones del transformador se observa en la figura 2.2.

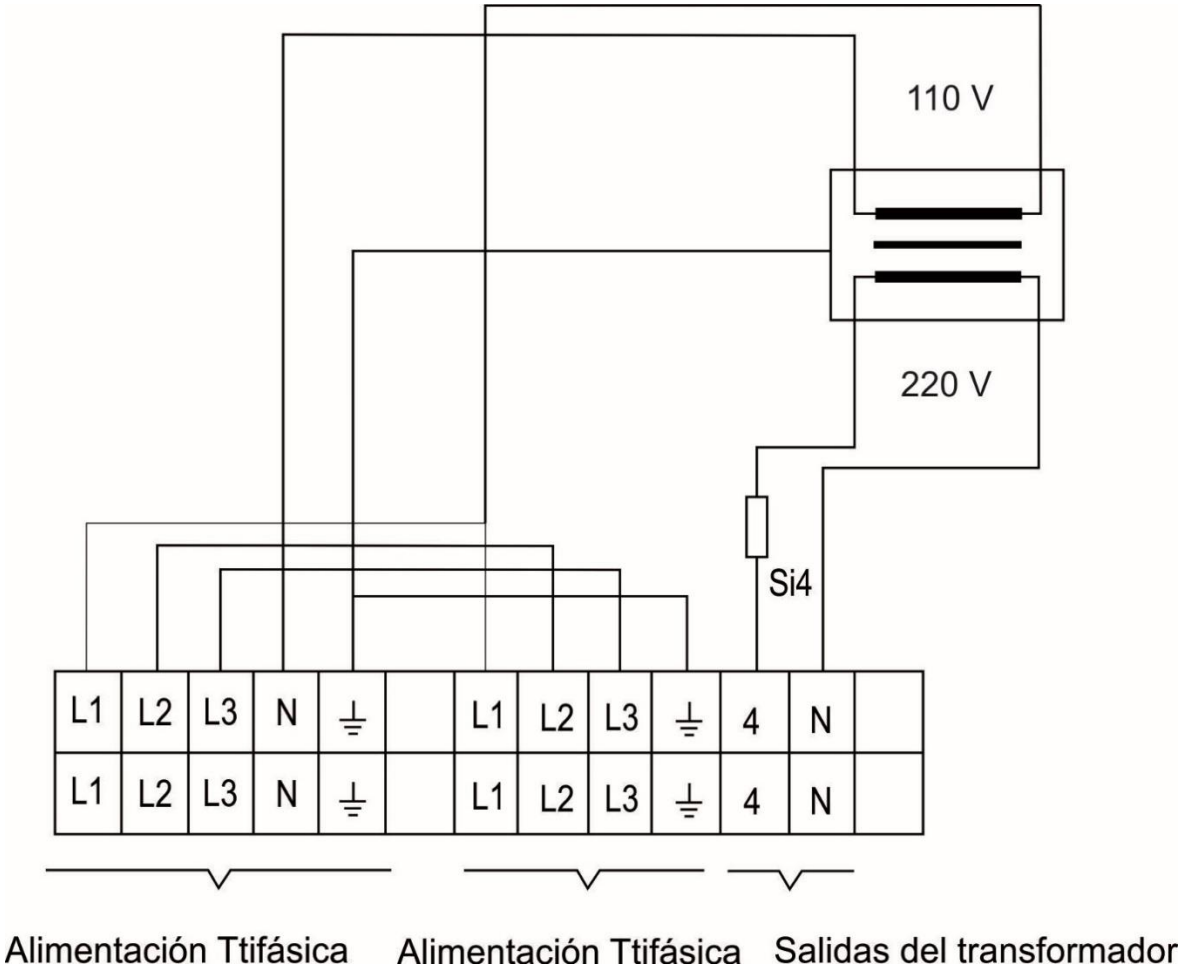


Figura. 2.2 *Diagrama de conexiones del transformador.*

Las características de este transformador son las siguientes:

- La relación de transformación es de 110/220.

- 5300/600 VA

- 100/85 % E.D.

El diagrama de conexiones de alimentación del sistema completo se muestra en la figura 2.3.

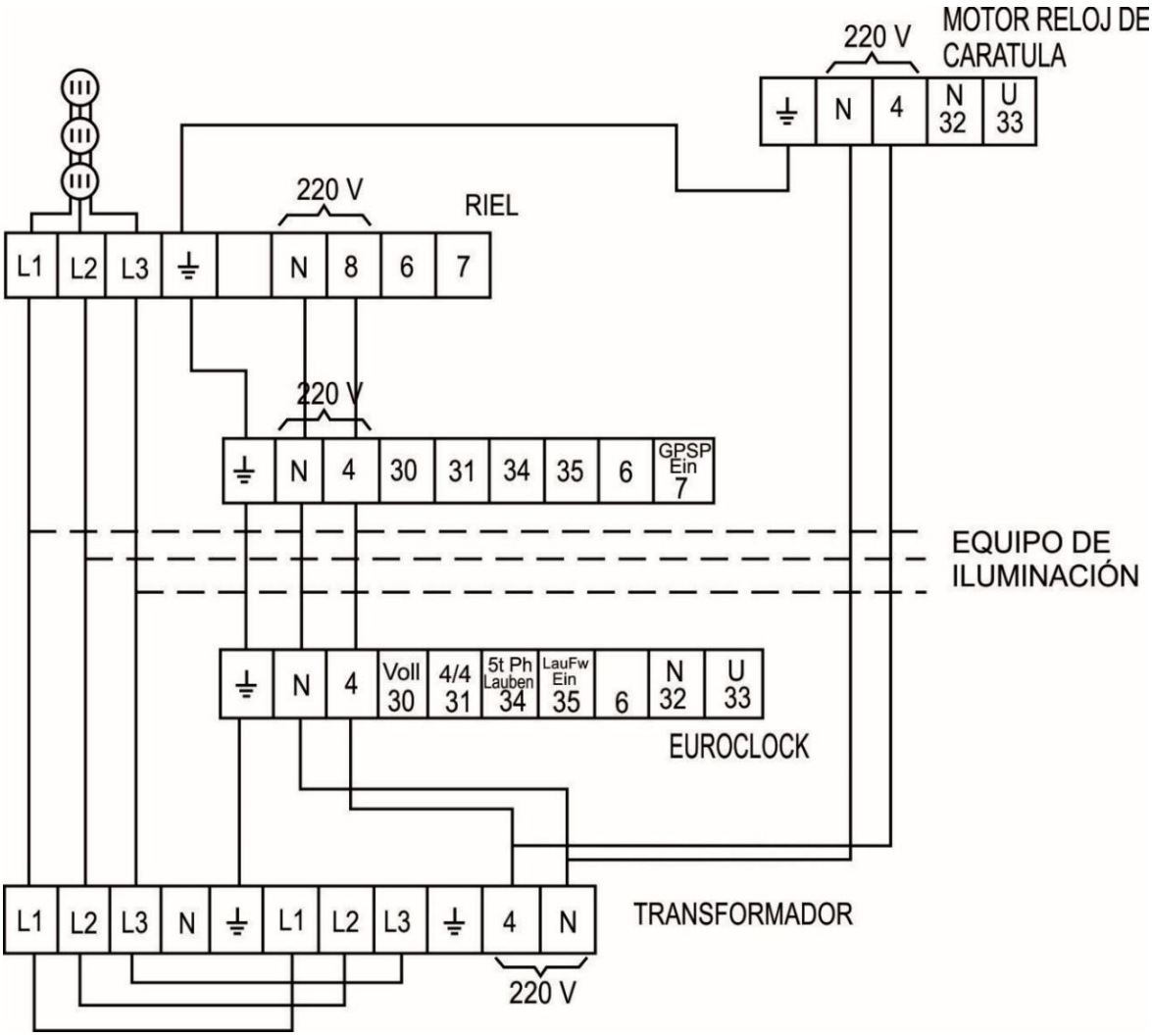


Figura. 2.3 *Diagrama de alimentación del sistema.*

El transformador proporciona la alimentación de los dispositivos de control tales como relees, temporizadores, bobinas, capacitores, etc., de cada uno los subsistemas.

La alimentación trifásica; que proviene del centro de carga ubicado en la planta baja de la torre; además de proporcionar la alimentación monofásica al transformador, alimenta directamente a través de contactores a los tres motores trifásicos encargados de abrir las puertas y mover el riel; lo mismo que al equipo de iluminación del sistema. Es por esto que se dice que el equipo de control está aislado del equipo de fuerza.

2.1.2 Euroclock

El euroclock es un reloj mecánico-electrónico que se encarga de controlar la secuencia correcta de activación y desactivación d los otros subsistemas, por lo que es el subsistema principal. Además de que es el reloj que entrega la señal a las 4 carátulas de la torre principal.

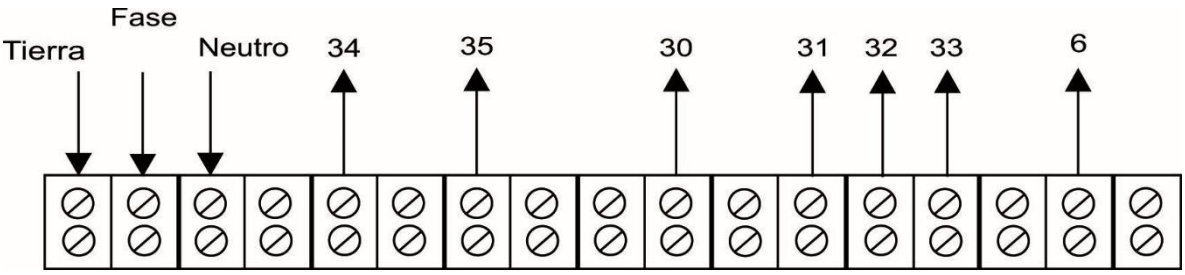


Figura. 2.4 *Barra de conexiones del Euroclock.*

La alimentación que recibe el euroclock proviene del transformador, que como puede observarse en la figura 2.3 ingresa en las 3 primeras terminales; tierra, fase y neutro.

Las salidas del Euroclock son las siguientes:

Señal 34.- La función de esta señal es activar el temporizador del órgano, que se encarga de activar las campanas de llamadas a misa. Esta señal se activa a las

6:30 am., 6:45 am., 7:00 am., 5:30 pm., 5:45 pm., 6:00 pm., 6:30 pm., 6:45 pm., 7:00 pm. Que corresponden a los llamados a misa de 7:00 a.m., 6:00 p.m. y 7:00 p.m.

Señal 35.- Esta señal activa directamente a una campana (la número 1, del primer nivel) la cual se activa justo antes de iniciar la misa, es decir a las 7:00 a.m., 6:00 p.m. y 7:00 p.m.

Señal 30.- Esta señal se activa cada hora en punto, y está conectada directamente a la campana que indica la hora que se ha cumplido. Es decir si se ha llegado las 5:00 en punto esta señal emitirá 5 pulsos, que harán sonar 5 veces la campana.

Esta señal se activa después de la señal 31.

Señal 31.- Esta señal está conectada a una campana que nos indica el transcurso de la hora; ya que cada cuarto de hora se activa y hace sonar una vez la campana; cada media hora se activa y veces y la hace sonar dos veces; dada cuarenta y cinco minutos la hace sonar 3 veces, y cada hora la hace sonar cuatro veces (esto indica que se ha llegado a una hora en punto); como se observa, esta salida se activa cada 15 min.

Señales 32 y 33.- Estas son las señales que se activan un micro switch que permite la activación del motor que mueve las manecillas del reloj de caráturla que se encuentra en la torre.

Señal 6.- Cada hora esta señal activa al selector del riel; el cual se encarga de llevar la secuencia de activación de los contactores que controlan a los motes de las puertas y el riel. Al mismo tiempo esta señal activa al Carrillón.

Esta señal se activa únicamente de 8:00 am a 12:00 pm.

2.1.3 Órgano del Carrillón

El órgano del carrillón es un mueble de madera tallada, que lleva en su parte delantera un teclado de piano con 48 notas musicales. Su cierre se efectúa por

medio de una cortina corrediza, también de la misma madera, el todo finamente lustrado.

Dentro del mueble está instalado el mecanismo con cuarenta y ocho contactos correspondientes al teclado, dicho mecanismo es un conjunto sólido, sencillo y de precisión. Su movimiento es impulsado por un motor eléctrico automáticamente regulado, el cual está fijado en el centro de un armazón mecánico especial, que soporta un sinfín y otros engranajes, en combinación con los portarrollos de cinta perforada para toque automático de las piezas musicales que se deseen, Para éstas, existe un dispositivo especial de funcionamiento que obedece a la presión de un contacto eléctrico automático desde el euroclock.

Figura 2.5. *Órgano del carillón.*

Conectado a los bornes correspondientes al teclado, se encuentran cuarenta y ocho partes de cables conductores, los cuales van a terminar cada uno, a los bornes de los cuarenta y ocho puentes de diodos que están fijados sobre el tablero de distribución.

El tablero de distribución con sus cuarenta y ocho puente de diodos de 220V a 4 A, consiste en una estructura de madera fijada a la pared, sobre el cual están todos los cables, tanto los de fase y neutro que recibe del órgano como los de CD que se dirigen a las campanas, prolijamente tendidos, fijados y conectados.

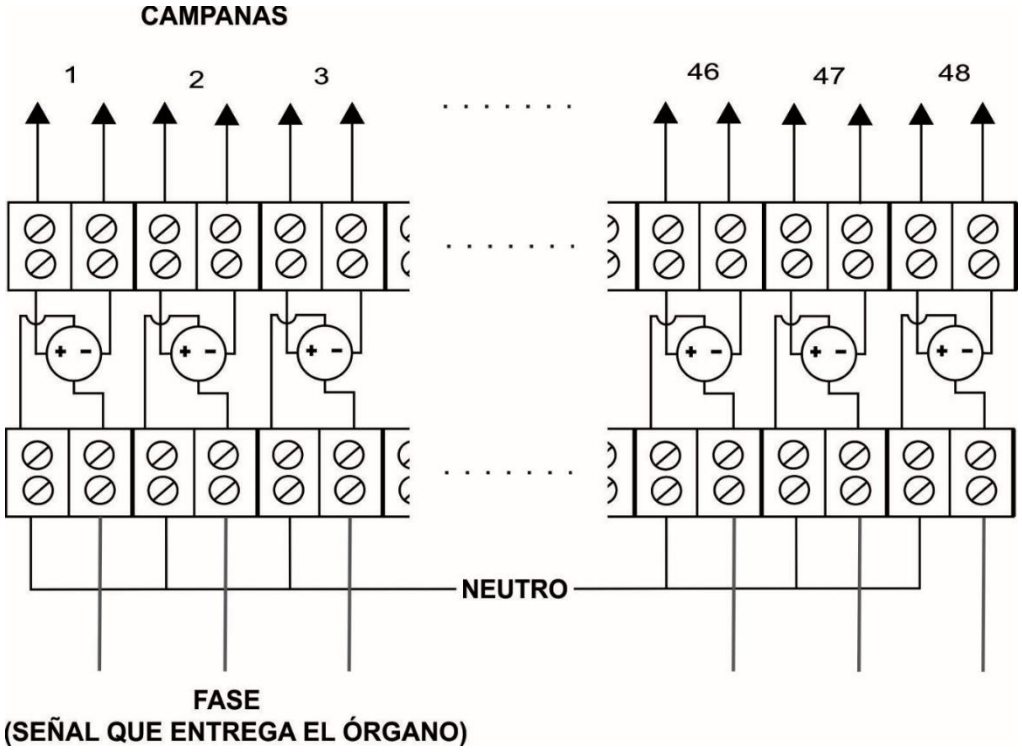


Figura. 2.6 *Diagrama de conexión del tablero de distribución.*

Del tablero salen 48 cables de tipo uso rudo, de alto aislamiento y llegan a los electroimanes de las campanas, los cuales excitados por la corriente de 220 VCD accionan los cuarenta y ocho martillos que golpean las campanas.

Los electroimanes que se encargan del movimiento de los martillos, son de 3 tamaños distintos, debido al tamaño de la campana, ya que para los martillos mas grandes necesitan de mayor corriente.

La estructura donde se encuentran las cuarenta y ocho campanas suspendidas, distribuidas en tres niveles según dimensiones y peso, es un armazón de hierro.

En el primer nivel se encuentran 24 campanas, y en los dos niveles superiores 12 campanas respectivamente.

2.1.4 Sistema electrónico

Dentro del órgano se encuentra es el sistema electrónico diseñado, donde se encuentra: fuente de 5V DC, memorias eeprom, interfase para la PC, convetidores serie/paralelo, etapa de potencia y microcontroladores.

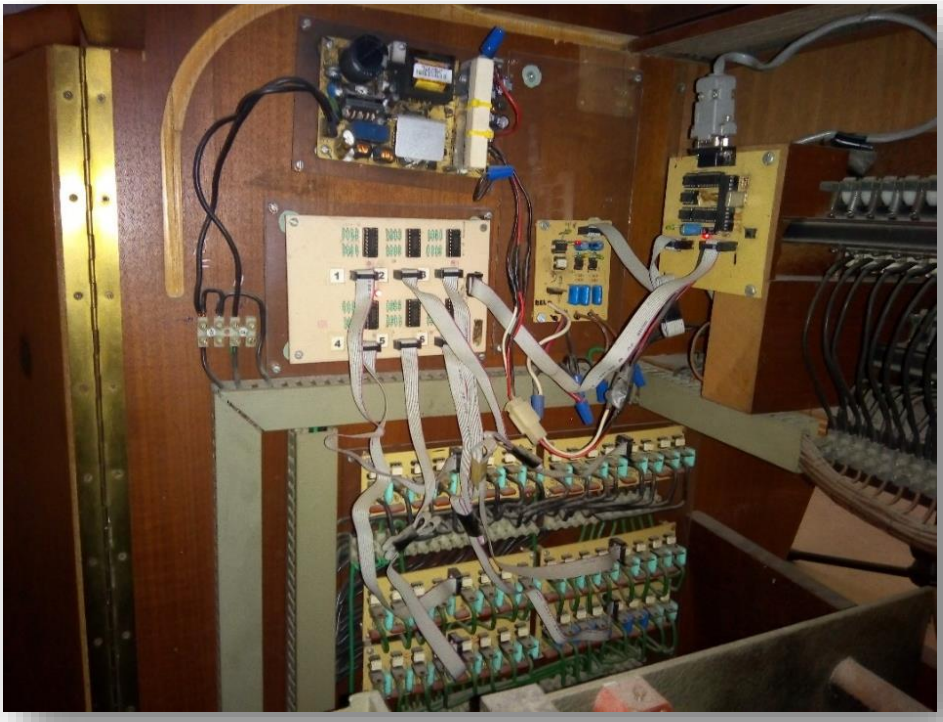


Figura 2.7. *Sistema electrónico.*

Las señales que provienen del euroclock y la señal que activa el riel, están acoplados en un circuito como se muestra en la Figura 2.9. Donde el microcontrolador pic12f675 que se encuentra en el mismo circuito, recepciona las señales #6 y #30 en el que envía uno lógico al microcontrolador pic16f873 que se encuentra en otra sección o en la figura 2.9.

Figura 2.8. *Señales de accionamiento.*

La interfaz del circuito con la PC consiste de un integrado MAX232, que la encontramos en la Figura 2.8 que convierte las señales lógicas del microcontrolador a un puerto serie RS-232 de tensión positiva y negativa utilizando una fuente de 5 voltios. En el mismo circuito se encuentra el microcontrolador pic16f873 que lleva a cabo todas las ejecuciones. También encontramos las memorias regrabables eeprom (24FC512), que tiene una capacidad de 64 Kbytes para cada memoria, en el que se guarda 4 canciones, un espacio de 16 Kbytes para cada canción.

Figura 2.9. *Interface de la PC.*

En la Figura 2.10. Encontramos el convertidor serie/paralelo usando el 74HC595 que están conectados en forma cascada donde se obtiene las 48 salidas para poder accionar las campanas a través de los módulos de la etapa de potencia.

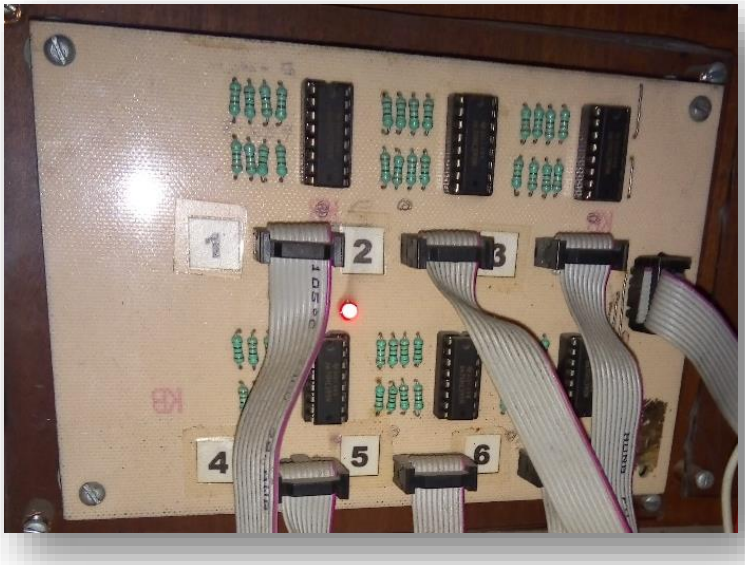


Figura 2.10. *Interface de la PC.*

Los módulos de la etapa de potencia son divididos en 6 secciones, con 8 salidas a cada circuito, donde se ocupan optoacopladores MOC3020 de entrada DC a AC, como se muestra en la figura 2.11. Que sirve para activar a cada una de las campanas, señales que provienen de los convertidores serie/paralelo, controlado por el microcontrolador pic16f873.

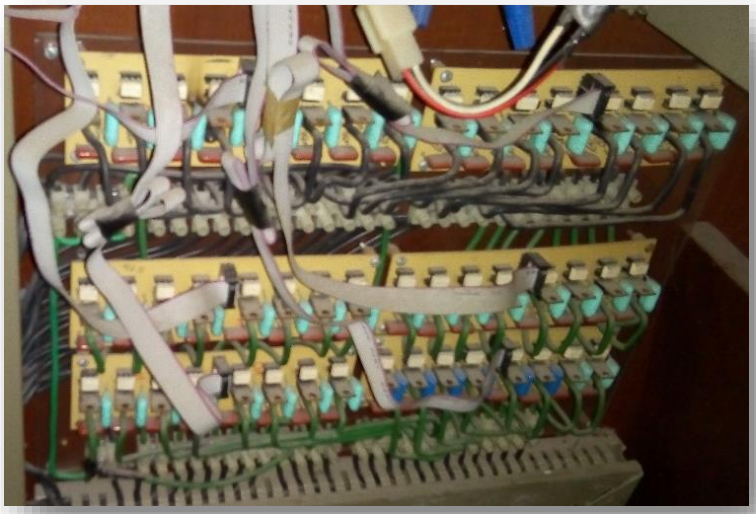


Figura 2.11. *Interface de la PC.*

2.2 ARDUINO

Arduino es una plataforma de electrónica de código abierto basado en hardware y software fácil de usar. Las placas Arduino pueden leer entradas (luz en un sensor, dedo en un botón o un mensaje de Twitter) y convertirlo en una salida, activar un motor, encender un LED y publicar algo en línea. Puede decirle a su tablero qué hacer enviarlo a un conjunto de instrucciones al microcontrolador en el tablero. Para hacerlo, utiliza el lenguaje de programación Arduino (basado en el cableado) y el software Arduino (IDE), basado en el procesamiento.

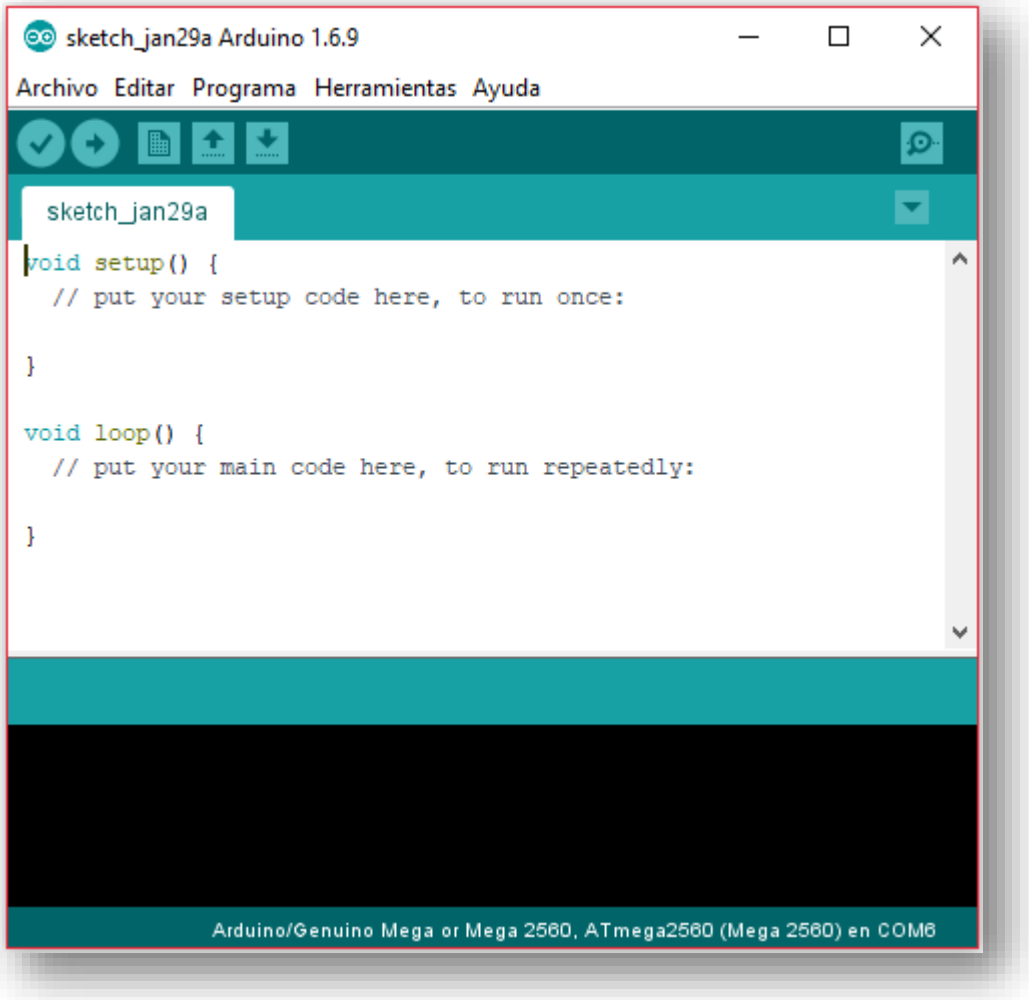


Figura 2.12. *Entorno de desarrollo de Arduino.*

2.2.1 Comunicación serie con la placa Arduino

Las instrucciones existentes dentro de un objeto (no todas las instrucciones de lenguaje Arduino pertenecen a un objeto) se escriben siguiendo la sintaxis *nombreObjeto.nombreIntruccion();.*

A continuación se explica la sintaxis, funcionamiento y utilidad de las instrucciones incluidas en el objeto “serial” ya que esto es fundamental para poder programar la descarga de archivos a través de la serial “buffer”. Empezando por una conocida:

Serial.begin(): abre el canal serie que pueda empezar la comunicación por él. Por tanto, su ejecución es imprescindible antes de realizar cualquier transmisión por dicho canal. Por eso normalmente se suele escribir dentro de la sección “void setup()”. Además, mediante su único parámetro (de tipo “long” y obligatorio), especifica la velocidad en bits/s a la que se producirá la transferencia serie de los dato. Para la comunicación con un computador, se suele utilizar el valor de 9600, pero se puede especificar cualquier otra velocidad. Lo que sí es importante es que el valor escrito como parámetro coincida con el que se especifique en el desplegable que aparece en el “Serial Monitor” del IDE de Arduino, o si no la comunicación no estará bien sincronizada y se mostraran símbolo sin sentido. Esta instrucción no tiene valor de retorno.

2.2.2 Instrucciones para enviar datos desde la placa al exterior

Serial.print(): envía a través del canal serie un dato (especificado como parámetro) desde el microcontrolador hacia el exterior. Ese dato puede ser cualquier tipo: carácter, cadena, número entero, número decimal por defecto de dos decimales), etc. Si el dato se especifica explícitamente (en vez de a través de una variable), hay que recordar que los caracteres se han de escribir entre comillas simples y las cadenas entre comillas dobles.

En el caso de que el dato a enviar sea entero, se puede especificar un segundo parámetro opcional que puede valer alguna constante predefinida de las siguientes: BIN, HEX o DEC. En el primer caso se enviará la representación binaria del número, en el segundo, la representación hexadecimal, y en el tercero, la representación decimal (la usada por defecto).

En el caso de que el dato a enviar sea decimal, se puede especificar además un segundo parámetro opcional que indique el número de decimales que se desea utilizar (por defecto son dos).

Su valor de retorno es un dato de tipo “byte” que vale el número de bytes enviados. En el caso de cadenas de caracteres, este valor de retorno coincide con el número

de caracteres enviados. El uso o no en nuestro sketch d este valor devuelto dependerá de nuestras necesidades.

La transmisión de datos ser realiza por Serial.print() es asíncrona. Eso significa que nuestro sketch pasará a la siguiente instrucción y seguirá ejecutándose sin esperar a que empiece a realizarse el envío de los datos. Si este comportamiento no es el deseado, se puede añadir justo después de Serial.print() la instrucción **Serial.flush()** (que no tiene ningún parámetro ni devuelve ningún valor de retorno), instrucción que espera hasta que la transmisión de los datos sea completa para continuar la ejecución del sketch.

Serial.println(): hace exactamente lo mismo que Serial.print(), pero además añade automáticamente al final de datos enviados dos caracteres extra: el de rotorno de carro (código ASCII n° 13) y el de nueva línea (código ASCII n° 10). La consecuencia es que al final de la ejecución de Serial.println() se efectúa un salto de línea. Tiene los mismos parámetros y los mismo valores de retorno que Serial.print().

Serial.write(): envía a través del canal serie un dato (especificado como parámetro) desde el microcontrolador hacia el exterior. Pero a diferencia de Serial.print(), el dato a enviar solo puede ocupar un byte. Por lo tanto, ha de ser básicamente de tipo “car” o “byte”. En realidad, también es capaz de transmitir cadenas de caracteres porque las trata como una mera secuencia de bytes independientes uno tras otro. En cambio, otros tipos de datos que ocupen más de un byte indisolublemente (como los “int”, “Word”, “float”…) no serán enviados correctamente. Su valor de retorno es, igual que en Serial.print(), un dato de tipo “byte” que vale el número de bytes enviados.

2.2.3 Instrucciones para recibir datos desde el exterior

Desde el “Serial monitor” enviar datos a la placa tenemos las siguientes instrucciones:

Serial.available(): devuelve el número de bytes (caractes) disponibles para ser leídos que provienen del exterior a través del canal serie (vía USB o vía pines TX/RX). Estos bytes ya han llegado al microcontrolador y permanecen almacenados temporalmente en una pequeña memoria de 64 bytes que tiene el chip TTL-UART (llamada “buffer”) hasta que sean procesados mediante la instrucción Serial.read(). Si no hay bytes para leer, esta instrucción devolverá 0. No tiene parámetros.

Serial.read(): devuelve el primer byte aún no leído de los que estén almacenado en el buffer de entrada del chip TTL-UART. Al hacerlo, lo elimina de ese buffer. Para devolver (leer) el siguiente byte, se ha de volver a ejecutar Serial.read(). Y hacer así hasta que se hayan leído todos. Cuando no haya más bytes disponibles, Serial.read() devolverá -1. No tiene parámentros.

Existen otras instrucciones además de Serial.read() que leen daos del buffer de entrada del chip TTL-UART de formas más específicas, las cuales nos pueden venir bien en determinadas circunstancias:

Serial.peek(): devuelve el primer byte aún no leído de los que estén almacenados en el buffer de entrada. No obstante, a diferencia de Serial.read(), ese byte leído no ser borra del buffer, con lo que las próximas veces que se ejecute Serial.peek() (o una vez Serial.read()) se volverá a leer el mismo byte. Si no hay bytes disponibles para leer, Serial.peek() devolverá -1. Esta instrucción no tiene parámetros.

Serial.find(): lee datos del buffer de entrada (eliminándolos de allí) hasta que se encuentre la cadena de caracteres (o un carácter individual) especificada como parámetro, o bien se haya leído todos los datos actualmente en el buffer. La instrucción devuelve “true” si se encuentra la cadena o “false” si no.

Serial.findUntil(): lee datos del buffer de entrada (eliminándolos de allí) hasta que se encuentre la cadena de caracteres (o un carácter individual) especificada como primer parámetro, o bien se llegue a una marca de final de búsqueda (la cual es la cadena o carácter especificada como segundo parámetro. La instrucción devuelve

“true” si se encutra la cadena a buscar antes de la marca de final de búsqueda o “false” si no.

Serial.readBytes(): lee del buffer de entrada (eliminándolos de allí) la cantidad de bytes especificada como segundo parámetro (o bien, si no llegan suficientes bytes, hasta que se haya superado el tiempo especificado por Serial.setTimeout()). En cualquier caso, los bytes leídos se almacenan en un array (de tipo “char[]”) especificado como primer parámetro. Esta instrucción devuelve el número de bytes leídos del buffer (por lo que un valor 0 significa que no se encontraron datos válidos).

Serial.readBytesUntil(): lee del buffer de entrada (eliminándolos de allí) la cantidad de bytes especificada como tercer parámetro, o bien, si se encuentra antes una cadena de caracteres (o carácter individual) especificada como primer parámetro que hace de marca de final, o bien, si no llegan suficientes bytes ni se encuentra la marca final, hasta que se haya superado el tiempo especificado por Serial.setTiemeout(). En cualquier caso, los bytes leídos se almacenarán en un array (de tipo “char[]”) especificado como segundo parámetro. Esta instrucción devuelve el número de bytes leídos del buffer (por lo que un valor 0 significa que no se encontraron datos válidos).

Serial.setTimeout(): tiene un parámetro (de tipo “long”) que sirve para establecer el número de milisegundos máximo que las instrucciones Serial.readBytesUntil() y Serial.readBytes() esperarán a la llegada de datos al búfer de entrada serie. Si alguna de estas dos instrucciones no recibe ningún dato y se supera ese tiempo, el sketch continuara su ejecución en la línea siguiente. El tiempo de espera por defecto es de 1000 milisegundos. Esta instrucción se suele escribir en “void setup ()”. No tiene valor de retorno.

Serial.parseFloat(): lee del buffer de entrada (eliminándolos de allí) todso los datos hasta que se encuentre con un número decimal. Su valor de retorno (de tipo “long”) será entonces ese número decimal encontrado. Cuando detecte el primer carácter

posterior no válido, dejará de leer (y por tanto, no seguirá eliminando datos del buffer). Esta instrucción no tiene parámetros.

Serial.parseInt(): lee del buffer de entrada (eliminándolos de allí) todos los datos hasta que se encuentre con un número entero. Su valor de retorno (de tipo “long”) será entonces ese número entero encontrado. Cuando detecte el primer carácter posterior no válido, dejará de leer (y por tanto, no seguirá eliminando datos del buffer). Esta instrucción no tiene parámetros.

2.2.4 Objetivo serie de la placa Arduino Mega

La placa Arduino Mega dispone de cuatro chips TTL-UART. Esto significa que podemos utilizar hasta cuatro objetos serie, llamados “Serial”, “Serial1”, “Serial2” y “Serial3”. El primero sigue estando asociado a los pines 0 y 1, el objeto “Serial1” está asociado a los pines 18 (TX) y 19 (RX), el “Serial2” a los pines 16 (TX) y 17 (RX) y el “Serial3” a los pines 14 (TX) y 15 (RX). Cada uno de estos objetos se puede abrir independientemente (escribiendo Serial.begin(9600); Serial1.begin(9600); Serial2.begin(9600); o Serial3.begin(9600); respectivamente), y pueden enviar y recibir datos también independientemente. No obstante, el único objeto asociado también a la conexión USB es “Serial” (porque es el único conectado al chip conversor ATmega16U2).

2.2.5 Librería SD

Permite leer y escribir datos en una tarjeta SD (o microSD) acoplada a un zócalo de algún shield o módulo específico. Las tarjetas SD son muy útilies para almacenar ficheros tales como audio, vídeo o imágenes, o bien datos textuales obtenido de diferentes sensores, ya que ofrecen un sistema de grabación mucho mayor que la memoria EEPROM.

2.3 ARDUINO MEGA 2560

Placa basada en el microcontrolador ATmega2560. Como características más destacables diremos que tiene 54 pines de entrada/salida digitales (de los cuales 14 pueden ser usados como salidas analógicas PWM), 16 entradas analógicas y 4 receptores/transmisores serie TT-UART. Consta de una memoria Flash de 256 Kilobytes (de los cuales 8 están reservados para el bootloader), una memoria SRAM de 8 KB y una EEPROM de 4 KB. Su voltaje de trabajo es igual al del modelo UNO: 5 V.

Arduino Mega es una tarjeta de desarrollo open-source construida con un microcontrolador modelo Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales. Esta tarjeta es programada en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede comunicarse a un PC a través del puerto serial (conversión con USB) utilizando lenguajes como Flash, Processing, MaxMSP, etc. Las posibilidades de realizar desarrollos basados en Arduino tienen como límite la imaginación.

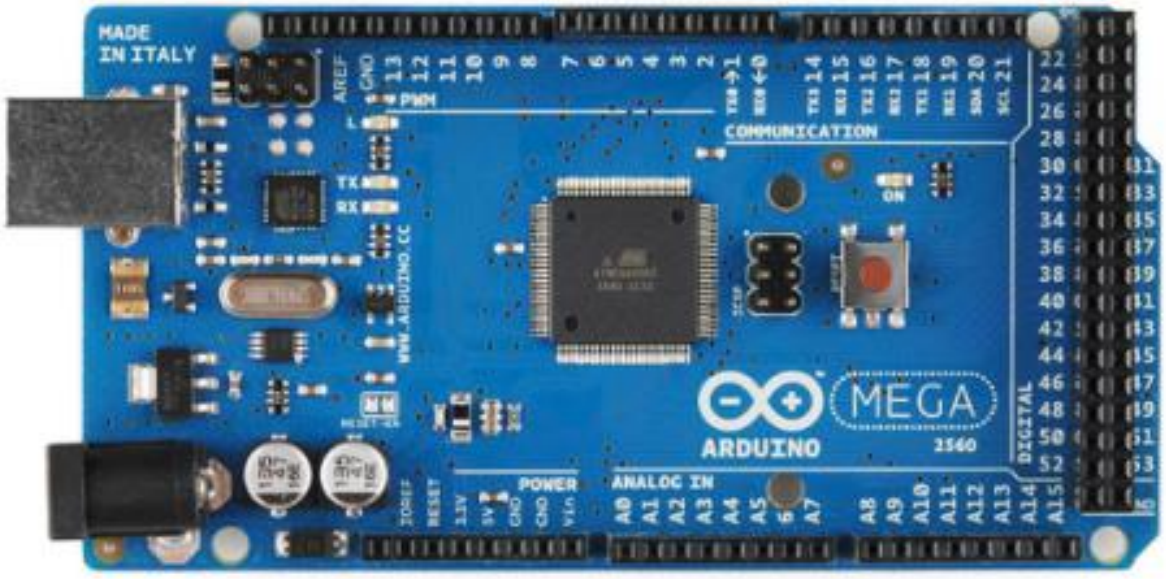


Figura 2.13. *Arduino Mega 2560*.

El Arduino Mega tiene 54 pines de entradas/salidas digitales (14 de las cuales pueden ser utilizadas como salidas PWM), 16 entradas análogas, 4 UARTs (puertos serial por hardware), cristal oscilador de 16MHz, conexión USB, jack de alimentación, conector ICSP y botón de reset. Arduino Mega incorpora todo lo necesario para que el microcontrolador trabaje; simplemente conéctalo a tu PC por medio de un cable USB o con una fuente de alimentación externa (9 hasta 12VDC). El Arduino Mega es compatible con la mayoría de los shields diseñados para Arduino Duemilanove, diecimila o UNO.

Esta nueva versión de Arduino Mega 2560 adicionalmente a todas las características de su sucesor utiliza un microcontrolador ATMega8U2 en vez del circuito integrado FTDI. Esto permite mayores velocidades de transmisión por su puerto USB y no requiere drivers para Linux o MAC (archivo inf es necesario para Windows) además ahora cuenta con la capacidad de ser reconocido por el PC como un teclado, mouse, joystick, etc.

Tabla 2.1. Características del Arduino Mega 2560

Descripción	**Característica**
Microcontrolador	ATmega1280
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límites)	6-20V
Pines de E / S digitales	54 (de los cuales 15 proporcionan salida PWM)
Clavijas de entrada analógica	Dieciséis
Corriente DC por Pin E / S	40 mA
Corriente DC para 3.3V Pin	50 mA
Memoria flash	128 KB de los cuales 4 KB utilizados por el gestor de arranque
SRAM	8 KB

EEPROM	4 KB
Velocidad de reloj	16 MHz

2.4 MEMORIA SD

Secure Digital (SD) es un formato de tarjeta de memoria no volátil diseñada para ser usada en dispositivos portátiles (teléfonos móviles, cámaras digitales, computadores portátiles, etc.) cuya especificación es mantenida por la SD Card Association (SD Association, 2017), entidad que engloba muchos fabricantes de hardware (Torrente Artero, 2013).

El estándar SD incluye 4 “familias”, las cuales son las siguientes (por orden cronológico de aparición): las tarjetas originales (SDSC –Standard capacity-); las tarjetas de alta capacidad (SDHC –High capacity-, que permiten almacenar hasta 32 Gbytes); las tarjetas de capacidad extendida (SDXC –Extended capacity-, que permiten almacenar hasta 2048 GBytes); y las tarjetas que combinan almacenamiento de datos con funciones de entrada/salida (SDIO).

Los dispositivos que alojan las tarjetas SD siempre son compatibles “hacia atrás” (es decir: si el dispositivo es compatible con SCXC, también lo será con SCHC y SDSC, por ejemplo). No obstante, cada una de estas familias redefine sus conexiones internas, por lo que una tarjeta SCXC (por ejemplo) no puede ser utilizada dentro de dispositivos que no sean explícitamente compatibles con esa familia en concreto.

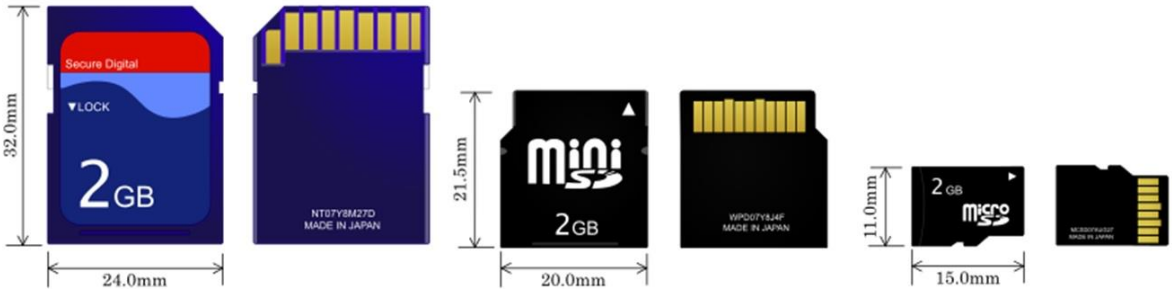


Figura 2.14. *Arduino Mega 2560*.

2.4.1 Formatos microSD

Igual que la norma general de tarjetas SD se divide en tres formatos (SD, SDHC y SDXC) las tarjetas microSD se han comercializado en estas variantes. Y es lo primero a valorar en la compra porque no son compatibles con todos los dispositivos, si bien en la actualidad la tercera es la más extendida y utilizada.

- **microSD:** La más antigua. Tienen una capacidad de hasta 2 Gbytes y se pueden utilizar en cualquier ranura microSD.
- **microSDHC:** Tienen una capacidad superior a los 2 GB y hasta 32 GB. se puede utilizar en dispositivos que soporten SDHC y SDXC.
- **microSDXC:** Son las más modernas y la referencia de compra siempre que el dispositivo la soporte porque solo son compatibles con hardware SDXC. Su capacidad varía desde 32 Gbytes a unos impresionantes 2 Tbytes. Es el máximo teórico de la norma, si bien, hasta ahora, el máximo ofertado por la industria es de 512 Gbytes.

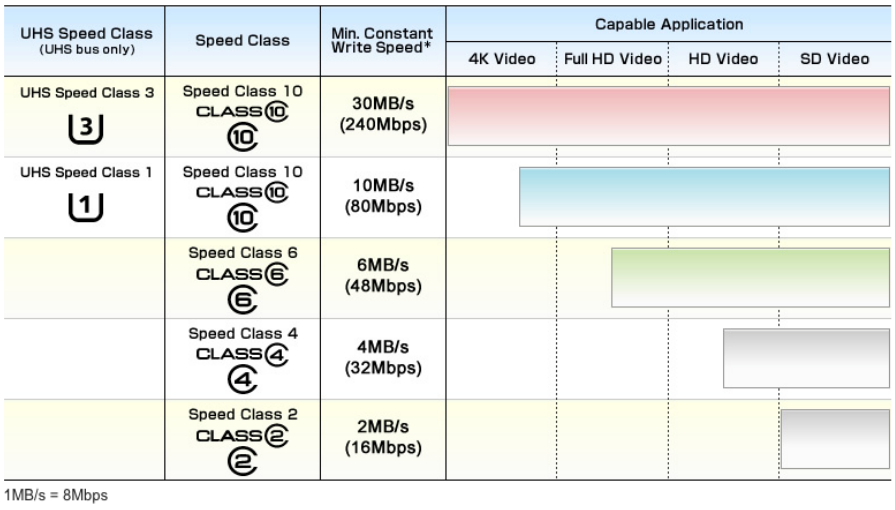


Figura 2.15. *Clasificación SD*.

Desgraciadamente, las tarjetas SDXC se comercializan preformateadas con el sistema de ficheros privativo y pantentado exFAT de Microsoft, por lo que algunos

dispositivos es posible que no sean capaces de leer su contenido. En estos casos, para que estas tarjetas sean reconocidas, se deben formatear con otro sistema de ficheros, generalmente FAT32. De hecho, la mayoría de tarjetas SD de las otras familias se suelen adquirir en cualquier tienda local de electrónica ya formateadas en el sistema FAT32. El FAT32 es el sistema de ficheros más extendido en tarjetas SD debido a que la gran mayoría de aparatos electrónicos que utilizan este tipo de tarjetas (cámaras, móviles, etc.) son capaces de reconocer este formato sin problemas.

2.5 MODULO DE LECTURA DE MICRO SD

La información recibida a través de la serial de Arduino serán almacenados en la tarjeta de un micro SD donde se empleara un módulo MicroSD Card Adapter de Catalex V1.0 (Véase Figura 2.16.) permite la lectura y escritura de datos y almacenarlos en la tarjeta de memoria. Está diseñado para trabajar con Raspberry Pi o Arduino, soporta tarjetas de hasta 2 GB y SDHC de hasta 32 GB.

Figura 2.16. *Lector de tarjetas micro SD.*

2.5.1 Requerimientos del módulo micro SD

- Voltaje de operación 5 V dc.
- Soporte de tarjetas micro SD.
- Memoria de 1 GB.
- Interfaz compatible con Arduino.
- Temperatura de operación máxima superior a 39 °C.

- Tamaño compacto.
- Implementación de hardware simple.

La tarjeta microSD es gestionada a través de la interfaz de comunicación SPI, se tendrá un puerto base/hembra de 6 pines, donde será colocado dicho módulo de los pines del bus SPI: SC (Selection Signal) pin de la señal de selección; SCK (Reloj serie), sincroniza la transmisión de datos generada por el micro; MOSI (Master Sale Slave In), la línea de maestro para el envío de datos al módulo microSD; MISO (Master En Slave Out), la línea de esclavo para el envío de datos al micro; y los pines de alimentación VCC y tierra GND.

2.6 DISPOSITIVOS DE POTENCIA

2.6.1 TRIAC (TRIode for Alternative Current)

El TRIAC es un dispositivo de estado sólido que su modo de operación emula a un relé. En estado de conducción tiene una impedancia muy baja que permite circular grandes niveles de corriente con una tensión MT2-MT1 del orden de 1V. En estado de corte la corriente es prácticamente nula y se comporta como un circuito abierto. Este dispositivo puede controlar corriente en cualquier dirección. Normalmente, tiene una tensión de ruptura alta y el procedimiento normal de hacer entrar en conducción a un TRIAC es a través de un pulso de disparo de puerta (positivo o negativo.

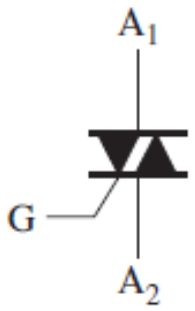


Figura. 2.17. *Símbolo del TRIAC.*

2.6.2 TRIAC con acoplado óptico (opto coupler TRIAC)

Los TRIACs acoplados óptimamente combinan un diodo emisor de luz (LED) con un TRIAC foto-detector (foto-TRIAC) dentro de un mismo encapsulado opaco. Al no existir conexión eléctrica entre la entrada y la salida, el acoplo es unidireccional (LED al foto-TRIAC) y permite un aislamiento eléctrico entre ambos dispositivos de hasta 7500 V (typ).

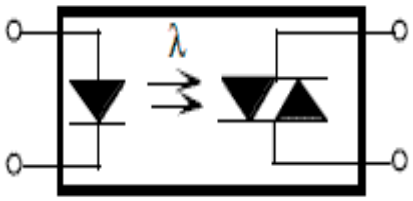


Figura 2.18. *Esquema de un opto-TRIAC.*

Como ejemplo de estos circuitos se encuentra el MOC3020 (Motorola) que necesita una corriente en el LED de 30 mA y soporta un voltaje de bloqueo en el TRIAC de 400 V. Cuando el LED está apagado, el foto-TRIAC está bloqueado conduciendo una pequeña corriente de fuga denominada I_{DRM} (peak-bloking current). Cuando el diodo conduce, dispara al foto-TRIAC pudiendo circular entre 100mA y 1A. Al no ser un dispositivo que soporte grandes niveles de potencia, el propio foto-TRIAC en muchos casos actúa sobre el control de un TRIAC de mucho mayor potencia, tal como se indica en la Figura 2.18. En este circuito, una señal digital (por ejemplo, una señal de un microcontrolador) activa el oto-acoplador que a su vez activa el TRIAC de potencia conectado a la red.

La figura 2.18 muestra un circuito de control de disparo de un TRIAC, usando el MOC3020. La máxima razón de la fuente de corriente de MOC3020 selecciona el mínimo valor de R1 a través de la ecuación:

$$R1(min) = \mathrm{V_{in(pk)}}/1.2\mathrm{A}$$

Si se trabaja en con el voltaje nominal de 240 V,

$V_{in(pk)} = 360V$, entonces

$$R1 = \frac{360}{1.2^{a}} = 300\Omega$$

Puede ser usado un valor estándar de 330Ω en la práctica.

2.7 CARGAS RESISTIVAS

Cuando se manejan cargas resistivas, el circuito de la Figura 2.19 puede ser usado. Elementos como lámparas incandescentes y resistencias calefactores son las dos clases más importantes de cargas resistivas para el cual los optoacopladores y los TRIACs son utilizados. La restricción más importante es que el TRIAC deberá ser elegido correctamente para soportar las variantes en la carga. Las lámparas incandescentes algunas veces pueden provocar picos de corriente conocidos como "flashover" los cuales pueden ser extremadamente altos.

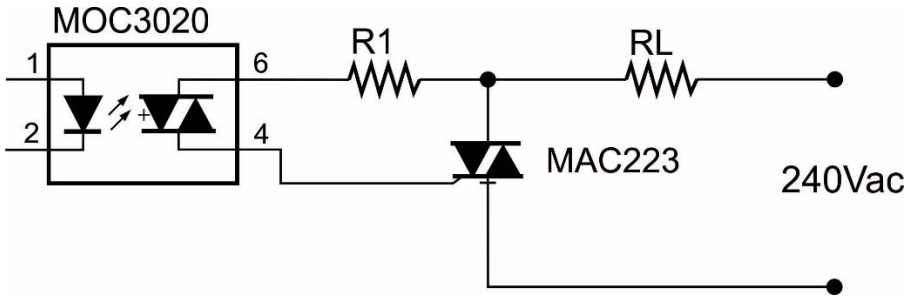


Figura 2.19. *Circuito de disparo para cargas resistivas.*

2.7.1 LINEA TRANSITORIA – ESTATICA dv/dt

Ocasionalmente alteraciones en el voltaje transitorio en la línea de CA excederá la razón estática dv/dt del MOC3029. En este caso es posible que el MOC3020 y el TRIAC asociado sean encendidos. Esto no es un problema, excepto en caso de ruidos inusuales, porque el MOC3020 y el TRIAC se apagan en el próximo cruce por cero de la línea de voltaje, y la carga no será afectada por un ocasional medio ciclo de potencia aplicada.

2.7.2 CARGAS INDUCTIVAS – CONMUTACIÓN dv/dt

Cargas inductivas (motores, solenoides, etc.) presentan un problema para el TRIAC y para el MOC3020, porque la corriente no está en fase con el voltaje. A partir de que el TRIAC apaga a la corriente cero, puede ser difícil apagarlo cuando la corriente aplicada es cero pero el voltaje aplicado es alto. Esto presenta al TRIAC un rizo imprevisto en el voltaje aplicado, el cual enciende al TRIAC si la razón de rizo excede la conmutación dv/dt del TRIAC o la estática dv/dt del MOC3020.

2.7.3 RED SNUBBER

La solución a este problema es provisto por el uso de redes snubbers para reducir la razón del rizo de voltaje visto por el dispositivo. En algunos casos esto puede requerir dos snubbers, uno para el TRIAC y otro para el optoacoplador.

2.7.4 Red snubber para el optoacoplador

Para asignar adecuadamente el valor de la red snubber del optoacoplador, uno debe conocer el factor de potencia de la carga reactiva, el cual es definido como el coseno de la fase cambiada causado por la carga. Desafortunadamente, no siempre es conocido, esto hace algunas veces la asignación de la red snubber de manera empírica. Sin embargo un método de asignación de la red snubber puede ser definida basando un factor de potencia alto. Esto puede ser usado como un “first cut” y después se modificara basado en la experimentación alta.

Asumiendo una carga inductiva con un factor de potencia de PF=0.1, será difícil apagar el TRIAC de potencia (MAC223) cuando el voltaje aplicado está dado por:

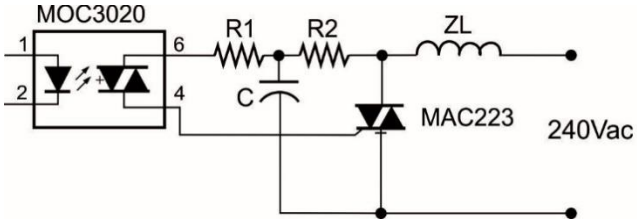
$$\mathrm{V_{to}} = \mathrm{V_{pk}}\sin\phi \approx \mathrm{V_{pk}} \approx 360\mathrm{V}$$

Primero, uno cambia el R1 para limitar el pico de la corriente de descarga del capacitor a través del MOC3020. Este resistor está dado por:

$$R1 = V_{pk}/I_{max} = 360V/1.2A = 300\Omega$$

Es necesario seleccionar la constante de tiempo para $\tau = R_2C$. Asumiendo que el TRIAC apaga muy rápidamente, tenemos una razón para el pico de rizo del MOC3020 dado por:

$$dv/dt = V_{to}/\tau = V_{to}/R_2C$$



TRIAC		
I_{GT}	R2	C
15 mA	2400	0.1
30 mA	1200	0.2
50 mA	800	0.3

Figura 2.20. *Lógica para cargas inductivas – Snubber para el optoacoplador.*

2.7.5 Red snubber para el TRIAC

Es un arreglo RC que se conecta en paralelo al TRIAC en un circuito de conmutación, como una protección para dv/dt (conmutación). Es básicamente un circuito de frenado al apagado del TRIAC, cuyo objetivo es amortiguar el efecto de una variación voltaje/tiempo que en algún momento pudiera ser destructiva para el TRIAC.

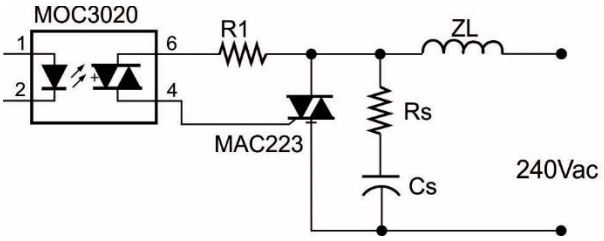
El diagrama del circuito correspondiente a la red Snubber se muestra en la figura 2.21. La relacón de sus componentes está dada por:

$$\frac{dv}{dt} = \frac{0.632RVs}{Cs(Rs + R)^2}$$

Es importante saber que el valor de la resistencia Rs, está ligado a la corriente de descarga, I_{TD} (que circulara cuando se descargue el capacitor), y que siempre se sugiere unas 10 veces mayor a la corriente de la carga. De esta forma:

$$Rs = \frac{Vs}{I_{TD}}$$

Por consiguiente si se conoce el valor de la carga R, y se sugiere el valor de la corriente de descarga, puede encontrarse fácilmente el capacitor a usar.



Corriente de carga A rms	Resistencia Ω	Capacitancia µF
5	47	0.047
10	33	0.1
25	10	0.22
40	22	0.47

Figura 2.21. *Lógica para cargas inductivas – Snubber para el TRIAC.*

CAPITULO III

DESARROLLO DEL PROYECTO

3.1 DESCRIPCION DEL SISTEMA

Para poder sustituir el sistema electrónico anterior con el sistema electrónico que se le va a implementar, utilizaremos los diagramas mostrados en la figura 3.1 y 3.2.

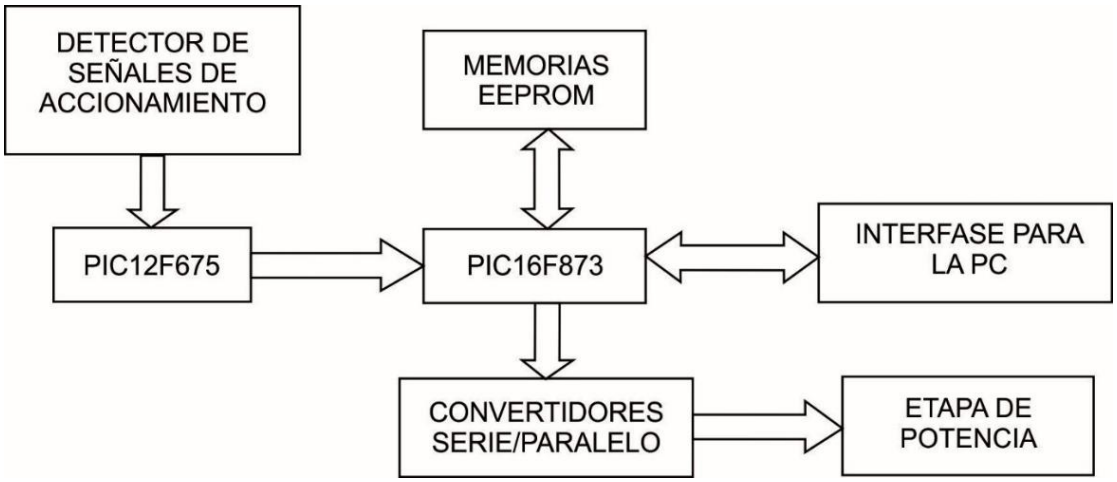


Figura 3.1. *Diagrama a bloques del sistema electrónico anterior.*

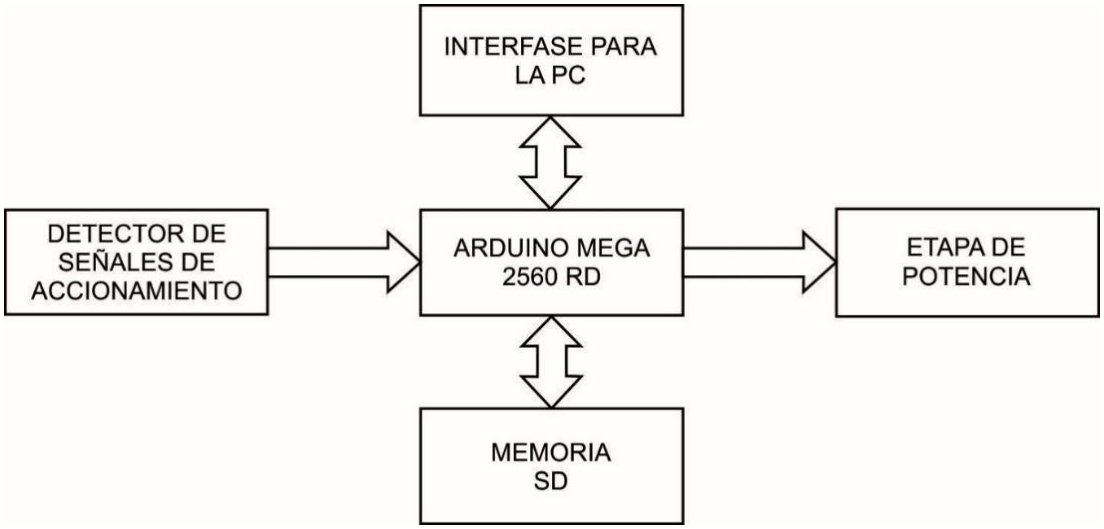


Figura 3.2. *Diagrama a bloques del sistema electrónico propuesto.*

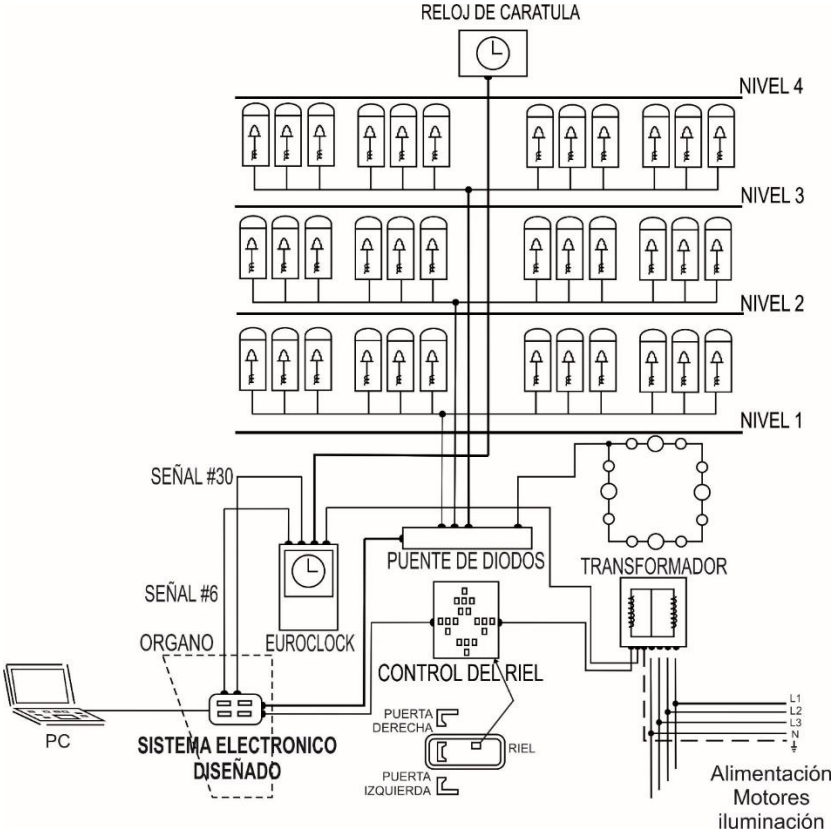


Figura 3.3 *Sistema general con el subsistema electrónico.*

En la figura 3.3 se puede observar el sistema general ya con la inclusión del subsistema electrónico diseñado, como podemos ver el sistema electrónico interactúa con las señales #6 y #30 provenientes del Euroclock (en la tabla 3.1 se detalla la función de cada una de estas señales) y la activación del control del riel es manipulada por el mismo sistema electrónico.

Las señales #6 y #30 en el sistema electrónico es para sincronizar la reproducción de las canciones con el reloj (en la figura 3.4 se muestra un diagrama de tiempo de dichas señales). La función original de la señal #6 es activar el motor de la consola y el control del riel, esta señal se presenta durante 16 horas al día (de 8 a.m a 12 p.m.); la señal #30 se presenta a cada hora e indica la hora que se ha cumplido; es decir, si las 2:00 en punto, esta señal emitirá dos pulsos, que hará sonar dos veces la campana que indica la hora. Se podría pensar que al igual que en el sistema original bastaría únicamente con la activación de la señal #6 para empezar la

reproducción de las canciones ahora con el sistema electrónico diseñado, ya que esta señal se activa a cada hora y tan solo en las horas en que debe haber música, pero la verdad es que no es suficiente, porque esta señal está desfasada con respecto al euroclock; es decir, se presenta 4 minutos antes de la señal #30, si nos conformáramos con tan solo esta señal, cundo se estuviera reproduciendo una canción con una duración mayor a 4 minutos, esta se vería traslapada con la presencia de las señal #31 (campanas de cuarto de hora) y con la señal #30 y no se entendería la canción. Es esta razón por la que se decidió sumar la señal #30 al sistema electrónico, así que este primero checa la activación de la señal #6, posteriormente la presencia de la señal #30 y después de que esta desaparece manda una señal al control del riel y empieza la reproducción de la canción.

Tabla 3.1. *Señales tomadas del euroclock.*

Señal	Funicón
30	Esta señal se activa cada hora en punto, y está conectada directamente a la campana que indica la hora que ese ha cumplido.
6	Cada hora esta señal activa al selector del riel, el cual se encarga de llevar la secuencia de activación de los contactores que controlan a los motores de las puertas y el riel, de igual forma activa al Carrillón, todo esto lo hace en el sistema original.

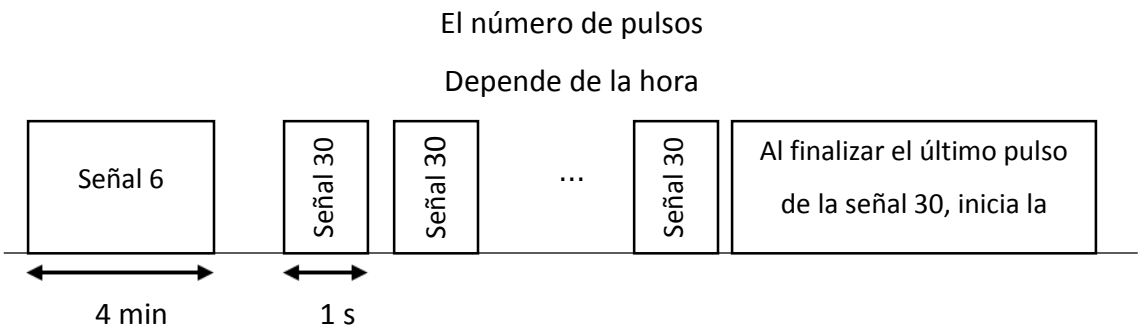


Figura 3.4. *Diagrama de tiempo de las señales tomadas del euroclock.*

3.2 DIAGRAMA A BLOQUES DEL SISTEMA ELECTRONICO

Para el desarrollo del proyecto se propuso el siguiente sistema electrónico:

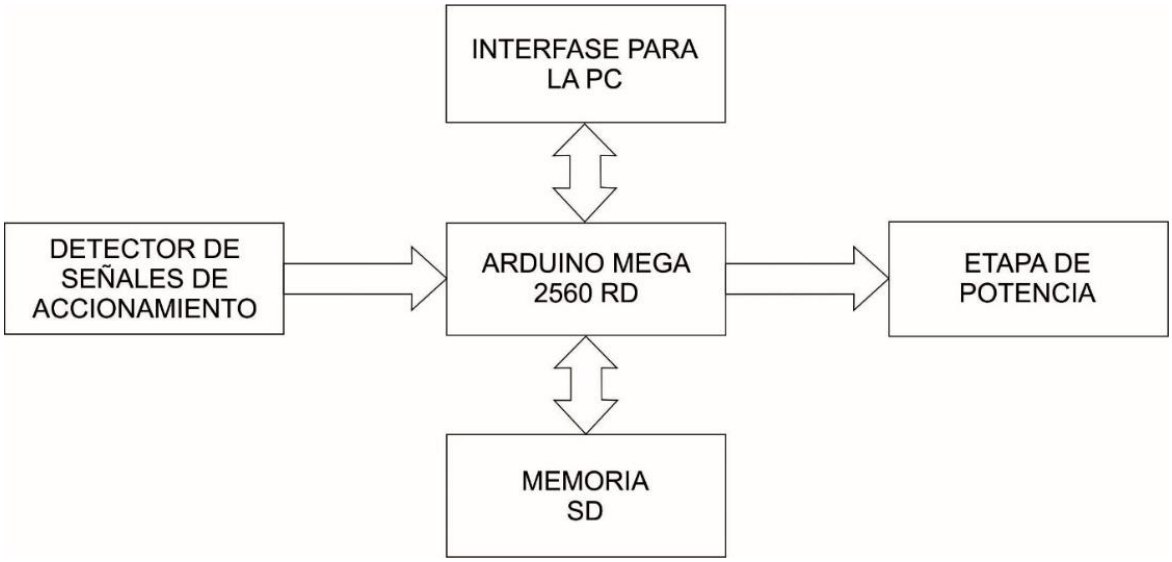


Figura 3.5. *Diagrama a bloques del sistema electrónico.*

3.2.1 Detección de señales de accionamiento

Las señales de accionamiento son dos, y provienen del euroclock, una es la señal #6 y la otra es la señal #30 (indica la hora), el arduino checa que estas dos señales se presenten, y posteriormente entra en modo de reproducción, esto sucederá únicamente de 8 am a 12 pm, ya que solo en este lapso se presenta la señal #h, y se presenta 4 minutos antes de cada hora, por lo que también es de vital importancia

considerar la habilitación de la campana de hora, con lo que se detectará la última campana de hora, para que posteriormente empiece la reproducción de la música en las campanas.

El circuito propuesto consiste en un optoacoplador NTE3089 (Electronics, 2013), que cumple con la función de aislar un voltaje de CA a un voltaje de CD, ya que las señales que proviene del euroclock son de tipo CA con una magnitud de 240V y deben ser acoplados a voltajes de CD para que el arduino mega pueda procesarlas.

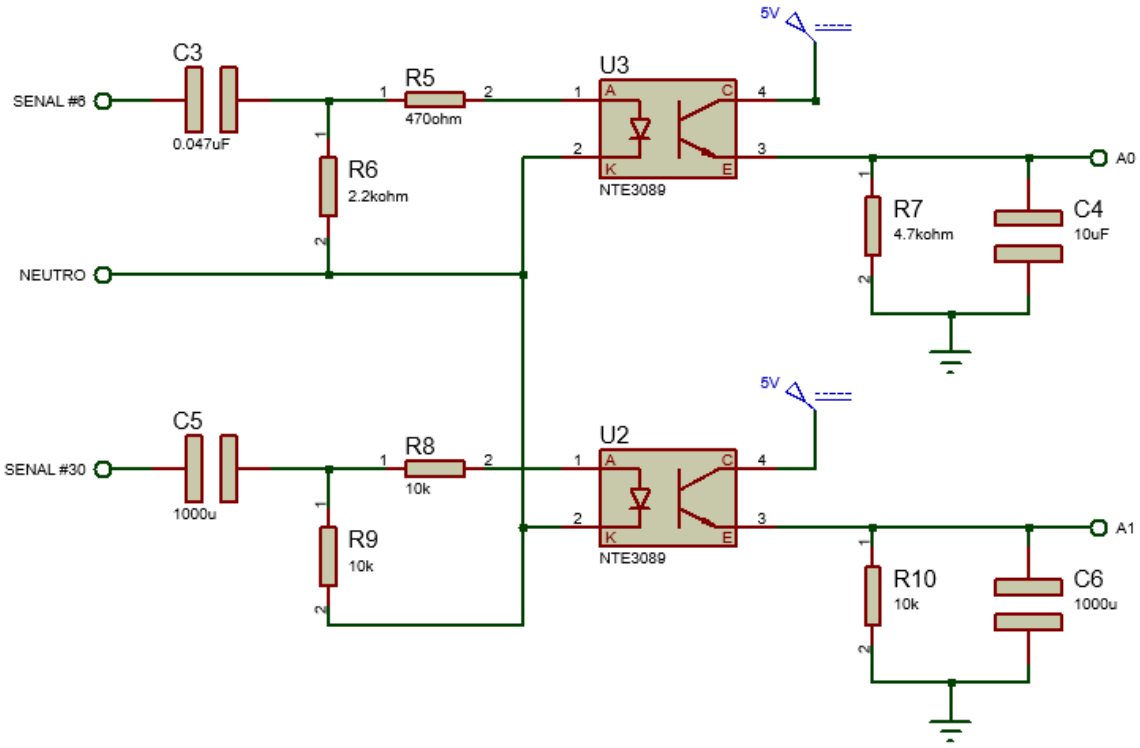


***Figura 3.6**. Esquema del detector de las señales de accionamiento.*

3.2.5. Almacenamiento de archivo

Para el almacenamiento de archivo consiste con la ayuda de un módulo (MicroSD Card Module) cmo se muestra en la Figura 3.7. que cuenta con 6 pines donde serán colocados del bus SPI del Arduino Mega 2560: SC al pin 53; SCK al pin 52; MOSI al pin 51; MISO al pin 50 y los pines de alimentación VCC y tierra GND del Arduino.

La tarjeta cuenta con un circuito regulador de voltaje a 3.3V suministro para una memoria Micro SD.

Se ocupó un Micro SDHC de clase 10, donde nos permite dar un formato de FAT3 que requiere el módulo de lectura de micro SD, este modelo de dicha memoria, tienen una capacidad superior a los 2 GB, suficientemente para almacenar una canción recibida a través del puerto serial del arduino, que puede llegar hasta los 2000 acordes, un espacio de 16 Kbytes.

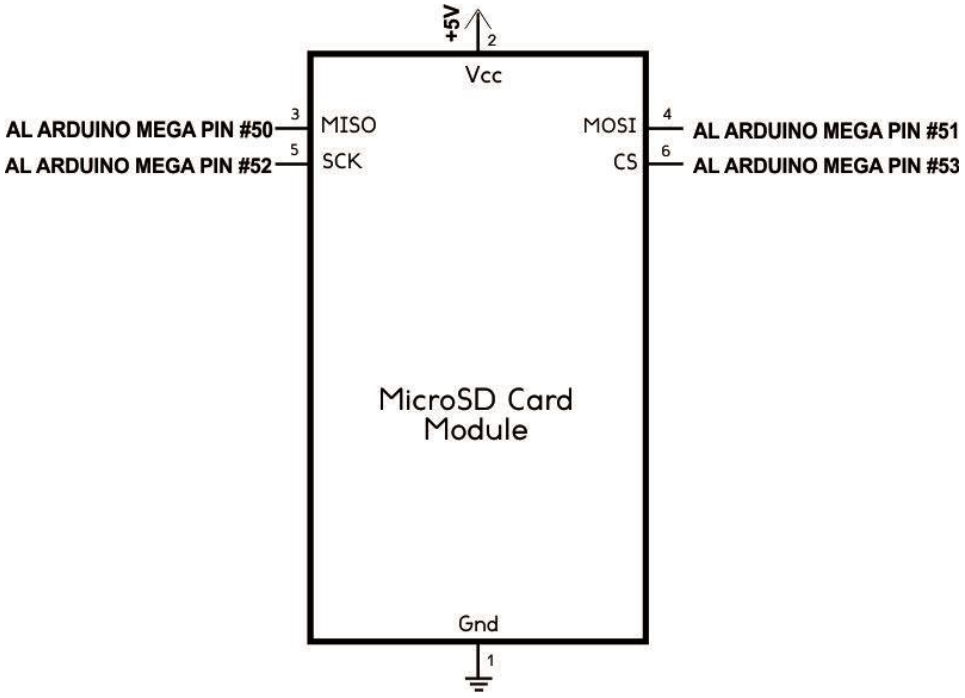


***Figura 3.7**. Esquema Modulo de Lectura de Micro SD.*

3.2.6. Sistema de Arduino Mega 2560

Usando Arduino Mega 2560 lleva a cabo todo el control de todas las operaciones dentro del sistema, así como se muestra en la Figura 3.8. Para la activación de las campanas se a ocupando desde al pin bus número 2 del arduino, hasta el pin bus número 49, que suma un total de 48 salidas para dichas campanas. La detección de las señales de activación se ocuparon tres pines analógicos del arduino que son: A0 para la señal 6 proveniente del euroclock; A1 para la señal 30 proveniente del

euroclock y A2 para activar el RIEL. Para las señales del módulo de lectura de micro SD: 50, 51, 52, 53 como se mencionó anteriormente, son ocupados hacia los pines SPI, para transmisión de datos recibidos desde el puerto serial del arduino.

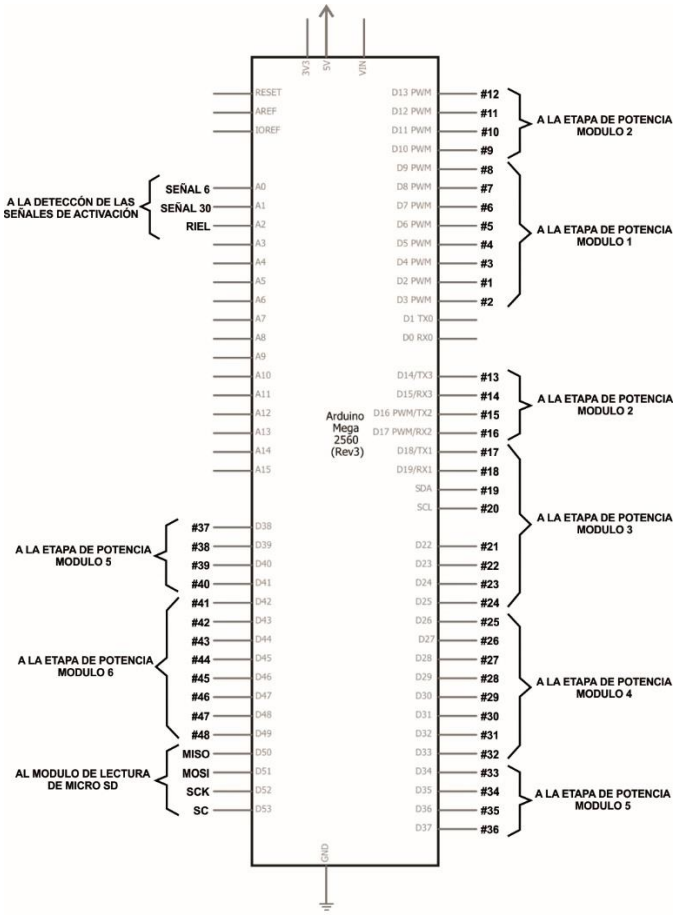


***Figura 3.8**. Sistema de Arduino Mega 2560.*

3.2.7. Etapa de potencia

El circuito de la figura 3.9 muestra la etapa de potencia, que sirva para activar a cada una de las campanas, la señal de activación viene de las 48 salidas tomadas del arduino mega, y está conectada al moc3021 a través de una resistencia de 0.33 KΩ. El moc3021 es un optoacoplador (FAIRCHILD SEMICONDUCTOR, 2017), que aísla la parte de potencia de la parte digital y soporta un voltaje de 240V a través de sus terminales.

La señal de activación para las campanas se mantendrá presente en los TRIACs durante un tiempo de 100 ms, tiempo suficiente para activar los electroimanes de cada campana.

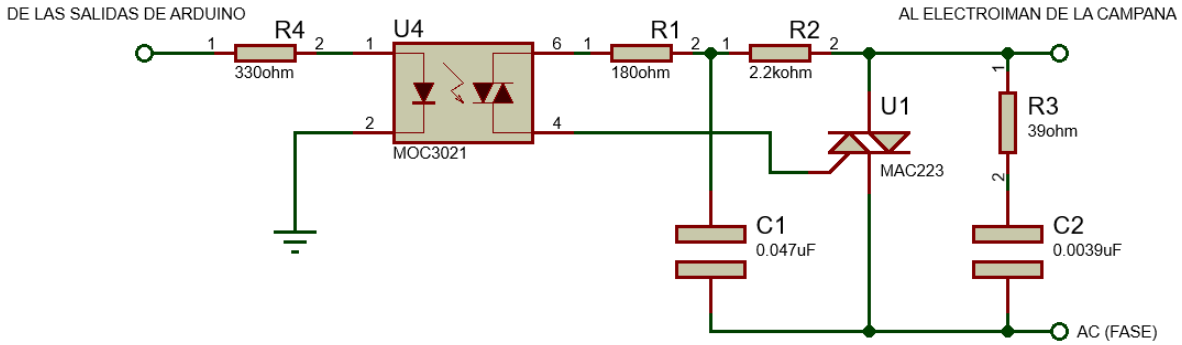


Figura 3.9. *Circuito de potencia.*

3.3 PROGRAMA ARDUINO

3.3.1 Diagrama de flujo para descarga de archivo

Para el modo de grabación mostrado en la Figura 3.10. Se checa primero si hay datos en el puerto serial entre la PC y el arduino; con el método Serial.available(), obtiene la cantidad de bytes almacenados en el búfer para poder leerlos, en nuestro programa, cuando esto sea mayor que 0, con el Serial.readString() lee los bytes del puerto serial en una cadena y, lo guarda en la variable num+1 empezando a nombrar el archivo 1.txt, 2.txt ... con la función sprintf(), al contrario si existe el archivo se elimina y crea un nuevo archivo. Una vez creado el archivo se escribe los datos recibidos, y cierra para finalizar.

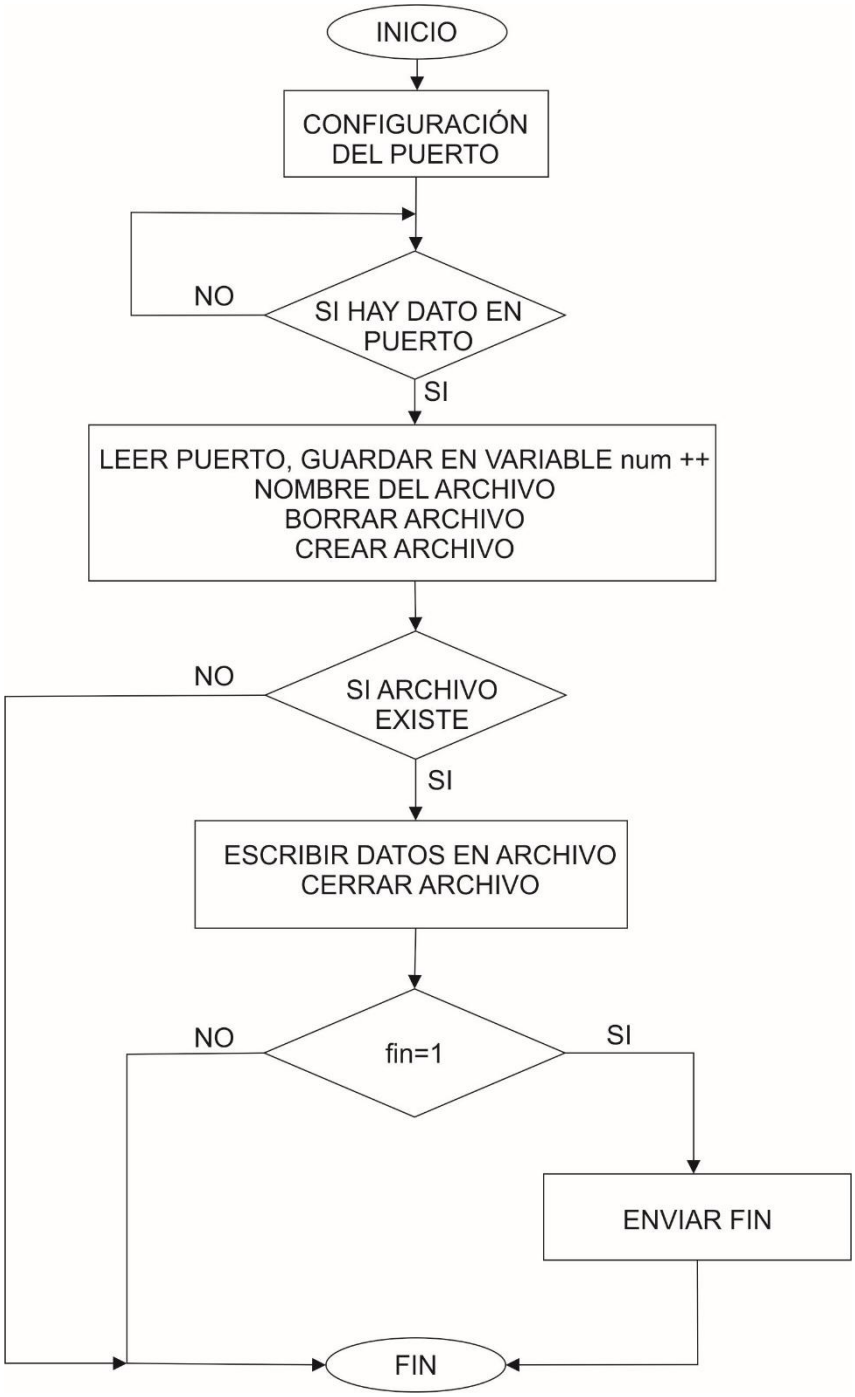


Figura 3.10. *Descarga de archivo.*

3.3.2. Diagrama de flujo para reproducción de música

Para el modo de reproducción consiste en los siguientes pasos: comenzando desde la figura 3.11. Inicia a leer la señal 6 y la señal 30, sino se a activo la señal 6 termina. Una vez que llega la señal 6 entonces entra en modo música (música=1), esperando los pulsos de la señales 30. Llegando la señal 30 checa los números de pulsos obtenidos, si solo es un pulso, comienza a leer los datos del primer archivo o 1.txt o en modo reproducción de la primera música.

Al contrario si el número de pulsos recibidos es igual a número de horas, entonces comienza a leer el archivo enumerado para dicho número de hora, esté proceso la podemos ver en la Figura 3.12.

Comenzando la reproducción de música, se selecciona el número de archivo para abrirlo, lee el total de bytes del archivo, si archivo no existe termina. Al contrario si archivo existe en la lista, se sitúa en una posición específica en el archivo. Este proceso se encuentra en la Figura 3.13.

En la Figura 3.14. Obtenemos que si tenemos posición=0 esto termina. Al contrario entra en modo posición del archivo y tamaño. Obtiene la cantidad de bytes disponibles para leer desde el archivo y procede a leer los caracteres. Selecciona notas para activar salidas hasta que última posición se iguale a cero termina proceso de reproducción.

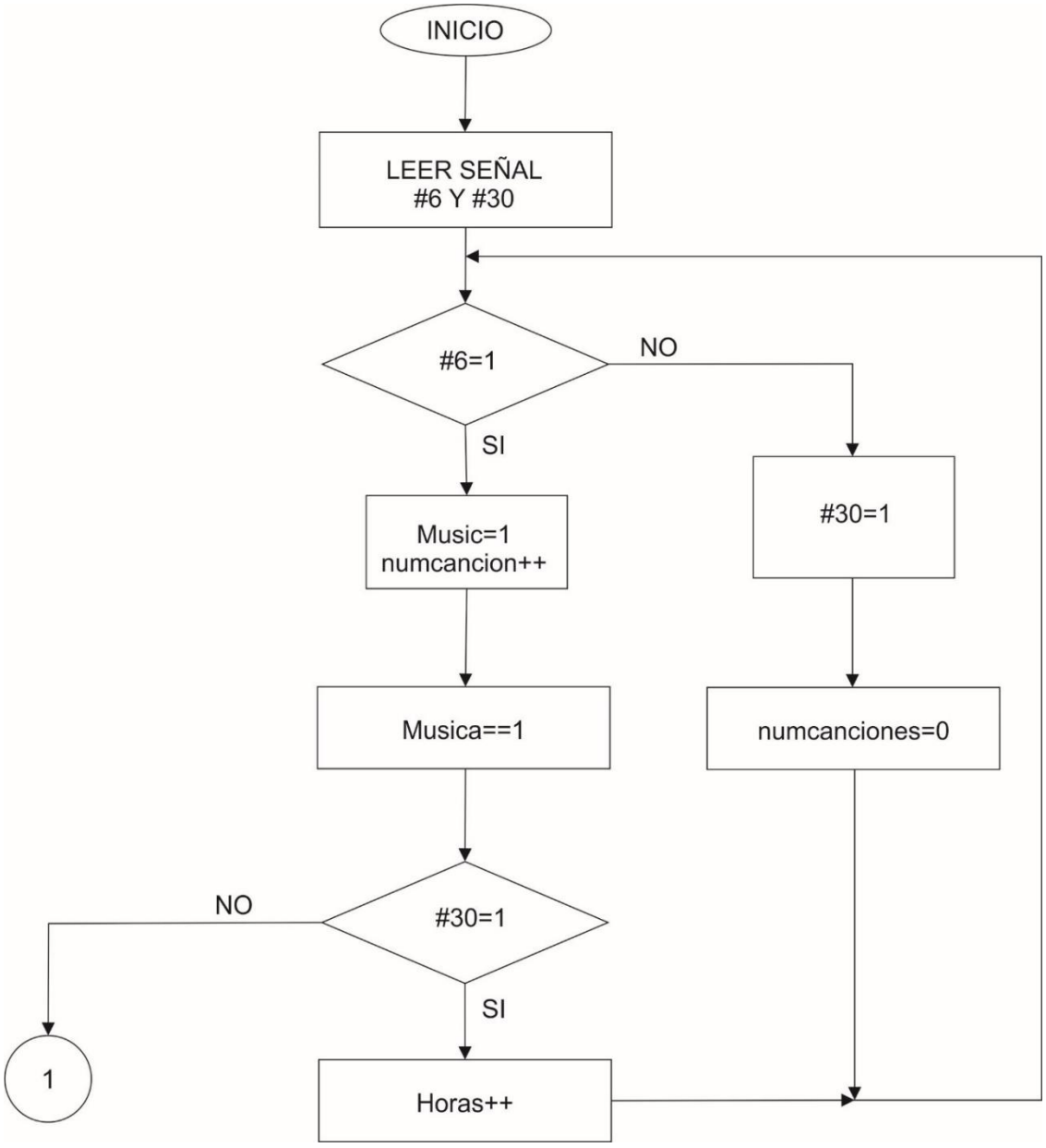


Figura 3.11. *Reproducción de música parte 1.*

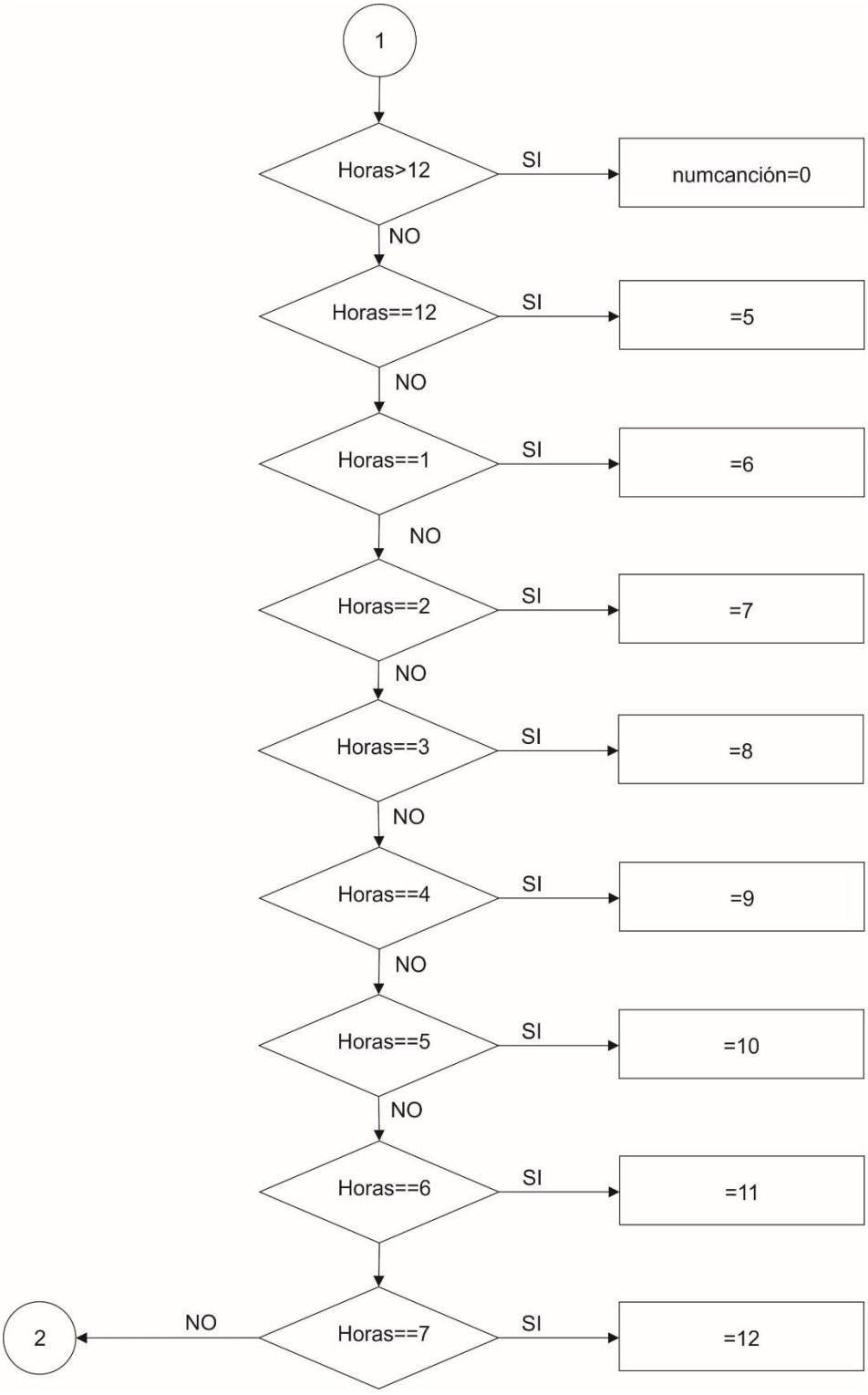


Figura 3.12. Reproducción de música parte 2.

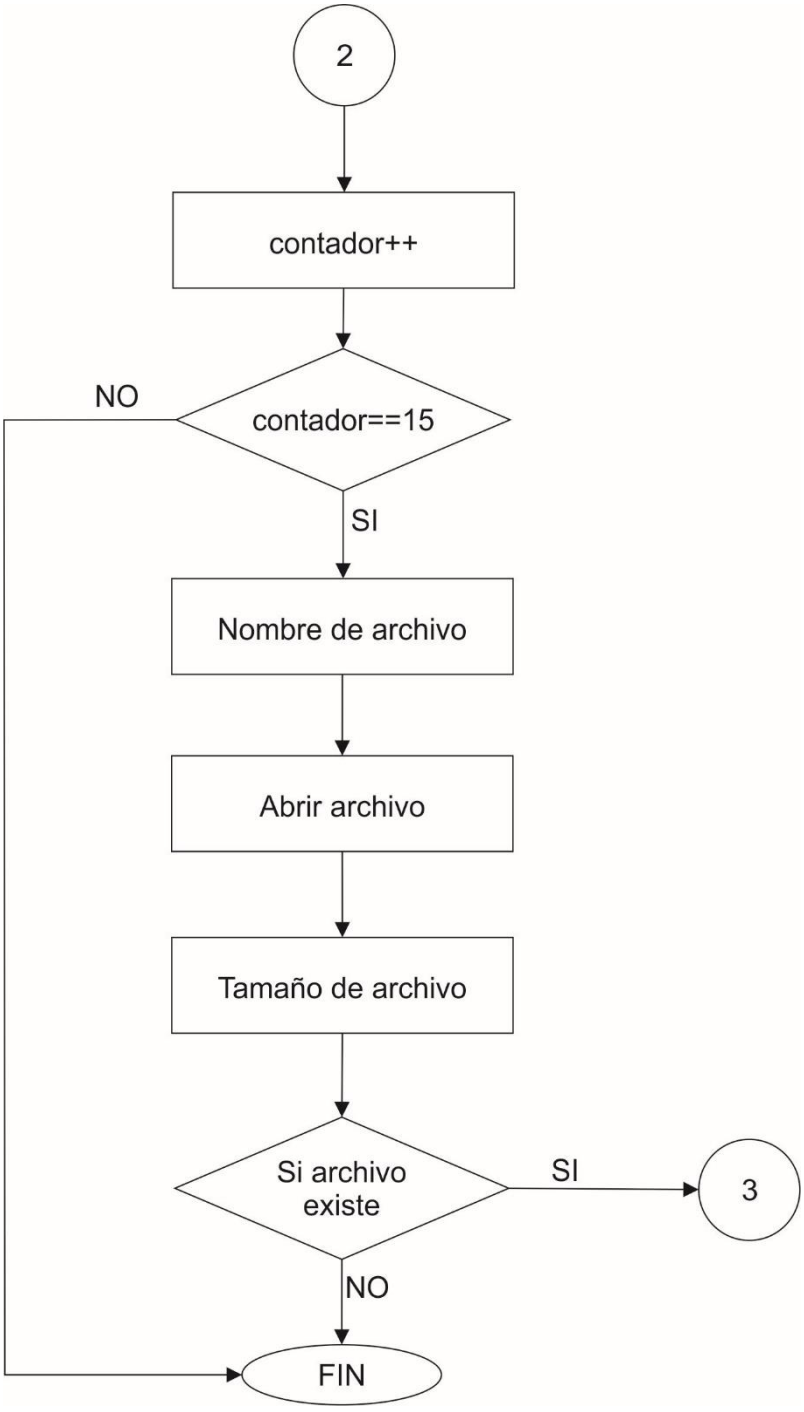


Figura 3.13. *Reproducción de música parte 3.*

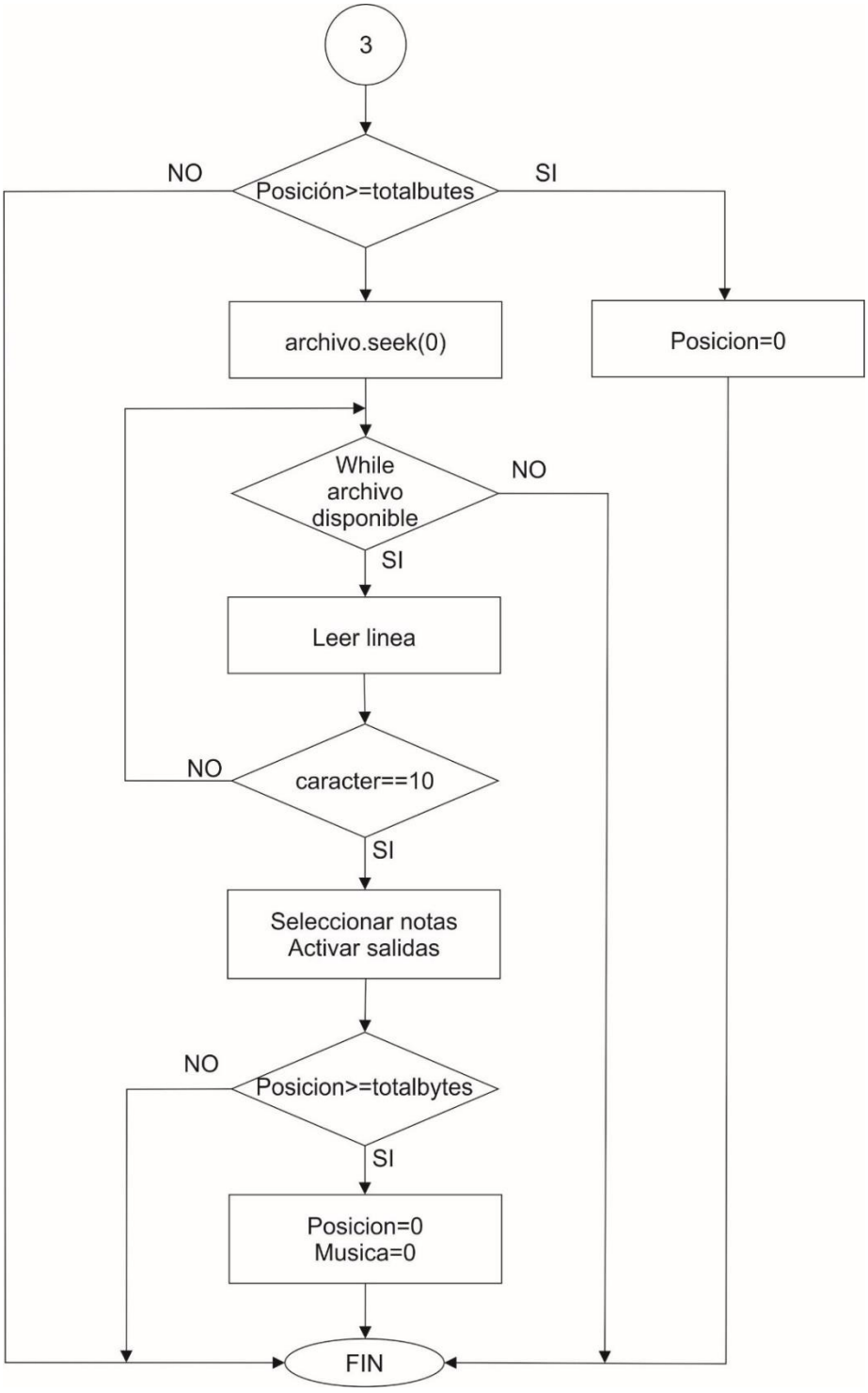


Figura 3.14. *Reproducción de música parte 4.*

3.4 IMPLEMENTACIÓN DEL SISTEMA

3.4.1 Diseño de placas

Las placas se diseñaron usando el programa Ares Proteus 8.1. Que nos permite desarrollar las conexiones necesarias para proyecto.

Se Crearon 6 puertos de salidas que se conectan para cada módulo de la etapa de potencia como se muestra en la Tabla 3.1. Cada puerto cuenta con 10 pines. Esto es debido a que un módulo de la etapa de potencia, cuenta con 8 etapas de potencia, que pertenecen a la activación de las campanas y dos más para la tierra. A cada puerto los pines 1, 3, 5, 7, 4, 6, 8, 10 son para salidas de activación de las campanas, 2 y 9 de tierra como se muestra en la Figura 3.14.

Tabla 3.2. Descripción de los de los puertos de salida a cada módulo de la etapa de potencia.

Pines	Puerto 1	Puerto 2	Puerto 3	Puerto 4	Puerto 5	Puerto 6
1	Salida 1	Salida 9	Salida 17	Salida 25	Salida 33	Salida 41
2	GND	GND	GND	GND	GND	GND
3	Salida 2	Salida 10	Salida 18	Salida 26	Salida 34	Salida 42
4	Salida 5	Salida 13	Salida 21	Salida 29	Salida 37	Salida 45
5	Salida 3	Salida 11	Salida 19	Salida 27	Salida 35	Salida 43
6	Salida 6	Salida 14	Salida 22	Salida 30	Salida 38	Salida 46
7	Salida 4	Salida 12	Salida 20	Salida 28	Salida 36	Salida 44
8	Salida 7	Salida 15	Salida 23	Salida 31	Salida 39	Salida 47
9	GND	GND	GND	GND	GND	GND
10	Salida 8	Salida 16	Salida 24	Salida 32	Salida 40	Salida 48

Se usan conectores IDC hembra de 10 vías para los pines pertenecientes a cada puerto y cable plano de 10 vías. Esto es para poder vincular de los puertos de salida

de la placa diseñada, hacia los puertos de entrada de cada módulo de la etapa de potencia.

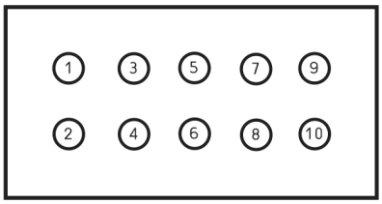


Figura 3.15. *Pines para conector IDC cable plano hembra aérea 10 vías.*

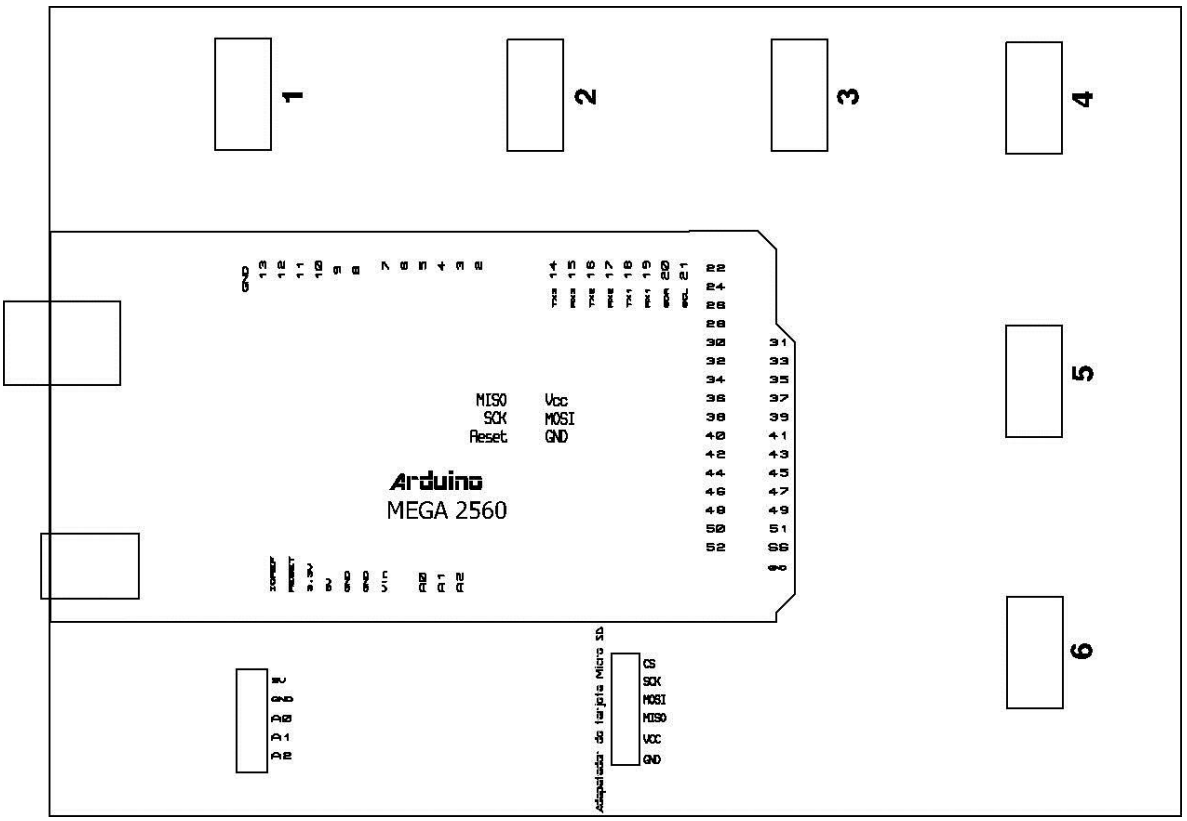


Figura 4.16. *Diseño de salida para puertos del sistema.*

Se implementó una nueva placa para las señales de activación, para su diseño se aprovechó el programa Ares en Proteus, como se puede observar en la Figura 3.17. Donde se nombraron los pines de entradas y salidas de señales que serán

conectados a la placa del sistema de reproducción de música. De la misma manera las salidas y entradas de 240V. Donde pueda activar el riel y las señales que provienen del euroclock que son señal 6 y señal 30.

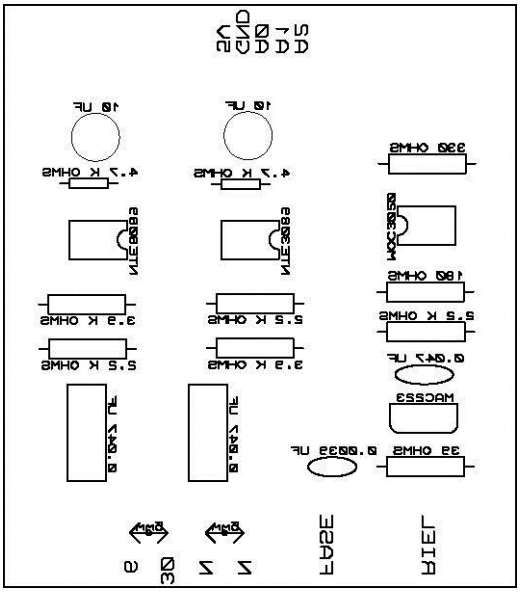


Figura 3.17. *Diseño para la etapa de potencia.*

3.5 PRUEBAS Y RESULTADOS

3.5.1 Función del sistema

El funcionamiento del sistema de reproducción de música, conlleva la etapa de potencia, debido a que la placa anterior ya no coincidía con el nuevo sistema que se había creado. En la Figura 3.20. Se muestra las conexiones de las dos placas. Donde las dos señales de activación #6 y #30 van a las entradas de arduino A0 y A1, más una seña de salida A2 que pertenece a la activación del riel.

Se sustituyeron las dos señales de activación #6 y #30 con dos push botóns y, la señal de salida del riel con un led blanco. Esto debido a que no había donde tomar los 240 V, ya que para ir a probarlo en la Catedral de San Marcos, era limitado el acceso; por eso se ocupó de esta forma.

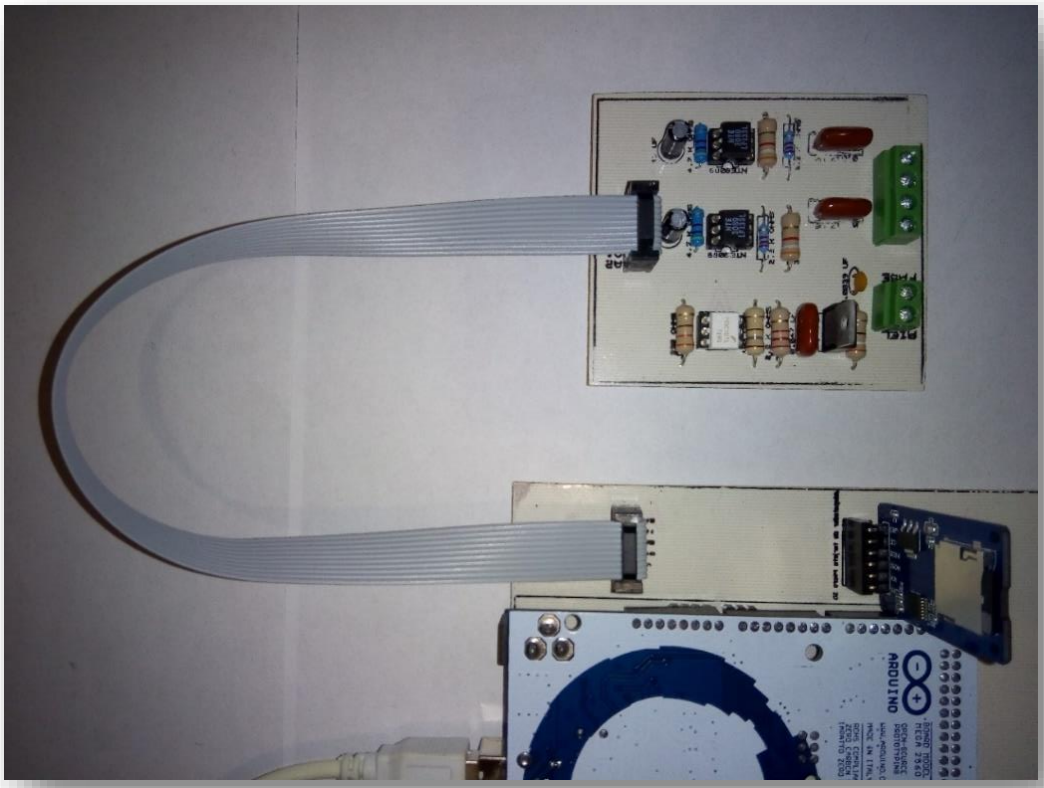


Figura 3.20. *Conexión del sistema de reproducción con etapa la de potencia.*

Para simulación del sistema de reproducción de música, se sustituyeron las 48 etapas de potencia que activan las campanas, por 48 leds rojos con sus respectivas resistencias, esto es simplemente para probar el correcto funcionamiento del sistema de la Catedral de San Marcos y, no tenga errores al momento de ensamblar al árgano de la catedral.

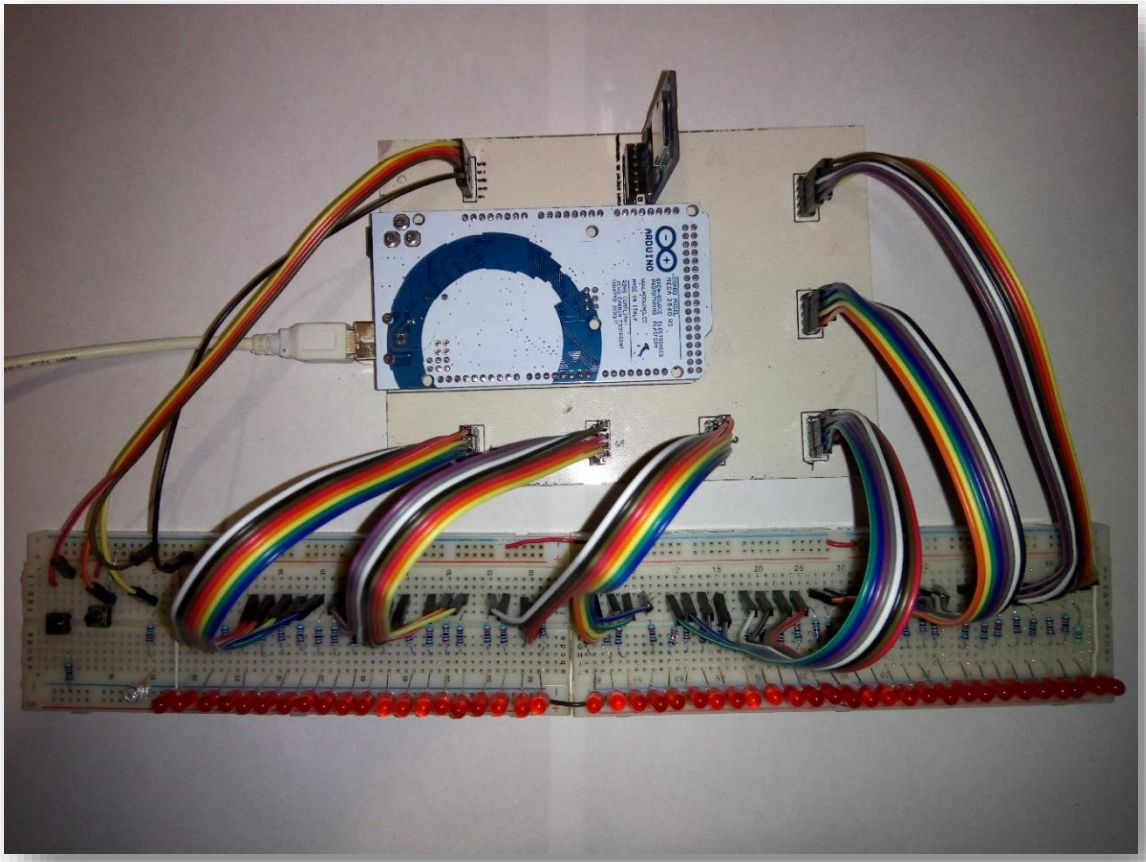


Figura 3.21. *Prueba del sistema de reproducción de música con las 48 campanas.*

3.5.2 Descarga de archivos

Para la descarga de los archivos se haces desde la aplicación iCatedral de la siguiente manera:

1.- Elegimos “Puerto Serial”.

2.- Le damos a “Nuevo”.

3.- “Agregar”.

4.- Buscamos archivo ABC generados por el programa iCatedral.

5.- Le damos a “Descargar”.

El usuario definirá el número de canciones que desea almacenar a la tarjeta de micro SD.

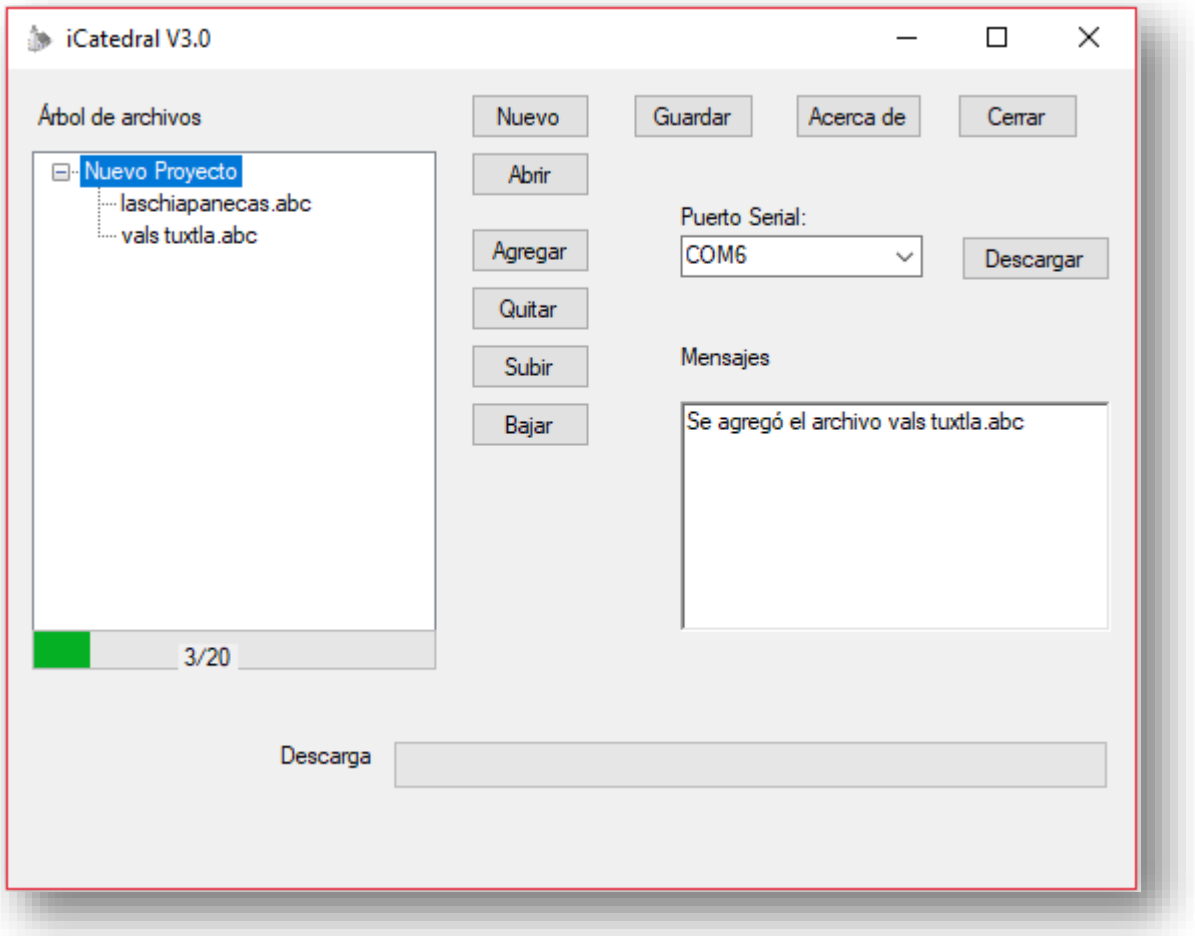


Figura 3.22. *Programa iCatedral.*

3.5.3 Representación de las notas en el archivo .TXT

El sistema del órgano que se encuentra en el interior de la Catedral de San Marcos se constituye a las salidas de las campanas, que están conformados por 48 notas, divididas en 5 escalas. La escala del órgano empieza con la nota La, que en nuestra lectura de datos representa como la nota número 01 y así consecutivamente como

podemos ver en la Tabla 3.3. Donde apreciamos el número total de campanas conformados por una nota distinta a cada una.

Tabla 3.3. Representación de las notas en el archivo .TXT

Nota	Escala 0	Escala 1	Escala 2	Escala 3	Escala 4
Do		04	16	28	40
Do#		05	17	29	41
Re		06	18	30	42
Re#		07	19	31	43
Mi		08	20	32	44
Fa#		09	21	33	45
Fa#		10	22	34	46
Sol		11	23	35	47
Sol#		12	24	36	48
La	01	13	25	37	
La#	02	14	26	38	
Si	03	15	27	39	

Para guardar los datos obtenidos que se lee desde el puerto serial, se crea archivos de texto en la micro SD, donde se va ordenando por lista: cuatro números para el tiempo, más dos números para salida de activación de las campanas; dependiendo el acorde que se va a tocar se crean las notas de dos en dos números, máximo 4 notas, donde se forma un acorde.

La lectura de archivos lo hace línea por línea como se muestra en la Figura 3.23.

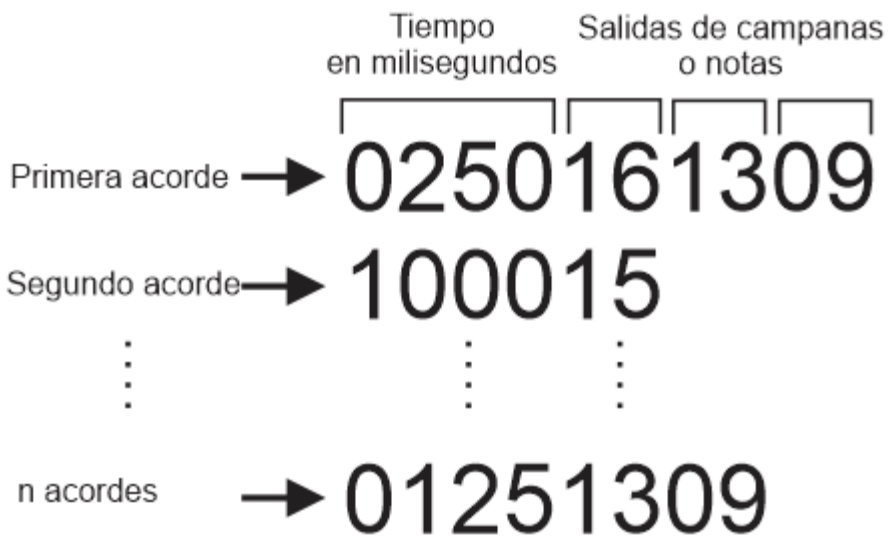


Figura 2.23. *Muestra para lectura de notas.*

Las melodías almacenadas en la micro SD, se ordena del 1 al 16. Esto es debido a las horas que estará activando la señal #6, proveniente del euroclock, será de 8:00 a.m. a 12 p.m., que cumple con las 16 horas permitidas al día. La descarga de los archivos desde la aplicación iCatedral, se actualiza a partir de las de las 8:00 a.m. para tocar la primera melodía, la segunda melodía a las 9:00 a.m., las 10:00 a.m. se tocara la melodía que contiene el archivo de texto número 3 y así consecutivamente una melodía a cada hora que irá transcurriendo. Al contrario, si el usuario cargue los archivos en la tarde empezara a tocar la melodía en esa hora que comienza a llegar la señal #6. Supongamos que el usuario empezó a descargar los archivos a las 3:30 p.m., entonces cuando llegue la señal #6 a las 4:00 a.m. empezara a tocar la melodía número 1 y luego la 2 de la siguiente hora y así consecutivamente hasta llegar las 12 p.m. Como la señal #30 seguirá activándose en la madrugada, se aprovechó para actualizar la lista de canciones. Una vez que llegue la señal #6, que está se presenta a las 8:00 a.m. es donde empezara a enlistar las canciones. Si el usuario solo carga 7 archivos, solamente estará tocándose ese número de archivos en un ciclo.

CONCLUSIONES

El proyecto realizado para el sistema de reproducción de música, es de gran cambio al que se tenía anteriormente elaborado con PIC's, donde es limitada de memoria para la grabación. Donde se reemplazó con un arduino mega, usando un módulo lector/escritor de tarjeta micro SD. Se diseñó una nueva etapa de potencia para la detección de las señales de accionamiento 6 y 30 provenientes del euroclock y accionamiento del riel, en el cual ya no era necesario el PIC que contenía para la detección de señales, el mismo arduino lo puede recibir.

Se tuvo crear una nueva programación en arduino en donde el sistema pueda grabar archivos .txt, ya que se pretendía usar los mismos archivos .hex generados por el programa de la computadora, pero las extensiones y tipos de archivos que lee nuestro SD a través de la interfaz de la computadora, es limitado, ya que es de tipo FAT, excepto utilizando cualquier extensión tipo ".txt" o ".log" no hubo dificultades.

Ahora nuestro sistema, el usuario puede grabar cualquier canción que genere a través del programa de la computadora, sin la necesidad de extraer el micro sd, simplemente conectando desde el puerto USB, puede descargar las melodías a su preferencia en cualquier momento y cambiarlos cuando se desee.

La implementación del sistema, puede ser usados por otros proyectos relacionados a la comunicación serial y guardado de archivos usando el módulo de lectura de la micro SD, donde podemos expandir la memoria del arduino y así almacenar más contenido, tal si se requiere hacer algún proyecto de lectura y escritura, usando diferentes componentes según la intención, que se desea implementar.

REFERENCIAS BIBLIOGRÁFICAS

Allan R., H. (2001). *Electrónica.* Michigan Technological University: Pearson Prentice Hall.

Arduino. (10 de noviembre de 2017). Obtenido de https://www.arduino.cc/

Arduino Mega. (2017). Obtenido de ARDUINO: https://www.arduino.cc/en/Main/ArduinoBoardMega

Cajas Narváez, K. J., & Vargas Salinas, S. R. (2015). Diseño y construcción de un equipo de fototerapia con control de intensidad de luz, posicionamiento y evaluación de cambios de bilirrubina para el tratamiento de bilirrubinosis en neonatos (Bachelor's thesis, Universidad de las Fuerzas Armadas).

Castro A., J. L. (4 de octubre de 2009). *Historia de la Catedral de San Marcos en Tuxtla Gutiérrez*. Obtenido de Todo Chiapas: http://todochiapas.mx/chiapas/historia-de-la-catedral-de-san-marcos-en-tuxtla-gutierrez/2173

Catalex. (s.f.). *MicroSD Card Adapter v1.1 (Catalex)*. Obtenido de Arduitronics: https://www.arduitronics.com/product/210/microsd-card-adapter-v1-1-catalex

Electronics, N. (2013). Obtenido de NTE ELECTRONICS, INC: http://www.nteinc.com/

FAIRCHILD SEMICONDUCTOR. (2017). Obtenido de https://www.fairchildsemi.com/

Floyd, T. L. (2006). *Fundamentos de sistemas digitales.* Madrid: PEARSON EDUCACIÓN S.A.

Grave Aguilar, I. A. (2007). Reparación y modernización del sistema eléctrico y electrónico del reloj de la catedral etapa de software (Tesis de residencia). Instituto Tecnológico de Tuxtla Gutiérrez, Tuxtla Gutiérrez.

López Dávalos, M. A., Ramírez Velasquez, J. A., & Valencia Zavala, J. A. (2004). Reparación y modernización del sistema eléctrico y electrónico del reloj de la catedral de San Marcos (Tesis de residencia). Instituto Tecnológico de Tuxtla Gutiérrez, Tuxtla Gutiérrez.

MUYCOMPUTER. (2015). *Guía MicroSD*. Obtenido de https://www.muycomputer.com/2017/07/24/guia-microsd/

NAYLAMP. (2016). *Tutorial Arduino y memoria SD y micro SD*. Obtenido de NAYLAMP MECHATRONICS: http://www.naylampmechatronics.com/blog/38_Tutorial-Arduino-y-memoria-SD-y-micro-SD-.html

PROMETEC. (2004). *Usando la tarjeta SD del Shield Ethernet*. Obtenido de SHIELD ETHERNET Y SD CARD: https://www.prometec.net/sdcard/

PROMETEC. (2016). *Detectando el paso por cero de una tensión AC*. Obtenido de ZERO CROSSING DETECTION: https://www.prometec.net/zero-crossing-detection/

SD Association. (2017). Obtenido de https://www.sdcard.org/

Thayer Ojeda, L. (s.f). *ARDUINO MEGA 2560 R3*. Obtenido de ARDUINO.cl: http://www.arduino.cl/arduino-mega-2560/

Torrente Artero, Ó. (2013). *ARDUINO Curso práctico de formación.* México: Alfaomega.

ANEXOS

Anexo A: Diseño de placas PCB

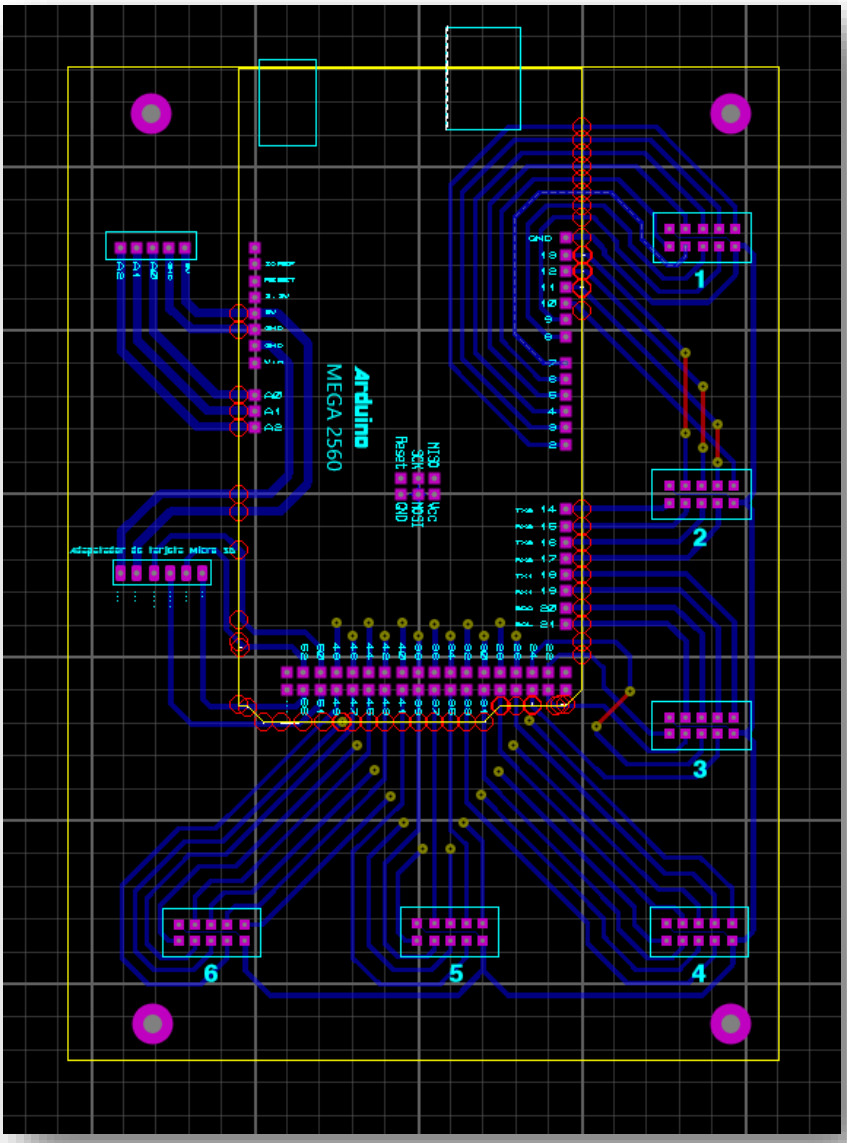


Figura A.1. *Vista previa placa para arduino en Ares Proteus.*

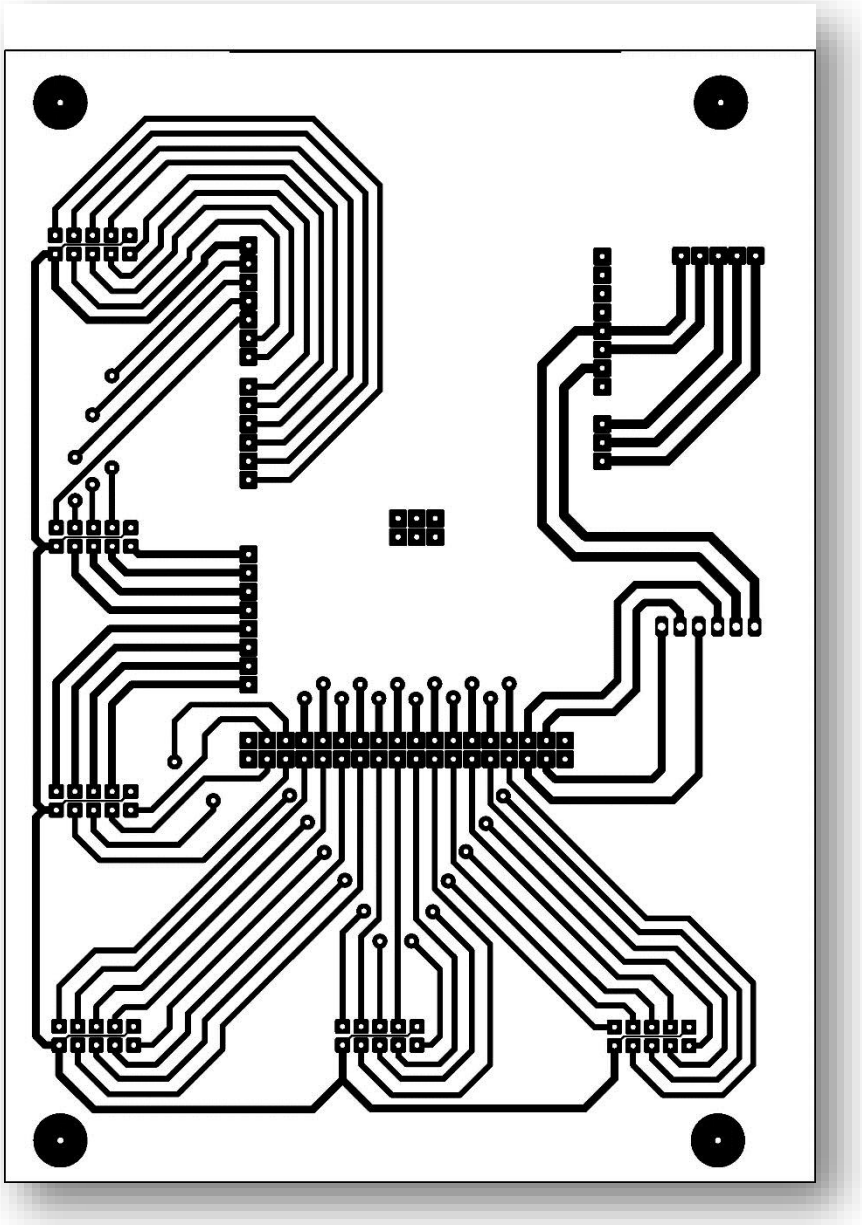


Figura A.2. *Pistas entradas y salidas de señales del sistema electrónico.*

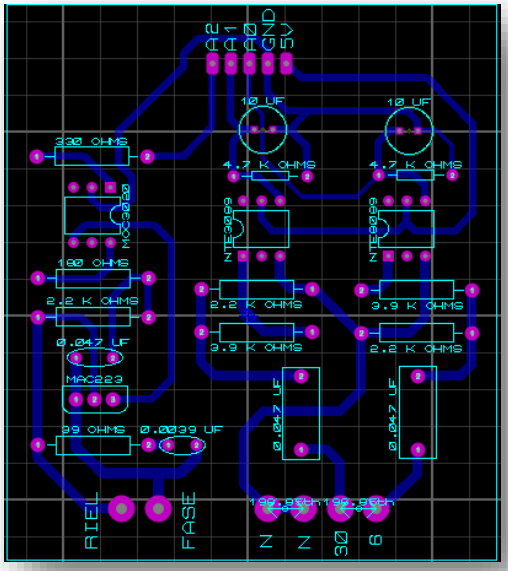


Figura A.3. *Vista previa placa para etapa de potencia en Ares Proteus.*

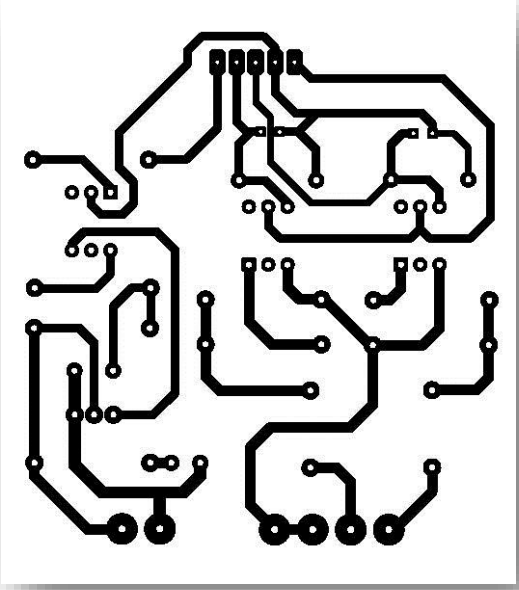


Figura A.4. *Pistas entradas y salidas de señales para etapa de potencia.*

Anexo B: Código fuente Arduino

En las siguientes páginas se incluye la programación en arduino tanto como la descarga del archivo y la reproducción de música:

```
#include<SD.h>

//:::::::::::::::DATOS PARA LA DESCARGA::::::::::::::::::::
File Myfile;    //variable con el que se crea el archivo de texto.
char filename[10]; //nombre del archivo
int num = 0;
int fin = 0;
int retardo = 0;
String inbox;
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

//:::::::::::::::DATOS PARA REPRODUCTOR:::::::::::::::::::::::
int E6; // Activa modo Musica
int E30; // Reproduce la cancion

int Musica = 0;
int Play = 0;
int actualizar = 0;
int tiempoRestante = 25;
int numcancion = 1;
File archivo;
int totalBytes;
int UltimaPosicion = 0;
String cadena = "";
char caracter;
String tiempo = "";
int LED1, LED2, LED3, LED4;
int Horas = 0;
int totalcanciones;
```

```
int contador = 0;
char cancion[10];

int PinLeds[48] = {2, 3, 4, 5, 6, 7, 8, 9, 10,
                   11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
                   23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
                   35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46,
                   47, 48, 49
                  };
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::



void setup() {
  Serial.begin(9600);
  Serial.println("Iniciando Tarjeta SD.....");
  if (!SD.begin(53)) {
    Serial.println("No se pudo iniciar la tarjeta SD!");
    return;
  }
  Serial.println(" Tarjeta SD iniciada correctamente!");
  Serial.println(" ");

  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, OUTPUT);



  for (int i = 0; i <= 47; i++) {
    pinMode(PinLeds[i], OUTPUT);
    digitalWrite(PinLeds[i], LOW);
    delay(20);

  }
```

```
}


void loop() {


  if (Serial.available() > 0)
  {
    Serial.flush();
    inbox = Serial.readString(); // byte dato


    num = num + 1;
    sprintf(filename, "%d.txt", num); //nombre del archivo 1.txt,
2.txt ...


    SD.remove(filename);
    Myfile = SD.open(filename, FILE_WRITE);



    if (Myfile)  //si el archivo existe
    {
      Myfile.print(inbox);  //Se escribe los datos recibidos en el
archivo de texto
      Myfile.flush();
      Myfile.close();    //Se cierra para finalizar
      fin = 1;
      inbox = "";
      delay(100);
    }
  }



  if (fin == 1)
  {
    delay(1000);
    Serial.println(fin);
```

```
    Serial.flush();
    fin = 0;
  }

  ModoMusica();
  TM();
}
//:::::::::::::::::::::::::::::::::::::::::::::::::::::::
void TM()
{

  retardo = retardo + 1;
  delay (100);

  if (retardo >= 3600)
  {
    num = 0;
    retardo = 0;
  }

}

//:::::::::::::::::::::::::::::::::::::::::::::::::::::::

void ModoMusica()
{
  E6 = digitalRead(A0);
  E30 = digitalRead(A1);
  //.....................................................
  if (E6 == HIGH)
  {

    Musica = 1;
    // Serial.println("Modo música activado");
```

```
    delay(3000);
  }



  if (Musica == 0 && E30 == 1)
  { numcancion = 1;
  }
  if (Musica == 1 && E30 == 1)
  {
    actualizar = 1;
  }
  //....................................



  //:::::::::::::::COMIENZA MODO MUSICA::::::::::::...

  if (actualizar == 1)
  {

    contador = contador + 1;

    if (contador >= 100)
    {
      digitalWrite(A2, HIGH ); //Señal de activacion del Riel
      delay(500);
      digitalWrite(A2, LOW);
      Play = 1;
      actualizar = 0;

    }
    // Serial.println(contador);
    delay(100);
  }
```

```
  if (Play == 1)
  { MusicaPlay();

  }


}

//:::::::::::::::::::::::::::::::::::::::::::::::::
void MusicaPlay()
{

  sprintf(cancion, "%d.txt", numcancion); //nombre del archivo
1.txt, 2.txt ...

  archivo = SD.open(cancion);
  totalBytes = archivo.size();

  if (archivo)
  {

    if (UltimaPosicion >= totalBytes)
    {
      UltimaPosicion = 0;
      delay(100);

    }



    archivo.seek(UltimaPosicion); // Nos ubicamos en una posición
específica en el archivo, inicialmente: posicion cero

    while (archivo.available())
```

```
    {
      caracter = archivo.read();
      cadena = cadena + caracter;
      UltimaPosicion = UltimaPosicion + 1;

      if (caracter == 10)
      {

        break;
      }
    }

    archivo.close();

    tiempo = cadena;
    tiempo.remove(4);

    String N1, N2, N3, N4;
    N1 = cadena;
    N2 = cadena;
    N3 = cadena;
    N4 = cadena;

    N1.remove(0, 4);
    N1.remove(2, 8);
    LED1 = N1.toInt() + 1;

    N2.remove(0, 6);
    N2.remove(2, 6);
    LED2 = N2.toInt() + 1;

    N3.remove(0, 8);
    N3.remove(2, 4);
    LED3 = N3.toInt() + 1;
```

```
  N4.remove(0, 10);
  N4.remove(2, 2);
  LED4 = N4.toInt() + 1;

  digitalWrite (LED1, HIGH);
  digitalWrite (LED2, HIGH);
  digitalWrite (LED3, HIGH);
  digitalWrite (LED4, HIGH);

  delay(200);

  digitalWrite (LED1, LOW);
  digitalWrite (LED2, LOW);
  digitalWrite (LED3, LOW);
  digitalWrite (LED4, LOW);
  delay(tiempo.toInt());
  cadena = "";
  if (UltimaPosicion >= totalBytes)
  {
    UltimaPosicion = 0;
    delay(100);
    //Serial.println("Stop ||");
    Play = 0;
    Musica = 0;
    contador = 0;
    numcancion = numcancion + 1;
    if (numcancion >= 16)
    { numcancion = 1;
    }
  }
}
else {
  //Serial.print("no existe! :(");
```

```
    Play = 0;
    Musica = 0;
    contador = 0;
    numcancion = numcancion + 1;
    if (numcancion >= 16)
    { numcancion = 1;
    }
  }
}
```

Anexo C: Fotos

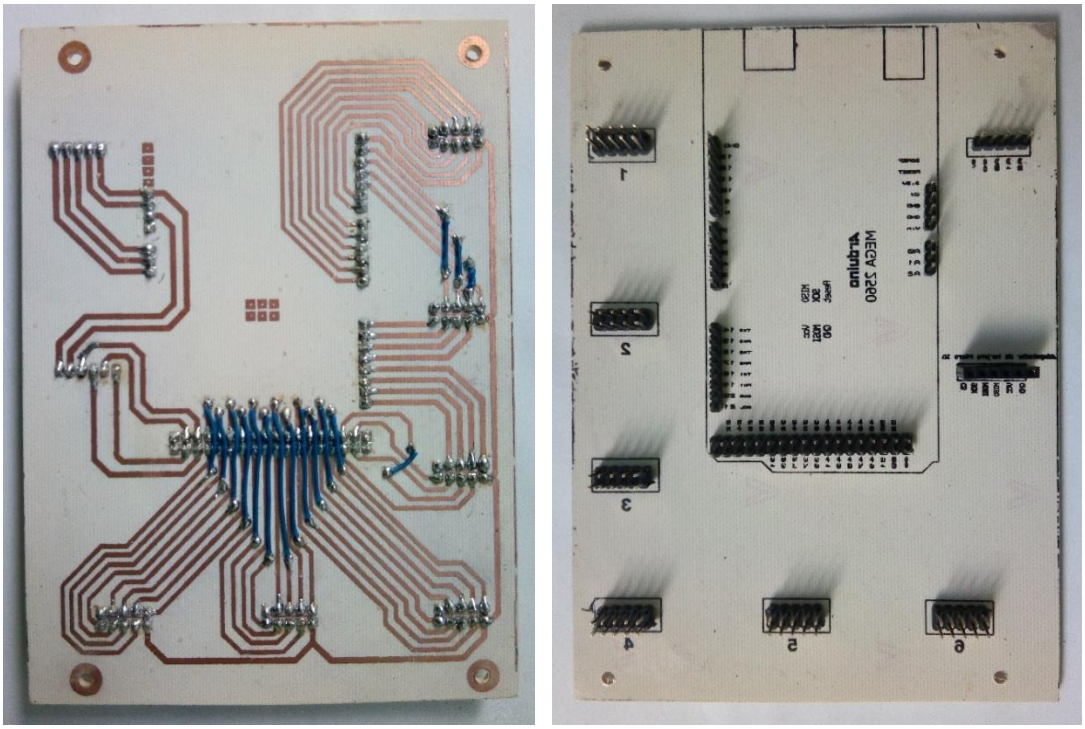


Figura C.1. *Placa sistema de reproducción de música.*

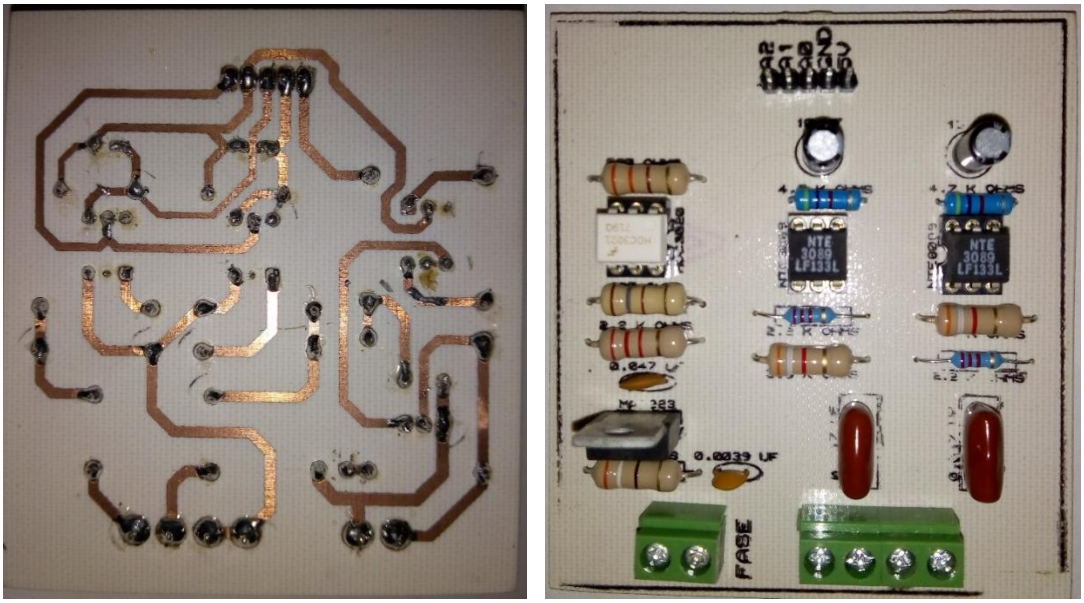


Figura C.2. *Etapa de potencia para activación del riel y entradas de señales 6 y 30.*

Anexo D: Contenido archivo digital

En este trabajo se anexa un disco en el que podrá encontrarse:

- ✓ Código del programa arduino.
- ✓ Diagramas electrónico.
- ✓ Diagrama de bloques.
- ✓ Manual de los optoacopladores usados para el desarrollo del proyecto.
- ✓ Archivo con las placas diseñadas en PCB.
- ✓ Programa iCatedral instalable.
- ✓ Ejemplos de melodías en archivos .txt.
- ✓ Programas para poder abrir estos archivos.
- ✓ Fotos de las placas diseñadas.
- ✓ Video.