UNIVERSIDAD DE LAS AMÉRICAS PUEBLA

Escuela de Ingeniería

Departamento de Computación, Electrónica y Mecatrónica

Inversor multinivel de 5 fuentes CD para aplicación de coche eléctrico solar

Tesis que, para completar los requisitos del Programa de Honores presenta la estudiante

Ana Karen Graciano Alvarez

166181

Ingeniería en Mecatrónica

Dr. Pedro Bañuelos Sánchez

San Andrés Cholula, Puebla. Primavera 2024

Tesis que, para completar los requisitos del Programa de Honores presenta el estudiante Ana Karen Graciano Alvarez 166181

Director de Tesis

Dr. Pedro Bañuelos Sánchez

Presidente de Tesis

Dr. José Luis Vázquez González

Secretario de Tesis

Dr. Cesar Martínez Torres

Resumen

En esta tesis se diseña un inversor multinivel de configuración puente H en cascada con cinco fuentes de voltaje CD. Se utiliza un esquema de conmutación para controlar los interruptores de potencia necesarios. Se presenta un método en el que los ángulos de conmutación se calculan de forma que se produzca un voltaje deseado y eliminar al mismo tiempo ciertos armónicos. Utilizando las series de Fourier, las ecuaciones que eliminan los armónicos se derivaron en términos de los ángulos de conmutación. Se utilizó Matlab para resolver las ecuaciones y obtener los ángulos. Con esto, se obtuvo la secuencia de cambios para los interruptores, primero en términos angulares y después en tiempo, utilizando una frecuencia de 60Hz. De ahí se estudió controlar los luego con el Arduino Mega 2560, con una configuración con optoacopladores para evitar el corto circuito. Con la programación, se consiguió controlar puente Hs de diferentes niveles, de 3 a 4 a 5 niveles. Por último, se usaron tres paneles solares para probar el concepto.

Índice

Capítulo 5. Introducción

5.1. Antecedentes

La movilidad personal se ha vuelto un tema importante en la sociedad, y una de las problemáticas es el tema de la sustentabilidad de sistema de transporte y la accesibilidad. En la actualidad, muchos modos de transporte consumen una cantidad considerable de recursos energéticos y, además, contribuyen a las emisiones de gases de invernadero. Por lo cual, se ha recurrido a la electrificación de la movilidad, usando energía proveniente de recursos renovables, permitiendo reducir las emisiones y lograr hacer eficiente el uso de energía [1].

Los retos que tiene la electrificación de los vehículos, es la autonomía máxima, el tiempo de recarga y el costo, los cuales son mejores en un motor de combustión interna. Estas características han mejorado a lo largo del tiempo, como ejemplo se han ido desarrollando tecnologías de carga rápida, y han mejorado la capacidad de las baterías. Incluso, las manufactureras han ido migrado a un sistema de mayor clasificación de voltaje (de 400V a 800V), lo cual mejora el rendimiento, logra tiempos de carga más rápido y disminuye la pérdida de energía y permite viajar en trayectorias más largas [1].

Aunque hacer este cambio de sistema trae mayores beneficios, es necesario tener un inversor adecuado que permita no tener pérdidas de conmutación con mayores voltajes y que no eleve los costos. El inversor que se estaba usando es uno trifásico de seis conmutadores y dos niveles, el cual es inadecuado para un voltaje de enlace de 800V CC. Por lo cual, la búsqueda de nuevos inversores ha sido relevante para la electromovilidad. Una de las tecnologías prometedoras es el inversor multinivel, el cual ha sido apto para aplicaciones de

alta potencia, como en plantas de energía eléctrica, o en áreas que requieren de una corriente sinusoidal de alta calidad. Un ejemplo de aplicación para los inversores sería las celdas fotovoltaicas, las cuales pueden ser utilizadas para los vehículos eléctricos. En este caso, su función, es ser de los intermediarios principales para convertir la radiación solar a energía eléctrica en forma de corriente directa y se han utilizado para aplicaciones CD a CD como para la carga de baterías o motores de corriente directa. Este sistema acoplado con un inversor puede usarse para suministrar a cargas de corriente alterna. El uso de inversores multinivel permite para las diferentes aplicaciones conseguir alta eficacia, alta densidad de potencia, mejor calidad de onda (menor THD), frecuencia baja de conmutación, y tener tolerancia a falla [1], [2].

5.2. Planteamiento del problema

La generación de energía y los sistemas de distribuciones han recurrido a fuentes de energía renovable como la solar. El consumo de energía ha ido incrementando, aunque la desventaja de las energías limpias es que tienen una salida que no tienen una alta calidad sinusoidal, causando que no sea adecuado para los consumidores. Es importante que la señal mantenga la calidad, para que se pueda aprovechar y no genere desgastes. Por lo tanto, en aplicaciones como vehículos eléctricos y en las redes fotovoltaicas, se han estado investigando soluciones para vencer sus limitaciones, incluyendo su autonomía máxima y el largo tiempo de recarga [1] [3].

El presente trabajo, busca generar una onda con baja cantidad de armónicos, alta calidad, y alta eficiencia a partir de la energía renovable solar para poder usar en un coche

eléctrico. Por lo tanto, se desarrolla un inversor, que pueda ser capaz de transformar la corriente directa proveniente de los paneles solares, a corriente alterna que pueda utilizar el motor de un coche eléctrico.

5.3. Alcances y Limitaciones

El uso de los inversores multinivel en general han demostrado ser una tecnología útil para aplicaciones de media a alta potencia. Se ha reportado que han alcanzado en su funcionamiento valores de hasta 13.8kV y 100MW. También tienen la característica de disminuir armónicos, tener alta velocidad de conmutación y una buena calidad sinusoidal. Características que pueden ser deseables en la industria automotriz y eléctrica [1] [3].

En este trabajo se busca estudiar un inversor multinivel simple: uno con 5 puentes H en cascada. Esta estructura no necesariamente se usaría en la industria, ya que es mejor trabajar con más niveles para reducir aún más los armónicos. Además, tiene la desventaja que las fuentes no se usan equitativamente, por lo que causaría que se gastaran unas más rápido que otras. Adicionalmente, en la industria convendría trabajar con corriente alterna trifásica. No obstante, el diseño propuesto es útil para entender el concepto del inversor multinivel y sus ventajas. Así como, tener un diseño flexible para usar con Arduino.

5.4. Descripción del sistema

Para cada nivel en el inversor, se utilizó de una fuente de voltaje proveniente de los paneles solares, cuatro interruptores, dos de ellos Mosfets tipo P y los otros dos tipo N. Además, un puente tenía dos señales digitales provenientes del Arduino Mega 2560, que se encargaba de

controlar la conmutación, y dos optoacopladores que aislaban la parte digital del Arduino de la parte del puente. Por último, contaba con resistencias: dos de 1kΩ y dos de 330 Ω. En total eran cinco puentes, teniendo en la salida una carga resistiva de 100Ohms.

5.5. Objetivos

En el presente trabajo, se buscó cumplir con lo siguiente.

- Primero entender el concepto de un inversor multinivel.
- Resolver un sistema de ecuaciones para obtener el ángulo de disparo en Matlab.
- Determinar las señales digitales para cada interruptor, usar Arduino para crear estas señales.
- Unirlo con el puente H en cascada y los optoacopladores.
- Analizar su comportamiento.
- Probar su comportamiento con paneles solares.

Capítulo 6. Inversores

6.1. Convertidores y conmutadores

Hay cuatro tipos de conversiones para el control energético de energía, basados en sistemas de corriente alterna (CA) y de corriente directa (CD), las cuales son: rectificación, inversión, conversión CD-CD, y conversión CA-CA. El primero se refiere a convertir la energía de CA a CD, el segundo de CD a CA, el tercero de CD a CD y el último de CA a CA [4].

Hay diferentes circuitos inversores que se pueden usar, ya sea para una fase o para tres, se podrían clasificar como aquellos que están conmutados externamente o que hacen autoconmutación. Tanto en los inversores como en los rectificadores, se usan interruptores eléctricos que controlan la conmutación. Al principio, se usaban interruptores operados mecánicamente, y en la actualidad se usan los semiconductores. Para los inversores conmutados externamente, se podrían usar tristores. En el caso de autoconmutados, dependen de dispositivos de apagado como transistores bipolares, MOSFETS (*metal–oxide–semiconductor field-effect transistor*), GTOs (*metal–oxide–semiconductor field-effect transistor*) e IGBTs (*insulated-gate bipolar transistor*), los cuales se apagan y prenden dependiendo de las corrientes en la base. Los transistores bipolares e IGBTs son útiles para aplicaciones de potencia bajas a medianas, y los GTOs para potencias altas. Mientras, los MOSFETs son los más rápidos y se pueden usar para frecuencias altas [4].

En los inversores conmutados externamente, se necesita de una fuente de voltaje adicional, que está separado del conmutador. Este va a proporcionar los voltajes de conmutación. Puede haber dos casos para esto: conmutación por línea o conmutación por carga. En el primero, la fuente de voltaje es el sistema de suministro, y en el último es en la

carga. Aplicaciones incluye accionamiento de motores CD de velocidad controlado con modos de frenado regenerativo y de reversa, y transmisión de potencia altas con estaciones lejanas [4].

Por otro lado, en la autoconmutación, se lleva a cabo al prender y apagar los interruptores con pulsos en la base y no depende de voltaje alterno en el sistema de suministro o en la carga. Este tiene más aplicaciones, pues se usa en las diferentes áreas de ingeniería eléctrica. Principalmente, se usa como tecnología de accionamiento en la industria, en electrodomésticos y en automóviles. Adicionalmente se ha visto avances en los inversores, con control vectorial, para utilizar en máquinas CA de velocidad controlada. El control vectorial se ha integrado con PWM y microprocesadores, para utilizar en aplicaciones de servicios públicos [4].

Hay diferentes configuraciones para los inversores autoconmutados; ejemplos incluyen inversores paralelos con conmutación por condensador, inversor McMurray-Bedford, inversor McMurray, inversor controlado por PWM, inversor multinivel e inversor resonante [4]. Como se ha mencionado, en este trabajo se usa el inversor multinivel, en el cual se detalla su funcionamiento en el siguiente apartado.

6.2. Inversor Multinivel

El inversor multinivel es útil para el área de energías renovables, por la necesidad de alta potencia, accionamiento de velocidad regulable, y sistemas de transmisión de CA flexible. Tradicionalmente se usaba transformadores para alcanzar los voltajes requeridos y para tener señales de multipulso en la salida. La desventaja de este componente es que es voluminoso,

es caro, y tienen limitaciones en su sistema de control por tener propiedades no lineales y de saturación. Entonces, el inversor multinivel es una configuración versátil, que permite alcanzar un alto voltaje y reduce armónicos sin la necesidad de transformadores. Por lo cual, lo vuelve una tecnología de alta eficiencia, compacta y de bajo costo [5].

En general, esta tecnología se basa en una conmutación alternada por interruptores (los semiconductores), para obtener niveles de voltaje CD que representan una señal CA con armónicos bajos. Hay diferentes tipos de topología. Entre las tradicionales se encuentra: puentes H en cascada, con diodos anclados, y con capacitor flotante. La más simple es el primero, ya que tiene menos componentes y modularidad. Además, es el más usado para la aplicación en energía renovable [5][6]. También puede haber combinaciones como convertidor modular y convertidores de multicelda apilados [5]. A lo largo de los años, se han ido proponiendo modificaciones para crear nuevas topologías, las cuales alteran la estructura original. Estas configuraciones se pueden clasificar como inversores multinivel: simétricos, asimétrico e híbridos. Como ejemplos, para el primero se encuentran: el interruptor reducido bidireccional (RSBMLI), el tipo T, y el de enlaces multinivel CD (MLDCL); y para el segundo están el inversor de fuentes de conexión cruzada (CCSMLI), el conmutado serie/paralelo CD (SSSDCMLI) y el tipo E. Entre ellos, pueden estar divididos entre aquellos con puente H o sin [7]. En los Anexos (Figura A.1), se encuentra el desglose de diferentes tipos de inversores, da una idea del desarrollo de la tecnología. A continuación, se hace la descripción de los inversores multinivel tradicionales.

En el inversor multinivel de diodos anclados, también llamado punto neutral anclado, el voltaje de alimentación está dividido en niveles definidos, usando condensadores en serie. Esta tecnología fue inventada en 1975. La cantidad de capacitores (c) dependen del número

de niveles (n), la cual está definida por la siguiente ecuación c=n-1. Mientras, la cantidad de diodos de enclavamiento (d) está dado por d=(n-1)(n-2) y el número de interruptores=2(n-1). La función de estos es bloquear el voltaje del capacitor y limitar estrés de los semiconductores. La estructura de esta se ve en la Figura 6.1, como son de tres niveles, hay dos capacitores, dos diodos de enclavamiento y cuatro interruptores. El punto neutro está localizado entre los dos capacitores. La salida puede tener tres valores: V_{c1}, 0 o $-V_{c2}$. En teoría, se puede usar esta topología para cualquier nivel de voltaje en dispositivos de bajo voltaje [2] [5] [6].

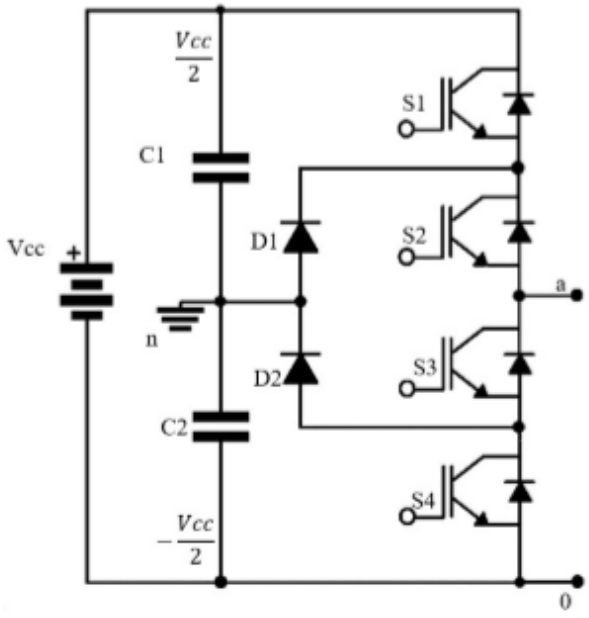


Figura 6.1. Diagrama de inversor multinivel de diodo anclado de tres niveles [6].

Las ventajas de esta topología es que es buena opción para convertidores de tres niveles pues ha tenido éxito hasta la fecha, es un método sencillo de modulación multiportadora basado en PWM con desplazamiento de nivel, y, además, tiene la posibilidad de inyección de secuencia cero para realizar el control de equilibrio de voltaje del enlace de CD. Las desventajas son: que hace uso no equitativo de los dispositivos de potencia que

provoca un envejecimiento imparejo y es complejo conseguir más de tres niveles debido a la imposibilidad de conseguir el control de equilibrio de tensión del enlace de CC. Aplicaciones típicas incluye FACTS (*Flexible AC transmisión systems*) y accionamientos de motores para bombas, ventiladores compresores y cintas transportadoras [8].

En el inversor multinivel con capacitores flotantes, se reemplaza los diodos anclados de la topología pasada, por capacitores auxiliares. La cantidad de estos componentes, está definido con 0.5*(n-1)(n-2) por fase. Mientras la cantidad de interruptores se da por 2(n-1) y los capacitores por n-1. Como ejemplo, la Figura 6.2, es de tres niveles y tiene un capacitor flotante más otros dos capacitores y cuatro interruptores. En general, puede generar salidas de voltaje de 3, 5 y 7 niveles usando una sola fuente. También puede trabajar con la técnica de SPWM (PWM senoidal) y SPWM multicarrier. Esta topología puede ser incrementada al aumentar los componentes (interruptores y capacitadores) o por conectar el multinivel en cascada [2] [5] [6].

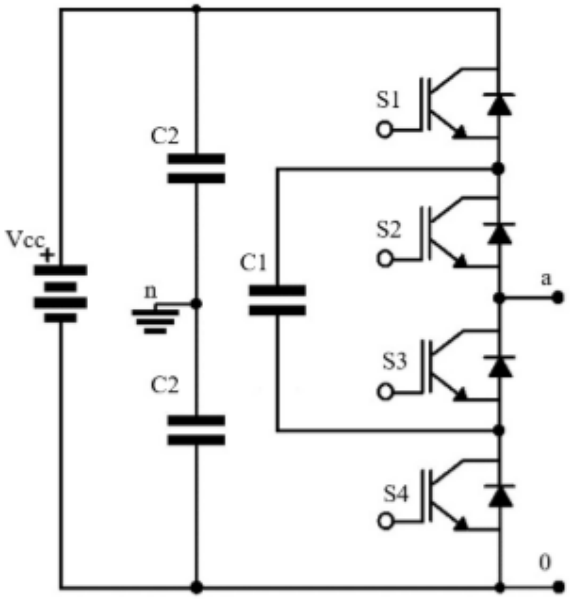


Figura 6.2. Diagrama de inversor multinivel de capacitores flotantes de tres niveles [6].

La ventaja de esta configuración es que puede funcionar y permite control natural de voltajes con el método de desplazamiento de fase de PWM, lo cual permite tener ondas de salida de alta calidad con bajas pérdidas de potencia por celda; también tiene envejecimiento equitativo y conexiones simples de celdas de potencia, alcanzando alta modularidad. Las desventajas es que tienen un gran número de condensadores, ocasionando una reducción de la vida útil del convertidor. Además, los condensadores no están cargados continuamente, lo cual no permite que los niveles de voltaje sean equilibrados, tiene un bajo rendimiento dinámico y requiere una alta frecuencia de salida. Para poder mejorar esta topología, se podría añadir un sistema de control complejo y, además, se debería de cargar los condensadores antes de usar el inversor. Esta consideración significa que podría tener un arranque lento. Aplicaciones típicas son FACTS y accionamientos de motores, principalmente de tracción [5] [6] [8].

Por último, en el puente H conectado en cascada, las celdas están conectados en serie, estando alimentada por una fuente de voltaje CD independiente, esto permite eliminar el uso de diodos o capacitores de enclavamiento. Se puede definir cada puente como un inversor de tres niveles, dando los tres voltajes: V_c, 0 o $-V_c$. La cantidad de celdas y fuentes de voltaje está dada por 0.5*(n-1), mientras el número de interruptores está dado por 2(n-1), donde n es el número de voltajes de salida. La topología también puede ser usado con las diferentes técnicas de PWM. El número de niveles puede ser incrementado al usar fuentes asimétricas, ya que puede dar más niveles con el mismo número de componentes. En la Figura 6.3, está un ejemplo de esta configuración. En este caso, se trata de un inversor de 5 niveles, ya que puede dar 5 voltajes diferentes, está constituido por dos puentes H o celdas, por dos fuentes de voltaje CD, y por 8 interruptores [2] [5] [6].

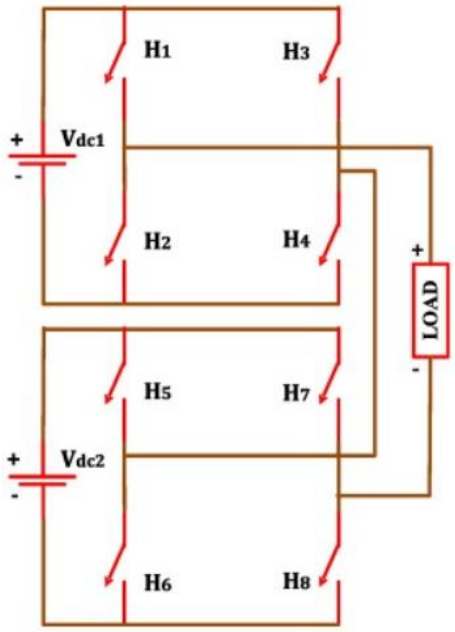


Figura 6.3. Inversor multinivel de puentes H conectado en cascada de 5 niveles [7].

Esta topología tiene como ventajas, que tiene una alta modularidad por su característica de tener los puentes conectados en serie, tiene la capacidad de tolerancia a fallos directa añadiendo interruptores de bypass, es capaz de alcanzar voltajes de CD altas al ir añadiendo puentes y también puede ser operado por PS-PWM alcanzando salidas de onda con alta rendimiento y con ecualización de pérdida de potencia. Como desventaja, es que se necesita de fuentes de voltaje CD por cada puente, propiedad que los otros convertidores no necesitaban, y la estructura puede ser compleja si se requiere una configuración *back-to-back*. Las aplicaciones típicas son FACTS, accionamientos de motores y sistemas fotovoltaicos [8].

En el siguiente capítulo, se detallará el funcionamiento de esta topología.

Capítulo 7. Inversor multinivel de puente H conectados en cascada

7.1. Configuración

Para el puente H en cascada, existe la configuración conocida como el convertidor de puente completo. En la Figura 7.1, se observa una fuente de voltaje CD, cuatro interruptores (S_1, S_2, S_3, y S_4), las diferentes corrientes (i_S, i_{S_1}, i_{S_2}, i_{S_3}, i_{S_4} y i_o) y el voltaje de salida (v_0). Dependiendo de que pares de interruptores se cierran, se obtiene un voltaje de salida. [9]. Para tomar en cuenta las opciones que existen, se utiliza la ecuación para las combinaciones sin repetir elementos: $\frac{n!}{r!(n-r)!}$, donde n es el número de elementos y r es cuántos elementos hay en la combinación. [10]. En este caso, n son los cuatro interruptores y r es dos, ya que se van a cerrar los interruptores en pares, por lo cual habría combinaciones igual a $\frac{4!}{2!(4-2)!} = 6$.

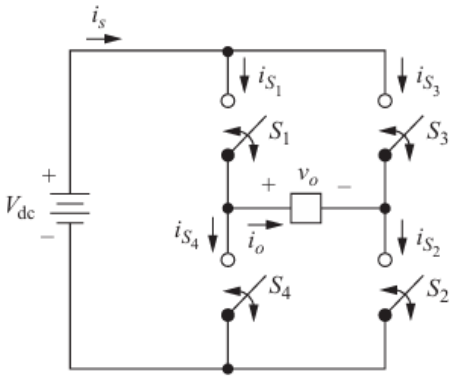


Figura 7.1.Convertidor de puente completo [9].

Las combinaciones se ven en la Tabla 7.1. Sin embargo, aunque teóricamente existen las 6 combinaciones, solo 4 de ellas son permitidas. Es decir, las combinaciones de S_1 & S_4

y S_2 & S_3 causan corto circuito a través de la fuente de voltaje, por lo que no se podrían usar para el inversor sin que se dañen los componentes. Las 4 opciones restantes, permiten tener un voltaje de salida de: $+V_{dc}$, $-V_{dc}$ o 0V. Estas combinaciones se pueden ver en el circuito en la Figura 7.2Figura 7.2. Combinaciones permitidas para cerrar par de switches [9].. Resumiendo, la información: el primer voltaje es resultado de cerrar S_1 y S_2, el segundo voltaje es de cerrar S_3 y S_4, y los 0V pueden ser causados por cerrar S_1 y S_3 o por cerrar S_2 y S_4 [9].

Tabla 7.1. Combinaciones para cerrar los interruptores en pares

Parejas de interruptores cerrados	Salida v_0
S_1 & S_2	$+V_{dc}$
S_1 & S_3	0 V
S_1 & S_4	Corto circuito
S_2 & S_3	Corto circuito
S_2 & S_4	0 V
S_3 & S_4	$-V_{dc}$

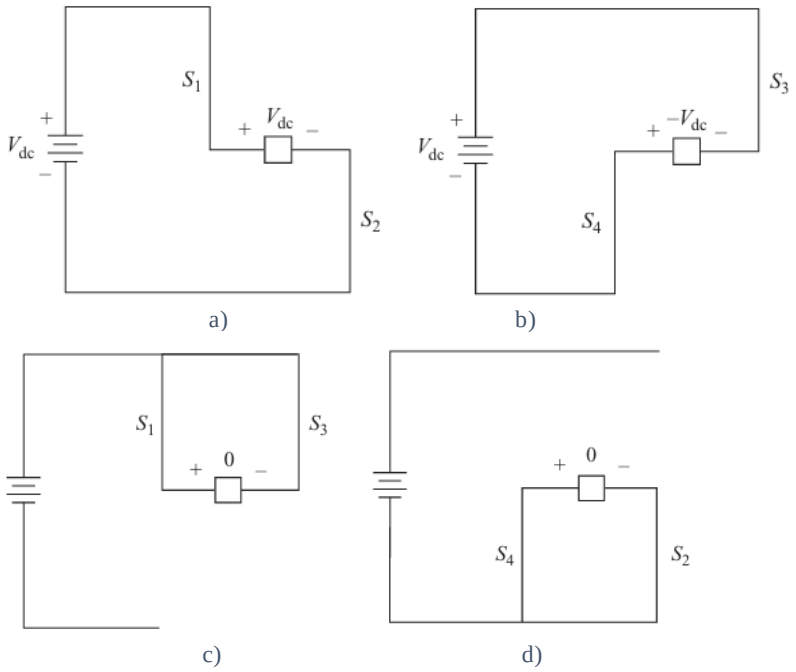


Figura 7.2. Combinaciones permitidas para cerrar par de switches [9].

Dependiendo de qué carga se coloca en la salida, es que se obtiene la forma de la señal. Al tener una carga resistiva, produce una señal cuadrada [9].

7.2. Análisis Fourier

El teorema de Fourier describe que para una función periódica $v_0(t)$, se puede describir de acuerdo a la ecuación 1[11].

$$v_o(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(n\omega t) + b_n \sin(n\omega t)) \qquad (1)$$

Donde $\frac{a_0}{2}$ es el valor promedio del voltaje de salida, y a_n, a_0 y b_n son constantes que se definen como: $a_0 = \frac{2}{T}\int_0^T v_0(t)dt$, $a_n = \frac{2}{T}\int_0^T v_0(t)\cos(n\omega t)dt$ y $b_n = \frac{2}{T}\int_0^T v_0(t)\sin(n\omega t)dt$.

En el caso de una simetría de media onda, donde la media onda negativa y la media onda positiva son iguales, pero con un desfase de T/2, a_n y a_0 son iguales a 0. Adicionalmente, la señal cuadrada pose está simetría y contiene solo armónicos impares. Su voltaje rms queda como $V_{rms} = \left(\frac{2}{T}\int_0^{\frac{T}{2}} V_s^2 dt\right)^{0.5} = V_s = v_0$. Lo cual significa que b_n y $v_o(t)$ queda como se muestran en las ecuaciones 2 y 3 [11].

$$b_n = \frac{2}{\pi}\left[\int_{-\frac{\pi}{2}}^{0} -V_s \sin(n\omega t)\, d(\omega t) + \int_0^{\frac{\pi}{2}} V_s \sin(n\omega t)\, d(\omega t)\right] = \frac{4V_s}{n\pi} \tag{2}$$

$$v_0(t) = \sum_{n=1,3,5,..}^{\infty} \left(\frac{4V_s}{n\pi}\sin(n\omega t)\right) \tag{3}$$

En el caso de la señal cuadrada de un inversor de puente completo, esta depende del voltaje de entrada y del control de la conmutación. Hay un intervalo α ajustable, para cuando la salida es 0V, +Vdc o -Vdc. Este intervalo se observa en la Figura 7.3, y es conocido como el ángulo de voltaje cero en cada pulso. Su voltaje rms queda como $V_{rms} = \left(\frac{1}{\pi}\int_{\alpha}^{\pi-\alpha} V_s^2 dt\right)^{0.5} = V_s\left(1 - \frac{2\alpha}{\pi}\right)^{0.5}$. Lo cual significa que b_n y $v_o(t)$ quedan como las ecuaciones 4 y 5 respectivamente [9].

$$b_n = V_n = \frac{2}{\pi}\int_{\alpha}^{\pi-\alpha} V_s \sin(n\omega t)\, d(\omega t) = \frac{4V_s}{n\pi}\cos(n\alpha) \tag{4}$$

$$v_0(t) = \sum_{n=1,3,5,..}^{\infty} \left(\frac{4V_s}{n\pi}\cos(n\alpha)\sin(n\omega t)\right) \tag{5}$$

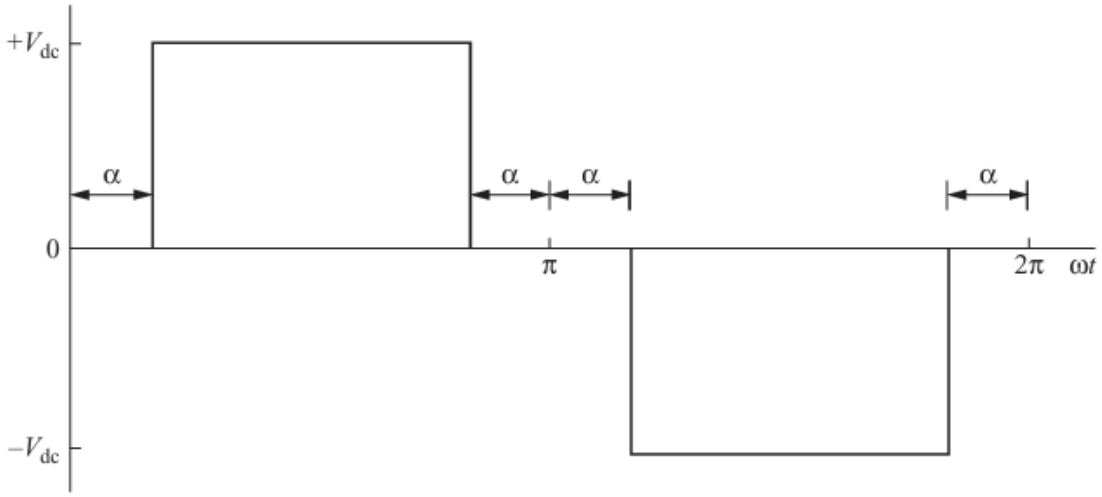


Figura 7.3. Salida del inversor con control de α [9].

La amplitud de cada frecuencia o armónico, está restringida por α. Adicionalmente, también es posible eliminar un armónico, por ejemplo, si se quiere eliminar el tercer armónico, α será igual a 30°. La regla general para eliminar un armónico está dada por α=90/n. La desventaja de este método es que no se puede controlar al mismo tiempo la amplitud de las frecuencias y la eliminación de armónicos [9].

Para un inversor multinivel en cascada, se obtienen niveles de salida de voltaje, los cuales van a tener mayor parecido a una señal senoidal y reducir el contenido de armónicos. Esta configuración utiliza fuentes de voltaje independientes para cada nivel. La salida para un inversor de dos fuentes ahora queda como $2V_{dc}$, V_{dc}, 0V,$-V_{dc}$ y $-2V_{dc}$. Para un puente de k fuentes, existen 2k+1 posibles voltajes en la salida. Al ir añadiendo más, hay más pasos, provocando una señal parecida a una escalera que se acerca más a la señal senoidal [9].

Cada puente opera con un ángulo de retardo α. En la siguiente figura se observa el resultado de un puente de dos niveles con un α_1 y α_2 Las últimas dos señales corresponden a cada puente usado; se puede observar que cada uno maneja su propio α, y en la salida total se ve el resultado de las dos α [9].

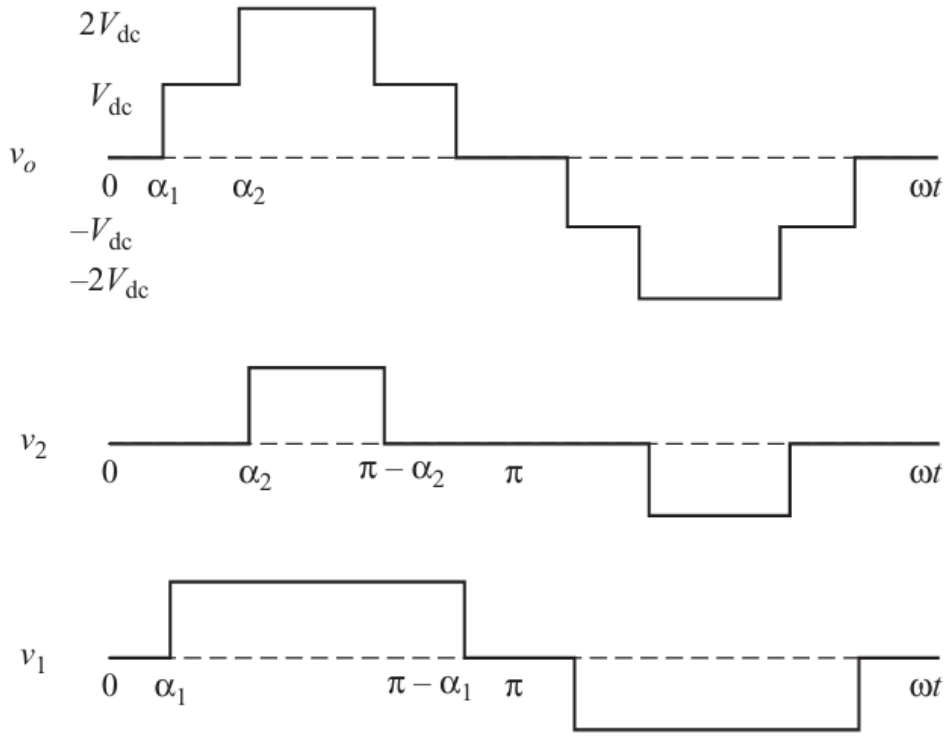


Figura 7.4. Salida total del inversor y de cada puente en un sistema de dos fuentes de voltaje [9].

Para este ejemplo, al tener dos α diferentes la serie de Fourier queda como: $v_0(t) = \sum_{n=1,3,5,..}^{\infty}(\frac{4V_s}{n\pi}[\cos(n\alpha_1) + \cos(n\alpha_2)]\sin(n\omega t))$. Mientras, para k niveles de voltaje la serie queda como: [9]

$$v_0(t) = \sum_{n=1,3,5,..}^{\infty}\left(\frac{4V_s}{n\pi}[\cos(n\alpha_1) + \cos(n\alpha_2) + \cdots + \cos(n\alpha_k)]\sin(n\omega t)\right) \qquad (6)$$

Para el control del inversor, existe el concepto de índice de modulación M_i, el cual es la razón entre la amplitud de las frecuencias fundamental de v_o y la amplitud de la frecuencia fundamental de una señal cuadrada con el voltaje máximo del voltaje (en el caso de dos niveles es 2Vdc). La ecuación queda como: $M_i = \frac{b_1\ (inversor)}{k * b_1\,(cuadrada)} =$

$\frac{\frac{4V_S}{1*\pi}[\cos(n\alpha_1)+\cos(n\alpha_2)+\cdots+\cos(n\alpha_k)]}{k*\frac{4V_S}{1*\pi}}$, donde k es el número de niveles. La expresión simplificada como:

$$M_i = \frac{\cos(n\alpha_1) + \cos(n\alpha_2) + \cdots + \cos(n\alpha_k)}{k} \tag{7}$$

Como ejemplo para uno de nos niveles, M_i es $\frac{\cos(n\alpha_1)+\cos(n\alpha_2)}{2}$ [9].

Es a partir de esto, que se pueden eliminar armónicos. La siguiente ecuación permite eliminar el armónico m para un inversor de dos niveles: $\cos(m\alpha_1) + \cos(m\alpha_2) = 0$. También se puede controlar la amplitud con el índice de modulación, por lo cual se usa la siguiente ecuación $\cos(\alpha_1) + \cos(\alpha_2) = 2M_i$. Esto forma un sistema de dos ecuaciones, que se puede resolver con un método como Newton-Raphson. Para un inversor de k fuentes de voltaje, se eliminan k-1 armónicos y se utiliza un M_i particular para hacerlo [9].

Capítulo 8. Diseño Teórico

8.1. Planteamiento del ejercicio

El inversor de cascada multinivel permite elegir los siguientes parámetros de diseño: nivel de fuentes de voltaje, armónicos a eliminar e índice de modulación. Para este trabajo, se sigue el ejemplo 8.7 planteado en el libro de Hart [9]. En el ejemplo se propone un inversor multinivel con 5 puentes H en cascada, dando 11 niveles de voltaje en la salida. Al tener estos puentes, significa que se pueden eliminar cuatro armónicos y existen cinco α por determinar. El sistema de ecuación simbólico aparece a continuación.

$$\cos(m_1\alpha_1) + \cos(m_1\alpha_2) + \cos(m_1\alpha_3) + \cos(m_1\alpha_4) + \cos(m_1\alpha_5) = 0$$

$$\cos(m_2\alpha_1) + \cos(m_2\alpha_2) + \cos(m_2\alpha_3) + \cos(m_2\alpha_4) + \cos(m_2\alpha_5) = 0$$

$$\cos(m_3\alpha_1) + \cos(m_3\alpha_2) + \cos(m_3\alpha_3) + \cos(m_3\alpha_4) + \cos(m_3\alpha_5) = 0 \qquad (8)$$

$$\cos(m_4\alpha_1) + \cos(m_4\alpha_2) + \cos(m_4\alpha_3) + \cos(m_4\alpha_4) + \cos(m_4\alpha_5) = 0$$

$$\cos(\alpha_1) + \cos(\alpha_2) + \cos(\alpha_3) + \cos(\alpha_4) + \cos(\alpha_5) = 5(M_i)$$

En el ejemplo 8.7, se propone un M_i como 0.8 y se busca eliminar los armónicos 5, 7, 11, y 13.

Por otro lado, en el circuito van a haber 20 interruptores y 5 fuentes CD. El circuito correspondiente a esto está en la Figura 8.1.

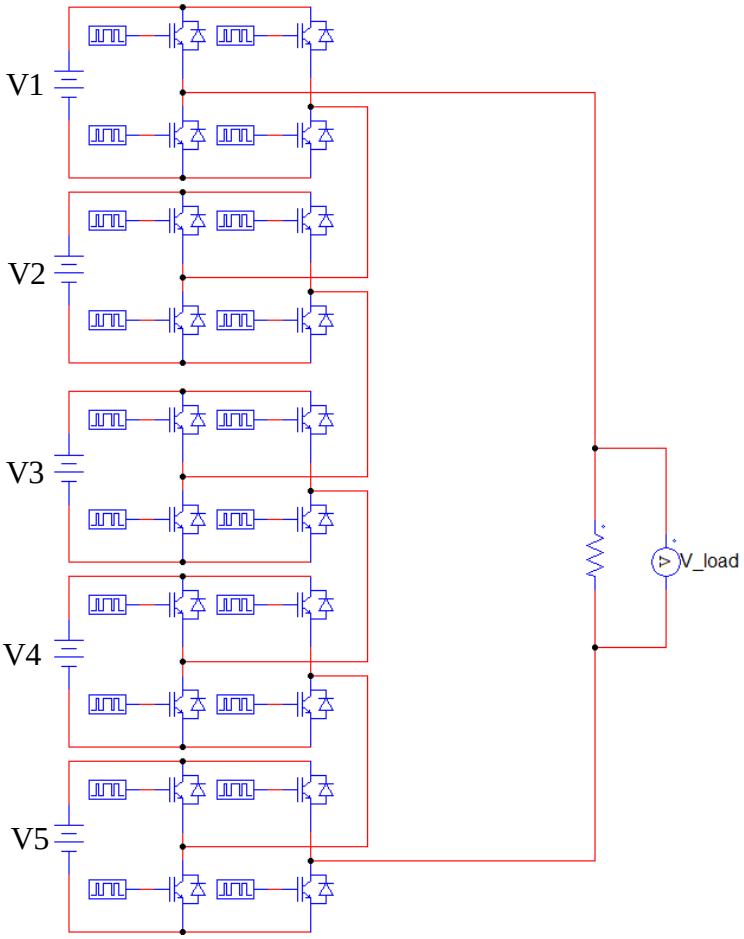


Figura 8.1. Configuración de inversor de 5 puentes H en PSIM.

8.2. Determinación de ángulos.

Teniendo los parámetros de diseño ya definidos, se realiza el siguiente sistema de ecuaciones para conseguir los ángulos de disparo.

$$\cos(5\alpha_1) + \cos(5\alpha_2) + \cos(5\alpha_3) + \cos(5\alpha_4) + \cos(5\alpha_5) = 0$$

$$\cos(7\alpha_1) + \cos(7\alpha_2) + \cos(7\alpha_3) + \cos(7\alpha_4) + \cos(7\alpha_5) = 0$$

$$\cos(11\alpha_1) + \cos(11\alpha_2) + \cos(11\alpha_3) + \cos(11\alpha_4) + \cos(11\alpha_5) = 0 \qquad (9)$$

$$\cos(13\alpha_1) + \cos(13\alpha_2) + \cos(13\alpha_3) + \cos(13\alpha_4) + \cos(13\alpha_5) = 0$$

$$\cos(\alpha_1) + \cos(\alpha_2) + \cos(\alpha_3) + \cos(\alpha_4) + \cos(\alpha_5) = 5(0.8)$$

Para resolverlo, se usa Matlab, y se ejecutó el código que se muestran en los anexos (Figura B.1), y se obtuvieron valores en radianes, los cuales están reportados como φ en la Tabla 8.1. Para el siguiente renglón en la tabla, se convirtieron estos a grados multiplicándolo por 180/π. Como este fue mayor a 360°, se buscó tener un valor reducido, con la siguiente ecuación: $\alpha = |360°(l) - \varphi°|$, donde i es el número entero que cumple con $\alpha < 90°$. Por ejemplo, para el primero, siendo 1.2941*10^4, se usa *l* como 36, de manera que |360*36-1.2941*10^4|°<90°. Otros valores como 35 o 37 ya no son válidos de acuerdo con esta lógica, ya que son valores mayores a 90°.

La razón por la cual se elige 90°, es que el principal ángulo de conmutación pertenece al primer cuadrante (0-90°) [12].

Tabla 8.1. Resultados para el ángulo de control.

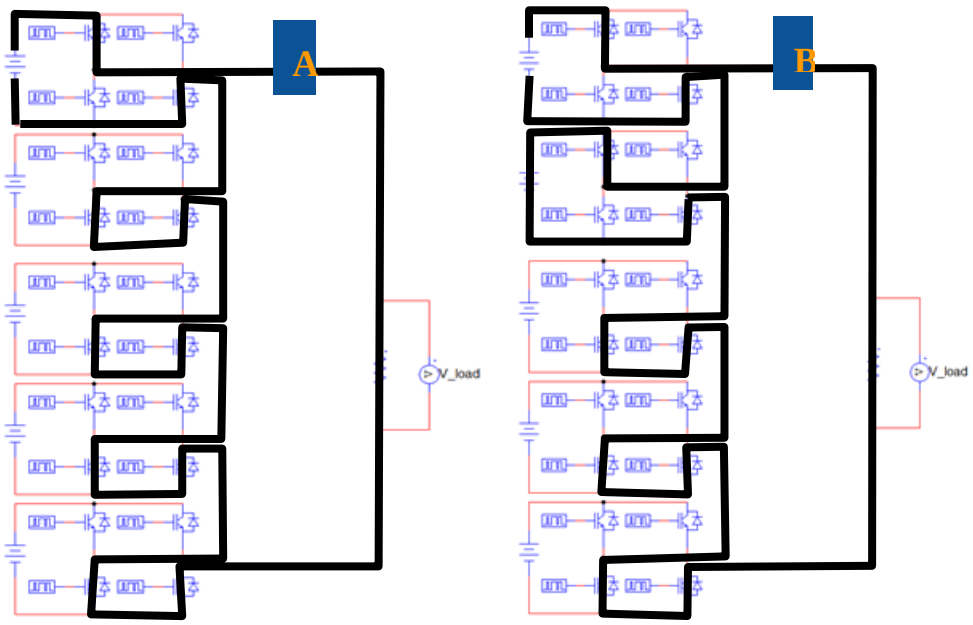
	α_1	α_2	α_3	$\alpha 4$	α_5
φ (rad)	-225.8641	99.4446	62.0441	25.0181	-163.8372
φ (10^3 °)	12.941	5.6978	3.5549	1.4334	-9.3872
α (°)	18.94	62.24	45.14	6.57	27.18

El método que se usó no es el mejor, ya que Matlab arroja un mensaje de "*Warning: Unable to solve symbolically. Returning a numeric solution using vpasolve.*" Esto significa que la función de *solve* no logró resolver las ecuaciones de manera simbólico, por lo que tuvo que usar la herramienta de *vpasolve* para dar un valor numérico. Entonces, además de usar

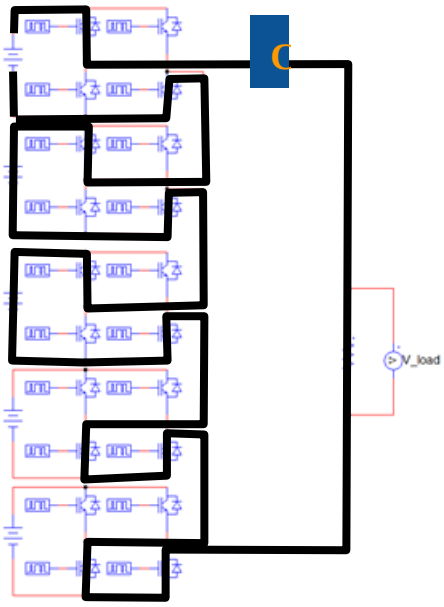
una función extra, a las soluciones que dio, se les hicieron un tratamiento extra. Por lo cual, a través de pasos empíricos, se puede llegar al valor correcto. Entonces, como funcionó este procedimiento para encontrar el valor, no se vio necesario hacer más modificaciones.

8.3. Determinación de interrupción

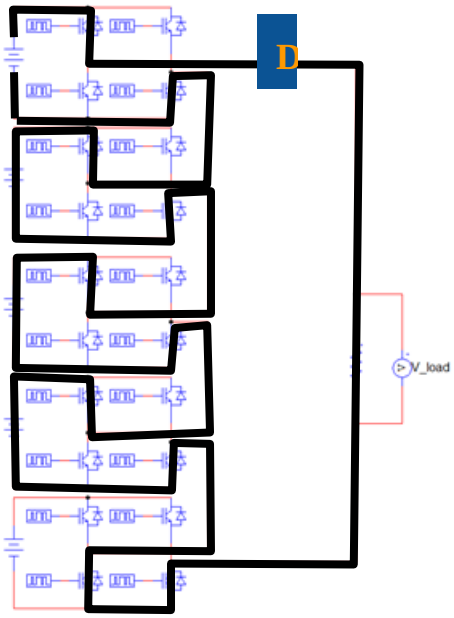
Con esto, se puede decidir que interruptores abrir o cerrar para lograr diferentes voltajes. Se determinó, que entre las dos posibilidades para conseguir 0V, se escogió la configuración d) de la Figura 7.2. Considerando esto se tienen las siguientes imágenes (Figura 8.2). Resumiendo esto, la configuración "A" da $+V_1$ en la salida, "B" da V_1+V_2, "C" da $V_1+V_2+V_3$, "D" da $V_1+V_2+V_3+V_4$, "E" da $V_1+V_2+V_3+V_4+V_5$, 0 da 0V, "V" da $-V_1$, "W" da $-V_1-V_2$, "X" da $-V_1-V_2-V_3$, "Y" da $-V_1-V_2-V_3-V_4$ y finalmente "Z" da $-V_1-V_2-V_3-V_4-V_5$.



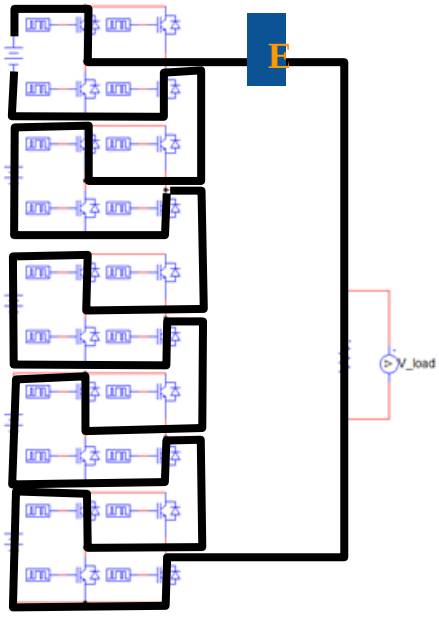
I) II)



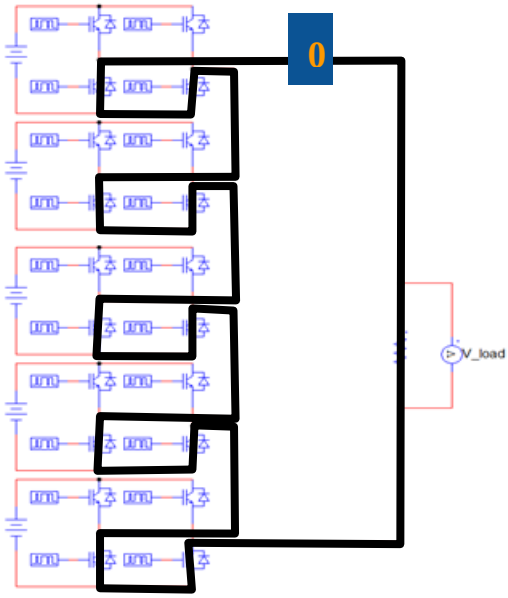
III)



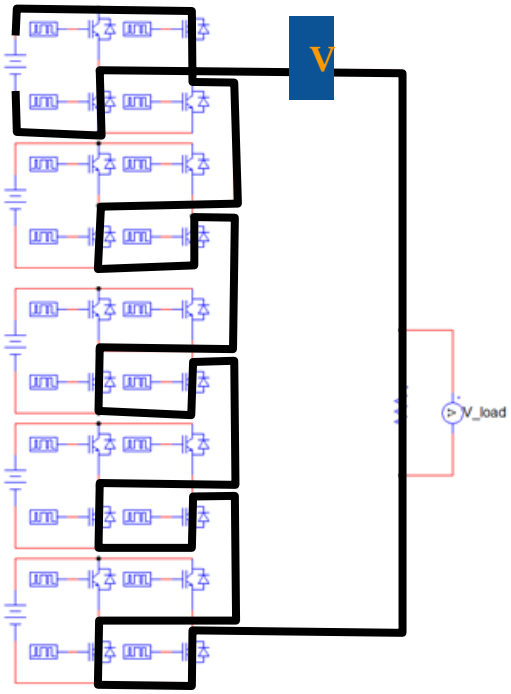
IV)



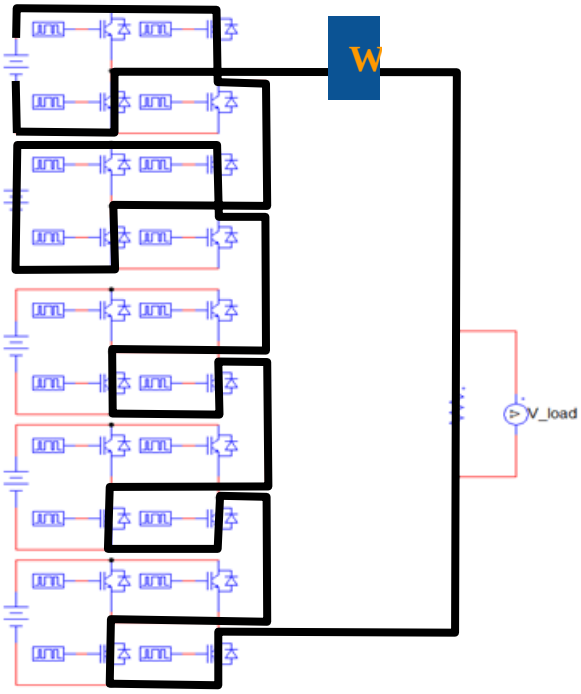
V)



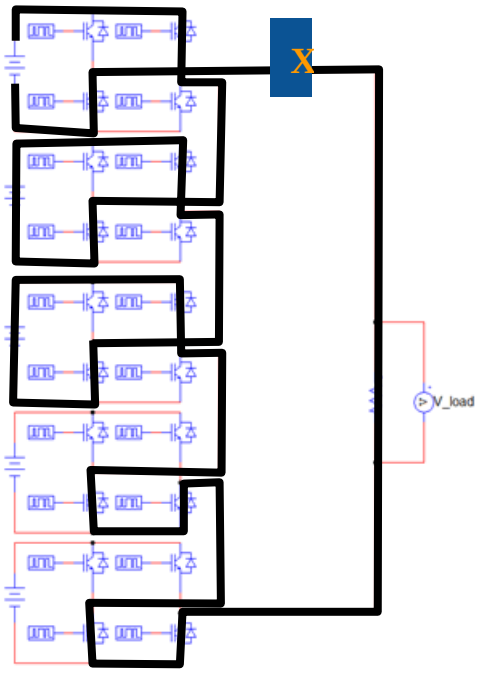
VI)



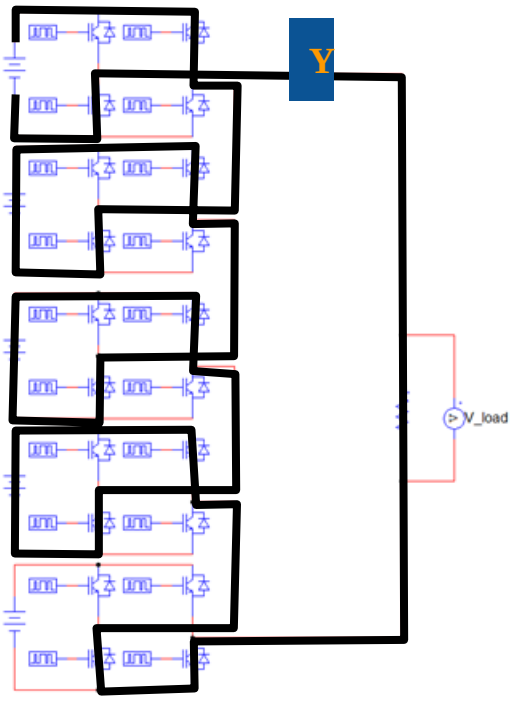
VII)



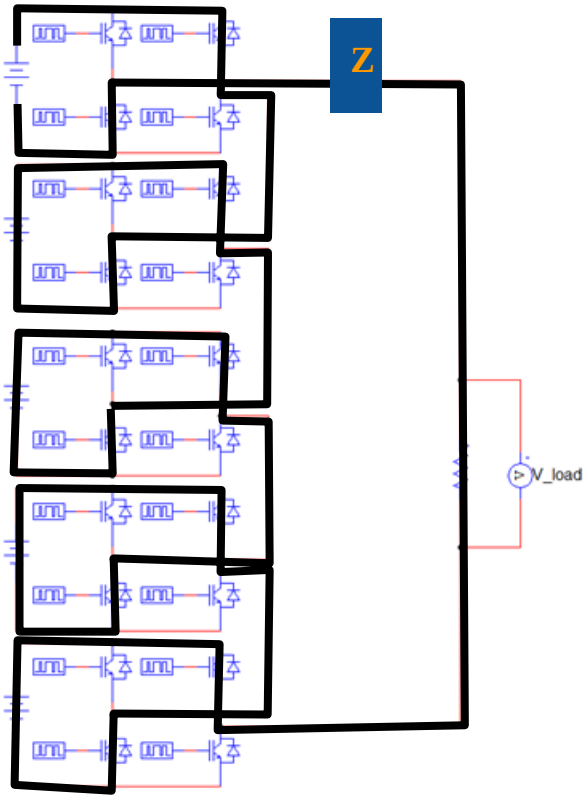
VIII)



IX)



X)



XI)

Figura 8.2. Configuraciones para conseguir los diferentes niveles: I) V_1, II) V_2, III) V_3, IV) V_4, V) V_5, VI) 0V, VII) $-V_1$, VIII) $-V_2$, IX) $-V_3$, X) $-V_4$, XI) $-V_5$.

En la Tabla 8.2 se conecta cada configuración de la Figura 8.2 con el debido ángulo de conmutación, contando los cuatro cuadrantes. El primer cuadrante es el mencionado anteriormente del ángulo principal (α), el segundo cuadrante es 180°-α, el tercer cuadrante es 180°+α y el cuatro cuadrante es 360°-α [12]. Para el primer y segundo cuadrante corresponden las configuraciones I)-V), y para el tercero y cuarto corresponden las configuraciones de VI) a XI). Lo cual da:

Tabla 8.2. Conexión entre ángulo de conmutación y configuraciones planteadas.

	Circuito	Ángulo (°)
t0	0	0
t1	$A_{inicial}$	6.57
t2	$B_{inicial}$	18.94
t3	$C_{inicial}$	27.18
t4	$D_{inicial}$	45.14
t5	$E_{inicial}$	62.24
t6	E_{final}	180-62.24
t7	D_{final}	180-45.14
t8	C_{final}	180-27.18
t9	C_{final}	180-18.94
t10	A_{final}	180-6.57
t11	0	180
t12	$V_{inicial}$	180+6.57
t13	$W_{inicial}$	180+18.94
t14	$X_{inicial}$	180+27.18
t15	$Y_{inicial}$	180+45.14
t16	$Z_{inicial}$	180+62.24
t17	Z_{final}	360-62.24
t18	Y_{final}	360-45.14
t19	X_{final}	360-27.18
t20	W_{final}	360-18.94
t21	V_{final}	360-6.57
tf	0	360

Con esto, se hace la Tabla 8.3, que describe en que intervalo angular aporta que voltaje. El propósito de estas tablas es para entender cómo van a operar los interruptores.

Tabla 8.3. Voltaje en la salida definido por el intervalo angular.

Voltaje	Intervalo angular (°)
V_1	6.57-173.43
$-V_1$	186.57-353.43
V_2	18.94-161.06
$-V_2$	198.94-341.06
V_3	27.18-152.82
$-V_3$	207.18-332.82
V_4	45.14-134.86
$-V_4$	225.14-314.86
V_5	62.24-117.76
$-V_5$	242.24-297.76

Entonces, esto permite entender y determinar la activación y desactivación de cada interruptor; lo cual se resume en la Tabla 8.4. El primer ángulo, se refiere a la primera activación del interruptor, el segundo ángulo es la primera desactivación, el tercer ángulo es la segunda activación y el cuarto ángulo es la segunda desactivación. Como algunos no van a tener ni la segunda activación ni desactivación, se deja como "-" para indicar esto. Es importante que esta configuración no es la única que se puede hacer. Por ejemplo, se puede cambiar que interruptores son responsables de 0V, teniendo diferentes opciones entre puentes, o incluso se puede trabajar con la configuración que permite que los tiempos en los

que cada la fuente de voltaje se utilice, sean equitativos y se desgasten equitativamente. Por diseño y simplicidad, se eligió la configuración actual [12].

Tabla 8.4. Ángulos de conmutación para los interruptores.

	θ_1 (°)	θ_2 (°)	θ_3 (°)	θ_4 (°)
S1.1	6.57	173.43	-	-
S1.2	186.57	353.43	-	-
S1.3	0	6.57	173.43	360
S1.4	0	186.57	353.43	360
S2.1	18.94	161.06	-	-
S2.2	198.94	341.06	-	-
S2.3	0	18.94	161.06	360
S2.4	0	198.94	341.06	360
S3.1	27.18	152.82	-	-
S3.2	207.18	332.82	-	-
S3.3	0	27.18	152.82	360
S3.4	0	207.18	332.82	360
S4.1	45.14	134.86	-	-
S4.2	225.14	314.86	-	-
S4.3	0	45.14	134.86	360
S4.4	0	225.14	314.86	360
S5.1	62.24	117.76	-	-
S5.2	242.24	297.76	-	-
S5.3	0	62.24	117.76	360
S5.4	0	242.24	297.76	360

8.4. Simulación en PSIM

Con estos valores se puede hacer la simulación en software de PSIM. Esto se hace realzando el circuito de la Figura 8.1; para esto se utilizan: cinco fuentes de voltaje CD, veinte modelos IGBT, con sus veinte *gating blocks*, una carga resistiva, y un voltímetro. Además, se colocan los valores de la Tabla 8.4 con una frecuencia de 60Hz en los *gating blocks for switches*, y se coloca 100V para cada fuente de voltaje. Corriendo la simulación, se obtiene la Figura 8.3. Esto indica que se hizo correctamente el análisis y describe el comportamiento deseado. Para ser más específicos, se alcanzan a ver los 5 niveles, primero al haber cinco incrementos de 100V a 100V iniciando en 0V y así mismo, disminuyendo cinco veces de 100V a 100V iniciando en 0V. Se midió la potencia y el THD, dando respectivamente 131W y 0.897%, lo que significa que tiene muy buena calidad de la señal.

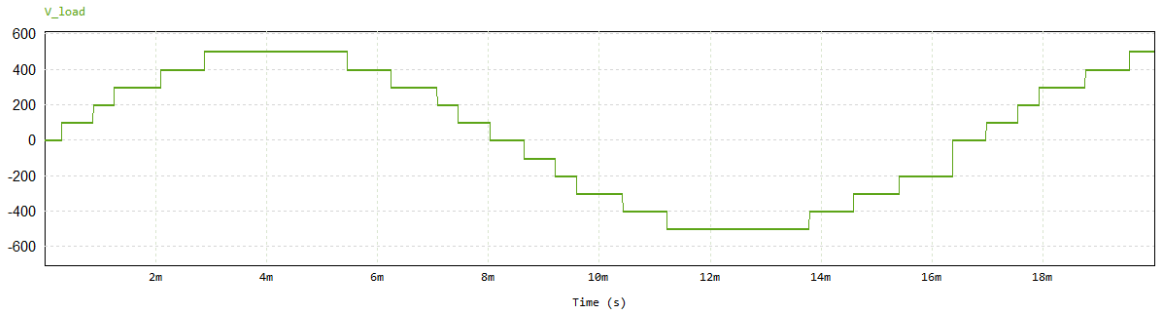


Figura 8.3. Resultado del voltaje de la carga para el inversor.

Para estudiar su comportamiento, se puede ver la característica de la modulación, con una primera impresión, la salida muestra que el tiempo no es equitativo entre un nivel y otro, causa de los diferentes ángulos de conmutación. Aunque para ver si se está haciendo esto correctamente, se necesita analizar puente por puente. En la siguiente figura se analiza el comportamiento del primer puente, y se ven los intervalos del ángulo. Los cuales son corresponden a lo estudiado.

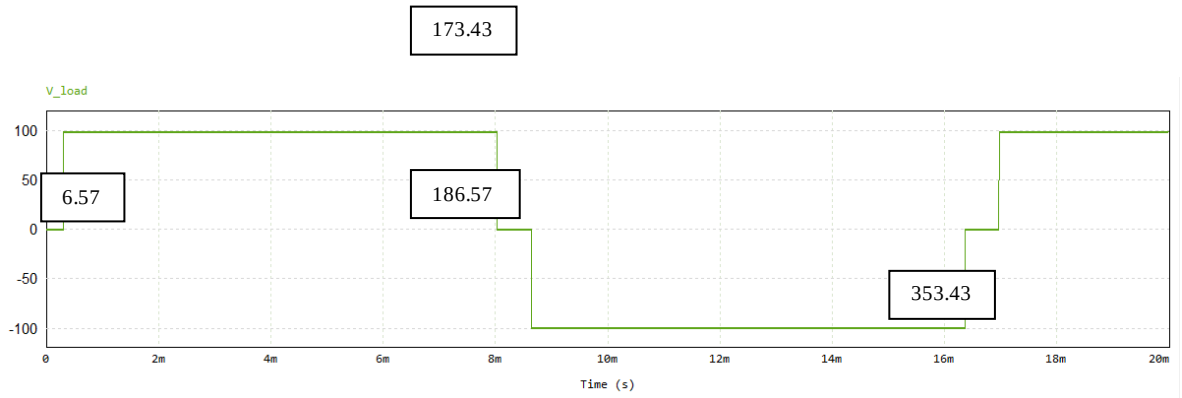


Figura 8.4. Salida de voltaje del primer puente con sus ángulos de cambio de estado marcados.

Estudiando cada nivel, se observa que los ángulos son correctos, por lo que el análisis estuvo bien hecho. Cumpliendo con esto, se puede empezar trabajando con un FPGA o un microcontrolador para poder hacer el diseño de cómo implementar las señales del gate. Para esto, es importante trabajar con frecuencias o tiempo. En el siguiente capítulo, se va a estudiar esto con un FPGA primero.

Capítulo 9. FPGA

9.1. Descripción

FPGA viene de las siglas en inglés *Field-programmable gate arrays*; se trata de un tipo especial de circuitos integrados, también puede ser pensado como chips que se programan en campo después de ser fabricado. Cuentan con tres bloques de construcción básicos: compuertas lógicas, flip-flops más memorias y cables. En la hoja de especificación, se encuentra normalmente el número de compuertas lógicas, la cantidad de memoria y cómo programarlo [13].

Una de las características del FPGA, es que permite diseñar un circuito digital hardware, lo que significa que no es un procesador con software. Además, se puede cambiar o realizar el diseño rápidamente. Lo que lo hace útil para hacer prototipos o desarrollar un producto rápidamente. Para la programación, se usa un lenguaje de descripción de hardware. La parte de *field-programmable*, se realiza con un archivo de configuración, también llamado archivo bit. Al cargarlo, el FPGA se comporta como el circuito digital diseñado. Lo que hace a partir de esto es generar unos lógicos (3.3V) y ceros lógicos (0V) [13].

Cuando se compra el componente, no tiene una función específica, lo cual da la flexibilidad de diseño en el dominio digital. Tiene la capacidad de definir el conjunto de características, hacer el trabajo más rápido que una computadora al separarlo en piezas discretas. Esto lo realiza por su habilidad de reducir las ordenes de las herramientas y su funcionamiento en paralelo. Esto significa que pueden reducir las operaciones complejas, ejecutando más rápido y consumiendo menos energía, y también puede paralelizar una tarea [13].

Entonces, el FPGA es diferente a un procesador, ya que este tiene una configuración de hardware permanente; y aquel se configura. Lo que es permanente en el FPGA son la configuración de los procesadores, es decir, los registros, transistores, estructuras de interfaz y las conexiones. Esto causa que las tareas que pueden realizar las FPGA no están predefinidas, y se configuren con el código escrito en lenguaje de descripción de hardware (HDL) en paralelo [13].

Como se mencionó, los FlipFlops es un componente principal del FPGA. Por otro lado, esto es un elemento importante para la programación de los pulsos para el interruptor. En la siguiente sección, se detalla el funcionamiento de este.

9.2. FlipFlops

Para los sistemas digitales, existen dos tipos de circuitos lógicos: combinacionales y secuenciales [14]. El primero se compone de compuertas lógicas que realizan operaciones booleanas con los valores de entradas para producir una salida, no tiene elementos de memoria ni retroalimentación. El segundo funciona como un elemento de almacenamiento y memoria; es capaz de guardar y almacenar información, para posteriormente acceder a ella. Este está compuesto por un circuito combinacional, conectado a un elemento de memoria para formar una ruta de realimentación. El elemento de memoria se encarga de guardar la información binaria y define el estado del circuito en ese momento. Además, el circuito secuencial recibe información de las entradas y produce una salida que depende de las entradas y del estado actual del circuito. Entonces, el siguiente estado que se guarda en los elementos de memoria, contribuirá a las entradas [14].

Existen dos tipos de circuitos secuenciales: síncronos y asíncronos. En los primeros, un intervalo discreto de tiempo producido por señales define el comportamiento. En los segundos, el comportamiento depende de las señales de las entradas en cualquier momento dado y en el orden de los cambios en las entradas. Para el primero, la sincronización de las señales es responsable del intervalo de tiempo, y se realiza mediante un dispositivo de temporización que produce una señal de reloj. La señal de reloj es un "tren periódico de pulsos de reloj". Esto significa que a cada impulso el circuito secuencial se ve afectado. En otras palabras, la señal de reloj determina cuándo cambiará el comportamiento del circuito (la salida es el complemento del último estado) o permanecerá igual (la salida es la misma que el último estado) [14].

Por otro lado, los elementos de memoria en los circuitos secuenciales están compuestos por flip-flops. Cada elemento de memoria tiene un flip-flop, que es capaz de almacenar 1 bit de información. Entonces, si uno quiere almacenar n bits de información, necesita n flip-flops. Además, la información que se guardará en el flip-flop viene determinada por las entradas y la última información guardada en el flip-flop. En el caso del circuito síncrono, el flip-flop guarda nueva información cada vez que se produce una transición de reloj, por ejemplo, cuando el reloj cambia de 0 a 1. Existen diferentes tipos de flip-flops, que son: RS, D, JK y T. [14].

El circuito RS tiene dos entradas, S (set) y R (reset), más dos salidas: Q y su complemento (Q'). Además, se compone de dos puertas lógicas del mismo tipo conectadas en cruz, usando ya sea las compuertas lógicas NOR o las NAND. Lo que hace es, que puede establecer o restablecer los valores, o permanecen iguales. Por ejemplo, cuando Q=1 y Q'=0 se llama estado de set (S=1), y cuando Q=0 y Q'=1 (R=1) se llama estado de reset. Además,

cuando ambas entradas son 1, entonces ambas salidas son 0. Como esto no es posible, porque establece que el complemento de 0 es 0, entonces es una condición prohibida. Por último, cuando ambas entradas son 0, puede estar en estado de set o de reset; lo que significa que adquiere el estado anterior (cuando una entrada era igual a 1). Esto significa que el flip-flop guardará un nuevo valor si R o S es igual a 1 [14].

En el flip-flop JK hay dos entradas (J y K) que controlan los estados y funciona de la misma forma que el flip-flop RS, con la diferencia de que se permite J=K=1. Cuando se produce esta combinación, el estado se invierte. Esto significa que si estaba en estado de set se invertirá al estado de reset y viceversa. Además, si se añade una señal de reloj, se producirá un cambio cuando haya un pulso, igual que en el flip-flop RS [15]. La tabla de verdad se muestra a continuación.

Tabla 9.1. Tabla de verdad para un FlipFlop JK asíncrono.

J	K	Q_{n+1}	
0	0	Q_n	Permanece igual
0	1	0	Reset
1	0	1	Set
1	1	Q'_n	Complemento

Al igual que el flip-flop JK, el flipflop D aborda el problema que tiene al evitar la combinación R=S=1 [14]. Sin embargo, a diferencia del flip-flop JK, el flip-flop D sólo tiene una entrada: D que representa los datos. La salida Q_{n+1} adquiere el mismo valor que D, lo que significa que el estado de reset se produce cuando D=0 y el estado de set se produce cuando D=1. Por lo tanto, D almacena el estado actual en el flip-flop. [15]

El flip-flop T tiene una sola entrada llamada toggle (T), y actúa como un flip-flop complementario. Se puede construirse utilizando un flip-flop JK. Esto significa que cuando T=0, actúa igual que cuando J=K=0: la salida no cambia. Adicionalmente, cuando T=1, se comporta igual que cuando J=K=1, lo que significa que el estado será invertido [14].

Tabla 9.2. Tabla de verdad para el flipflop T.

T	Q_n+1
0	Qn (mismo estado)
1	Qn' (complemento)

En la sección 9.4. , se va a detallar la implementación del Flip-Flop en el código, aunque primero se mencionara a continuación el FPGA usado.

9.3. Basys 3

El Basys 3 pertenece a la familia Artix-7TM, su número de parte Xilinx es XC7A35T-1CPG236G, e incluye 1.8Mbits, 16 interruptores manuales, 16 LEDs, 5 botones, display de siete segmentos, tres compuertas Pmod, puerto de USB, 41600 flip-flops, 31200 entradas LUTs, entre otras herramientas. Puede funcionar desde para circuitos combinacionales básicas hasta circuitos secuenciales complejos. Este FPGA puede ser programado con Vivado. Sin contar el RAM y la parte auxiliar, el equipo requiere de un máximo de 2A, el I/O y PMOD suministra 3.3V [16].

Los componentes, como puertos, se ven en la **Error! Reference source not found.** Con el componente 15, se apaga o prende el FPGA y el componente 1 indica si se prendió

correctamente. Hay dos opciones de conexión a una fuente: con USB o con una fuente externa, esto se elige con el componente 16. Por otro lado, para configurar el FPGA, hay tres opciones: USB-JTAG, QUAD SPI, USB-Host, el componente 10 permite esta selección. En este caso, se usa el primero, ya que es el modo para transferir el archivo bit del PC al FPGA [16].

En la Figura 9.1 se ven los pines. Buscándolo bien, se puede encontrar el pin W5 en la vista frontal, que es donde se ubica un oscilador de 100MHz, el cual puede ser usado para generar el CLK. Con este oscilador, se hará una división de frecuencias como se verá más adelante.

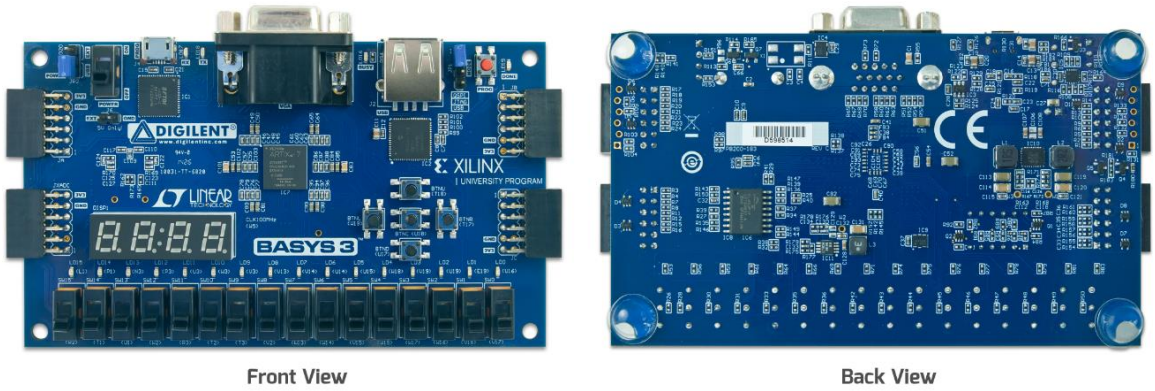


Figura 9.1. Vista frontal y trasera del Basys3 [16].

Para generar señales digitales, se usa del Pmod Connectors, la palabra viene de *Peripheral Module*. Este componente se puede usar para expandir puertos. En cada PMOD, hay dos fuentes VCC, dos tierras y ocho señales [16]. Teóricamente se puede usar para generar señales cuadradas. En la siguiente sección, se plantea la implementación para el uso del inversor.

9.4. Implementación

En el Basys 3 se trabaja con el reloj interno (el oscilador), y se necesita para poder trabajar con tiempos y CLK. En comparación, como se va ver en el capítulo de Arduino, en el código no se puede trabajar con los "delay", sino que se trabaja con los flancos del oscilador. Además, como se está trabajando con el concepto de Flip-Flops, se trabaja con cambios de estado.

Pensando en la Tabla 9.2, se podría trabajar con el Flip-Flop T, ya que lo que se busca es que, al llegar a un tiempo, hipotéticamente T es uno en ese instante, y permite cambiar el estado de la salida. Esto es importante, ya que se puede empezar la señal de salida con un 0, luego cambiando su estado a 1 en el instante θ_1, y después volver a cambiar el estado a 0 en el instante θ_2. Lo que permite una señal cuadrada que corresponde a la señal para controlar un interruptor.

Los instantes mencionados, tienen que realizarse a partir de la división de reloj del oscilador. Lo que se puede realizar, es contar cada flanco de subida del reloj, creando instantes en el tiempo. Por lo que se hará en términos básicos, un sistema de conteos. Teniendo 100MHz de frecuencia para el oscilador, significa que su periodo es de 10ns y el primer flanco de subido ocurre cada 5ns. Por lo que el tiempo en el que ocurre un flanco de subida $t = 5ns * c$, donde c es el conteo de flancos de subida. Es importante mencionar que esto ocurre si el ciclo de trabajo es de 50%, lo cual es necesario definir en el archivo XDC (donde se definen los puertos a usar).

Entonces para poder controlar los interruptores cómo se desea, se tiene que usar la Tabla 8.4 y pasar de ángulo a conteos. Esto se puede realizar con la siguiente ecuación:

$$c = \frac{\theta * f_{Clk}}{360° * f} \tag{10}$$

Donde f_{Clk} es la frecuencia del oscilador del FPGA, el cual es en este caso 100MHz. Mientras, f es la frecuencia de la salida, el cual tendría que ser de 60Hz. Esto es para una señal de reloj que tenga un 50% de Duty Cycle. Es posible trabajar con otro porcentaje, pero en este caso es ideal dejarlo así. Por otro lado, los valores se tienen que redondear a un valor entero, ya que no se puede dividir la señal de reloj en un número decimal.

Teniendo en cuenta que el estado inicial para todos los interruptores es abierto (tienen estado 0), y la frecuencia es de 60Hz, se genera la siguiente tabla con la ecuación mencionada, para cambiar de estado en cada conteo. Es importante agregar que a los 360° o en el conteo 1666667, se reinicie el conteo.

Tabla 9.3. Conteo para cambio de estado para 60Hz.

	c1	c2	c3	c4
S1.1	30417	802917	-	-
S1.2	863750	1636250	-	-
S1.3	0	30417	802917	1666667
S1.4	0	863750	1636250	1666667
S2.1	87685	745648	-	-
S2.2	921018	1578981	1666667	-
S2.3	0	87685	745648	1666667
S2.4	0	921018	1636250	1666667
S3.1	125833	707500	1666667	-
S3.2	959167	1540833	1666667	-

S3.3	0	125833	707500	1666667
S3.4	0	959167	1540833	1666667
S4.1	208981	624352	1666667	-
S4.2	1042315	1457685	1666667	-
S4.3	0	208981	624352	1666667
S4.4	0	1042315	1457685	1666667
S5.1	288148	545185	1666667	-
S5.2	1121481	1378519	1666667	-
S5.3	0	288148	545185	1666667
S5.4	0	1121481	1378519	1666667

Para los conocimientos de programación, se usó [17], permite entender los conceptos y cómo programar con VHDL. De acuerdo al mismo libro, HDL se refiere el modelo del comportamiento en múltiples niveles de abstracción. En la siguiente sección, se reporta la simulación hecha en VHDL.

9.5. Testbench y código

Se hicieron dos códigos sencillos para ver si se está creando la onda correctamente para los primeros dos interruptores. El código de estas dos pruebas se encuentra en los anexos. La Figura C.1, muestra el código para realizar la señal del interruptor S1.1 y la Figura C.2, muestra el código para la señal de control del interruptor S1.2. Los dos son similares, la diferencia es que, en el segundo, se inicia en 1 y termina en 1; el otro inicia y termina en 0.

Fuera de eso, tienen la misma estructura: la entidad con un entrada (reloj) y una salida (señal para el interruptor), luego en la arquitectura se crean dos señales: tmp (para realizar el cambio de estado) y conteo (para ir contando el flanco de subido), y después se crean los procesos if. En estos, primero se realiza el conteo para cada flanco de subida, y después se realiza el cambio de estados a la señal tmp en base a los conteos. Al final se asigna el valor actualizado de tmp a la señal de salida.

Las siguiente dos figuras, se muestra la simulación del código usando los *testbench.* En esta parte, se asigna un estímulo a clk para que tenga la señal de 100MHz. También, se dejó correr para que hiciera dos ciclos de 60Hz, es decir hasta 8.33ms. Como se observa, el código se realizó correctamente. Por ejemplo, el primer cambio se realiza en 304165ns o en 304.165us, el segundo cambió ocurre a los 8029.165ns y el fin del periodo ocurre a los 16,666.67us. Esto se puede comparar con la Tabla 10.4, donde se empieza a trabajar con tiempos. Los cambios para el segundo interruptor ocurren en los mismos momentos, solo que es el complemento del primero. Algo interesante a notar, es que se redondean los conteos a números enteros, y hay un efecto en los tiempos. Es decir, tomando varios dígitos se ve que los últimos no coinciden. Por ejemplo, el periodo para 60Hz es 16,666.66667us pero el primer testbench lo marca por 16,666.667328us. Por lo cual se ve que tomando en cuenta varios decimales, se ve una ligera diferencia. No obstante, ya que la diferencia es considerablemente pequeña, se toma como correcto la simulación.

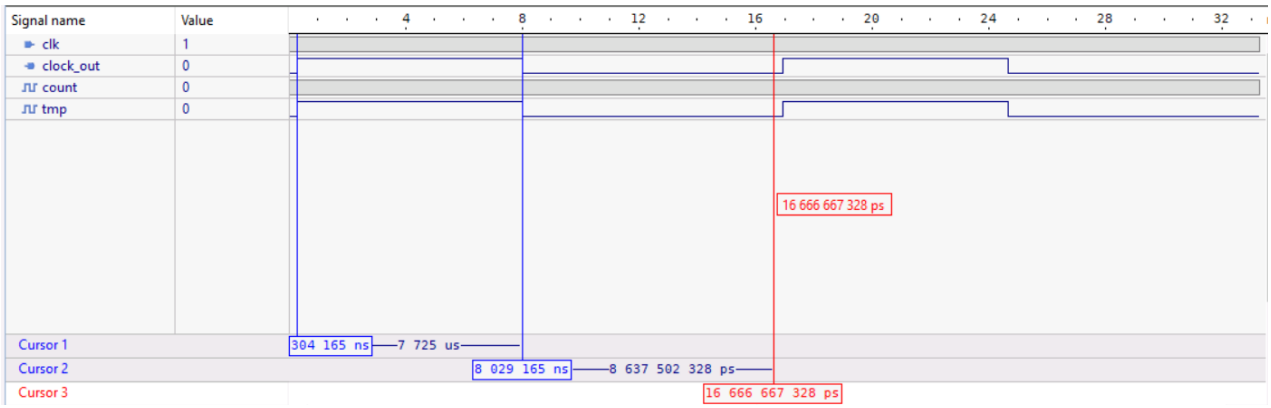


Figura 9.2 Testbench para el interruptor S1.1.

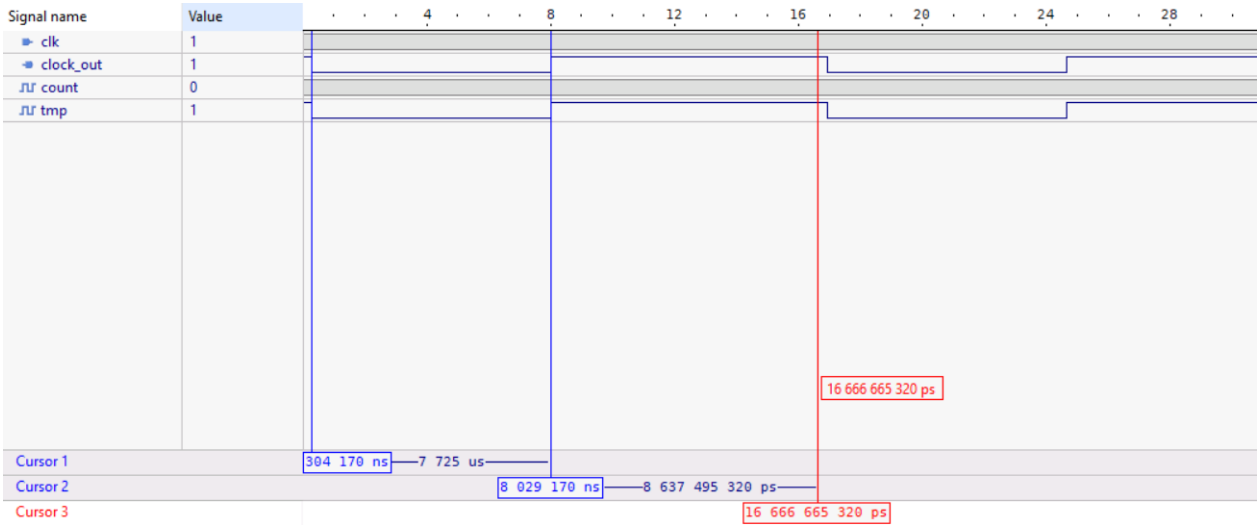


Figura 9.3. Testbench para el interruptor S1.2.

Viendo que estos dos códigos usan la misma estructura, significa que permite usar esto como plantilla para los veinte interruptores, y no ir haciendo uno código por cada uno. Por lo cual es posible, usar port maps con parámetros de conteo, para ir definiendo las veinte salidas.

9.6. Problemática

El FPGA si se ha usado para generar señales PWM, pero no como tal para generar señales como estas. El manual del Basys3, marca que los Pmod se usan más para extensión de puertos. Adicionalmente, no se encontró documentación o ejemplos que seguir en este FPGA para generar señales digitales. Lo más cercano que se vio, es que en el Pmod se añadió una extensión de motor, y se usaron justamente las señales PWM para trabajar con el motor. Esto no significa que no se pueda usar el FPGA para el inversor.

Además, esta tecnología tiene más complicaciones y consideraciones para aplicar en este proyecto. Por un lado, porque el Basys3 es para fines educacionales, y se tendría que hacer una revisión detallada de la documentación. Por otro lado, hay que considerar los tiempos muertos y retrasos para que no exista cortos circuitos. Por estas razones, se decidió que la implementación sería más sencilla si se usa un microcontrolador como el Arduino Mega 2560.

Capítulo 10. Arduino

10.1. Microcontrolador

Un microcontrolador se refiere a una computadora con la mayoría de chip de soportes. Las computadoras tienen los siguientes componentes: un CPU (ejecuta el programa), RAM (almacena información variable), ROM (programas a ejecutar son almacenados), y dispositivos I/O (permiten la comunicación entre el mundo externo). Por otro lado, lo que clasifica que sea un microcontrolador es que está dedicado a una tarea, ya que corre un programa en específico, que es un dispositivo de baja potencia, y puede estar embebido dentro de otro dispositivo para poder controlar las características de un producto. El microcontrolador entonces puede funcionar como un controlador, tomando una entrada del producto y mandado señales a partir de esto al producto. El microcontrolador es en la mayoría, compacto y de bajo costo [18].

Los componentes de un microcontrolador incluyen lo que se había mencionado de una computadora (CPU, RO, RAM), más otras partes como: oscilador de reloj (determina la velocidad en la qué corre el programa), circuito de reinicio y detector de caídas de voltaje (los componentes comienzan en un estado predefinido para ser inicializados propiamente y se monitorean las caídas de voltaje para resetear si sea necesario), puerto serial (permite comunicación entre dispositivos externos en una base de información serial), puertos digitales I/O (permite intercambiar información digital con el mundo externo como bytes) , puertos analógicos I/O (con un ADC puede adquirir información de un sensor), temporizador (controla eventos en el tiempo), temporizador de vigilancia (para prevenir *software crashes*), reloj en tiempo real (mantiene las fechas) [18].

Hay diferentes tipos de microcontroladores y compañías que los producen o los hacen felices. Entre ellos está la compañía de Arduino, que ofrece diferentes placas de microcontroladores, uno de ellos es el Mega 2560, el cual se va a usar y describir a continuación.

10.2. Mega 2560

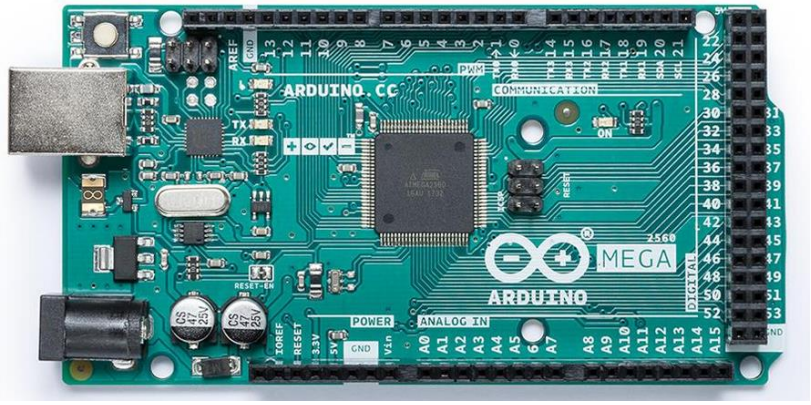


Figura 10.1. Vista frontal del Arduino Mega 2560 [19].

El Arduino Mega 2560 es una placa que acomoda el microcontrolador ATmega2560. En la Figura 10.1 se observa la vista frontal de la placa. Este opera a una frecuencia de 16MHz, y cuenta con 54 pines digitales I/O, 16 entradas analógicas, 4 UARTs (puertos serie por hardware), una conexión USB, un cabezal ICSP, un botón de reinicio y un conector de alimentación. En la Figura A.3 se muestran algunos de los componentes mencionados [19].

En la Figura A.4 y la Figura 10.2 se muestra el mapeo de los pines en la placa, entre ellos se ubican los pines analógicos, digitales, fuente de poder, tierra y los LEDs. Para el trabajo, se utilizan los pines digitales y la tierra. El voltaje de entrada del conector USB, es

de 5V, teniendo como mínimo 4.8V y como máximo 5.5V, y los voltajes digitales es igual. [19]. El uno lógico es el 5V y el cero lógico es 0V.

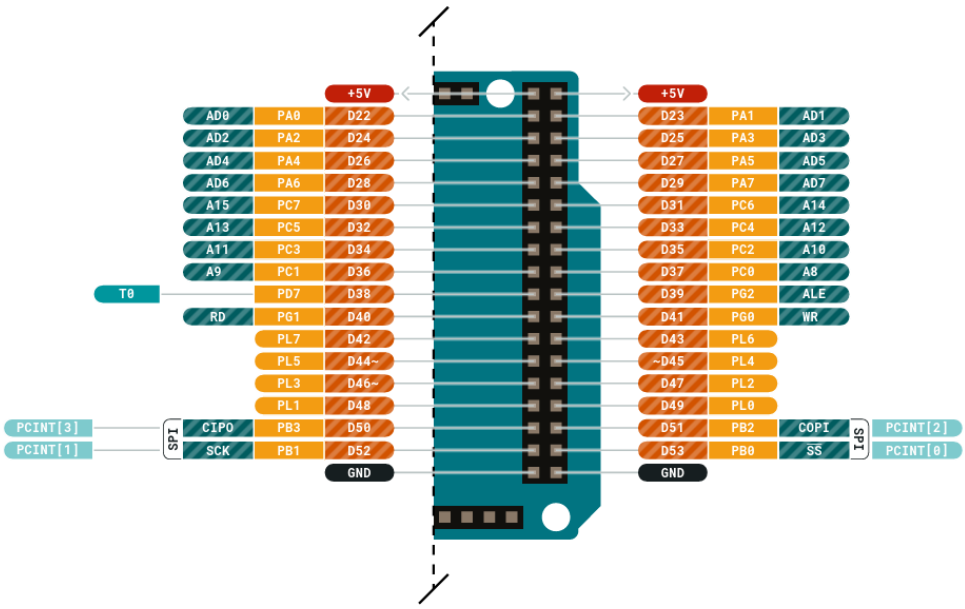


Figura 10.2 Pines digitales D22 A D53.

10.3. Implementación

Para la implementación, no se va a generar una señal digital para cada interruptor. Esta vez, se hará uso de optoacopladores, como se ve en la Figura 10.3, lo que permite generar una señal por pares de interruptor. De esta manera, se puede evitar que existan cortocircuitos o una implementación más compleja. También, va a significar que se van a generar la mitad de los pulsos requeridos (en vez de 40 son 20), lo cual simplifica la implementación. Cada pulso va a estar conectado a los extremos (en C o D). Como describe la figura al tener 5V (o un 1 lógico) en C, en la salida A hay +Vcc, y al tener 0V (o un 0 lógico) en C, hay 0V en la salida.

En el otro lado, si hay 5V en D, en la salida hay -VCC, y si hay 0V en D hay 0V en la salida también. En otras palabras, al tener un 0 lógico en cualquier sección, es como si se cerrará el interruptor inferior y el superior se mantiene abierto en el lado correspondiente, y si hay un 1 lógico, se cierra el interruptor superior y se abre el interruptor inferior.

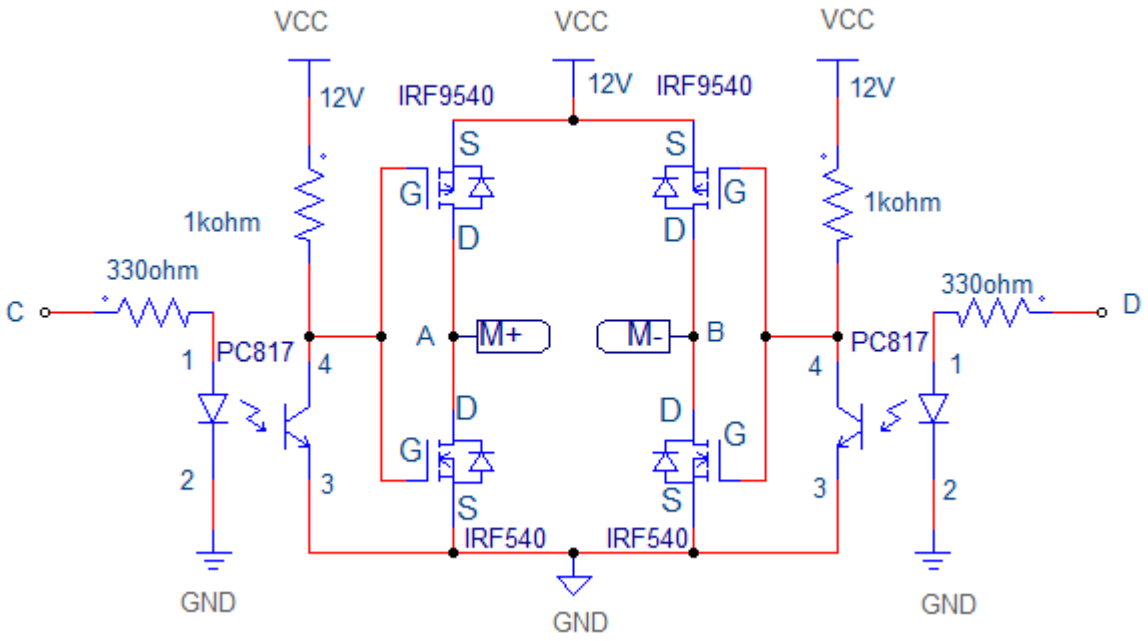


Figura 10.3. Puente H usando Mosfets y optoacopladores basado en [20].

El diseño del control de los interruptores va a tener el mismo planteamiento. En otras palabras, los mismos interruptores se van a abrir o cerrar con la misma lógica propuesta. Sin embargo, en vez de usar la Tabla 8.4 se cambia a usar la siguiente tabla, donde C se refiere al lado izquierdo del puente, D al lado derecho y el número de 1 a 5 es el puente. Se puede notar, que se están tomando los valores de los dos interruptores para cada puente (SX.1 y SX.2).

Tabla 10.1. Señal cuadrada producida para cada sección.

	θ (°)	
	Flanco de subida	Flanco de bajada
C1	6.57	173.43
D1	186.57	353.43
C2	18.94	161.06
D2	198.94	341.06
C3	27.18	152.82
D3	207.18	332.82
C4	45.14	134.86
D4	225.14	314.86
C5	62.24	117.76
D5	242.24	297.76

Los valore de θ se pasan a tiempo en μs, para poder usarlos con el Arduino utilizando los delaysMicroseconds. Para la conversión se utiliza la siguiente ecuación $\theta * \frac{10^6}{360° * f}$, donde f es 60Hz. Con esto se obtiene la siguiente tabla.

Tabla 10.2. Señal cuadrada para cada sección en términos de tiempo.

	Time (μs)	
	Flanco de subida	Flanco de bajada
C1	304.1667	8029.167

D1	8637.5	16362.5
C2	876.8519	7456.481
D2	9210.185	15789.81
C3	1258.333	7075
D3	9591.667	15408.33
C4	2089.815	6243.519
D4	10423.15	14576.85
C5	2881.481	5451.852
D5	11214.81	13785.19

Como Arduino utiliza programación secuencial, cada vez que se usan los delays, se van agregando los segundos, por lo que se tiene que trabajar con diferencias de ángulo o de tiempo. En la siguiente tabla primero se organiza en orden angular, los momentos en donde ocurre algún cambio en los interruptores, y luego se va haciendo la diferencia. Para t1 se hace 6.57-0, y para los otros t, se hace $\Delta\theta = t(\theta) - t(\theta - 1)$.

Tabla 10.3. Diferencia en ángulo para cada cambio.

Instante	θ	$\Delta\theta$
t1	6.57	6.57
t2	18.94	12.37
t3	27.18	8.24
t4	45.14	17.96
t5	62.24	17.1

t6	117.76	55.52
t7	134.86	17.1
t8	152.82	17.96
t9	161.06	8.24
t10	173.43	12.37
t11	186.57	13.14
t12	198.94	12.37
t13	207.18	8.24
t14	225.14	17.96
t15	242.24	17.1
t16	297.76	55.52
t17	314.86	17.1
t18	332.82	17.96
t19	341.06	8.24
t20	353.43	12.37
t21	360	6.57

Lo mismo se hace con el tiempo, lo que se muestra a continuación.

Tabla 10.4. Diferencia de tiempo para cada cambio.

Instante	t (us)	Δt (us)
t1	304.1667	304.1667
t2	876.8519	572.6852

t3	1258.333	381.4815
t4	2089.815	831.4815
t5	2881.481	791.6667
t6	5451.852	2570.37
t7	6243.519	791.6667
t8	7075	831.4815
t9	7456.481	381.4815
t10	8029.167	572.6852
t11	8637.5	608.3333
t12	9210.185	572.6852
t13	9591.667	381.4815
t14	10423.15	831.4815
t15	11214.81	791.6667
t16	13785.19	2570.37
t17	14576.85	791.6667
t18	15408.33	831.4815
t19	15789.81	381.4815
t20	16362.5	572.6852
t21	16666.67	304.1667

Los valores de Δt son los que se van a poner en los delay(), cada vez que hay una acción en un interruptor.

Otra tabla importante para usar para el programa es la Tabla 10.5. Lo que hace es organizar en que orden va actuando un interruptor y cuál es su estado. Esto se hizo al observar el comportamiento que se fue describiendo en el Capítulo 8.

Tabla 10.5. Estado lógico producido en cada instante.

Instante	Secuencia	Estado
t1	C1	1
t2	C2	1
t3	C3	1
t4	C4	1
t5	C5	1
t6	C5	0
t7	C4	0
t8	C3	0
t9	C2	0
t10	C1	0
t11	D1	1
t12	D2	1
t13	D3	1
t14	D4	1
t15	D5	1
t16	D5	0
t17	D4	0
t18	D3	0

t19	D2	0
t20	D1	0

Con esta información, se puede empezar con el código. La idea brevemente es que un for, se vayan cambiando los estados dependiendo del instante del tiempo. En la siguiente sección se describe el código.

10.4. Código

En la sección D de los Anexos, se muestra el código que se usa en la práctica, este se podría usar no solo para un inversor de 5 puentes, sino de 1 a 16 puentes. Esto está limitado por las salidas de un Arduino Mega. Al inicio del programa, está el encabezado, donde van declaradas las variables globales. Para cambiar de número de puentes, se cambia la constante *level* y se cambian los valores de alpha a los adecuados. En la parte de setup, se coloca lo que se va a correr una sola vez. Es decir, los puertos que se van a usar (corren con la función de portValues), y declararlos como salido. Además, se utiliza Serial.begin(9600) para poder imprimir mensajes en el Serial Monitor. Después, se corre la función degree(), en donde se coloca lo que aparece en la tabla (C y D). El primer índice representa el nivel, y el segundo indica si es el primer C (0), el primer D (1), el segundo C (2) o el segundo D (3) del puente correspondiente.

En la función loop, se coloca el programa principal, que va a correr continuamente hasta que se desconecta el Arduino. Se crean dos variables local p y status, para poder controlar cómo actúa cada interruptor a lo largo

del tiempo. Después, se manda a llamar la función time. Lo que hace es primero guardar los valores de alpha de la

Tabla 10.1 como thetaSeq, aunque no incluye el último valor. Por otro lado, como estos valores están desordenados, se usa el algoritmo de quickSort, que permite ordenarlos con el concepto base de ir dividiendo los valores. Después, se guarda la secuencia del tiempo para la frecuencia definida y para timeSeqDiff, se guardan los valores de la Tabla 10.4, incluyendo el último valor. Con esto ya se puede usar los delayMicroseconds.

Siguiendo con el programa principal, se diseñó un algoritmo que permite controlar interruptores en tiempos específicos. Para esto, observando el comportamiento de los cambios de estado, se hicieron cuatro grupos. El primer grupo se encarga de mandar un 1 a los puertos pares, el segundo grupo manda un 0 a estos puertos, el tercer grupo manda un 1 a los puertos impares y el cuarto grupo manda un 0 a los puertos impares. Estos grupos se hicieron, ya que se quiere ir trabajando con la secuencia descrita en la tabla. Además, al observar un comportamiento en común entre los elementos de cada grupo, siendo el estado y siendo D o C, se decidió formar estos cuatro grupos en específico. Como esta un ciclo for, con la variable i contando los ciclos, se fue jugando con el álgebra, para poder conectar el número de puentes, con el grupo y los conteos. Entonces se usó para el primer grupo la expresión $port[2 * i]$, para el segundo grupo fue $port[4 * level - 2 * i - 2]$, para el tercer grupo fue $port[2 * i - level * 4 + 1]$, y para el cuarto grupo fue $port[-2 * i + level * 8 + 1]$ para indicar que puerto se usa en orden. La consecuencia de esto es lo que aparece en la siguiente tabla para el caso de cinco niveles. Como se mencionó anteriormente, este algoritmo se puede realizar por otros niveles, lo cual se puede comprobar rápidamente, por ejemplo, en Excel.

Tabla 10.6. Resultado de las expresiones.

Secuencia	i	Expresión	Port[expresión]	Estado
t1	0	0	22	1
t2	1	2	24	1
t3	2	4	26	1
t4	3	6	28	1
t5	4	8	30	1
t6	5	8	30	0
t7	6	6	28	0
t8	7	4	26	0
t9	8	2	24	0
t10	9	0	22	0
t11	10	1	23	1
t12	11	3	25	1
t13	12	5	27	1
t14	13	7	29	1
t15	14	9	31	0
t16	15	9	31	0
t17	16	7	29	0
t18	17	5	27	0
t19	18	3	25	0
t20	19	1	23	0

10.5. Observaciones

Algo importante a notar del código, es que todavía falta un ciclo más, cuando i=20. Esto se hace para indicar que termina a los 360°. Aunque como el programa sigue corriendo, se repite el mismo estado para el último puerto. Esto no afecta al último interruptor, ya que se mantiene en 0, pero es una redundancia que podría cambiarse en un futuro. Por otro lado, hay varias maneras de diseñar el algoritmo, por lo que permite flexibilidad la programación. Un ejemplo, es que por arbitrariedad se decidió usar en la expresión 2*i, y de ahí formar una ecuación. Se pudo decidir otro camino a empezar, o incluso multiplicar por 1 u otro número.

Por otro lado, es posible hacer la división de reloj que se mencionó en el Capítulo 9. , y de hecho hay puertos hechos para realizar esto. El Arduino Mega tiene algunos puertos destinados para señales PWM, los cuales van a estar controlado por timers. Cada timer controla dos salidas PWM. No obstante, el programa es diferente, ya que se trabaja con registros y el timer tiene un pre-escalador que se encarga de hacer la división del reloj de sistema; el factor puede ser 1, 8, 64, 256 o 1024. Esto no lo haría útil para esta aplicación, ya que se quiere trabajar con una división específica, y las señales PWM del Arduino no es lo que se requiere [21].

Teniendo el código de Arduino listo, se puede comenzar con el circuito físico del proyecto. Esto estará detallado en el siguiente capítulo.

Capítulo 11. Circuito

11.1. Componentes

Los materiales que se van a usar para el circuito están en la siguiente tabla.

Tabla 11.1. Componentes usados para el proyecto.

Cantidad	Clasificación	Parte
1	Resistencia de carga	100Ω a 25W
10	Resistencia	330Ω
10	Resistencia	1kΩ
10	Mosfet Canal P	IRF9520
10	Mosfet Canal N	IRFIZ34E
10	Optoacoplador	PC817
1	Arduino Mega 2560	
1	Cable USBB	
3	Generadores de voltaje CD	5V a 1A
3	Paneles solares	≈15V
2	Protoboards	
Necesarios	Cables Jumper	
1	Osciloscopio	Telektronix T2010
1	Puntas de osciloscopio	Tek 10X

Los materiales fueron rentados en la universidad. Como se observa no se usan los mismos componentes de la Figura 10.3, ya que no se tenían algunos disponibles, en específico los Mosfets. Por lo cual, se buscaron unos que podían trabajar a los voltajes requeridos, y se propusieron los que están en la tabla.

Con estos componentes se puede trabajar con el circuito físico. Aunque primero, lo que se hará en las siguientes secciones, es hacer la planeación de conexión en el circuito. Primero se hace la simulación en Multisim y luego se usa Tinkercad para poder simular los protoboards.

11.2. Multisim

En esta plataforma, se mezcla lo que se hizo en PSIM (Figura 8.1) y el circuito visto en la Figura 10.3. No obstante, como no es tan sencillo realizar la simulación de la salida de Arduino, se decidió remplazar con un switch ideal sencillo, para mostrar el abrir o cerrar el Mosfet sin tomar en cuenta los ángulos. Como en la librería no se encontraron los Mosfets y optoacopladores propuestos, se usaron otros componentes. El fin de Multisim en este trabajo, no es para hacer una simulación completa del inversor, sino solo es para entender las conexiones. Además, si se corre la simulación, si se puede ver en el osciloscopio que trabaja con cinco niveles, sin embargo, va a marcar continuamente que existe un error. La siguiente figura es el resultado del circuito.

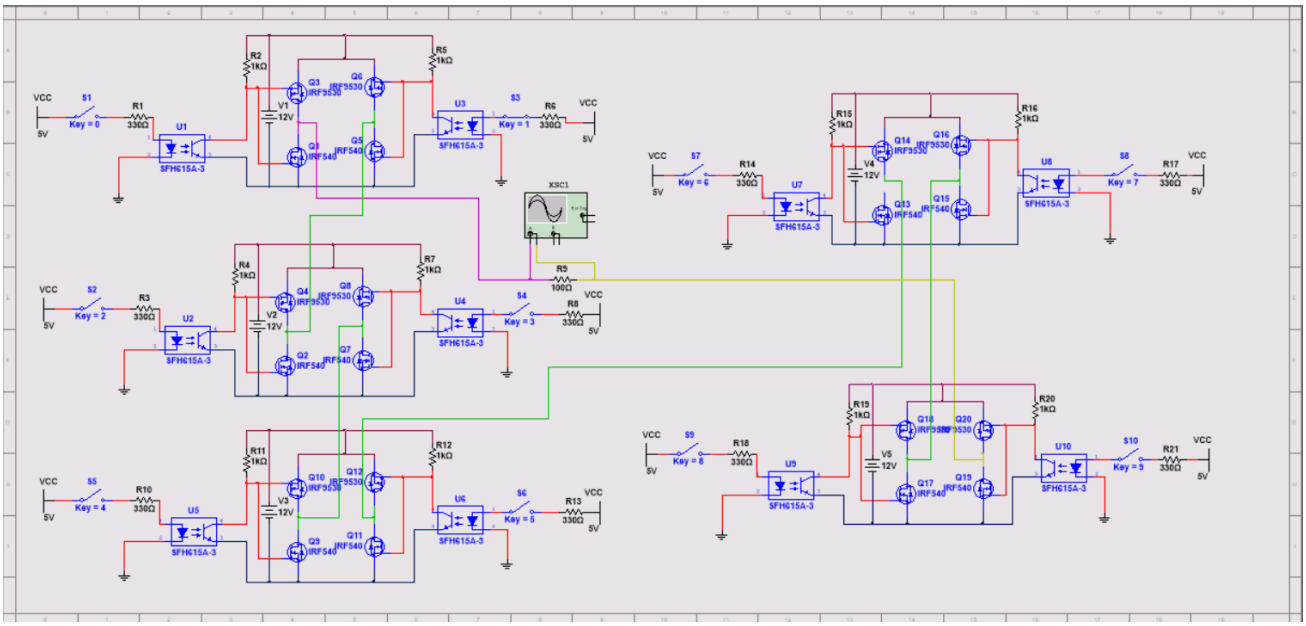


Figura 11.1. Circuito del inversor hecho en Multisim.

11.3. TinkerCad

Teniendo el circuito en Multisim, es posible plantear un diseño en Tinkercad, el cual va a asemejar el protoboard físico. Como planteamiento, se propone realizar los cinco puentes en un solo protoboard, contando los optoacopladores, los interruptores, las resistencias de 1k, lado de las resistencias de 330, el polo positivo de VCC y la tierra del Arduino. Por lo tanto, se hace un espacio equitativo, en el que se divide el protoboard en cinco secciones iguales. Para el segundo protoboard, se planea colocar las salidas del Arduino, el otro lado de las resisitencias de 330, el polo negativo de VCC conectadas a su propia tierra, y por último la

carga. El planteamiento por lo tanto, queda como se muestra en la siguiente figura.

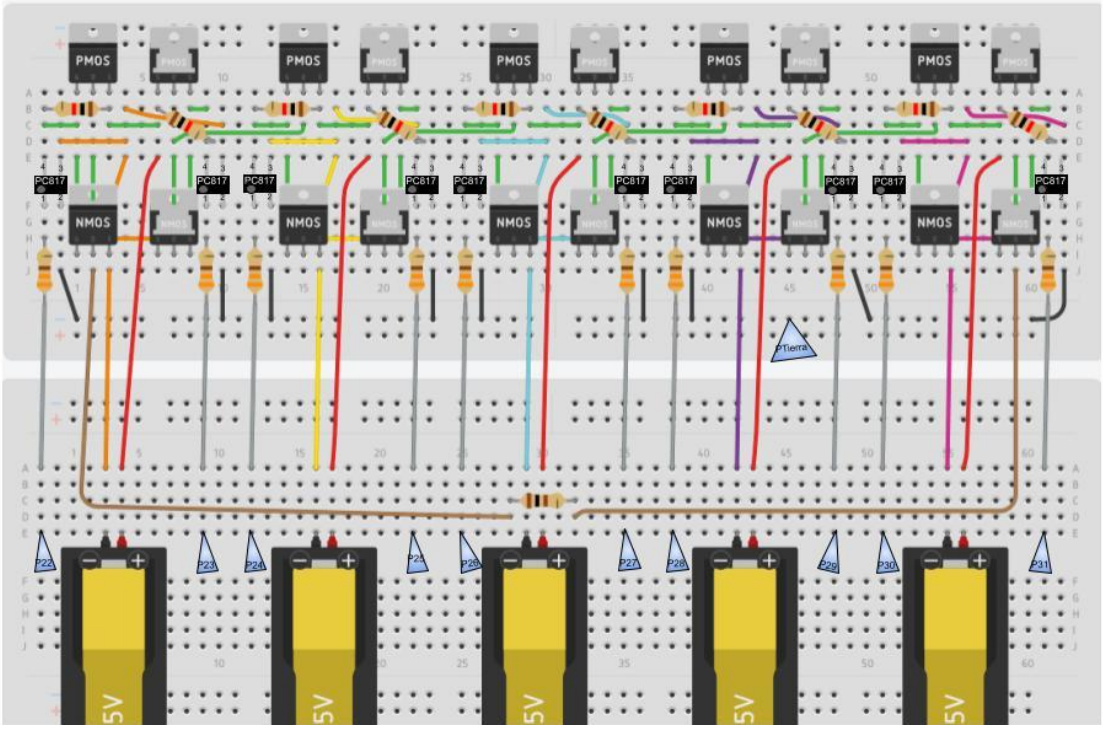


Figura 11.2. Circuito del inversor hecho en Tinkercad.

Algo que posiblemente se pueda notar, es que el circuito no fue hecho completamente en Tinkercad. Esto se debe, es que los siguientes componentes no estaban en la plataforma: un optoacoplador de cuatro pines y la salida del Arduino Mega 2560. No obstante, si hay manera de colocar otro Arduino, aunque se decidió que era más fácil representar las salidas como triángulos. La primera versión que se hizo está mostrada en la Figura A.5, aunque se quiso simplificar las tierras de potencia, ya que es más fácil de entender. Por otro lado, el diseño propuesto puede traer dificultades, ya que es compacto. Esto significa, que no es tan fácil de leer las conexiones y puede ser fácil cometer un error. No obstante, se quiso hacer de esta manera, ya que se quería tener un diseño que no fuera tan grande y pudiera ahorrar espacio.

Con esto listo, es posible trabajar con el circuito físico. En la siguiente sección se muestra esto.

11.4. Físico

Basandosé en el diseño de Tinkercad, se completó el siguiente circuito:

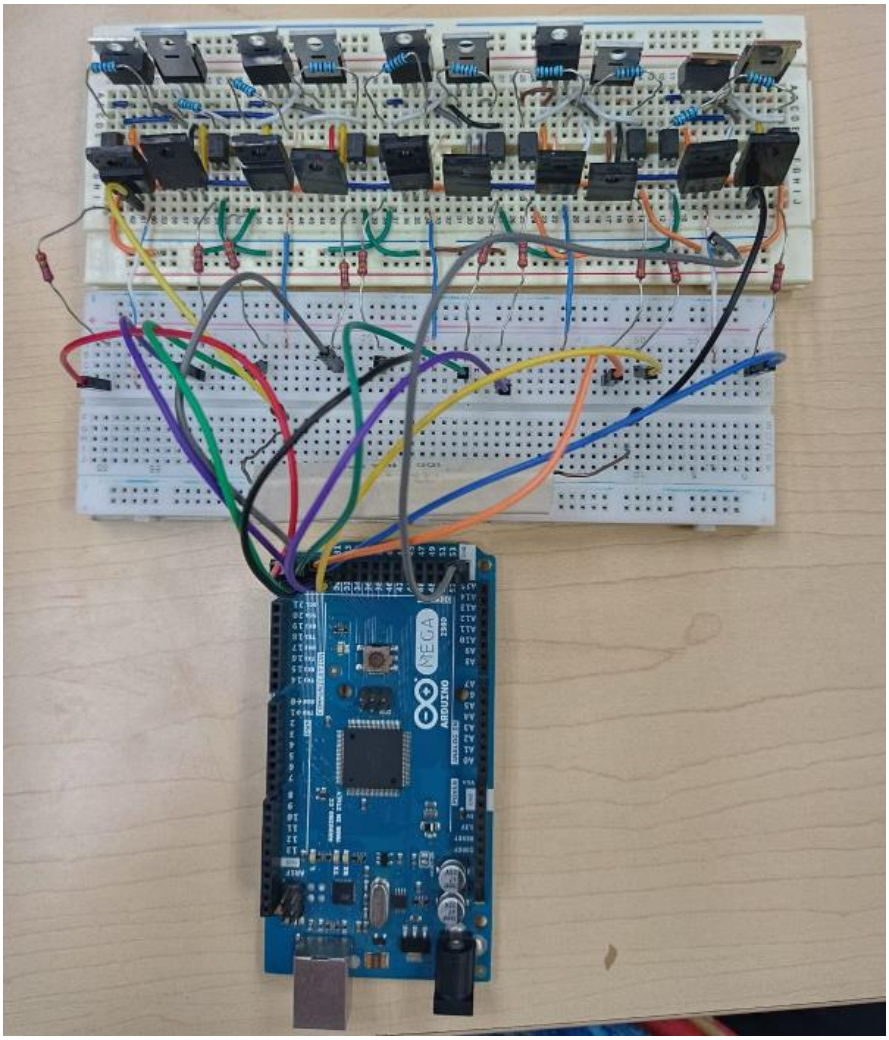


Figura 11.3. Circuito físico.

Se fue probando de nivel a nivel, hasta completar el quinto. Teniendo el circuito físico listo, es posible ver cuál es su resultado.

Capítulo 12. Resultados

Para ir comprobando que esté funcionando correctamente el código, se fue probando nivel por nivel hasta llegar al quinto. La Figura 12.1 corresponde un inversor de dos puentes H, la Figura 12.2 es de uno de tres puentes, la Figura 12.3 muestra uno de cuatro puentes y la Figura 12.4 es de cinco. Al generar diferentes tipos de niveles con el mismo código, se muestra que se realizó de manera exitosamente en este sentido. No obstante, el código hecho en Arduino tiene algunas limitaciones por arreglar. Para empezar, al observar la frecuencia que se marca en el osciloscopio, se ve que no es igual a 60Hz. Esta característica se puede ver desde la salida del Arduino. Se puede notar que, al aumentar los niveles, la frecuencia iba disminuyendo; se empezó con 58.04Hz para dos puentes y se terminó con 55.34Hz para cinco puentes. Es decir, se obtuvo pasó de un error experimental de 3.27% a uno de 7.77%. Por lo cual, aunque el código genera diferentes niveles, la frecuencia va a ir disminuyendo de la frecuencia ideal. Entonces, para tener un código mejor, se podría ver experimentalmente si teniendo un factor de corrección mejoraría la salida. Por otro lado, si solo se quiere acercar la frecuencia a 60Hz para el de cinco niveles, podría cambiar la constante freq por uno mayor (por ejemplo 65).

La razón por la diferencia en frecuencia puede ser debido a que hay un costo de tiempo en el que el microprocesador accede a la memoria y se corre el programa. Esto podría explicar también, que la frecuencia vaya disminuyendo por nivel. Esto es debido a las funciones *for*, cada vez que se aumenta un puente, va a haber un bucle más que se corre. Por lo que el tiempo va a aumentar por cada bucle corrido y por cada *for*. Lo interesante, es que este tiempo de diferencia sea suficiente como ir de un periodo de 16.67ms (60Hz), a uno de

17.23ms (dos puentes), a uno 18.07ms (cinco puentes). Por lo que, al cambiar de dos puentes a cinco puentes, se tardó 0.84ms más. Lo cual, en realidad es un tiempo considerablemente rápido. Aunque, posiblemente sería de ventaja si se trabajará con frecuencias más bajas, o si se busca una herramienta (microprocesador o FPGA) que trabaje a mayores frecuencias, o incluso si se tratará de optimizar el código.

Otro aspecto para notar es que en el nivel de 0V, se ve una diferencia de periodo. Es decir, se logra observar que cada media onda no tiene simetría en el nivel de 0V. De la misma manera que con la frecuencia, el efecto de la simetría se ve marcado conforme se agregan niveles. Por otro lado, el nivel de 0V es el único que tiene diferencia de duración a primera vista, en comparación con los otros niveles. La razón de este suceso es que el código permitió que esto pasara. Al añadir el "factor de corrección", se pensó que iba a mejorar la cercanía a 60Hz, y mientras esto es cierto, afectó el Δt de la Tabla 10.4, ya que a todos los tiempos se restó 50us, excepto a 0us. Por lo que analizando este criterio de diseño, que se usó para obtener una mejor frecuencia, no fue un buen criterio. Para empezar, al no restar 0us, el primer *delay*, es el único que va a ser afectado del planteamiento teórico. Es por esta razón, que se ve una característica diferente en el primer nivel de 0V. No obstante, incluir el 0V en la resta, significa que en realidad no existe algún cambio, y la idea de acercarse a la frecuencia no resultaría, ya que se estarían cancelando los 50us en la resta. Por lo que significa que añadir este factor en el delay no es una buena idea. Añadir un factor de corrección, podría ser buena opción si hay una función compleja, que tome en cuenta cada nivel de voltaje, cada puente, y la duración que incrementa al añadir un nuevo algoritmo para generalizar más el tiempo en el que toma para correr el algoritmo por cada puente. Ya que esto sería más extenso, significa que cambiar la constante freq es una mejor opción. Por lo cual, el código

propuesto (sin el factor), es una buena opción para generar teóricamente los ángulos, tiempos y diferencia de tiempos para diferentes números de puentes, pero se tendría que hacer una modificación. Es decir, se tendría que observar en el osciloscopio la frecuencia, y ver empíricamente que valor de freq es mejor para acomodar la frecuencia deseada. De esta manera, se podría implementar el código. Aunque requiere algo más de trabajo, es algo que se puede hacer rápidamente.

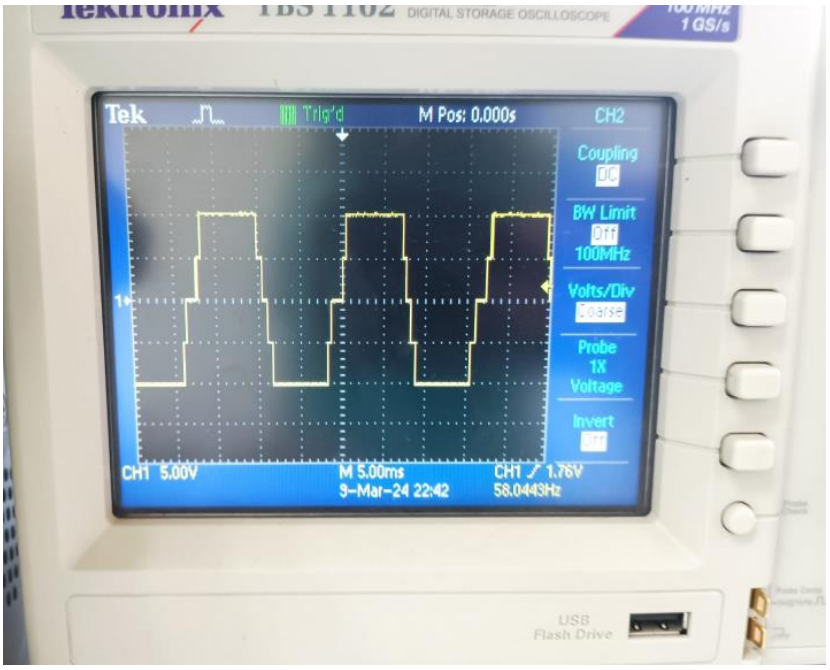


Figura 12.1. Inversor multinivel de dos puentes.

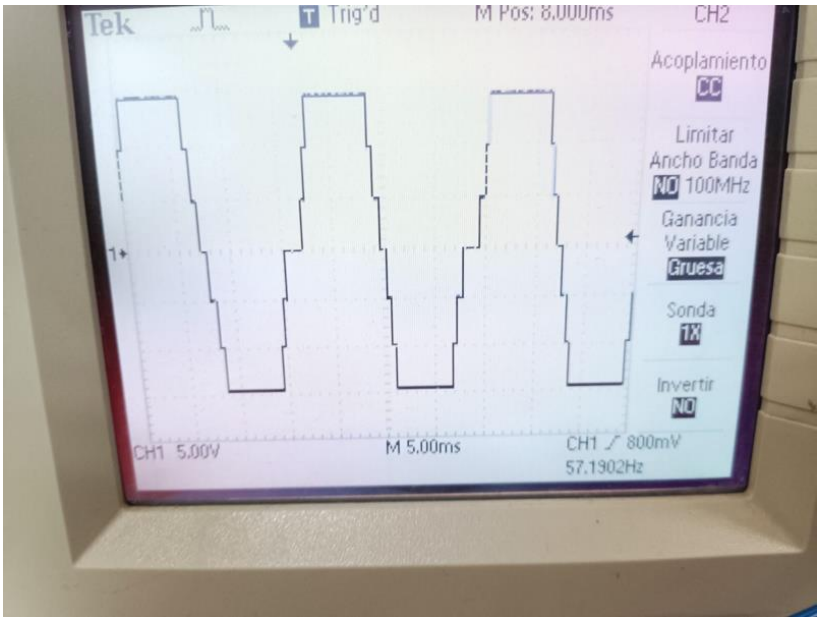


Figura 12.2. Inversor multinivel de tres puentes.

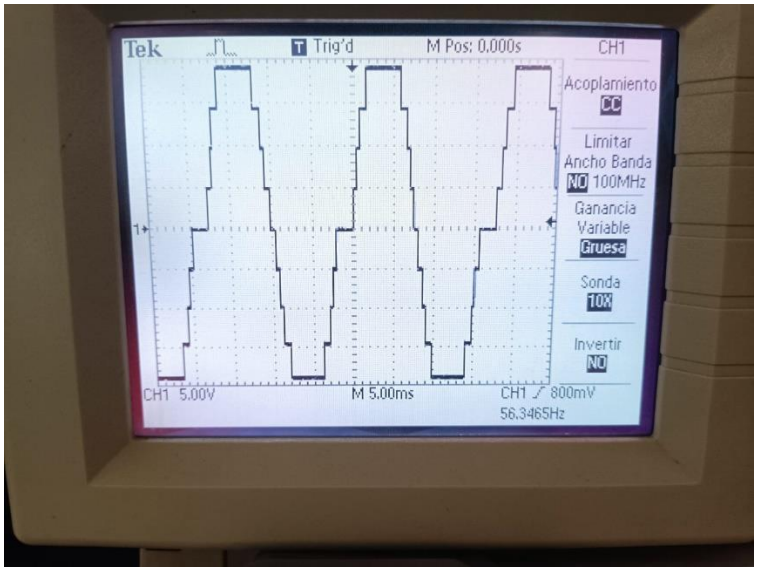


Figura 12.3. Inversor multinivel de cuatro puentes.

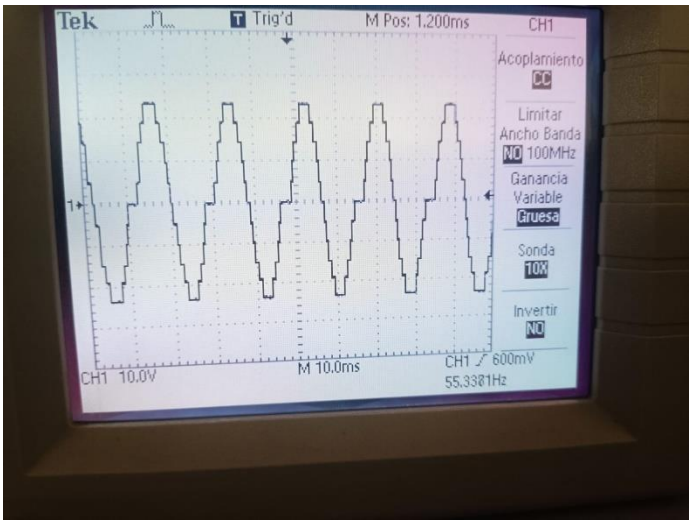


Figura 12.4. Inversor multinivel de cinco puentes.

Para mejorar el circuito, se realizaron dos cambios importantes: se añadió un potenciómetro que permite controlar la frecuencia y se optimizó el código para que se tarde menos en iniciar los niveles. Con esto, se puede llegar a los 60Hz con más certeza, y en la situación de un coche eléctrico, se puede controlar las velocidades. Además, se mejoraría la calidad de la señal. El resultado del uso del potenciómetro se ve en la Figura 12.5, este permitió ir de 10Hz a 60Hz aproximadamente. El error de los 60Hz para dos niveles es esta vez 0.50217%. Por otro lado, también se puede notar que con 10Hz, el tiempo en el que el microcontrolador calcula Δt, no es considerable para afectar la señal.

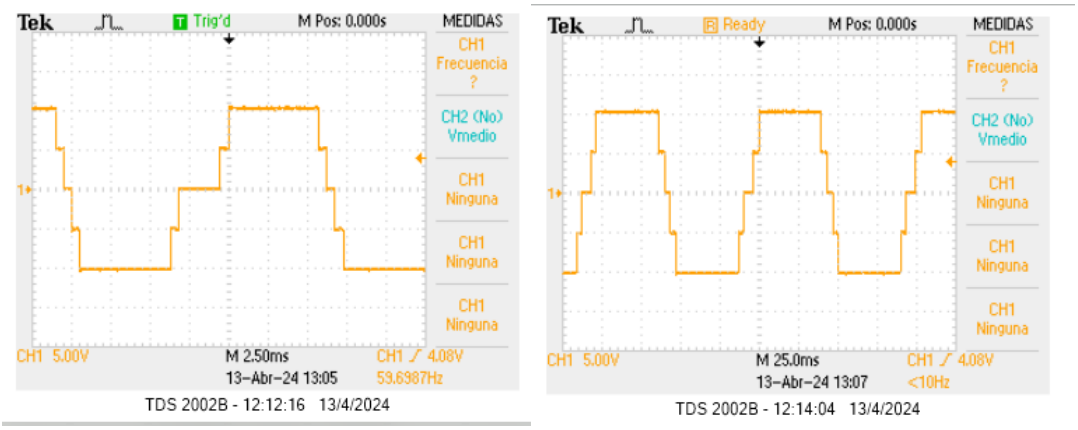


Figura 12.5. Inversor multinivel de cinco puentes usando potenciómetro para generar 60Hz (izquierda) y 10Hz (derecha).

La optimización del código también mejoró el comportamiento de la señal, ya que redujo el tiempo de cálculo. En la Figura 12.8, se puede observar que, si cambió algo el comportamiento, aunque todavía se ve un ligero retardo.

Después de las mejoras, se pudieron usar tres paneles solares. Para esta parte, el osciloscopio con el inversor se mantuvo en el salón y los paneles se llevaron al exterior (ver Figura 12.6). Para conectar los paneles al circuito, se usaron cables dúplex, los cuales iban del exterior al interior, los cuales se conectaron con cables caimán y luego jumpers. Al hacer un primer intento, se vio que la potencia era mayor que los transistores de tipo N y el resistor podían usar. Por lo cual se cambió el resistor de carga por uno de 5.6kΩ, y se cambiaron los transistores por el IRF640. Este último cambió permitió aumentar el límite de V_{DSS}; de 60V a 200V.

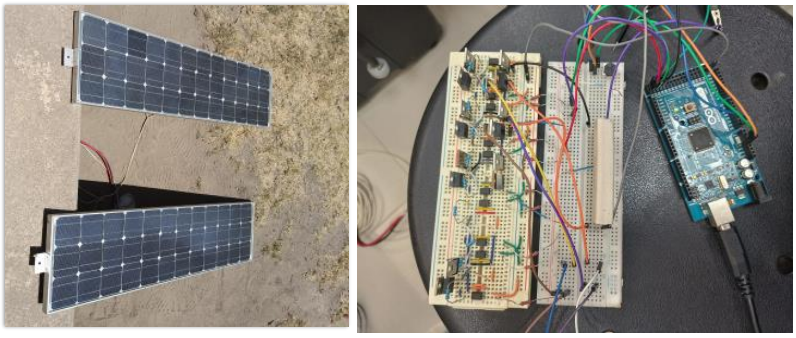


Figura 12.6. *Setup* de experimentación con paneles solares

Con esta parte modificada, se probó el circuito primero con dos paneles solares y luego con tres, los resultados se pueden ver en la Figura 12.7 y Figura 12.8. Esto muestra que el inversor puede convertir la corriente directa de los paneles solares a una corriente alterna. Los valores medidos fueron: I_{CA} de 283mA, V_{CA} es de 46.52V y P_{CA} es de 13.30W. Teniendo

más paneles, y ajustando los valores, se podría llegar a una potencia más alta. No obstante, como demostración, estos resultados son positivos.

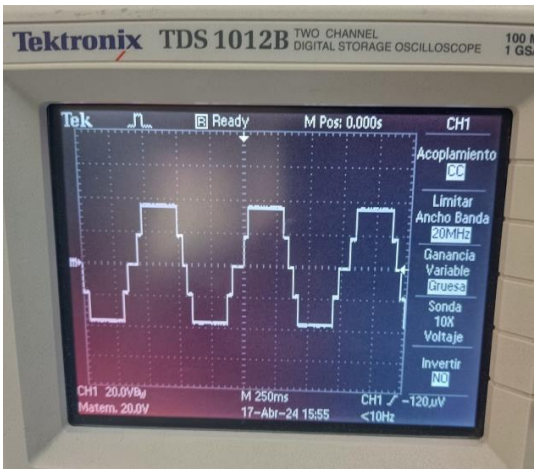


Figura 12.7. Inversor multinivel de dos puentes usando paneles solares.

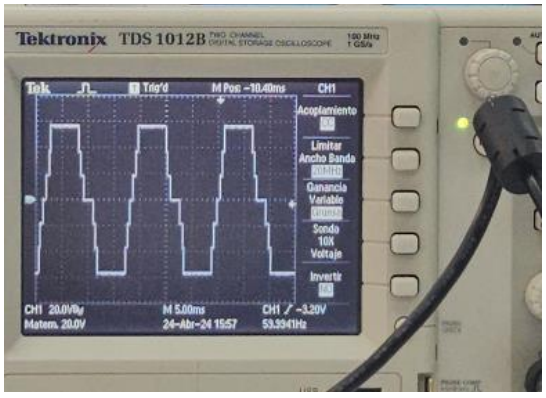


Figura 12.8. Inversor multinivel de tres puentes usando panales solares.

Capítulo 13. Conclusiones y Recomendaciones

Dando por terminado el proyecto, solo queda cerrar el trabajo con esta sección. En el transcurso del trabajo, se presentaron diferentes capítulos que representan las diferentes partes del trabajo. Empezando con Capítulo 5. Introducción, se presentó el planteamiento del problema y la perspectiva que tomó, luego en Capítulo 6. Inversores, se definieron los inversores y sus tipos, siguiendo con Capítulo 7. Inversor multinivel de puente H conectados en cascada, se dieron los detalles de este tipo de este inversor, continuando con Capítulo 8. Diseño Teórico, se reportó el procedimiento usado para entender y luego simular un inversor de cinco puesto, prosiguiendo con Capítulo 9. FPGA, en el que se hace la planeación de una implementación con el Basys3, llegando a Capítulo 10. Arduino, en el que ahora se hace la planeación con el Arduino Mega 2560, dando paso a Capítulo 11. Circuito, se hizo la planeación del circuito físico y entonces llegar al Capítulo 12. Resultados, que se muestra el performance del inversor.

Los objetivos se cumplieron, ya que se entiendo el concepto del inversor y se llevo a cabo su implementación al resolver un sistema de ecuaciones para obtener el ángulo de disparo, determinar las señales digitales para cada interruptor, usar Arduino para crear las estas señales, realizar el circuito, y por último analizar el resultado. De lo que se aprendió, con esto, es que el inversor multinivel de 5 puentes H en cascada es una buena opción para convertir una señal CD a una CA con bajo THD. Por lo que lo hace posible como una herramienta para el área de vehículos eléctricos, en el que se obtiene energía a partir de fuentes renovables como el sol y se usa para el motor de este. No obstante, para la

implementación de esta, todavía se requiere más investigación. Por ejemplo, ver otros métodos, o incluso usar otra topología inversora que sea óptima para la aplicación planteada.

Aspectos que se podrían mejorar de este trabajo, incluye ver la posibilidad de usar otros microcontroladores o incluso realizar el diseño completo con un FPGA, ya que con este, no habría el problema de la simetría. También se podría comenzar a trabajar con más niveles.

Bibliografía

[1] A. Hren, M. Truntič, y F. Mihalič, «A Survey on the State-of-the-Art and Future Trends of Multilevel Inverters in BEVs», *Electronics*, vol. 12, n.º 13, Art. n.º 13, ene. 2023, doi: 10.3390/electronics12132993.

[2] P. Anjaneya Vara Prasad y C. Dhanamjayulu, «An Overview on Multi-Level Inverter Topologies for Grid-Tied PV System», *International Transactions on Electrical Energy Systems*, vol. 2023, p. e9690344, may 2023, doi: 10.1155/2023/9690344.

[3] H. Vahedi y M. Trabelsi, «Multilevel vs Two-Level Inverters», en *Single-DC-Source Multilevel Inverters*, H. Vahedi y M. Trabelsi, Eds., Cham: Springer International Publishing, 2019, pp. 1-6. doi: 10.1007/978-3-030-15253-6_1.

[4] K. Heumann, «DC--AC POWER CONVERTERS.», *Encyclopedia of Electrical & Electronics Engineering*, vol. 5, pp. 23-44, ene. 1999.

[5] B. Ge, F. Z. Peng, y Y. Li, «Multilevel Converter/Inverter Topologies and Applications», en *Power Electronics for Renewable Energy Systems, Transportation and Industrial Applications*, 1.ª ed., H. Abu-Rub, M. Malinowski, y K. Al-Haddad, Eds., Wiley, 2014, pp. 422-462. doi: 10.1002/9781118755525.ch14.

[6] F. E. L. Monteagudo, J. de la T. y Ramos, C. A. O. Olvera, S. V. Barraza, L. O. S. Sánchez, y H. G. Ozuna, «Desempeño de inversores multinivel para energías renovables, una alternativa a la generación distribuida en México», *Ingeniería Energética*, vol. XLIV, n.º 1, pp. 94-105, 2023.

[7] G. Ezhilarasan *et al.*, «An empirical survey of topologies, evolution, and current developments in multilevel inverters», *Alexandria Engineering Journal*, vol. 83, pp. 148-194, nov. 2023, doi: 10.1016/j.aej.2023.10.049.

[8] J. I. Leon, S. Vazquez, y L. G. Franquelo, «MULTILEVEL CONVERTERS – CONTROL AND OPERATION IN INDUSTRIAL SYSTEMS», en *Power Electronics in Renewable Energy Systems and Smart Grid*, 1.ª ed., B. K. Bose, Ed., Wiley, 2019, pp. 219-270. doi: 10.1002/9781119515661.ch4.

[9] D. W. Hart, *Power electronics*. New York: McGraw-Hill, 2011.

[10] W. Mendenhall, R. J. Beaver, y B. M. Beaver, *Introduction to probability and statistics*, 15th Edition. Australia ; United States: Cengage, 2020.

[11] M. H. Rashid, *Power electronics: devices, circuits, and applications*, Fourth edition. Upper Saddle River, NJ: Pearson, 2014.

[12] M. Salman, A. Basit, M. S. Khalid, y A. Qamar, «Reduction in Total Harmonic Distortion of Cascaded H-Bridge Multilevel Inverter with Using Phase Method», en *2018 Clemson University Power Systems Conference (PSC)*, Charleston, SC, USA: IEEE, sep. 2018, pp. 1-6. doi: 10.1109/PSC.2018.8664070.

[13] A. Pang y P. Membrey, «What Is an FPGA and What Can It Do?», en *Beginning FPGA: Programming Metal: Your brain on hardware*, A. Pang y P. Membrey, Eds., Berkeley, CA: Apress, 2017, pp. 3-12. doi: 10.1007/978-1-4302-6248-0_1.

[14] Mano, M. Morris, *Diseño Digital*. México: Pearson Education de México, 2013.

[15] Tocci, Ronald J., Widmer, Neal S., y Moss, Gregory L., *Sistemas digital Principios y aplicaciones*. Edo. de México: Pearson Educación de México, 2007.

[16] «Basys 3 Reference Manual - Digilent Reference». Accedido: 31 de marzo de 2024. [En línea]. Disponible en: https://digilent.com/reference/programmable-logic/basys-3/reference-manual

[17] B. J. LaMeres, *Quick start guide to VHDL*. Cham, Switzerland: Springer, 2019.

[18] Vikrant Vij, *Microcontroller and Embedded System*, Second edition. New Delhi: Laxmi Publications Pvt Ltd, 2021.

[19] Arduino, «Product Reference Manual. Arduino® MEGA 2560 Rev3». 2024.

[20] YOUR-NAME, «How to make simple H-Bridge from Mosfet ...» Accedido: 10 de marzo de 2024. [En línea]. Disponible en: https://onebyzeroelectronics.blogspot.com/2016/03/how-to-make-simple-h-bridge-from-mosfet.html

[21] «Secrets of Arduino PWM | Arduino Documentation». Accedido: 2 de abril de 2024. [En línea]. Disponible en: https://docs.arduino.cc/tutorials/generic/secrets-of-arduino-pwm/

Anexos

A. *Esquemas*

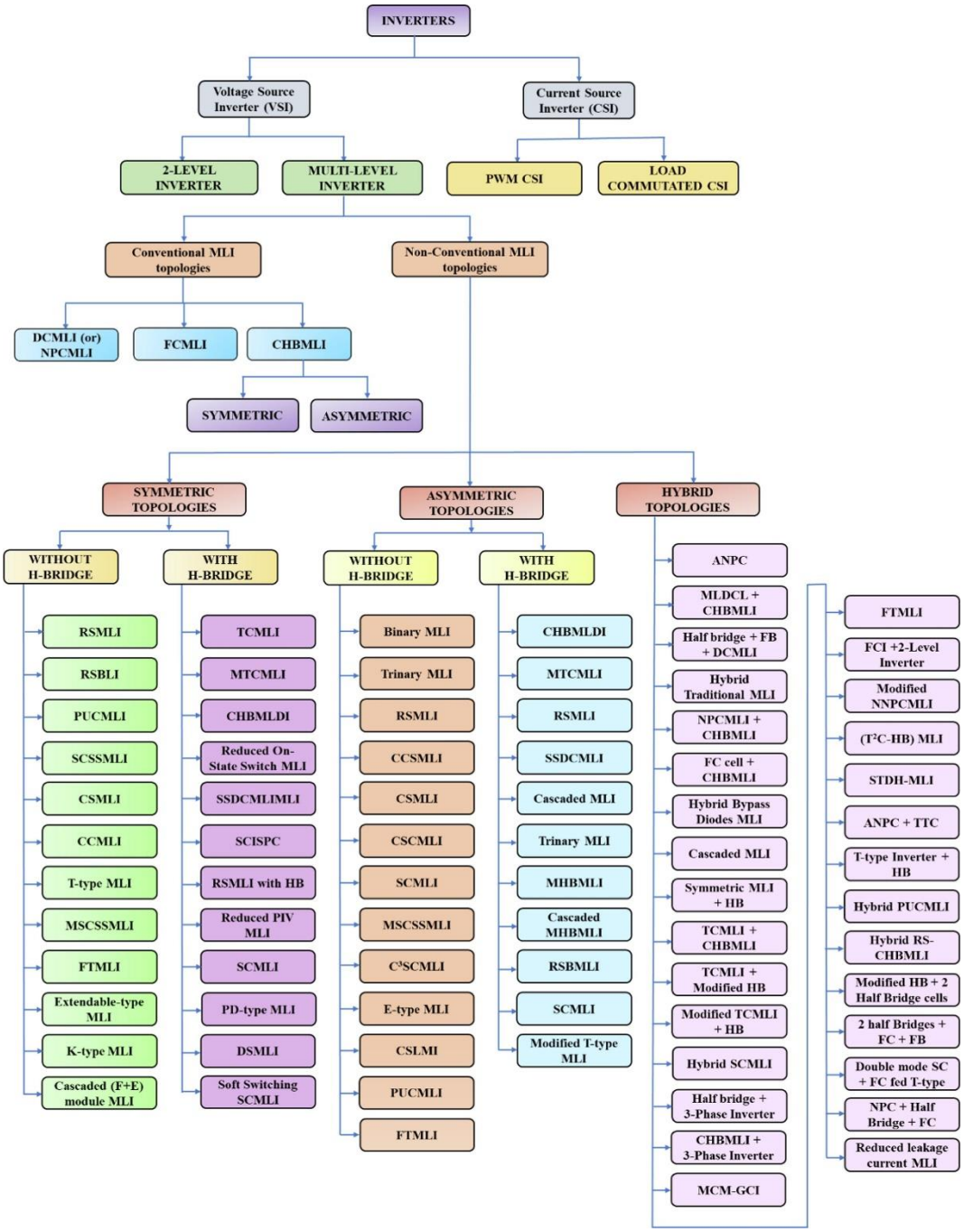


Figura A.1. Tipos de inversores [7].

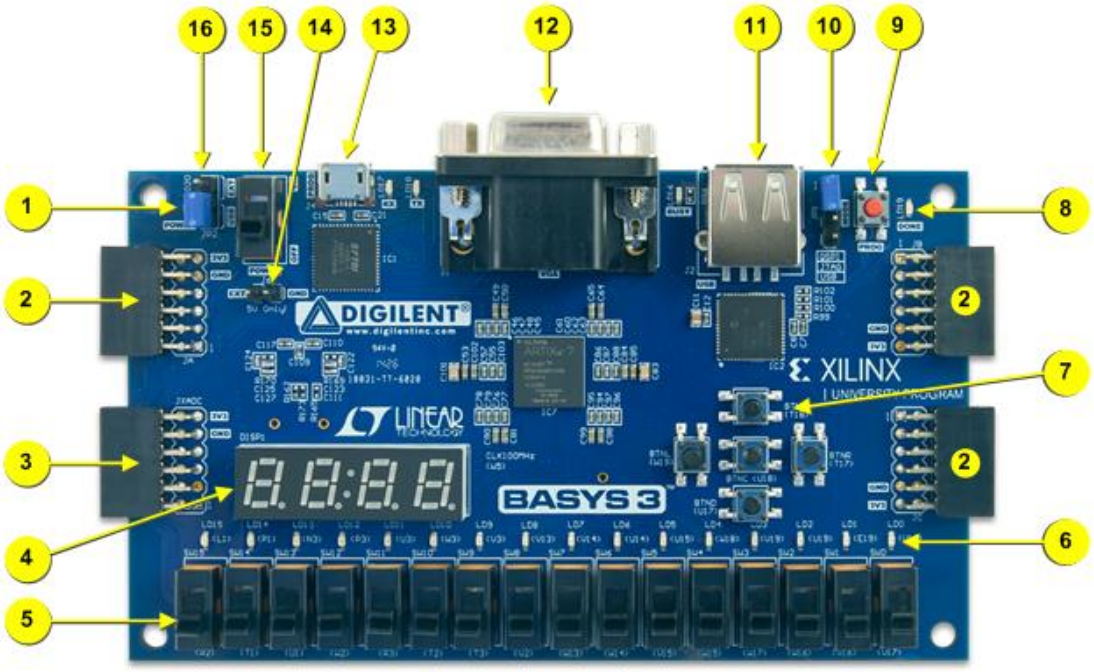
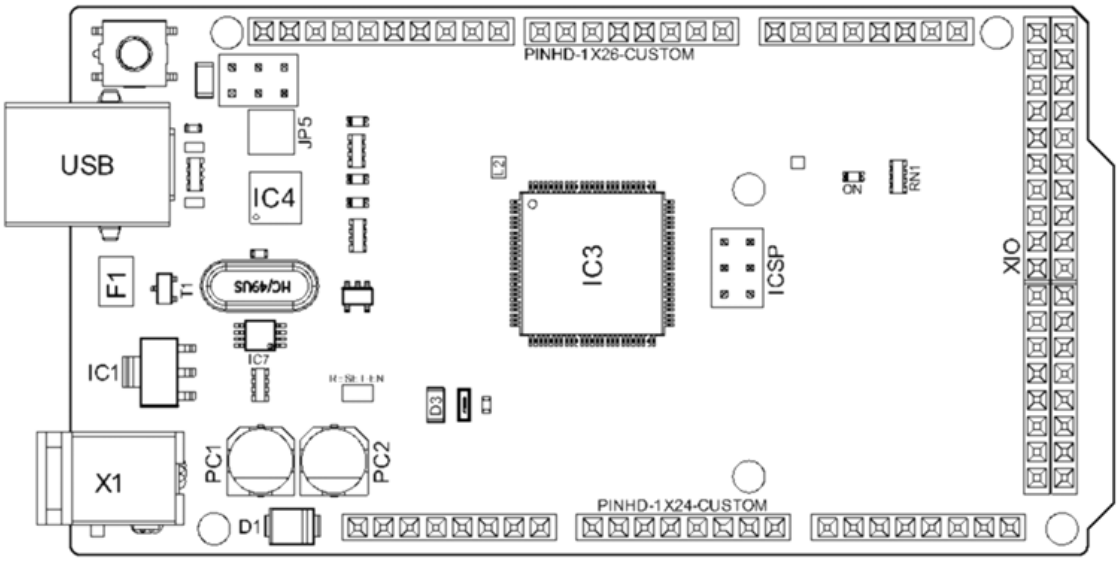


Figure 1. Basys3 board features

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod connector(s)	10	Programming mode jumper
3	Analog signal Pmod connector (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Figura A.2. Componentes del Basys3 [16].

Front View



Arduino MEGA Top View

Ref.	Description	Ref.	Description
USB	USB B Connector	F1	Chip Capacitor
IC1	5V Linear Regulator	X1	Power Jack Connector
JP5	Plated Holes	IC4	ATmega16U2 chip
PC1	Electrolytic Alumninum Capacitor	PC2	Electrolytic Alumninum Capacitor
D1	General Purpose Rectifier	D3	General Purpose Diode
L2	Fixed Inductor	IC3	ATmega2560 chip
ICSP	Connector Header	ON	Green LED
RN1	Resistor Array	XIO	Connector

Figura A.3. Topología del Arduino Mega 2560 [19].

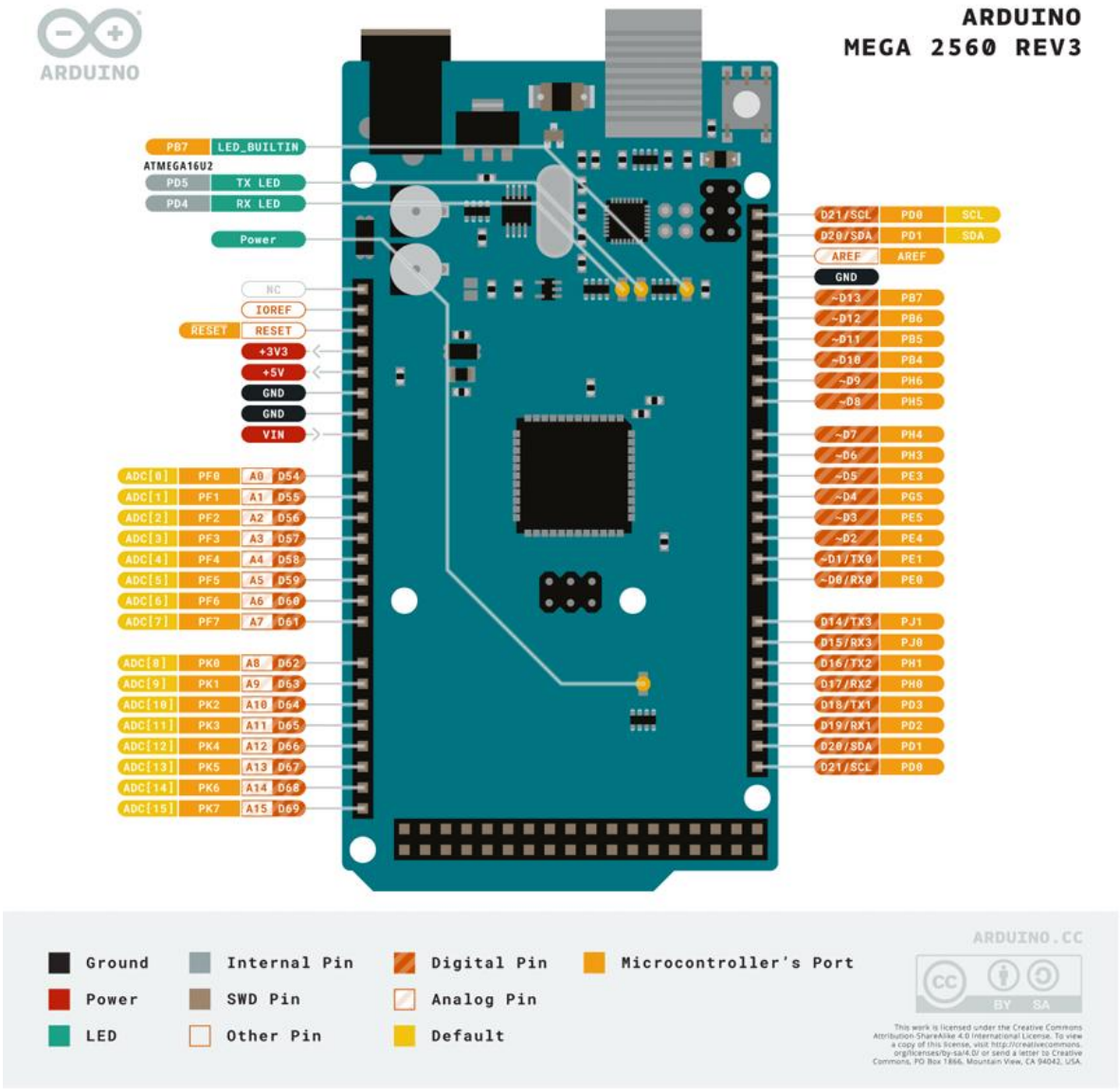


Figura A.4. Conectores para el Arduino Mega 2560 [19].

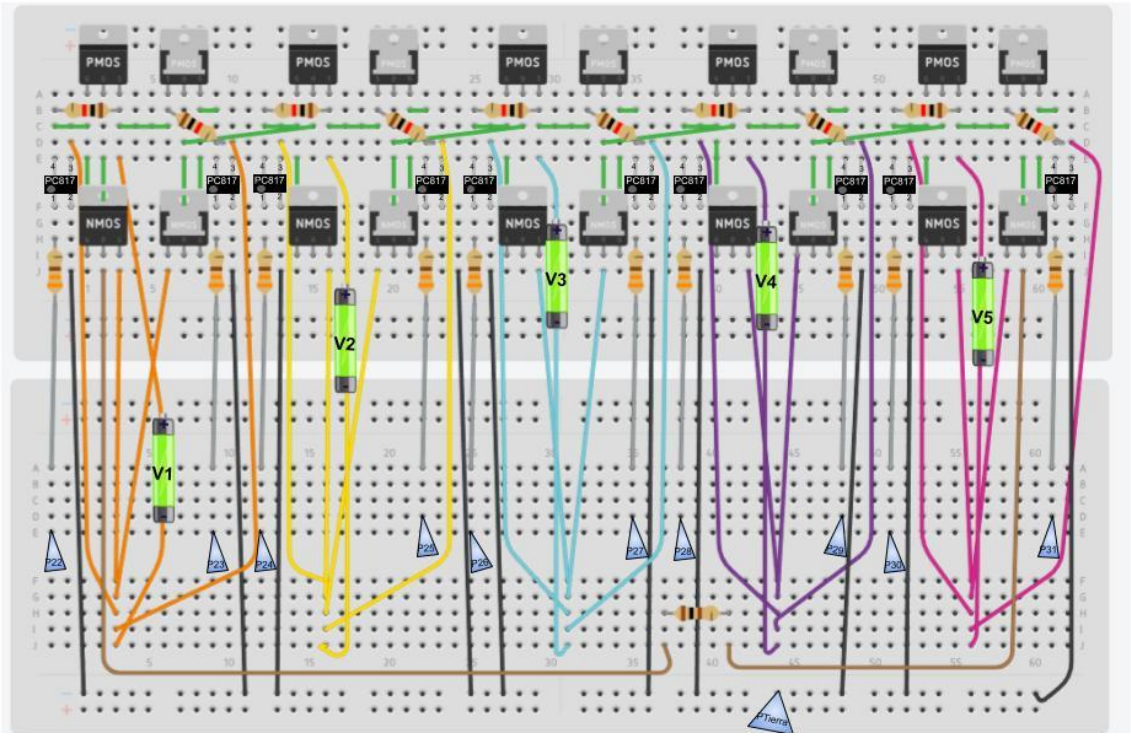


Figura A.5. Primera versión hecha en Tinkercad del inversor.

B. Código de Matlab

```
syms a1 a2 a3 a4 a5
F1=cos(5*a1)+cos(5*a2)+cos(5*a3)+cos(5*a4)+cos(5*a5);
F2=cos(7*a1)+cos(7*a2)+cos(7*a3)+cos(7*a4)+cos(7*a5);
F3=cos(11*a1)+cos(11*a2)+cos(11*a3)+cos(11*a4)+cos(11*a5);
F4=cos(13*a1)+cos(13*a2)+cos(13*a3)+cos(13*a4)+cos(13*a5);
F5=cos(a1)+cos(a2)+cos(a3)+cos(a4)+cos(a5)-5*(0.8);

sol=solve([F1==0,F2==0, F3==0, F4==0, F5==0],[a1 a2 a3 a4 a5]);
```

Figura B.1. Código de Matlab para resolver sistema de ecuaciones.

C. Código de VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;

entity SP is
port ( clk: in std_logic;
clock_out: out std_logic);
end SP;

architecture bhv of SP is

signal count: integer:=1;
signal tmp : std_logic := '0';

begin

process(clk)
begin
if(clk'event and clk='1') then
count <=count+1;
if (count = 30417) then
tmp <=NOT tmp;
elsif(count=802917) then
tmp <= NOT tmp;
elsif(count=1666667) then
tmp <= '0';
count<=0;
end if;
end if;
clock_out <= tmp;
end process;
```

Figura C.1. Código VHDL para el interruptor S1.1.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;

entity SP is
port ( clk: in std_logic;
clock_out: out std_logic);
end SP;

architecture bhv of SP is

signal count: integer:=1;
signal tmp : std_logic := '1';

begin

process(clk)
begin
if(clk'event and clk='1') then
count <=count+1;
if (count = 30417) then
tmp <=NOT tmp;
elsif(count=802917) then
tmp <= NOT tmp;
elsif(count=1666667) then
tmp <= '1';
count<=0;
end if;
end if;
clock_out <= tmp;
end process;
end bhv;
```

Figura C.2. Código VHDL para el interruptor S1.2.

D. Código de Arduino

```
const int level=5, freq=60;
const float alpha[level]={6.57, 18.94, 27.18, 45.14, 62.24};
float theta[level][4];
float thetaSeq[level*4], timeSeqDiff[level*4+1]; //S(groupLevel)(C or D)(On or Off)
int port[level*2];
void setup() {
  // put your setup code here, to run once
  portValues();
  for(int i=0; i<level*2; i++){
    pinMode(port[i], OUTPUT);
  }
  Serial.begin(9600);
  degree();
}

void loop() {
  // put your main code here, to run repeatedly:
  int p;  //p is to select the port
  bool status; // refers to open (0) or close (1) the switch
  time();
  for(int i=0; i<level*4+1;i++){
    delayMicroseconds(timeSeqDiff[i]-50);
    if(i<level){
      p=port[2*i];
      status=1;
    }
    else if(i<2*level){
      p=port[4*level-2*i-2];
      status=0;
    }
    else if(i<3*level){
      p=port[2*i-level*4+1];
      status=1;
    }
    else if(i<4*level){
      p=port[-2*i+8*level-1];
      status=0;
    }
     digitalWrite(p,status);

    /*Serial.print(timeSeqDiff[i]);
    Serial.print(" =Time. ");
    Serial.print(p);
    Serial.print(" =Port. ");
    Serial.print(status);
    Serial.println(" =Status.");*/
  }
  Serial.println("End");
}
void degree(){
  for(int i=0; i<level; i++){
    theta[i][0]=alpha[i];
    theta[i][1]=180-alpha[i];
```

```
    theta[i][2]=180+alpha[i];
    theta[i][3]=360-alpha[i];
  }
}

void swap(float arr[],int a,int b){
  float c;
  c= arr[a];
  arr[a] = arr[b];
  arr[b] = c;
}
int partition(float arr[],int low,int high){
  //choose the pivot

  float pivot=arr[high];
  //Index of smaller element and Indicate
  //the right position of pivot found so far
  int i=(low-1);

  for(int j=low;j<=high;j++){
    //If current element is smaller than the pivot
    if(arr[j]<pivot){
      //Increment index of smaller element
      i++;
      swap(arr,i,j);
    }
  }
  swap(arr,i+1,high);
  return (i+1);
}

// The Quicksort function Implement

void quickSort(float array[], int low, int high){
  // when low is less than high
  if(low<high)
  {
    // pi is the partition return index of pivot
    int pi=partition(array,low,high);

    //Recursion Call
    //smaller element than pivot goes left and
    //higher element goes right
    quickSort(array,low,pi-1);
    quickSort(array,pi+1,high);
  }
}
void time(){
  float timeSeq[level*4+1];
  for(int i=0; i<level;i++){
    for(int j=0; j<4;j++){
      thetaSeq[i*4+j]=theta[i][j];
    }
  }
  quickSort(thetaSeq, 0, (level*4)-1);
  for(int i=0; i<level*4; i++){
```

```
    timeSeq[i]=thetaSeq[i]*pow(10,6)/(360*freq);
  }
  timeSeq[level*4]=pow(10,6)/freq;
  timeSeqDiff[0]=timeSeq[0];
  for(int i=1; i<level*4+1; i++){
    timeSeqDiff[i]=timeSeq[i]-timeSeq[i-1];
  }
}
void portValues(){
  for(int i=0; i<level*2; i++){ //work max 16 levels
    port[i]=22+i;
  }
}
```