Universidad Zaragoza
1542

Trabajo Fin de Grado

Sistema de verificación para circuitos integrados en laboratorios de electrónica

Autor

Raúl Arenaz Callao

Directores

José Barquillas Pueyo

José María García del Pozo Faldós

Departamento de Ingeniería Electrónica y Comunicaciones

Junio 2015

Resumen

Sistema de verificación para circuitos integrados en laboratorios de electrónica

El objetivo es el diseño y construcción de un prototipo electrónico para la rápida verificación del correcto funcionamiento de una serie de circuitos integrados (ICs) utilizados habitualmente en las prácticas de laboratorio y proyectos/trabajos.

El trabajo se ha realizado en varias fases. Una primera fase, de revisión bibliográfica, ha marcado, en líneas muy generales, los pasos a seguir para la construcción del sistema. Después se ha realizado la selección de los componentes mecánicos y electrónicos necesarios, de acuerdo con nuestras necesidades (microcontrolador, display, etc), y teniendo en cuenta el costo y prestaciones de los mismos.

Tras ello se comienza el trabajo de laboratorio, realizando medidas y comprobaciones experimentales sobre chips defectuosos, centrando especialmente la atención en su consumo, que constituye un claro indicio de su mal funcionamiento.

De hecho, la programación del microcontrolador comienza por medir este consumo y desechar el chip si es demasiado elevado.

En paralelo, se ha diseñado y verificado software específico para detectar posibles errores en los chips, mediante la comprobación de su funcionamiento lógico.

Cabe destacar que la labor desarrollada en el laboratorio ha proporcionado competencias en aspectos prácticos experimentales y en la construcción de interfaces electrónicos auxiliares.

El resultado final ha sido la construcción de un prototipo capaz de detectar chips defectuosos, con la capacidad de ser reprogramado y ampliado fácilmente para posibles nuevos dispositivos.

INDICE:

1. Introducción

Un problema clásico en todos los laboratorios donde se realizan prácticas con circuitos electrónicos, analógicos o digitales, es la existencia de componentes de circuito defectuosos, dando lugar a que los montajes experimentales realizados por los alumnos no funcionen adecuadamente.

Así, además de desviar su atención de los objetivos básicos de la práctica a realizar, se produce una pérdida de tiempo para localizar el componente defectuoso e incluso una sensación frustrante con la consiguiente pérdida de interés.

Sin descartar posibles defectos ya en el momento de su adquisición, esta situación es habitualmente el resultado de su uso continuo “por muchas manos”, y no siempre en las mejores condiciones, o de conexiones claramente incorrectas.

El problema se complica por el hecho de que, por error o despiste, algunos de los componentes estropeados son devueltos a su caja de almacenamiento original, mezclando de esta forma componentes con un funcionamiento correcto con otros defectuosos.

Lógicamente, los laboratorios de prácticas del Dpto. de Ingeniería y Comunicaciones en la Facultad de Ciencias no son ajenos a este problema, especialmente significativo en la asignatura de Electrónica Digital, donde se utiliza un número elevado de chips, similares en formato de encapsulado y apariencia, pero con funcionalidades muy distintas.

La verificación manual de estos dispositivos no resulta fácil ni rápida, ya que se requiere montar circuitos concretos de test para cada uno de ellos y someterlos a una serie de pruebas para determinar si su funcionamiento es el correcto.

Aunque se realiza una comprobación de este tipo, de forma selectiva en función de la experiencia del personal técnico de mantenimiento de los laboratorios, es prácticamente imposible extender esta labor periódicamente a todos los chips disponibles.

Así, sería de gran ayuda disponer de un sistema automático de verificación que determine, a partir de la referencia del chip, y tras la realización de una serie de test, si funciona adecuadamente, de acuerdo a las especificaciones del fabricante.

En la actualidad existen diversos sistemas de verificación de este tipo, resultado de proyectos específicos, como el nuestro, o comercializados por empresas (Tesca S.A., Kitek, Anexo IV) de instrumentación de laboratorio.

Su principal desventaja es su elevado costo, debido a que son aparatos capaces de testear una gran cantidad de componentes, la mayoría de los cuales están fuera de nuestro interés. Otro inconveniente se presenta a la hora de intentar reprogramar el sistema para mejorar o ampliar sus prestaciones, el software es cerrado y su modificación resulta imposible.

A partir de estas consideraciones surgió el presente proyecto: la construcción de un sistema de verificación de bajo costo, adaptado a las necesidades específicas de los laboratorios del área de Electrónica, y totalmente abierto a ampliaciones/modificaciones futuras para incluir nuevos componentes.

Para adquirir unas ideas previas que nos ayuden a construir un prototipo con estas características se han revisado distintas referencias, en las que se especifican los componentes utilizados e incluso se comentan, de una forma general, posibles rutinas de verificación de los chips [Cia87] [Pan04] [Man13].

2. Objetivos

En concreto, los objetivos de este proyecto son diseñar, construir y poner a punto un sistema automático de verificación de los chips digitales utilizados habitualmente en las prácticas y trabajos/proyectos asociados de la asignatura Sistemas Digitales del Grado de Física, en este momento centrados en tecnología CMOS (74HCXX).

Este sistema tendrá una aplicación real e inmediata, permitiendo verificar los chips disponibles actualmente en el laboratorio y garantizando así un alto margen de fiabilidad de los mismos de cara al próximo curso académico.

Además, la disponibilidad de este sistema permitirá una comprobación rutinaria mucho más rápida y frecuente de los mismos, siendo posible realizarla en los periodos entre prácticas para detectar posibles “nuevos” chips defectuosos.

Así mismo, y dentro del ámbito formativo de los proyectos Fin de Grado, es un objetivo paralelo y no menos importante la adquisición por parte del alumno de conocimientos complementarios relacionados con la materia en cuestión y no necesariamente incluidos en los programas docentes.

En nuestro caso, dado que constituyen el núcleo del sistema, se realiza un estudio a fondo de los microcontroladores, de su arquitectura y programación, muy superior al expuesto en la correspondiente asignatura del Grado de Física.

Se abordan aspectos nuevos del diseño y construcción de sistemas electrónicos, como es el interconexionado entre los distintos componentes del sistema, para lo cual se ha diseñado una placa de circuito impreso específica.

Es necesario también identificar los diversos fallos encontrados en una recopilación de chips defectuosos para intentar detectarlos mediante los adecuados procesos de test, así como un estudio detallado de las etapas de entrada y salida de la tecnología CMOS.

Además, para la construcción física del prototipo se requiere ensamblar sus diversos componentes electrónicos y mecánicos, dándole de este modo una forma funcional lo más cómoda y fácil de usar.

3. Desarrollo

3.1 Hardware utilizado

Básicamente, para llevar a cabo la comprobación funcional de un chip digital se requiere un zócalo donde colocarlo [Far02][Cia87], una red de interruptores que permitan conectar o desconectar los terminales de datos del sistema a los pines del chip y un sistema de control que vaya aplicando vectores de test en las entradas y verifique la validez de las respuestas obtenidas.

La alternativa idónea para esta labor es utilizar un microcontrolador (μC) que, bajo el control del correspondiente software (lenguaje C), es capaz de generar vectores binarios (0/1) en los terminales configurados como salidas y leer los datos presentes en los configurados como entradas [Pan04][Cia87].

Dado que se trata de una reconfiguración dinámica de puertos se puede adaptar a la arquitectura de entradas y salidas de cada chip a verificar sin más que realizar los oportunos cambios en el software de control.

Con el fin de adquirir una idea global del prototipo diseñado se procede a continuación a una breve exposición de sus diversos bloques, para realizar posteriormente una descripción más detallada de los más importantes.

Placa de Microcontrolador

El microcontrolador elegido ha sido el Atmel ATMega2560, insertado en una placa Arduino que lleva su misma referencia, y una de las más usadas en la actualidad, por su bajo coste y sus elevadas prestaciones

Dicha placa incluye todos los componentes periféricos necesarios para el correcto funcionamiento del μC (reloj, regulador, etc). Permite utilizar cerca de 80 pines del μC, los cuales pueden configurarse como entrada/salida, algunas analógicas, o como buses de comunicación con otros dispositivos (I2C, SPI, puerto serie)

En cuanto a su capacidad de memoria incluye 256 Kb de memoria Flash, 8 Kb de RAM y 4 Kb de EEPROM, más que suficiente para nuestro propósito.

Además, posee un entorno de desarrollo muy completo y fácil de usar, dentro de la filosofía de software libre, que una vez instalado en el ordenador permite escribir, depurar, compilar y cargar en la memoria Flash del μC el código objeto correspondiente a través del puerto USB conectado a la placa Arduino.

Zócalo de inserción del chip

Se utiliza un zócalo tipo ZIF (Zero Insertion Force), que permite la inserción y retirada de chips de una forma cómoda y rápida (figura 1), sin tener que realizar ningún tipo de presión, evitando así dañar sus terminales. Está provisto de una palanca de fijación para conseguir que los chips queden firmemente sujetos y la conexión eléctrica sea correcta.

Aunque los circuitos integrados a verificar tienen un máximo de 16 pines (2x8) se ha seleccionado un zócalo ZIF de 24 (2x12) para dejar abierta la posibilidad de extender el sistema a chips más complejos con el mínimo de modificaciones en el montaje.

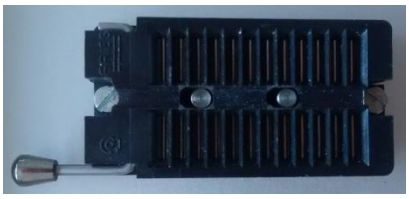

Figura 1

Red programable de interruptores

La interconexión entre el μC y los pines del chip a verificar se realiza a través de los puertos C y L (8 bits), que se conectan al zócalo ZIF mediante dos buses de 8 líneas.

Su adecuada configuración como entradas o salidas permite escribir en las entradas del chip los vectores de test necesarios para cada una de las comprobaciones funcionales y leer en las salidas los vectores respuesta correspondientes, además de proporcionarle la tensión de alimentación (V_{cc} y GND)

Para evitar daños al sistema o al dispositivo a verificar, esta red de interruptores debe aislar totalmente el dispositivo del μC cuando se inserta un nuevo chip y cada vez que se cambie de vector de test.

Como alternativa más simple para el diseño de esta red podrían usarse relés, pero presentan los inconvenientes de ser caros, lentos, poseen un consumo elevado y tienen una vida limitada.

Una solución mejor y más actual es utilizar interruptores de estado sólido, con prestaciones notablemente superiores en velocidad, costo y consumo de potencia.

Tras una revisión de diversos circuitos integrados de este tipo se ha seleccionado el ADG715 de Analog Devices, que incluye 8 interruptores bidireccionales CMOS controlados por bus I2C, cuyo funcionamiento y características detalladas se exponen más adelante.

Display LCD

Como elemento básico de visualización se utiliza un display LCD de 20x4 caracteres (figura 2) para mostrar una lista de las referencias de chips que el dispositivo es capaz de verificar. Además, una vez el proceso se haya completado, mostrará un mensaje indicando si el chip es defectuoso.

Combina simplicidad de programación (diversas librerías C disponibles) con un precio reducido y se controla fácilmente desde la placa Arduino mediante un bus de 6 líneas. Se incluye también un potenciómetro de ajuste de la intensidad luminosa.

Figura 2

Para desarrollar el software necesario para el control del display LCD se ha utilizado la librería *LiquidCrystal.h* [LCD], disponible en la página web de Arduino y ampliamente referenciada en la red. Contiene funciones para borrar la pantalla, configurar el cursor y representar textos y números con suma facilidad.

Como dispositivos de representación adicional se utilizan dos leds, verde y rojo, para indicar si el chip funciona correctamente o es defectuoso.

Se incorpora también un zumbador (buzzer) que avisará acústicamente del resultado del proceso de verificación: dos señales sonoras en caso de chip defectuoso y una para correcto.

Selector/pulsador rotatorio ("Rotary Encoder")

Para seleccionar el chip a verificar se utiliza un selector rotatorio que permitirá al usuario desplazarse por el listado que contiene la referencia y descripción de los chips y, accionando el pulsador incorporado, iniciar la rutina de verificación. El modelo elegido es el siguiente, con 18 pasos por vuelta, cuyo funcionamiento se explica en secciones posteriores:

Figura 3 [Alp15]

Alimentación

La etapa de alimentación está compuesta por dos reguladores, de 9V y 5V (7809 y 7805), ampliamente usados en dispositivos electrónicos. El 7809 se alimentará con una entrada exterior de 12V y su salida (+9V) se conectará a la entrada del 7805. De esta forma tendremos disponibles tensiones de 9V y 5V con un mínimo factor de rizado.

Detección de consumos

Para garantizar la seguridad del sistema se ha diseñado y construido un sistema de detección de consumo de alimentación cuyo funcionamiento se explica más adelante.

Reset

Se ha incluido un pulsador en la parte trasera del dispositivo, conectado al Reset de la placa Arduino, que servirá para inicializar el sistema.

Placa de circuito impreso

El cableado requerido para interconectar todos los dispositivos del sistema es impracticable y se hace necesaria por tanto la realización de una placa de circuito impreso para montaje superficial (SMD) específica para este proyecto.

Con el fin de conseguir un montaje lo más compacto posible el diseño se ha realizado de modo que la placa Arduino se inserta directamente en un zócalo que reproduce exactamente su arquitectura de pines, sin ningún cable de conexión.

Dicha placa incluye, asimismo, los terminales de entrada y salida necesarios para la interconexión con el resto de dispositivos del sistema.

Para su diseño y fabricación se ha contado con la colaboración del Servicio de Instrumentación Electrónica de la Universidad de Zaragoza, que dispone de los medios técnicos necesarios, en este caso la máquina *LPKF 93s*, grabando una placa de fibra de vidrio "single layer" metalizada en cobre. El fabricante de la misma es *Circuitos JCF*.

El esquema general es el siguiente, donde se detalla la localización de cada uno de los bloques funcionales del sistema:

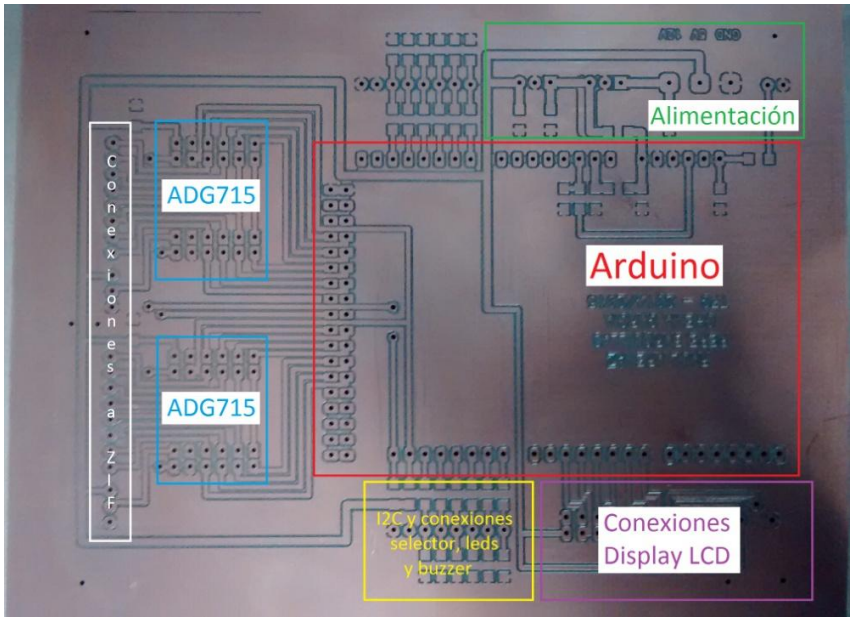

Figura 4

Prototipo final

Para el montaje definitivo y compacto de todos los componentes se ha utilizado una caja comercial adecuada a las características del sistema (figura 5), provista de un panel frontal para colocar el display LCD y los LEDs adicionales, así como el zócalo ZIF y el selector de chips.

En la parte posterior se sitúa la entrada de tensión de alimentación (12V) y el conector USB necesario para posteriores reconfiguraciones del sistema.

Así, el trabajo final adquiere el siguiente aspecto:

Figura 5

El diagrama de bloques del sistema completo se muestra a continuación:

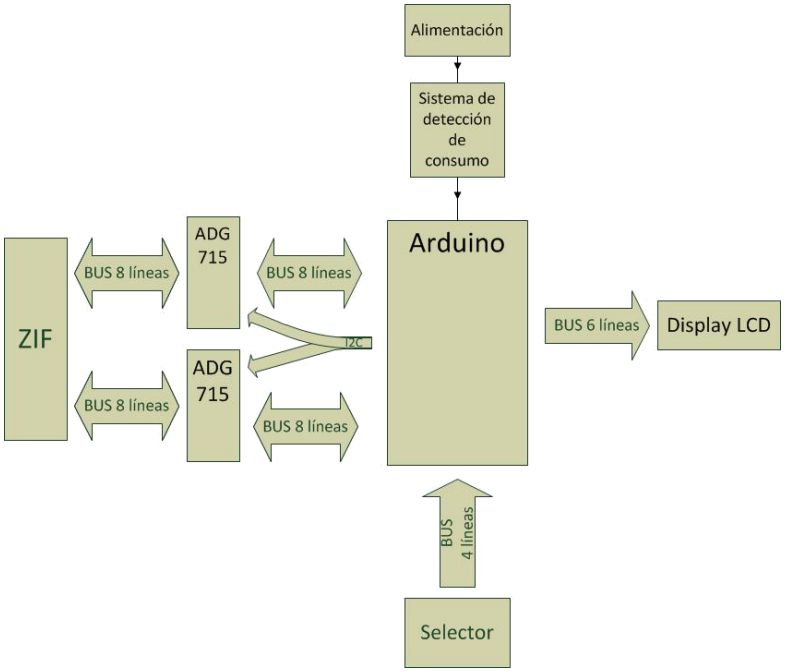

Figura 6

3.2 Alimentación y detección de consumos

Como ya se comentó con anterioridad se usan dos reguladores integrados para generar las tensiones de alimentación necesarias para el sistema.

El regulador de 9V se conecta a la entrada de alimentación externa de la placa Arduino (7-12V), de forma que midiendo su consumo de corriente se puedan detectar valores demasiado elevados que indican posibles cortocircuitos en el chip bajo verificación.

El regulador auxiliar de 5V sirve para alimentar los componentes externos: display LCD e integrados ADG715, evitando así que la placa Arduino tenga que suministrar la corriente que éstos consumen (>200mA).

3.2.1 Detección de consumos

Dado que nuestro interés se centra en la tecnología CMOS, que presenta un consumo estático de corriente prácticamente nulo, supondremos que si el chip a verificar consume corriente, será defectuoso.

Así, de modo previo a la verificación funcional de la lógica que contiene el chip bajo test, es conveniente realizar una medida de los consumos de corriente, para detectar posibles cortocircuitos y evitar daños al propio sistema de verificación. Esta tarea la llevaremos a cabo mediante el conversor analógico digital de 10 bits (ADC) que incorpora el ATMega2560 [ATM06].

Ante la dificultad de medir individualmente las corrientes en los distintos terminales del chip, se ha optado por determinar la intensidad total que consume el sistema y asignarle un valor umbral que califica el consumo como excesivo, y por tanto, al chip, como defectuoso.

Aunque se han considerado otras opciones, finalmente se ha fijado un valor de 10 mA, teniendo en cuenta que los dispositivos secuenciales pueden presentar un cierto consumo dinámico (<2mA), siempre inferior a este valor.

La placa Arduino Mega incluye un regulador de tensión, con rango de entrada de 7 a 12V, para obtener la tensión de alimentación del microcontrolador.

Se han realizado una serie de medidas del consumo del regulador en función del voltaje, V_{in}, con el que se alimenta:

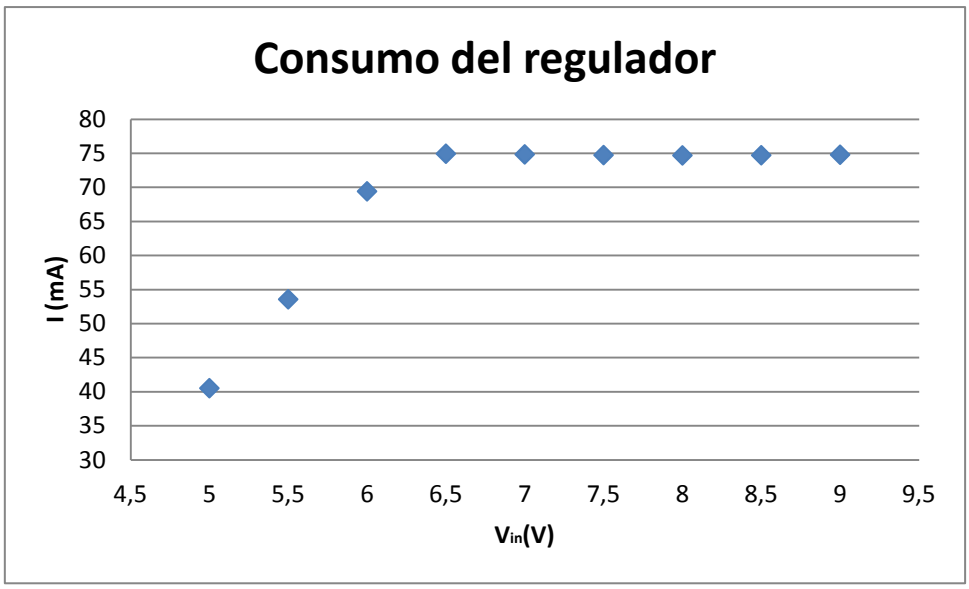

Figura 7

Se puede apreciar cómo, para valores bajos de V_{in}, el consumo es proporcional al voltaje. En el momento en que éste toma un valor de 6.5V, el consumo permanece constante en torno a 75 mA (con pequeñas variaciones), independientemente de V_{in}. Como se ha comentado, la tensión de alimentación debe estar comprendida entre 7 y 12 voltios. Es aceptable considerar por tanto que el consumo del regulador es de 75 mA cuando trabaja en condiciones de alimentación óptimas. A este valor hay que añadir en cada momento la suma de las intensidades de los terminales configurados como salida, según su estado lógico, que sería el hipotético consumo del chip ($I_{salidas}$).

Una resistencia R en serie con la entrada del regulador, como se muestra en la figura 8, producirá una caída de potencial, a partir de la cual podremos determinar la intensidad que consume la placa.

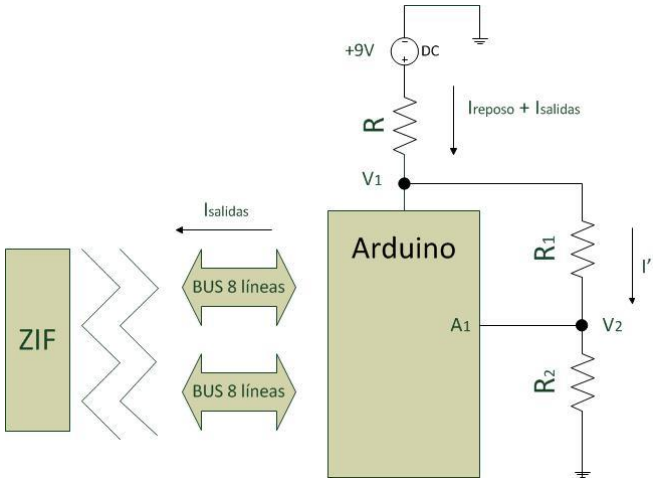

Figura 8

La tensión V_1 es:

$$V_1 = 9V - \left(I_{reposo} - I_{salidas}\right)R$$

Para una resistencia de 15Ω resulta (sustituyendo I_{reposo} por 75 mA):

$$V_1 = 7.88V - 15\Omega I_{salidas}$$

Este valor V_1 debe ser superior a 7V para el correcto funcionamiento del regulador.

Además, V_2 ha de ser inferior a 5V, límite superior del rango dinámico del ADC que utilizamos para la medida, con una resolución de 5V/1024 ≅ 5mV.

Se requiere por tanto un divisor de tensión de factor de atenuación 5/7.88, de forma que en estado de reposo, la tensión V_2 aplicada a la entrada del conversor sea 5V, lo que se consigue con R_1=33kΩ y R_2=56kΩ.

Conviene observar la baja precisión de las medidas, ya que sólo se va a utilizar una pequeña parte del rango dinámico de entrada del ADC.

Como ejemplo ilustrativo consideramos que el chip consume 50 mA, $I_{salidas}$=50 mA. Este es un valor muy elevado que se daría bajo condiciones muy raras y lo tomaremos como intensidad máxima.

En este caso, V_2 resulta ser 4,52V, con un ΔV de tan sólo 0.48V.

Por tanto:

$$ADC = \frac{1024(5V - 4.52)}{5V} \cong 98$$

La resolución resulta ser algo menos de 7 bits (2^7=128) de modo que la precisión de la medida resulta bastante reducida.

Con estos parámetros como punto de partida, se han realizado una serie de medidas de la intensidad para distintos valores de R y del factor de amortiguación del divisor de tensión y los resultados no son satisfactorios.

Estudiando a fondo las características del ADC que se exponen en el datasheet del ATMega2560, vemos que se pueden optimizar sus prestaciones para nuestras necesidades concretas [ATM10].

Es posible utilizarlo en modo diferencial, así como modificar la tensión de referencia y la ganancia del amplificador incorporado para adaptar el rango dinámico de entrada al intervalo de posibles valores de la tensión V_2 y por tanto mejorar la precisión.

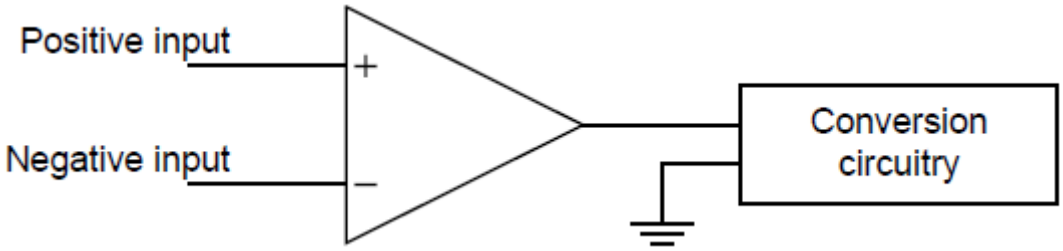

Figura 9

Como no es posible utilizar para este fin las funciones incluidas en el entorno de programación de Arduino, es necesario configurar directamente los diversos registros del microcontrolador que controlan la operación del ADC.

Esto nos obliga a un estudio minucioso de dichos registros y a una cuidada selección de los valores asignados a cada bit de control.

ADCSRA: un 1 en su bit 7 activa el conversor y un 1 en el bit 6 inicia un nuevo proceso de conversión, mientras que los tres bits menos significativos (0, 1, 2) permiten seleccionar la frecuencia de reloj que utiliza.

ADMUX: es el principal registro de control del conversor. Configura el modo de operación diferencial y selecciona (junto con el registro auxiliar ADCSRB) las entradas analógicas a comparar. Permite, además, modificar la ganancia y la tensión de referencia V_{REF} para el proceso de conversión, siendo en nuestro caso 1.1V el valor seleccionado.

En estas condiciones el rango de entrada es de $-V_{REF}$ a $+V_{REF}$ y la resolución es de:

$$\frac{2{,}2V}{1024} \cong 2mV$$

El resultado de la conversión viene dado por:

$$ADC = \frac{512(V_{POS} - V_{NEG})}{V_{REF}}$$

que se almacena en forma de complemento a 2 en los registros de datos ADCH y ADCL.

Seleccionamos las entradas analógicas A2 para V_{POS} (5V) y A1 para V_{NEG}, de acuerdo con el siguiente esquema:

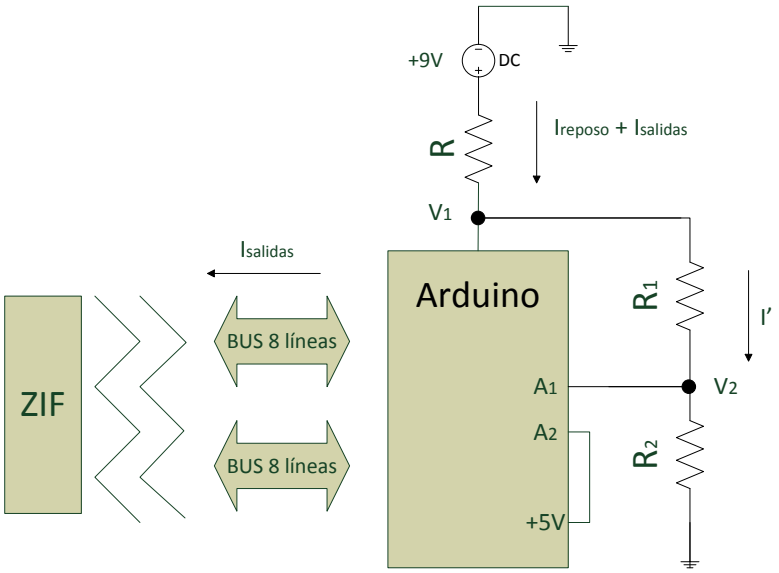

Figura 10

y obtenemos el siguiente valor para el resultado de la conversión:

$$ADC = \frac{512(5V - V_2)}{1.1V}$$

Si ahora volvemos al ejemplo anterior, con una intensidad de 50 mA la tensión V_2 baja a 4.52V y resulta una resolución de:

$$ADC = \frac{512(5V - 4.52V)}{1.1V} \cong 224$$

La resolución ahora es cercana a los 8 bits, más del doble de la que teníamos antes, suficiente ya para nuestro propósito.

Hay que tener en cuenta que aunque inicialmente hemos supuesto constante el consumo de la placa Arduino (75mA), éste puede variar con la temperatura e incluso con la placa concreta utilizada.

Con el fin de minimizar la influencia de esta posible variación, hemos optado por medirlo en vacío (sin ningún chip conectado en el zócalo) cada vez que queremos examinar un chip y almacenarlo como referencia en una variable, de modo que posibles cambios en su valor no afecten al proceso de verificación.

Así, el sistema de detección de consumo determinará la diferencia entre el valor en vacío y en el momento de conectar el chip, de forma que si es superior al nivel umbral consideraremos el chip como defectuoso.

3.3 Red de interruptores: arquitectura y control

Los interruptores integrados ADG715 son los encargados de activar o interrumpir, bajo control del software desarrollado al efecto, las conexiones entre el chip bajo test y el microcontrolador (figura 11).

Están fabricados en tecnología CMOS e incluyen 8 interruptores bidireccionales, con una resistencia de tan solo 2.5Ω. El control de los mismos se realiza mediante el protocolo de comunicación I2C (Inter-Integrated Circuit), compatible con el ATMega2560.

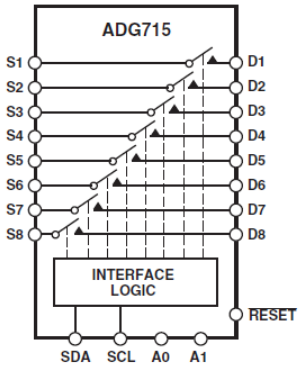

Figura 11 [ANA13]

Los terminales S (1 a 8) se conectan al zócalo ZIF que contiene los chips a verificar, mientras que los terminales D (1 a 8), se conectarán a los puertos del µC.

Las líneas SDA y SCL (bus I2C) permiten controlar el estado de los interruptores en cada momento, siendo A0 y A1 las dos entradas que determinan la dirección del dispositivo en el bus. Además, el RESET, activo en bajo, cortará las 8 líneas de comunicación abriendo todos los interruptores.

I2C es un bus de comunicación muy generalizado para comunicar un microcontrolador con circuitos integrados auxiliares. Se denominará master/maestro al dispositivo que controla el bus en cada momento y slave/esclavo a los dispositivos controlados. Es además multi-maestro, permitiendo así que varios dispositivos puedan actuar como maestros, lógicamente en momentos distintos [ELC].

Consta de dos líneas: SDA, que transmite los datos, y SCL, que constituye el reloj y establece el sincronismo entre dispositivos, de forma que con cada pulso de SCL se transmite un bit de SDA.

Aunque para el sistema desarrollado se precisan sólo dos chips ADG715, dado que existen dos pines de dirección, A0 y A1, es posible incluir hasta cuatro, lo que permite ampliar el sistema si fuera necesario.

De acuerdo con el protocolo I2C, para comenzar una trama general de comunicación el master enviará por la línea SDA un bit de start, que “alertará” a los esclavos poniéndolos a la espera de una transacción, seguido de un byte de dirección (7 bits) que seleccionará el esclavo

concreto con el que quiere comunicar, y finalmente un último bit que indica si la acción a realizar es de lectura o escritura.

Por su parte, el esclavo responde con un nuevo bit de reconocimiento, que confirma que el byte enviado previamente ha llegado a su destino, y consiste en una transición de bajo a alto (ACK).

Se envían ahora todos los bytes de datos a transmitir y finalmente, un bit de stop (transición de bajo a alto) cierra la transmisión y deja el bus libre.

Para desarrollar el software necesario para el control de los chips de interruptores usados en este proyecto se ha utilizado la librería *Wire.h*, disponible en la página web de Arduino y ampliamente referenciada en la red [WIR]

Los chips ADG715 tienen prefijados de fábrica cinco de los siete bits de dirección, de forma que con los dos bits restantes, A1 y A0, se podrán configurar cuatro direcciones físicas distintas (00, 01, 10 ó 11)

En estos dispositivos los bytes de datos se utilizan para establecer el estado de los interruptores del chip seleccionado. Así, por ejemplo, el byte de datos 10101010 cierra los interruptores 2, 4, 6 y 8 (1) y abre los 1, 3, 5 y 7 (0).

Como ilustración de lo explicado anteriormente se muestra un código que abre los interruptores impares y cierra los pares:

```
Wire.beginTransmission(0b1001000);  //byte de dirección. Dos últimos bits son A1 y A0 (00)

Wire.write(0b10101010); // byte de datos. Abre/cierra los interruptores impares/pares

Wire.endTransmission(); //Acaba la transmisión, dejando las líneas de comunicación libres
```

Se representan asimismo las señales eléctricas que se envían por las líneas SDA y SCL, pudiéndose observar el byte de dirección y un byte de datos.

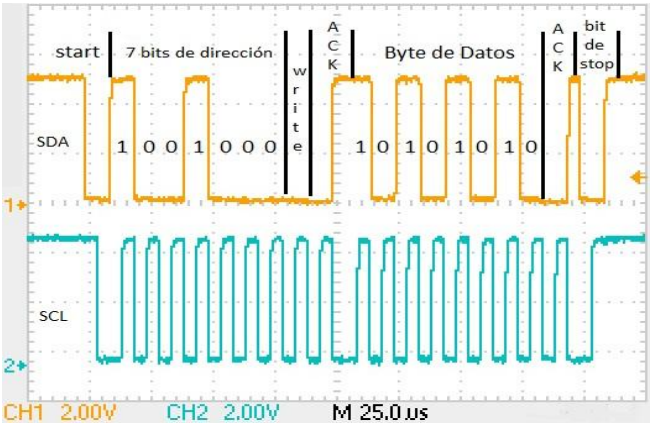

Figura 12

3.4 Dispositivos de entrada/salida

Como ya se ha comentado anteriormente, para que el usuario pueda interactuar con el sistema de verificación de chips se requieren los adecuados dispositivos de entrada/salida: display LCD, LED's de visualización, un buzzer y un selector/pulsador rotatorio.

Por su interés, se expone en primer lugar el funcionamiento del selector/pulsador, que posee cinco terminales de conexión

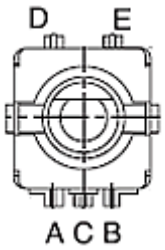

Figura 13 [Alp15]

Los terminales A, B, C corresponden al selector rotatorio y los D, E al pulsador mecánico que lleva incorporado el dispositivo. Los terminales C y D se conectan a tierra, mientras que los A, B y E lo están a entradas de Arduino.

Si conectamos el terminal E a una tensión de alimentación mediante una resistencia de "pull-up" cada vez que se accione el pulsador la tensión del terminal E bajará a 0V y el pulso producido será reconocido por el μC, que iniciará el proceso de verificación.

El selector consta de dos fases A y B, que es necesario conectar a una tensión de alimentación mediante resistencias de "pull-up", como se ve en el siguiente esquema:

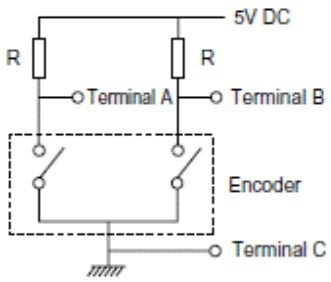

Figura 14 [Alp15]

Con el fin de evitar el efecto de los rebotes asociados a los elementos mecánicos de conmutación, que producen múltiples contactos de muy corta duración, se han incluido redes RC en todos los terminales, con R=1KΩ y C=10 nF.

Se ha comprobado experimentalmente que basta incorporar estos circuitos para evitar el efecto de los rebotes, no siendo necesario así incluir rutinas de corrección en el software desarrollado.

Cada una de las fases se puede representar por un interruptor ON/OFF que puede dejar los terminales a 5V (abierto) o conectarlos a tierra (cerrado). Cuando el selector se gira a

la derecha el sistema mecánico hará que el interruptor A conmute antes que el B, cambiando así los estados de ambos interruptores (de 0 a 5V o viceversa) con un cierto retardo, τ, entre ellos. Cuando giremos en el otro sentido, será el interruptor B el que conmute antes [Alp15].

Se han medido en el laboratorio las dos fases o terminales, A y B, con giro hacia la derecha (figura 15) y hacia la izquierda (figura 16):

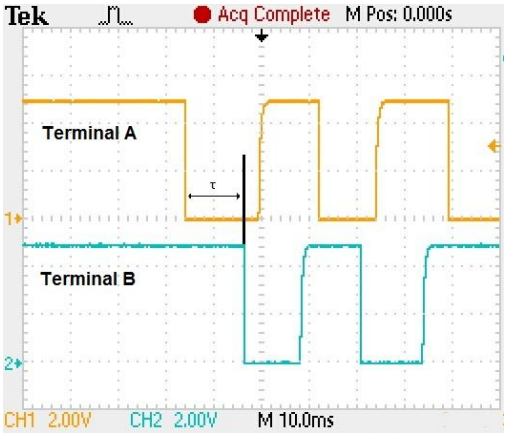

Figura 15

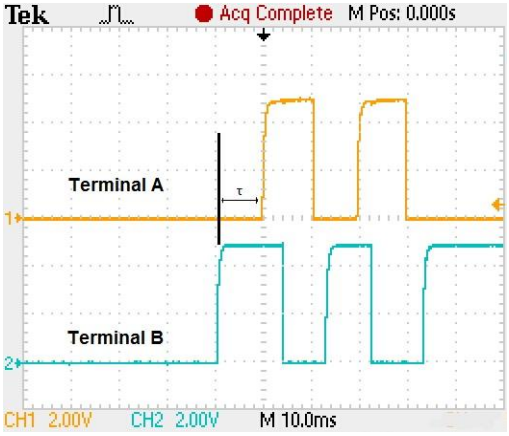

Figura 16

En la primera figura se observa que el terminal A cambia de estado y un tiempo τ después lo hace el B. En la segunda se produce la situación contraria, el primero en cambiar de estado es B y un tiempo τ después lo hace A. En ambas imágenes pueden observarse varios pasos del selector para mayor claridad.

En definitiva, el selector de posición nos proporciona dos señales digitales, desfasadas entre ellas, que pueden ser procesadas por el μC. De esta forma, podremos conocer el sentido de giro y desplazarnos por el listado de chips disponibles de acuerdo con la elección del usuario.

3.5 Software

En esta sección se explica el software desarrollado para el control y operación del sistema de verificación propuesto. Como ya se ha apuntado previamente, la programación se lleva a cabo en lenguaje C, y dentro del entorno operativo Arduino.

En este ámbito la estructura de cualquier programa consta de dos funciones o procedimientos básicos: *setup()* y *loop()*

setup()

Encabeza cualquier programa y contiene líneas de código como configuraciones de registros internos, declaración de variables, configuración de entradas/salidas, en definitiva sentencias que sólo se ejecutan una vez. Cada vez que Arduino comience a funcionar, ejecutará esta función e inmediatamente pasará al *loop().* Para volver al *setup()* hay que resetear el µC.

***loop()*:**

Es un bucle que se repite indefinidamente mientras el µC esté conectado a la alimentación, ejecutando así de modo continuo las instrucciones contenidas en la función.

A continuación se muestra la arquitectura de un programa básico de Arduino con las funciones *setup()* y *loop()* vacías:

```
void setup() {
  // El código aquí insertado se ejecutará una vez

}

void loop() {
  // El código aquí insertado se ejecutará repetidamente

}
```

Figura 17

Lógicamente este programa no realiza ninguna acción, siendo preciso introducir las correspondientes instrucciones declarativas o ejecutivas en ambas funciones para que realice una función concreta.

Se van a explicar a continuación detalladamente las principales funciones que componen el programa desarrollado. Unas están asociadas a interrupciones y otras son llamadas directamente en líneas de código.

En primer lugar se explican las funciones asociadas a la selección del chip a verificar y a la orden de comienzo del proceso, (ambas asignadas a interrupciones) para describir después la función de verificación de chips, que realiza todas las operaciones funcionales para determinar su estado.

El código completo del software desarrollado para el prototipo construido para este proyecto se adjunta en el anexo I, donde se han incluido también los comentarios adecuados para ayudar a comprenderlo.

3.5.1 Selección del chip y comienzo de verificación

En la zona de declaración de variables se han incluido, en forma de matriz, las referencias y descripciones de los circuitos integrados digitales que el prototipo construido permite verificar.

Un simple índice identifica unívocamente cada dispositivo, de forma que resulta muy sencillo ampliar dicha lista sin más que introducir nuevos componentes.

Además, un segundo índice auxiliar realiza un direccionamiento indirecto de las matrices, lo que permite que las referencias de los chips aparezcan siempre ordenadas alfabéticamente en la pantalla, independientemente del orden en que han sido introducidas.

Para agilizar la respuesta y facilitar la programación se ha decidido el uso de interrupciones como método para la interacción del selector/pulsador con el μC.

Una interrupción permite detectar un evento asíncrono independientemente de las líneas de código que se estén ejecutando en ese momento y responder en consecuencia. Así, al ocurrir el evento, se ejecutará la sección de código incluida en su función correspondiente, asignada con anterioridad.

El Arduino Mega posee seis terminales que se pueden usar como entrada de interrupciones. Las declaraciones de las interrupciones utilizadas deberán incluirse en la función *setup()*, especificando siempre la interrupción seleccionada, la función que se ejecutará cuando actúe y el evento asíncrono que deberá producirse para que se ejecute.

A continuación se muestra un ejemplo de declaración de una interrupción:

```
void setup() {

attachInterrupt(4, SelPosicion, CHANGE);  //Interrupción 4 (pin 19)
                                          //Función que actuará: SelPosicion
                                          //Evento asíncrono: cambio

}

void loop() {

}
```

Figura 18

En nuestro caso se hace necesario el uso de dos interrupciones distintas: una para el selector de posición rotatorio y otra para el pulsador incorporado.

De acuerdo con el funcionamiento del selector expuesto en la sección 3.4, se requiere una interrupción para la fase B y una entrada digital para la fase A [HOB]. La interrupción se debe activar con cada cambio de la fase B y "muestreará" el valor de ambas fases en ese momento.

Cuando el mando del selector rotatorio gira a la izquierda, la fase B cambia de estado y se activa la interrupción, siendo en este momento A y B distintos, mientras que si el giro es a la derecha, la interrupción vuelve a actuar, pero ahora A y B son iguales (figuras 15 y 16).

Por tanto, la función *SelPosicion(),* asignada a la interrupción, debe leer los bits de las fases A y B, que constituyen el estado del selector en ese momento (00, 01, 10 ó 11). Si el estado es 0 ó 3 (los dos bits iguales) el sentido de giro es hacia la derecha, mientras que si es 1 ó 2 (los dos bits distintos) se estará girando hacia la izquierda.

La estructura de la función *SelPosición()* es la siguiente:

```
void SelPosicion() {

 byte estado;

 int MSB = digitalRead(pinEncoderA); //Considera la fase A como el bit más significativo
 int LSB = digitalRead(pinEncoderB); //Considera la fase B como el bit menos significativo

 estado = 2*MSB+LSB;   //Convierte el número binario en su correspondiente en baseo 10

 if(estado==0 || estado==3) {if (indice<numChips) {indice++;}} //Si es 0 ó 3, suma uno a indice
 if(estado==1 || estado==2) {if(indice>1){indice--;}}          //Si es 1 ó 2, resta uno a indice

}
```

Figura 19

Teniendo en cuenta que el pulsador de selección de chip se cierra al activarlo y por tanto conecta a tierra su terminal E, el evento que activa su interrupción será un cambio de alto a bajo (FALLING), mientras que la función correspondiente *StartTest()* será la asignación a la variable *indiceTest* del valor de la variable *índice*, que indica el chip actualmente seleccionado.

3.5.2 Verificación de un chip determinado: *VerificaChip()*

La idea ha sido incluir en una única función *VerificaChip()* todo el proceso de verificación de los chips, pasándole como argumento el *índice* que lo identifica. Se parte de una lista de circuitos integrados disponibles en la que cada uno tiene asignado un número único.

Mediante la estructura lógica *switch case* se seleccionaran las alternativas de comprobación para cada tipo y referencia de chip.

```
switch (var) {
    case 1:
      //realiza esta acción si var es igual a 1
      break;
    case 2:
      //realiza esta acción si var es igual a 2
      break;
  }
```

Figura 20

La estructura *switch case* toma como referencia la variable *var* y realiza una u otra acción dependiendo de su valor. En nuestro caso, dependiendo del valor de *índice*, se realizará una u otra rutina de verificación.

La verificación del correcto funcionamiento del chip se lleva a cabo en dos fases. En la primera se mide el consumo, si éste es elevado el chip se considera defectuoso y se descarta. Por el contrario, si el consumo no es elevado, se pasará a la verificación funcional de la lógica intrínseca de cada chip.

En primer lugar hay que configurar los bits de los puertos del μC como entradas o salidas, dependiendo del carácter de entrada o salida de los terminales del chip a comprobar.

Para detectar posibles cortocircuitos se mide el consumo del regulador en vacío y se almacena este valor en una variable. Después se realizan dos medidas de consumo más: una con las entradas en valor lógico 1, para detectar posibles cortocircuitos a tierra, y otra con las entradas en valor 0, para detectar posibles cortocircuitos a V_{cc}. Se calculan las diferencias con el consumo en vacío, de forma que si alguna de ellas es mayor que un valor umbral, el consumo se considerará excesivo y se interrumpe el proceso de verificación aplicando un Reset a los interruptores.

Se consideran las diferencias de consumos en lugar de los valores absolutos con el fin eliminar las posibles variaciones de consumo del regulador.

La función que realiza esta tarea es la siguiente:

```
byte consumo (byte puertoC1, byte puertoL1, byte puertoC0, byte puertoL0)
{
    byte Iperm=0;
    digitalWrite(39, 0); //Corte de comunicación entre Arduino y zócalo
    delay(1);
    intensidad[0]=MedirConsumo(); //Medida de consumo en vacío
    cerrar(); delay(1); //conexión entre Arduino y zócalo
    PORTC=puertoC1;   PORTL=puertoL1; delay(10); //Primera configuración de los puertos: salidas en 1
    intensidad[1]=MedirConsumo(); //Medida de consumo
    PORTC=puertoC0;   PORTL=puertoL0; delay(10); //Segunda configuración de los puertos: salidas en 0
    intensidad[2]=MedirConsumo(); //Medida de consumo
    if(intensidad[1]-intensidad[0]>IMax || intensidad[2]-intensidad[0]>IMax){Iperm=1;}//Detección de cortocircuitos
    return Iperm;
}
```

Figura 21

La verificación lógica o funcional consiste en aplicar, al circuito integrado bajo test, todas las posibles combinaciones de vectores entrada y leer sus vectores de salida correspondientes, para comprobar que los valores de todos ellos son correctos.

Antes de entrar en más detalles de este proceso conviene hacer una distinción previa entre tipos de chips, de acuerdo con su funcionamiento:

Combinacionales

Puertas lógicas, decodificadores, comparadores, sumadores, multiplexores. En todos ellos los vectores salida son función exclusivamente de los vectores entrada y no incluyen reloj en su funcionamiento.

Se utilizan dos métodos distintos y complementarios para realizar el proceso de verificación:

Verificación mediante vectores de test [Cia87]

Con este procedimiento se comprobarán los chips de puertas lógicas, el decodificador BCD y el multiplexor, debido a su simplicidad. Se declara al inicio del programa una matriz que contiene los vectores test para las entradas del chip y otra con los vectores respuesta esperados para las salidas, de acuerdo a la lógica interna de cada uno

Se aplican a cada circuito integrado específico sus vectores de test correspondientes y se leen las respuestas que generan. Si son distintas de los vectores declarados como respuestas correctas, se entiende que la lógica del chip no funciona adecuadamente y será considerado defectuoso. Este método permite, además, verificar al mismo tiempo todas las puertas lógicas de un chip [Bru06].

Constituye, por su sencillez, una gran simplificación de los métodos basados en vectores de test que utilizan las empresas fabricantes de circuitos integrados.

Verificación individual

Se utiliza para los chips en los que sería muy complejo realizar una comprobación mediante vectores test debido a la gran cantidad de combinaciones posibles para sus entradas, como por ejemplo en el comparador de magnitud de 4 bits. En estos casos se lleva a cabo una verificación individualizada implementando la tabla de funcionamiento que aporta el fabricante.

Secuenciales

Son aquellos que para su correcto funcionamiento requieren la presencia de una señal de reloj, pudiendo ser activo con flancos de subida o de bajada. Por lo tanto, para su verificación se debe simular la presencia de un reloj, para lo cual se han construido dos funciones que provocan un paso de alto a bajo o de bajo a alto por el pin de Arduino que se requiera.

Las funciones, que emulan los flancos de subida o bajada de un reloj, son las siguientes:

```
void subida (byte Psub)
{
  digitalWrite(Psub, 0);//Inicialmente en 0.
  delay(1);
  digitalWrite(Psub, 1);//Subida del reloj.
  delay(1); //Espera de un milisegundo para respetar el tiempo de respuesta.
  digitalWrite(Psub, 0); //Vuelta a 0.
}
void bajada (byte Pbaj)
{
  digitalWrite(Pbaj, 1);//Inicialmente en 1.
  delay(1);
  digitalWrite(Pbaj, 0);//Bajada del reloj.
  delay(1); //Espera de un milisegundo para respetar el tiempo de respuesta.
  digitalWrite(Pbaj, 1); //Vuelta a 1.
}
```

Figura 22

En este caso el proceso de verificación será individualizado, implementando, fila a fila, la tabla de funcionamiento que proporciona el fabricante.

Se establece en primer lugar el estado inicial del chip mediante la oportuna configuración de las entradas, se aplica una transición activa de reloj y se lee el nuevo estado para determinar si es el correcto. Basta repetir esta tarea sucesivamente para ir recorriendo y comprobando la tabla de transición de estados del dispositivo.

Resulta muy conveniente para la fiabilidad del proceso introducir pequeños retardos antes y después de las transiciones activas de reloj, para eliminar efectos no deseados asociados al efecto de los tiempos $t_{set\text{-}up}$ y t_{hold}, de acuerdo con las especificaciones indicadas en el datasheet de los dispositivos.

3.5.3 Programa principal: *loop()*

Como es habitual en la programación de μC´s el código introducido en la función *loop()* representa una mínima parte del software desarrollado y se limitará, en nuestro caso, a representar en el display la referencia y descripción del chip actualmente seleccionado, sólo si se ha producido una modificación en el mismo por la acción del selector rotatorio. De esta forma se evitan parpadeos molestos en la pantalla, consecuencia de borrar y escribir continuamente los mismos datos.

Son las funciones asociadas a las interrupciones las que realizan, bajo control de los dispositivos de entrada que las activan, la mayor parte del trabajo operacional del sistema, fijando el chip bajo test e iniciando el correspondiente proceso de verificación.

Se puede decir que el programa principal, función *loop(),* se limita a esperar que el usuario active alguno estos controles, y a llamar a la respectiva función respuesta.

Así, si se gira el selector a derecha o izquierda, se activará su respectiva interrupción que modificará la variable global *índice* que identifica a cada chip y por tanto se actualizará de modo inmediato en el display la referencia y descripción del chip.

Actualmente, la lista de chips disponibles es la siguiente:

74HC00 – 4 puertas lógicas NAND de 2 entradas

74HC02 – 4 puertas lógicas NOR de 2 entradas

74HC04 – 4069UB– 6 inversores

74HC08 – 4 puertas lógicas AND de 2 entradas

74HC10 – 3 puertas lógicas NAND de 3 entradas

74HC11 – 3 puertas lógicas AND de 3 entradas

74HC27 – 3 puertas lógicas NOR de 3 entradas

74HC32 – 4 puertas lógicas OR de 2 entradas

74HC4511 – Decodificador BCD de 7 segmentos

74HC74 – 2 flip-flops D

74HC85 – Comparador de magnitud de 4 bits

74HC86 – 4 puertas lógicas XOR de 2 entradas

74HC107 – 2 flip-flop JK

74HC151 – Multiplexor de 8 entradas

74HC163 – Contador binario de 4 bits

74HC194 – Registro bidireccional de 4 bits

No obstante, en la lista de las referencias de los chips se han incluido algunos más aunque todavía no están disponibles para su verificación.

En resumen, para verificar el chip mostrado en la pantalla, habrá que presionar el pulsador, de modo que su interrupción se activará y asignará a la variable *indiceTest* el valor *indice* actual, lo que provocará a su vez la ejecución de la función *VerificaChip()* y por tanto su comprobación funcional.

```
void loop()
{
  if(indice!=indiceOld) {      //Comprueba si se ha girado el selector
  lcd.setCursor(0,0); lcd.print(ref[indice]);  //Si es así, se escribe en el diplay la nueva descripción
  lcd.setCursor(0,1); lcd.print(descrip[indice]);
  lcd.setCursor(0,2);
  indiceOld=indice;
  }

  if (indiceTest>0) {         //Comprueba si se ha pulsado el interruptor
    VerificaChip(indiceVerif[indiceTest]);   //Se pasa a la función de verificación del chip
    }
 delay(200);
}
```

Figura 23

4. Resultados

El resultado final ha sido el diseño y construcción de un prototipo capaz de detectar chips defectuosos, con la capacidad de ser reprogramado y ampliado fácilmente a posibles nuevos dispositivos.

Se ha realizado un detallado estudio experimental de una serie de circuitos integrados digitales defectuosos con el fin de determinar características que los identifiquen como tales, siendo un consumo elevado una de ellas. Obviamente, un funcionamiento lógico incorrecto es otra.

Para el diseño del prototipo se ha utilizado una arquitectura funcional basada en un microcontrolador, con los dispositivos periféricos necesarios para la interacción con el usuario, buscando siempre un compromiso razonable entre costo y prestaciones.

De acuerdo con lo anterior se ha seleccionado la placa Arduino que incorpora el microcontrolador ATMega 2560, con una amplia gama de dispositivos de entrada/salida compatibles.

Se ha desarrollado y comprobado el software necesario para detectar las posibles características de error de los chips defectuosos: consumo excesivo de corriente o funcionamiento lógico incorrecto.

Dada la complejidad del cableado necesario para la interconexión de los diversos bloques del prototipo, se ha diseñado y construido una placa de circuito impreso específica que simplifica considerablemente el montaje físico.

Ha sido necesario adaptar los conocimientos generales de programación C++ a los requerimientos de un microcontrolador, con recursos de memoria RAM y de programa limitados.

Con la labor desarrollada en el laboratorio se han adquirido competencias en aspectos prácticos experimentales y en la construcción de interfaces electrónicos auxiliares.

5. Conclusiones

A la vista del trabajo terminado, se puede concluir que su realización ha sido muy interesante tanto desde el punto de vista de ampliación de conocimientos como por su disponibilidad futura dentro del laboratorio de electrónica.

Dado que en el montaje físico del prototipo han surgido problemas de hardware no previstos, a la hora de abordar la construcción de un sistema hay que tener en cuenta aspectos particulares, no tenidos en cuenta tal vez en el diseño teórico pero que luego resultan decisivos para su funcionamiento (cableado, interferencias, rebotes mecánicos, etc).

En cuanto a la revisión bibliográfica, se ha comprobado que actualmente resulta fundamental consultar la información en la red. En este ámbito las referencias clásicas en libros es escasa y desfasada, debido a la actualización y desarrollo constante en este campo.

Aunque se han considerado diversas alternativas, algunas muy ambiciosas, que incluirían incluso la detección automática del chip [Bru06], se ha optado por escoger la expuesta en esta memoria teniendo en cuenta el tiempo y recursos disponibles.

De acuerdo con lo anterior, no ha sido nuestra intención construir un prototipo con un enfoque comercial, por lo que tal vez se ha perdido flexibilidad y modularidad del dispositivo en algunos aspectos.

6. Bibliografía

[Alp15] ALPS, "ALPS Manufacturer of Electronic Components/Parts Catalog", 2015. (Anexo II)

[ANA13] Analog Devices, "Serially Controlled, OCTAL SPST Switches", *Datasheet ADG715,* 2013. (Anexo II).

[ATM10] Atmel, "25. ADC – Analog to Digital Converter", *Datasheet ATMega2560*, 2010. (Anexo II).

[ATM06] Atmel, "Characterization and calibration of the ADC on an AVR", *Application Note,* 2006. http://www.atmel.com/images/doc2559.pdf (Anexo II).

[Bru06] M. Brutscheck, M. Franke, A. Th. Schwarzbacher, St. Becker, "Determination of Pin Types and Minimisation of Test Vectors in Unknown CMOS Integrated Circuits", *School of Electronic and Communications Engineering, Dublin Institute of Technology, Ireland and Department of Computer Science and Communications Systems, University of Applied Sciences Merseburg, Germany,* 2006.

[Cia87] Steven A. Ciarcia, "Ciarcia´s Circuit Cellar", *McGraw-Hill Publishing Company,* 1987.

[ELC] http://www.electroensaimada.com/i2c.html. "Tutorial Arduino: I2C".

[Far02] Joe Farr, "Digital I.C. tester", *Everyday Practical Electronics, October 2002.*

[HOB] "Tutorial 6 – Rotary Encoder", http://www.hobbytronics.co.uk/arduino-tutorial6-rotary-encoder.

[LCD] LiquidCristal. http://www.arduino.cc/en/pmwiki.php?n=Tutorial/LiquidCrystal.

[Man13] Mannaf Hossain, "Computer Interfaced Logic IC Tester and R-C Meter", *global journal of researches in engineering electrical and electronics engineering,* volume 13, publicación 6, 2013.

[Pan04] Fang Pang, Tyler Brandon, Brucee Cockburn, Michael Hume, "A reconfigurable digital IC tester implemented using the ARM integrator rapid prototyping system", *Department of Electrical and Computer Engineering, University of Alberta, Canada,* 2004.

[WIR] Wire library. https://www.arduino.cc/en/Reference/Wire.