Recibido: 7 de noviembre del 2017
Aprobado: 5 de mayo del 2017

Llave electromagnética con combinación de un dígito para accionamiento de un servomotor, mediante el uso del Arduino y Simulink

Edwin Vinicio Altamirano Santillán
edwin.altamirano@espoch.edu.ec
Escuela Superior Politécnica del Chimborazo. Riobamba, Ecuador.

Geovanny Estuardo Vallejo Vallejo
geesvava@gmail.com
Escuela Superior Politécnica del Chimborazo. Riobamba, Ecuador.

Juan Carlos Cruz Hurtado
juankacruzhurtado@gmail.com
Instituto Superior Politécnico José A. Echeverría. La Habana, Cuba.

Resumen

Los interruptores electromagnéticos tipo *reed* suelen sustituir a los de fin de carrera cuando estos últimos no accionan debido a condiciones desfavorables, y tienen múltiples aplicaciones eléctricas y electrónicas. En este trabajo se muestra la introducción de un dígito en una combinación para el accionamiento de un servomotor. El estudio, además de mostrar el diseño de una llave electromagnética para el accionamiento de un servo, tiene como objetivo realizar el mencionado diseño empleando la plataforma Arduino programado con la herramienta Simulink. Esto abarata, facilita, acelera y dinamiza el diseño y, asimismo, posibilita efectuar la simulación del modelo en Simulink sobre la misma plataforma *hardware*. De esa manera, se permite precisar y hacer modificaciones en el diseño de forma muy rápida y simple.

Palabras clave: ***plataforma Arduino / hardware libre / Simulink / llaves electrónicas / interruptor magnético / servomotor***

Abstract

Electromagnetic key with combination of a digit for servomotor working, using Arduino and Simulink

Los interruptores electromagnéticos tipo *reed* suelen sustituir a los de fin de carrera cuando estos últimos no accionan debido a condiciones desfavorables, y tienen múltiples aplicaciones eléctricas y electrónicas. En este trabajo se muestra la introducción de un dígito en una combinación para el accionamiento de un servomotor. El estudio, además de mostrar el diseño de una llave electromagnética para el accionamiento de un servo, tiene como objetivo realizar el mencionado diseño empleando la plataforma Arduino programado con la herramienta Simulink. Esto abarata, facilita, acelera y dinamiza el diseño y, asimismo, posibilita efectuar la simulación del modelo en Simulink sobre la misma plataforma hardware. De esa manera, se permite precisar y hacer modificaciones en el diseño de forma muy rápida y simple.

Keywords: ***Arduino platform / free hardware / Simulink / electronic keys / magnetic switch / servomotors***

1. Introducción

1.1 Antecedentes

La primera plataforma de Arduino surgió en el 2005, en Italia. Inicialmente se produjeron doscientas placas por encargo de los estudiantes de un instituto de diseño, debido a la necesidad de contar con un *hardware* de bajo costo y de implementación fácil. La aceptación de esta plataforma prosperó rápidamente, pues varios comerciantes percibieron las ventajas de su bajo costo y sencillo uso (Evans, Noble y Hochenbaum, 2013). En el mercado existen diversas variantes de Arduino que se diferencian en el tipo de procesador, frecuencia de reloj, tamaño de la memoria flash y la cantidad de pines de entrada/salida que presentan (Evans *et al*., 2013). La plataforma Arduino es un *hardware* y *software* de fuente abierta, constituida sobre una tarjeta electrónica de entradas y salidas simples, con un ambiente de desarrollo realizado en lenguaje Processing. Se puede utilizar de forma autónoma con dispositivos interactivos (sensores y actuadores), o puede conectarse a un ordenador a través de un *software* con entornos esclavos. Por ser una plataforma de *software* y *hardware* libre, puede ser adquirida preensamblada o ensamblada a mano, pudiéndose descargar su entorno de desarrollo o IDE (por sus siglas en inglés) desde el sitio oficial de Arduino (Banzi, 2011).

En el IDE se escriben los *sketches*, o el programa, que corre en el ordenador con el que se programa el Arduino. Este *software* tiene una sintaxis determinada que cuando se suministra a la placa se traduce en un lenguaje "C", que luego se compila y se convierte en un lenguaje comprensible por el microcontrolador que soporta esta plataforma. También existen varias herramientas gráficas de programación para Arduino, tanto los de entornos autónomos como los de entornos esclavos. Estas herramientas persiguen, además de la programación del dispositivo, la monitorización del esquema implementado en el microcontrolador, así como la obtención de prototipo físico. Algunas de estas herramientas presentan un entorno de programación tan simple y poco descriptivo que dificultan la formaciónde una idea precisa de la topología del circuito.

La herramienta Simulink es un instrumento muy útil para programar el Arduino y muy difundida en el ámbito académico y profesional; y no precisa que se domine el lenguaje de programación de esta plataforma. Es un *software* apropiado para modelar, simular y analizar sistemas, tanto lineales como no lineales; además de contar con innumerables bloques funcionales propios, tiene dos librerías asociadas a la programación del Arduino. Las implementaciones que más aparecen en la literatura, sobre todo en videos, son las efectuadas mediante el uso del IDE del Arduino, el cual requiere un conocimiento del lenguaje y de más tiempo para el diseño.

1.2 Descripción problemática

El mercado ofrece diversas herramientas de programación gráfica para Arduino, las cuales no disponen de entornos y bloques gráficos, que faciliten, de forma fácil y rápida, la comprensión de la funcionalidad y de la visibilidad de la topología del diseño implementado.

1.3 Objetivo

Este trabajo se desarrolla con el fin de brindar una solución a la problemática descrita. Para ello, se plantea describir el diseño e implementación de una llave electromagnética con combinación de un dígito para accionamiento de un servomotor, mediante el uso del *hardware* Arduino programado con la herramienta Simulink.

Esto permite realizar el diseño de forma rápida, relativamente fácil y dinámica, igualmente posibilita la simulación del modelo sobre la misma placa Arduino. Así, se logra precisar, adecuar y modificar el diseño de manera muy ágil y simple.

1.4 Hipótesis

Para resolver el problema de investigación se propone la hipótesis de que si se cuenta con la herramienta Simulink y la placa Arduino, es posible diseñar modelos de fácil comprensión funcional y de bajo costo, asociados a la adquisición y al procesamiento relativamente complejo.

1.5 Enfoque

En el trabajo se describe la placa de Arduino Uno, la herramienta Simulink, las herramientas gráficas de programación y los dispositivos utilizados en la aplicación como el módulo servomotor, el interruptor magnético y la lámpara siete segmentos.

Del mismo modo, se especifica el diseño de la implementación propuesta, asociada a un sistema de llave electromagnética implementada con un interruptor magnético tipo *reed*, señalizándose los eventos de accionamiento del servomotor en una lámpara siete segmentos.

2. Metodología

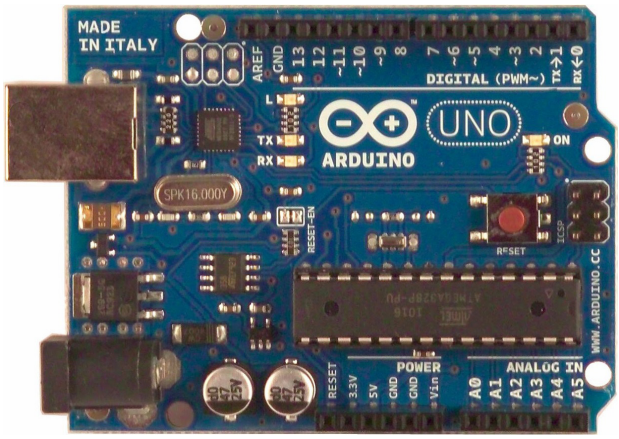
A continuación se detalla el diseño propuesto con el uso de la placa Arduino y la herramienta Simulink, y se describen las ventajas de este *software* respecto al resto de las herramientas gráficas de programación. Asimismo, se exponen las características de los elementos que se utilizan en el esquema propuesto.

2.1 Placa Arduino Uno

Toda la familia Arduino es de *hardware* libre y existen varias versiones de placas en el mercado internacional, fundamentadas en procesadores de 8bit Atmel AVR. Estas se diferencian en el tipo específico de microcontrolador, la frecuencia de reloj, el tamaño de su memoria y la cantidad de entradas y salidas con las que cuentan. Entre los diferentes tipos de placas se tienen el Arduino NG +, el *Decimilia*, *Duemilianove*, los Arduino Uno, el *Mega* 1280 y el *Mega* 2560 (Evans *et al*., 2013).

La placa Arduino Uno cuenta con 14 pines de entrada/salida, seis de los cuales se utilizan para señales de modulación de ancho de pulso (PMW) y coinciden con los pines 3, 5, 6, 9, 10 y 11. Tiene, además, 6 pines de entradas analógicas, un interruptor de *reset*, un conector USB para su conexión con la PC, fundamentalmente, y un conector para alimentarla externamente, ya sea a través de una fuente de tensión o por medio de baterías. En la figura 1 se muestra la imagen de este *hardware*, con su distribución de pines (Evans *et al*., 2013; Martín y Del Río, 2013; Lledó, 2012; Flores, 2012). Esta versión presenta compatibilidad de pines con respecto a versiones previas (Evans *et al*., 2013; Martín y Del Río, 2013; Lledó, 2012).

Figura 1. Esquema del *layout* de la placa Arduino Uno con su distribución de pines



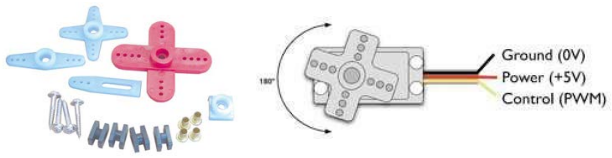
Fuente: Evans ***et al***. (2013)

2.2 Servomotor

Un servomotor es un dispositivo que con una tensión de corriente directa (CD) o un tren de pulso en su entrada de control, se hace posible que su eje gire un ángulo determinado. El eje permanecerá girado mientras no cambie la señal de control. Tiene tres entradas para su alimentación y controla su posición en 180º (Future Electronics, s. f.; González, s. f.).

Los servomotores, según su tamaño o el torque que pueden suministrar, se clasifican en: miniservo, servoestándar y servogigantes. Tanto los miniservos y los servoestándares pueden ser manipulados por las placas Arduino, sin necesidad de utilizar fuentes de potencia externas. Estos dispositivos se pueden controlar con circuitos simples como temporizadores NE555, o con otros microcontroladores, como los Raspberry, Lego, entre otros. En la figura 2 se observa la imagen de un servomotor tipo mini con sus cables de conexión y los diferentes tipos de hélices que tienen (Future Electronics, s. f.; González, s. f.).

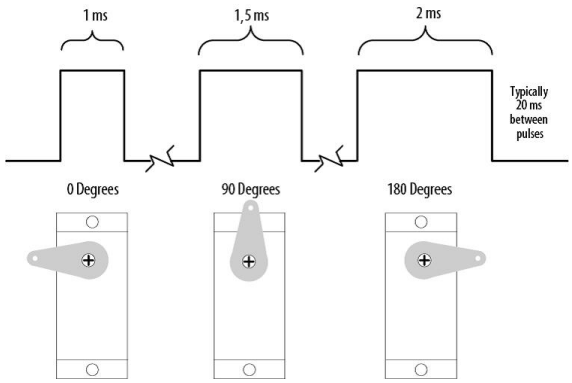
Figura 2. Miniservo con sus cables de conexión y sus diferentes tipos de hélices



Fuente: Future Electronics (s. f.)

Los servomotores se pueden controlar por señales de pulso, que pueden obtenerse de los pines de entrada/*salida* de los microcontroladores y placas de Arduino, en el régimen de señales moduladas por ancho de pulso (PWM). En este caso, el ángulo de giro del eje del servo es directamente proporcional con la longitud, en tiempo, del pulso de control en nivel alto. En la figura 3 se muestran algunos anchos de pulso de una señal PWM y su correspondiente ángulo del eje del servo.

Figura 3. Carta de tensión de la señal de control con los diferentes anchos de pulso, ángulos y posiciones del eje del servo



Fuente: Future Electronics (s. f.)

Entre las aplicaciones más comunes de los servomotores se encuentran las siguientes:

- Control de giro de las ruedas de carro de radiocontrol
- Movimiento de las articulaciones de un robot
- Control de las alas de los aviones no tripulados
- Control de las aspas de un helicóptero
- Movimientos de impulsos
- Control del timón de un barco de radiocontrol

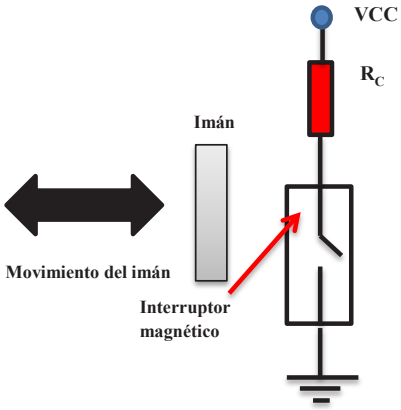
2.3 Interruptor *reed*

En el diseño se utilizó un interruptor tipo *reed*. Estos dispositivos son usados cuando los interruptores de fin de carrera, accionados mecánicamente, no pueden funcionar de manera satisfactoria debido a condiciones inapropiadas en su entorno como, por ejemplo, frecuencias de conmutación elevadas, existencia de polvo y suciedad, humedad elevada, etc. (Schmersal, s. f.; Satel, 2009).

Estos interruptores constan de dos elementos: el interruptor propiamente dicho y el imán que lo acciona. Presentan un empaquetamiento termoplástico y varían su forma según su aplicación; sus contactos pueden ser normalmente abiertos (NA), normalmente cerrados (NC) o biestables, y el imán —lateral o frontal— puede tener diferentes distancias de accionamiento. El contacto utilizado en la aplicación es del tipo NA y la distancia mínima de accionamiento entre imán e interruptor es 2,5 cm.

Este dispositivo se utilizó para proporcionar la combinación necesaria para el accionamiento del servomotor y la señalización del evento. Es el encargado de suministrar el dígito de la combinación para que ocurra el accionamiento o no. En la figura 4 se aprecia un esquema funcional y eléctrico del interruptor.

Figura 4. Esquema funcional y eléctrico del interruptor magnético tipo *reed*



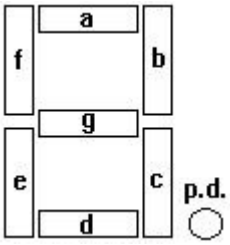
Elaboración propia

2.4 Lámparas siete segmentos

Son utilizadas para representar dígitos y algunas letras en los equipos electrónicos. A pesar de que en la actualidad se utilizan mucho los *displays* de cristal líquido LCD, aún se usan las lámparas siete segmentos por su simplicidad y bajo costo. Estas lámparas se componen por siete u ocho *leds* con diferente disposición y forma. Los primeros siete segmentos se encargan de formar los símbolos, y el octavo se designa para encender y apagar el punto decimal.

Cada segmento se concibe para que pueda activar de forma separada (como un diodo LED), y de esta forma se logra combinar los elementos y representar los números del 0 al 9. Los *display* siete segmentos más comunes son los de color rojo y amarillo, y a cada segmento se le asigna una letra que identifica su posición en el arreglo del *display*. En la figura 5 se muestra un *display* siete segmentos con la asignación de letras de cada segmento.

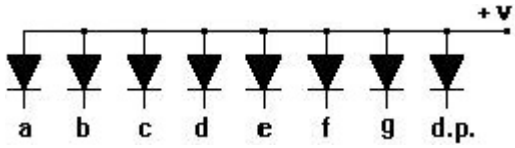
Figura 5. Lámpara siete segmentos



Fuente: Unicrom (2016)

Dependiendo de la forma en que se alimenten estos dispositivos, los hay de ánodo común y de cátodo común. En el de ánodo común, los ánodos de los diodos de cada segmento se encuentran unidos y se conectarán al polo positivo de la fuente de alimentación. En este caso, para activar el segmento correspondiente se debe conectar el cátodo a tierra mediante una resistencia determinada, para limitar la corriente a la nominal. En la figura 6 se observa el esquema eléctrico con la topología de alimentación de este tipo de *display*.

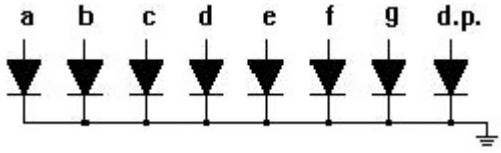
Figura 6. Esquema de alimentación de lámpara siete segmentos de ánodo común



Fuente: Unicrom (2016)

En el caso del *display* de cátodo común, los cátodos son los que se encuentran unidos y conectados a la toma de tierra. Para activar el elemento correspondiente se necesita conectar el ánodo al potencial positivo (V_{CC}) empleando una resistencia para garantizar la corriente nominal. En la figura 7 se puede ver el esquema de alimentación del *display* cátodo común.

Figura 7. Esquema de alimentación de lámpara siete segmentos de cátodo común



Fuente: Unicrom (2016)

2.5 Herramientas gráficas de programación para Arduino

A continuación se presentan las características fundamentales de las diferentes herramientas de programación gráfica que interactúan con la placa Arduino.

2.5.1 Entornos de programación gráfica para Arduino

La descripción de los entornos de programación es como sigue:

- **Entornos autónomos.** Se utilizan para programar el *hardware* y garantizan su trabajo autónomo. Dentro de este tipo se encuentran el *Minibloq*, el *Ardubloq*, el *Amici*, el *Mind+*, el *ModKit*, el *VirtualBreadBoard* y el *VBB-JARVIS,* entre otros.

- **Entornos esclavos** Se usan solamente para el monitoreo de la aplicación. Dentro de este tipo se encuentran el *Etoys (Squeak)*, el *S4A (Scratch)*, el *Snap!,* el *Scratch* 2.0, el *Pure Data*, el *Labview*, el *Firefly (Rinho-Grashoper)*, el *MyOpenLab*, entre otros.

- **Otros entornos.** Son utilizados fundamentalmente para el desarrollo de prototipos y para confeccionar el circuito impreso (PCB). Entre estos se destaca el *software Fritzing* (Ruiz, 2014).

Características fundamentales de las herramientas gráficas

- **Herramientas de entornos autónomos**

  i) Herramienta *Minibloq*

  Es un entorno de programación para el kit de robótica Multiplo, de Arduino. Su objetivo fundamental es llevar la computación física y las plataformas robóticas a la enseñanza primaria y a los principiantes. Proporciona soluciones portables, rápidas, modulares y ampliables.

  ii) Herramienta *Ardublock*

  Su función fundamental es generar un código compatible con el entorno IDE de Arduino. Sus ventajas fundamentales son: gratuidad, genera *sketch* para Arduino directamente, ofrece colección de bloques muy elementales, es una herramienta indicada para niveles educativos básicos, su instalación es muy sencilla, etcétera.

  iii) Herramienta *Amici*

  Fue de los primeros entornos utilizados para programar Arduino, que genera su código. Se suministra con una versión de IDE de Arduino, lo que permite que cuando se realiza el programa en un entorno gráfico se genere el código a la vez que se abre el IDE y se descargue la aplicación en la placa Arduino.

  iv) Herramienta *Modkit*

  Permite programar el Arduino con bloques gráficos y con código de texto tradicional. La herramienta se ejecuta en un navegador web y requiere un *widget* para la comunicación con la placa. Las etapas de trabajo de esta herramienta son: selección del *hardware*, configuración de las E/S, confección del algoritmo con los bloques de la librería (Output, Input,

Operadores, Control y variables); y la última etapa es la descarga de la aplicación en la placa Arduino.

v) Herramienta *VirtualBreadBoard*

Constituida por un entorno de simulación y desarrollo fácil de usar, puede sustituir un *protoboard* para experimentar con nuevos diseños. Esta herramienta posibilita el diseño virtual en el *protoboard*, igual que realizar el PCB, importar *sketches* de Arduino y descargar la aplicación sobre la placa de Arduino.

vi) Herramienta *VBB-JARVIS*

También es un tipo de herramienta *protoboard* virtual que facilita el prototipado rápido de circuitos electrónicos. Permite el diseño y emulación, proporciona la posibilidad de montaje y cableado así como de generación automática de montajes *hardware* partiendo de un fichero determinado (.ino), y admite la generación de código a partir de la realización manual de montaje (Ruiz, 2014).

- **Herramientas de entornos esclavos**

i) Herramienta *Physical Etoys*

Esta herramienta de programación da la posibilidad de conectar el mundo virtual de los ordenadores con el mundo físico. En consecuencia, con esta herramienta se pueden programar objetos del entorno real (como robots), o mover objetos gráficos en la pantalla del ordenador a través de variables del mundo físico. La herramienta cuenta con un entorno adecuado para la enseñanza de niveles iniciales.

ii) Herramienta S4A (*Scratch*)

Es una herramienta de programación gráfica desarrollada en el entorno *Scratch*; se creó en el Instituto Tecnológico de Massachusetts (MIT) y es uno de los entornos más conocidos.

iii) Herramienta *Labview*

Debido a la conexión de modo esclavo de esta herramienta, solo es posible visualizar y controlar las E/S del Arduino. Se emplea en aplicaciones de instrumentación donde el *hardware* se utiliza como dispositivo de adquisición de datos, que en este caso sería de bajo costo.

iv) **Herramienta *Rinho+Firefly***

El entorno gráfico de esta herramienta se conecta al Arduino por medio del *plugin Grasshopper,* que es un entorno muy fácil de utilizar. Una de las librerías de este *plugin* se llama *Firefly*, pensada para interactuar con todas las E/S del Arduino.

v) **Herramienta *MyOpenLab***

Orientada a la simulación y modelado de sistemas físicos, electrónicos, robóticos y de control de procesos, está dirigida al área de las aplicaciones didácticas. La filosofía de su entorno de desarrollo es muy parecida a la del *LabView* (Ruiz, 2014).

- Otros entornos

Entre otros entornos se puede citar a la herramienta *Fritzing,* que constituye una ayuda en la confección de prototipos. Es un *software* de código abierto de ayuda para diseñadores, investigadores y aficionados para facilitar la elaboración del prototipo real. Entre sus características fundamentales e encuentran las siguientes: es de uso fácil, posibilita la exportación del diseño del prototipo de formatos compatibles con los existentes en las empresas encargadas de su confección, y permite el añadido de nuevos bloques en la librería de componentes (Ruiz, 2014).

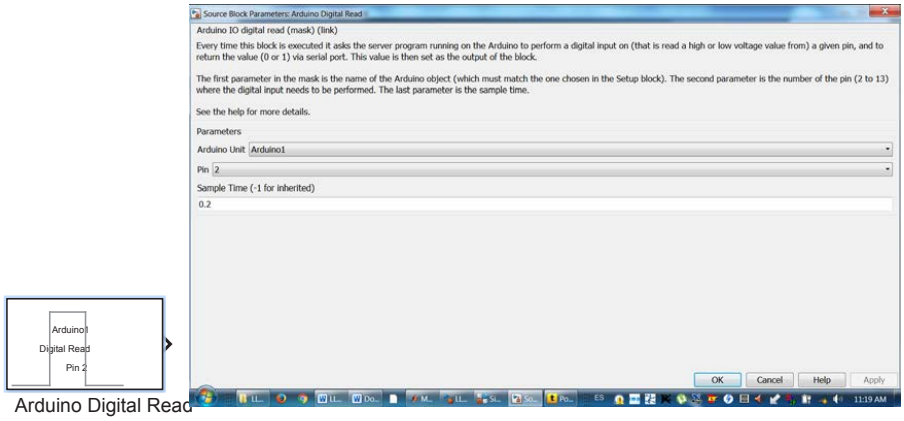
2.6 Descripción de la herramienta Simulink

La herramienta Simulink se utiliza en diferentes áreas de las ciencias exactas, fundamentalmente en las ingenierías, tanto en el diseño y simulación como en el análisis de modelos. Tiene la facilidad de un GUI para el desarrollo de diagramas en bloque de las diferentes aplicaciones por instrumentar. Dispone de un gran número de librerías que van desde las de ingeniería hasta las matemáticas, lógicas y especiales. Además, cuenta con dos librerías para el trabajo con las plataformas de Arduino. A continuación se detallan, en forma breve, los bloques que se usan en el diseño que se describe en el trabajo, comenzando por los bloques de Arduino que se emplean en la aplicación propuesta (Karris, 2006).

2.6.1 Bloque Digital Read

Se utiliza para la lectura digital de un pin determinado del Arduino, obteniéndose el valor lógico de 0 o 1 a través del puerto USB. Este valor se obtiene en la salida de este bloque dentro del modelo Simulink. En la figura 8 se muestra el bloque y su caja de diálogo para su configuración.

Figura 8. Bloque funcional *Digital Read* y caja de diálogo de configuración del bloque



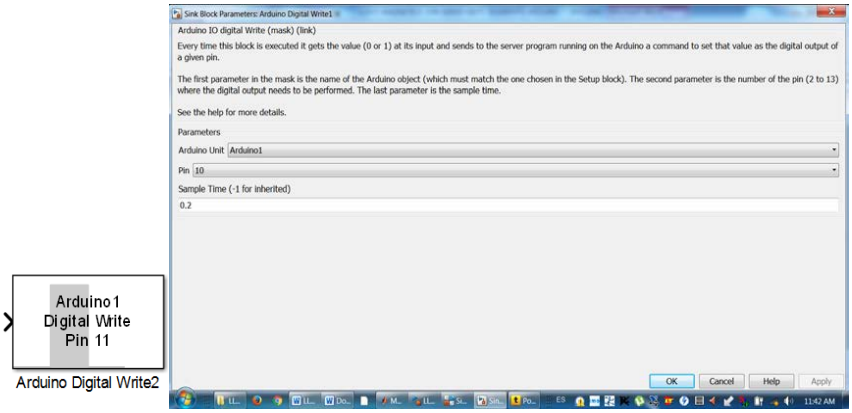
Fuente: Simulink (2013)

El primer parámetro sirve para definir el tipo de placa Arduino que se utilizará. El segundo parámetro es el número del pin que se usa para la adquisición de la tensión digital de entrada (pines del 2 al 13 para el Arduino Uno). El número de pin seleccionado figura dentro del bloque que aparece en el modelo.

2.6.2 Bloque *Arduino Digital Write*

Tiene la misma filosofía que el bloque anterior, con la diferencia de que su función es la de brindar el valor lógico (1 o 0) que se ubique en la entrada del bloque, dentro del modelo en Simulink, en el pin de salida seleccionado en la placa Arduino. El primer parámetro que aparece en la caja de parametrización es el tipo de Arduino con el que se trabajará. El siguiente parámetro es el pin de salida donde se requiere ubicar la tensión lógica en la placa. En la figura 9 se muestra el bloque funcional y la caja para su configuración.

Figura 9. Bloque *Digital Write* y caja de diálogo de configuración del bloque



Fuente: Simulink (2013)

2.6.3 Bloque contador

Este bloque realiza el conteo ascendente o descendente hasta una cantidad determinada por el parámetro: rango de conteo. El bloque también habilita la entrada incremento (Inc.) cuando se configura para conteo ascendente. Si se configura para el conteo descendente se habilita la entrada Dec. En caso de configurarse en corrida libre, se deshabilitan los puertos Inc. y Dec. y se efectúa un conteo libre en un tiempo determinado. Para el resto de los ajustes del parámetro evento, el bloque realiza un incremento o decremento del contador, cada vez que el evento disparo ocurre en la entrada Inc. o en la entrada Dec., respectivamente. Al ocurrir un pulso en el puerto opcional R_{ST}, se resetea el contador a su estado inicial de cero. Dentro de los parámetros que se pueden configurar en su caja de diálogo se encuentran los siguientes:

– Dirección de conteo: ascendente o descendente.

– Tipo de evento de conteo: por flanco de subida, por flanco de caída, por flanco de caída o de subida.

– Tamaño del contador: definida por el usuario, de 8, 16 o 32 bits y especificar por puerto de entrada.

– Conteo máximo: es el número máximo de conteo del contador antes de pasar a su estado inicial.

– Conteo inicial: es el valor por donde inicia el conteo de eventos.

– Salida: puede ser de conteo, de *Hit* o de ambos.

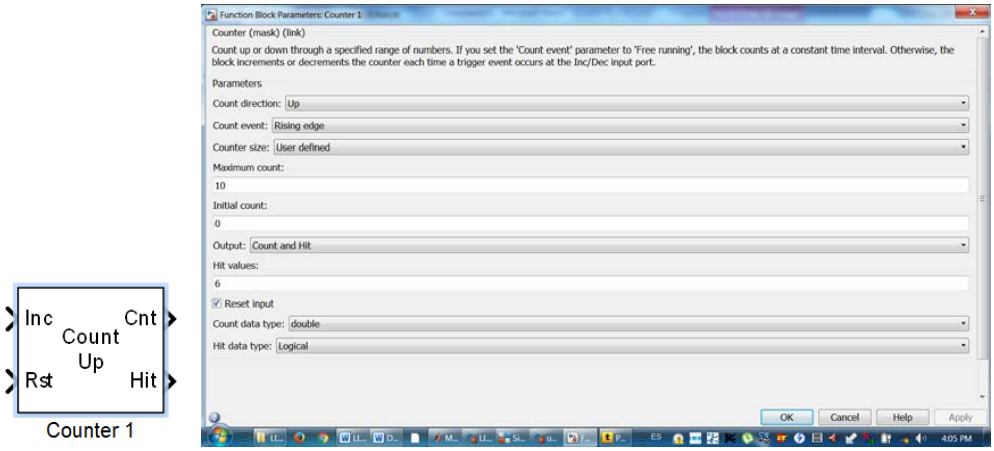
– Valores del *Hit*: cuando el contador llega a este valor, la salida *Hit* pasa a un "1" lógico.

– Tipo de dato del contador: especifica el tipo de dato en el puerto de conteo Cnt. Este parámetro aparece solamente cuando se configura el parámetro de salida del contador a *Count* o a *Count and Hit*.

– Tipo de dato de *Hit*: puede ser lógico o booleano.

En la figura 10 se muestra el bloque contador y la caja de diálogo de parametrización del contador.

Figura 10. Bloque contador y caja de diálogo de parámetros del bloque

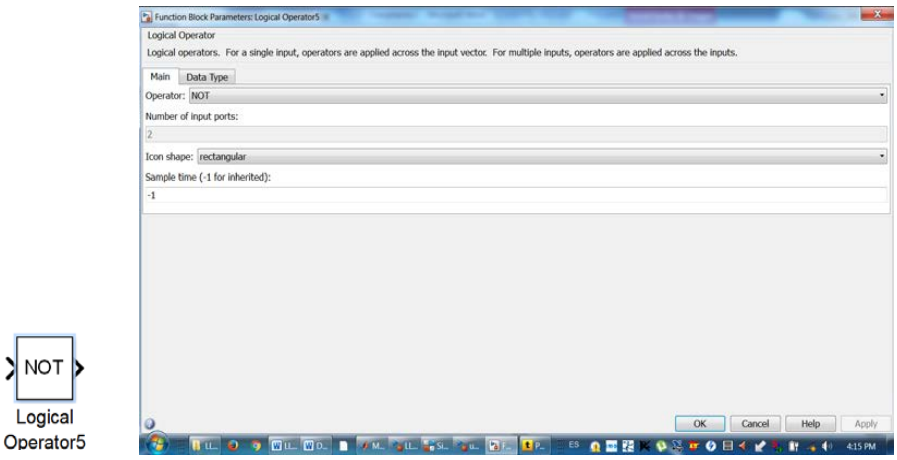


Fuente: Simulink (2013)

2.6.4 Bloque operador lógico

Este bloque está constituido por las compuertas lógicas conocidas: NOT, AND, OR, NAND, NOR, XOR y NXOR. En la figura 11 se aprecian el bloque y su caja para la parametrización.

Figura 11. Bloque del operador lógico y caja de diálogo de parámetros del bloque



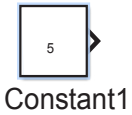
Fuente: Simulink (2013)

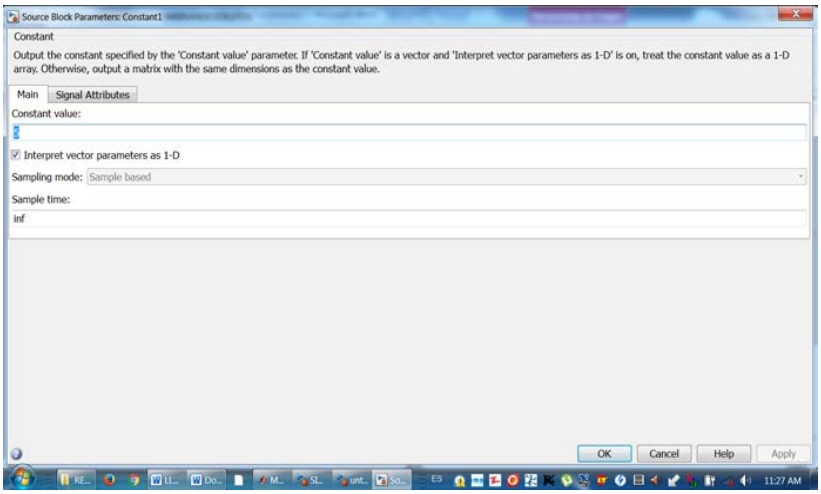
2.6.5 Bloque constante

Este bloque genera un valor real o complejo. Además, puede generar un escalar, vector, o matriz de salida, dependiendo de la dimensionalidad del parámetro, del

parámetro de valor constante o del ajuste del parámetro del vector. En la figura 12 se presentan el bloque y la caja de diálogo para su parametrización.

Figura 12. Bloque constante y caja de diálogo de parámetros del bloque





Fuente: Simulink (2013)

2.6.6 Bloque comparador

Designado para comparar la señal de entrada con un valor constante que se asigna por el usuario. Esta constante se especifica en el parámetro "valor constante". Además, el bloque cuenta con la opción de cómo la entrada se compara con este valor constante por medio del parámetro "Operador". Estas son las opciones de comparación que brinda este parámetro:

== Determina si la variable de entrada es igual a la constante especificada.

~= Determina si la variable de entrada no es igual a la constante especificada.

< Determina si la variable de entrada es menor que la constante especificada.

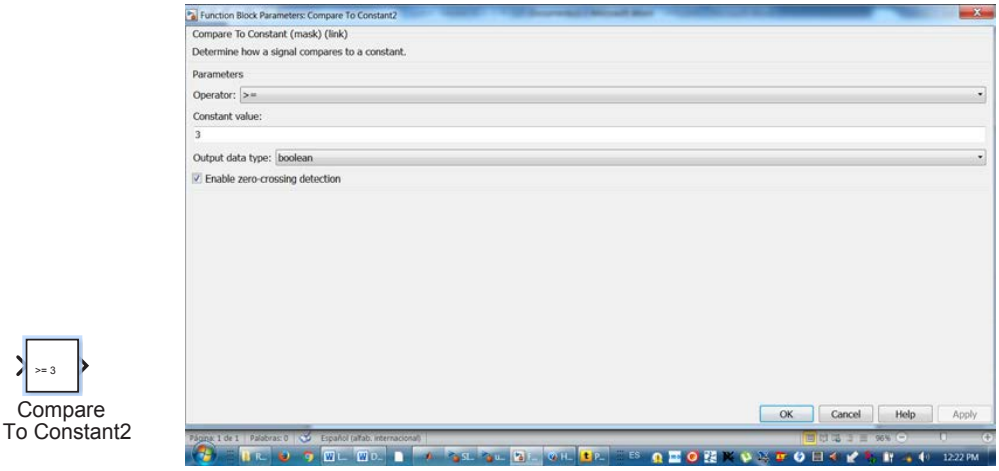
<= Determina si la variable de entrada es menor o igual a la constante especificada.

> Determina si la variable de entrada es mayor que la constante especificada.

>= Determina si la variable de entrada es mayor o igual a la constante especificada.

En la figura 13 se observan el bloque comparador en cuestión y la caja de diálogo para la parametrización del bloque.

Figura 13. Bloque comparador y caja de diálogo de parámetros del bloque

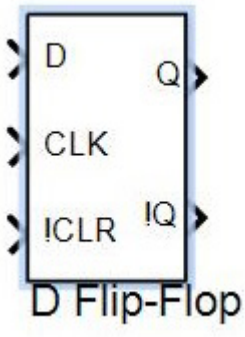


Fuente: Simulink (2013)

2.6.7 Bloque Flip-Flop tipo D

El bloque Flip-Flop D tiene tres entradas: la entrada de datos D, la señal de reloj CLK y !CLR, que habilita la señal de entrada. Un pulso ascendente de la señal de reloj, si el bloque está habilitado (!CLR si está en "1" lógico), hará que la salida Q tome el mismo valor que la entrada D. En la figura 14 se muestra el bloque.

Figura 14. Bloque Flip-Flop tipo D



Fuente: Simulink (2013)

2.6.8 Bloque de escritura de servo estándar (*Arduino Standard Servo Write*)

Se utiliza para rotar el eje de un servomotor desde 0 a 180 grados. Para efectuar la rotación, el bloque deberá enviar valores comprendidos en este rango, que se

ubican en la entrada de este bloque, hacia el pin seleccionado en cuestión. También se pueden utilizar pulsos para colocar el eje del servo en el ángulo deseado, como ya se explicó. Existen limitaciones que se deben tener en cuenta a la hora de utilizar este bloque; las principales son:

– No utilizar el bloque en modo de simulación externo.

– Si se utiliza este bloque en modelos con los bloques *Serial Receive y Serial Transmit*, usar números de muestra mayores para evitar *overruns*.

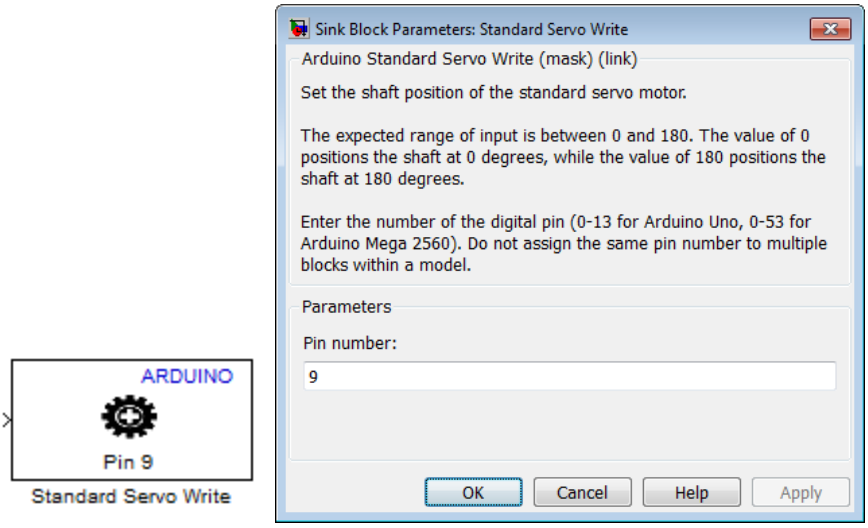
– El máximo número de estos bloques en un modelo de Simulink es de 12 para el Arduino Uno, y 48 para Arduino Mega 2560.

– Si se utiliza el Arduino Uno en el modelo, el bloque PWM no puede utilizar los pines 9 y 10 cuando el modelo contiene bloques servo.

En el caso de utilizar el *Arduino Mega 2560*, el bloque PWM no podrá utilizar los pines 11 o 12 cuando el modelo contenga más de 12 bloques servo.

En la figura 15 se muestran el bloque *Arduino Standard Servo Write* y su correspondiente caja de diálogo.

Figura 15. Bloque *Arduino Standard Servo Write* y caja de diálogo de configuración del bloque



Fuente: Simulink (2013)

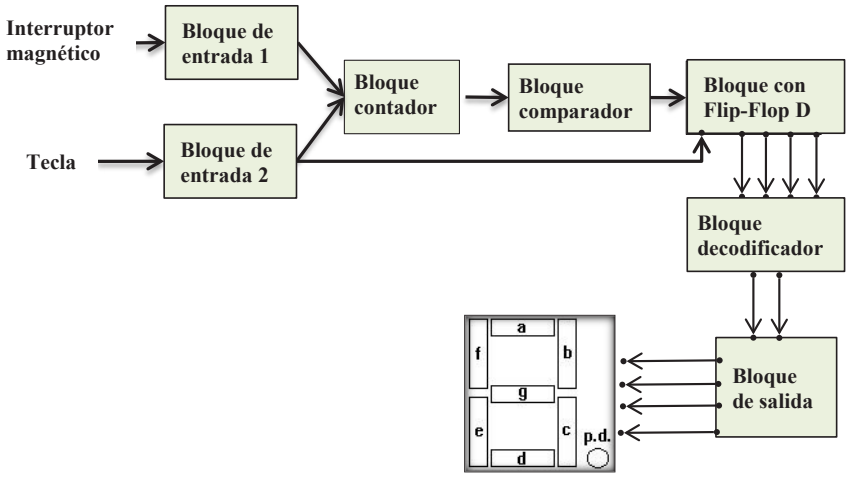
3. Diseño de la implementación propuesta con Arduino y Simulink

A continuación se describe la aplicación propuesta para el accionamiento de un servomotor y su señalización en una lámpara siete segmentos, asociado a una combinación que se proporciona a través del número de veces que se acerca un imán a un interruptor magnético tipo *reed*. La combinación en cuestión viene dada por un dígito y ceros.

3.1 Implementación del accionamiento del servomotor por combinación de un dígito y ceros

El accionamiento del servo y la señalización en la lámpara siete segmentos vienen dados por la combinación de cuatro dígitos. Esta aplicación tiene la particularidad de que la combinación se conforma con ceros y un dígito que se repite en la secuencia. Su diagrama en bloques está conformado por bloques de entrada, bloque contador, bloque comparador, bloque con Flip-Flops tipo "D", bloque decodificador, bloques de salida y bloque de lámpara siete segmentos. En la figura 16 se presenta el diagrama en bloques de la aplicación.

Figura 16. Diagrama en bloques de accionamiento por combinación de un dígito



Elaboración propia

A continuación se detallan las funciones de los bloques del diagrama anterior.

- **Bloques de entrada.** El diagrama está compuesto por dos bloques de entrada, constituidos por los bloques *Arduino Digital Read* de la herramienta Simulink. Tiene uno conectado a la salida del interruptor magnético, en este caso a través del pin 2 del Arduino, y el otro está conectado a una tecla por medio del pin 8

entrada del Arduino. La entrada 1, conectada a la salida del interruptor magnético, se acopla a la entrada de incremento del contador (Inc.) y la entrada 2 se conecta al *reset* del contador y a las entradas de los relojes de los Flip-Flop tipo "D".

- **Bloque contador.** El contador de Simulink es el encargado de contar los acercamientos que tiene el imán al interruptor magnético. Y su entrada de *reset*, como se comentó, está conectada a la tecla. Cada vez que se acerca y se aleja el imán, se produce un pulso a la entrada Inc. del contador y cuando se pulsa la tecla la salida del contador, este pasa a cero su salida.

- **Bloque comparador.** Este bloque se utiliza para detectar cuando el contador llega a su valor de constante. Este valor constituye el dígito que se combina con los ceros de la secuencia de apertura, o de accionamiento del servomotor. Este valor se modificará cada vez que se requiera cambiar la secuencia combinatoria. La entrada de este bloque es la salida del contador y su salida se conecta a la entrada del primer Flip-Flop "D" de la cadena o registro. Su salida pasará a uno lógico cada vez que su entrada llegue al valor de su constante y, será cero para cualquier otro valor de su entrada.

- **Bloque de Flip-Flop tipo "D".** La entrada del bloque se encuentra conectada a la salida del comparador y sus salidas se acoplan a la entrada del decodificador. Este bloque se utiliza para ir pasando el valor lógico de la entrada a sus diferentes salidas. No es más que un registro de desplazamiento encargado de ir trasladando el dato del comparador cada vez que se pulse la tecla y ocurra un pulso de reloj en sus entradas CLK.

- **Bloque decodificador.** Este bloque es el encargado de determinar la combinación correcta requerida para accionar el servomotor y señalizar este evento en la lámpara siete segmentos. Se trata de un decodificador de cuatro a dos líneas. Está constituido por una combinación de compuertas lógicas AND y NOT, en este caso. Las entradas a este bloque son las salidas de los Flip-Flops y sus salidas se acoplan al bloque de salida. Las combinaciones dadas por el dígito que se repite, al igual que los ceros, van pasando al registro de desplazamiento (Flip-Flop D). Una línea de salida del decodificador irá conectada al bloque *Arduino Standard Servo Write* para accionar el servo.

- **Bloque de salida.** Está constituido por los bloques *Digital Write* de la herramienta Simulink. Dos salidas provenientes del bloque decodificador y las cuatro salidas de este bloque son suficientes para garantizar el encendido de los segmentos correspondientes, según sea el caso.

- **Bloque de lámpara siete segmentos.** Es el bloque designado para señalizar

los eventos que ocurran en el sistema. Aparecerá una "A" para el caso de que se produzca una combinación verdadera de entrada. En el caso contrario en la lámpara permanecerá una "C". Se debe observar que cuando la lámpara pasa de "C" a "A" (ver figura 16), se apaga el segmento "d" y se encienden "b", "c" y "g". Los segmentos comunes tanto para "A" y "C" son los "a", "f" y "e", por lo que estarán siempre activados. En este caso, la lámpara siete segmentos utilizada es de ánodo común, por lo que habrá que conectar el segmento por activar, a través de una resistencia limitadora, a una tensión de 0 volt.

3.2 Explicación de la implementación de llave de un dígito

Con el apoyo del diagrama general de la figura 16 se explicará en detalle esta implementación. En la figura 17 aparece el modelo en Simulink de este diseño.

Como se comentó, la entrada 1 (bloque, *Arduino Digital Read* del modelo en Simulink), con el pin 2 seleccionado, se conectó al nodo formado por la resistencia Rc y el interruptor magnético. Este nodo se conectó, físicamente, al pin 2 de la placa Arduino. Cada vez que se acerque el imán al interruptor, esta salida irá a un nivel de tensión de 0 volt, y cuando se aleje pasará a un nivel alto (5 volt). Por cada ciclo de acercamiento se producirá un pulso en la entrada Inc. del contador. La entrada de *reset* del contador está acoplada a la entrada 2 configurada para el pin 8 del Arduino. En este pin 8 del Arduino se conecta, físicamente, la salida de la tecla.

Por otro lado, cada vez que ocurra una oscilación de acercamiento del imán, el contador realizará un conteo de este evento. Esta será la forma de introducir el dígito de la combinación del sistema de accionamiento del servo. Respecto a la tecla, cada vez que se pulse dará un pulso de *reset* al contador y otro a las entradas de reloj de los Flip-Flop tipo "D" de forma simultánea. Por tanto, cada vez que se pulse la tecla se reiniciará el conteo del contador y pasará a cero su salida, además de que transferirá a sus salidas el dato que tenga cada Flip-Flop en sus entradas.

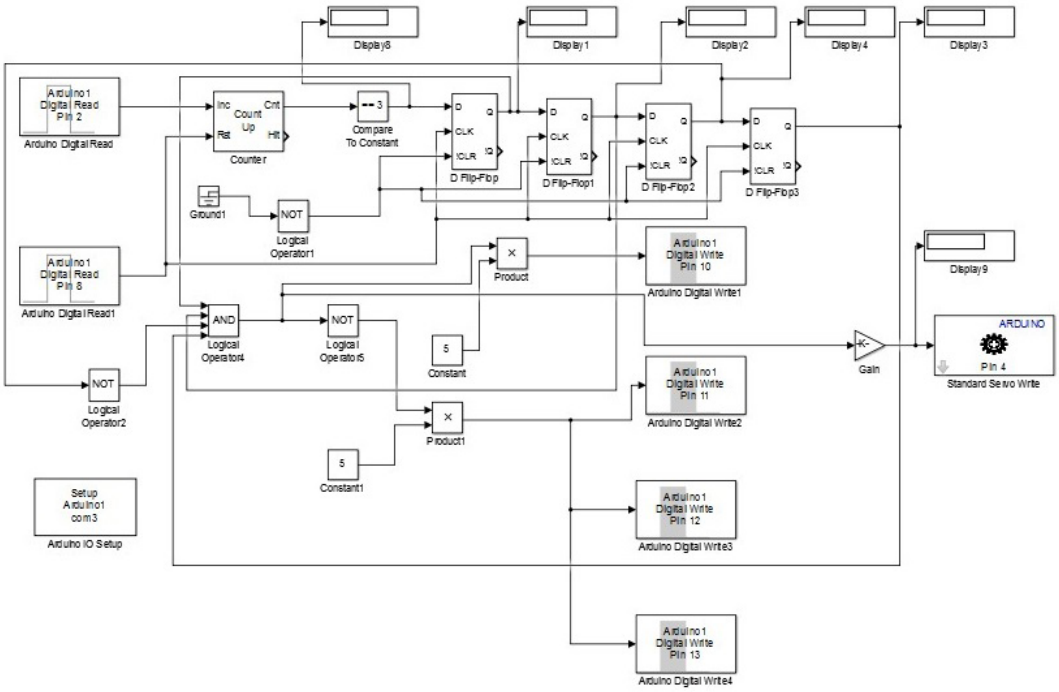
El contador se configura de la forma siguiente: dirección de conteo ascendente, conteo con pulso de subida y máxima cantidad de conteo en 6. Este valor máximo de conteo es importante ya que estará directamente relacionado con el dígito de la combinación, y con la cantidad de veces que se repita en dicha combinación. Este valor de 6 se debe a que, en este caso específico, la combinación es: 3033 y, como se observa, la sumatoria de los dos últimos dígitos de la combinación suma 6. Esto provocará que una vez que se introduzca el primer "3" se limpie el contador al introducir el cero próximo de la secuencia, y solo faltarían dos números 3 de la secuencia. Cuando se introduzcan estos dos últimos dígitos, el contador irá a cero y se reiniciará.

Para introducir el cero de la combinación, luego de haber introducido el primer

"3", solo habrá que pulsar la tecla. Esto hará que se introduzca un cero en el registro, formado por el Flip-Flop "D", en la posición requerida dentro de la combinación. Note que al introducirse un dígito se pulsa la tecla para pasar el "1" lógico correspondiente al registro, lo que hace que se *resetee* el contador y aparezca un cero en el comparador.

El comparador con constante, conectado a la salida del contador, se encuentra configurado para que pase a "1" lógico su salida cuando en su entrada se tenga el número 3 de conteo. En ese caso, se pasará un "1" a la salida del primer Flip-Flop al pulsarse la tecla.

Figura 17. Modelo en Simulink del circuito de accionamiento del servo con combinación de un dígito



Fuente: Simulink (2015)

El registro de desplazamiento formado por los Flip-Flops (F-F) conectados en serie, serán los encargados de ubicar la combinación lógica en su salida. Esta combinación lógica se identifica con la secuencia correcta de dígitos preestablecida, que accionará el servo y señalizará este evento en la lámpara siete segmentos. Las salidas de cada Flip-Flop se encuentran conectadas al circuito combinacional que responde a la combinación digital verdadera.

El circuito combinacional, o decodificador, está formado por una compuerta AND y dos NOT configurados respecto a la tabla de la verdad, que escribirá una "A" en la lámpara siete segmentos y accionará el servo, luego de pasar por un bloque de ganancia "K" cuando ocurra la combinación verdadera. En caso contrario, aparece una "C" en la lámpara y no se accionará el servomotor. Las dos salidas del decodificador se encargan de la señalización y del accionamiento. La salida de la compuerta AND, como se observa en la figura 17, se conectó al segmento "d" y al servomotor. La otra salida, a través de una compuerta NOT, está conectada a los segmentos "b", "c" y "g". También se aprecia que estas dos salidas pasan por sendos bloques multiplicadores, por un factor constante, para ser transformados en tensiones de 0 o 5 voltios y poder manejar la lámpara.

Los bloques de salida se implementan a través de los bloques *Digital Write*, conectados a los pines del 10 al 13 del Arduino. La salida conectada al pin 10 se acopla al terminal 4 (segmento d) de la lámpara, y las salidas conectadas a los pines del 11 al 13 se acoplan a los terminales 15 (segmento b), 2 (segmento c) y 18 (segmento g), respectivamente. Este bloque de salida también comprende el bloque *Standard Servo Write* para manejar el servo.

3.3 Explicación del funcionamiento del sistema de accionamiento y señalización para un dígito

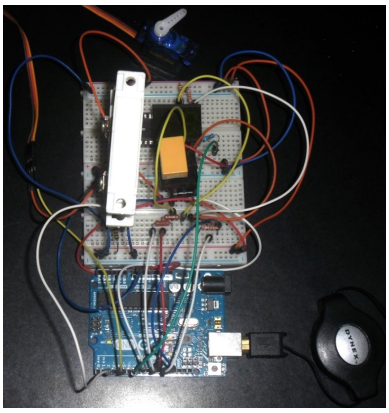
Como ya se dijo, cada dígito que aparece en la secuencia constituye el número de acercamientos del imán al interruptor y el cero será solo con la pulsación de la tecla. A continuación se explica, en forma breve, el funcionamiento del esquema completo dada la secuencia combinatoria: "3033".

Una vez que se producen los primeros tres acercamientos del imán, el contador tendrá el dígito 3 a su salida y la salida del comparador pasará a "1". Cuando se pulsa la tecla este "1" pasa a la salida del primer F-F y se *resetea* el contador. Para obtener el cero, que sigue en la secuencia, solo se pulsaría la tecla, con lo que el "1" anterior pasa a la salida del segundo F-F y el cero a la del primer F-F. Luego se realizan otros tres acercamientos del imán y se tendrá otro "1" a la salida del comparador, que al pulsar la tecla, nuevamente, traslada la combinación "101" introducida hasta este momento. El último dígito "3" de la secuencia, se introduce de la misma manera, conformándose la combinación 1101 en el registro de desplazamiento. Y es esta combinación la que provocará la escritura de la "A" en la lámpara siete segmentos y el accionamiento del servo. Obviamente, en caso de efectuarse otra cantidad de acercamientos o alteración en la secuencia, permanecería la "C" en el siete segmentos y no se accionaría el servo.

En caso se quiera cambiar la combinación de entrada y se pudiera situar otro dígito en la constante del comparador, se puede rediseñar el circuito combinacional o se hacen

ambas cosas. En la figura 18 se puede ver la imagen del circuito de accionamiento en el *board* y el Arduino.

Figura 18. Imagen del montaje del circuito de accionamiento propuesto en *board* y el Arduino conectado



Fotografía de los autores

4. Resultados

Para realizar la comprobación del funcionamiento del esquema de accionamiento y señalización con un dígito, se efectuaron estas maniobras:

- Al efectuar el procedimiento correcto para la introducción del dígito, se probó introduciendo otro dígito con la misma secuencia y no resultó.
- Al realizar la maniobra correcta de introducción del dígito, se cambió el lugar del cero en la secuencia utilizando la tecla y no hubo accionamiento del servo.
- Se hicieron varias secuencias de combinaciones de acercamiento del imán al interruptor solamente, sin el uso de la tecla, y no hubo accionamiento del servo.

Después de las comprobaciones anteriores, se corroboró que solamente con la secuencia y maniobra correctas se produce el accionamiento del servo y la señalización de conformidad de este evento en la lámpara siete segmentos.

5. Conclusiones y recomendaciones

5.1 Conclusiones

i. En el trabajo se describe detalladamente el diseño de una aplicación que utiliza un *hardware* libre programado con una herramienta gráfica como el Simulink. Para esto

también se presentó la descripción de los dispositivos utilizados y los bloques de Arduino que tiene el Simulink. Asimismo, se mencionaron brevemente algunos de los programas de programación gráfica que se usan para el Arduino. Al utilizar esta plataforma y el soporte para programarla, se realizó de forma muy fácil y dinámica el diseño, y se tuvo la posibilidad de simularlo en la misma placa de Arduino, lo que permitió precisar y comprobar este diseño de forma muy simple y ágil.

ii. Se obtuvo una especie de llave electromagnética de un dígito, de muy bajo costo, que posibilita el accionamiento de un servomotor con la debida señalización en una lámpara de siete segmentos. Este esquema podría tener muchas otras aplicaciones.

iii. Aun utilizando el Arduino Uno, el esquema tiene un consumo de recursos que podría albergar otras aplicaciones simples.

iv. El trabajo ilustra la forma de utilizar la herramienta Simulink para diseñar un esquema, muy simple en este caso, y simularlo.

5.2 Recomendaciones

i. Implementar un diseño de experimento que automatice, lo máximo posible, la introducción de la combinación para evaluar el tiempo máximo de introducción de la secuencia.

ii. Realizar la programación de la placa utilizando otros lenguajes y programas gráficos y comprobar el tiempo máximo de introducción de secuencia; si hay más optimización de recursos, el consumo de potencia que haga factible el uso de baterías, etcétera.

Referencias

Banzi, M. (2011). *Getting Started with Arduino*. California, Estados Unidos: O'Reilly Media.

Evans, M., Noble, J., y Hochenbaum, J. (2013). *Arduino in action*. Nueva York, Estados Unidos: Manning Publications.

Flores, F. (2012). *Estudio, diseño e implementación de módulos de entrenamiento sobre plataforma Arduino para el Laboratorio de Microcontroladores de la Facultad de Ingeniería Electrónica de la Universidad Tecnológica Israel* (Tesis de pregrado). Universidad Tecnológica Israel, Quito, Ecuador. Recuperado de 190.11.245.244/bitstream/47000/815/1/UISRAEL-EC-ELDT-378.2%2042-94.pdf.

Future Electronics. (s. f.). Servo Motors Control & Arduino. Future Electronics. Recuperado de https://www.yumpu.com/en/document/view/22889971/introduction-to-servo-motors-arduino-arduino-egypt

González, L. (s. f.). Control de un servo con Arduino [Material de clase]. Sevilla: IES Los Viveros Dpto. Electrónica. Recuperado de http://www.ieslosviveros.es/alumnos/asig8/carpeta811/control_de_un_servo_con_arduino.pdf

Karris, S. (2006). *Introduction to Simulink® with engineering applications*. California, Estados Unidos: Orchard Publications.

Lledó, E. (2012). *Diseño de un sistema de control domótico basado en la plataforma Arduino* (Tesis de pregrado). Universidad Politécnica de Valencia, España. Recuperado de https://riunet.upv.es/bitstream/handle/10251/18228/Memoria.pdf

Martín, A., y Del Río. M. (2013). *Control de posición de un balancín con Arduino* (Proyecto de fin de carrera). Universidad de Valladolid, España. Recuperado de https://uvadoc.uva.es/bitstream/10324/3407/2/PFC-P-78%3B79.pdf

Ruiz, J. (2014). Herramientas gráficas de programación para Arduino. En *II Jornadas Murcia Lan Party. OSHWCon Murcia*. Murcia, España: Centro Social Universitario.

Satel. (2009). Detector de vibración con contacto magnético VD-1. Gdansk, Polonia: Autor. Recuperado de https://www.satel.pl/es/download/instrukcje/vd1_io_es_ 0908.pdf

Schmersal. (s. f.). Interruptores magnéticos reed. Recuperado de http://www.mercado-ideal.com/catalogoss/SCHMERSAL%20INTERRUPTORES%20MAGNETICOS%20REED.pdf

Simulink. (2013). Versión R2013b [software]. Massachusetts, Estados Unidos: Mathworks.

Unicrom. (2016). Display de 7 segmentos. Recuperado de https://unicrom.com/display-de-7-segmentos/