UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

DESARROLLO E IMPLANTACIÓN DE UNA PLACA DE TEST GESTIONADA POR ARDUINO

Autor:

Abad Carpintero, Borja

Tutor:

Zamarreño Cosme, Jesús María

Dpto. Ingeniería de Sistemas y Automática

Valladolid, Julio 2015.

RESUMEN

La placa desarrollada en este proyecto es un dispositivo electrónico que permite al usuario interactuar con los diferentes elementos instalados en ella mediante un programa diseñado en LabVIEW, todo ello gestionado por la tarjeta Arduino UNO.

Los componentes instalados en la placa son:

- Un LED
- Un LED RGB
- Un pulsador
- Un termistor
- Una fotorresistencia
- Un ventilador

El programa desarrollado controla cada uno de estos elementos, ya sea de forma individual o como un conjunto formado por varios de ellos involucrados en alguna función más compleja. A su vez, se visualizan todas las acciones que se llevan a cabo mediante una placa virtual situada en la misma pantalla del panel de control principal.

Por último, se crea una aplicación portable del programa que permite controlar este dispositivo hardware en cualquier ordenador sin la necesidad de tener instalado LabVIEW en él.

Palabras clave

- **LabVIEW**
- **Microcontrolador**
- **Test**
- **Arduino**
- **Tarjeta**

ÍNDICE

1. Introducción 7

1.1. Introducción 7

1.2. Objetivo del Trabajo Fin de Grado 8

1.3. Estructura de la memoria 9

2. Arduino UNO 11

2.1. ¿Qué es Arduino? 11

2.2. Su origen y evolución 12

2.3. Características del Arduino UNO 13

2.4. El Entorno de Desarrollo Integrado de Arduino 14

2.5. Posibles lenguajes de programación 18

2.6. Usos de Arduino 19

2.7. Conclusiones 23

3. LabVIEW 25

3.1. Introducción a LabVIEW 25

3.1.1. ¿Cómo se trabaja en LabVIEW? 26

3.1.2. Ejemplo sencillo 28

3.1.3. Creación de un SubVI 30

3.2. Comunicación con Arduino UNO 32

3.3. LabVIEW Interface For Arduino (LIFA) 33

3.3.1. Funciones (SubVIs) 34

4. Desarrollo de la placa de test 39

4.1. Elementos individuales de control 39

4.1.1. LED 39

4.1.2. LED RGB 41

4.1.3. Pulsador 44

4.1.4. Termistor 45

4.1.5. Fotorresistencia (LDR) 47

4.1.6. Ventilador 49

4.2. Programa general de la placa de test 51
4.2.1. Descripción de las funciones de la placa de pruebas....51
4.2.2. Panel frontal....52
4.2.3. Diagrama de bloques....54
4.3. Diseño del circuito impreso....61
4.4. Instalación de la placa de test....64

5. Estudio económico 65
5.1. Introducción....65
5.2. Costes directos....65
5.2.1. Costes de personal....66
5.2.2. Costes de materiales....68
5.2.3. Costes de amortización de equipos y programas....69
5.2.4. Costes directos totales....72
5.3. Costes indirectos....73
5.4. Costes totales....73

6. Conclusiones....75
6.1. Conclusiones 75
6.2. Líneas futuras 76

Bibliografía....79

Apéndice A.
Diagramas de conexión 83

Apéndice B.
Fabricación de una placa de circuito impreso 89

Apéndice C.
Creación del software de distribución....93

CAPÍTULO 1
INTRODUCCIÓN

1.1 INTRODUCCIÓN

El presente Trabajo Fin de Grado se ha realizado como ejercicio de integración de los conocimientos adquiridos durante el curso de adaptación al Grado en Ingeniería en Electrónica Industrial y Automática. Dicho curso ha sido presentado por la Universidad de Valladolid para aquellos alumnos o titulados del plan antiguo que quieran cambiarse o actualizarse al nuevo plan adaptado al Espacio Europeo de Educación Superior (EEES).

Se utilizará como punto de partida el trabajo desarrollado previamente por ***Paul Fouet***, del *Institut Universitaire et Technologique de Poitiers*, durante su estancia en la Universidad de Valladolid bajo la dirección de Jesús María Zamarreño Cosme, actual tutor de este proyecto.

Además de emplear algunos de los elementos y programas de funcionamiento básico del trabajo anterior, se han añadido nuevos componentes y funciones que permiten una interacción más completa, obteniendo en algunos casos un conjunto sensor-indicador-actuador.

Como elemento integrador de todos los componentes empleados y sus respectivos programas de control, se ha diseñado y construido un prototipo final del sistema.

1.2 OBJETIVO DEL TRABAJO FIN DE GRADO

La finalidad de este trabajo consiste en desarrollar un sistema hardware montado en una placa con sensores y actuadores típicos, y el sistema Arduino UNO de forma que se puedan realizar pruebas de demostración a través del entorno de programación LabVIEW.

La placa hardware constará de los siguientes elementos:

- Un LED
- Un LED RGB
- Un pulsador
- Un termistor
- Una fotorresistencia (LDR)
- Un ventilador

También se desarrollará un programa que permita la interacción con todos estos dispositivos, ya sea de forma individual, como por ejemplo el *LED* o el *pulsador*, como de forma agrupada, creando los conjuntos *termistor/LED RGB/ventilador* y *LDR/LED RGB*.

A su vez, se visualizará el comportamiento de los elementos de la placa, de tal forma que la actividad llevada a cabo por cada uno de ellos se represente y monitorice en una placa virtual situada en la misma pantalla del panel de control principal de LabVIEW.

Por último, se creará una aplicación de este programa que pueda ser instalada en cualquier ordenador basado en un sistema operativo Windows sin necesidad de tener instalada la plataforma de desarrollo LabVIEW, logrando así una mayor portabilidad del proyecto.

1.3 ESTRUCTURA DE LA MEMORIA

El proyecto está organizado en seis capítulos incluyendo éste, además de tres apéndices como información complementaria al mismo.

En el *Capítulo 2*, se analiza la placa **Arduino**, su entorno de desarrollo integrado (IDE), las características técnicas del Arduino UNO, además de su versatilidad para trabajar con diferentes lenguajes de programación, y su facilidad para la creación de proyectos.

El *Capítulo 3* está dedicado a **LabVIEW**. En él se comentan algunas de sus características y ventajas, que hacen de este programa un entorno de desarrollo revolucionario. También se describe un pequeño ejemplo y las pautas para crear un *SubVI*. Por último, se indican los pasos y programas necesarios para una correcta comunicación entre LabVIEW y Arduino UNO, además de una explicación de los bloques que podemos encontrar en la librería LIFA (LabVIEW Interface For Arduino).

El *Capítulo 4* se centra en la aplicación de forma conjunta de los capítulos anteriores sobre una **placa de test**. Se describen individualmente los diagramas de conexión de cada componente, su programa de control en LabVIEW y las funciones específicamente diseñadas para su uso más sencillo. También se explica el programa de control general de esta placa de pruebas, el panel frontal y las funciones que puede realizar a través de su diagrama de bloques. Además, se muestra el programa **Fritzing** como una herramienta capaz de proporcionar el circuito impreso necesario que nos permite pasar nuestro circuito prototipo a un producto final. Para terminar este capítulo, se detallan los pasos a seguir para la correcta **instalación** de la placa de test en cualquier ordenador.

En el *Capítulo 5* se realiza el **estudio económico** y se describen todos los gastos que afectan a la elaboración del proyecto.

Para finalizar, en el *Capítulo 6* se destacan las **conclusiones** más relevantes y se propone una serie de **trabajos futuros** para distintas líneas del proyecto.

Apéndice A, en él se muestran los **circuitos de conexión** individuales de todos los componentes que forman esta placa de test, así como el circuito impreso para su revelado y soldadura.

El *Apéndice B* describe el proceso de **fabricación de una placa de circuito impreso** y los materiales que se necesitan. También indica los pasos que se deben dar para realizar un correcta soldadura de componentes.

Apéndice C, este último apéndice es una guía sobre la creación del software necesario para **distribuir y ejecutar aplicaciones** diseñadas en LabVIEW en cualquier ordenador.

CAPÍTULO 2

ARDUINO UNO

2.1 ¿QUÉ ES ARDUINO?

Arduino es una plataforma de hardware libre, basado en una placa con un microcontrolador *Atmel AVR*, varios puertos de entrada/salida y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

Puede ser utilizado para desarrollar objetos autónomos e interactivos, como prototipos o interactuar con software instalado en el ordenador. Dada su rápida curva de aprendizaje y su precio económico, es ideal para educadores, diseñadores y cualquier persona interesada en la electrónica y robótica.

Una vez cargado el proyecto en el microcontrolador, éste puede ejecutarse sin necesidad de tener conectado el Arduino al ordenador, sólo sería necesario proporcionarle una fuente de alimentación, ya sea mediante un adaptador de corriente o con baterías.

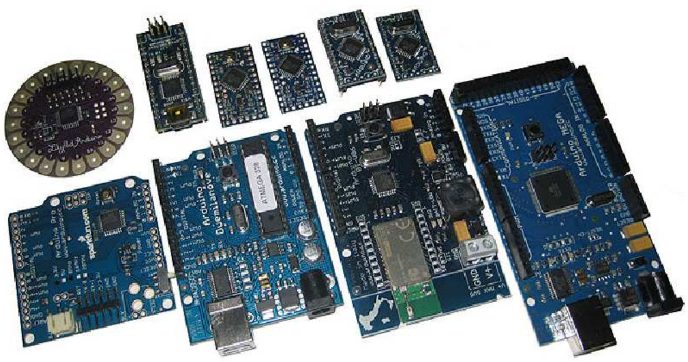


Fig.1 Gama de placas desarrolladas por Arduino

2.2 SU ORIGEN Y EVOLUCIÓN

La historia de Arduino comienza en Italia, en un instituto dedicado a la enseñanza de diseño interactivo en la ciudad de Ivrea, donde *Massimo Banzi*, uno de sus docentes, se propuso diseñar su propia placa de hardware para trabajar con sus estudiantes, ya que las disponibles en el mercado estaban a precios prohibitivos.

Los fundadores del proyecto, *Massimo Banzi* y *David Cuartielles*, junto con otros colaboradores, decidieron publicar sus avances en la red, liberándolo como software y hardware libre. También se inclinaron por los microcontroladores de Atmel, que si bien son el elemento de hardware no-libre dentro de la placa, el fabricante publica sus librerías de compilación libres, de tal forma que resulta fácil para la comunidad portarlas a diferentes sistemas operativos.

Debido a su gran acogida por parte de la comunidad de desarrolladores, Arduino ha seguido creciendo, y actualmente posee una gran cantidad de productos, tanto placas como módulos y accesorios.

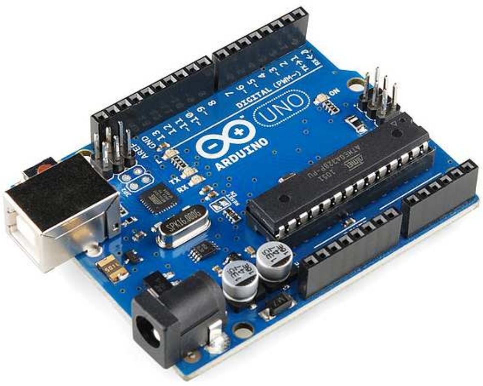
Placas:

- Arduino Uno
- Arduino Due
- Arduino Tre
- Arduino Yún
- Arduino Leonardo
- Arduino Micro
- Arduino Robot
- Arduino Esplora
- Arduino Mega 2560
- Arduino Mega ADK
- Arduino Ethernet
- Arduino Mini
- LilyPad Arduino
- Arduino Nano
- Arduino Pro
- Arduino Pro Mini
- Arduino Fio

Módulos:

- GSM
- Ethernet
- WiFi
- Motor
- Wireless SD

2.3 CARACTERÍSTICAS DEL ARDUINO UNO



El **Arduino UNO** es una placa microcontrolada basada en el ATmega328. Tiene 14 Entradas/Salidas digitales (6 de ellas pueden emplearse como salidas PWM), 6 entradas analógicas, conector USB, clavija hembra tipo Jack, conector ICSP (In-Circuit Serial Programming) y botón de reset. La posición de todos estos elementos queda reflejada en la imagen de la Fig.3.

Si se elige alimentar la placa con una fuente externa, en vez de la conexión USB, el adaptador debe tener un conector de centro positivo y diámetro de 2.1mm para poder conectarlo a la clavija tipo Jack como se muestra en la Fig.2:

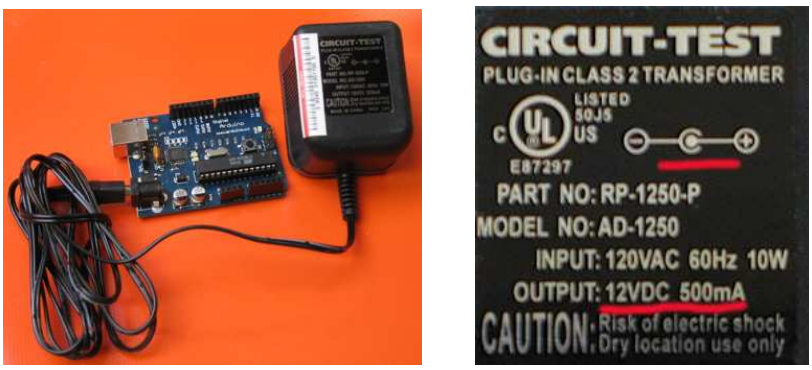


Fig.2 Arduino UNO conectado a una fuente externa.
Detalle de las características de la fuente de alimentación

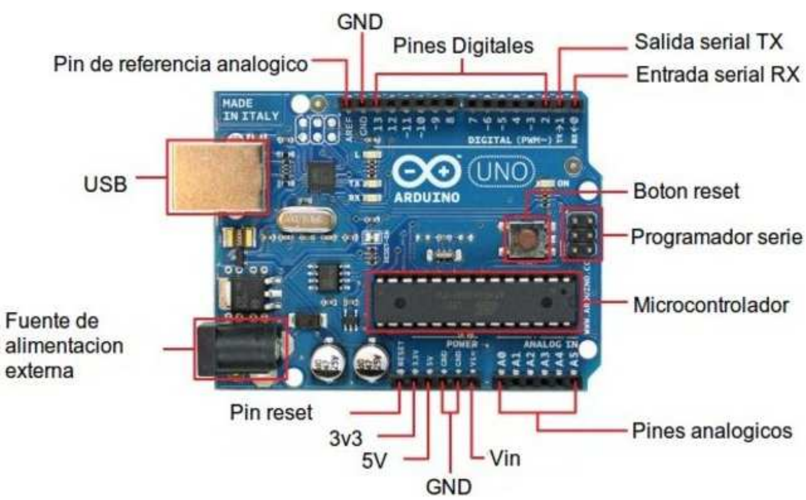


Fig.3 Situación de los componentes en la tarjeta

- **Especificaciones técnicas del Arduino UNO**

Microcontrolador	ATmega328
Voltaje de funcionamiento	5V
Alimentación (recomendada)	7-12V
Voltaje de entrada (límites)	6-20V
Pines digitales I/O	14 (de los cuales 6 dan salida PWM)
Pines de entrada analógica	6
Intensidad de corriente por cada Pin I/O	40 mA
Intensidad de corriente para el Pin 3.3V	50 mA
Memoria Flash	32 KB (ATmega328) 0.5 KB usados por el bootloader
SRAM	2 KB
EEPROM	1 KB
Velocidad del reloj	16 MHz

2.4 EL ENTORNO DE DESARROLLO INTEGRADO DE ARDUINO

El IDE de Arduino es una aplicación multi-plataforma escrita en *Java*, basado en los entornos de desarrollo utilizados en los lenguajes de programación *Processing* y *Wiring*.

Según se aprecia en la Fig.4, el IDE está constituido por un editor de texto para escribir el código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús.

Un programa para Arduino escrito en el editor de texto se denomina “***sketch***”, y puede ser compilado y cargado a la placa rápidamente pulsando un botón. En el área de mensajes se muestra información mientras se cargan los programas o los posibles errores que surjan. La barra de herramientas permite verificar el proceso de carga, creación, apertura y guardado de programas, y la monitorización serie.

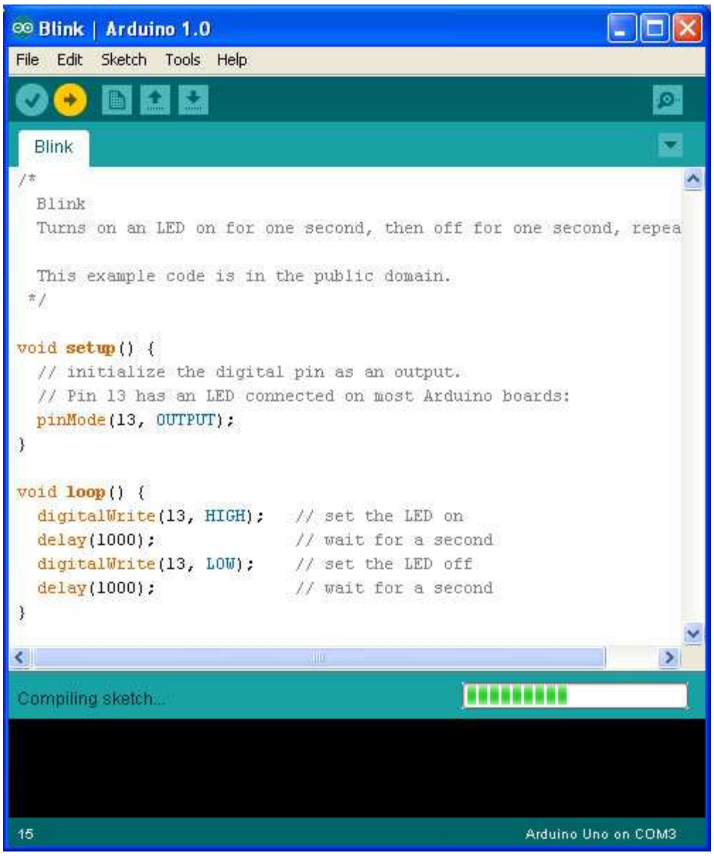


Fig.4 Entorno de Desarrollo Integrado de Arduino

Arduino está basado en ***C*** y soporta todas las funciones del estándar ***C*** y algunas de ***C++***. Los usuarios sólo necesitan definir dos funciones para construir un programa que se ejecute de forma cíclica:

- **setup ( )**: la función se ejecuta una única vez al inicio del programa para determinar la configuración de la placa Arduino.

- **loop ( )**: esta función es llamada de forma repetitiva hasta que la placa se desconecta.

Cualquier programa de Arduino podría no ser reconocido por un compilador estándar de C++ como un programa válido, por eso cuando el usuario presiona el botón “*Cargar*“ del IDE, una copia del código se escribe en un archivo temporal con una cabecera *#include* al inicio y una función *main()* al final, para convertirlo en un programa válido de C++.

➢ FUNCIONES BÁSICAS Y OPERADORES

- **Sintaxis básica**

<table>
<tr><td>Delimitadores</td><td>; { }</td></tr>
<tr><td>Comentarios</td><td>// /* */</td></tr>
<tr><td>Cabeceras</td><td>#define #include</td></tr>
<tr><td>Operadores Aritméticos</td><td>+ - * / %</td></tr>
<tr><td>Asignación</td><td>=</td></tr>
<tr><td>Operadores de comparación</td><td>== != < > <= >=</td></tr>
<tr><td>Operadores Booleanos</td><td>&& || !</td></tr>
<tr><td>Operadores de acceso a punteros</td><td>* &</td></tr>
<tr><td>Operadores de bits</td><td>& | ^ ~ << >></td></tr>
<tr><td>Operadores compuestos</td><td>++ -- += -= *= /= &= !=</td></tr>
</table>

- **Estructuras de control**

<table>
<tr><td>Condicionales</td><td>if , if…else , switch…case</td></tr>
<tr><td>Bucles</td><td>for , while , do…while</td></tr>
<tr><td>Bifurcaciones y saltos</td><td>break , continue , return , goto</td></tr>
</table>

- **Variables**

  Su uso sigue las mismas líneas que se utilizan para el lenguaje C.

- **Constantes**
  - *HIGH/LOW*: representan los niveles alto y bajo de las señales de entrada y salida. Los niveles altos son aquellos de 3 voltios o más.
  - *INPUT/OUTPUT*: entrada o salida.
  - *false*: señal que representa el cero lógico o falso.
  - *true*: señal que indica que cualquier número entero diferente de cero es verdadero.

- **Tipos de datos**
  - void, boolean, char, unsigned char, byte, int, unsigned int, word, long, unsigned long, float, double, string, array.

- **Funciones básicas**

E/S Digital	pinMode(pin, modo) digitalWrite(pin, valor) int digitalRead(pin)
E/S Analógica	analogReference(tipo) int analogRead(pin) analogWrite(pin, valor)
Tiempo	delay(ms) dealyMicroseconds(microsegundos)
Matemáticas	min(x, y) max(x, y) abs(x) sqrt(x) constrain(x, a, b)
Trigonometría	sin(rad) cos(rad) tan(rad)
Números aleatorios	randomSeed(semilla) long random(max) long random(min, max)
Bits y Bytes	lowByte() highByte() bitRead() bitWrite() bitSet() bitClear() bit()

Podemos encontrar más información sobre los comandos y las librerías de Arduino en el siguiente enlace:

http://arduino.cc/en/Reference/HomePage

2.5 POSIBLES LENGUAJES DE PROGRAMACIÓN

La plataforma Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel *Processing*. Sin embargo, es posible utilizar otros lenguajes populares, debido a que Arduino usa la transmisión serial de datos soportada por la mayoría de los lenguajes que se mencionan a continuación. Para aquellos que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida.

Algunos lenguajes de programación permitidos son los siguientes:

- Adobe Director
- C
- C++
- C#
- Cocoa/Objetive-C
- Flash
- Java
- LabVIEW
- Liberlab
- Mathematica
- MATLAB/Simulink
- Perl
- Php
- Physical Etoys
- Processing
- Pure Data
- Python
- Ruby
- VBScript
- Visual Basic .NET

2.6 USOS DE ARDUINO

La tarjeta Arduino se creó para artistas, diseñadores, aficionados y cualquier interesado en crear entornos u objetos interactivos mediante el desarrollo de un prototipo basado en software y hardware flexibles y fáciles de utilizar. Debido a esta simplicidad de creación, podemos encontrar innumerables diseños y usos tanto a nivel artístico como industrial, todos ellos realizados por personas aficionadas que ven en Arduino una forma de aprender y expresar sus inquietudes.

A continuación se muestran algunos ejemplos comunes:

- **ARDUINO COMO ELEMENTO CONECTADO A PC**

  - ***Control de velocidad de un motor de cc***

El ejemplo de la Fig.5, es un ejemplo básico de control, donde se utiliza un pequeño motor de corriente continua al que iremos variando su velocidad de giro mediante el ordenador, empleando para ello una de las salidas PWM de las que dispone Arduino UNO.

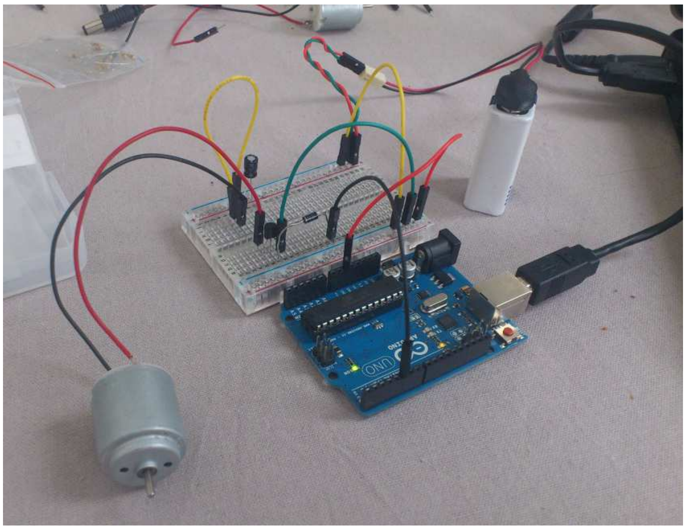


Fig.5 Control de velocidad de un motor de cc

  - ***Control de luces dinámicas***

En la Fig.6 se muestra el funcionamiento sobre el encendido/apagado de varios didos LED, todo ello gestionado con una tarjeta Arduino según las entradas proporcionadas por el ordenador.

Este ejemplo se puede aplicar en cualquier espectáculo visual donde las luces vayan creando un entorno cambiante a gusto del artista.

Fig.6 Control de luces dinámicas

- ***Representación de información en un display***

Otro ejemplo muy común, es la utilización de una pantalla LCD para mostrar cualquier mensaje que el usuario desee. Para ello se deberá incluir en el *sketch* de Arduino alguna librería de control de displays disponibles en Internet totalmente gratuitas. De esta forma, se podrá representar cualquier tipo de información que el usuario introduzca en el ordenador, como se muestra en la Fig.7.

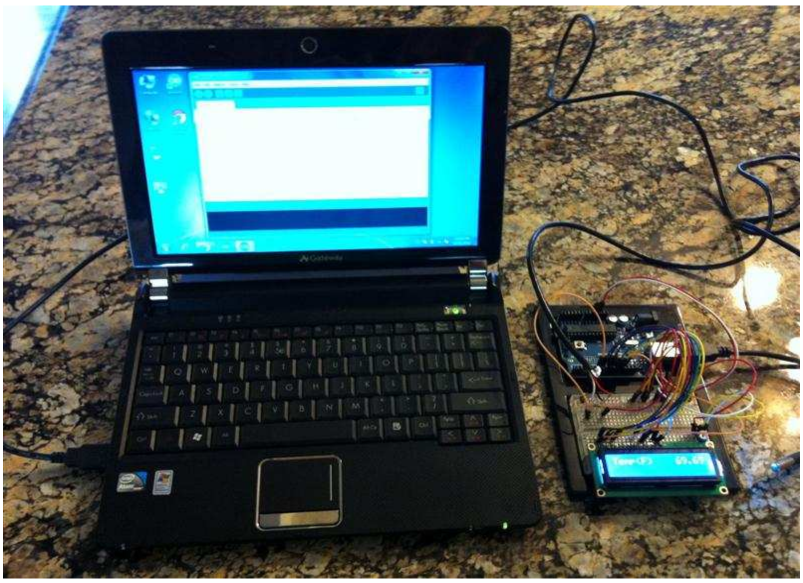


Fig.7 Representación de información en un display

➢ ARDUINO COMO ELEMENTO AUTÓNOMO

o *Control autónomo de Robots*

Cualquier persona aficionada a la electrónica o la automática ha imaginado crear su propio robot, un sistema capaz de moverse de forma autónoma salvando los obstáculos de su entorno, y esto ya es posible gracias a Arduino. En Internet se puede encontrar una gran variedad de modelos, uno de ellos es el representado en la Fig.8, en el cual, observamos que el robot está formado por un sensor de ultrasonidos para evitar los obstáculos y una placa destinada al control de movimiento de los servomotores, todo ello integrado en un shield acoplado a la tarjeta Arduino UNO.

Fig.8 Control autónomo de Robots

o *Sensor de movimiento sin cables*

Con Arduino se puede construir un sistema automático de seguridad para instalar en casa sin necesidad de gastar mucho dinero en sus componentes.

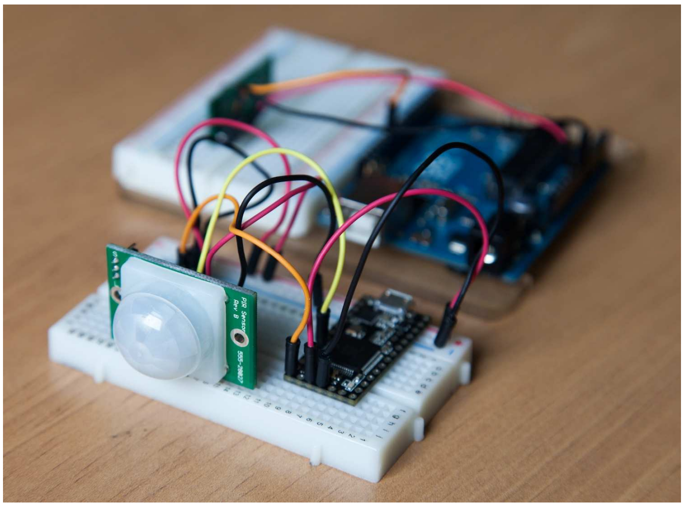


Fig.9 Sensor de movimiento sin cables

Como se aprecia en la Fig.9, se necesitaría un sensor de movimiento IR, un módulo RF para transmitir y recibir los datos del sensor, y dos tarjetas Arduino en donde se escribirá el código necesario para controlar toda la información. De esta forma, se consigue un sistema de seguridad barato, prácticamente sin excesivos cables y totalmente funcional.

- ***Cubo de LEDs***

El ejemplo de la Fig.10 es uno de los más vistosos y conocidos que se puede desarrollar con Arduino. Se basa en una matriz de diodos LED, monocolor o RGB, que se van encendiendo según el programa cargado en la tarjeta, creando juegos de luces y formas.

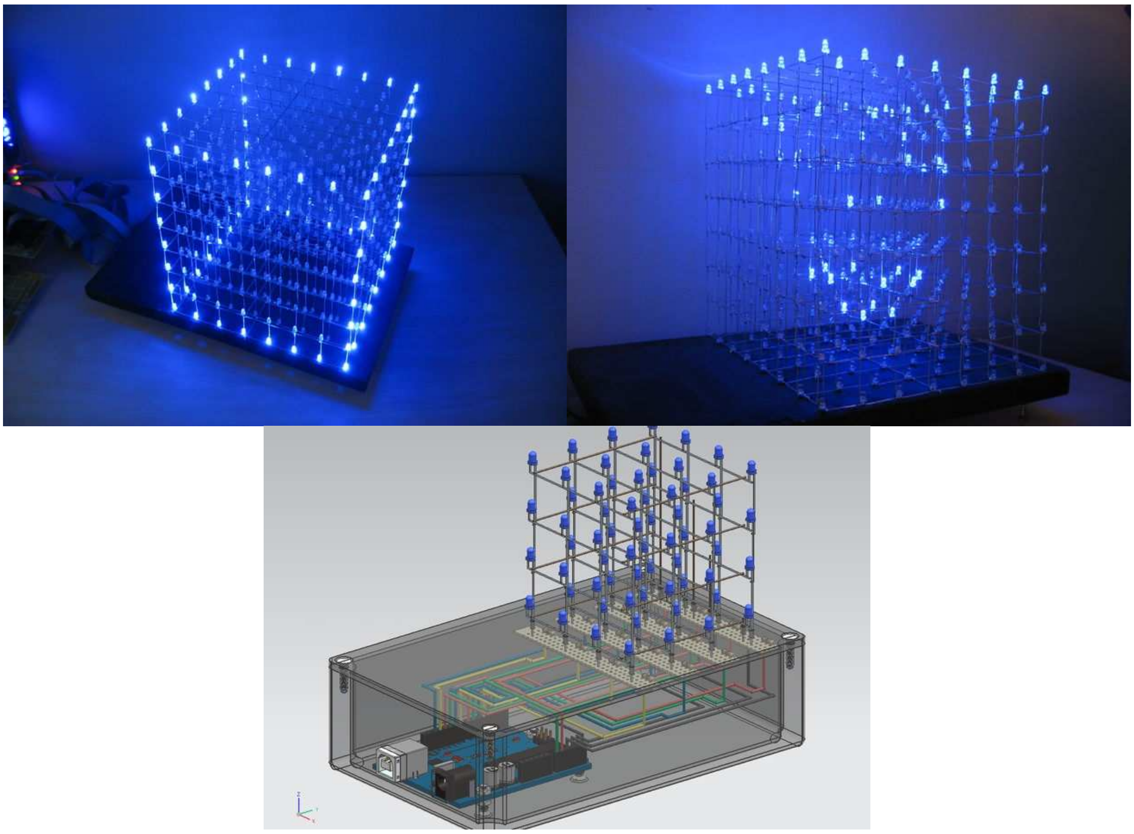


Fig.10 Cubo de diodos LED

2.7 CONCLUSIONES

A la vista de todo lo explicado en este capítulo, podemos decir que la tarjeta Arduino posee unas características idóneas para emprender cualquier proyecto relacionado con la electrónica, la electricidad y elementos hardware. Su bajo precio facilita la adquisición de este dispositivo a todas aquellas personas que tengan interés en la tecnología, ya sea con carácter docente o meramente un simple pasatiempo; además permite una programación con diversos lenguajes, aumentando así sus posibilidades de ser utilizado en distintas áreas.

Su gran versatilidad y el carácter libre de sus librerías, permiten a aficionados y expertos crear una infinidad de proyectos que posteriormente son publicados en internet; siendo el foro oficial de Arduino (http://forum.arduino.cc/) el lugar donde se puede obtener más información acerca de su potencial, y cualquier duda será resuelta por la comunidad.

Por todo esto, y por su interés en seguir ofreciendo nuevos productos que amplían su gama de tarjetas de forma continua, Arduino se ha convertido en la marca de referencia para todos aquellos amantes de la tecnología con espíritu creativo que quieran diseñar y construir sus propios dispositivos de forma fácil y sin gastar mucho dinero.

CAPÍTULO 3

LABVIEW

3.1 INTRODUCCIÓN A LABVIEW

LabVIEW (*Laboratory Virtual Instrumentation Engineering Workbench*) es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje revolucionario de programación gráfica. Este programa fue creado por ***National Instruments*** para aplicaciones que involucren adquisición, control, análisis y presentación de datos.

Las ventajas que proporciona el empleo de LabVIEW se resumen en las siguientes:

- Se reduce el tiempo de desarrollo de las aplicaciones al menos de 4 a 10 veces, ya que es muy intuitivo y fácil de aprender.
- Dota de gran flexibilidad al sistema, permitiendo cambios y actualizaciones tanto del hardware como del software.
- Da la posibilidad a los usuarios de crear soluciones completas y complejas.
- Con un único sistema de desarrollo se integran las funciones de adquisición, análisis y presentación de datos.
- Tiene la posibilidad de incorporar aplicaciones escritas en otros lenguajes.

LabVIEW se diferencia de los programas de desarrollo de aplicaciones comerciales en un importante aspecto; los lenguajes tradicionales de programación se basan en líneas de texto para crear el código fuente del programa, mientras que LabVIEW emplea la programación gráfica o ***lenguaje G*** para crear programas basados en diagramas de bloques.

Al apoyarse sobre símbolos gráficos en lugar de lenguaje escrito, resulta más intuitivo a la hora de realizar los programas, otorgando una gran facilidad de uso tanto a programadores profesionales como a personas con menores conocimientos en programación.

3.1.1 ¿Cómo se trabaja en LabVIEW?

Al ser una herramienta gráfica de programación, los programas no se escriben, sino que se dibujan, facilitando su comprensión. Como tiene una gran cantidad de bloques pre-diseñados, hace más sencilla la creación del proyecto, con lo que el usuario invierte mucho menos tiempo en programar un dispositivo/bloque y puede dedicarse un poco más en la interfaz gráfica y la interacción con el usuario final.

Los programas creados mediante LabVIEW se denominan *Instrumentos Virtuales (**VIs**)*, porque su apariencia y funcionamiento imitan los de un instrumento real. Sin embargo son análogos a las funciones creadas con los lenguajes de programación convencionales. En la Fig.11 se puede apreciar que cada *VI* consta de dos partes diferenciadas:

- ***Panel Frontal:*** se trata de la interfaz gráfica del *VI* con el usuario. Se utiliza para interactuar con él cuando el programa se está ejecutando, de forma que pueda observar los datos actualizados en tiempo real. En esta interfaz se definen los *controles* (se usan como entradas) e *indicadores* (salidas).

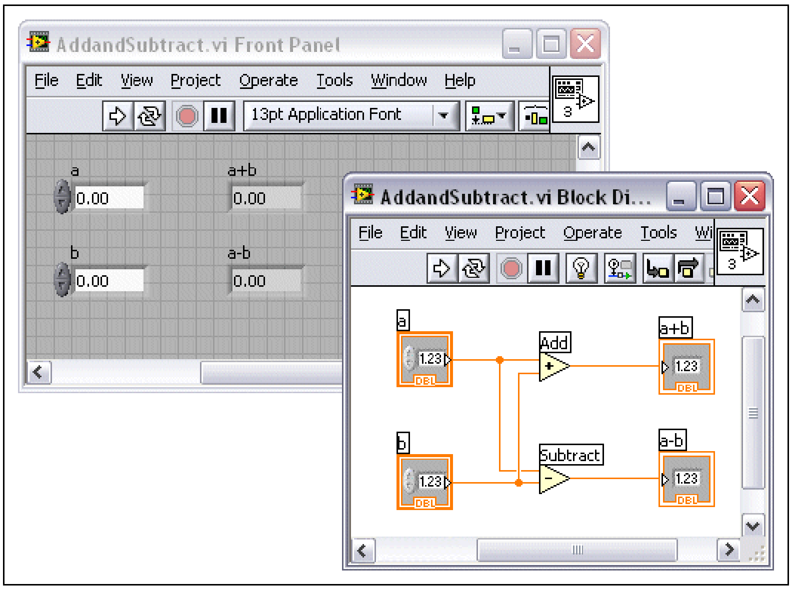


Fig.11 Panel de Control y Diagrama de bloques

- ***Diagrama de Bloques:*** constituye el código fuente del *VI*. Aquí es donde se realiza la implementación del programa del *VI* para controlar o realizar cualquier procesado de las entradas y salidas que se crearon en el Panel Frontal. Los controles e indicadores que se colocaron previamente, se materializan en el Diagrama de Bloques mediante los *terminales*. Uniendo todos estos distintos elementos entre sí, se construye el circuito del programa.

Las **Paletas** nos proporcionan las herramientas para crear y modificar todos los elementos necesarios. Son las siguientes:

- ***Paleta de Controles (Controls Palette), Fig.12:*** se utiliza sólo en el Panel Frontal. Contiene todos los controles e indicadores que se emplearán para crear la interfaz del *VI* con el usuario.

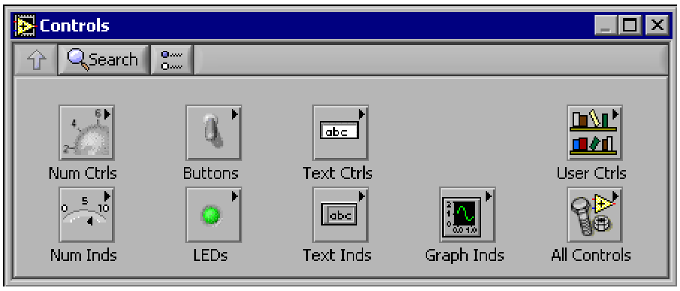


Fig.12 Paleta de Controles

- ***Paleta de Funciones (Functions Palette), Fig.13:*** se usa en el Diagrama de Bloques. Contiene todos los objetos que se emplean en la implementación del programa del *VI*, ya sean funciones aritméticas, de entrada/salida de señales, adquisición de datos, temporización de ejecución del programa, etc.

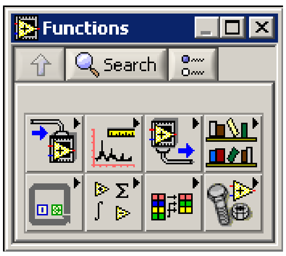


Fig.13 Paleta de Funciones

- ***Paleta de Herramientas (Tools Palette):*** se emplea para operar y modificar objetos tanto en el panel frontal como en el diagrama de bloques.

 *Operating tool:* cambia el valor de los controles.

 *Positioning tool:* desplaza, cambia de tamaño y selecciona objetos.

 *Labeling tool:* edita texto y crea etiquetas.

 *Wiring tool:* une los objetos en el diagrama de bloques.

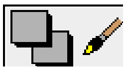
 *Object Pop-up Menu tool:* abre el menú desplegable de un objeto.

 *Scroll tool:* desplaza la pantalla.

 *Breakpoint tool:* fija puntos de interrupción en la ejecución del programa en *VIs*, funciones y estructuras.

 *Probe tool:* crea puntos de prueba en los cables, donde se visualiza el valor en cada instante.

 *Color Copy tool:* copia el color seleccionado.

 *Color tool:* establece el color de fondo y el de los objetos.

3.1.2 Ejemplo sencillo: *conversión de ºC a ºF*

A continuación, se realizará un ejemplo básico de paso de grados centígrados (ºC) a grados Fahrenheit (ºF) para demostrar la sencillez y facilidad de programación que nos ofrece LabVIEW.

Creamos un nuevo documento VI y se nos abrirán dos ventanas a la vez; una es el Panel Frontal, y la otra el Diagrama de Bloques como ya hemos descrito anteriormente. En el Panel Frontal añadiremos un control numérico en forma de puntero deslizante a través de la paleta *Controls >> Numeric Controls*, que será la entrada del valor en ºC que deseamos convertir. También seleccionamos un indicador en la paleta *Controls >> Numeric Indicators*, el cual nos indicará el resultado en ºF. Ambos objetos se crean de forma automática en el Diagrama de Bloques como se aprecia en la Fig.14, lo que nos permite enlazar la entrada y la salida de valores dentro del cuerpo del programa principal.

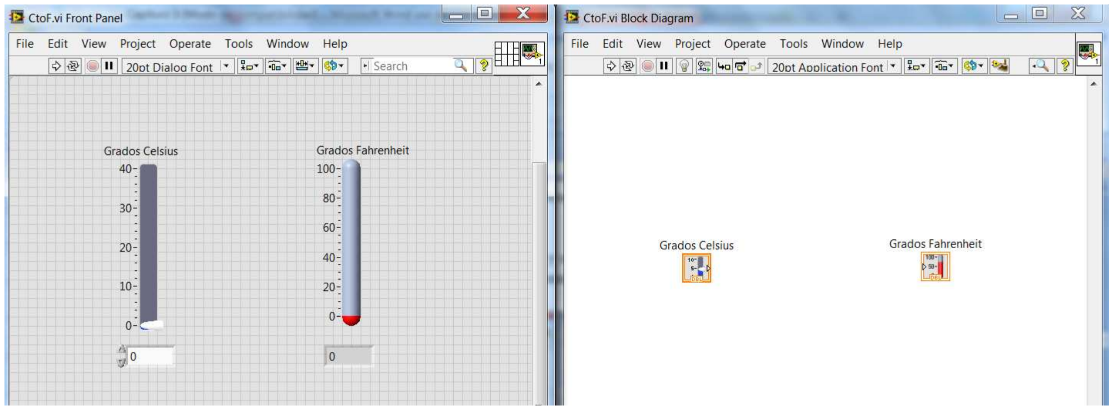


Fig.14 Entrada y salida del programa de conversión de ºC a ºF

Ahora que ya están definidas la entrada y la salida, pasamos a desarrollar el interior del programa. Para convertir ºC a ºF se debe aplicar la siguiente fórmula:

$$\mathbf{T(^{o}F) = (T(^{o}C) \times 1.8) + 32}$$

Dicha fórmula se implementa mediante las funciones matemáticas que encontramos en la paleta *Functions >> Numeric*, y se añaden al Diagrama de Bloques. Por último, definimos las constantes e interconectamos los elementos como se muestra en la Fig.15.

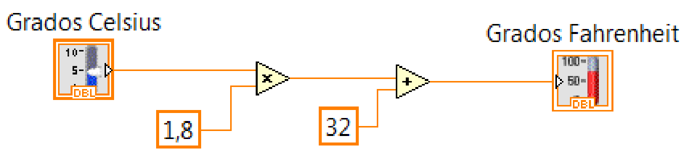


Fig.15 Fórmula de conversión expresada mediante bloques

Para ejecutar el programa de conversión, introducimos un valor de entrada de ºC y pulsamos el botón *Run*  del Panel Frontal. En la Fig.16 se muestra el programa terminado y ejecutado para un valor de entrada de 35 ºC, obteniendo 95 ºF como resultado.

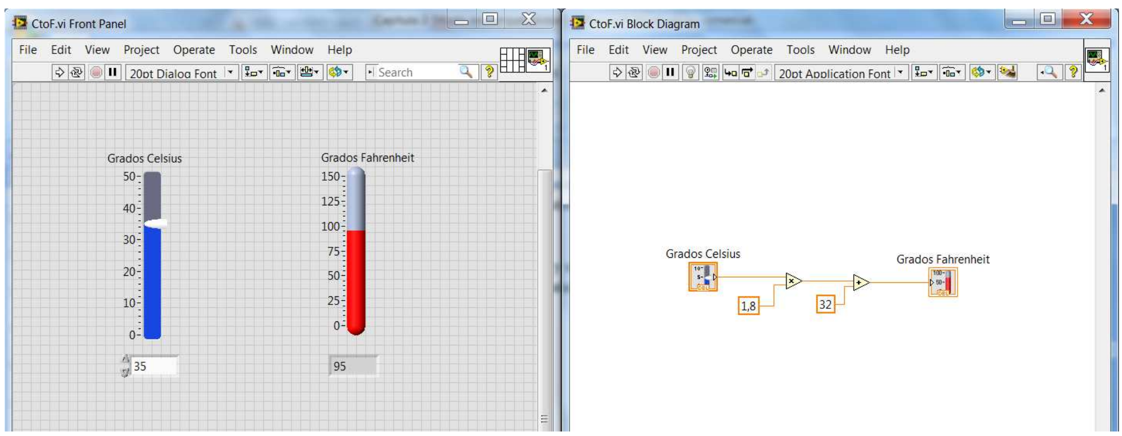


Fig.16 Programa finalizado

3.1.3 Creación de un SubVI

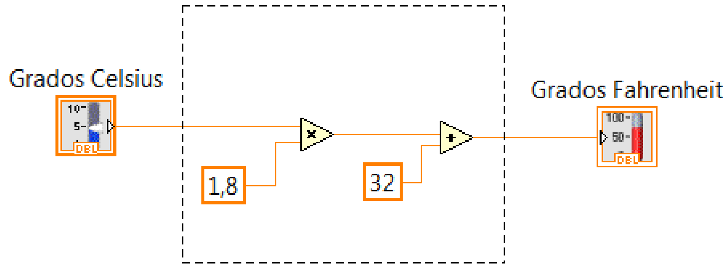
Un *SubVI* es sólo otro *VI* de LabVIEW que puede ser colocado en cualquier lugar de nuestro programa del Diagrama de Bloques, al igual que los elementos de las paletas. Es equivalente a las funciones o métodos de los lenguajes de programación como C++ y Java, con la excepción de que se puede obtener más de una salida. El SubVI tiene **3 partes principales**; el código real (Panel Frontal y Diagrama de Bloques), un icono personalizable, y el panel conector.

Sus **ventajas** son las siguientes:

- Modular.
- Reutilizable.
- Rápida comprobación y eliminación de errores.
- Requiere menos memoria.

Ahora que hemos definido qué es un *SubVI*, pasamos a crear uno basado en el ejemplo del apartado anterior, la conversión de ºC a ºF.

1) Definimos la sección de código del Diagrama de Bloques que queremos convertir en un *SubVI*:



2) Desde la barra de herramientas seleccionamos **Edit** >> **Create SubVI**. La sección elegida es reemplazada por un bloque de una entra y una salida que tiene un icono y un panel conector predeterminados, según se aprecia en la Fig.17.

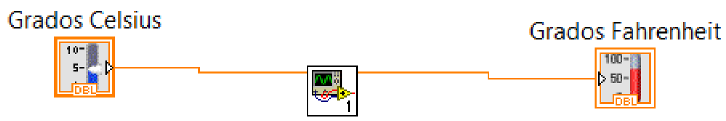


Fig.17 Sustitución de todos los bloques seleccionados por un *SubVI*

3) **Personalizar el icono:** debemos crear un nuevo icono que defina las funciones del *SubVI* utilizando las herramientas del editor. Para ello, pulsamos 2 veces en el bloque, y en la nueva pantalla que aparece, seleccionamos **Edit Icon** haciendo click derecho con el ratón en el icono de la esquina superior derecha. En la Fig.18 se ha creado un icono diferente al predeterminado que indica la operación que realiza el nuevo bloque.

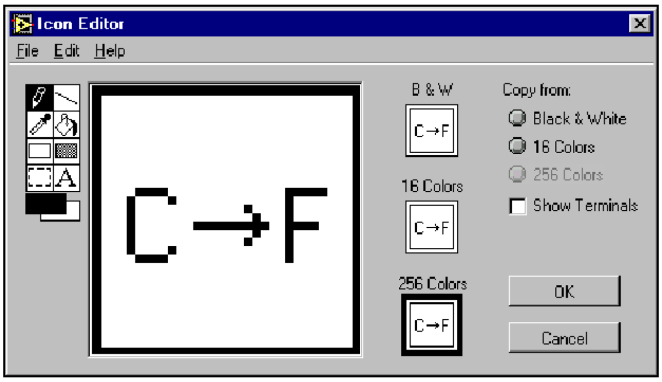


Fig.18 Edición del icono del *SubVI*

4) **Construcción del panel conector:** este panel es una representación visual de cómo las entrada y salidas están conectadas al *SubVI* desde el *VI* de llamada.

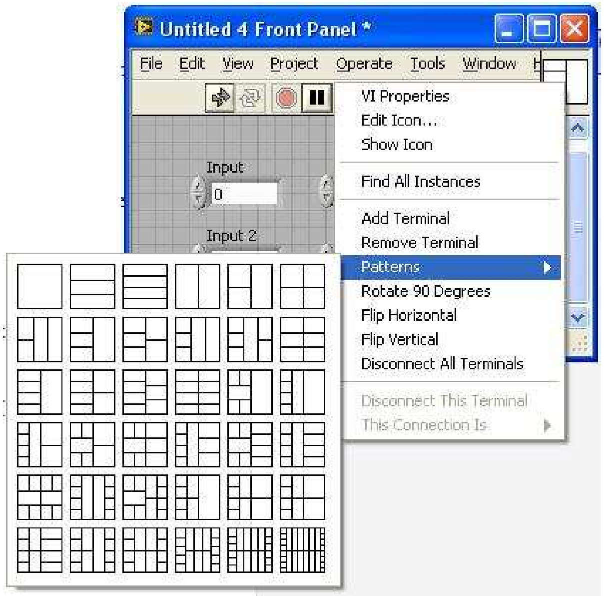


Fig.19 Diferentes patrones del panel conector

Hacemos click derecho en el icono superior derecho del Panel Frontal y seleccionamos **Show Connector**. En la Fig.19 vemos la variedad de patrones que podemos elegir, dependiendo del número de entradas y salidas que tenga el *SubVI*. A continuación, seleccionamos una región en blanco del panel (terminal) y asignamos el control o indicador que se quiera. Por último, guardamos el *SubVI*.

5) **Insertar el *SubVI* dentro de un *VI* principal:** en la paleta de *Funciones*, elegimos la opción **Select a VI**, navegamos hasta donde se haya guardado nuestro *VI* y hacemos doble click para añadirlo al Diagrama de Bloques.

3.2 COMUNICACIÓN CON ARDUINO UNO

Para conectar la placa Arduino con el entorno de trabajo LabVIEW y comenzar a desarrollar cualquier proyecto, se necesitará una *herramienta gratuita* creada específicamente por *National Instruments* llamada **LabVIEW Interface for Arduino (LIFA)**.

Esta herramienta es una API basada en *VIs* que permiten al usuario leer entradas analógicas, controlar las entradas/salidas digitales y usar otras características del hardware de Arduino.

La configuración del *toolkit LIFA* es un proceso de seis pasos, los cuales se describen a continuación:

1) **Instalar LabVIEW**

   Prerrequisito: es necesario tener **LabVIEW 2009 o una versión superior** para que los *VIs* de la herramienta LIFA funcionen correctamente.

2) **Instalar los controladores NI-VISA**

   Para LabVIEW, Arduino aparece como un instrumento conectado a un puerto serie (RS-232), por lo que es necesario instalar la última versión de los drivers de NI-VISA.

3) **Instalar el IDE de Arduino y sus controladores**

   Se pueden descargar gratuitamente de la página oficial de Arduino siguiendo este enlace:

   http://arduino.cc/en/Main/Software

   Si es la primera vez que se conecta el Arduino UNO al ordenador mediante un cable USB, el sistema operativo nos avisará de la existencia de un dispositivo nuevo y deberemos instalar sus controladores para que sea reconocido en las futuras conexiones. Dichos controladores se encuentran en la subcarpeta *drivers*.

4) **Instalar JKI VI Package Manager (VIPM)**

VIPM es un gestor que permite instalar nuevas librerías y herramientas en LabVIEW de forma rápida y sencilla. Se descarga a través de su página oficial:

http://jki.net/vipm/download

5) **Instalar el Interfaz de LabVIEW para Arduino (LIFA)**

Abrimos el VIPM, buscamos la herramienta LIFA en la lista de paquetes y pulsamos el botón *Install Package(s)*.

6) **Cargar el firmware en el Arduino**

- Debemos abrir el IDE de Arduino, que hemos descargado en el paso 3, haciendo doble click en el ejecutable *arduino.exe*.
- Vamos a *File>>Open* y abrimos el archivo ***LIFA_Base.ino*** que se suele encontrar en:

  C:\Program Files\National Instruments\LabVIEW 20xx\vi.lib\LabVIEW Interface for Arduino\Firmware\LIFA_Base
- Seleccionamos la placa Arduino UNO en *Tools>>Board*.
- Indicamos el puerto COM asignado por el ordenador a Arduino UNO en *Tools>>Serial Port*, (el número del puerto puede verse en el *Administrador de dispositivos* del Panel de Control de Windows).
- Para finalizar, pulsamos el botón de *Upload*, para cargar el firmware al Arduino UNO.

Después de estos seis pasos, la comunicación entre LabVIEW y la tarjeta Arduino será perfecta, y podremos empezar a desarrollar nuestros programas rápidamente gracias a los bloques y ejemplos básicos incluidos en LIFA.

3.3 LABVIEW INTERFACE FOR ARDUINO (LIFA)

Es un conjunto de herramientas gratuitas que permiten al usuario controlar y adquirir datos a través del microcontrolador Arduino y procesarlo en el entorno de programación gráfica de LabVIEW.

LIFA proporciona una simple y poderosa API para controlar las E/S Digitales, las salidas PWM, las entradas Analógicas, y las comunicaciones I2C y SPI, permitiendo al desarrollador centrarse en la aplicación en vez de la implementación a bajo nivel.

➢ **Características**

- Comunicación mediante USB, Bluetooth o XBee.
- Velocidades de hasta 200Hz con USB y 50Hz en conexiones inalámbricas.
- Ejemplos para tareas básicas y sensores.
- El firmware y los *VIs* de LabVIEW son de código abierto, permitiendo una completa personalización.

3.3.1 Funciones (SubVIs)

❖ ***INICIO Y CIERRE DE COMUNICACIONES***

Estos módulos son imprescindibles en cualquier programa de Arduino, ya que nos permiten iniciar y cerrar la comunicación con la tarjeta de forma segura, incluso nos avisan si ha surgido algún problema en la transferencia de datos entre el ordenador y Arduino. En la Fig.20 se indican los parámetros que vienen por defecto, así como una breve descripción de cada uno de los módulos proporcionada por la ayuda de LabVIEW.

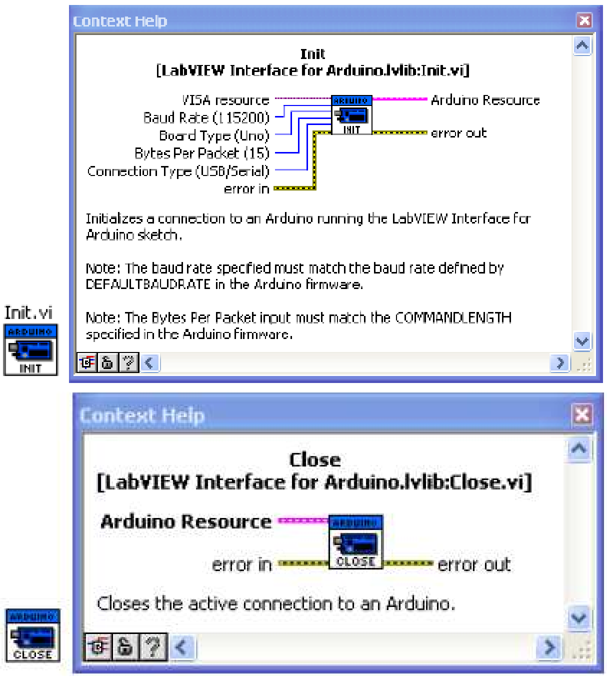


Fig.20 Bloques de inicio y cierre de comunicaciones

❖ *FUNCIONES "LOW LEVEL"*

- **Entradas analógicas**

Lectura de un pin de entrada analógico	
Lectura de todo el puerto de entrada analógico	

- **E/S Digitales**

Configuración de un pin digital como entrada o salida	
Lectura o escritura de un único pin digital	
Lectura o escritura de todo el puerto digital	

- **PWM**

Escritura en un pin de salida PWM	
Configuración y escritura en todos las salidas PWM	

- **Generación de Tono**

Genera una onda cuadrada de frecuencia y duración variables	

- **Inter-Integrated Circuit (I2C)**

Master, escritura y lectura del bus I2C	I2C Init.vi I2C Write.vi I2C Read.vi

- **Serial Peripheral Interface (SPI)**

Master, escritura/lectura y cierre del bus SPI	SPI Init.vi SPI Send Receive.vi SPI Close.vi
Configuración del orden de bits (MSBF, LSBF)	
Soporte para múltiples velocidades de reloj	SPI Set Clock Divider.vi

❖ *FUNCIONES PRE-CONSTRUIDAS PARA SENSORES, MOTORES Y PANTALLAS*

- **Termistor, Fotocélula y sensor IR**

Termistor configurable para leer temperaturas	
Fotocélula para leer intensidades de luz	
Sensor IR de proximidad para detectar objetos	

- **LED RGB**

Configuración de los pin para la salida de cada color	
Escritura del color en cada pin seleccionado	

- **Display de 7 segmentos**

Configuración de los pines de salida	
Escritura de un carácter	
Escritura de un string	

- **Motores paso a paso**

Especificación de una salida para el motor	
Escritura del número de pasos a girar y su velocidad	
Verificación de la finalización del paso indicado antes de iniciar el siguiente	
Bloque de espera hasta que se completan todas las sentencias para el giro	
Fin de la comunicación con el motor	

- **Servomotores**

Definición del número de servomotores conectados	
Asignación de un pin de salida a cada motor definido	
Eliminación del servo asignado a un pin	
Definición del ángulo que debe tomar el servo	
Lectura en grados del último escrito en el motor	
Define el ancho de pulso, en µs, que se va a enviar al servo	
Lectura del último ancho de pulso en µs	

En este apartado se han mostrado las funciones de algunas de las librerías que más se usan habitualmente en cualquier proyecto, pero existen más aquí no comentadas, como por ejemplo BlinkM y LCD, y funciones tanto de configuración como de ayuda. LIFA también posee ejemplos ya construidos para facilitar el desarrollo de nuevos programas.

Todos estos bloques están perfectamente explicados en la propia ayuda de LabVIEW, y es altamente recomendable analizar sus entradas y salidas para saber qué tipos de datos utilizar.

CAPÍTULO 4
DESARROLLO DE LA PLACA DE TEST

Este capítulo está enteramente dedicado al diseño y desarrollo de la placa de test en que se basa el actual Trabajo Fin de Grado, donde se aplican de forma conjunta los elementos vistos en los capítulos anteriores: Arduino UNO y LabVIEW. Una vez finalizado el programa y definido su circuito eléctrico, se indicarán los pasos a seguir y las herramientas necesarias para obtener el circuito impreso o PCB, y así poder replicar dicho proyecto tantas veces como sea necesario.

4.1 ELEMENTOS INDIVIDUALES DE CONTROL

Antes de comenzar a diseñar el programa completo se va a analizar cada elemento de forma individual, tanto a nivel de conexión con Arduino como su control mediante LabVIEW, lo que nos proporcionará una visión más clara e independiente del funcionamiento de cada uno de los componentes, facilitando así la compresión del programa completo que controla la placa de test.

4.1.1 LED

El LED va en serie junto con una resistencia de 470Ω y conectado al *pin 8* de Arduino como se muestra en el Circuito 1 del Apéndice A. No olvidar conectar el cátodo (patilla corta/borde plano) al pin de tierra para cerrar el circuito.

Todos los programas de LabVIEW relacionados con Arduino en este proyecto tienen las mismas **4 fases**: *inicio de la comunicación*, *lazo de control*, *reseteo de variables* y *fin de la comunicación.*

En la Fig.21 se muestra el programa que controla el encendido y apagado del LED y las 4 fases antes mencionadas que pasamos a describir:

- **Inicio de la comunicación**

  El programa detecta el puerto USB donde se ha conectado el Arduino y comienza la transmisión con la tarjeta. Después se definen los pines que vayamos a utilizar, siendo en este caso el pin 8 como salida digital.

- **Lazo de control**

  Este *bucle While* es el que controla el encendido o apagado del LED mediante una señal introducida a través de un control del panel frontal, y se repetirá cada 250ms hasta que el usuario presione el botón de parada. La señal del control es escrita en el pin 8 definido en la etapa anterior como un pin de salida digital.

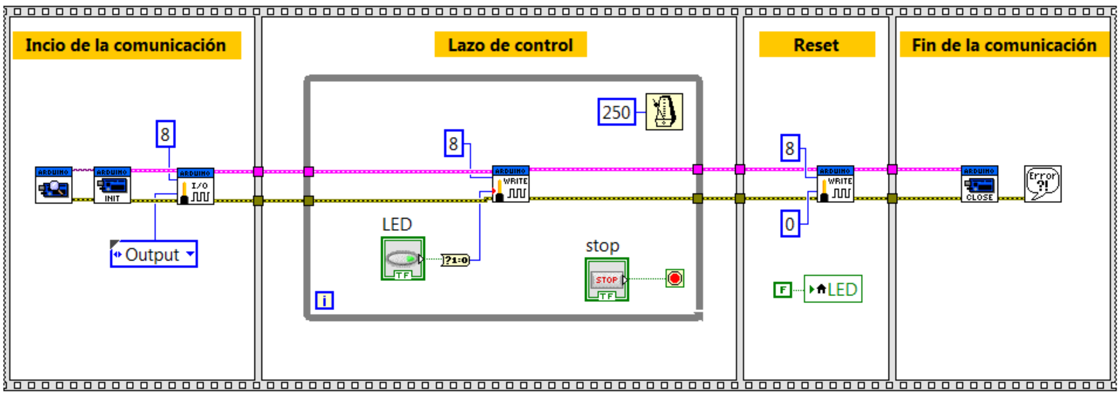


Fig.21 Programa de control del LED

- **Reset**

  Una vez que el usuario decide finalizar el programa saliendo del *bucle While*, se pasa a la etapa de reseteo de elementos. Se envía al pin 8 un valor nulo para que el LED no se mantenga encendido, y se reinicia la variable de entrada.

- **Fin de la comunicación**

  Por último, se cierra la comunicación con Arduino.

Para reducir el número de elementos que se deben utilizar al controlar el LED, se ha creado el *SubVI* de la Fig.12; donde se aprecian las entradas y salidas de comunicaciones y errores, además de las entradas de pin digital y de valor booleano asociado a un control en el panel frontal.

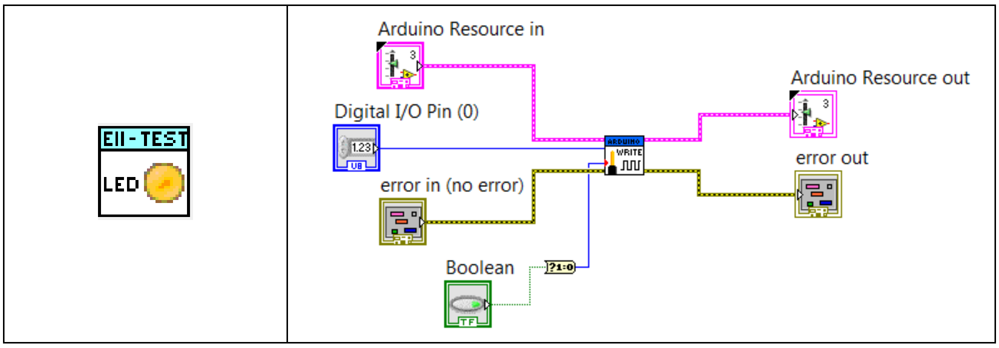


Fig.22 *SubVI* de control del LED. Entradas y salidas de la función

4.1.2 LED RGB

El LED RGB utilizado en este proyecto es el representado en la Fig.23; posee 4 patillas, siendo la más larga el ánodo del dispositivo que debe ir conectado a los 5V que suministra la tarjeta Arduino. El resto de patillas corresponden a los colores **Rojo**, **Azul** y **Verde** según se aprecia en la representación, y cada una debe ir conectada en serie con una resistencia de 470Ω, tal y como se indica en el Circuito 2 del Apéndice A.

Estas patillas de colores se conectarán en los siguientes pines digitales:

- El **rojo** utilizará el *pin PWM 6*.
- El **verde** el *pin PWM 5*.
- El **azul** usará el *pin PWM 3*.

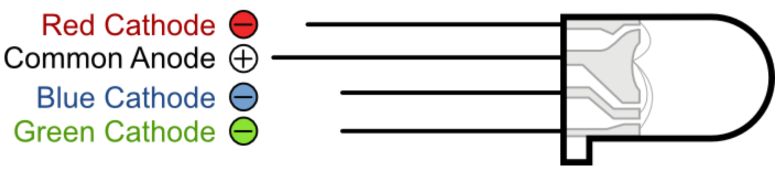


Fig.23 Significado de las patillas del LED RGB de ánodo común

Se emplean estos pines digitales ya que poseen la característica de modificar el ancho de pulso emitido (PWM), es decir, podemos variar el dato de salida para aumentar o reducir el voltaje que se aplica a las patillas del LED RGB y así tener diferentes intensidades lumínicas.

El programa diseñado para controlar este LED es el representado en la Fig.24, donde se mantienen las cuatro fases que se describen a continuación:

- **Inicio de la comunicación**

  Igual que en el elemento anterior, detecta el puerto USB donde está conectado Arduino e inicia la comunicación. Se designan los pines digitales 6, 5 y 3 para los colores rojo, verde y azul respectivamente.

- **Lazo de control**

  Dentro del *bucle While* se le indicará a cada color la intensidad que necesitemos mediante un control deslizante. Al ser un LED de ánodo común, si introducimos un valor de 255 estará apagado, mientras que si es un 0 tendrá máxima intensidad, por lo que se aplicará la operación ***255 – x***, donde ***x*** es el valor de luminosidad que el usuario quiere obtener realmente.

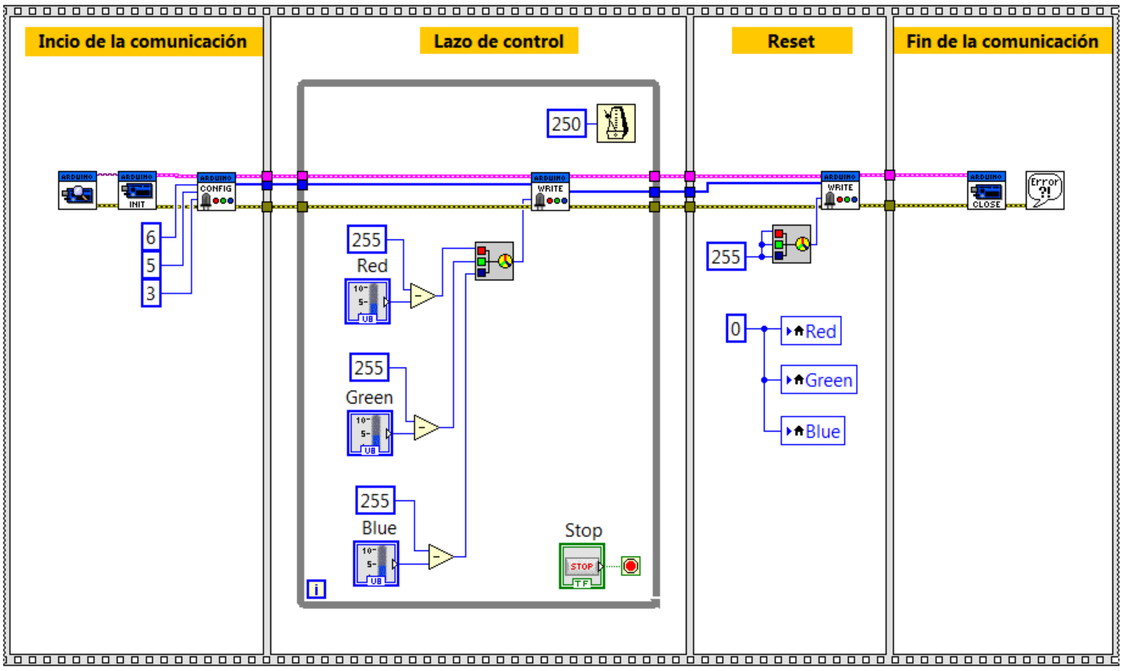


Fig.24 Programa de control del LED RGB

Así, con esta operación, al aumentar la barra deslizadora del control, se corresponderá con una subida de luminosidad en el LED. Posteriormente se transforman los valores de 0 a 255 en el color resultante, y se escriben en los pines asignados a cada color. Estas operaciones se repiten cada 250ms hasta que el usuario decide finalizar el programa.

Nota: si el LED es de ***cátodo común*** deberá tener su patilla más larga conectada a tierra y en su programación no hace falta realizar ninguna operación previa; sólo es necesario introducir un valor de 0 a 255 en el color correspondiente.

- **Reset**

  Después de salir del bucle se apaga el LED enviando un valor de 255 a sus patillas de colores para que no se mantenga encendido al cerrar la comunicación. También se resetea el valor de los controles de selección de color.

- **Fin de la comunicación**

  Se finaliza la comunicación con la tarjeta Arduino.

Con el fin de hacer más sencillo el control de este elemento, se ha diseñado el *SubVI* de la Fig.25, siendo sus entradas y salidas las siguientes:

Entradas: comunicación, errores, pines asociados al LED y los valores entre 0 a 255 correspondientes a los colores rojo, verde y azul.

Salidas: comunicación, errores, pines asociados al LED y color resultante.

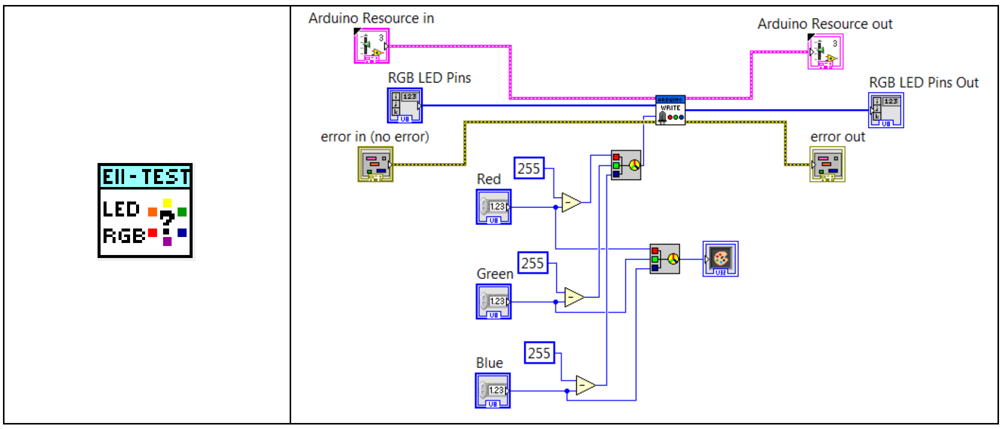


Fig. 25 *SubVI* de control del LED RGB. Entradas y salidas de la función

4.1.3 Pulsador

Se utilizará un pulsador de 4 patillas que recibe una tensión de 5V directamente de la tarjeta Arduino. Su conexión es la representada en el Circuito 3 del Apéndice A, donde el pulsador es alimentado en una de sus patillas, mientras la otra patilla correspondiente al mismo terminal, está conectada en serie a una resistencia de 1kΩ y se une al pin de masa de Arduino para cerrar este lado del circuito. Dicha resistencia es necesaria para evitar que se produzca un cortocircuito cuando el pulsador no está presionado. Por último, se conecta la patilla opuesta a la que recibe los 5V con el *pin digital 2*, el cual detectará la señal de entrada cuando se presione el pulsador.

Para el control de este elemento se ha diseñado el programa de la Fig.26 que pasamos a comentar:

- **Inicio de la comunicación**

  Detecta automáticamente el puerto USB al que se ha conectado Arduino y comienza la comunicación con él. Además, se define el pin 2 como una entrada digital.

- **Lazo de control**

  El *bucle While* se encarga de leer el pin 2 cada 250ms. Si la señal de entrada es diferente de 0V significa que el pulsador está presionado, y por lo tanto, se enciende el indicador del Panel Frontal.

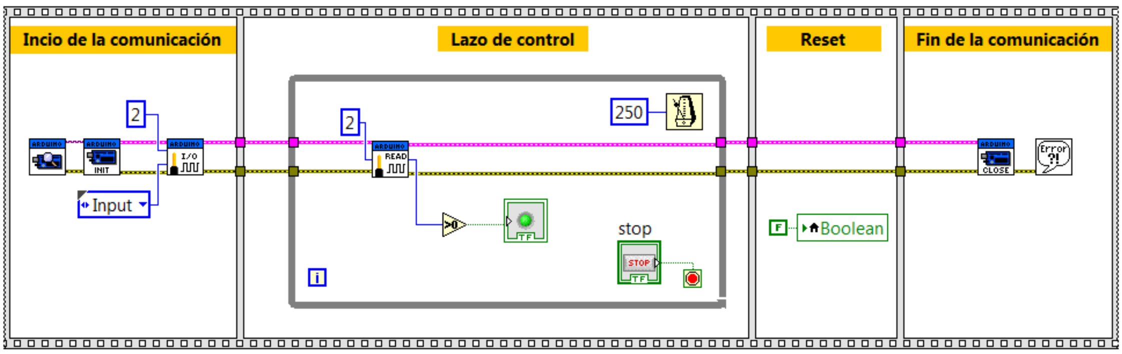


Fig.26 Programa de control del pulsador

- **Reset**

    Cuando se finaliza el programa mediante el botón de parada, se introduce un valor *false* en el indicador del pulsador para que no se mantenga encendido.

- **Fin de la comunicación**

    Se termina la comunicación con Arduino.

Igual que en los elementos anteriores, se ha diseñado un *SubVI* específico tal y como se observa en la Fig.27. Posee entradas y salidas para las comunicaciones y los posibles errores, además de la entrada de pin digital para la lectura y salida booleana que indica cuándo el pulsador está presionado.

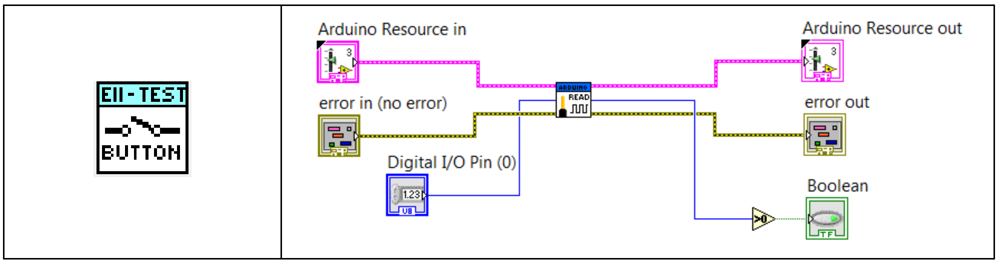
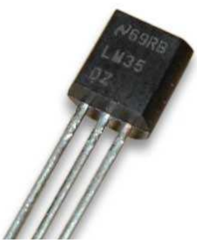


Fig.27 *SubVI* de control del pulsador. Entradas y salidas de la función

4.1.4 Termistor



El termistor en un sensor resistivo de temperatura, cuyo funcionamiento se basa en la variación de la resistencia interna del semiconductor debido al cambio de la temperatura ambiente. Para este TFG se ha elegido el termistor **LM35**, ampliamente utilizado en proyectos electrónicos debido a su sencillez, bajo coste y fiabilidad.

En la Fig.28 se indica la configuración de sus patillas; la de la izquierda estará alimentada con los 5V suministrados por Arduino, la central se conectará al *pin analógico 0* proporcionando el valor de la temperatura, y la última se une al pin de masa. El diagrama de conexiones resultante se representa mediante Circuito 4 del Apéndice A.

Fig.28 Configuración del LM35

El programa que controla al termistor es el mostrado en la Fig.29, siendo sus fases las siguientes:

- **Inicio de la comunicación**

  Detecta de forma automática el puerto USB al que está conectado Arduino y comienza la comunicación con la tarjeta. En esta fase no es necesario definir los pines analógicos ya que todos ellos son pines de entrada.

- **Lazo de control**

  Cada 250ms el programa lee la tensión de entrada (V) suministrada por el termistor a través del pin analógico 0. El Conversor Analógico-Digital (ADC) de Arduino tiene una resolución de 10 bit, lo que nos indica que el sensor LM35 proporcionará un valor comprendido entre 0 y 1023. Además, sabemos que el termistor posee un factor de escala de 10mV/ºC, y que el dato de 1023 corresponde a un valor de 5V, por lo que se puede calcular la tensión leída por Arduino con una simple regla de tres. La temperatura ambiente será el resultado de multiplicar ese voltaje de entrada por 100.

- **Fin de la comunicación**

  Después de salir del bucle no es necesario reiniciar ninguna variable, y por lo tanto, se cierra inmediatamente la comunicación con la tarjeta.

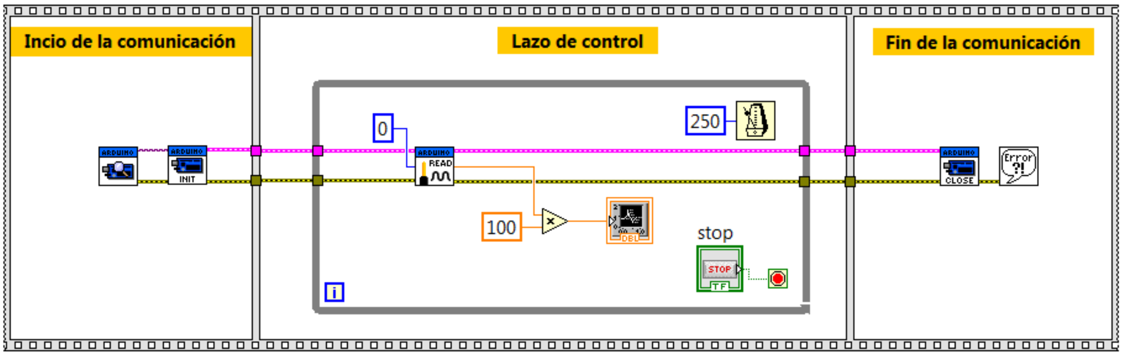


Fig.29 Programa de control del termistor

La Fig.30 representa el *SubVI* creado para facilitar el control del termistor, el cual posee entradas y salidas de comunicación y errores, igual que en las funciones diseñas para los elementos anteriores, e incluye además una entrada para indicar del pin analógico de lectura, y una salida que proporciona el valor numérico de la temperatura ambiente expresada en ºC.

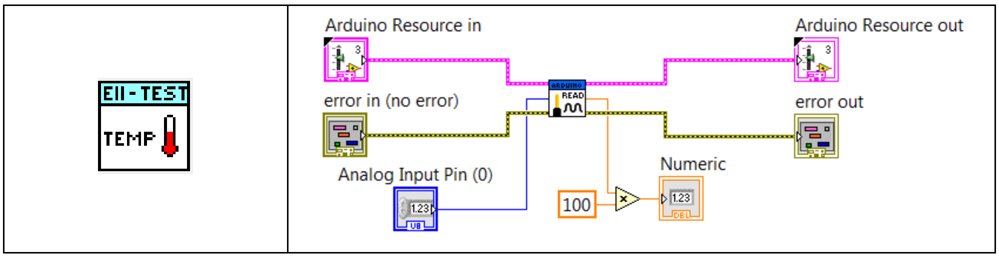


Fig. 30 *SubVI* de control del termistor. Entradas y salidas de la función

4.1.5 Fotorresistencia (LDR)

Una fotorresistencia o LDR (Light-Dependent Resistor) es un componente electrónico cuya resistencia varía en función de la luz. El valor de la resistencia eléctrica de un LDR es bajo cuando hay luz incidiendo en él y muy alto cuando está a oscuras.

La fotorresistencia se alimenta mediante los 5V suministrados por Arduino y se conectará según la Fig.31, formando un divisor de tensión junto con una resistencia de 10kΩ en serie. La señal de entrada sale del punto de unión de ambos elementos y se conecta al *pin analógico 1*, tal y como se aprecia en el Circuito 5 del Apéndice A.

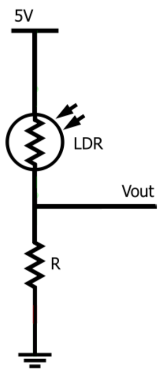


Fig. 31 Divisor de tensión del LDR

Para controlar a la fotorresistencia se ha diseñado el programa de la Fig.32, que se pasa a comentar a continuación:

- **Inicio de la comunicación**

  El programa detecta el puerto USB donde se ha conectado la tarjeta e inicia la comunicación. Como en el caso anterior, no es necesario definir si el pin que se utiliza es entrada o salida, ya que al ser un pin analógico sólo puede ser de entrada.

- **Lazo de control**

  Arduino lee el pin analógico 1 y devuelve el valor de intensidad lumínica en tanto por ciento en función de la tensión de entrada, siendo el 100% correspondiente a un valor de 5V cuando recibe luz directamente y 0% cuando está a oscuras su entorno. Estas operaciones se realizan durante 250ms hasta que el usuario decida finalizar el programa.

- **Fin de la comunicación**

  Al salir del *bucle While* no es necesario reiniciar ninguna variable, por lo que se cierra la comunicación con la tarjeta.

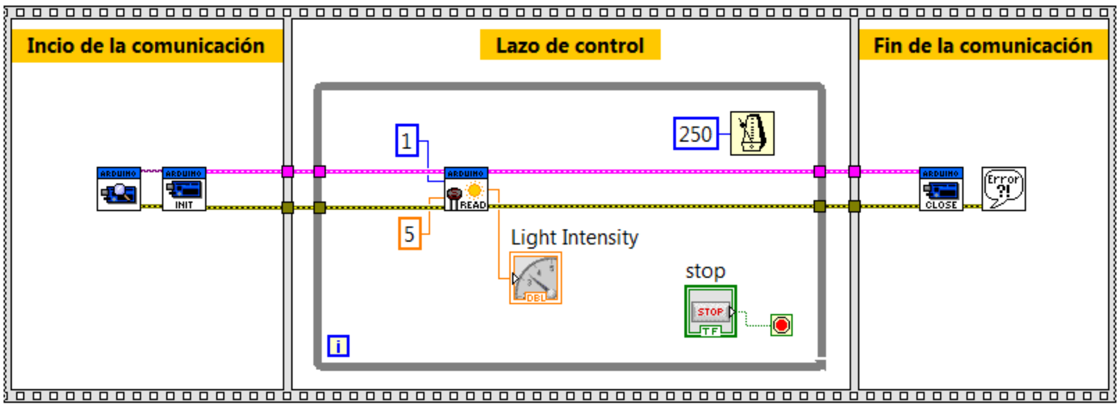


Fig.32 Programa de control del LDR

Ya que esta función de control viene integrada en la librería LIFA (LabVIEW Interface For Arduino) instalada previamente, no es necesario diseñar ningún *SubVI* específico. Este bloque tiene como entradas el pin analógico de lectura y una constante que representa el valor de 5V suministrados al LDR, y su salida es la intensidad lumínica en tanto por ciento; aparte de las entradas/salidas de comunicaciones y errores, presentes en todos los bloques de la librería de Arduino.

4.1.6 Ventilador

Con el fin de ampliar las posibilidades que ofrecen algunos de los elementos anteriores y aprovechar las posibilidades de Arduino, se ha añadido un pequeño ventilador al conjunto alimentado con los 5V propios de la tarjeta.

Se utilizará el *pin PWM 9* de Arduino para enviar la señal que determina la velocidad del ventilador según el diagrama de la Fig.33, que consta de los siguientes elementos:

- Resistencia de 1kΩ
- Transistor NPN 2N2222A
- Diodo 1N4007
- Condensador de 0.1µF
- Ventilador CC 5V – 0.7W

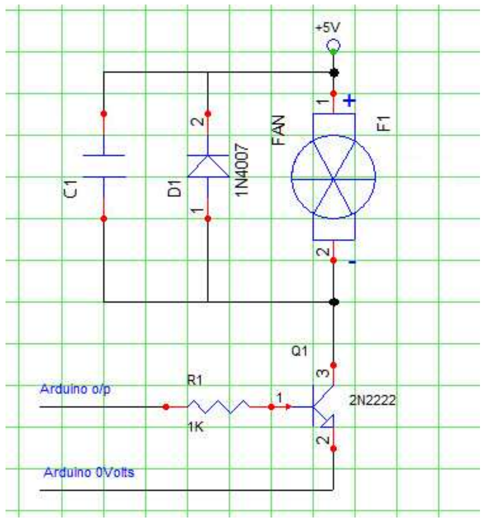


Fig.33 Circuito de conexión del ventilador

La base del transistor se une con la resistencia de 1kΩ, el emisor va al pin de masa de la tarjeta y el colector se conecta al cátodo del ventilador, cuyo ánodo se alimenta con los 5V suministrados por Arduino. Para proteger el circuito de las corrientes inversas del motor del ventilador, se ha colocado un diodo en paralelo que recircula la corriente cuando el ventilador se para. A su vez, se ha dispuesto un condensador también en paralelo que facilita las transiciones al variar la velocidad del ventilador, evitando así los arranques bruscos.

El diagrama de conexiones de todos estos elementos pertenecientes al control del ventilador a través de Arduino, está representado en el Circuito 6 del Apéndice A.

Se ha diseñado el programa de la Fig.34 para controlar la velocidad del ventilador de forma sencilla. El programa sigue las 4 fases vistas en los elementos anteriores:

- **Inicio de la comunicación**

  Se detecta el puerto USB donde está conectado Arduino de forma automática y empieza la comunicación con él. En esta etapa se define el pin PWM 9 como una salida, ya que enviará la señal con que debe girar el ventilador.

- **Lazo de control**

  A través de un control deslizante se va indicando al pin PWM 9 si debe aumentar o disminuir la tensión de salida para obtener una variación en la velocidad de giro. Este *bucle While* se repite cada 250ms hasta que el usuario decide finalizar el programa.

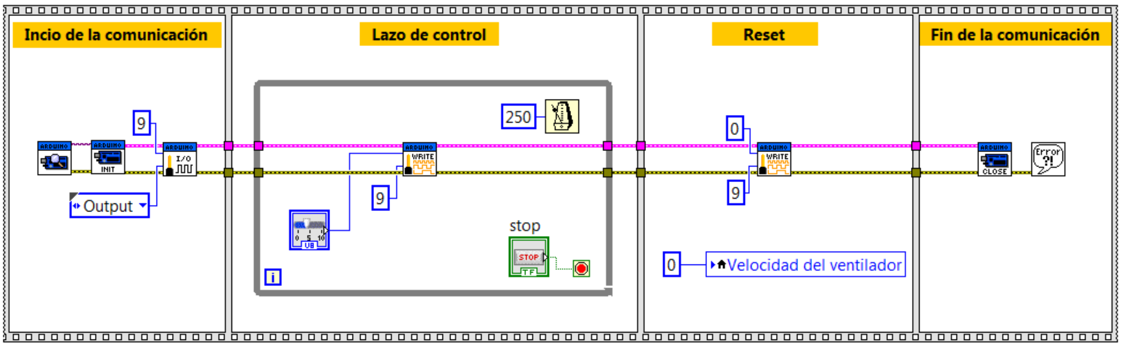


Fig.34 Programa de control del ventilador

- **Reset**

  Después de salir del bucle, se para el ventilador introduciendo el valor 0 en el pin PWM 9 para evitar que siga girando al terminar el programa. También se resetea el control deslizante.

- **Fin de la comunicación**

  Por último, se cierra la comunicación con Arduino.

4.2 PROGRAMA GENERAL DE LA PLACA DE TEST

Una vez que se ha explicado el funcionamiento de cada componente de forma independiente, pasamos a describir las actividades que se pueden realizar en la placa de test, donde algunas de ellas están formadas por dos o tres elementos relacionados entre sí; además, en este apartado se mostrará el panel de control que utilizará el usuario para controlar todos los dispositivos de este proyecto.

4.2.1 Descripción de las funciones de la placa de pruebas

La placa desarrollada en este proyecto es un dispositivo electrónico que permite al usuario interactuar con los diferentes elementos instalados en ella mediante un programa diseñado en LabVIEW, siendo todo ello gestionado a través de la tarjeta Arduino UNO.

Con este programa se pueden realizar tareas sencillas, con un único componente, o más complejas, relacionando los sensores con otros elementos para que se activen dependiendo de los valores recogidos en su entorno. De esta forma, las funciones que se pueden llevar a cabo son las siguientes:

- **Encendido/apagado de un LED**

  Desde el panel del control el usuario podrá encender o apagar de forma directa el LED de la placa.

- **Detector de presión del pulsador**

  Cada vez que se presione el pulsador se generará un aviso visual encendiendo un indicador en el panel de control.

- **Unidad de refrigeración**

  Constituido por un sensor de temperatura, un LED RGB y un ventilador.

  El sensor mide la temperatura ambiente y representa ese valor en una gráfica. A su vez, según sea el valor leído se realizarán las siguientes operaciones:

- Si la temperatura es inferior a un valor mínimo establecido por el usuario (21ºC por defecto), el LED RGB se iluminará en azul y el ventilador estará parado.
- Si la temperatura se encuentra entre los valores mínimo y máximo, el diodo tendrá un color verde y el ventilador comenzará a girar a velocidad media.
- Por el contrario, si el valor del sensor supera el valor máximo introducido por el usuario (valor preestablecido en 27ºC), el diodo cambiará a un color rojo y el ventilador girará a una velocidad mayor.

▪ **Interruptor crepuscular**

Formado por un sensor lumínico (LDR) y un LED RGB.

La fotorresistencia mide la luminosidad del entorno y muestra su valor en un indicador. Se utilizará el LED RGB como si fuera una lámpara cuya intensidad lumínica será inversamente proporcional al valor leído por el sensor, es decir, brillará más cuanto menos luz incida sobre el detector y al contrario.

Estas son las funciones que se encuentran implementadas en el programa desarrollado para este proyecto, aunque todos los elementos pueden controlarse de forma individual o formar parte de un programa más complejo que los englobe en su totalidad.

4.2.2 Panel frontal

El programa diseñado posee el panel de control de la Fig.35, donde se observan tres partes bien diferenciadas entre sí, lo que nos permite controlar y visualizar los cambios que se producen sobre los elementos de la placa de test.

El panel frontal se divide en las siguientes partes:

- Panel de control
- Placa de test virtual
- Gráficas de temperatura y luminosidad

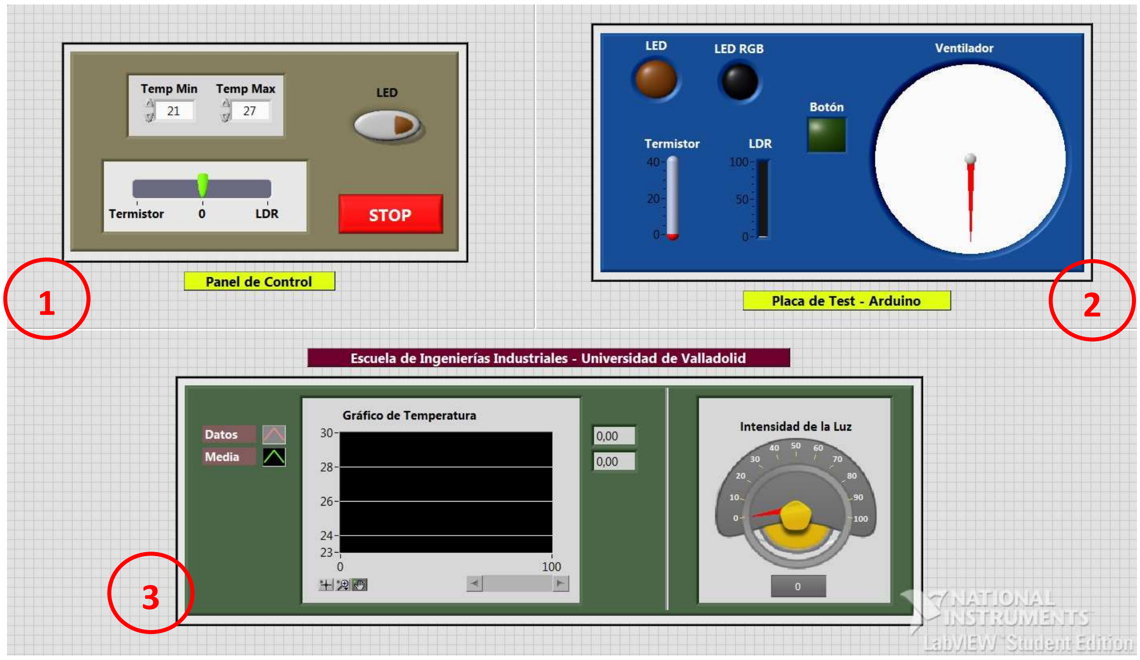


Fig.35 Panel Frontal del programa de control de la placa de test

1) Panel de control

Desde aquí se pueden realizar las siguientes acciones:

- Encender y apagar el LED mediante su botón correspondiente.
- Elegir a través de un control deslizante, la selección de la función de temperatura o la de intensidad lumínica.
- Definir los niveles máximo y mínimo de temperatura correspondientes a la función de refrigeración. Por defecto, el nivel de temperatura mínima está a 21ºC y la máxima a 27ºC.
- Finalizar el programa con el botón STOP.

2) Placa de test

Es la representación virtual de la placa real gestionada por Arduino, de tal forma que cualquier efecto que se muestre en la placa tendrá su acción correspondiente en este modelo. Este dispositivo virtual es capaz de señalar los siguientes estados:

- Indica si el LED se encuentra encendido o apagado.
- Muestra el color que el LED RGB está emitiendo.
- Señaliza mediante un indicador la acción de presionar el pulsador.

- Si se elige la función del termistor, el sensor leerá el valor de la temperatura ambiente y éste se mostrará en su indicador correspondiente.
- A su vez, se representará el estado de funcionamiento del ventilador, según se encuentre en reposo o en movimiento a diferentes velocidades.
- Por el contrario, si se ha elegido la función de luminosidad, se visualizará a través de un indicador el valor de la intensidad lumínica recibida en la fotorresistencia.

3) Gráficas de temperatura y luminosidad

En esta última zona, aparecen representadas las lecturas realizadas por los sensores de temperatura y luminosidad, de tal forma que el valor de la temperatura se indica mediante una gráfica, y la intensidad de luz se muestra con un indicador porcentual.

Una vez finalizada la descripción del panel frontal donde se encuentra la zona de control, la placa virtual y las gráficas, se explicará detalladamente en el siguiente apartado el funcionamiento del propio programa y su diagrama de bloques.

4.2.3 Diagrama de bloques

El núcleo que gestiona y controla todos los elementos de la placa que forman este proyecto es el diagrama de bloques de la Fig.36; en él se pueden observar las 4 fases que se han comentado en los apartados anteriores:

❖ **Inicio de comunicación**

Detecta de forma automática el puerto USB donde está conectado Arduino y comienza la comunicación con él; además se declaran los pines digitales que se vayan a utilizar, ya sean como entradas o salidas.

❖ **Lazo de control**

Este *bucle While* controla las funciones que puede realizar la placa de test definidas por el usuario a través del panel de control comentado anteriormente.

Dicho bucle se repite cada 250ms para tener un funcionamiento continuo hasta que se pulse el botón de parada. Debido a la complejidad de algunas funciones, se explicarán más adelante cada una ellas de forma independiente según la distribución reflejada en la Fig.36.

- **Reset**

Después de finalizar la ejecución del bucle, se apagan los LED y el ventilador para que ninguno de ellos permanezca encendido de forma constante. También se resetean los controladores y el resto de variables e indicadores.

- **Fin de la comunicación**

Por último, se cierra la comunicación con Arduino.

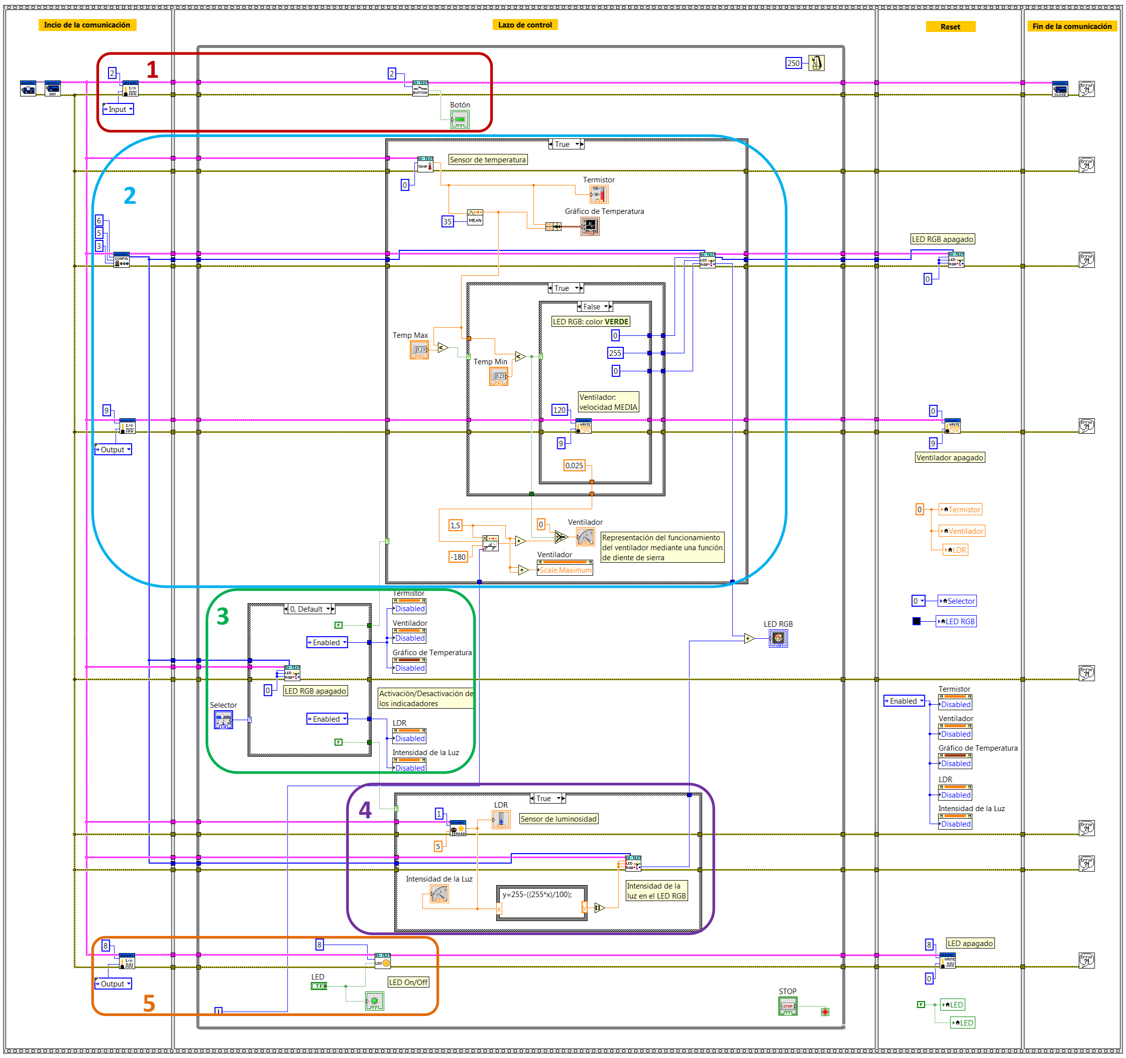


Fig.36 Diagrama de bloques del programa de control de la placa de test gestionada por Arduino

El circuito completo de todos los elementos será la agrupación de cada diagrama de conexión de cada uno de ellos visto en el apartado 4.1 de este mismo capítulo, obteniendo como resultado el Circuito 7 del Apéndice A.

A continuación, se van explicar cada una de las distribuciones de la Fig.36 realizadas para facilitar la comprensión sobre el funcionamiento del programa y cada una de sus acciones.

1) Pulsador

En la Fig.37 se configura el pin 2 como una entrada digital y en el lazo de control utilizamos el *SubVI* creado específicamente para este elemento, ya explicado en el apartado 4.1.3. La conexión del pulsador se realiza según el Circuito 3 del Apéndice A.

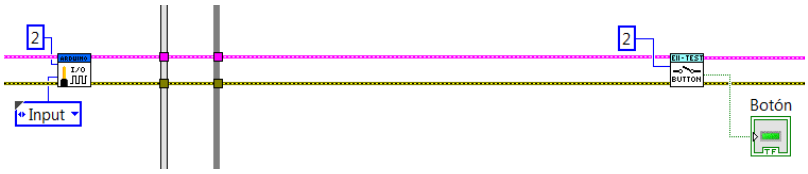


Fig.37 Bloques de control para el pulsador

2) Unidad de refrigeración

Si el usuario elige la función del termistor se activará el *bucle While* de la Fig.38 que controla el sensor LM35, el LED RGB y el ventilador de la placa de test, así como la representación virtual de las acciones de cada uno de ellos.

Según se observa en la Fig.38, primero se deben definir los pines digitales que se van a utilizar, siendo los pines PWM 6, 5 y 3 las entradas para los colores rojo, verde y azul del LED RGB respectivamente, y el pin digital 9 se configura como la salida que controla la velocidad de giro del ventilador, tal y como se ha visto en los apartados 4.1.2 y 4.1.6 de este capítulo.

Una vez dentro del bucle, el sensor es leído a través de la entrada analógica 0 de Arduino y se representa su valor en el indicador de la placa virtual. Por otra parte, en la gráfica de temperatura se muestran los valores medidos directamente del sensor, y el valor medio de todos ellos dentro de un rango definido para evitar picos o valores erráticos y obtener un valor fiable.

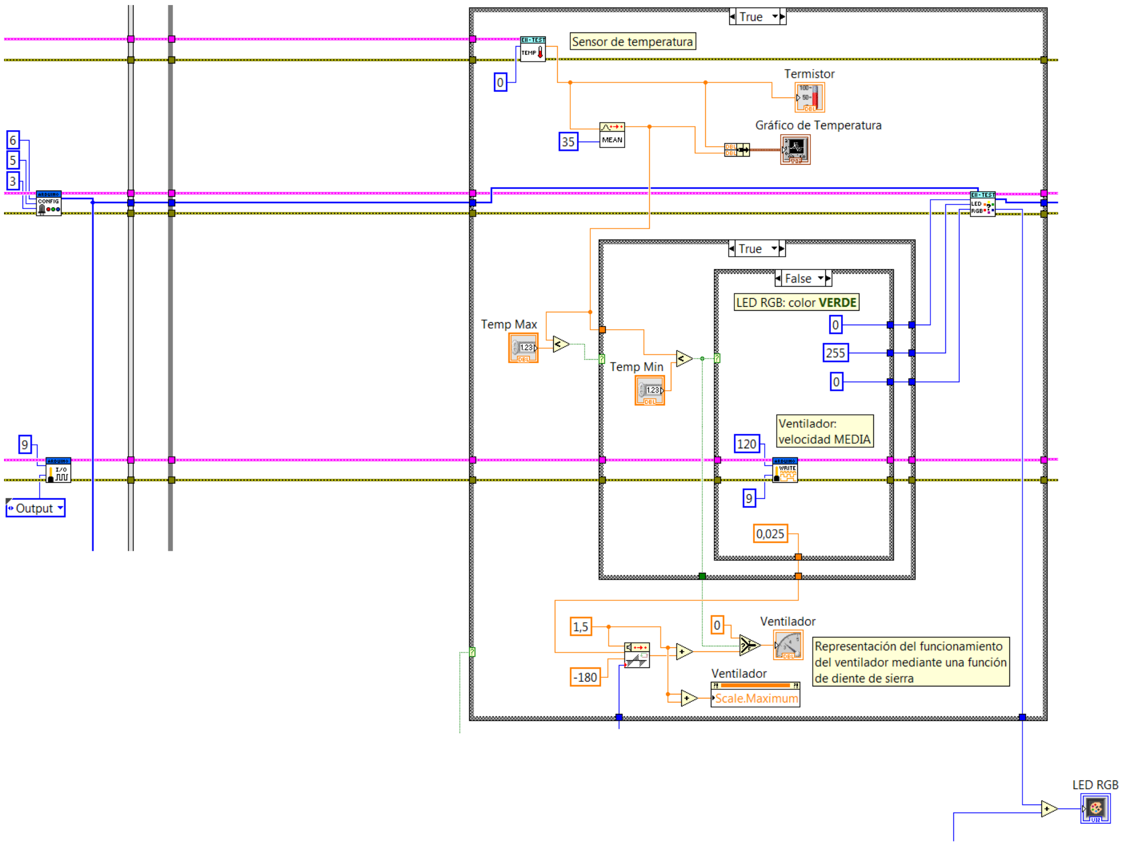


Fig.38 Unidad de refrigeración

A continuación, se emplea ese valor medio como variable de control en los siguientes bucles, comparándolo con los valores de temperatura máximo y mínimo para determinar el color que debe mostrar el LED RGB y la velocidad de giro del ventilador. Así, en la Fig.39 se muestran los tres posibles casos que se pueden dar en función de la temperatura ambiente:

- Si su valor es menor que el límite mínimo, se ordena al LED RGB que se ilumine en azul y el ventilador permanecerá en reposo.
- Si la temperatura se encuentra entre ambos valores límite, se mostrará una luz verde y el ventilador comienza a girar a una velocidad moderada.
- En cambio, si el sensor detecta una temperatura superior al nivel máximo, el ventilador aumentará su velocidad de giro y se informará al usuario de esta situación mediante una luz roja.

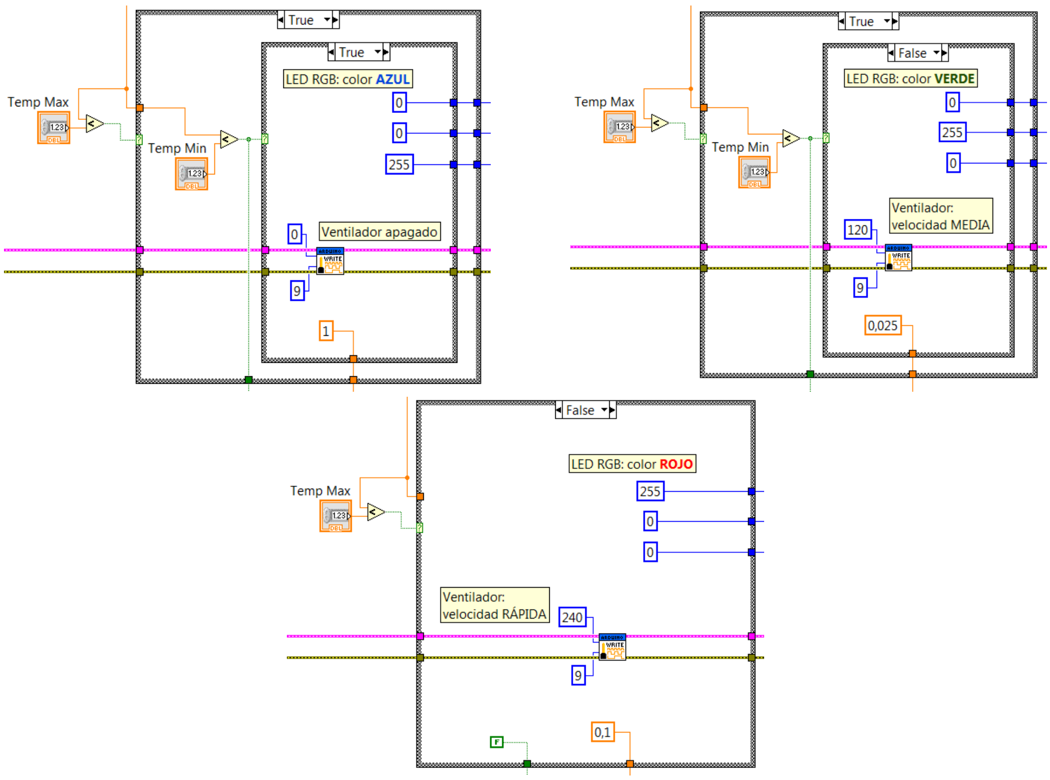


Fig.39 Casos posibles según el valor de la temperatura ambiente

Todos estos cambios relacionados con los diferentes colores en el LED RGB y la velocidad del ventilador, se verán reflejados en la tarjeta virtual del panel frontal del programa. Para representar las distintas velocidades de giro del ventilador, se ha utilizado la función de diente de sierra de la Fig.40 debidamente modificada, en donde su frecuencia de generación está ligada a cada uno de los casos de la Fig.39 vistos anteriormente; por ejemplo, la frecuencia de la onda aumentará si nos encontramos por encima del límite máximo de temperatura, y por lo tanto el ventilador virtual girará más deprisa.

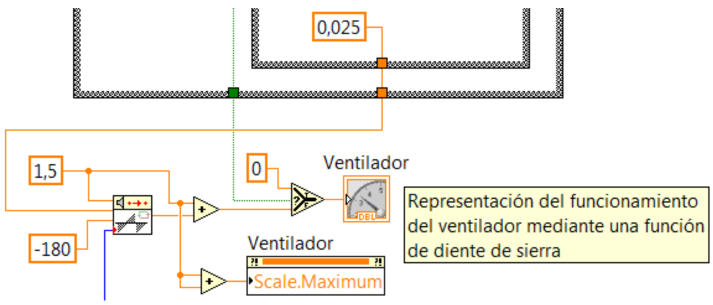


Fig.40 Configuración del ventilador virtual

3) Selector deslizante

El selector del panel de control permite elegir entre la función del termistor (*unidad de refrigeración*), la de la fotorresistencia (*interruptor crepuscular*) o la posición intermedia en donde ambas funciones están deshabilitadas. Al seleccionar una de las dos funciones, se activará su bucle de control correspondiente realizando las operaciones en él definidas, y además, ocultará de forma parcial los elementos indicadores de la otra función. Así, en la Fig.41, se muestra cómo en la posición intermedia los indicadores de ambas funciones se encuentran activos, pero al seleccionar la función del termistor, se desactivan y ocultan parcialmente los indicadores del LDR.

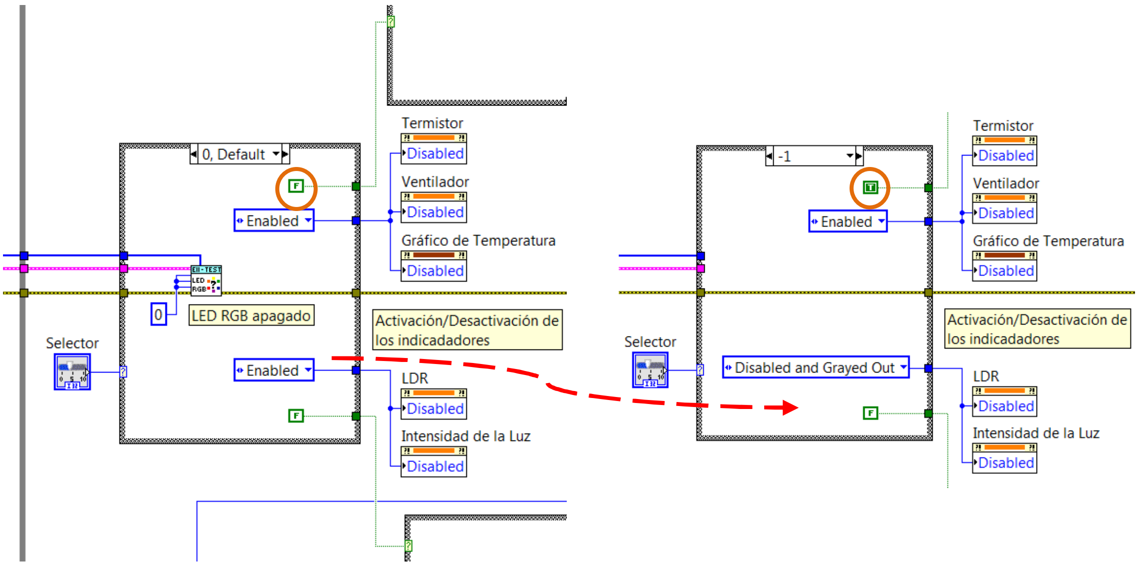


Fig.41 Selector deslizante de funciones

4) Interruptor crepuscular

Al elegir la función de la fotorresistencia se activa el *bucle While* de la Fig.42, donde Arduino lee los valores del sensor LDR utilizando la entrada analógica 1 y los muestra en el indicador de la placa virtual; a continuación, se emplean esos valores en la ecuación de la recta de pendiente negativa del *nodo fórmula* para determinar el valor del brillo del LED RGB. Este brillo es inversamente proporcional a la luz que incide sobre el sensor, es decir, a mayor intensidad lumínica menor brillo tendrá el dispositivo.

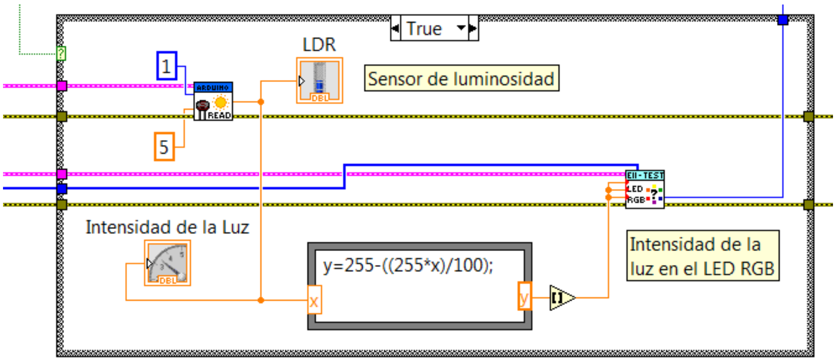


Fig.42 Interruptor crepuscular

El valor resultante del *nodo fórmula* es convertido a un número entero e introducido en las tres entradas del *SubVI* diseñado para el LED RGB correspondientes a los pines digitales PWM 6, 5 y 3 definidos previamente, para de esta forma conseguir una luz blanca que simule una bombilla.

La fotorresistencia y el LED RGB van conectados según se indica en sus respectivos apartados pertenecientes a este mismo capítulo.

5) Encendido/apagado del LED

El LED va conectado al pin 8 tal y como se muestra en el Circuito 1 del Apéndice A. En la Fig.43, el pin es declarado como una salida digital que enviará la señal de activación mediante el *SubVI* diseñado, al accionar su botón en el panel de control.

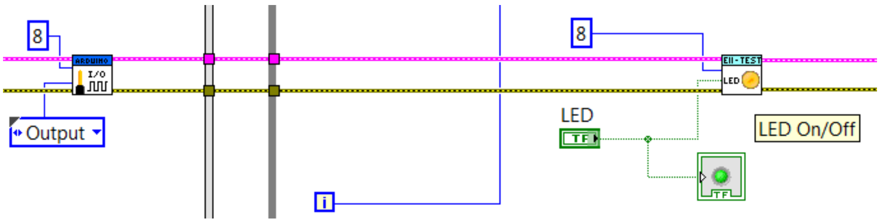


Fig.43 Bloques de control del LED

Este es el funcionamiento del programa que controla la totalidad de los elementos que conforman la placa de test del actual proyecto. Para aumentar sus posibilidades, se ha creado una **aplicación portable** junto con su instalador, lo que nos permitirá ejecutar este programa en cualquier ordenador independientemente de que tenga instalado LabVIEW en él. Los pasos para la creación de la aplicación y su instalador se han detallado en el Apéndice C de esta memoria.

4.3 DISEÑO DEL CIRCUITO IMPRESO

El último paso para la finalización de este proyecto es diseñar el circuito impreso para posteriormente soldar todos los componentes. Para ello, se utilizará la herramienta open-source *Fritzing*.

Fritzing es un programa de automatización de diseño electrónico libre que busca ayudar a diseñadores y artistas para que puedan pasar de sus prototipos en placas de pruebas a productos finales. Este software fue creado bajo los principios de Processing y Arduino, y permite a los diseñadores, artistas, investigadores y aficionados documentar sus prototipos basados en Arduino, y crear esquemas de circuitos impresos para su posterior fabricación. Esta herramienta se puede descargar gratuitamente desde su página oficial:

http://fritzing.org/home/

El proceso de instalación es muy rápido y sencillo:

1. Descargamos el archivo dependiendo de nuestro sistema operativo (Windows 32-64 bit, Mac OS X o Linux 32-64 bit).
2. Descomprimimos en una ubicación que hayamos elegido.
3. Ejecutamos la aplicación haciendo doble click sobre *fritzing.exe*.

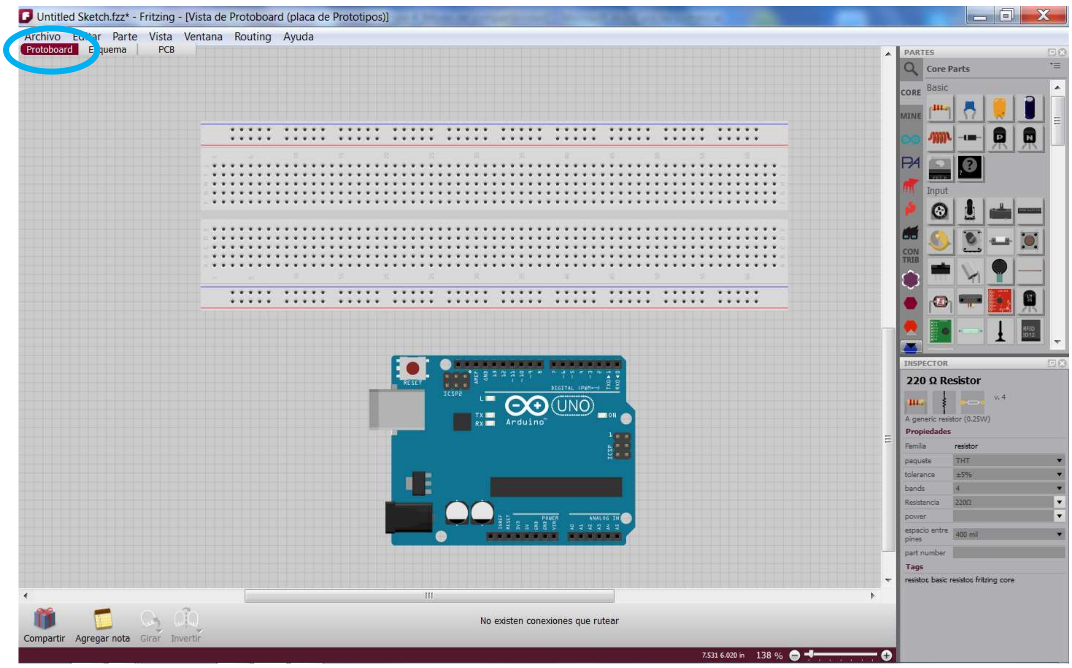


Fig.44 Programa Fritzing para el desarrollo de circuitos impresos

Al abrir el programa veremos la pantalla de la Fig.44, en donde se distinguen 3 partes bien diferenciadas entre sí:

- Zona principal; donde se recrea el mismo circuito con Arduino que tenemos en nuestra protoboard o placa de pruebas.
- Zona de componentes; situada en la mitad superior de la parte derecha de la pantalla. Aquí se seleccionan los elementos que debemos colocar en la zona principal.
- Características de los componentes; localizada en la mitad inferior de la parte derecha de la pantalla. Nos permite definir las características de los elementos, tales como valor de las resistencias, encapsulados, colores y tamaños de LED, etc.

Colocamos todos los elementos con sus respectivas características, atendiendo a los circuitos individuales ya comentados en el apartado 4.1, hasta obtener el diagrama de conexiones completo en la protoboard de Fritzing como se indica en el Circuito 7 del Apéndice A. Una vez conectados todos los elementos, pasamos a la pestaña PCB indicada en la Fig.45 y colocamos los componentes de forma que no se corten las pistas entre sí, obteniendo una distribución dentro de la placa de test como la de la siguiente imagen.

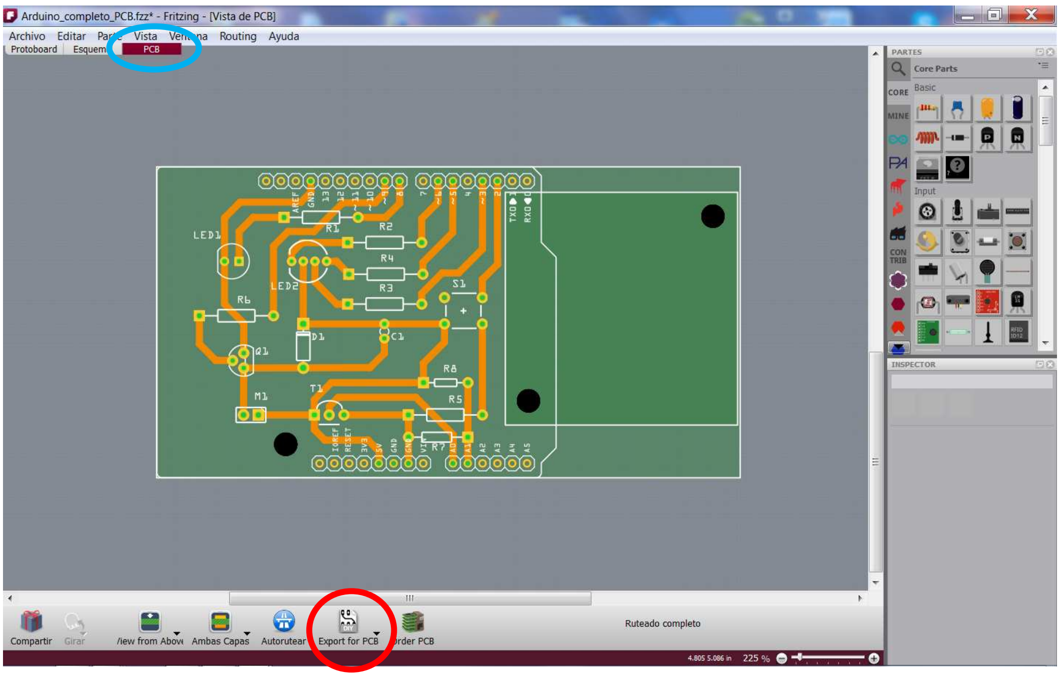


Fig.45 Circuito impreso o PCB

Por último, se crea el circuito impreso pulsando el botón *Export for PCB* de la parte inferior de la pantalla según se muestra en la Fig.45 y se elige el formato PDF. De esta forma, hemos conseguido el circuito PCB de la placa de test gestionada por Arduino de la Fig.46 (Circuito 8 del Apéndice A), listo para ser imprimido e introducido a la insoladora junto con la placa de cobre.

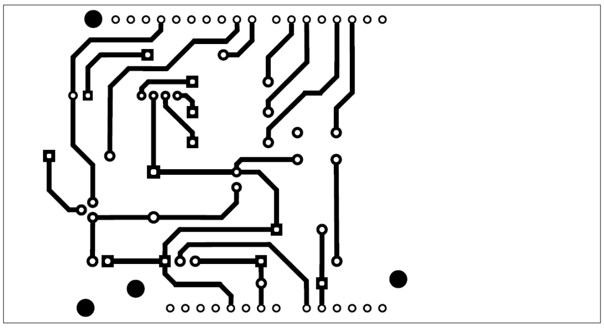


Fig.46 PCB de la placa de test

Como breve recordatorio, en el Apéndice B se indican los pasos de revelado del circuito impreso y algunos consejos para soldar los componentes de agujero pasante de este proyecto.

El resultado final es la placa de la Fig.47, con todos los elementos soldados en sus respectivos circuitos individuales y completamente funcionales, lista para ser conectada al ordenador con un cable USB y alimentada mediante su conector Jack por un adaptador de corriente.

Fig.47 Placa de test gestionada por Arduino

4.4 INSTALACIÓN DE LA PLACA DE TEST

Ahora que ya se ha creado el programa de control y fabricado la propia placa de test, se detallarán los pasos necesarios para poder utilizarla en cualquier ordenador.

1) Conectar la tarjeta Arduino al ordenador mediante un cable USB.

2) Instalar los controladores de Arduino correspondientes, según sea el sistema operativo de 32 o 64 bit. Se pueden utilizar los drivers incluidos en el CD de esta memoria, aunque es más recomendable descargar su última versión directamente de la página oficial de Arduino:

http://arduino.cc/en/Main/Software

3) Instalar el programa de control de la placa de test ejecutando el archivo *setup.exe* ubicado en la carpeta *Instalador\Volume*. Al hacer doble click sobre él, aparece la pantalla de bienvenida de la Fig.48, donde se deberán seguir los pasos indicados por el propio instalador.

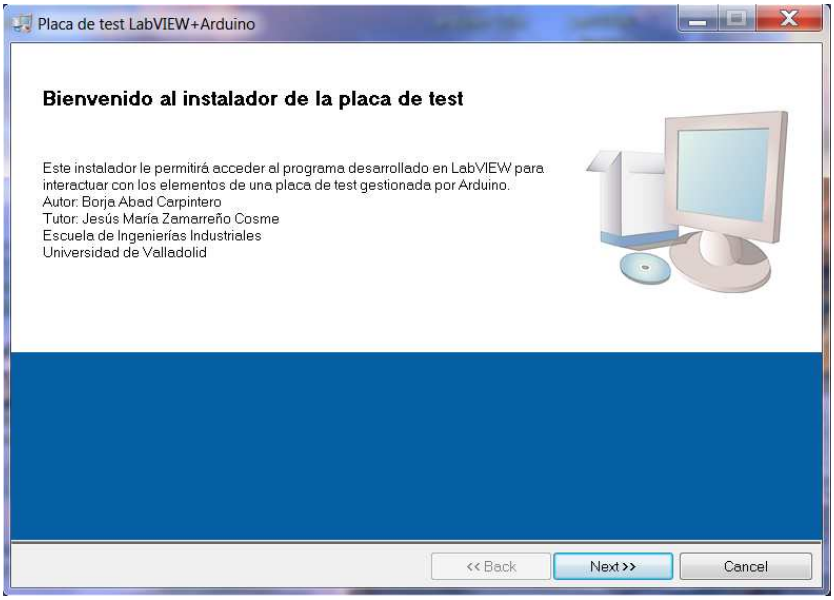


Fig.48 Pantalla de inicio del instalador

4) Ejecutar el programa de la placa de test mediante su icono en el escritorio o a través del Menú Inicio de Windows.

CAPÍTULO 5
ESTUDIO ECONÓMICO

5.1 INTRODUCCIÓN

En el estudio económico se realiza una valoración del coste que produce la realización del proyecto, siempre apoyándose en el hecho de que los programas y los equipos utilizados no lo serán en exclusividad para el mismo, por lo que se realizará un estudio de la parte proporcional del gasto. Hay que tener en cuenta que la evaluación de los costes es compleja y en muchos aspectos subjetiva.

Para realizar el análisis, se comienza diferenciando los diferentes costes existentes, los costes directos y los indirectos:

Los costes directos deben incluir:

- Los costes de personal.
- Los costes de material amortizable.
- Los costes de material fungible.

En el caso de los costes indirectos se incluirán:

- Los gastos de explotación.
- Los gastos administrativos y de dirección.

5.2 COSTES DIRECTOS

En este apartado se evalúan los costes directos, es decir, el coste del personal que ha intervenido en la realización, el coste de los materiales empleados y los costes de los materiales de oficina amortizables.

5.2.1 Costes de personal

Para la realización de este proyecto se ha contado con un ingeniero técnico industrial de especialidad electrónica industrial encargado del diseño, desarrollo y puesta a punto de la placa de test basada en Arduino UNO. Se calcula el coste anual total, para posteriormente adecuarlo al número de horas trabajadas.

Coste anual de un ingeniero técnico industrial:

Incluye sueldo bruto anual, así como posibles incentivos que recibirá por su trabajo. También tiene en cuenta la cotización a la seguridad social y el descuento del I.R.P.F. que es de un 25% del sueldo bruto.

Teniendo en cuenta esto, el coste anual del ingeniero técnico industrial será:

COSTE ANUAL	
Sueldo bruto más incentivos	30.000,00 €
Seguridad Social e I.R.P.F. (25 % sueldo bruto)	7500,00 €
Total coste del ingeniero técnico industrial	**37.500,00 €**

Tabla 5.1 Coste anual

Días efectivos de trabajo de un año:

Se calcula una estimación de días efectivos trabajados al año:

DÍAS EFECTIVOS POR AÑO	
Año medio	365,25 días
Sábados y domingos	-104,36 días
Días de vacaciones efectivos	-20 días
Días festivos reconocidos	-15,00 días
Días perdidos estimados	-5,00 días
Total días efectivos estimados	**220,89 días**

Tabla 5.2 Días efectivos por año

Coste horario de un ingeniero técnico industrial

Sabiendo el número total de días efectivos de trabajo y que el número de horas diarias trabajadas es de 8, se obtiene el total de horas efectivas de trabajo:

220,89 días/año x 8 horas/día = 1.767,12 horas/año

El coste por hora de un ingeniero técnico industrial se calcula dividiendo el sueldo anual del ingeniero entre el número de horas efectivas trabajadas al año:

$$\frac{37.500€ / \textit{Año}}{1.767{,}12\text{horas/Año}} = 21{,}22€ / \textit{hora}$$

Horas empleadas

A continuación, se muestra una tabla donde se hace una distribución temporal del trabajo empleado para el desarrollo del proyecto:

DISTRIBUCIÓN TEMPORAL DE TRABAJO	
Formación y documentación	50 horas
Estudio del problema	75 horas
Desarrollo de la aplicación	100 horas
Puesta a punto del sistema	25 horas
Elaboración de la documentación	50 horas
Total horas empleadas en el proyecto	**300 horas**

Tabla 5.3 Distribución temporal por trabajo

El coste de personal directo se calcula multiplicando el número de horas empleadas por el coste efectivo de una hora de trabajo:

300 horas x 21,22 €/hora = 6.366,00 €

COSTE PERSONAL DIRECTO	**6.366,00 €**

Tabla 5.4 Coste personal directo

5.2.2 Costes de materiales

Para el desarrollo de este proyecto, ya se disponía de antemano de varios equipos (ordenador, impresora, etc.) y del programa LabVIEW, por lo que no es necesario incluir su precio de compra en este estudio pero sí su amortización como se verá en el apartado siguiente. En cambio, la tarjeta Arduino y todos los componentes que forman la placa de test deben ser comprados, siendo el coste de materiales el especificado a continuación:

Hardware:

- Arduino UNO.
- Cable USB.
- Diodo LED.
- Diodo LED RGB.
- Pulsador.
- Sensor de temperatura LM35.
- Fotorresistencia.
- Ventilador CC 0,7W.
- Transistor NPN 2N2222A.
- Diodo 1N4007.
- Condensador 0,1µF.
- Resistencias de 470Ω, 1kΩ y 10kΩ.
- Pines para soldar.
- Tornillos, tuercas y arandelas.
- Placa positiva Bungard 100 x 150 mm.
- Caja para Arduino.
- Fuente de alimentación 12V CC 1,5A.

MATERIAL	UNIDADES	VALOR
Arduino UNO	1	28,77 €
Cable USB	1	2,53 €
Diodo LED	1	0,10 €
Diodo LED RGB	1	1,50 €
Pulsador	1	0,70 €
Sensor temperatura LM35	1	1,45 €
Fotorresistencia	1	1,30 €
Ventilador CC 0,7W	1	21,95 €
Transistor NPN 2N2222A	1	1,00 €
Diodo 1N4007	1	0,05 €
Condensador 0,1µF	1	0,10 €
Resistencia 470Ω	4	0,24 €
Resistencia 1kΩ	2	0,12 €
Resistencia 10kΩ	1	0,06 €
Pines para soldar	32	2,81 €
Tornillos, tuercas y arandelas	2	0,35 €
Placa positiva Bungard 100x150mm	1	11,00 €
Caja para Arduino	1	10,41 €
Fuente de alimentación 12V CC 1,5A	1	18,78 €
	TOTAL MATERIAL	**103,22 €**

Tabla 5.5 Material

5.2.3 Costes de amortización de equipos y programas

Se comienza definiendo un valor económico del material necesario para el desarrollo de este proyecto. Una vez conocida la inversión inicial de los diferentes elementos, podremos calcular el coste de amortización repercutible sobre los costes del proyecto.

Al material informático, tanto software como hardware, se le estima una vida media de tres años. El material de oficina sin embargo tiene una amortización de cinco años.

Para calcular los costes de amortización, se utilizará el modelo de amortización lineal en el tiempo, por tanto, calcularemos el valor de la amortización a través de la diferencia entre el valor inicial y el valor residual entre el número de horas totales.

Utilizaremos el principio de prudencia, aconsejado en el Plan General Contable, para tomar como nulo el valor residual del material hardware, aunque por defecto podría tomarse como el 5% del valor inicial.

Hardware:

- PC compatible, Intel Core i5 – 3,20 GHz, 4 GB de memoria RAM, 1 TB de disco duro.
- Impresora HP LaserJet Pro P1102.
- Arduino UNO
- Componentes de la placa de test.
- Placa positiva Bungard 100 x 150 mm.
- Caja para Arduino.
- Fuente de alimentación 12V CC 1,5A.

Software:

- Sistema operativo Windows 7 64-bit.
- Sistema de programación: LabVIEW 2012 32-bit.
- Paquete ofimático: Microsoft Office 2013.

MATERIAL HARDWARE					
MATERIAL	**UDS.**	**VALOR**	**AÑOS AMOR.**	**VALOR RESIDUAL**	**AMORTIZACIÓN POR HORAS**
PC compatible	1	599,00 €	3	0 €	0,34 €/h
Impresora	1	82,80 €	3	0 €	0,05 €/h
Arduino UNO	1	28,77 €	3	0 €	0,02 €/h
Cable USB	1	2,53 €	3	0 €	-
Diodo LED	1	0,10 €	3	0 €	-
Diodo LED RGB	1	1,50 €	3	0 €	-
Pulsador	1	0,70 €	3	0 €	-
Sensor temperatura LM35	1	1,45 €	3	0 €	-
Fotorresistencia	1	1,30 €	3	0 €	-
Ventilador CC 0,7W	1	21,95 €	3	0 €	0,01 €/h
Transistor NPN 2N2222A	1	1,00 €	3	0 €	-
Diodo 1N4007	1	0,05 €	3	0 €	-

Condensador 0,1µF	1	0,10 €	3	0 €	-
Resistencia 470Ω	4	0,24 €	3	0 €	-
Resistencia 1kΩ	2	0,12 €	3	0 €	-
Resistencia 10kΩ	1	0,06 €	3	0 €	-
Pines para soldar	32	2,81 €	3	0 €	-
Tornillos, tuercas y arandelas	2	0,35 €	3	0 €	-
Placa positiva Bungard 100x150mm	1	11,00 €	3	0 €	-
Caja para Arduino	1	10,41 €	3	0 €	-
Fuente de alimentación 12V CC 1,5A	1	18,78 €	3	0 €	0,01 €/h
TOTAL INVERSIÓN EQUIPOS		**785,02 €**			

MATERIAL SOFTWARE					
MATERIAL	**UDS.**	**VALOR**	**AÑOS AMOR.**	**VALOR RESIDUAL**	**AMORTIZACIÓN POR HORAS**
Windows 7	1	129,00 €	3	0 €	0,07 €/h
LabVIEW 2012	1	580,00 €	3	0 €	0,33 €/h
Microsoft Office 2013	1	119,00 €	3	0 €	0,07 €/h
TOTAL INVERSIÓN SOFTWARE		**828,00 €**			

MATERIAL	**HORAS DE UTIIZACIÓN**	**AMORTIZACIÓN POR HORAS**	**COSTE TOTAL DE AMORTIZACIÓN**
PC compatible	275	0,34 €/h	93,50 €
Impresora	2	0,05 €/h	0,10 €
Arduino UNO	100	0,02 €/h	2,00 €
Windows 7	275	0,07 €/h	19,25 €
LabVIEW 2012	175	0,33 €/h	57,75 €
Microsoft Office 2013	50	0,07 €/h	3,50 €
COSTE TOTAL DE AMORTIZACIÓN			**176,10 €**

Tabla 5.6 Amortización

Las tablas de costes de amortización se han realizado calculando la división de la amortización anual por el número de horas de dichos equipos. Por ejemplo, como las horas trabajadas al año son 1.767,12 horas, los costes de utilización de la tarjeta Arduino UNO por hora son:

$$\frac{28{,}77€}{1.767{,}12\text{horas}} = 0{,}02€ / hora$$

Como es necesario el dispositivo en las etapas de análisis, diseño, programación y documentación, se considera que el tiempo de uso es de 100 horas:

100 horas x 0,02 €/hora = 2,00 €

<u>Costes derivados de otros materiales</u>

Los elementos que se recogen a continuación se denominan consumibles, incluyéndose por ejemplo: los libros de consulta, el papel de impresora, CD's, fotocopias de la documentación, cartuchos de tinta, etc. Este tipo de material es necesario para la realización de los diferentes trabajos en cada una de las fases del proyecto. El coste total de este material es de 50,00 €.

COSTE DE CONSUMIBLES	**50,00 €**

Tabla 5.7 Costes de consumibles

5.2.4 Costes directos totales

De todo lo anterior, se obtiene que los costes directos totales sean los derivados de la suma de los costes de personal, amortización de programas y equipos, y del material empleado.

El valor de los costes totales será:

COSTES DIRECTOS TOTALES	
COSTES DE PERSONAL	6.366,00 €
COSTE TOTAL DE MATERIALES AMORTIZABLES	176,10 €
COSTE TOTAL DE MATERIALES FUNGIBLES	50,00 €
TOTAL COSTES DIRECTOS	**6.592,10 €**

Tabla 5.8 Total coste directo

5.3 COSTES INDIRECTOS

Son costes indirectos los gastos producidos por la actividad requerida para la elaboración del proyecto y que no se pueden incluir en ninguno de los apartados de gastos directos.

COSTES INDIRECTOS PARCIALES	
Dirección y servicios administrativos	180,00 €
Consumo de electricidad	60,00 €
TOTAL COSTES INDIRECTOS	**240,00 €**

Tabla 5.9 Costes indirectos parciales

Por lo tanto, los costes indirectos totales ascienden a:

TOTAL COSTE S INDIRECTOS	**240,00 €**

Tabla 5.10 Costes indirecto totales

5.4 COSTES TOTALES

Los costes totales se obtienen de la suma de los costes directos y los indirectos, siendo el montante total para este proyecto:

COSTES TOTALES	
Costes directos	6.592,10 €
Costes indirectos	240,00 €
COSTE TOTAL DEL PROYECTO	**6.832,10 €**

Tabla 5.10 Costes totales

CAPÍTULO 6
CONCLUSIONES

En este último capítulo se resumen las conclusiones extraídas del presente Trabajo Final de Grado detallado en los capítulos anteriores. Además, se incluye un listado de líneas de investigación para futuros trabajos que mejoren y amplíen las capacidades de este proyecto.

6.1 CONCLUSIONES

Tras finalizar el diseño, programación y construcción de la placa de pruebas, se han obtenido diversas conclusiones que se detallan en los siguientes puntos:

- El conjunto formado por Arduino y LabVIEW permite una programación muy rápida y sencilla de cualquier dispositivo que se haya conectado a la tarjeta gracias a la librería LIFA, evitando los errores de sintaxis típicos de los lenguajes de programación secuencial.

- La placa de test está formada por diferentes sensores, indicadores y actuadores, que otorgan al dispositivo hardware de varios elementos de entrada y salida, permitiendo así una comunicación en ambos sentidos entre el usuario y su entorno.

- Cada elemento tiene su propio circuito de conexión, por lo que se pueden controlar de forma individual o en cambio, agrupar varios de ellos y crear un conjunto sensor-controlador-actuador que realice alguna acción compleja.

- Se han utilizado los pines analógicos y digitales de Arduino UNO, siendo éstos últimos configurados como entradas o salidas dependiendo del elemento conectado. También se han empleado los pines PWM, que nos posibilitan modificar la tensión de salida para variar el brillo de la luz de un LED o la velocidad de giro de un motor de CC.

- Arduino posee un gran número de pines de conexión que incluso después de conectar todos nuestros elementos, han quedado varios libres para añadir más componentes.

- La creación de una aplicación portable del programa de control facilita la forma de mostrar y explicar este proyecto en cualquier ordenador, eliminando la necesidad de tener instalado LabVIEW en él.

Como conclusión final, podemos decir que el conjunto Arduino-LabVIEW es una herramienta con un gran potencial didáctico, tanto a nivel hardware como software, ya que debido a las capacidades de la tarjeta Arduino UNO y la sencilla programación que ofrece LabVIEW, se pueden desarrollar proyectos complejos sin requerir profundos conocimientos en el campo de la electrónica.

6.2 LÍNEAS FUTURAS

Del presente trabajo se desprenden una serie de posibles líneas de investigación que se exponen a continuación:

- Incluir uno a varios servomotores, aprovechando que los pines PWM 10 y 11 están libres, para que interactúen con el resto de componentes o permitir al usuario que controle su dirección de giro mediante la instalación de un potenciómetro.

- Servirnos de la versatilidad de programación que ofrece Arduino para diseñar una nueva aplicación de la placa de test empleando otros lenguajes como C++, Simulink, Processing, etc.

- Utilizar el bus de comunicaciones en serie I2C de Arduino para diseñar nuevos dispositivos que no requieran la ocupación de más pines en la tarjeta. El protocolo I2C sólo necesita dos líneas para transmitir la información: una para los datos (SDA) y la otra para la señal de reloj (SCL). En Arduino UNO esas líneas corresponden con los pines analógicos 4 y 5 respectivamente, los cuales se encuentran libres en este proyecto, permitiéndonos añadir nuevos componentes, como por ejemplo un display LCD, sin necesitar los pines ya utilizados en la placa de pruebas.

- Gracias al conjunto de herramientas LIFA (LabVIEW Interface For Arduino), se pueden emplear otras tarjetas Arduino con más potencia de procesamiento para desarrollar dispositivos industriales de bajo coste. Control de electroválvulas, brazos robóticos, sistemas SCADA, etc., son algunos ejemplos de sus capacidades.

- Diseñar un programa más complejo en LabVIEW que englobe la totalidad de los componentes instalados en esta placa de test.

BIBLIOGRAFÍA

Adafruit, "Guía y ejemplos sencillos basados en Arduino", http://learn.adafruit.com/category/learn-arduino, 2005. Online; accedido el 15-Octubre-2013.

Aivaliotis, Michael, "Getting Started with the LabVIEW Intarface for Arduino", http://vishots.com/getting-started-with-the-labview-interface-for-arduino/, 2011. Online; accedido el 17-Octubre-2013.

Arduino, "Foro oficial de las tarjetas Arduino", http://forum.arduino.cc/, 2005. Online; accedido el 15-Octubre-2013.

Arduino, "Tarjeta electrónica abierta basada en software y hardware flexibles para la creación de prototipos", http://arduino.cc/, 2005. Online; accedido el 10-Octubre-2013.

BlenderNation, "Luces dinámicas con Arduino", http://www.blendernation.com/2012/08/21/blender-arduino-led-control/, 2012. Online; accedido el 3-Marzo-2014.

BricoGeek, "Tutorial: cómo hacer una placa PCB desde principio a fin", http://blog.bricogeek.com/noticias/tutoriales/tutorial-como-hacer-una-placa-pcb-desde-principio-a-fin/, 2008. Online; accedido el 10-Octubre-2014.

Derecho a leer, "Arduino, electrónica libre", http://dalwiki.derechoaleer.org/Arduino/, 2011. Online; accedido el 17-Octubre-2013.

Factoría do Son, "Cómo soldar componentes a la PCB", http://www.factoriadoson.com/index.php/construccion-de-pedales-de-efectos-vi-%E2%80%93-como-soldar-con-estano-los-componentes-a-la-pcb/, 2011. Online; accedido el 10-Octubre-2014.

Fouet, Paul. **Conception d'un système électronique simple de démonstration de fonctionnement de capteurs et d'actionneurs. Mise en oeuvre de la conception à l'aide de l'interface LabVIEW et Arduino**. 2013. Dèpartement Génie Électrique et Informatique Induistrielle. Institut Universitaire et Technologique de Poitiers.

Fritzing, "Programa de automatización de diseño electrónico libre", http://fritzing.org/home/, 2007. Online; accedido el 12-Diciembre-2013.

Girod, Anton, “Tutorial: Fritzing en español”, http://www.youtube.com/watch?v=x2qg9nvZ5LI, 2012. Online; accedido el 15-Diciembre-2013.

Grupo de Tecnología Electrónica. **Tutorial de LabVIEW**. 2006. Universidad de Sevilla. Online; http://www.esi2.us.es/~asun/LCPC06/TutorialLabview.pdf, accedido el 14-Noviembre-2013.

JKI, “Download VI Package Manager”, http://jki.net/vipm/download, 2002. Online; accedido el 16-Octubre-2013.

LabVIEW Hacker, “Getting Started with LabVIEW Interface for Arduino”, https://www.labviewhacker.com/doku.php?id=learn:libraries:lifa:getting_started, 2011. Online; accedido el 16-Octubre-2013.

Mind2Move, “Representación de información en un display”, http://mind2move.com/m2m/arduino-monitoring-system-with-pogoplug-and-archlinux/, 2013. Online; accedido el 4-Marzo-2014.

Moreno Velasco, Ignacio y Sánchez Ortega, Pedro L. **Introducción a la Instrumentación Virtual. Programación en LabVIEW**. 2011. Grupo de Tecnología Electrónica. Universidad de Sevilla. Online; http://www.gte.us.es/ASIGN/IE_4T/Programacion%20en%20labview.pdf, accedido el 14-Noviembre-2013.

National Instruments, “Creating an Application Installer with LabVIEW”, http://www.ni.com/white-paper/5406/en/, 2010. Online; accedido el 29-Enero-2013.

National Instruments, “Distributing Applications with the LabVIEW Application Builder”, http://www.ni.com/white-paper/3303/en/, 2013. Online; accedido el 29-Enero-2014.

National Instruments, “Distributing your application in LabVIEW”, https://www.youtube.com/watch?v=1pEhgqW9268, 2009. Online; accedido el 29-Enero-2013.

National Instruments, “LabVIEW Interface for Arduino”, https://decibel.ni.com/content/groups/labview-interface-for-arduino, 2011. Online; accedido el 16-Octubre-2013.

National Instruments, “LabVIEW Interface for Arduino Setup Procedure”, https://decibel.ni.com/content/docs/DOC-15971, 2011. Online; accedido el 17-Octubre-2013.

National Instruments, “NI LabVIEW 101: Instrucciones en Video para Estudiantes”, http://www.ni.com/academic/students/learnlabview/esa/, 2013. Online; accedido a partir del 15-Octubre-2013.

National Instruments, “NI Virtual Instrument Software Architecture (VISA)”, http://www.ni.com/visa/, 2009. Online; accedido el 16-Octubre-2013.

National Instruments, “Software de Desarrollo de Sistemas NI LabVIEW”, http://www.ni.com/labview/, 2013. Online; accedido el 15-Octubre-2013.

Ruiz Gutiérrez, José Manuel. **LabVIEW+Arduino. Utilización de LabVIEW para la Visualización y Control de la Plataforma Open Hardware Arduino**. 2012. Online; http://josemanuelruizgutierrez.blogspot.com.es/p/mis-publicaciones_7189.html, accedido el 05-Noviembre-2013.

Science? Hell yeah!, "Artículo sobre el control de velocidad en un motor de CC mediante Arduino UNO", http://sciencehellyeah.wordpress.com/electronics/motor-control-part-one/, 2012. Online; accedido el 20-Octubre-2013.

Wikipedia, "Enciclopedia libre", http://en.wikipedia.org, 2001. Online; accedido entre Octubre-2013 y Marzo-2014.

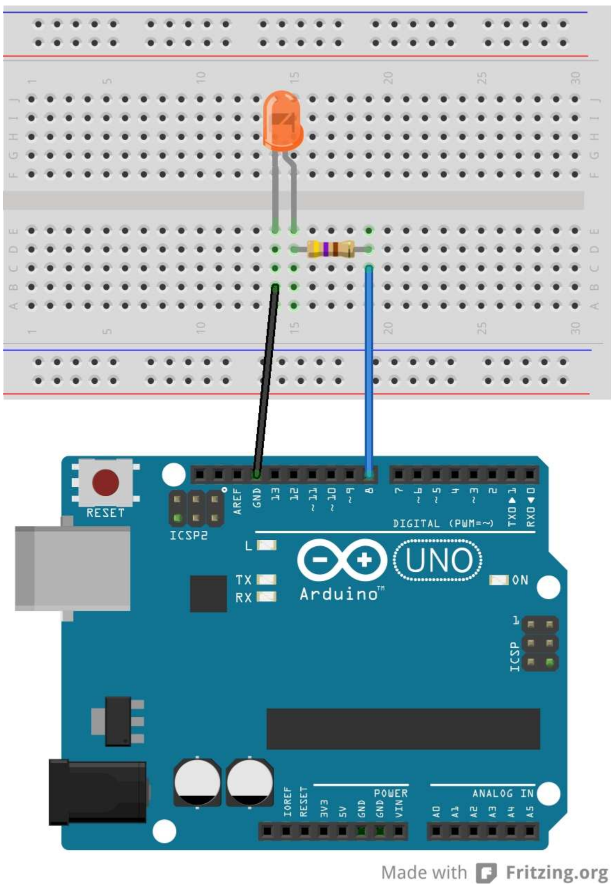
Zona Arcade, "Tutorial: Fabricación de placas de circuito impreso (PCB) – Técnicas y materiales", http://www.zonaarcade.com/index.php?option=com_content&view=article&id=95:tutorial-fabricacion-de-nuestras-placas-de-circuito-impreso-pcb-tecnicas-y-materiales-necesarios-&catid=38:tutoriales-controles&Itemid=106, 2014. Online; accedido el 10-Octubre-2014.

APÉNDICE A

DIAGRAMAS DE CONEXIÓN

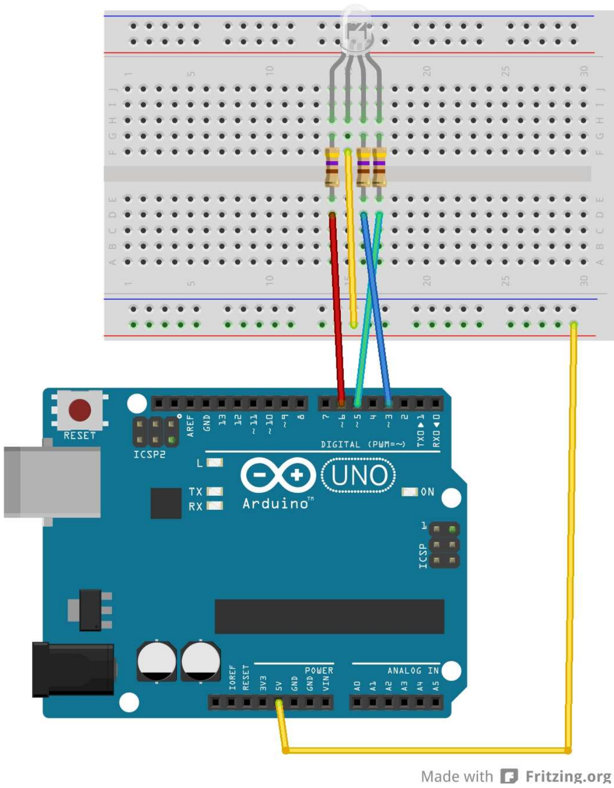
En este apéndice se muestran los circuitos de conexión entre la tarjeta Arduino UNO y los diferentes elementos que se van a controlar mediante LabVIEW, así como el circuito final con todos los elementos pertenecientes a la placa de test.

- **Diagrama de conexión del LED:**



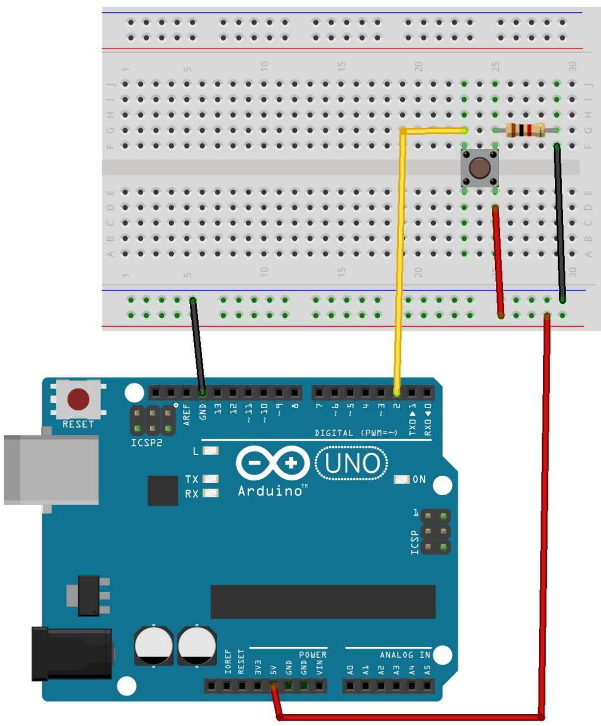
Circuito 1. Conexión del LED

- **Diagrama de conexión del LED RGB:**



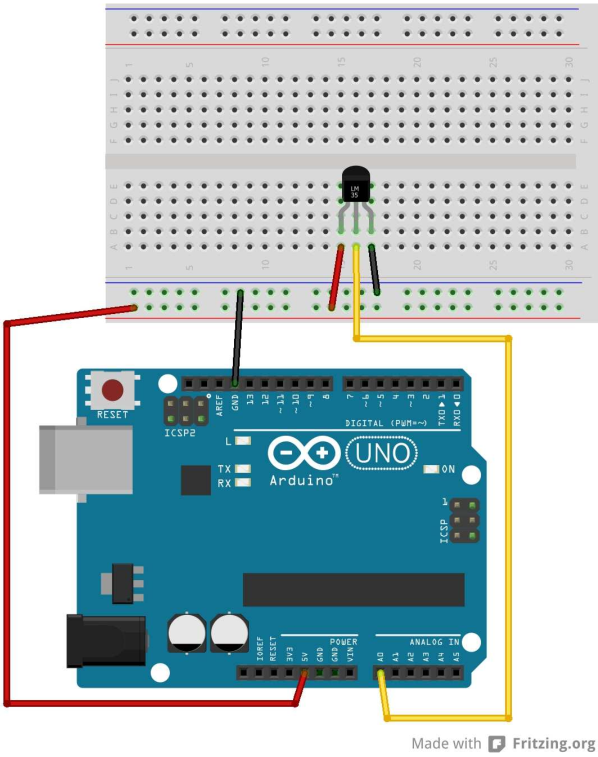
Circuito 2. Conexión del LED RGB

- **Diagrama de conexión del pulsador:**



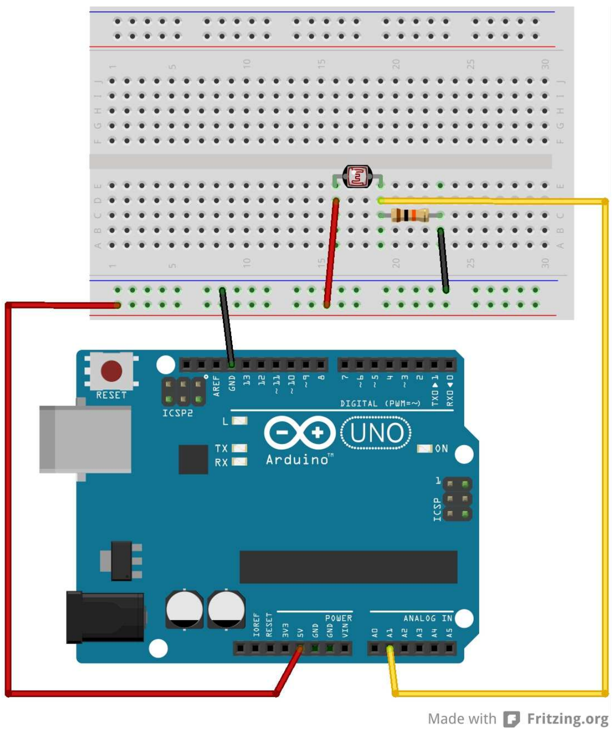
Circuito 3. Conexión del pulsador

- **Diagrama de conexión del termistor:**



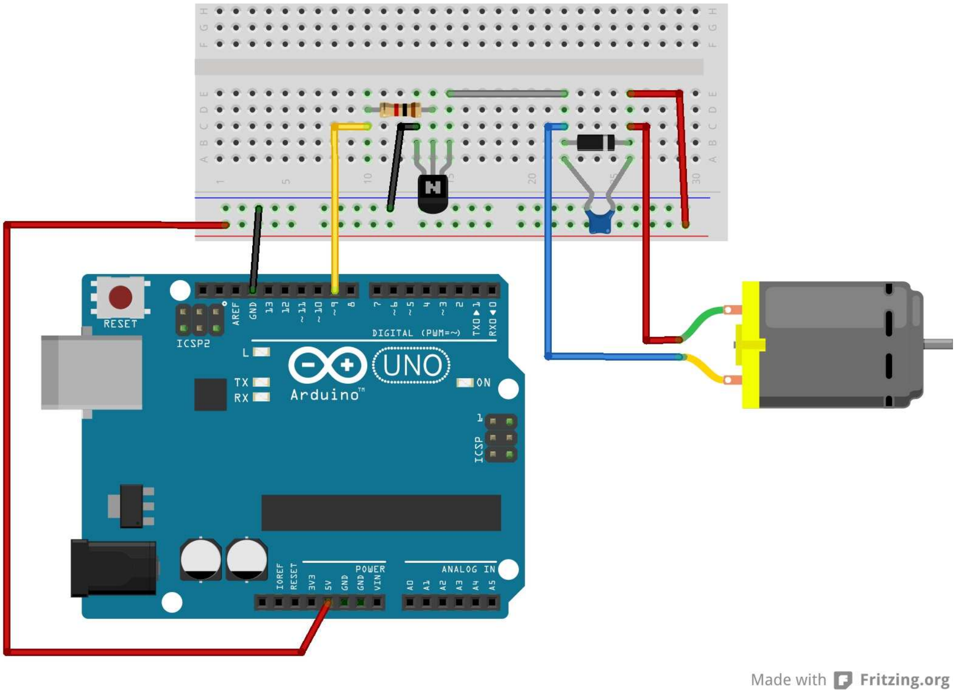
Circuito 4. Conexión del termistor

- **Diagrama de conexión de la fotorresistencia (LDR):**



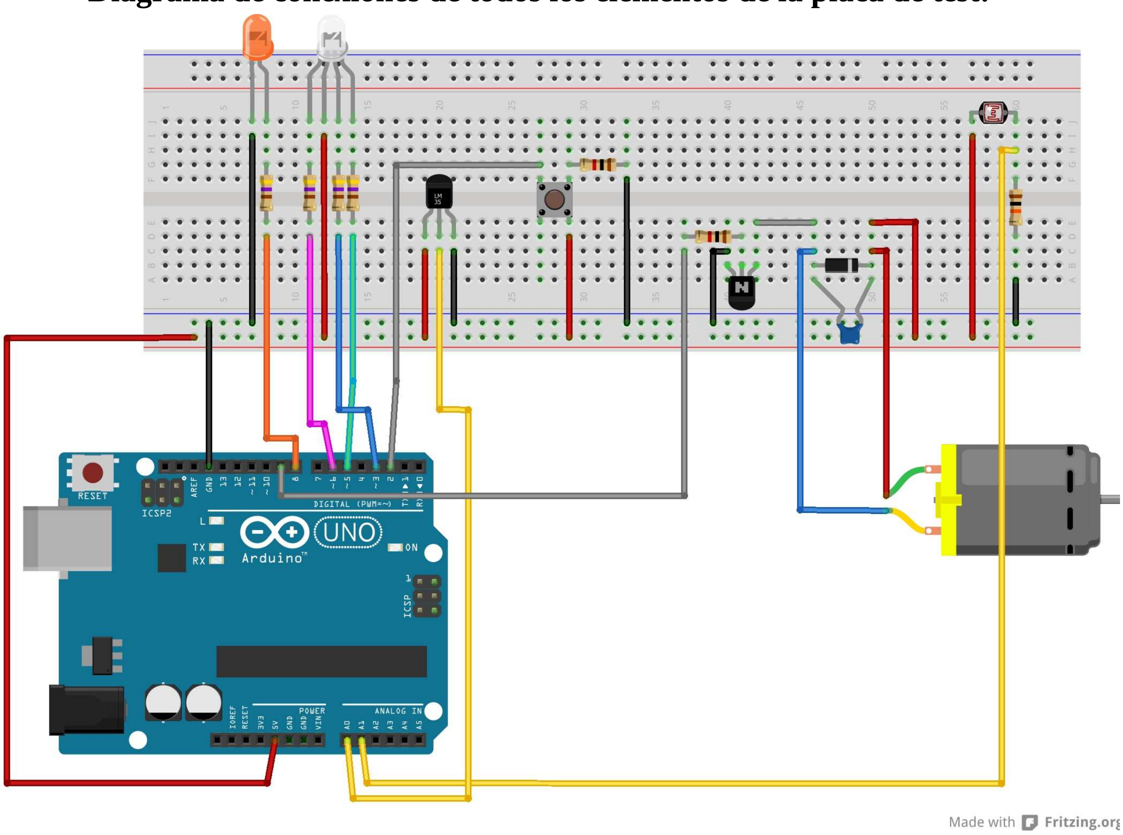
Circuito 5. Conexión de la fotorresistencia (LDR)

- **Diagrama de conexión del ventilador:**



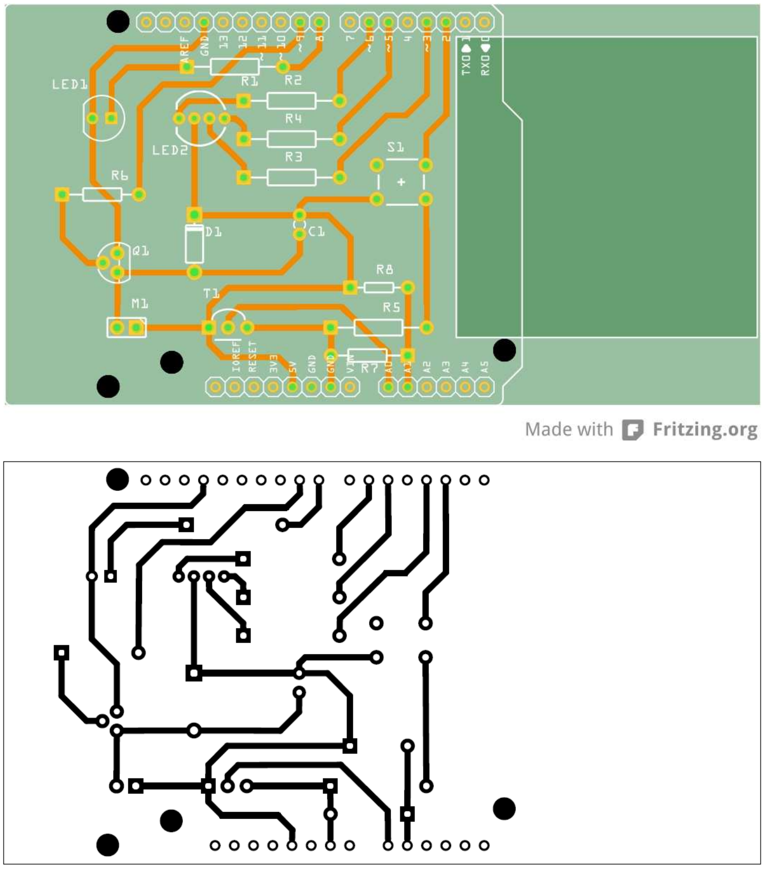
Circuito 6. Conexión del ventilador

- **Diagrama de conexiones de todos los elementos de la placa de test:**



Circuito 7. Conexiones de todos los elementos de la placa de test

El Circuito 8 representa el circuito impreso o PCB del diagrama de conexiones completo de la placa de test, donde se puede apreciar la situación de cada elemento incluido el ventilador.



Circuito 8. PCB

APÉNDICE B
FABRICACIÓN DE UNA PLACA DE CIRCUITO IMPRESO

Tras finalizar el diseño del circuito impreso de la placa de pruebas, éste debe ser revelado sobre una placa de cobre en el interior de una insoladora para posteriormente soldar todos los componentes en sus respectivos diagramas de conexión. Con el fin de indicar los pasos necesarios para un correcto revelado y dar algunos consejos en el proceso de soldadura, se ha decidido introducir este apéndice en la memoria del Trabajo Final de Grado.

PROCESO DE FABRICACIÓN

Antes de comenzar el proceso, debemos elegir una **ubicación bien ventilada**, ya que se trabaja con productos químicos bastante fuertes que generan reacciones químicas exotérmicas, y cercana a una fuente de agua para evitar quemaduras por posibles salpicaduras generadas durante el proceso. Se necesitarán los siguientes materiales:

- Guantes de látex
- Gafas de protección
- Pinzas de plástico
- Fotolitos con los dibujos de las pistas
- Placa fotosensible positiva
- Líquido revelador
- Ácido clorhídrico
- Agua oxigenada (110 vol)

➢ PASOS DEL PROCESO DE FABRICACIÓN

1) Placas positivas:

Nota: se han utilizado las placas de cobre de la Fig.49, de la marca **Bungard,** ya que han sido las únicas que nos han proporcionado buenos resultados.

Fig.49 Placas fotosensibles positivas

2) Transferencia del dibujo del fotolito a la placa:

Colocar la placa, con el fotolito perfectamente alineado, en la insoladora, debajo del cristal, el tiempo que indique el fabricante de la placa (normalmente 4 o 5 minutos).

3) Revelado de la película fotosensible:

Introducir la placa en una cubeta con 75ml de revelador positivo (Fig.49)

4) Lavado con agua.

5) Ataque del cobre no protegido con la fotorresina.

Preparar en una cubeta el atacador (Fig.50), mezclando:

- 25ml de agua
- 25ml de ácido clorhídrico 35%
- 25ml de agua oxigenada (Peróxido de Hidrógeno 110 vol)

6) Lavado con agua.

7) Limpieza de la placa con alcohol para eliminar la fotorresina.

8) Taladrado de los pads.

Fig.50 Cubetas con líquido revelador y atacador

SOLDADURA

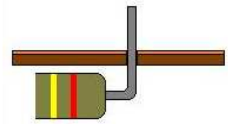
Soldar es unir dos piezas, metálicas en este caso, aportando un material que será el encargado de realizar esa unión. Para estos trabajos utilizaremos estaño, que es un metal con un punto de fusión bajo, con lo que dando calor con el soldador en la zona a soldar, el estaño se fundirá y unirá los elementos que estén en contacto con él. Necesitaremos para este trabajo un *soldador eléctrico lento* como el de la Fig.51, que no es más que una punta metálica especial que se calienta por medio de una resistencia eléctrica. Este soldador debe ser de punta fina y de una potencia adecuada (15 o 30W), ya que un exceso podría dañar los componentes electrónicos más sensibles. También necesitamos el material de aporte a la unión, es decir, el *estaño*, que en realidad es una aleación de estaño y plomo, al 60% y 40% respectivamente.

Fig.51 Juego de soldadura

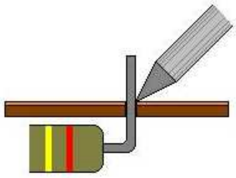
Las zonas y piezas a soldar deben estar totalmente limpias, sin grasa ni suciedad. Igualmente, la punta del soldador debe estar limpia, para ello se puede usar un cepillo de alambres suaves o una esponja humedecida que suelen estar incluidos en el soporte del soldador, y ligeramente estañada.

- **TÉCNICAS DE SOLDADURA MANUAL DE COMPONENTES THT**

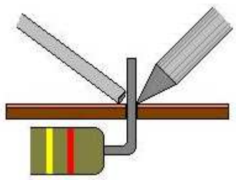
1) Introducir la patilla del componente en el orificio del pad de la placa de circuito impreso. Sujetarlo, evitando que pueda moverse durante el proceso.



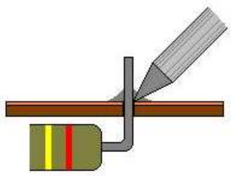
2) Calentar con la punta del soldador tanto la patilla del componente como el pad de cobre de la placa para que ambas piezas alcancen la misma temperatura y evitar el choque térmico que podría producirse.



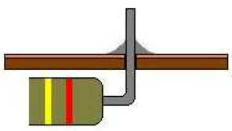
3) Una vez estén suficientemente calientes ambos elementos, se aplica la aleación de estaño hasta que funda. Evitar poner en contacto directamente el soldador con el hilo de la aleación de estaño. Continuar esta operación hasta que el estaño se distribuya alrededor de la patilla del componente de un modo uniforme y en cantidad suficiente para que tome forma de cono o “volcán”.



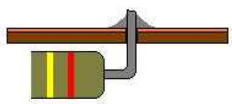
4) Retirar el estaño y mantener unos segundos la punta del soldador en la zona con el fin de que el estaño se distribuya uniformemente.



5) Mantener el componente inmóvil unos segundos hasta que se enfríe y solidifique el estaño. No se debe forzar el enfriamiento del estaño soplando porque se reduce la resistencia mecánica del conjunto.



6) Con la herramienta adecuada, generalmente unos alicates, se corta el trozo de patilla que sobresale de la soldadura.



APÉNDICE C
CREACIÓN DEL SOFTWARE DE DISTRIBUCIÓN

En este proyecto, se ha creado una aplicación portable del programa de control de la placa de test que permite la instalación y utilización de dicho programa en cualquier ordenador independientemente de que LabVIEW esté instalado en él. Por eso, el objetivo de este apéndice es indicar los pasos necesarios para distribuir una aplicación de un programa finalizado a través del constructor que LabVIEW nos ofrece.

Toda aplicación de LabVIEW está formada por estos dos elementos:

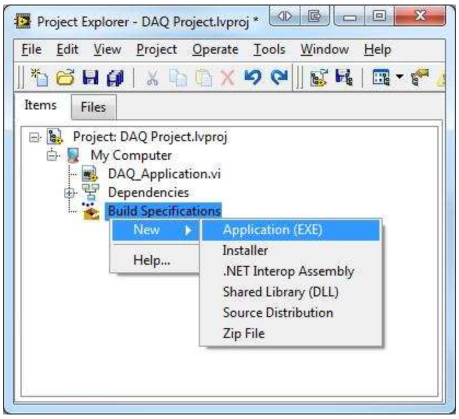
- El archivo **ejecutable** del programa. Se encarga de agrupar todos los archivos .vi que forman parte del programa principal en un único programa.
- El **instalador** de la aplicación. El instalador está compuesto por el archivo ejecutable anterior y por el entorno de ejecución (Runtime Environment) de LabVIEW, permitiendo que la aplicación pueda ser utilizada en cualquier otro ordenador.

El constructor de aplicaciones aprovecha la organización proporcionada por el proyecto de LabVIEW, el cual organiza y gestiona todos los archivos asociados a una aplicación. Para **construir un proyecto** que incluya nuestro programa, se deben seguir los siguientes pasos:

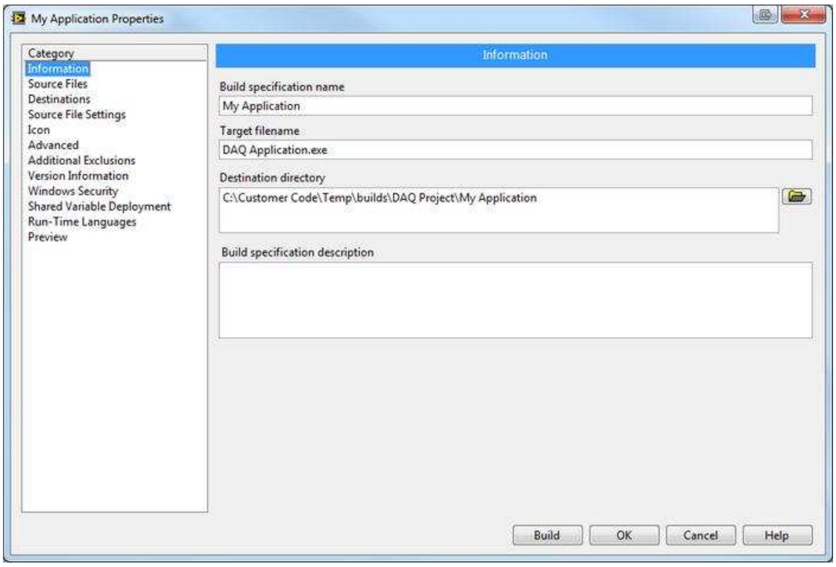
1) Se selecciona *Blank Project* en la página principal de LabVIEW para crear un proyecto vacío.
2) Hacemos click derecho sobre *My Computer* y elegimos *Add* >> *Folder* desde el menú para añadir la carpeta donde se encuentra los archivos del programa principal.
3) Aceptamos y guardamos el proyecto dándole un nombre.

Ahora, esos archivos aparecen incluidos dentro del proyecto y podremos empezar a **crear el ejecutable de la aplicación:**

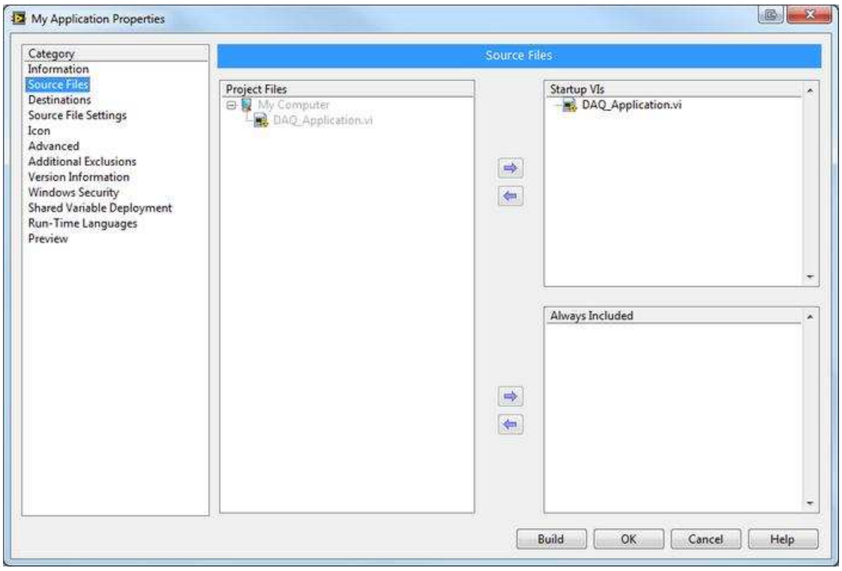
1) Hacemos click derecho en *Build Specifications* y elegimos *New* >> *Application*



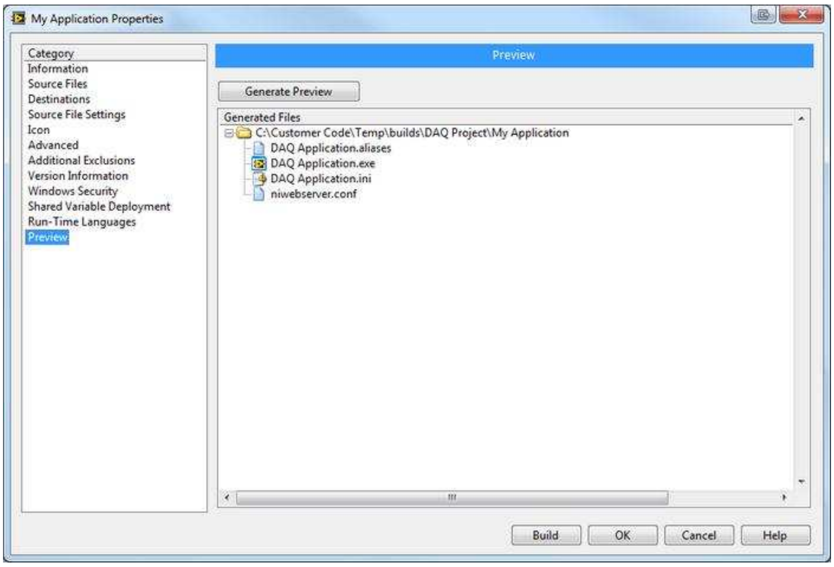
2) En la ventana *Properties*, configuramos el nombre de la aplicación, el directorio destino y la descripción.



3) De la lista *Category* de la izquierda, elegimos *Source Files* para seleccionar los *VIs* que queremos usar en la aplicación. Debe haber al menos un *VI* en la lista *Startup VIs*. Cualquier otro *VI* que sea llamado de forma dinámica deberá ir en la lista *Always Included*.



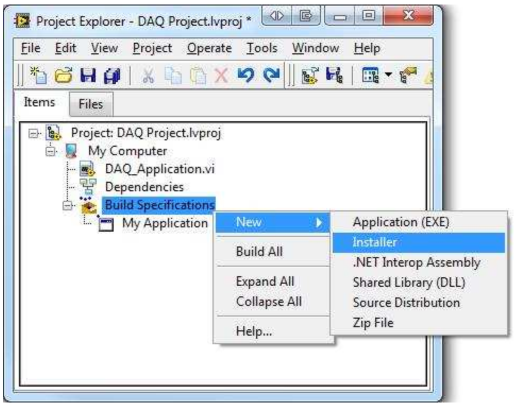
4) En la categoría *Preview*, hacemos click en *Generate Preview* para ver los archivos y ejecutables que serán creados y su carpeta de destino.



5) Una vez que hayamos comprobado que todo esté correcto, hacemos click en el botón *Build*, y el archivo ejecutable será automáticamente construido y colocado en la carpeta de destino especificada.

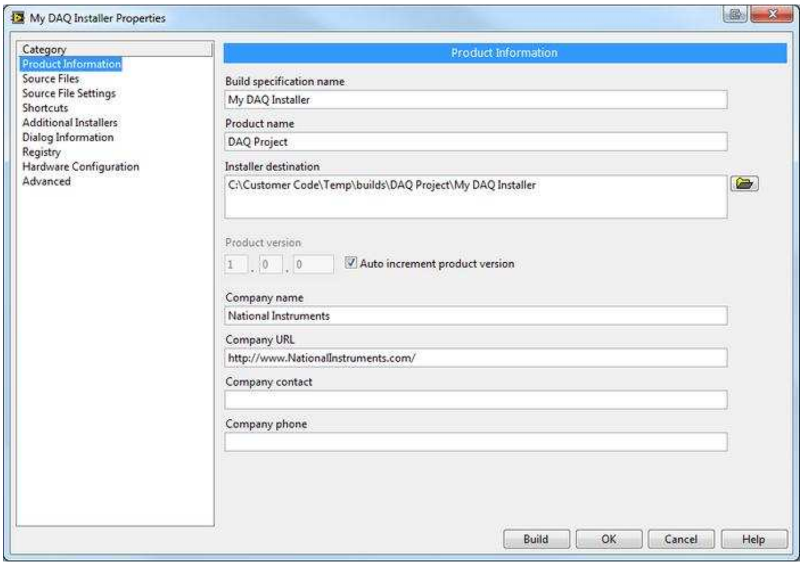
Al crear un ejecutable de nuestro programa, podremos iniciar dicho programa sin tener que abrir LabVIEW ni pulsar el botón *Run*  del Panel Frontal. El siguiente paso es construir un instalador que nos permita utilizar el programa en cualquier ordenador.

Construcción del instalador de la aplicación

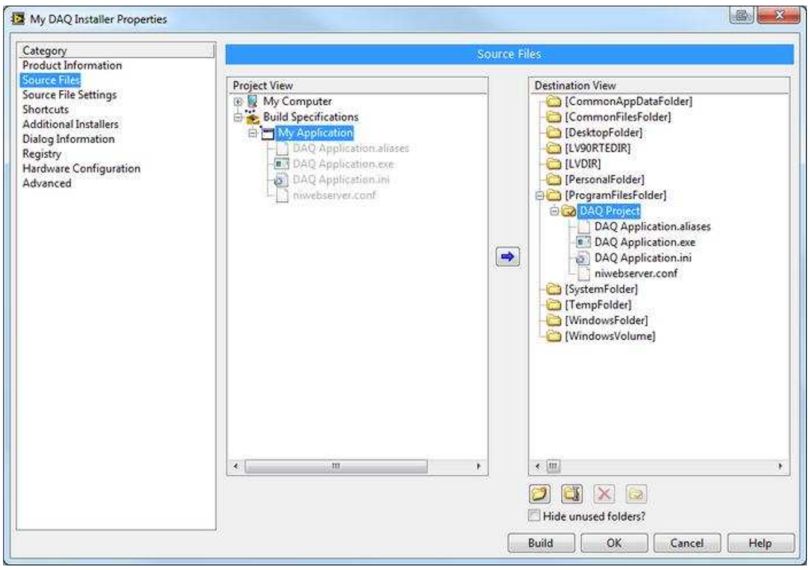


1) Guardamos los avances realizados, pulsamos con el botón derecho del ratón sobre *Build Specifications* y seleccionamos *New >> Installer*.

2) En la ventana *Properties*, configuramos el nombre del instalador, el nombre del producto y el directorio destino.



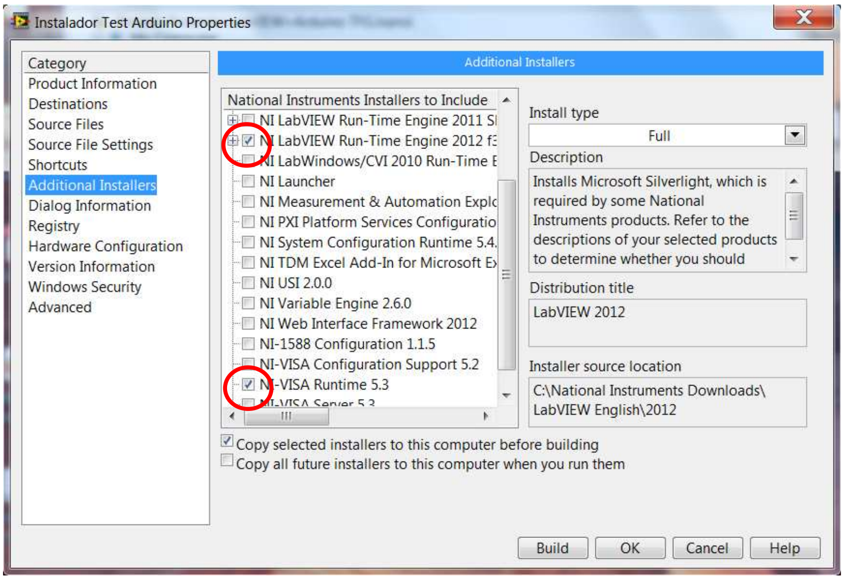
3) En la categoría *Source Files*, movemos el archivo ejecutable antes generado a *Destination View*. Así incluimos el ejecutable en el propio instalador.



4) En la lista de categorías seleccionamos *Shortcuts*, que nos permite crear accesos directos a la aplicación cuando se instale en un ordenador. Algunos lugares donde se pueden crear estos accesos directos son: Menú Inicio, Carpeta de Inicio y Escritorio.

5) En *Additional Installers* se muestra una lista con todos los instaladores que podemos incluir en la construcción de nuestro instalador. Debemos verificar que esté marcada la opción **NI LabVIEW Run-Time Engine**, ya que es el motor de LabVIEW que nos permitirá ejecutar la aplicación.

Nota: en el presente proyecto, además del Run-Time Engine se debe marcar el instalador **NI-VISA Runtime** para que la aplicación del programa sea capaz de leer los puertos USB y detectar en cuál de ellos se ha conectado la tarjeta Arduino.



6) Una vez que hayamos comprobado que todo esté correcto, hacemos click en el botón *Build*, y el instalador será automáticamente construido y colocado en la carpeta de destino especificada.

7) Guardamos los cambios en el proyecto.

Siguiendo todos estos pasos, seremos capaces de crear una aplicación portable de nuestro programa diseñado en LabVIEW totalmente independiente.