# Automatización de pequeña escala con *Open Hardware*

## Automation of small-scale with *Open Hardware*

Luis Diego Murillo-Soto[1]

*Fecha de recepción: 21 de mayo del 2014*
*Fecha de aprobación: 2 de setiembre del 2014*

Murillo-Soto, L. Automatización de pequeña escala con *Open Hardware. Tecnología en Marcha.* Vol. 28, Nº 1, Enero-Marzo. Pág 15-23.

1 Profesor Asociado, Máster en Computación, Escuela de Ingeniería Electromecánica, Instituto Tecnológico de Costa Rica. Costa Rica. Teléfono: (506)2550-9347. Correo electrónico: lmurillo@tec.ac.cr

## Palabras clave

Aplicaciones industriales; tecnología para control; educación en control.

## Resumen

Una posible solución a los problemas de automatización en pequeña escala es realizar los proyectos con la plataforma de prototipado rápido llamada Arduino. El presente trabajo muestra el estado actual del dispositivo y su utilización en labores de control industrial. Además, se programó la lógica de control de un sistema que posee dos alternadores de bombas; la codificación se realizó usando el lenguaje C y el lenguaje LabView. Las pruebas de funcionalidad del controlador mostraron el cumplimiento de ambos lenguajes. Finalmente, se discuten las ventajas y desventajas de la utilización de *open hardware* en proyectos de automatización.

## Keywords

Industrial applications; Control technology; Control education.

## Abstract

One possible solution to small-scale automation problems, is use the rapid prototyping platform called Arduino, this paper shows the current status of the device and its use in industrial control tasks. Additionally, the control logic of a system that contain two pumps alternators have been programmed; the coding is performed using the C language and the LabView language. Tests showed compliance controller functionality of both. Finally we discuss the advantages and disadvantages of using open hardware automation projects.

## Introducción

La automatización de los procesos industriales se puede entender como una actividad tecnológica que busca sustituir, en una actividad determinada, al operador humano por dispositivos mecánicos o electrónicos (Moreno, 2001). El modelo estructural general de las soluciones de automatización es similar al de la figura 1, en la cual se muestra que para la correcta automatización del proceso es necesario: a) comprender el proceso productivo en detalle, b) definir las variables del proceso por medir y controlar, y c) definir la tecnología de control que se encargará de capturar, procesar y manipular las señales desde los sensores y actuadores. Además, el dispositivo lógico de control debe poseer capacidades de comunicación y diálogo con otras unidades u operarios.

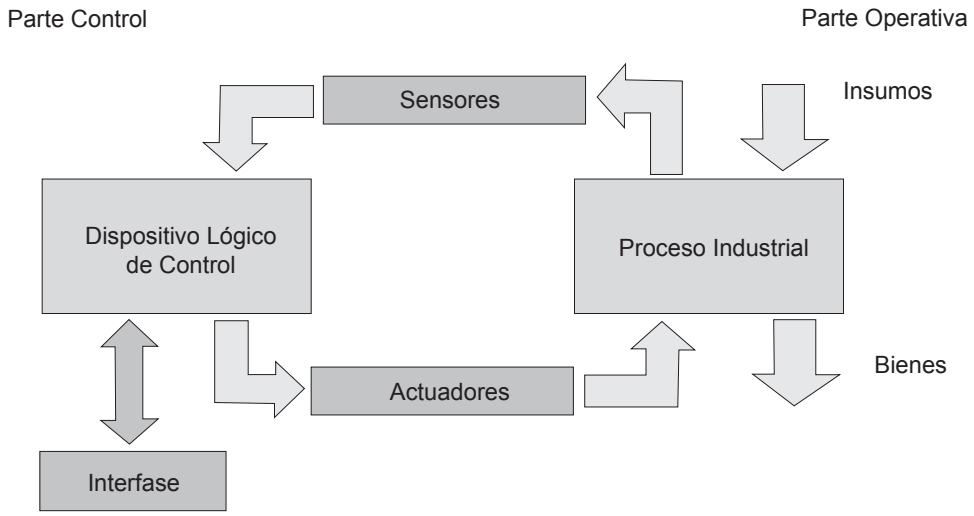



**Figura 1.** Modelo de estructura de la automatización de procesos.

La implementación del dispositivo lógico de control se puede realizar en varias tecnologías, entre ellas los circuitos de relés, circuitos operacionales, circuitos digitales no programables y programables, etc. A nivel industrial comercial, los controladores lógicos programables (PLC, por sus siglas en inglés) constituyen la tecnología dominante; por ejemplo, un *software* especializado para la programación de PLC de distintos fabricantes como el CODESYS (CODESYS, 2013) agrupa más de 300 fabricantes con miles de controladores comerciales. Por otra parte, el paradigma del *hardware* abierto (OSHWA, 2012) como medio para solucionar problemas diversos es cada vez más notorio, tal como se mostró en el "Open Hardware Summit 2013" llevada a cabo en el M.I.T (Mathilde, 2013), en donde se observó un crecimiento de las compañías basadas en este modelo de negocio, tal como se aprecia en la figura 2.

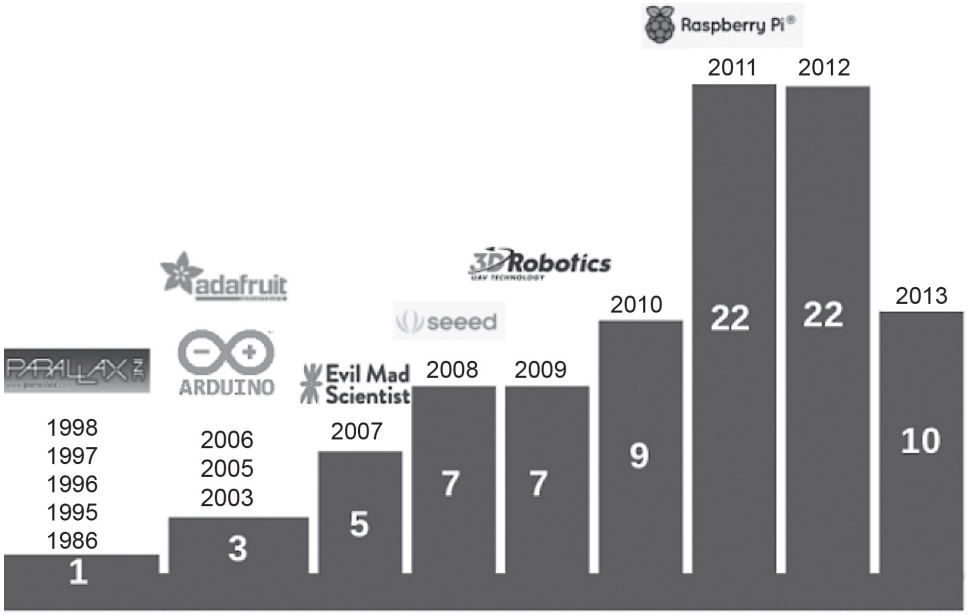


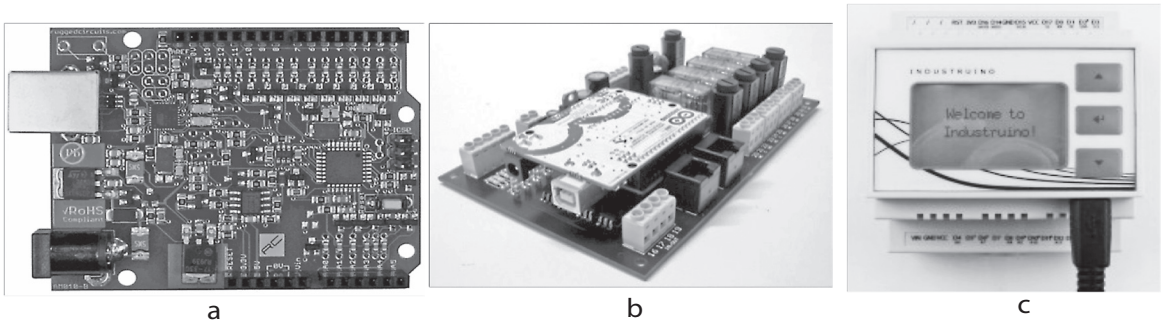

**Figura 2.** Fundación de compañías que desarrollan *Open Hardware* en Estados Unidos. Tomado de Mathilde, 2013.

## La Plataforma Arduino

Si bien es cierto existen muchas plataformas de desarrollo de *hardware* abierto, los microcontroladores conocidos como Arduinos se han convertido en el ambiente de prototipado más usado a nivel global (Leung, 2012), por su costo, su variada oferta de controladores y accesorios y su forma de programación basada en C con múltiples bibliotecas de código abierto, aunado a su capacidad de procesamiento. Por ejemplo, si se revisa la especificación técnica para la tarjeta Arduino MEGA (Arduino, 2014a), se observa que posee un procesador marca ATMEL modelo ATmega 2650 (ATMEL, 2014), que es un procesador de 8 bits tipo RISC que procesa 16 millones de instrucciones por segundo (MIPS) a 16 MHz; una memoria de trabajo *flash* de 256K Bytes, 54 pines configurables como entradas o salidas, 16 entradas analógicas de 0 a 5 voltios con una resolución de 10 bits y frecuencia de muestreo de 10kHz, etc. Con estas especificaciones de procesamiento y capacidad para manejar entradas y salidas digitales, es válido plantear la automatización de pequeña escala con estos dispositivos, es decir, como sustituto de controladores industriales tales como los relés inteligentes y los nano y micro PLC. Actualmente existe en la red de profesionales LinkedIn un debate abierto sobre la utilización o no de los Arduinos en la industria (Shervin, 2013), en procesos no críticos, no riesgosos o en sistemas automáticos de magnitud pequeña o en máquinas aisladas.

Si se revisan las especificaciones técnicas del micro PLC de última generación modelo CPU 1212C (SIEMENS Corp., 2012), se ve que posee una memoria de trabajo de 50 K Bytes, procesa 10 MIPS y tiene 8 entradas y 6 salidas digitales, con 2 entradas analógicas, etc. Si se comprara solamente la velocidad de procesamiento, cantidad de memoria y cantidad de entradas y salidas, el Arduino Mega es una opción superior a este PLC particular. Sin embargo, estas no son las únicas variables que se deben analizar; el PLC tiene características a su favor, por ejemplo, las entradas están aisladas optoeléctricamente, posee salidas a relés o a transistor de 24 voltios, tiene carcasas plásticas con algún grado de protección IP, el *software* de programación utiliza lenguajes de programación estandarizados definidos en la norma IEC 61131-3, -lo que facilita las modificaciones a los programas de automatización-, tiene respaldo del fabricante, certificaciones eléctricas UL, CE, etc., modularidad y expansión, manejo de protocolos industriales, etc.

Dadas algunas debilidades que se le señalan al Arduino, han aparecido versiones de *hardware* robustas, pensadas para ambientes más hostiles, tal es el caso de Ruggeduino (Rugged-Circuit, 2013) que incorpora protección para todas las entradas, montaje para riel DIN; el PLC Arduino, que consiste de *shield* (tarjeta de expiación) para Arduino Uno, en el que las entradas son optoacopladas para señales de 24 voltios DC, salidas a relé, puertos de comunicación RS 485 y *ethernet* (Biemme-Italia , 2014); y el Industrino, con carcasa para riel DIN, pensado para la automatización domótica (Ainura, 2014) (ver figura 3).




a b c

**Figura 3.** a) El Ruggeduino, b) PLC Arduino, c) Industrino. Tomado de (Rugged-Circuit, 2013; Biemme-Italia , 2014; Ainura, 2014)

Por otra parte, a nivel industrial ya aparecen casos exitosos reportados, como la puesta en marcha de una impresora industrial de etiquetas plásticas controlada totalmente por cinco Arduinos (Romano, 2013), o la actualización con Arduinos de una máquina cuyas tarjetas obsoletas no se conseguían en el mercado (MAAC, 2012). Por otra parte, y atacando otra debilidad señalada como la interconexión con otros dispositivos, la compañía Annikken (Piasim Corporation, 2014) ofrece el *shield* de conexión *bluetooth* para controlar, monitorear, guardar datos en cualquier dispositivo móvil con el sistema operativo Android o IOS. Asimismo, el control o monitoreo a través de internet se realiza de forma muy sencilla usando el Arduino Yun y las bibliotecas de Temboo (Temboo, 2013), que permiten publicar en tiempo real datos en Facebook, Twitter o tablas de datos y curvas en Plotly (Plotly es una herramienta científica basada en la WEB, para importar datos y genera gráficas de tendencias, líneas de ajuste, graficas de barra, etc.).

Otra tendencia marcada para la utilización de los Arduinos consiste en realizar la adquisición de datos y el preprocesamiento en el sitio, con enlaces inalámbricos a otros dispositivos, sin embargo, para esto se requiere mayor potencia de procesamiento. En este sentido, Intel incursiona desarrollando el Arduino llamado Galileo (Arduino, 2014c), que tiene un procesador Intel de 32 bit, que corre a 400 MHz, con 256 Mega Byte de memoria de trabajo, redes *ethernet* de 100 Mbits, mini-PCI y *memory card* de 32 Giga bytes, capacidades de sobra suficientes para realizar labores de automatización, adquisición y preprocesamiento de datos, etc.

Para la compra de tarjetas, sensores, accesorios y distintos *shields para* Arduino, existen varios proveedores en Costa Rica, como Radio Shack o tiendas virtuales como CRCibernepetica (Gridshield, 2014), que ofrecen una amplia gama de *hardware* con precios similares o idénticos a los publicados en tiendas especializadas en Estados Unidos como Sparkfun (SparkFun-Electronics, 2009).

### Programación de los Arduinos

En cuanto a su programación, los Arduinos se codifican con el lenguaje C o C++ en el entorno del desarrollo integrado (IDE) de Arduino (Arduino, 2014b). Además, existen paquetes de programación alternativos, por ejemplo, el *CodeBlocks for Arduino* (Huang, 2013). También se puede programar de manera gráfica usando *Ardublocks* (Ardublocks, 2014) o el paquete *QP Framework* (Quantum-Leaps, 2014) diseñado para programar máquinas de estado complejas a partir de los SateChart. La utilización de *software* para el procesamiento numérico o computación científica, tal como Scilab 5.5, es posible gracias al lenguaje de programación gráfico de Xcos y una biblioteca desarrollada por Bruno Jofret (Jofret, 2013). Cabe destacar que los paquetes mencionados y sus bibliotecas son gratuitos y de uso público.

Desde la perspectiva de los *softwares* comerciales, los microcontroladores Arduinos se pueden programar de manera gráfica desde el *software* de programación *LabView* de National Instrument (National-Instrument, 2013) o desde Simulink de Matlab (Picker, 2012). Si bien estos últimos paquetes son reconocidos para realizar computación científica, se requiere la compra de las licencias, y en el caso de Matlab la compra de la biblioteca para el Arduino.

## Programación del Arduino Mega

Para comprobar las facilidades de programación del Arduino como sustituto de un nano o micro PLC, se decidió programar un controlador para un sistema de eventos discretos, ejemplificado por dos alternadores de bombas, donde cada alternador responde de forma independiente a la dinámica de los eventos. Con la finalidad de que el diseño del control sea poco complejo, el diseño no contempla la detección, ni diagnóstico de faltas o fallas en el sistema.

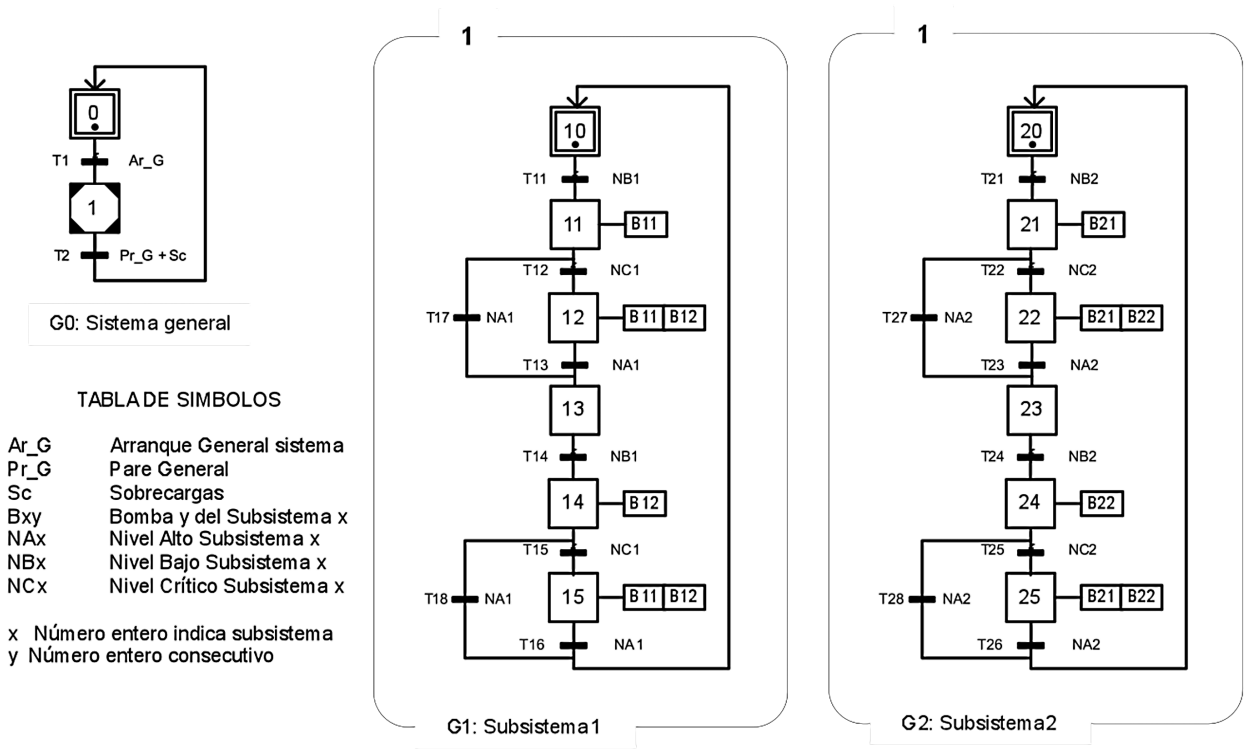



**Figura 4.** GRAFCET para dos alternadores de bombas que arrancan y paran por las mismas señales de mando.

El automatismo cuenta con dos botoneras, un botón arranque general y otro para el apagado general o sobrecarga. Cada sensor y actuador tendrá un código que indica a qué alternador pertenece; ese número se indica con la letra y. Cada subsistema de alternancia de bombas posee un tanque de captación, que a su vez tiene tres sensores de nivel, llamados nivel alto $NA_y$, nivel bajo $NB_y$ y nivel crítico $NC_y$; además existen dos bombas llamadas $B_{y1}$ y $B_{y2}$. Cuando el agua activa el sensor de nivel $NB_y$, la bomba $B_{y1}$ se activa y se apaga hasta alcanzar el nivel $NA_y$. La próxima vez que se active el sensor $NB_y$, se activa la segunda bomba $B_{y2}$ y se apaga hasta alcanzar $NA_y$. Si la demanda de agua es tal que el nivel en el tanque descienda, el sensor $NC_y$ se encenderán ambas bombas. El sistema debe recordar el orden de la alternancia. Esta secuencia de operación se repite hasta que se presione el apagado general del sistema o la señal de sobrecarga. La solución lógica se muestra en la figura 4, y se realizó según la norma IEC 60848 (IEC-60848, 1999), conocida como GRAFCET.

La implementación del controlador se hizo de dos formas; en la primera se usó el paradigma de la programación gráfica mediante el *software* LabView 12.0. En la figura 5 se muestra la implementación del ciclo de escaneo de cinco etapas, lectura de entradas, ejecución de código, escritura de salidas, comunicación y manejo de error de código. El código de los alternadores no se muestra por razones de espacio y estos están encapsulados en dos *Virtual Instruments* indicados con los nombres de Alternado 1 y Alternado 2 en la figura 5. Para la otra implementación se utilizó el lenguaje C a través del ambiente de desarrollo de Arduino (ver código en Apéndice A).

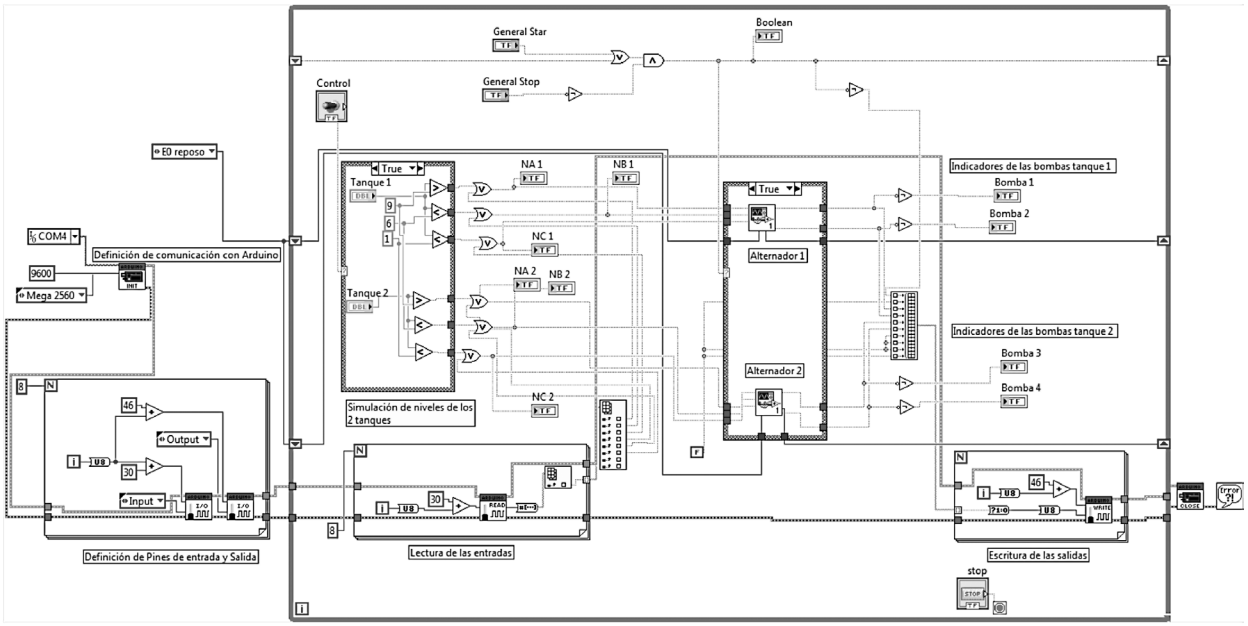



**Figura 5.** Programación del Arduino Mega usando LabView 12 de dos alternadores de bombas.

### Análisis de las implementaciones

Las implementaciones del sistema de alternadores de bombas se realizaron sin mayores inconvenientes. Ambas soluciones controlaron el sistema de eventos discretos de acuerdo a los requerimientos de funcionamiento planteados por el GRAFCET. Sin embargo, se encontró una diferencia sustancial entre ambos modos de programación. Cuando se programó en lenguaje C, el microcontrolador AMTEL es el que ejecutó el código, por tanto es ideal para aplicaciones de control en tiempo real tipo *stand-alone*; pero cuando se programó con LabView, el Arduino funcionó como una tarjeta de entradas/salidas digitales y el procesamiento de los datos y la ejecución de los algoritmos los realizó el ambiente de desarrollo en la PC.

## Discusión: Arduino vs Nano PLC

Las razones principales para utilizar Arduino en proyectos de automatización de pequeña escala se encuentran en las capacidades del *hardware* y sus costos. Se sabe que los Arduinos de gama alta poseen prestaciones superiores de procesamiento que muchos nano PLC, poseen una cantidad elevada de entradas y salidas suficiente para automatizar procesos pequeños, soportan gran número de protocolos abiertos de comunicación, existe mucha oferta de tarjetas de expansión para soportar todo tipo de sensores, etc., todo esto a un precio reducido comparado con la contraparte comercial.

Por otra parte, las razones para no utilizar los Arduinos se deben a factores tales como: el grado de criticidad del proceso productivo, las políticas de estandarización de la empresa, la poca o nula escalabilidad del proyecto a futuro, el escaso soporte sobre productos, la pobre documentación y el nulo manejo de protocolos industriales. Además, todos los lenguajes de programación se alejan de la norma IEC 61131-3, lo cual dificulta que técnicos de planta puedan brindar soporte adecuado a los algoritmos de control. Finalmente, dado que el Arduino no ha sido concebido como PLC, recae sobre el programador implementar todas las funciones comunes de los nano-micro PLC, tales como temporizadores, contadores, PID, levas horarias, etc., así como muchas otras funciones de mayor o menor nivel.

Por las razones expuestas anteriormente, cada proyecto debe valorarse cuidadosamente para ver si los productos basados en la filosofía del *open hardware* son la solución adecuada. A mediano plazo se vislumbra que estos dispositivos solventarán las deficiencias señaladas dado el rápido avance tecnológico, e incursionarán no solo en la automatización de procesos industriales sino en la instrumentación científica de laboratorios, autotrónica, domótica, etc., como un competidor más de esos mercados.

## Bibliografía

Piasim Corporation. (2014). *Annikken Andee*. Obtenido de http://www.annikken.com/

Quantum-Leaps. (2014). *Event-Driven Programming for Arduino*. Obtenido de http://www.state-machine.com/arduino/

Ainura, L. (2014). *Industrino Products*. (Industrino Corp.). Obtenido de http://www.industrino.com/products.html

Ardublocks. (2014). *A Graphical Programming Language for Arduino*. Obtenido de http://blog.ardublock.com/

Arduino. (2014a). *Arduino Mega: Overview*. Obtenido de http://arduino.cc/en/Main/arduinoBoardMega

Arduino. (2014b). *Download the Arduino Software*. Obtenido de http://arduino.cc/en/Main/Software

Arduino. (2014c). *Intel Galileo*. Obtenido de http://arduino.cc/en/ArduinoCertified/IntelGalileo

ATMEL. (2014). *ATmega2650*. Obtenido de http://www.atmel.com/devices/ATMEGA2560.aspx

Biemme-Italia. (2014). *PLC Arduino*. Obtenido de http://www.biemmeitalia.net/bmstore1x/index.php?main_page=index&cPath=1

CODESYS. (2013). *CODESYS: The Company*. Obtenido de http://www.codesys.com/company.html

Gridshield. (2014). *Open Source Hardware*. (CRCibernética). Obtenido de http://www.crcibernetica.com/

Huang, S. (2013). *CodeBlocks for Arduino*. Obtenido de http://arduinodev.com/codeblocks/

IEC-60848. (1999). *Specification language GRAFCET for sequential function charts*. 2 ed. IEC.

Jofret, B. (2013). *Arduino Communication through Serial*. Obtenido de atoms.scilab.org/toolboxes/arduino

Leung, K. (2012). *Arduino: A Brief History*. Obtenido de http://www.kenleung.ca/portfolio/arduino-a-brief-history-3/

MAAC. (2012). *Designing a replacement for an obsolete Electro Cam control system in a Maac thermoformer using a Teensy Arduino-compatible*. Obtenido de http://redbinary.com/designing-a-replacement-for-an-obsolete-electro-cam-control-system-in-a-maac-thermoformer/

Mathilde. (2013). *The State of Open Hardware Entrepreneurship 2013*. (Making Society). Obtenido de http://makingsociety.com/2013/09/the-state-of-open-hardware-entrepreneurship-2013/

Moreno, E. G. (2001). *Automatización de Procesos Industriales.* Valencia: Alfaomega.

National-Instrument. (2013). *NI LabVIEW Interface for Arduino Toolkit*. Obtenido de http://sine.ni.com/nips/cds/view/p/lang/es/nid/209835

OSHWA. (2012). *Declaración de Principios 1.0: Hardware de fuente abierta.* (OSHWA). Obtenido de http://www.oshwa.org/definition/spanish/

Picker, G. (2012). *Control an Arduino en MATLAB*. (MATLAB). Obtenido de http://blogs.mathworks.com/pick/2012/08/24/control-an-arduino-en-matlab/

Romano, Z. (2013). *Using Arduino on industrial digital printing machines*. Obtenido de http://blog.arduino.cc/2013/07/04/using-arduino-on-industrial-digital-printing-machines/#.Uzcr2fl5OSo

Rugged-Circuit. (2013). *The Ruggeduino: quick look*. Obtenido de http://www.ruggedcircuits.com/ruggeduino

Shervin, G. (2013, 11). *Debate: Arduino for industrial automation*. Obtenido de http://www.linkedin.com/groups/Arduino-industrial-automation-1268377.S.275300359

SIEMENS Corp. (2012). *SIMATIC S7-200*. Obtenido de https://www.swe.siemens.com/spain/web/es/industry/automatizacion/sce_educacion/documentacion/Documents/SIMATIC%20S71200R.pdf

SparkFun-Electronics. (2009). *Arduino Catalog*. Obtenido de https://www.sparkfun.com/categories/103?page=all

Temboo. (2013). *Get Started with Temboo*. Obtenido de https://www.temboo.com/library/

## Apéndice A

Apéndice A. CÓDIGO DE LOS ALTERNADORES

```
/*
AUTOMATIZACIÓN DE DOS ALTERNADORES DE BOMBAS.
                      _____
 Arraq Gener (A)   ----->|     |
 Pare o SC (P)     ----->|     |---> Bomba 1(B1)
 Nivel Alto (NA)   ----->|     |
 Nivel Bajo (NB)   ----->|     |---> Bomba 2(B2)
 Nivel Crtc (NC)   ----->|_____|

 Este ejemplo es una máquina de estados implementada
con la estructura CASE
 Los Pines asignados son del 30 al 37 como entradas
para sensores de nivel
 y los pines del 46 al 53 como salidas.

 Por Ing. Luis D. Murillo ,  Fecha: 16/01/14
 */
// Declaracion de variables

int NivelCritico =30;
int NivelBajo=31;
int NivelAlto=32;
int Bomba1=48;
int Bomba2 =49;
int ArrGeneral=37;
int ParGeneral=36;
int PinSalida=46;
boolean encendido=false;
int EstadoAlter1=1;
int EstadoAlter2=1;

// Configuracion de Pines de entrada y salida
void setup(){
  // initializa los pines:

  for (int i = 0; i < 8; i++) {
    pinMode((30 + i), INPUT);
    pinMode((PinSalida + i), OUTPUT);
  }
}

void loop ()
{
  if ( encendido == false &&  (digitalRead(ArrGeneral
)==HIGH))
  {
    encendido=true;
    digitalWrite(PinSalida, HIGH);
  }
  else if (encendido == true && (digitalRead(ParGener
al)==HIGH))
  {
    encendido=false;
    for (int i=0; i<8 ; i++)
    {
      digitalWrite((PinSalida + i), LOW);
    }
  }
  else if (encendido == true)
  {
  alternador (& EstadoAlter1, Bomba1, Bomba2, Nive-
lAlto, NivelBajo, NivelCritico);
  alternador (& EstadoAlter2, Bomba1+2, Bomba2+2,
NivelAlto+3, NivelBajo+3, NivelCritico+3);
  }
}
```

```
// Implementacion de la funcion alternador

void alternador(int * estado, int Bomba1, int Bomba2,
int NivelAlto, int NivelBajo, int NivelCritico ) {
 switch (*estado)
    {
    case 1:
      digitalWrite(Bomba1, LOW);
      digitalWrite(Bomba2, LOW);

      if (digitalRead(NivelBajo)==HIGH)
      {
        *estado =2;
      }
      break;
    case 2:
      digitalWrite(Bomba1, HIGH);
      digitalWrite(Bomba2, LOW);

      if (digitalRead(NivelAlto)==HIGH)
      {
        *estado =3;
      }
      else if (digitalRead(NivelCritico)==HIGH)
      {
        *estado =5;
      }
      break;
    case 3:
      digitalWrite(Bomba1, LOW);
      digitalWrite(Bomba2, LOW);

      if (digitalRead(NivelBajo)==HIGH)
      {
        *estado =4;
      }
      break;
    case 4:
      digitalWrite(Bomba1, LOW);
      digitalWrite(Bomba2, HIGH);

      if (digitalRead(NivelAlto)==HIGH)
      {
        *estado =1;
      }
      else if (digitalRead(NivelCritico)==HIGH)
      {
        *estado =5;
      }
      break;
    case 5:
      digitalWrite(Bomba1, HIGH);
      digitalWrite(Bomba2, HIGH);

      if (digitalRead(NivelAlto)==HIGH)
      {
        *estado =1;
      }
      break;
    }
}
```