UNIVERSIDADE DA CORUÑA

ESCUELA UNIVERSITARIA POLITÉCNICA

Grado en Ingeniería Eléctrica

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

TFG. Nº: **770G01A66**

TÍTULO: **SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB**

AUTOR: NÉSTOR DE JUAN VÁZQUEZ

TUTOR: FRANCISCO PRIETO GUERREIRO

FECHA: **FEBRERO DE 2015**

Fdo.: EL AUTOR

Fdo.: EL TUTOR

TÍTULO: **SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB**

ÍNDICE GENERAL

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBRERO, S/N

15405 - FERROL

FECHA: **FEBRERO DE 2015**

AUTOR: EL ALUMNO

Fdo.: NÉSTOR DE JUAN VÁZQUEZ

1. MEMORIA

TÍTULO: **SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB**

MEMORIA

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBRERO, S/N

15405 - FERROL

FECHA: **FEBRERO DE 2015**

AUTOR: EL ALUMNO

Fdo.: NÉSTOR DE JUAN VÁZQUEZ

Índice

Índice de figuras

Índice de tablas

1. OBJETO

El objetivo del consiguiente trabajo reunirá la introducción a los conceptos básicos, el diseño y la posterior implementación de un SCADA desarrollado empleando el lenguaje informático Visual Basic (VB), para la realización del control electrónico de un acuario mediante el empleo de una tarjeta Arduino MEGA.

El desarrollo del trabajo será tal que cumpla con todos los objetivos descritos en el alcance del consiguiente trabajo, así como las especificaciones deseadas y establecidas como requisitos propios del diseño. Todo lo posteriormente redactado permitirá profundizar en las diversas fases del proceso de desarrollo del mencionado trabajo, desde la fase de diseño inicial hasta la calibración de los dispositivos que forman parte del sistema e implementación finales.

2. ALCANCE

Este trabajo se marca como objetivo principal el diseño y desarrollo de un acuario controlado a través de un SCADA, con opción de llevar a cabo de forma remota a través de la web. Por ello, a lo largo del mismo se tratarán los puntos descritos a continuación:

- Introducción al concepto de SCADA y a su funcionalidad dentro de un sistema de control automatizado.
- Enumeración de los elementos que componen la arquitectura de la planta, definiendo sus funcionalidades y características principales.
- Diseño del SCADA y conectividad de la tarjeta de adquisición de datos con el mismo para llevar a cabo la transmisión bidireccional de información.
- Diseño de la comunicación del sistema con la red y de la página web empleada como enlace para el control remoto del mismo.
- Calibración y prueba del funcionamiento de los elementos del sistema de forma conjunta.
- Implementación física del sistema, con explicación del conexionado y del montaje físico realizado.

3. ANTECEDENTES

El concepto de “acuario electrónico” ya ha sido desarrollado previamente, en mayor o menor magnitud, por otras personas. El aplicar el control electrónico a un entorno de este tipo facilita enormemente su mantenimiento a largo plazo y coste, entre otras diversas características, acompañando en este caso como parte importante todo lo relacionado con el aislamiento y la seguridad eléctrica, al hablar de un entorno en el que hay una presencia constante de líquido.

Lo que se desarrolla a lo largo de este trabajo es la aplicación de dicho concepto en la práctica, mediante el empleo de una tarjeta de adquisición de datos y un software SCADA, que permitan al usuario el control del acuario de forma manual o automática y puedan proporcionarle los datos más importantes acerca del mismo (temperatura, acidez,...) con el objetivo de permitirle tomar la decisión más adecuada. Además, el concepto de control remoto a través de Internet permite también desarrollar un método de control alternativo que evite la presencia constante del usuario si elige un método de supervisión manual.

La decisión de haber realizado este trabajo está basada en la idea de convertir un entorno común y conocido en muchas viviendas como puede ser un acuario en un sistema controlable electrónicamente, permitiendo hacer así práctico su control por parte del dueño o usuario del mismo.

3.1. La electrónica en el cuidado de animales

Coetáneamente al momento en el que el ser humano comenzó a emplear la electrónica para facilitar su vida aumentando considerablemente su comodidad, aplicó la tecnología disponible para simplificar toda clase de tareas con mayor o menor grado de complejidad, permitiendo que estas se realicen de forma automática sin la intervención del mismo. Dentro de este ámbito, podemos incluir todas aquellas actividades relacionadas con los animales.

El reino animal siempre ha tenido una gran importancia para el ser humano, tanto desde el punto de vista de la alimentación, el más importante, como desde el correspondiente a la capacidad de trabajo que ofrecen para poder realizar tareas engorrosas, imposibles o de peligrosas características para nosotros mediante su fuerza física, como es el caso claro de la agricultura.

Figura 3.1.1 - Proceso de ordeño automático en vacas lecheras

Por ello, es vital que animales con destinos como los mencionados estén en las mejores condiciones físicas posibles, para lo cual en algunos casos puede ser necesario un cuidado constante por parte del ser humano, que puede llegar a ser extenuante para el mismo.

En este punto, y gracias al desarrollo de la tecnología, se facilita enormemente la supervisión de los mismos gracias a sistemas automatizados que permiten, entre otras cosas, alimentar a los animales de forma adecuada o permitir localizarlos fácilmente en caso de que se desconozca su posición (por ejemplo, la extracción de leche de las vacas y la localización vía GPS de las mismas mediante dispositivos de localización cuando se hallan pastando a cielo abierto).

En la actualidad, en este campo la electrónica también se ha abierto a otros casos en el cuidado de animales relacionados bien con la conservación de ejemplares de los mismos (reservas naturales, zoológicos) o con las mascotas domésticas, como los dos ejemplos más claros.

Figura 3.1.2 - UCI para animales desarrollada para veterinaria

En este último caso, destaca especialmente la proliferación cada vez mayor de usuarios que deciden emplear la tecnología para facilitar el cuidado de sus animales. Uno de las situaciones más frecuentes es el cuidado de peces, engorroso para sus dueños hasta el punto de que necesitan una gran atención en comparación con otros animales, y que se ve considerablemente facilitado gracias al empleo de la electrónica. En este último ámbito es en el que se centra este trabajo.

3.2. Control automatizado mediante SCADA

La automatización de sistemas ha sido uno de los principales objetivos en el ámbito del trabajo del ser humano desde la Edad Antigua. Las primeras máquinas simples facilitaban la realización de tareas que requerían un importante esfuerzo humano sin llegar a eliminar la intervención de éste en el proceso, disminuyendo considerablemente la tarea realizada por el mismo a través de diversos sistemas, tales como los sistemas de poleas o el tornillo de Arquímedes, entre otros. Desde este momento, y con cada avance en el campo de la tecnología, la automatización de procesos siguió evolucionando a ritmo lento hasta la llegada del siglo XX.

Es en este periodo cuando comienzan a desarrollarse gracias a la electricidad y a la electrónica los primeros sistemas automatizados en el ámbito industrial, así como posteriormente los primeros robots industriales, ambos sucesos destinados a liberar al ser humano de tareas peligrosas o imposibles de realizar por sí solo.

Gracias a estos avances, una planta puede ser controlada de forma automática, e incluso autorregularse mediante el empleo de los denominados reguladores, sin apenas intervención del ser humano a lo largo del proceso.

Figura 3.2.1 - Planta automatizada

La aparición del computador permitió ir un paso más allá en el control de sistemas con la aparición de los denominados sistemas controlados por computador. Y es en este campo en el que nace el concepto de SCADA (definido en el apartado 5), y que permite, a través de una interfaz hombre-máquina o HMI, que cualquier usuario sin grandes conocimientos informáticos pueda controlar un gran sistema sin especial dificultad, pudiendo visualizar en pantalla y en tiempo real todos los datos significativamente importantes de la planta y actuar sobre los actuadores existentes en la misma cambiando el proceso según sus designios.

Así, se considera que un SCADA puede permitir el control de cualquier tipo de sistema en el que sea necesario llevar a cabo un seguimiento y control de variables a lo largo del tiempo, como es el caso del diseñado en este trabajo.

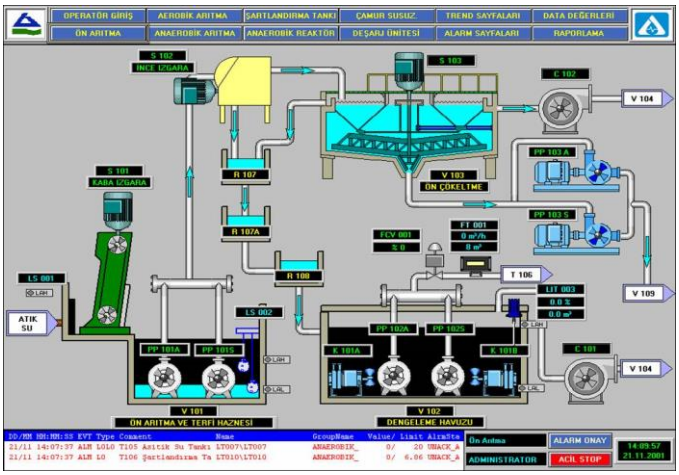


Figura 3.2.2 - Ejemplo de SCADA diseñado para entorno industrial

3.3. Visual Basic y Arduino

Como se describe posteriormente en el apartado 5, tanto Visual Basic como Arduino son dos potentes herramientas, la primera en el ámbito de la programación y de mayor antigüedad que la segunda, y ésta última en el ámbito de la adquisición y el tratamiento de datos, que de forma conjunta ofrecen una gran cantidad de posibilidades al usuario a la hora de programar y desarrollar gran cantidad de sistemas.

Para la creación de un SCADA existen una gran cantidad de lenguajes informáticos en el mercado que permitan desarrollar entornos gráficos. Entre los más conocidos, se pueden emplear lenguajes como Vijeo Citect (Siemens), Matlab (MathWorks), Unity o Visual Basic (Microsoft), el empleado en este caso.

Visual Basic ofrece un entorno de programación muy simple, con un lenguaje de alto nivel que ofrece gran cantidad de posibilidades y que puede resultar muy familiar para la mayoría de los usuarios del mundo de la informática, debido a que es el lenguaje en el que están basadas las aplicaciones gráficas de Windows.

En el otro lado, y como también se definirá más adelante, Arduino es una potente herramienta para la adquisición y el manejo de datos y aplicaciones, ofreciendo unas buenas características técnicas a un coste más que razonable.

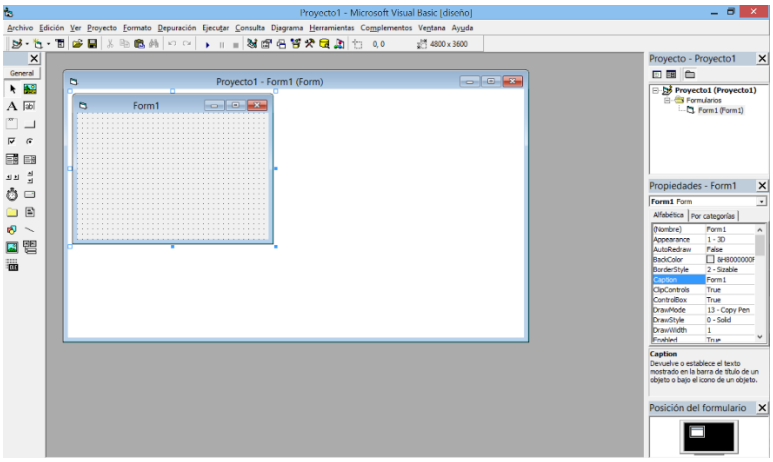


Figura 3.3.1 - Entorno estándar de programa en Visual Basic 6.0

Debido a que el primero es un lenguaje de programación muy extendido y que el segundo ha sufrido una gran expansión en los últimos años, se han desarrollado gran cantidad de proyectos que se basan en la conectividad y en el intercambio de datos entre uno y otro. El sistema diseñado no es más que otro ejemplo a añadir a la lista de los ya realizados en el ámbito mencionado.

3.4. Control remoto por web

Internet es uno de los mayores fenómenos, sino el que más, en el ámbito de la informática y las comunicaciones desarrollado en los últimos años del siglo XX. Entre sus principales funciones, permite a sus usuarios conversar en tiempo real mediante el empleo de servidores de chat, subir y descargar todo tipo de archivos, y emplear el servicio World Wide Web (WWW) para la visualización de páginas web entre otros tantos. El relacionado con este trabajo hace referencia al control remoto de sistemas, útil tanto a nivel industrial como doméstico.

Tradicionalmente, las comunicaciones mediante empleo de SCADA han sido del tipo punto-multipunto en comunicación serie a través de estándares de comunicación como RS-232, RS-485 o BEL-202 entre otros. Con la llegada del Protocolo de Internet o IP, la tecnología relacionada con Internet ha visto incrementado su funcionalidad con el empleo de comunicaciones con SCADAs. Así, un SCADA basado en un entorno web hace uso de la tecnología IP y permite, mediante los estándares empleados para la comunicación a través de la red (fibra óptica, RJ-45, redes inalámbricas), conectar el sistema de adquisición de datos

empleado a un SCADA diseñado íntegramente en web a través de un determinado servidor web.

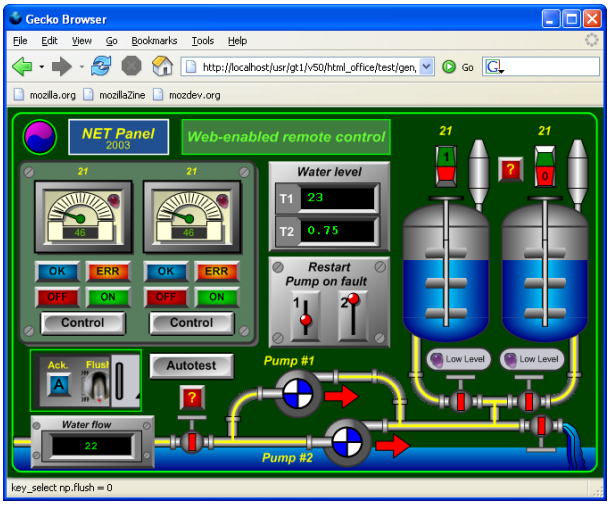


Figura 3.4.1 - Ejemplo de SCADA web

Entre las principales ventajas que se pueden enumerar en cuanto al uso de un SCADA web destacan:

- Amplio rango de direccionamiento.
- Amplia área de conectividad.
- Estandarización.
- Integración de Internet a las redes de automatización.

Para poder desarrollar un SCADA del tipo mencionado, es vital que el sistema de adquisición de datos sea capaz de permitir una comunicación en red. Para esta función existen tarjetas preparadas específicamente para esta función, o bien disponen de módulos que permiten expandir sus capacidades.

4. NORMAS Y REFERENCIAS

En el siguiente capítulo se describen todas las normas y referencias a las que se ha hecho referencia en el trabajo, así como la bibliografía consultada para la realización del mismo.

También serán incluidos en este capítulo aquellos programas de cálculo que hayan sido empleados para la realización de los mismos.

4.1. Disposiciones legales y normas aplicadas

En el consiguiente trabajo se han empleado y/o seguido una serie de normas y referencias acordes con lo realizado en el mismo. A continuación se mencionan dichas normativas:

- *IEC 60529: Grados de protección proporcionados por las envolventes (Código IP)*. Fecha de edición: 2006-01-10. Documento nacional: UNE 20324:1993.
- *768/2008/CE*: *Marcado de conformidad CE*. Fecha de edición: 2008-09-07.
- *2011/65/UE: RoHS (Restriction of Hazardous Substances), sobre restricciones a la utilización de determinadas sustancias peligrosas en aparatos eléctricos y electrónicos.* Fecha de edición: 2011-08-06. Documento nacional: Real Decreto 219/2013.
- *UNE-EN ISO 80000-2: Magnitudes y unidades. Parte 2: Signos matemáticos y símbolos matemáticos que se utilizan en las ciencias naturales y en la tecnología*. Fecha de edición: 2013-11-27.
- *GEMMA: Guide d'Etude des Modes de Marches et d'Arrets*. Desarrollada por la ADEPA (Agence nationale pour le Developpement de la Productique Appliquée à l'industrie). Fecha de edición: 1993.

4.2. Bibliografía

[1] NICOL, N; ALBRECHT, R. *Todo sobre Visual Basic 6*. Traducido por JL. Cortés Díaz. 1ª ed. Barcelona: Marcombo S.A, 1999. 256p. ISBN: 8426712312; ISBN-13: 9788426712318.

[2] LAJA VIZCAÍNO, JR; PELEGRÍ SEBASTIÁ, J. *Sistemas integrados con arduino*. 1ª ed. Barcelona: Marcombo S.A, 2011. 320p. ISBN: 9788426720931.

[3] PÉREZ GARCÍA, MA; ÁLVAREZ ANTÓN, JC; CAMPO RODRÍGUEZ, JC. *Instrumentación electrónica.* 1ª ed. Madrid: Ediciones Paraninfo S.A, 2003. 862p. ISBN: 9788497321662.

[4] "Propiedades y eventos del objeto MSComm", [en línea]. Junio 2000. Disponible en la web: http://uttinfor.tripod.com/index/id4.html.

[5] "Arduino – Home", [en línea]. Disponible en la web: http://arduino.cc.

[6] "Arduino Ethernet Shield Web Server Tutorial", [en línea]. Enero 2013. Disponible en la web: http://startingelectronics.com/tutorials/arduino/ethernet-shield-web-server-tutorial.

[7] "Arduino Tutorial – Lesson 4 – Serial communication and playing with data", [en línea]. Disponible en la web: http://www.ladyada.net/learn/arduino/lesson4.html

[8] "Tutoriales sobre HTML y CSS – Construye tu propio sitio web", [en línea]. Disponible en la web: http://es.html.net.

[9] "Wikipedia", [en línea]. Disponible en la web: http://es.wikipedia.org.

[10] "PH meter (SKU: sen0161)", [en línea]. Disponible en la web: http://dfrobot.com/wiki/index.php/PH_meter%28SKU:_SEN0161%29

[11] ATMEL: *Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V*. Atmel, 2014.

[12] MAXIM: *DS18B20 Programmable Resolution 1-Wire Digital Thermometer*. Maxim Integrated Products Inc, 2008.

[13] CEBEK COMPONENTS: *Sensores de nivel de líquidos C-7235/C-7236*. Cebek Components, 2008.

4.3. Programas empleados

En este apartado se indicará la relación de los diversos programas y herramientas utilizadas para desarrollar los diversos cálculos y diseños del TFG, así como la escritura de los diversos programas realizados.
Dicha lista consta de los siguientes programas:

- Microsoft Excel 2013.
- OrCAD 16.6.
- Layout Plus.
- Visual Studio 6.0.
- IDE de Arduino.

El primer programa de la lista ha sido empleado para la realización de tablas, presupuestos y gráficas. Los programas OrCAD y Layout Plus, ambos de la compañía PsPice; así como AutoCAD, han sido empleados para el diseño de circuitos y placas de circuito impreso y para la realización de planos.

5. DEFINICIONES Y ABREVIATURAS

En el consiguiente apartado se definirán los conceptos básicos y abreviaturas empleados a lo largo del documento que tengan algún tipo de relación con el trabajo.

5.1. Abreviaturas

- SCADA: Supervisory Control And Data Adquisition (Control de Supervisión y Adquisición de Datos).
- DAQ: Data Adquistion (Adquisición de datos).
- TAD: Tarjeta de Adquisición de Datos.
- VB: Visual Basic.
- IP: Internet Protocol (Protocolo de Internet).
- MAC: Media Access Control (Control de Acceso al Medio).
- HTML: Hyper Text Markup Language (Lenguaje de marcación de Hipertexto).

5.2. Definiciones

- SCADA: aplicación software de control de producción, tal que se comunica con los dispositivos de campo y permite el control del proceso de forma automática desde la pantalla del computador a través de una interfaz gráfica. Proporciona información del proceso a diversos usuarios: operadores, supervisores de control de calidad, supervisión, mantenimiento, etc. Para poder ser realizado, un SCADA debe ser programado en algún tipo de lenguaje informático, habiendo múltiples opciones en este caso.
- DAQ constituye el proceso de medir, mediante un sistema de adquisición de datos, un determinado fenómeno del entorno con el objetivo de convertirlo en una señal eléctrica de voltaje o corriente para que pueda ser interpretada. Un sistema DAQ consta de una serie de dispositivos de campo (sensores y actuadores), un hardware de adquisición de datos o TAD y un computador para el control.

Figura 5.2.1 - Proceso de transmisión de datos en DAQ

Comparados con los sistemas de adquisición más tradicionales, los sistemas DAQ basados en computador aprovechan la potencia del procesamiento, productividad, visualización y habilidades de conectividad de los ordenadores estándares en la industria proporcionando una solución de medidas más potente, flexible y rentable.

- Tarjeta de adquisición de datos: hardware necesario para la adquisición y tratamiento de los fenómenos del entorno que se deseen conocer o medir, actuando como puente entre el proceso a controlar y del que obtener datos y el computador, necesaria en el proceso de DAQ.

- Baudio: en la transmisión de datos, puede definirse como unidad de medida la cual hace referencia al número de veces que cambia el estado del medio de transmisión por segundo. Es necesario considerar que en algunas ocasiones cada cambio de estado puede afectar a más de un bit, por lo que no debe confundirse con éste, la unidad de información más pequeña en el ámbito informático.
- Velocidad de transmisión: también denominada tasa de transferencia, magnitud que tiene como unidad el baudio, y que permite medir el número de intervalos de información transmitidos en una línea. En forma de ecuación, la tasa de baudios se expresa con el baudio como unidad, tal que:

$$Rs = \frac{nº\ bits}{s} = baudios \tag{5.2.1}$$

Como se ha comentado en la anterior definición, un baudio puede no representa únicamente un bps, sino que puede hacer referencia a una señal con más de uno. Por ello, la tasa de baudios (Rs) es siempre menor o igual a la tasa de bits por segundo (Rb) de un proceso:

$$Rs \leq Rb \tag{5.2.2}$$

- Lenguaje de programación: lenguaje artificial que puede ser empleado para controlar el comportamiento de una máquina, especialmente una computadora, y el cual está compuesto de un conjunto de reglas sintácticas y semánticas que permiten expresar instrucciones que posteriormente serán interpretadas.

  Debe distinguirse del concepto de “lenguaje informático”, definición más amplia que incluye lenguajes de formato de texto y que por ello no deben considerarse programación en sí misma.
- Visual Basic: lenguaje de programación basado en eventos desarrollado por Microsoft y empleado antiguamente por esta compañía hasta su última versión estable, la 6.0. Permite el desarrollo de aplicaciones en un entorno

gráfico de manera sencilla a través del empleo de objetos y los eventos asociados a éstos.

Dentro de éste, cabe mencionar una serie de definiciones asociadas a dicho lenguaje para facilitar el entendimiento del código[1]:

- Formulario: nombre empleado para definir cada una de las ventanas existentes en el programa diseñado, pudiendo considerarse como una especie de contenedor para los controles. Una aplicación puede tener varios formularios en función de la complejidad de la misma, pero un único formulario puede ser suficiente para las aplicaciones más sencillas.
- Control: también llamado de forma genérica en programación orientada a objetos como clase, se concibe como una entidad genérica de la que puede haber varios ejemplares concretos en cada programa. Cada tipo de control tienen un conjunto de propiedades que definen tanto su aspecto gráfico (tamaño, color, posición en la ventana, tipo y tamaño de letra, etc.) como su forma de responder a las acciones del usuario (si está activo o no, por ejemplo). En la siguiente tabla se muestran los controles existentes en Visual Basic:

Abreviatura	Control	Abreviatura	Control
chk	check box	cbo	combo box
cmd	command button	dir	dir list box
drv	drive list box	fil	file list box
frm	form	fra	frame
hsb	hor. scroll bar	img	image
lbl	label	lin	line
lst	list	mnu	menu
opt	option button	pct	pictureBox
shp	shape	txt	text edit box
tmr	timer	vsb	ver. scroll bar

Tabla 5.2.2 - Abreviaturas para los controles más usuales en Visual Basic

- Objeto: cada ejemplar de un control determinado (por ejemplo, dentro de la clase CommandButton, en un programa pueden existir

varios objetos de esta clase denominados CommandButton1, CommandButton2 y así).

- Ethernet: también conocido como IEEE 802.3, estándar de transmisión de datos para redes de área local para computadores, con acceso al medio por detección de la onda portadora con detección de colisiones (CSMA/CD). Se pueden distinguir diferentes variantes de tecnología Ethernet según el tipo de enlace físico empleado así como la comunicación entre los equipos de la red: 10Base2, 10Base-T, 10Base5,…

  Las redes Ethernet se caracterizan por poseer una topología determinada, la cual al igual que el tipo de enlace físico establece diversos tipos de comunicación. Existen diversas variantes, y algunas de ellas pueden incluir componentes extras a las redes, como es el caso de los conmutadores en las redes en estrella.
- WiFi: nombre comercial dado para el estándar IEEE 802.11 o WLAN, se define como un estándar de comunicación inalámbrica mediante ondas, permitiendo a todo tipo de dispositivos electrónicos que tengan habilitada este tipo de comunicación conectarse a Internet a través de un punto de acceso de red inalámbrica. El acceso a dicha red puede ser codificado mediante el empleo de diversos protocolos de cifrado de datos tales como WEP, WPA o WPA2 entre otros.
- IP: literalmente Protocolo de Internet, es parte del conjunto de protocolos TCP/IP, el cual permite el desarrollo y transporte de paquetes de datos. El protocolo IP determina el destinatario del mensaje mediante el empleo de tres campos:
  - Campo de dirección: hace referencia a la dirección del equipo. Existen diversos tipos o clases de dirección IP según el rango en el que se muevan.
  - Campo de máscara de subred: permite al protocolo IP establecer la parte de la dirección IP que se relaciona con la red.
  - Campo de puerta de enlace predeterminada: permite al protocolo saber a qué equipo debe enviar un paquete, si el equipo de destino no se encuentra dentro de la red de área local.
- Dirección MAC: identificador numérico de 6 bytes el cual se corresponde de forma única a una tarjeta o interfaz de red. Es de carácter individual

para cada dispositivo, siendo determinados los últimos 24 bits por la IEEE y los primeros 24 por el fabricante mediante el empleo del OUI (Identificador Único de Organización).

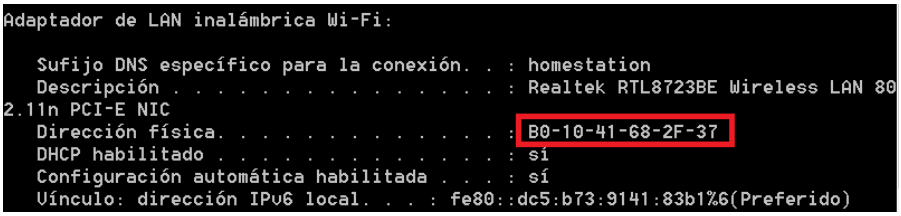


Figura 5.2.2 - Ejemplo de dirección MAC en un ordenador

Cada dirección MAC es única a nivel mundial, puesto que son escritas directamente y de forma binaria en el hardware en el instante de su fabricación.

- HTML: lenguaje informático de marcas de texto utilizado normalmente como lenguaje de diseño de la World Wide Web. Basado en el concepto de Hipertexto y el SGML o Lenguaje Estándar de Marcación General. Se caracteriza por no poseer un compilador propio, por lo que los errores, si existen, no se detectarán a la hora de presentarse gráficamente la página web.

  El entorno de trabajo de HTML es simplemente un procesador de texto cualquiera. Desde su creación HTML ha tenido varias versiones evolucionadas desde el primer estándar, como HTML5.

6. REQUISITOS DEL DISEÑO

Será necesario establecer una serie de requisitos a cumplir en el diseño del sistema, y que determinarán el funcionamiento tanto de éste en su conjunto como de cada uno de sus componentes de forma individual.

A continuación se describen los requisitos mencionados:

- La alimentación de los distintos sensores y relés conectados a los actuadores que se emplearán en el sistema será de +5V, tensión que estará proporcionada por la TAD empleada.

- Se controlarán de forma remota como variables básicas de control la temperatura, acidez y nivel del acuario, a través de los sensores correspondientes.
- Se podrá actuar sobre la acción de alimentar a los animales del acuario y la modificación de la acidez del agua mediante el empleo de un servomotor así como sobre el resto de actuadores del sistema, tanto desde el SCADA principal como de forma remota.
- Se diseñará el SCADA en Visual Basic con dos modos de operación: manual para una constante supervisión por parte del usuario y automático para una mayor liberación de la acción del supervisor humano.
- El SCADA permitirá que el control del acuario sea llevado de forma tanto manual como automática a través del ordenador por parte del usuario, que podrá elegir entre ambos modos de funcionamiento.
- La tarjeta de adquisición de datos se comunicará con un servidor web diseñado exclusivamente para el control del SCADA online, con un periodo de refresco independiente del intervalo de muestreo del SCADA en VB.

7. ANÁLISIS DE LAS SOLUCIONES

A continuación, se analizarán las diversas opciones escogidas en cada uno de los ámbitos claves del trabajo, exponiendo las razones por las cuáles dicha alternativa ha sido considerada como la más adecuada.

7.1. Tarjeta de adquisición de datos

La elección de una adecuada tarjeta de adquisición de datos para el sistema a diseñar es una tarea importante, ya que determinará el funcionamiento de toda la planta y deberá permitir una comunicación con el SCADA fluida y continua así como una correcta adquisición de datos del entorno. Por ello, en los siguientes apartados se definirán las razones por las cuales se ha elegido la TAD empleada en el consiguiente trabajo.

7.1.1. Modelo de tarjeta elegida

A continuación, se expondrán de forma resumida las distintas alternativas o posibilidades existentes en el mercado en el campo de las tarjetas de adquisición de datos.

- RaspBerry Pi B+: se trata de una tarjeta preparada para funcionar como un ordenador, pudiendo considerarse un PC en miniatura. Basada en el procesador Broadcom BCM2835, posee conexiones tales como salida de vídeo, lector de tarjetas, salida de sonido o conexión Ethernet entre otras. A su favor cuenta con el gran potencial de procesamiento que posee, un precio muy asequible y una elevada memoria. Sin embargo, para la aplicación a realizar podría suponer un sobredimensionamiento del sistema, ya que no es una placa destinada exclusivamente a la adquisición de datos. Su número de entradas/salidas no es muy elevado en comparación con otras alternativas y no presenta un lenguaje de programación tan sencillo y enfocado a los procesos tratados en este trabajo.
- Arduino MEGA 2560: es una tarjeta de adquisición de datos basada en el microcontrolador ATmega2560[11]. Tiene 54 entradas/salidas digitales (de las cuales 14 permiten generar una señal PWM), 16 entradas digitales, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz y conexión USB entre otras características, las cuales demuestran ser perfectamente compatibles con el sistema a diseñar y con los requisitos del diseño. Además de sus diversas ventajas frente a otras tarjetas del campo, como su menor coste, permite el “acoplamiento” de diversos shields que expanden sus funciones, como conectividad con Ethernet o control de dispositivos inductivos.

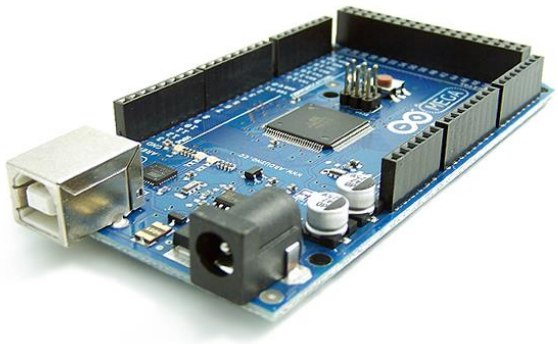


Figura 7.1.1.1 - Tarjeta Arduino MEGA 2560

- Otras variantes: además de las dos mencionadas, existe en el mercado una amplia gama de TADs que permiten llevar a cabo procesos de adquisición de datos como el realizado en este trabajo. Sin embargo, sus costes considerablemente más elevados respecto a las anteriores alternativas, así como el desconocimiento total de los lenguajes de programación que emplean en el ámbito del aprendizaje, hacen que no sean alternativas reales a tener en cuenta como solución final.

Por las razones mencionadas, se ha elegido esta placa como tarjeta de adquisición de datos para la planta en cuestión. En concreto, la versión que se utilizará en este trabajo de la tarjeta en cuestión será la 2560, debido a que posee ventajas como un elevado número de entradas y salidas entre otras.

7.1.2. Velocidad de transmisión

A la hora de diseñar un sistema como el que constituye el objetivo de este trabajo, es muy importante determinar de forma adecuada la velocidad de transmisión que va a existir en la comunicación bidireccional existente entre la tarjeta de adquisición de datos, que se encarga de recibir la información de la planta a controlar, y entre el SCADA empleado para controlar el mismo.

En el sistema presente, la velocidad de transmisión estará determinada y limitada por la permitida por el lenguaje de programación elegido, VB (en el siguiente apartado se determina su elección), que es el que permite la velocidad de transmisión de menor valor entre éste y Arduino. El objeto que determina dicha magnitud y su propiedad asociada son MSComm[3] y Settings, respectivamente.

Concretamente, dicho objeto permite comunicaciones por el puerto serie a baud rates de 50, 100, 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200 y 28800 baudios. El valor máximo de la velocidad en baudios teórica permitida por la tarjeta de adquisición de datos empleada partiendo de la documentación del procesador de la tarjeta[11], la cual ha sido calculada en el anexo correspondiente de cálculos, es de $1 \cdot 10^6$ baudios o bps, tasa muy superior a la máxima permitida por el lenguaje base del SCADA diseñado.

Además de atender a todos estos datos, es necesario comprender también cuáles son las necesidades del sistema a tratar. Se trata de una planta que envía y recibe una cantidad razonablemente baja de señales, debido a que no es de dimensiones considerables ni existe una gran cantidad de dispositivos de campo con los que comunicarse. Debido a este volumen de información transferida tan pequeño, una velocidad baja sería más que suficiente para permitir una comunicación fluida. Esto puede demostrarse numéricamente atendiendo a la equivalencia entre caracteres y bytes del código ASCII, el cual muestra la representación de caracteres a través de combinaciones de bits.

Así, y según éste último, cada carácter está codificado con 8 bits (1 byte), aunque al ser enviados a través del puerto serie Arduino se les añade un bit de comienzo y un bit de fin de carácter, por lo que realmente cada carácter supone 10 bits en la comunicación serie. Por otro lado, se están enviando 10 caracteres de información y se deben añadir los caracteres de retorno de carro y salto de línea a razón de la instrucción empleada por Arduino para el envío (println), que al igual que el resto ocuparán 10 bits. Por lo tanto, el número total de caracteres enviados será:

nº caracteres = 10 caracteres + CR (retorno de carro) + LF (salto de línea) =
nº caracteres = 12 caracteres

Si cada carácter está constituido por 10 bits, se estarán enviando 120 bits de cada vez. Tal y como se puede observar más abajo en la tabla 7.1.3.1, el valor medio de ejecución de lo contenido en el Timer es de 780 ms, lo que deja 220 ms libres para realizar el envío de los datos antes de que vuelva a pasar el periodo de desbordamiento del temporizador. Según los cálculos realizados en el apartado

2.1 del anexo Cálculos, será necesaria como mínima una tasa de transferencia de datos de:

$$b_r \geq 546\ baudios \qquad (7.1.2.1)$$

De esta forma, cualquier tasa de transferencia superior a los 546 baudios que se escoja será adecuada para la comunicación. Además, pueden escogerse tanto valores cercanos como elevados para dicha tasa, ya que ello no supone un consumo de recursos adicional y de magnitud para el sistema.

Las pruebas realizadas en el sistema dan la razón a los cálculos: la velocidad inmediatamente menor a la calculada, 300 baudios, impide que se puedan transmitir los datos a tiempo. Para la velocidad inmediatamente mayor a la obtenida, 600 baudios, la transferencia se lleva a cabo correctamente. En concreto, para el sistema diseñado se ha escogido una tasa de 9600 bps para la cual y según lo reflejado en el anexo de cálculos, se tardarían 13.5 ms en transmitir la información. Se ha elegido dicho valor por el hecho de ser uno de los más empleados en proyectos realizados con Arduino que tengan que ver con la comunicación serie. Además, se encuentra dentro de los recomendados por la página de Arduino[5] para las comunicaciones de este tipo y es un valor aceptado por la propiedad Settings del objeto MSComm en el lenguaje de programación empleado.

Para lograr que la conexión sea estable y la información transmitida sea correcta, se igualarán los valores de las velocidades de transmisión en el puerto serie en el SCADA y en la tarjeta de adquisición de datos. Tanto uno como otro entorno de programación, además, tienen configurada por defecto la comunicación con tramas de datos de 8 bits y un bit adicional de paro.

7.1.3. Periodo de muestreo

El periodo de muestreo es un concepto muy importante en cualquier aplicación de control, ya que determinará el tiempo existente entre la toma de dos muestras consecutivas por parte del sistema de adquisición de datos. Su valor es establecido en esta aplicación dentro del código de la placa Arduino y en el

SCADA a través del lenguaje de programación, pero es necesario llevar a cabo una serie de consideraciones.

En cuanto al valor numérico elegido, se ha tenido en consideración el hecho de que, en un entorno como el diseñado, la evolución temporal de las señales a muestrear es pequeña, y además éstas no sufren cambios bruscos de valor. Por ello, se ha definido un periodo de muestreo o intervalo de toma de datos de 1 min de mínimo y 1 h de máximo, elegible en el panel de personalización del usuario tanto en modo manual como en automático.

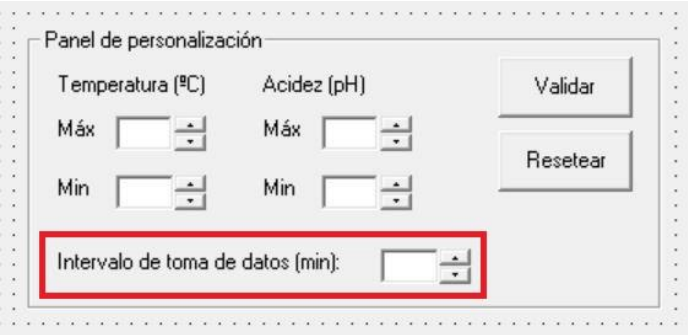


Figura 7.1.2.1 - Elección de periodo en panel personalizable del SCADA

En cuanto al tiempo de ejecución del programa de la TAD, las instrucciones que realiza poseen su propio tiempo de ejecución, variable en función de si son funciones o instrucciones simples, y a su vez en el primer caso dependiendo de qué instrucciones contengan. Ello influye, en programas con un código especialmente extenso, en el tiempo que tarda el sistema en poder realizar el muestreo.

En este caso, antes de elegir la forma según la cual se llevará a cabo el muestreo del sistema, se han realizado tomas del tiempo de ejecución normal del código, para evaluarlos y poder determinar la precisión con la que podemos establecer el periodo buscado. Para ello, se ha empleado la función micros(), que se puede ver a continuación, y que permite determinar el tiempo de ejecución del programa en microsegundos hasta que se alcanza dicha función de nuevo:

Tiempos de ejecución		
Nº de ciclo	Tiempo (us)	Diferencia (us)
1	1267000	-
2	2046772	779772
3	2826540	779768
4	3606308	779768
5	4386076	779768
6	5165840	779764
7	5945608	779768
8	6725376	779768
9	7505144	779768
10	8284904	779760

Tabla 7.1.3.1 - Medidas de los tiempos de ejecución del script

Como se puede observar, el tiempo de ejecución de la función de muestreo corresponde aproximadamente a unos 0.78s, valor elevado debido sobre todo a la lentitud de respuesta del sensor de temperatura, el DS18B20.
Partiendo de los datos de la tabla y tal y como se demuestra extendido en el anexo Cálculos (todos los resultados a continuación mostrados se encuentran desarrollados en dicho documento), se calculará la media del tiempo de ejecución, atendiendo a dos condiciones en el cálculo:

- El primer valor es distinto a los anteriores al ser la primera "vuelta" de ejecución del código.

$$\Delta T_e \cong 779767\ \mu s \qquad (7.1.3.1)$$

Con este resultado, puede ser calculado el valor del retardo a introducir en el sistema para obligarlo a igualar el valor de 1sg mencionado anteriormente, tal que:

$$delay \cong 220\ ms \qquad (7.1.3.2)$$

A priori, bastaría con introducir este valor en la función delay() para obtener el periodo buscado. Haciendo la prueba, se puede observar que la diferencia escogiendo dos tiempos cualesquiera consecutivos cerca del inicio del programa es de:

$$\Delta T_e = 999780\ \mu s \tag{7.1.3.3}$$

Como se puede observar, es un resultado muy cercano al periodo de muestreo buscado, y que únicamente causa un error relativo del:

$$e_r = 0.022\ \% \tag{7.1.3.4}$$

Sin embargo, existe un pequeño problema. A medida que el funcionamiento del programa se extiende a lo largo del tiempo, la diferencia existente entre el periodo real obtenido mencionado arriba y el periodo ideal buscado (1sg) se observa que no es constante, obteniendo valores por encima y por debajo del obtenido, lo que provoca que haga al sistema inestable. Además, el tiempo de ejecución dependerá de si la función de muestreo tarda mayor o menor tiempo en ejecutarse, ya que ésta tiene un periodo de duración fijo. Por ello, y tal como recomiendan diversos autores, el uso de la función delay(), así como de su variante más precisa, delayMicroseconds(), es inviable a la larga.

Por ello, se ha escogido la opción de implementar un Timer. Estudiando la documentación del microchip implementado en la placa Arduino[11], el ATMega2560, se puede observar que el mismo dispone de 6 temporizadores/contadores. En el apartado de cálculos queda reflejado que la opción elegida para la temporización del muestreo será el Timer1, debido a que por su condición de tener 16 bits permite configurar el reloj con una frecuencia de 1Hz. Hay que tener en cuenta antes de continuar que el Timer no permite conseguir una temporización ni siquiera del mínimo elegible por el usuario (1 min), ya que su registro no admite un valor de comparación tan alto; por ello se usará una variable auxiliar para contar las interrupciones producidas cada segundo, y así poder emplear el tiempo de muestreo que se desee del intervalo empleado.

En el ámbito de la programación, tener que programar todos los registros del Timer así como controlar las interrupciones podría resultar muy engorroso. Por ello, se ha decidido escoger la opción de emplear la librería TimerOne, descargable desde la página de Arduino y que simplifica la programación considerablemente. El siguiente código muestra la programación sin la librería, en caso de que se quisiera programar un Timer1 con periodo de 1 sg, que será el periodo final elegido para el Timer como se acaba de mencionar más arriba:

```
…
#include <avr/interrupt.h>
…
// Inicializar el Timer1
noInterrupts();                    // Se deshabilitan todas las interrupciones
TCCR1A = 0;
TCCR1B = 0;
TCNT1 = 15625;    // Valor del registro de comparación
TCCR1B |= (1 << CS10) | (1 << CS12);    // Establece un prescaler de valor 1024
TIMSK1 = (1 << TOIE1);    // Habilita la interrupción por overflow del Timer
interrupts();    // Habilita todas las interrupciones
…
ISR(Timer1_OVF_vect)
{
  TCNT1 = 15625;
  …
}
…
```

Y el siguiente código correspondería a emplear dicha librería:

```
…
#include <TimerOne.h>
…
Timer1.initialize(periodo);              // Inicializa el Timer1
```

```
Timer1.attachInterrupt(timerIsr);
…
void timerIsr()
{
  …
}
```

Como se puede observar, la simplificación de código es considerable, y el funcionamiento del Timer no se ve modificado respecto a las sentencias anteriores.

7.2. Diseño de SCADA

El diseño del SCADA es una de las partes vitales del trabajo, ya que permitirá al usuario interaccionar con el sistema de control del acuario, permitiendo visualizar el estado del entorno y actuar sobre los actuadores del sistema de forma práctica y sencilla mediante una serie de controles.
A continuación, se analizan las soluciones tomadas para cada uno de los apartados más importantes a la hora de llevar a cabo el diseño del SCADA en VB.

7.2.1. Creación del GRAFCET

Como sistema de supervisión y control de datos, un SCADA incluye en su diseño y posterior funcionamiento una serie de modos de marcha y paro que permiten hacer que el sistema trabaje de una u otra determinada forma. Para determinar qué modos debe incluir el sistema se han empleado como referencia las directrices determinadas en la guía GEMMA, descrita en el apartado correspondiente a Normas y referencias empleadas.
Se ha seguido la misma para el diseño del panel de botones y todo lo relacionado con la estandarización de los controles del usuario. La guía GEMMA contiene todos los estados posibles en la mayoría de instalaciones automatizadas. El diseñador deberá estudiar el sistema estado por estado para determinar cuáles son los estados necesarios en el automatismo y determinará las condiciones de

evolución o paso de un estado a otro. El método empleado para representar lo mencionado será un GRAFCET.

Los estados seleccionados son, por clasificación:

- Puesta en servicio y funcionamiento normal:
    - F1: Producción normal.
        - Ensayos y verificaciones:
            - F6: Marchas de test.
        - Parada o puesta en marcha:
            - A1: Parada en el estado inicial.
            - A4: Parada obtenida.
            - A6: Preparación para la puesta en marcha después de un defecto.
        - Defecto:
            - D1: Parada de emergencia.

El GRAFCET correspondiente al sistema diseñado se corresponde al de la siguiente figura:

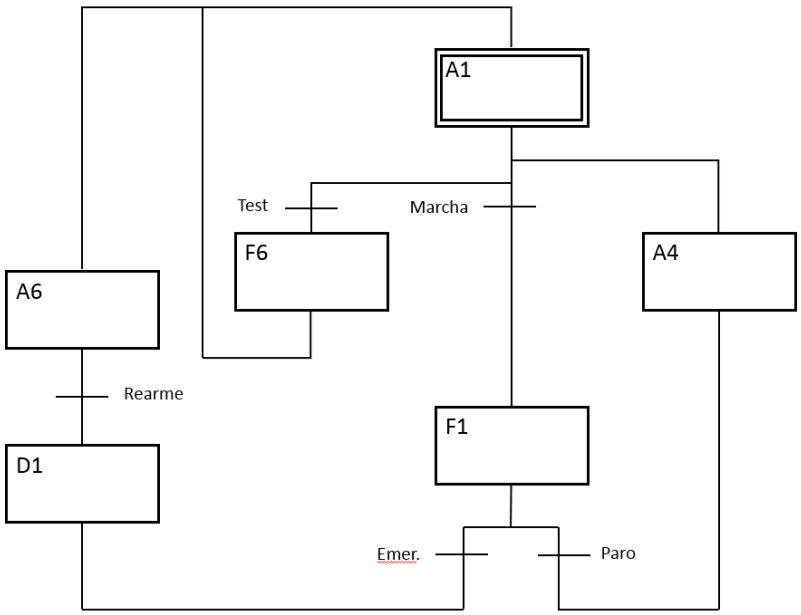


Figura 7.2.1.1 - GRAFCET del sistema diseñado

El GRAFCET, tal y como se muestra en el mismo, tendrá las siguientes operaciones:

- Marcha permite evolucionar desde el estado A1 o A4 hasta el F1.
- Test permite evolucionar desde el estado A1 o A4 hasta el F6.
- Paro hace pasar al sistema del estado F1 al A4.
- Emer. hace pasar al sistema del estado F1 al D1.
- Rearme permite pasar del estado D1 al A6.

Donde:

- Marcha: botón de marcha.
- Paro: botón de paro.
- Test: botón de verificación y testeo.
- Emer.: pulsador de emergencia.
- Rearme: botón de rearme.

Se seguirán las directrices de la guía mencionada para la asociación de colores a los botones. Los colores asociados serán los siguientes, atendiendo a la disponibilidad de gráficos y colores en VB 6.0:

- Marcha: verde.
- Paro: Negro.
- Rearme: azul.
- Emergencia: rojo (sobre fondo amarillo).

7.2.2. Lenguaje de programación empleado

Debido a la proliferación de actividades tanto de carácter industrial como educativo que incluyen la presencia de SCADAs en sistemas automatizados, existe una gran variedad de programas y lenguajes de programación que permiten diseñarlos de forma simple y estructurada. De todas las posibles alternativas, se ha considerado como solución más adecuada para la actual aplicación Visual Basic.

Las razones que han llevado a elegir dicho lenguaje como el adecuado para el diseño del SCADA son:

- Su programación es sencilla, y no necesita de un largo aprendizaje para poder realizar programas de cierta complejidad.
- Permite la creación de una pantalla de operador rica en elementos y muy personalizable gracias a la gran cantidad de controles disponibles en el

entorno de programación, tanto por defecto como a través de complementos.

- Presenta una buena compatibilidad con sistemas operativos de Microsoft actuales, Windows 8.1 inclusive, a pesar de tratarse de un entorno de programación desarrollado en su última versión para Windows 98.

En este trabajo se empleará la última versión estable desarrollada por la compañía, Visual Basic 6.0.

7.3. Conexión con Ethernet

Como se ha mencionado en apartados anteriores, el sistema contará, además de con el SCADA principal desarrollado en VB, de un SCADA simplificado para el control remoto vía web. Ello requiere que, de algún modo, Arduino pueda conectarse a Internet, ya que si no sería imposible crearlo. Para ello, a continuación se mostrarán las posibles opciones que permitan llevar a cabo esta operación, y las soluciones elegidas.

7.3.1. Módulo empleado

Para poder conectar el sistema con un SCADA desarrollado en la web, tal y como se ha mencionado en el apartado 4 del capítulo Antecedentes, es necesario disponer de un módulo o shield para Arduino, que permita a éste expandir sus funciones para poder crear un servidor web que permita controlar el sistema en línea.

Existen diversas alternativas en el mercado, muy similares entre ellas, centrándose a continuación en las más destacadas e importantes por las características que ofrecen:

- Arduino Ethernet Shield R3: versión oficial fabricada por la propia empresa Arduino, está basada en el microcontrolador W5100, con un búfer interno de 16K. Permite velocidades de conexión de hasta 100 Mbps y lleva a cabo la conexión con Arduino a través del puerto SPI.

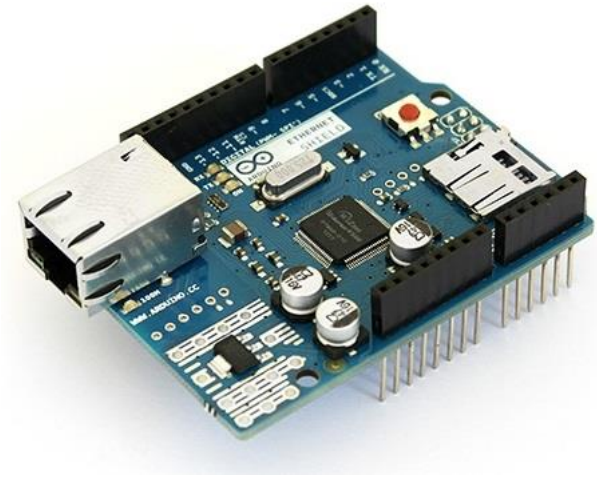


Figura 7.3.1.1 - Ethernet Shield oficial de Arduino

- Otras variantes: se ha considerado englobar el resto de Ethernet Shields existentes en el mercado en este grupo, al estar prácticamente en su totalidad basadas en la oficial mencionada anteriormente y al haber una gran cantidad de fabricantes distintos. Sus características son muy similares a las del módulo oficial, ofreciendo unas prestaciones ligeramente menores y con una calidad menos asegurada, aunque también a menor precio.

De esta forma, y con lo descrito anteriormente, se ha elegido como alternativa para este trabajo la Ethernet Shield desarrollada por la empresa IMC, variante no oficial perteneciente al segundo grupo. Las razones de la elección son:

- Precio más asequible: en comparación con el módulo oficial, la Ethernet Shield elegida tiene un precio más bajo, al igual que la mayoría de las variantes existentes en el mercado.
- Prestaciones similares: al igual que el Shield de la compañía Arduino, el módulo elegido está basado en el mismo procesador (W5100), y posee también la posibilidad de conectar tarjetas microSD.
- Compatibilidad: la variante elegida presenta compatibilidad con el modelo de tarjeta Arduino empleado, lo que lo hace adecuado para la misma.

7.3.2. Método de conexión a red

Para llevar a cabo la conexión a Internet del sistema no basta únicamente con conectar el módulo Ethernet a la TAD. Las posibilidades de conexión con la red en combinación con el script de Arduino explicado más adelante, se han tenido en cuenta diversas posibilidades:

- Conexión directa al computador: se puede realizar mediante un cable de enlace RJ-45 para conexiones Ethernet, y tiene como ventaja la sencillez de enlace físico que supone. Para permitir la conexión vía cable Ethernet, a la hora de determinar la dirección IP será necesario conocer la dirección correspondiente al adaptador de red Ethernet del computador, tal que la dirección a establecer en el script de Arduino para establecer la conexión (el proceso queda descrito en el apartado 7.7.1 de este documento) cumpla con la máscara de subred correspondiente a dicho adaptador:

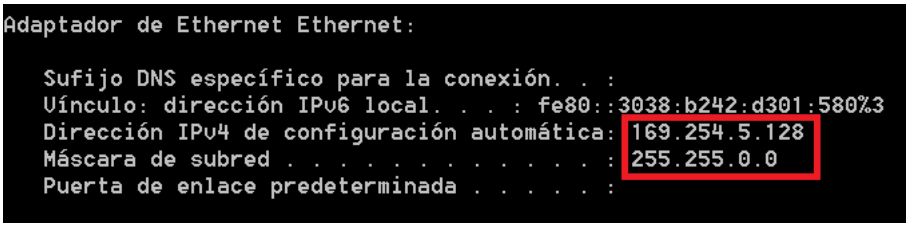


Figura 7.3.2.4 - IP y máscara de subred del adaptador Ethernet del PC

Por ello, bastará con determinar como dirección IP de la Ethernet shield una dirección dentro del rango determinado por las dos variables mencionadas. Así, por ejemplo, una dirección IP válida para el computador empleado en el trabajo sería 169.254.5.67, con la cual se ha comprobado que el programa funciona adecuadamente.

Sin embargo, existe un problema: la conexión es únicamente alámbrica por medio de Ethernet. Ello supone que Arduino no podrá conectarse ni crear el servidor web a través de una red inalámbrica, por lo que este método sólo tendría utilidad si el equipo que se va a usar para controlar a distancia el acuario está conectado también de la misma forma, algo que puede resultar engorroso ya que se necesitaría otro equipo con un enlace para conexión Ethernet, probablemente otro computador, y la comunicación

sería exclusivamente alámbrica. Ello hace que la alternativa no sea la más adecuada para establecer un control remoto.

- Conexión a router: otra forma alternativa de conectar el sistema a la red una vez instalado el módulo Ethernet es el empleo de un router. La gran ventaja que tiene este método respecto al anterior, a pesar de implicar un mayor cableado al necesitar el router una conexión al módulo Ethernet y otra a la red eléctrica para alimentación, es la posibilidad de conectar al sistema a la red WiFi disponible en el domicilio en el que esté situado el acuario, permitiendo así acceder de forma sencilla al SCADA web diseñado desde cualquier punto de la casa de forma inalámbrica y mediante multitud de dispositivos como otros ordenadores, smartphones o tablets siempre que dispongan de algún navegador web.

Teniendo en cuenta lo mencionado hasta el momento, se ha elegido como solución para este apartado del trabajo la segunda alternativa presentada. A continuación se describe el método de conexionado y configuración llevado a cabo para la adecuada conexión del router al sistema. Los pasos a seguir son:

1. Inicialmente es necesaria la adquisición de un router inalámbrico. Se propone el router TL-WR702N de la marca TP-LINK, empleado en este trabajo y elegido debido a:
    - Pequeño tamaño: las dimensiones del router (5.6x1.8x5.6 cm) lo hacen adecuado debido a que apenas ocupa espacio alguno al lado del acuario.
    - Fácil configuración: permite ser configurado de diversas formas (cliente, punto de acceso,..) y de forma muy sencilla, incluyendo el producto una pequeña guía con sencillas y concisas descripciones para ello.
    - Cableado incluido: incluye el cableado de alimentación y el cable RJ-45 necesarios para llevar a cabo la conexión con la red eléctrica y el módulo Ethernet, respectivamente.
    - Precio asequible: se trata de un router de bajo precio, debido también a la sencillez de sus prestaciones, perfectamente acordes con el sistema diseñado.

Figura 7.3.2.5 - Router TL-WR702N

2. Tras tener disponible el router, es necesario configurarlo siguiendo la guía de configuración que incluye la caja del producto. Es necesario esclarecer que el router debe ser configurado en modo cliente para que el sistema funcione.
3. Finalizada la configuración del router, será necesario determinar una dirección IP válida dentro del rango de la red inalámbrica elegida. Para el caso presente, y accediendo a la configuración del router se observa que será válida cualquier dirección entre 192.168.1.33 y 192.168.1.254 y que se encuentre disponible (fácilmente comprobable escribiendo la instrucción "C:\>ping 192.168.1.x", donde "x" es el número entre los límites anteriores a probar). En el caso presente, se ha elegido la dirección 192.168.1.67.

Una vez cumplidos los anteriores pasos, Arduino podrá conectarse a la red WiFi doméstica, permitiendo el control del sistema desde cualquier dispositivo conectado a ella.

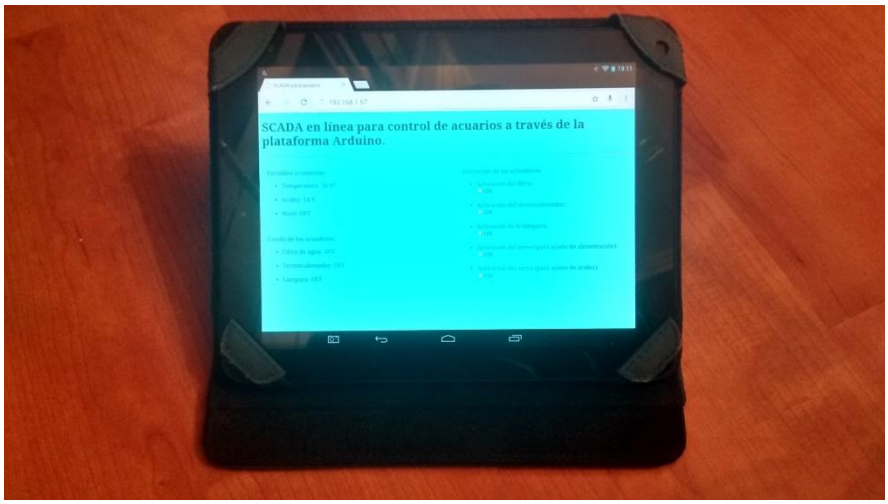


Figura 7.3.2.6 - SCADA web abierto desde navegador en una tablet

7.4. Sensores

De vital importancia en cualquier sistema automatizado es la parte de la instrumentación electrónica correspondiente a los sensores, que constituyen la mayor parte de la instrumentación electrónica de la planta y que son vitales para un correcto funcionamiento del sistema, ya que se encargan de "recoger" toda la información procedente del exterior (temperatura, acidez, presión,…) transformándola en señales eléctricas, adecuarla (si el sensor incluye esta opción) y enviarla al sistema de control correspondiente.

Por ello, a continuación se definirán todos los sensores distribuidos por la planta, explicando de entre todos los disponibles en el mercado cuál se ha escogido y las razones correspondientes.

7.4.1. Sensor de nivel

Controlar el nivel del agua en un acuario es una tarea importante ya que permite comprobar y tener la posibilidad de solucionar a tiempo determinados problemas antes de que se agraven, tales como la evaporación progresiva del agua del acuario o las pérdidas de líquido que pueda tener éste último.

Para la medición de esta variable, se considerarán dos posibilidades:

- Sensores de nivel de depósito: mediante diversas tecnologías, permiten medir con exactitud el nivel de líquido existente en un depósito, enviando

una señal proporcional al mismo tras llevar a cabo la correspondiente calibración.

- Flotadores de nivel: están compuestos de un pequeño flotador que contiene en su interior un imán, tal que cuando éste se separa lo suficiente de un segundo imán existente en el cuerpo del sensor la señal cambian. A diferencia de los anteriores, se trata de sensores todo o nada: enviarán un 1 o un 0 digital (según las características del mismo) cuando sus cabezales flote en el agua y enviarán la señal contraria cuando no floten.

Al pretender que en el sistema a diseñar el sensor determine si el acuario está por debajo de un determinado nivel o no, se ha elegido como solución la segunda alternativa. Para este trabajo, se ha escogido el flotador de nivel C-7236[13] de Cebek® con las siguientes características:

- Bajo consumo: el sensor enviará un 0 cuando esté flotando (nivel adecuado) y un 1 cuando no lo haga. Debido a que la mayor parte del tiempo se encontrará en el primer estado mencionado, supondrá un mayor consumo que si el caso fuese el contrario.
- Tensión de alimentación: el sensor se alimentará directamente a través de un pin digital de la tarjeta Arduino empleada a través de 5V, lo que supone no necesitar adecuar la tensión de alimentación del sensor ni los datos recibidos.
- Rebotes en la señal: al especificar que el pin asociado al sensor es de entrada en el código del programa de Arduino, éste activa una resistencia de pull-up interna que evita la aparición de los posibles “rebotes”.

Figura 7.4.1.1 - Imagen del sensor flotador de nivel C-7236

7.4.2. Sensor de temperatura

Un sensor de temperatura, como su propio nombre indica, permite medir la temperatura del entorno que le rodea convirtiendo una magnitud eléctrica y enviar una señal que, adecuadamente tratada, permite obtener el valor de la misma. Existen diversos tipos de sensores de temperatura en el mercado, cada uno con unas determinadas características.

De todos los disponibles, se ha elegido el DS18B20[12] por las siguientes características:

- Preparado para sumersión: el sensor es sumergible, lo que le da una importante ventaja frente a otros al poder medir la temperatura de un líquido (en este caso, agua) estando sumergido en él sin posibilidad de que haya problemas eléctricos.
- Tensión de alimentación: el sensor, al igual que el flotador de nivel anteriormente descrito, es alimentado a 5V desde la placa Arduino, lo que facilita el no necesitar aumentar o disminuir la tensión procedente de la tarjeta para alimentar el sensor.
- Alta resolución: el sensor puede ser programado para trabajar desde 9 bits hasta 12 bits de resolución máximo, lo que le proporciona una precisión elevada y personalizable por el usuario.
- Simplificación de cableado: el sensor únicamente posee tres líneas (tensión, masa y datos), los cuales se encuentran en el interior de un único cable que facilitan y simplifican el cableado del sistema.

Figura 7.4.2.1 - Sensor de temperatura DS18B20

Este sensor es capaz de medir valores de temperatura entre los -55ºC y los +125ºC. Además, presenta una resolución programable, como se acaba de mencionar arriba, entre 9 y 12 bits. A través de los cálculos reflejados en el anexo Cálculos, los valores del LSB serán:

$$LSB9bits = 0.35ºC \quad (7.4.2.1)$$

$$LSB12bits = 0.044ºC \quad (7.4.2.2)$$

Por otro lado, la sensibilidad del sensor valdrá:

$$Sensibilidad = 27{,}8\ mV/ºC \quad (7.4.2.3)$$

7.4.3. Sensor de acidez

Los sensores de acidez permiten medir el pH del líquido en el que se encuentran sumergidos, tal que permiten determinar la acidez del mismo enviando una señal eléctrica proporcional a la lectura realizada.
De entre las diversas alternativas posibles en el mercado, se ha elegido el sensor SEN-0161[10] debido a:

- Tensión de alimentación: al igual que en los casos anteriores, está preparado para trabajar a la tensión de alimentación proporcionada por Arduino, por lo que no es necesario trabajar a tensiones distintas a ésta ni adecuar el sensor a éstas.
- Acondicionamiento: el sensor incluye un pequeño circuito integrado que permite un acondicionamiento automático de la señal que procede del sensor, facilitando su empleo por parte del usuario.
- Calibración sencilla: a través de un potenciómetro situado en la placa que acompaña al sensor, éste puede ser calibrado fácilmente mediante un potenciómetro situado en la misma.

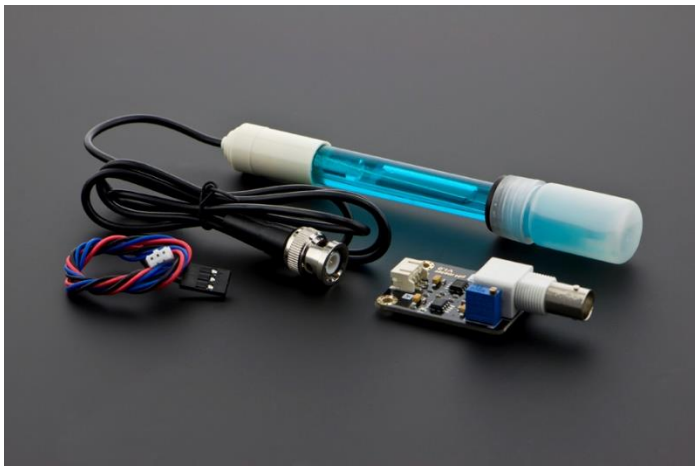


Figura 7.4.3.1 - Sensor de acidez SEN0161

Al igual que en el caso del sensor de temperatura, y tal y como queda reflejado en el anexo Cálculos, el valor del escalón unitario o LSB para el sensor de acidez es de:

$$LSB10bits = 0.014pH \quad (7.4.3.1)$$

La sensibilidad del sensor mencionado será de:

$$Sensibilidad = 0.36\,V/pH \quad (7.4.3.2)$$

7.5. Actuadores

Así como los sensores son importantes para poder obtener los valores de las variables críticas para el control del sistema, los actuadores permiten al usuario interaccionar con el sistema en función de la decisión que él decida llevar a cabo teniendo en cuenta los valores de las variables muestreadas.

A continuación se definen y describen los actuadores escogidos para cada uno de los ámbitos considerados, los cuales son:

- Ajuste de temperatura: se hace necesario controlar la temperatura del acuario, permitiendo que exista siempre una temperatura adecuada para los seres vivos que existan en el mismo.

- Ajuste de acidez: en cualquier acuario es de vital importancia controlar la acidez del medio (que en este caso es agua) para evitar que existan en él valores que puedan perjudicar a la flora y fauna del acuario.
- Alimentación de los animales: todo ser vivo necesita ser alimentado; por ello, es necesario crear algún sistema que permita alimentarlos de forma adecuada.
- Filtrado del agua: el filtrado del medio es necesario para limpiarlo de impurezas que puedan dañar a los seres vivos contenidos en el acuario.
- Iluminación del acuario: la posible existencia de plantas dentro del acuario exige la existencia de un sistema de iluminación que favorezca los procesos fotosintéticos de las mismas.

7.5.1. Conexión a relés

La tarjeta Arduino MEGA, al igual que el resto de placas de dicha marca, permite el control de determinados actuadores que no requieran de elevados valores de corriente para poder funcionar, como el servomotor empleado en este mismo sistema y descrito más adelante. Sin embargo, cualquier actuador que necesite de la energía eléctrica proporcionada por la red eléctrica general no puede ser conectado directamente a la placa, debido a que ésta no puede proporcionar el voltaje y la corriente que necesite el mismo para funcionar.

Una solución a dicho problema, la cual ha sido adoptada en este trabajo, es el empleo de relés. Por un lado, éstos pueden ser conectados a la TAD empleada para ser cerrados o abiertos desde la misma, y a la vez tienen una corriente máxima lo suficientemente elevada para permitir conectar los actuadores a ellos, abriendo y cerrando el circuito cuando sea necesario.

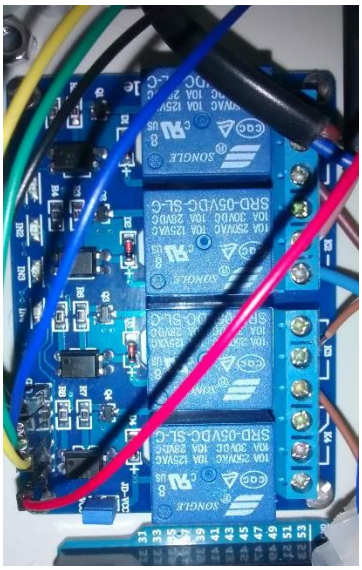


Figura 7.5.1.2 - Placa de relés para actuadores

Tal y como se muestra en la figura superior a este texto, la solución elegida ha sido una PCB preparada con 4 relés. Se ha elegido esta alternativa como la más adecuada debido a:

- Facilidad de uso: los relés están preparados para trabajar a la tensión de alimentación que puede proporcionar Arduino, y las conexiones a realizar entre la placa y la TAD son muy sencillas y requieren poco cableado.
- Número de relés: debido a que en el sistema a diseñar se necesitarán tres relés para el control de actuadores, la placa elegida es más que adecuada al poseer incluso uno más de los necesarios, el cual podría funcionar como sustituto en caso de que alguno de los empleados en el funcionamiento normal fallase.
- Circuito ya diseñado: no es necesario incluir ningún componente para permitir el funcionamiento de los relés, al incluir la PCB que los contiene todos los circuitos y componentes necesarios para su adecuada puesta en funcionamiento.

7.5.2. Termocalentador

El termocalentador es una parte fundamental para poder controlar la temperatura a la que se encuentra el medio líquido. Las distintas alternativas presentes en el

mercado son muy similares entre ellas, diferenciándose casi exclusivamente en una característica:

- Termocalentadores analógicos: disponen de un pequeño mando circular que permite elegir la temperatura de calentamiento del entorno. Pueden basarse en diversas tecnologías para llevar a cabo su propósito, como es el caso de los que se valen de fenómenos magnéticos para modificar la temperatura del sistema.
- Termocalentadores digitales: se diferencian de los anteriores en que presentan una pequeña pantalla LED que permite modificar de forma táctil la temperatura en unos rangos similares o idénticos a los incluidos en la categoría anterior.

Inicialmente, se consideró la posibilidad de incluir en el sistema un termocalentador digital, debido a la posible manipulación de éste de forma interna para permitir modificar su temperatura sin que fuera necesario que el usuario accediese de forma manual a él. Sin embargo, la diferencia considerable de coste entre unos y otros ha decantado la solución a favor de los analógicos, más sencillos pero de prestaciones igualmente fiables, a pesar de que no permitan ser controlados en su totalidad de forma automática por el sistema.

Figura 7.5.2.1 - Termocalentador XiLONG

De entre todas las alternativas pertenecientes a ese grupo, se ha escogido el XiLONG XL-025A por razones como:

- Bajo consumo: al ser instalado en un acuario de relativamente poca capacidad como el empleado (20 litros), no es necesario un termocalentador de gran potencia. El escogido posee una potencia eléctrica de 25W, más que adecuada para el tipo de acuarios definido según la descripción del producto ofrecida por el fabricante y que no provoca un elevado consumo de energía.
- Bajo coste: precisamente por la razón anterior, al ser de una potencia más baja que otros de su rango su precio también es menor, lo que siempre es una ventaja.
- Bajo consumo: comparado con otros termocalentadores del mercado preparados para volúmenes de líquido más elevados, la potencia consumida por el calentador es considerablemente baja, lo que permite ahorrar en consumo eléctrico.
- Rango amplio: el rango de temperaturas que ofrece es amplio (entre 18 y 35ºC), más que suficiente para la aplicación que se quiere desarrollar y con capacidad para adaptarse a animales de diversos ecosistemas, con distintas temperaturas del medio.

7.5.3. Lámpara

La iluminación es otra parte esencial en el control de la vida de un acuario, ya que en caso de que en el mismo existan vegetales, éstos necesitan de luz para poder realizar la fotosíntesis.

Dentro de las lámparas preparadas para acuarios en el mercado, se ha escogido el modelo de lámpara GL-18T los siguientes motivos:

- Bajo consumo: con una potencia de 9W, este modelo está capacitado para dotar de suficiente iluminación al acuario sin necesidad de incurrir en un consumo energético considerable.
- Sumersión: al estar preparada específicamente para su uso en acuarios, esta lámpara puede introducirse en el agua, sin temor a que se vea afectada por la presencia de la misma.

Figura 7.5.3.1 - Lámpara GL-18T colocada en acuario

7.5.4. Filtro

Dentro de los actuadores elegidos para ser incluidos en el trabajo, se ha considerado que el filtro sea el único de ellos que esté permanente activado en el control automático (salvo situación de emergencia), y el que más tiempo se recomienda que esté activado en el modo de control manual. La razón de esto es que el filtro, desde el punto de vista del entorno y no del control, es un elemento fundamental en cualquier acuario para garantizar la calidad del agua; por ello, se tendrá en cuenta esta consideración a la hora de llevar a cabo la programación del funcionamiento del mismo.

Figura 7.5.4.1 - Filtro de agua JET-FLO 50

Dentro del mercado, se ha decidido escoger un filtro adecuado a la capacidad en litros del acuario. Uno para menor capacidad podría no ser suficiente y uno de mayor, estar sobredimensionado y ser considerablemente superior en cuanto a su coste de adquisición. Por ello, se ha escogido el Elite JET-FLO 50, acorde a las características definidas.

7.5.5. Servomotor

Un servomotor es un dispositivo pequeño, el cual posee un eje de rendimiento controlado. Este puede ser llevado a posiciones angulares específicas al enviar una señal codificada. Con tal de que una señal codificada exista en la línea de entrada, el servo mantendrá la posición angular del engranaje. Cuando la señala codificada cambia, la posición angular de los piñones cambia. En la práctica, se usan servos para posicionar superficies de control como el movimiento de palancas, pequeños ascensores y timones.

Para el trabajo desarrollado en cuestión, se han barajado diversas posibilidades a tener en cuenta, entre las cuales destacan:

- Ángulo de giro del servomotor: existen servos de distintos tipos, algunos capaces de girar entre los 0 y los 360 grados, y otros con la capacidad de girar únicamente media vuelta. Para el caso dispuesto, un servo perteneciente al segundo grupo es suficientemente válido.
- Número de servomotores empleados: es necesario, por un lado, permitir la alimentación de los animales del acuario, y por el otro poder realizar el ajuste de la acidez del medio.

  Si se emplean compuestos que aumenten o disminuyan siempre el pH, como los ácidos y las bases respectivamente, sería necesario disponer de un servomotor adicional dedicado exclusivamente al ajuste. Sin embargo, la existencia de compuestos denominados tampón o buffer permiten mantener siempre un nivel estable de pH en el acuario, por lo que no será necesario disponer de compuestos adicionales más allá del mencionado. De esta manera, y tal y como se muestra en la imagen siguiente, se puede diseñar un pequeño contenedor que divida mitad de su contenido para el alimento y mitad para el buffer.

Figura 7.5.5.1 - Servomotor del acuario

Para el trabajo desarrollado, se ha escogido el Micro Servo de 180º, por las siguientes razones:

- Tensión de alimentación: al igual que en los casos anteriores, está preparado específicamente para trabajar con las alimentaciones proporcionadas por la placa Arduino, siendo idóneo en este ámbito.
- Facilidad de programación: la placa permite utilizar una librería específica para la programación del servo, lo que facilita considerablemente el código correspondiente al mismo.
- Ángulo de rotación: el Micro Servo permite un movimiento de giro de hasta 180º, suficiente para la función a realizar en este trabajo. Un servo de mayor libertad de giro hubiese supuesto mayor coste, algo que no es necesario.

7.6. Protección contra corrientes

Es necesario tener en cuenta que en el sistema diseñado el medio acuático está presente durante el funcionamiento del sistema, estando en contacto con sensores y actuadores y por ello es de vital importancia que el mismo éste protegido contra posibles “fugas” de corriente que, al entrar además en contacto con un medio tan conductor de la electricidad como el agua, puede tener consecuencias catastróficas para los componentes del sistema así como para los organismos presentes en el acuario.

Como solución a este problema, se ha decidido elegir una sencilla sonda de titanio, la cual permite, al conectarla a la tierra de la red eléctrica, hacer que cualquier mínima corriente que pueda derivarse al agua se vaya a la tierra de la red eléctrica evitando así los consecuentes daños que podría provocar si permaneciese en el medio.

Figura 7.6.1 - Pica de titanio para protección contra corrientes

Cabe comentar que, a diferencia de actuadores y sensores, dicho sistema de protección estará siempre encendido y conectado a la red eléctrica, independientemente de que el sistema de control esté o no esté activado. Se ha tomado esta decisión porque se considera que, al ser un sistema de protección, es crucial que esté activado en todo momento.

7.7. Programas

En una aplicación como la desarrollada en este trabajo, una correcta y completa programación es imprescindible para un adecuado funcionamiento del sistema así como para su estabilidad a lo largo del tiempo de uso. Además, la comunicación entre las tres aplicaciones desarrolladas (script de Arduino, SCADA principal en VB y control remoto web en HTML) debe ser consistente y estar adecuadamente sincronizada.

A continuación, se desarrollan los códigos diseñados tanto para la tarjeta Arduino como para el software SCADA, diseñado a partir del lenguaje de programación Visual Basic 6.0, y la web para control remoto del sistema en HTML.

7.7.1. Programa de Arduino

La plataforma Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel Processing, ofreciendo también la posibilidad de utilizar otros lenguajes de programación y aplicaciones populares en Arduino. En este caso, se ha optado por emplear directamente el lenguaje propio de la placa, el cual no es excesivamente complejo y dispone de todos los mecanismos para ejecutar todas las operaciones que se requieren.
A continuación se muestra un flujograma del funcionamiento del mismo:

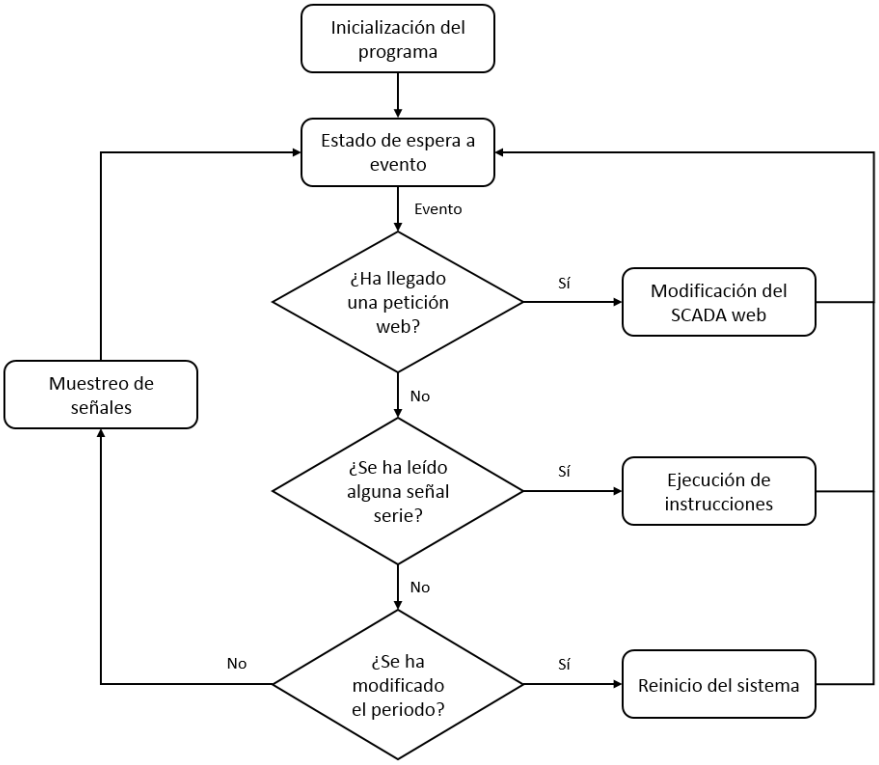


Figura 7.7.1.1 - Flujograma de funcionamiento del script de Arduino

Para facilitar el entendimiento del código así como de las diversas variantes elegidas a la hora de crearlo, se dividirá el mismo en diversas partes en orden ascendente de número de líneas de código. A lo largo de dicha explicación, cuando sea necesario definir el propósito de alguna función se escribirá entre paréntesis el nombre que se le ha dado en el programa. Además, los nombres de funciones, variables y librerías empleadas se escribirán en cursiva.

Dichas partes se definen a continuación:

- Definición de librerías y constantes: antes de comenzar a declarar las variables que se emplearán a lo largo del programa diseñado, se hace necesario declarar las librerías que permitirán usar algunas de las funciones declaradas más adelante. Las librerías son declaradas de la siguiente forma:

  ```
  #include <librería_1.h>
  #include <librería_2.h>
  …
  #define constante_1 (valor numérico)
  #define constante_2 (valor numérico)
  ```

  En el script diseñado, las librerías *OneWire* y *DallasTemperature*, así como la constante *Pin_temp* se emplearán para el control del sensor de temperatura empleado, el DS18B20. Dichas librerías permitirán, respectivamente, conectar el sensor con Arduino y emplear las funciones diseñadas por el fabricante para la obtención de la temperatura. Por otro lado, la librería Servo será empleada para permitir el control del servomotor; y finalmente, la librería *TimerOne* ha sido declarada con el objetivo de simplificar (como se menciona en el apartado 7.1.3 de este documento) el empleo del Timer1 del procesador ATMega2560. La constante *Pin_acid* se ha declarado simplemente para facilitar el entendimiento a la hora de programar; dicha constante va asociada al pin analógico A0 al que va conectado el sensor de acidez. De la misma forma, la constante *Pin_nivel* va asociada al pin en el que se encuentra conectado el flotador de nivel; dicho pin será el pin digital 42. El resto de librerías añadidas serán necesarias para permitir diversas tareas como la conexión a Internet o el acceso al lector de tarjetas de la Ethernet Shield.
- Configuración de la Ethernet Shield: antes de declarar formalmente el resto de variables del sistema, se hace imprescindible establecer los valores de las variables características que permitirán al módulo de Ethernet, apoyándose en la Arduino MEGA, conectarse a Internet.

  Para ello, será suficiente declarar tres propiedades:

- Dirección MAC: hace referencia a la dirección identificativa de la placa Ethernet. Se declarará como tipo byte, y es proporcionada por el fabricante:

  byte mac[] = { 0xDE, 0x3C, 0x05, 0x00, 0x00, 0xF7 };

- Dirección IP: ya definida anteriormente, debe encontrarse dentro del rango de direcciones proporcionado por la red. Será del tipo *IPAddress*, tipo especial de variable característico de la librería *Ethernet.h*:

  IPAddress ip(192, 168, 1, 67);

- Número de puerto: determinará que puerto se usará para establecer el servidor web. Por defecto, se ha empleado el puerto dedicado a este tipo de operaciones en sistemas operativos Windows, el número 80. La variable será del tipo *EthernetServer*, propia de la librería *Ethernet* al igual que la *IPAddress*:

  EthernetServer server(80);

Estas tres propiedades serán usadas más adelante en el programa para permitir la conexión a la red Ethernet del sistema.

- Declaración de variables: es una parte muy importante del código, ya que en ella se declararán las variables que se usarán posteriormente en el programa. A continuación se muestra cómo declarar algunos de los tipos más comunes:

```
String cadena_1;
int vector_1[número_elementos];
byte dato_1;
…
```

Como se observará más adelante, y como está comentado de forma concisa en el archivo del código, cada variable corresponde a un determinado sensor/actuador o bien es empleada de forma auxiliar en alguna de las líneas correspondientes del programa. Se ha procurado asociar a cada variable un nombre claro y sencillo, de tal manera que se pueda comprender su función dentro del programa a través de su denominación.

- Inicialización del programa (función *setup*): en dicha función se establecerán las características básicas del programa, a mencionar:
    - Velocidad de transmisión.
    - Carácter de entrada/salida de los pines (pinMode).
    - Escritura de valores iniciales (HIGH/LOW) en dichos pines.
    - Activación del Timer empleado para el muestreo.

A continuación se muestra un ejemplo del código asociado a dicha función simplificado:

```
void setup()
{
  sensors.begin();
  pinMode(nombre_pin, OUTPUT);
  myservo.attach(pin_servo);

  digitalWrite(nombre_pin, HIGH);
  Serial.begin(velocidad_transmisión);

  Timer1.initialize(periodo);
  Timer1.attachInterrupt(función_interrupción);
}
```

Como se observará más adelante, se configurarán los pines asociados a los relés como OUTPUT (salida) y el asociado al flotador de nivel como INPUT (entrada). Se escribirá una señal HIGH en los relés para que estén inicialmente abiertos (los niveles lógicos están invertidos), y se inicializará

la comunicación a través del puerto serie a la baud rate definida (apartado 7.1.2).

También será necesario, para comprobar que la detección de la tarjeta microSD es correcta, enviar al SCADA en VB un mensaje de error si éste existiese, para informar al usuario del problema. También se comprobará si existe el archivo .htm dentro de la susodicha tarjeta:

```
error_1:
if (!SD.begin(4)) {
  Serial.println(errorsd);
  goto error_1;
}
 error_2:
 if (!SD.exists("html_web.htm")) {
  Serial.println(errorhtm);
  goto error_2;
}
```

Para finalizar, y si no se produce ninguno de los errores mencionados, se inicializará el Timer empleado.

- Función de interrupción del temporizador (función *timerIsr*): es la función que se activa cada vez que se desborda el Timer al alcanzar el periodo definido. Dentro de ella se ha decidido llevar a cabo el muestreo de las señales procedentes de los sensores, con el periodo de muestreo determinado más arriba en esta memoria. Dicha función lleva asociado el siguiente código:

```
void función_interrupción()
{
  …
}
```

En el código diseñado se reinicializa la cadena en cada ciclo para no enviar datos incorrectos, y posteriormente se llama a cada una de las funciones asociadas a los sensores: acidez, temperatura y nivel respectivamente. Tras esto, se imprime el valor de la cadena, ya completa, a través del puerto serie. Las funciones se muestran en el anexo correspondiente a la programación.

- Función de repetición continua (función *loop*): en dicha función se llamará a la función *lectura_serie*, la cual se encargará de leer el contenido del puerto serie cada vez que se reciba un dato a través de éste; y por otro lado se llamará a la función de generación y actualización del control web remoto. Dichas funciones se ejecutan en esta parte del código debido a que, especialmente en caso de situación de emergencia, es vital que los actuadores respondan al momento a las señales enviadas desde el SCADA, algo posible en la función *loop* ya que se ejecuta una y otra vez sin paro a lo largo del tiempo de ejecución del programa.

  A continuación, se describen las dos funciones mencionadas:

  - Lectura del puerto serie (función *lectura_serie*): se encarga de interpretar la orden procedente del SCADA principal, ejecutando las instrucciones correspondientes en función del valor de dicha variable:

```
void lectura_serie(){
  if (Serial.available()){
  dato_serie = Serial.read();
  switch(dato_serie){
    case 0:
    …
    break;
    case 1:
    …
    break;
    …
    default:
    …
```

```
    break;
  }
 }
}
```

- Para facilitar la programación, se ha empleado la instrucción *switch* pasando como argumento la variable *orden*, en la que se almacena el dato que llega por el puerto serie, de carácter numérico. Cada código tiene asociada una función en el script empleado, que se recogen de manera resumida en la siguiente tabla:

Instrucciones switch	
Valor señal	Significado
0	Termocalentador ON
1	Filtro ON
2	Lámpara ON
3	Termocalentador OFF
4	Filtro OFF
5	Lámpara OFF
6	Servo para alimentación
7	Servo para ajuste acidez
8	Testeo
9	Configuración
default	Apagado de actuadores

Tabla 7.7.1.1 – Traducción para las instrucciones de *lectura_serie*

Cabe mencionar que a la hora de escribir las instrucciones de dicha función, por la configuración dada a los pines los estados están invertidos; es decir, HIGH corresponde a un nivel bajo mientras que LOW a un nivel alto.

- Comunicación con el SCADA web (función *SCADA_web*): en esta función, la placa Arduino se encarga de comprobar si se ha recibido alguna petición desde el servidor web creado previamente por parte del cliente, actuando en consonancia con lo recibido, y que será almacenado en una cadena de caracteres (variable *HTTP_req* en el

programa). Así, Arduino actualizará los valores de las variables muestreadas por el sistema o el estado de los actuadores en función de la petición del usuario del SCADA remoto.

A continuación se puede visualizar un ejemplo del código que conforma dicha función:

```
void SCADA_web()
{
   EthernetClient client = server.available();
   if (client) {
      boolean currentLineIsBlank = true;
      while (client.connected()) {
         if (client.available()) {
            char_in = client.read();
            HTTP_req += char_in;
            if (char_in == '\n' && currentLineIsBlank) {
               …
               …
               webFile = SD.open("html_web.htm", FILE_WRITE);
               if (webFile) {
                 …
                 …
                 webFile.close();
               }
               webFile = SD.open("html_web.htm");
               if (webFile) {
                 while(webFile.available()) {
                   client.write(webFile.read());
                 }
                 webFile.close();
               }
               HTTP_req = "";
               break;
```

```
                }
                if (char_in == '\n') {
                    currentLineIsBlank = true;
                }
                else if (char_in != '\r') {
                    currentLineIsBlank = false;
                }
            }
        }
        client.stop();
    }
}
```

Como se puede observar, la función comprueba si llega alguna petición desde el servidor; si es así, la tarjeta lleva a cabo dos operaciones principales:

- Abre el archivo .htm en modo escritura para modificar lo necesario en el mismo en función de la petición realizada por el usuario.
- Abre el mismo archivo en modo lectura para mostrar la página actualizada con los cambios realizados.

A su vez, dicha función contiene una serie de funciones con los objetivos hasta ahora comentados repartidos entre ellas. Dichas funciones son:

- *muestreo*: se encarga, como su propio nombre indica, de muestrear las magnitudes a controlar para mostrar sus valores en la web.
- *evaluar_req*: comprueba el estado de los actuadores, indicando al programa si éstos deben encenderse o apagarse en función de la petición recibida.
- *esc_variables*: imprime en el servidor los valores de las variables muestreadas en la función muestreo.

- *estado_actuadores*: envía al servidor el estado de los actuadores para mostrarlos en la página web correspondiente al SCADA.
- *checkbox_rele*: con los valores obtenidos del estado de los actuadores en la función *evaluar_req*, esta función activa o desactiva cada actuador si así lo exige la petición recibida desde el servidor.
- *evaluar_servo*: chequea el estado del servomotor, para indicar si es necesario llevar a cabo bien un proceso de alimentación para los animales o bien un ajuste de pH del agua.
- *servo_alim*/*servo_acid*: se encargan, con lo resultado de la función anterior, de activar el servo para uno u otro proceso.
- *habilitar_checkbox*: en el modo automático, los actuadores son controlados por el computador de forma automática. Por lo tanto, no tendría mucho sentido que en este modo el usuario pudiese controlar desde el SCADA web, que no es más que una extensión del SCADA principal, cada uno de ellos a través de las checkbox correspondientes.

  Para evitarlo, cada vez que se inicia el SCADA en VB, el mismo SCADA envía a Arduino a través de una variable entera denominada *modo_arduino*, el modo de funcionamiento seleccionado, para que habilite o deshabilite la opción de activar/desactivar los actuadores en función de si el modo de funcionamiento es manual o automático, respectivamente. Dicho valor es recogido por la variable modo, empleada únicamente para este propósito.

- Testeo del sistema (función *testeo*): esta función se encarga, como su propio nombre indica, de testear o comprobar que los pines de la TAD asociados a los relés funcionan adecuadamente para permitir una correcta apertura o cierre de los mismos, a la vez que testea la comunicación serie entre la TAD y el SCADA diseñado en VB.

-

Esta función, llamada desde *lectura_serie*, se encarga de comprobar, por orden:

- El envío de datos por el puerto serie, mediante la función *Serial.print*.
- El encendido y apagado de todos los relés empleados en el sistema.
- El correcto funcionamiento del servomotor empleado para las tareas de alimentación y ajuste de la acidez.

Una vez finalizado este proceso, los resultados son enviados a través del puerto serie al SCADA principal, que se encargará de tratarlos y evaluarlos. Esta función incluye una serie de retardos, con el único motivo de sincronizarse de forma adecuada con la evolución del proceso de testeo en el SCADA diseñado.

- Configuración inicial del Timer y del modo de funcionamiento (función configurar): al igual que la función anterior, configurar es llamada desde la función *lectura_serie*, y se encarga de hacer que el temporizador empleado (Timer1) funcione al periodo de muestreo que el usuario haya elegido en el SCADA de VB. Un ejemplo de la función se muestra a continuación:

```
void configurar(){
  Timer1.stop();
  while(puerto_serie.available() <= 0){}
  t_serie = puerto_serie.read();
  while(puerto_serie.available() <= 0){}
  modo = puerto_serie.read();
  t_elegido = t_serie * 60;
  … (muestreo de señales)
  Timer1.initialize(t_defecto);
  Serial.println(prueba_comunicacion);
  Serial.println(cad_datos);
}
```

El proceso consistirá en la espera de la función en un bucle hasta que el dato numérico que indica el periodo (valor entre 1 y 60) llegue por el puerto serie; y, tras éste, el modo elegido en el SCADA principal. Finalizadas dichas lecturas, se actualiza el valor del periodo del Timer a 1 sg, y se llevará a cabo un muestreo inicial (correspondiente a un instante t=0 inicial) antes de que el temporizador se inicialice. El motivo de este muestreo es el de proporcionar al usuario valores iniciales de las variables para conocer el estado inmediato del sistema en su puesta en marcha, y que no tenga que esperar excesivo tiempo para recibir datos si el periodo de muestreo que ha elegido es excesivamente grande.

La función que desempeña la variable *t_elegido* del ejemplo es la de compararse con la variable *cont*, variable auxiliar que aumenta de valor una unidad cada vez que se desborda el Timer; es decir, cada vez que pasa 1 sg. Cuando ambos valores se igualan, se muestrea el sistema y se envían vía serie las variables del entorno al SCADA de VB.

- Señal de acidez (función *leer_acid*): dicha función se encarga de recibir la señal procedente del sensor de acidez y adecuarla para que pueda ser enviada e inmediatamente mostrada por pantalla en el SCADA. Para ello, el código consta de tres partes:
    - Muestreo de la señal a través de la función *analogRead*.

      señal = analogRead(pin_sensor);

    - Adecuación de la señal recibida para transformarla en un valor de acidez válido.

      acidez = (float)señal*5.0/1024;
      acidez = 3.5*acidez + Offset;

    - Adición del valor a la cadena a enviar, enviando una construcción distinta si el valor es menor, o mayor o igual que 10.

      dtostrf(acidez, 3, 1, pH);

```
if(acidez < 10)
{
  envío = '0' + pH;
}
else
{
  envío = ' ' + pH;
}
```

- Señal de temperatura (función *leer_temp*): con un objetivo muy similar a la función inmediatamente descrita arriba, la función *leer_temp* se encarga de recibir la señal procedente del sensor de temperatura, para posteriormente adecuarla para su envío. El código empleado es el siguiente:

```
void leer_temp()
{
 sensors.requestTemperatures();
 Temp_en_grados = sensors.getTempCByIndex(pin_sensor) + Offset;
}
```

  Como se puede deducir del código, éste es de una longitud considerablemente menor que el de la función anterior. Sin embargo, su tiempo de ejecución (apartado 7.1.3) es muy similar. Esto se debe a que las dos funciones empleadas (*requestTemperatures* y *getTempCByIndex*), procedentes de la librería *DallasTemperature.h*, tienen un amplio número de instrucciones contenidas en su interior y la respuesta del sensor es bastante lenta.
- Señal de nivel (función *leer_nivel*): la función nivel permite recibir la señal del flotador de nivel, indicando si el nivel de agua del acuario está por debajo del nivel adecuado o si está en el mismo. Dicha función es muy simple, debido a que también lo es el funcionamiento del sensor al enviar una señal binaria que indica si está abierto o cerrado:

```
void leer_nivel()
```

```
{
  nivell = digitalRead(pin_sensor);
}
```

7.7.2. Programa del SCADA (Visual Basic)

Una vez diseñado el código correspondiente a la placa Arduino MEGA, es necesario diseñar el SCADA en Visual Basic que se empleará para el control del sistema por parte del usuario, y que permitirá de forma gráfica e interactiva controlar el acuario mediante un conjunto de botones y controles, a la vez que recibe toda la información del mismo y la visualiza en pantalla en tiempo real.

A la hora de diseñar el SCADA, y teniendo en cuenta la guía GENMA mencionada en el apartado 4.1 de este documento, se ha dividido el SCADA en varios formularios, tal que permitan facilitar el entendimiento y la programación del mismo, confiriéndole cierto carácter "modular". La relación entre dichos formularios queda definida en la siguiente figura:

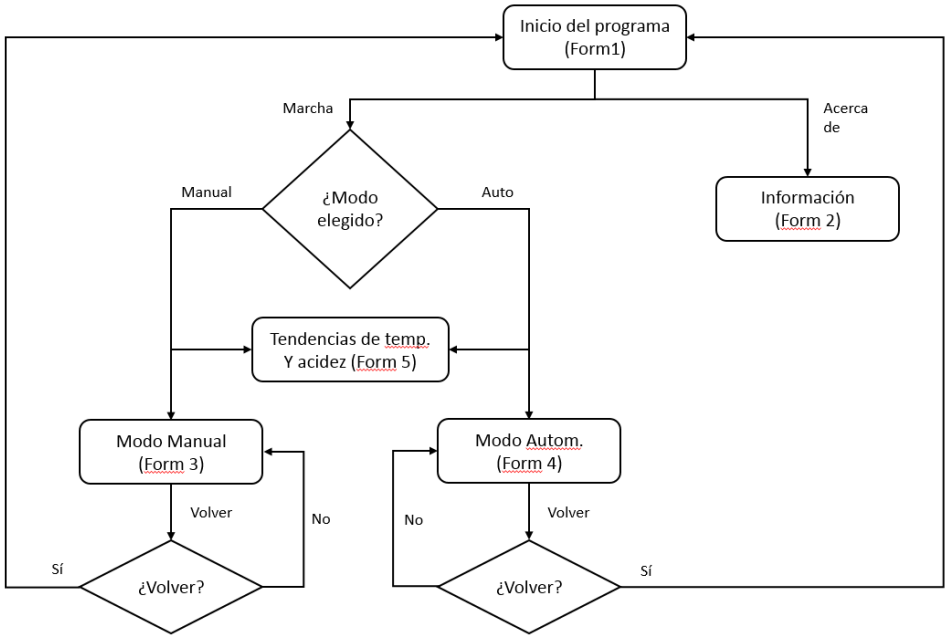


Figura 7.7.2.1 - Diagrama de flujo del SCADA en Visual Basic

Al igual que en subapartado anterior, los nombres dados en el programa para cada uno de los formularios se mostrarán entre paréntesis al lado del número del

formulario descrito, y las variables empleadas, así como los nombres de formularios y funciones, serán escritos en cursiva. A continuación, se definen las funciones los mismos

- Form1 (*Inicio*): es la pantalla de inicio del SCADA, en la cual el usuario del sistema debe seleccionar el modo de control deseado, y desde donde se accederá a la pantalla de control del sistema para comenzar el mismo, sea manual o automático. Además, y de forma común a todos los formularios (a excepción del Form2), se incluye una barra de menú para facilitar el acceso a determinados comandos. En la siguiente figura se muestra dicho formulario:

Figura 7.7.2.2 - Pantalla de inicio del SCADA correspondiente al Form1

La programación asociada a dicho formulario es muy simple, ya que consta únicamente de los siguientes controles:

- Un CommandButton.
- Una Image.
- Un Frame con dos OptionButton.
- Una Label.

A continuación se describe el objetivo de cada una de las funciones privadas del formulario empleadas, así como de la inicial y común a todas ellas *Option Explicit*:

- *Option Explicit*: permite declarar de forma pública o privada (en este caso, la primera opción) para las distintas funciones contenidas en el formulario las variables empleadas a lo largo del mismo.
- *Form_Load*: describe la inicialización del formulario, estableciendo valores iniciales de variables en caso de ser necesario así como otras operaciones (como, en este caso, la determinación de la imagen contenida en el control Picture).
- *Comenzar_Click*: es un CommandButton cuya función es, tras haber escogido el modo de funcionamiento del sistema, iniciar el control del acuario pasando al formulario correspondiente.
- *Info_Click*: menú que permite abrir el formulario Form2, cuya función se determina más adelante.
- *Auto_Click*: activa la variable correspondiente que indique que el tipo de control será de carácter automático.
- *Manual_Click*: igual que el control anterior, pero para el caso manual.
- *Salir_Click*: permite salir del SCADA, clicando en el control correspondiente del menú Archivo.

El último de los controles, *Form_Unload*, no se comentará debido a que su contenido es exactamente igual que el del subformulario *Salir_Click*. La razón de incluirlo es dar respuesta a la posibilidad de que el usuario quiera finalizar el SCADA activando el control propio de Windows para cerrar ventanas y no el comando existente en la barra de menú.

- Form2 (*Info*): este formulario es mostrado en pantalla cuando el usuario activa el control del menú de nombre “Acerca de”, con el objetivo de mostrar una breve descripción de en qué consiste el trabajo realizado, como se muestra en la siguiente figura:

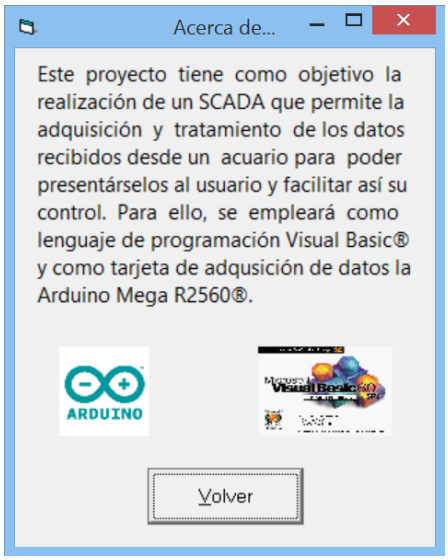


Figura 7.7.2.3 - Formulario de información del SCADA

El código asociado a dicho formulario es muy simple, ya que únicamente es programable el CommandButton que permite volver al formulario desde el cual se ha activado el control del menú correspondiente. Todo el texto mostrado en la anterior figura está escrito en la propiedad Caption de la Label o etiqueta existente en el formulario.

- Form3 (*Manual*): es el formulario al que se accede seleccionando la opción de control manual, y uno de los dos formularios principales del SCADA, ya que desde él se controlará directamente el funcionamiento del acuario. La imagen mostrada a continuación muestra el aspecto del formulario:

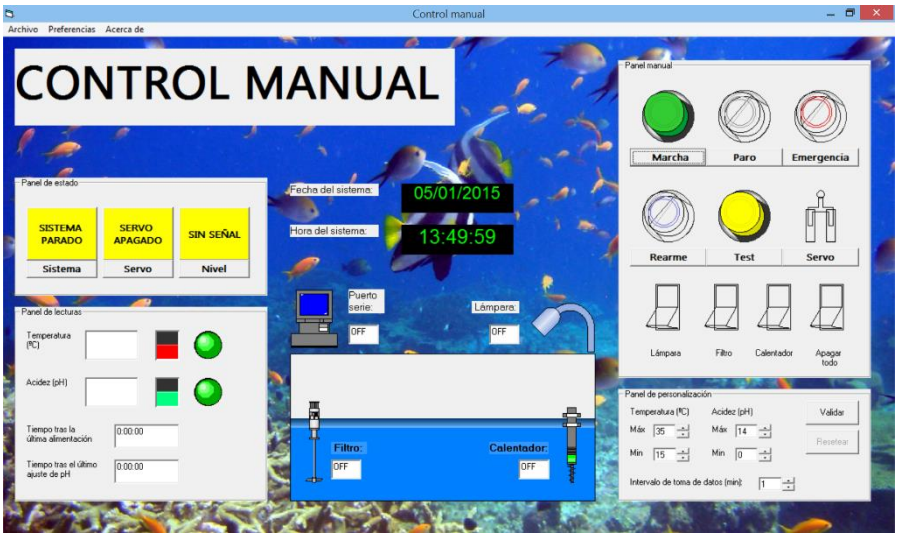


Figura 7.7.2.4 - Pantalla de control manual del SCADA

Como resumen de todo lo desarrollado, se dividirá el código del formulario en secciones. Todo el código desarrollado se encuentra recogido para su consulta en el Anexo III:

- Declaración de variables, situada en el *Option Explicit*. En ella, como su propio nombre indica, se definirán variables para diversos cometidos: almacenamiento de datos, flags, envío y recepción de datos por puerto serie, etc.
- Inicialización del programa, en *Form_Load*. Se inicializarán las variables a sus correspondientes valores iniciales, se definirá las propiedades iniciales del formulario así como las de los objetos. Entre todo esto, se pueden poner como ejemplos destacados:
  - Puesta a cero de variables iniciales.
  - Inicialización de los temporizadores.
  - Determinación del apartado gráfico inicial (tamaño, colores,…) tanto del formulario como de los objetos situados en él.

  En dicha inicialización del programa, tanto en este formulario como en el siguiente (control automático), es vital establecer correctamente los parámetros de la comunicación por el puerto serie, ya que de ella dependerá el correcto envío y recepción de datos. Para esto, se emplea el control MSComm accesible a través del componente o librería Microsoft Comm Control. Las propiedades a determinar del mismo se muestran a continuación:
  - *CommPort*: indica el número de puerto serie usado. Admite valores desde 1 hasta 255.
  - *InBufferSize*: permite establecer el tamaño del buffer de entrada.
  - *RThreshold*: determina el número de caracteres necesarios en el buffer de entrada para que se produzca el evento OnComm mencionado más adelante.
  - *InputLen*: determina el número de caracteres a leer. Si se coloca un valor de 0 o igual a *InBufferSize*, se lee el buffer de entrada al completo.

- *Settings*: características básicas del puerto serie. Su sintaxis determina, por orden de izquierda a derecha: velocidad de transmisión (en baudios), paridad, número de bits por trama y bits de stop a emplear en la comunicación.
- *RTSEnable*: debe ponerse a 1 para indicar al equipo que va a recibir la comunicación que deseamos enviar datos.
- *DTREnable*: se pone a 1 para indicar al módem que el terminal (ordenador) está preparado para recibir datos.

Un ejemplo de código en el que se configura dicho componente es el siguiente:

```
MSComm1.RThreshold = 10
MSComm1.InputLen = 10
MSComm1.InBufferSize = 10
MSComm1.Settings = "9600,n,8,1"
MSComm1.CommPort = 3
MSComm1.DTREnable = True
MSComm1.RTSEnable = True
```

En el programa realizado, se puede conocer de forma sencilla el número de caracteres que serán necesarios para la recepción de los datos desde Arduino. Teniendo en cuenta lo establecido en el apartado 7.1.2 de este documento, el valor de las propiedades *RThreshold*, *InputLen* e *InBufferSize* es de 12.

- Recepción de datos, producida en el subformulario *MSComm1_OnComm*. El objetivo de esta parte del programa es recibir los datos procedentes del puerto serie, adecuarlos para su uso y visualización y decodificar el mensaje, para saber si se trata de un mensaje de error, un mensaje que contiene las variables muestreadas u otros casos. Además, en esta función se realizará la representación de los valores recibidos del muestreo del entorno en las gráficas de tendencia y se controlarán los valores de dichas variables, comprobando que no queden por encima o por debajo de

los límites críticos definidos por el usuario en el panel de personalización.

- Programación del panel de configuración del usuario, recogiendo las preferencias determinadas por el mismo para el intervalo de toma de datos y valores máximos y mínimos de temperatura (entre 10 y 30ºC) y acidez (entre 0 y 14 pH).
- Selección de ficheros de datos y eventos, desde los botones correspondientes del menú del formulario y que llaman a los subformularios *SelecData* y *SelecEvent*, respectivamente. Permiten seleccionar los ficheros donde se almacenará la información recogida a lo largo del funcionamiento del programa, tanto para el registro de los datos recibidos en el muestreo del entorno como para el registro de alarmas y eventos.
- Programación de los botones del panel de control, indicando qué función debe realizar cada uno de ellos en cada momento, atendiendo a la razón de su diseño e inclusión en el citado panel.
- Programación de los diversos Timers empleados en el programa para realizar diversas tareas, entre las que destacan:
  - Hora y fecha del sistema.
  - Contabilización del tiempo existente desde la última alimentación.
  - Temporizaciones de espera.
  - Toma de datos en intervalos fijos para su almacenamiento en registros.
- Programación de los distintos menús y opciones desplegables desde la barra de menús diseñados para el formulario, como volver al formulario anterior o salir del programa.
- Programación de los controles correspondientes a la detención del control del acuario, así como a su finalización definitiva cerrando el SCADA.

Cabe mencionar que, en el ámbito de las alarmas del sistema, existirán alarmas las cuales detendrán el control del acuario, como si el usuario hubiese accionado la parada de emergencia, aunque se esté hablando de control manual. Se ha decidido incluir dichas alarmas para evitar que el

usuario, al ser un control de carácter presencial, pueda no estar cerca del sistema durante un determinado tiempo y no sepa cuánto tiempo llevan las variables del sistema en estado crítico. El sistema presentará al usuario las siguientes alarmas en caso de ser necesario:

- Alarma por exceso o defecto de temperatura, pasado un tiempo establecido a partir del periodo de muestreo elegido por el usuario del SCADA.
- Alarma por exceso o defecto de acidez, pasado un tiempo establecido a partir del periodo de muestreo elegido, de la misma forma que en el caso anterior.

No se ha considerado la idea de incluir una alarma para un nivel de agua bajo, debido a que no es una situación crítica que pueda afectar al desarrollo del control o del ecosistema existente en el acuario; simplemente se indicará al usuario a través del panel de estado.

Las alarmas arriba mencionadas serán controladas por sendos Timer que comenzarán a contar el tiempo cuando se reciban muestras fuera de los límites.

- Form4 (*Auto*): de forma similar al anterior, este formulario permite controlar el funcionamiento del acuario, en este caso cuando es seleccionada la opción de control automático en el formulario de inicio.

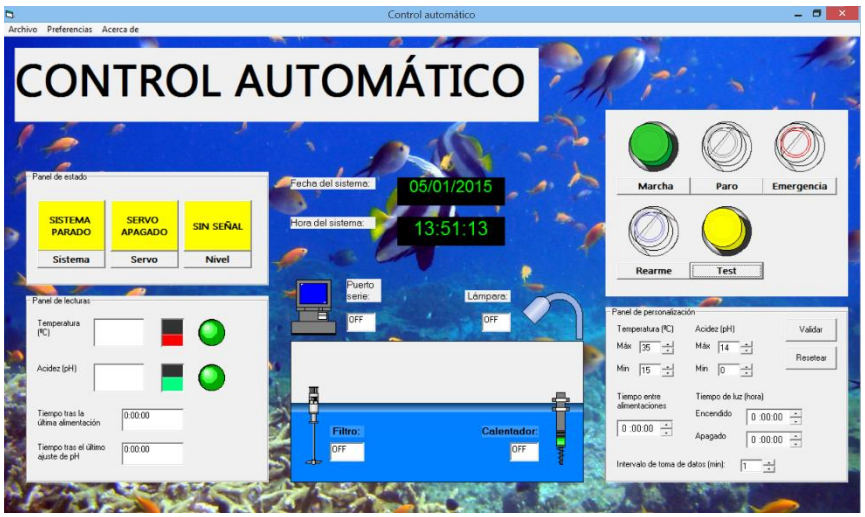


Figura 7.7.2.5 - Pantalla de control automático del SCADA

Debido a su similitud con el formulario anterior, y teniendo en cuenta que la programación de las partes comunes a ambos es casi idéntica, se comentarán los cambios existentes entre dicho formulario y el anteriormente mencionado, asociado al modo de control manual, y que hacen referencia a los paneles de botones y personalización. Al igual que para el anterior formulario, todo el código está recogido en su totalidad en el Anexo de códigos:

- Se ha eliminado el accionador del servomotor, debido a que al ser un control de carácter automático, el propio software SCADA debe encargarse de alimentar a los animales (a través de un periodo de tiempo determinado por el usuario) o de ajustar el pH del agua, revisando el valor de la variable que controla la magnitud de la acidez.
- Se han eliminado los controles asociados a la activación y desactivación de los dispositivos conectados a la placa de relés (filtro, termocalentador y lámpara); ahora el SCADA se encargará de controlar y regular el funcionamiento de dichos dispositivos de forma automática, en función de la situación en la que se encuentre el entorno en un determinado momento.
- El panel de personalización del usuario incluye ahora la posibilidad de determinar tanto el intervalo de tiempo entre alimentaciones como las horas de encendido y apagado de la lámpara.

Se han mantenido respecto al formulario anterior las alarmas presentes en el sistema en caso de que las variables de temperatura y acidez se salgan de los límites establecidos por el usuario.

- Form5 (*Tend*): se encarga de mostrar las gráficas de tendencia temporal tanto de temperatura como acidez, y es accesible desde el menú Archivo de la barra de menú, tanto desde el control manual como desde el automático. Dichas gráficas son actualizadas con cada llegada de los datos muestreados por la TAD, escribiendo un nuevo valor de las mismas en el instante de tiempo correspondiente. Dichas gráficas se han creado para que muestren la evolución de las variables en cuestión a lo largo de 48 horas ininterrumpidas, siendo reseteadas cuando se apaga el SCADA o cuando se alcanza el tiempo límite.

Figura 7.7.2.6 - Formulario de tendencias de temperatura y acidez

Debido a que cada vez que se cierra la ventana de tendencias y se vuelve a abrir posteriormente el programa resetea las gráficas, en el SCADA se ha incluido programación de tal manera que, tanto en control manual como automático, se guarden los valores históricos para “redibujarlos” cada vez que se vuelva a abrir la ventana, evitando así la pérdida de información.

Además de todos los “componentes” del programa hasta ahora mencionados, los formularios Form3, Form4 y Form5 hacen uso de los módulos Module1 (*Funciones*) y Module2 (*Variables*). La razón de que se haya decidido emplear la opción del módulo es que permite desarrollar funciones y variables de carácter público; es decir, a las que se puede acceder desde cualquier formulario. Esta opción, a pesar de estar disponible también para cada uno de los formularios desarrollados, provocaría que, si no se accede a ese formulario, dichas variables no llegarán a declararse y para los formularios restantes se comportarían como variables privadas.

En el primero de los módulos mencionados, el primer módulo diseñado contiene las siguientes funciones, las cuales han sido declaradas en el módulo en cuestión para que los formularios 3 y 4 puedan acceder fácilmente a ellas, y así evitar duplicarlas teniendo que escribirlas de forma idéntica dos veces:

- *dibujar*: permite dibujar inicialmente las gráficas de temperatura y acidez pertenecientes al Form5.

- *redibujar_temp*: redibuja la tendencia de temperatura reseteándola, una vez que se ha alcanzado el valor máximo de muestras de la misma.
- *redibujar_acid*: realiza la misma función que *Redibujar_temp*, aunque en este caso en referencia a la tendencia de acidez.
- *espera*: permite añadir el retardo en segundos que el usuario quiera especificar a través de una variable entera pasada como argumento.
- *toma_datos*: se encarga, cada vez que se reciben los datos de las variables muestreadas por el puerto serie, de escribir sus valores en el fichero de datos escogido previamente al arrancar el sistema.

El segundo módulo (*Variables*) tiene como función principal la declaración pública de variables que van a ser empleadas tanto en modo manual como en el automático de forma común. Con esta medida, se pretende evitar la declaración duplicada de una cantidad considerable de variables: al ser declaradas una única ve, se simplifica la programación y se produce un ahorro de memoria.

La programación de todos los formularios, así como de los módulos empleados, está descrita de forma detallada y tal cual se puede observar en el IDE de Arduino y en Visual Studio en el anexo de programación correspondiente (Anexo III).

7.7.3. Página de HTML

Haciendo referencia al título de este mismo apartado, se ha decidido denominar como página y no como programa al código a continuación descrito, debido a que HTML no es un lenguaje de programación en sí. Todo el código correspondiente a la conexión entre Arduino y la página web, así como el intercambio bidireccional de datos entre ambas, se explica con detenimiento en el apartado del programa de Arduino (7.5.1).

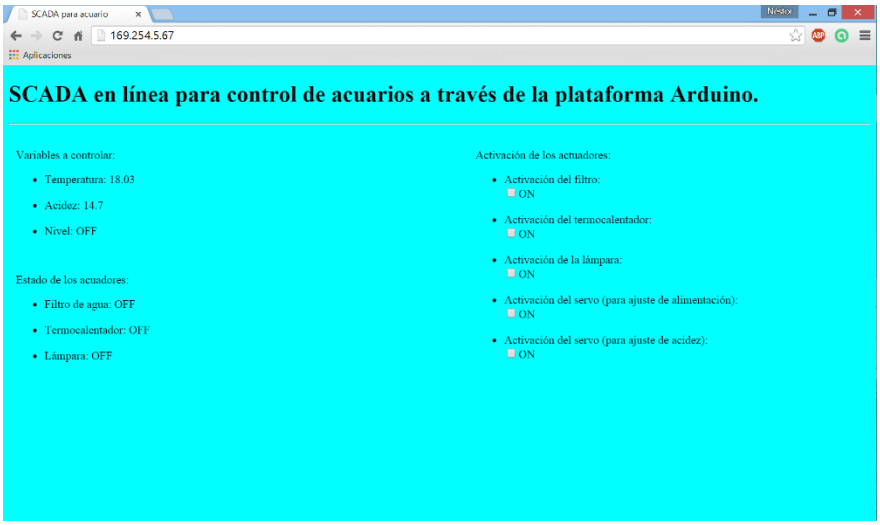


Figura 7.7.3.1 - Página web correspondiente al SCADA diseñado

La imagen superior muestra el aspecto que presenta el SCADA remoto para el usuario. De forma simple, el código correspondiente a la página web diseñada se puede dividir en las siguientes partes:

- Encabezado: muestra el título que llevará la página web, y que por lo tanto se mostrará en la pestaña del navegador empleado. Se corresponde con la sentencia correspondiente:

  <title>SCADA para acuario</title>

- Refresco: permite establecer un periodo en segundos tal que, cuando el mismo se alcance, la página se refresce o actualice automáticamente. En el código de Arduino diseñado, se puede observar que dicho periodo es independiente del periodo de muestreo del SCADA diseñado en VB.
  La sentencia empleada para ello en HTML es la siguiente:

  <meta http-equiv="refresh" content="num_segundos">

  Cabe mencionar que dicho periodo funcionará siempre que no se active o desactive alguno de los actuadores. Si esta acción se realizase, la página se actualizaría automáticamente, sin necesidad de que se alcanzase el timeout para el refresco. Se ha elegido un valor de 5 minutos.

- Cuerpo: situado entre las sentencias <body> y </body>, contiene como su propio nombre indica el cuerpo o parte principal de la página. En el código diseñado, contiene:
    - Una columna (izquierda) en la cual se muestran los valores de las variables procedentes de los sensores y el estado (ON/OFF) de los actuadores del sistema:

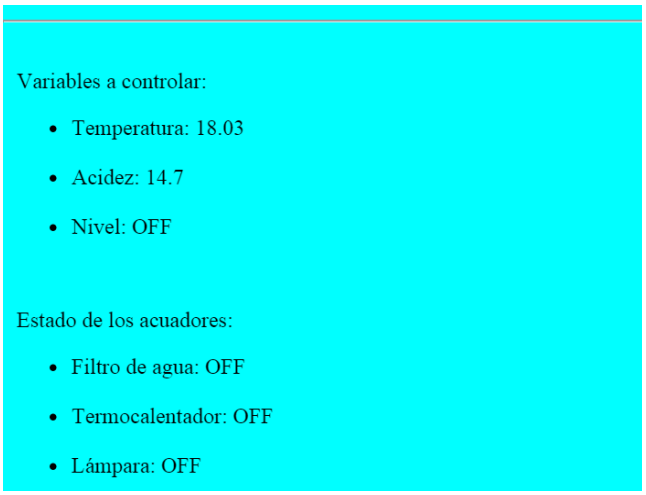


Figura 7.7.3.2 - Columna izquierda del SCADA web

    - Una columna (derecha) en la cual se muestran los botones para la activación/desactivación de los actuadores pertenecientes al acuario, los cuales serán:
        - Filtro.
        - Termocalentador.
        - Lámpara.
        - Servomotor (para alimentación).
        - Servomotor (para ajuste de acidez).

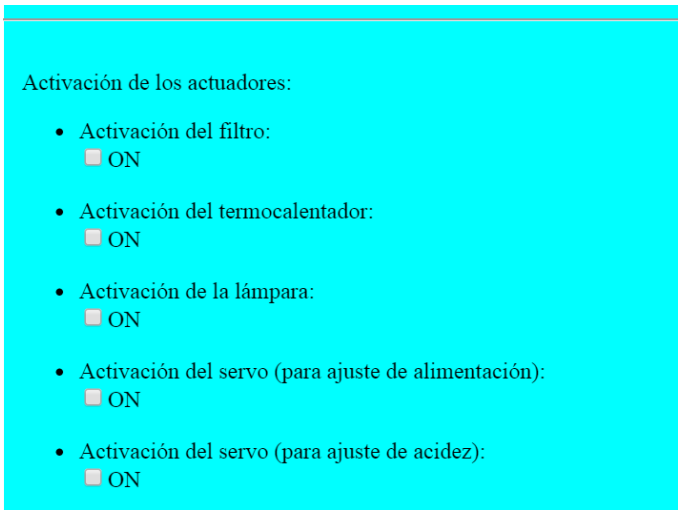


Figura 7.7.3.3 - Columna derecha del SCADA web

Tal y como se hace mención en el apartado 7.7.2, la placa Arduino está preparada para habilitar o deshabilitar las checkbox de la página web en función de si el modo elegido en el SCADA en VB es manual o automático. Como se comenta también en la parte correspondiente al código de Arduino más arriba, ello se debe a que no tendría mucho sentido poder actuar sobre los actuadores activándolos o desactivándolos si el usuario ha escogido el modo de funcionamiento automático y, por ello, no considera importante la necesidad de controlarlos él mismo.

8. RESULTADOS FINALES

En el siguiente apartado, y tras la lectura y comprensión de todo lo anterior a éste, se describirá el producto según las soluciones elegidas a lo largo de este documento.

El producto diseñado, como se ha redactado y definido a lo largo de todo el trabajo, se caracteriza por cumplir las siguientes funciones, una vez finalizado su desarrollo:

- Proceso de adquisición de datos mediante tarjeta Arduino MEGA 2560.
- Control del acuario a través de un SCADA desarrollado en VB.
- Posibilidad para el usuario de control remoto del acuario a través de la web.

Para su implementación física final, se han llevado diversos montajes e inclusiones de elementos que permiten una mejor presentación del producto de cara al posible futuro usuario. Dichas aportes a mayores de todo lo anterior, y que no guardan relación con la funcionalidad del sistema, se describirán en los siguientes apartados.

8.1. Placa de conexiones

Como punto aparte a todo el hardware existente en el sistema diseñado, se ha decidido desarrollar una placa de circuito impreso, con los objetivos consiguientes:

- Reducción del volumen de cableado necesario.
- Compartición de pistas para alimentación y masa, facilitando el empleo de las mismas.

Para la realización de la consiguiente placa se ha empleado el software de diseño de circuitos impresos Layout de Pspice, en combinación con el software de diseño de circuitos eléctricos y electrónicos OrCAD, del mismo fabricante.

Inicialmente, se ha desarrollado el circuito eléctrico correspondiente al real que contendrá dicha PCB en el segundo software mencionado, con un resultado obtenido como el de la figura siguiente obtenida del plano nº 02:

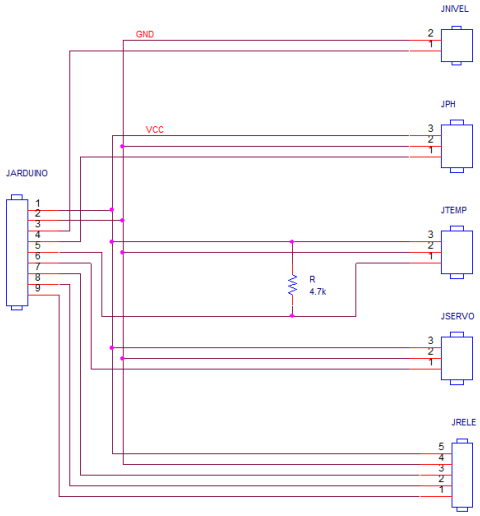


Figura 8.1.1 - Esquema eléctrico de la placa de conexión en OrCAD

En dicho diseño se pueden diferenciar diferentes headers, los cuales serán empleados de la siguiente manera:

- JARDUINO: llevará hasta la placa Arduino las señales procedentes de los sensores, relés y servo, y será el puente de conexión para la alimentación VCC y la masa GND desde Arduino al resto de componentes.
- JNIVEL: empleado para el flotador de nivel, transporta las señales de tierra del sensor y la señal de datos/alimentación.
- JPH: asociado al sensor de acidez, permite la conexión con Arduino de las señales de alimentación, masa y datos.
- JTEMP: funciona con el mismo conexionado que para el sensor de acidez, aunque en este caso este header irá asociado al sensor de temperatura, con la configuración que ello conlleva.
- JSERVO: al igual que los para las regletas mencionadas de temperatura y acidez, el servomotor empleado tendrá conexiones para tensión, masa y datos.
- JRELE: el último de los headers está destinado a los relés que controlan la lámpara, el filtro y el termocalentador. Por lo tanto, necesitará conexiones para encender/apagar cada uno de ellos, más las correspondientes comunes a tensión y 0V.

Con este diseño y empleando el software de diseño de PCBs de Pspice Layout Plus®, se realizó la implementación final de lo que sería más tarde la placa de conexiones. Las características básicas que se han tenido en cuenta en el diseño de la misma son:

- La placa tendrá unas dimensiones aproximadas de 10x8 cm, descritas con mayor exactitud en el plano nº 03.
- El ancho de las pistas empleado será de valor 40 según lo elegido en el programa empleado.
- El diámetro y posición de los taladros existentes a realizar, tanto los correspondientes a la sujeción de la placa como a la colocación de las regletas de conexión, estarán establecidos según lo descrito en el plano nº 03.

Teniendo en cuenta todo lo mencionado, se desarrolló el diseño partiendo de estas especificaciones. En la siguiente figura se muestra el resultado obtenido:

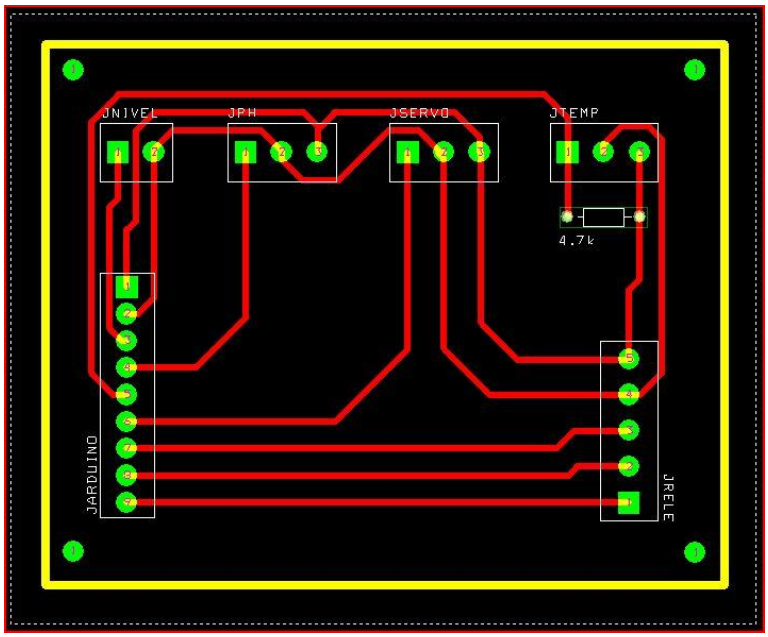


Figura 8.1.2 - Diseño en Layout Plus de la placa de conexiones

Tras tener preparado el diseño de la placa en el ámbito software, el mismo es plasmado físicamente. Para esto, se ha seguido el siguiente proceso:

1. Se imprimió el diseño en papel vegetal mediante el empleo de una impresora.
2. Utilizando una insoladora, el diseño quedó "grabado" en la placa fotosensible empleada para el proceso.
3. La placa es lavada con líquido revelador para marcar el circuito imprimido previamente del resto de la placa.
4. Finalmente, mediante una máquina de ácido se finalizó la creación del circuito impreso base, en cuyo proceso el ácido elimina todo el cobre de la placa que no pertenezca al diseño creado.

Tras la implementación física descrita, se llevó a cabo la soldadura de las diversas regletas atornilladas necesarias para el conexionado de los elementos del sistema, obteniendo un resultado final como el mostrado en la consiguiente imagen:

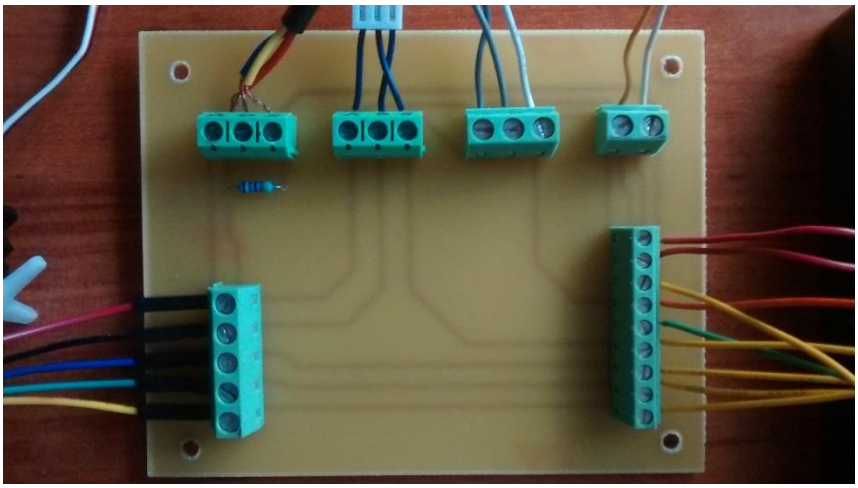


Figura 8.1.3 - Placa de circuito impreso para conexionado finalizada

Como último paso, la placa es atornillada a la caja modificada para la contención de las PCB empleadas en el sistema, descrita en el siguiente apartado. En la siguiente imagen se puede observar el resultado final:

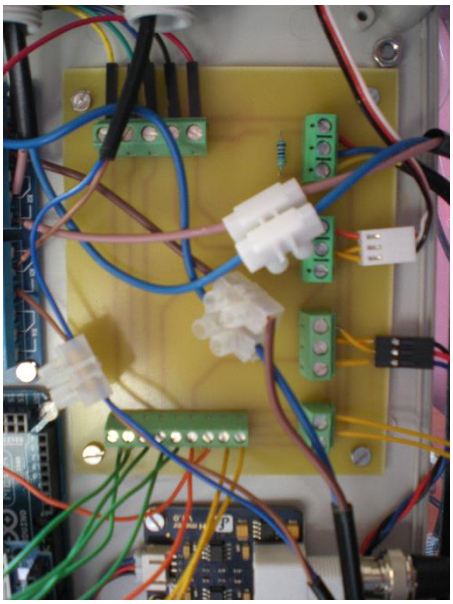


Figura 8.1.4 - PCB atornillada a la caja

Todos los planos correspondientes a los archivos desarrollados en OrCAD y Layout, así como a las dimensiones de la susodicha placa, están incluidos en el documento de planos de este trabajo.

8.2. Montaje de caja para circuitos

Como parte final del trabajo, y tras haber realizado todo lo anterior, se ha decidido por fines tanto prácticos como estéticos la instalación de una caja para montajes eléctricos en uno de los laterales del acuario, con el objetivo de facilitar el almacenamiento de todo el cableado y de las diversas PCB pertenecientes al sistema, evitando que su manipulación, en caso de ser necesaria, sea demasiado engorrosa para el usuario.

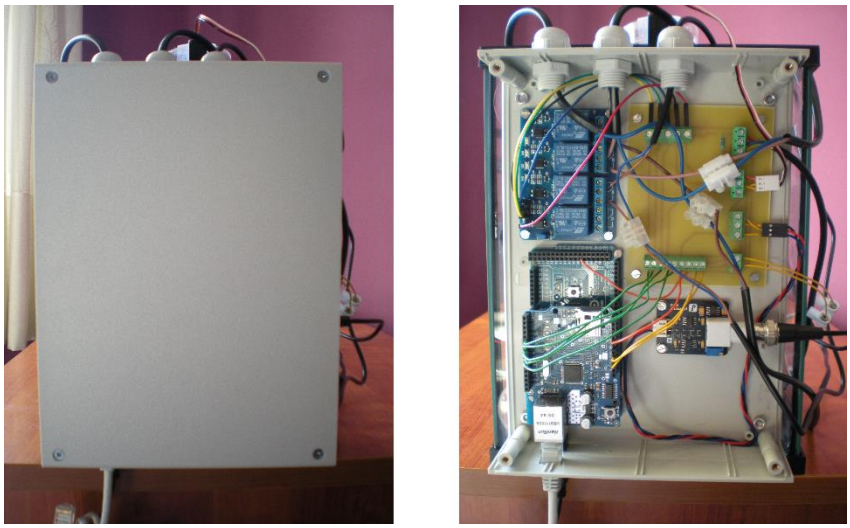


Figura 8.2.1 - Caja de circuitos tapada (izqda.) y abierta (drcha.)

La caja elegida para el proceso posee unas dimensiones aproximadas de 22 cm de alto por 16.5 cm de ancho (dichas dimensiones se encuentran descritas con exactitud en el plano nº 04), y sus especificaciones básicas se encuentran descritas en el apartado 1 del Pliego de Condiciones. Para su sujeción a la pared lateral del acuario se ha tomado la opción de emplear ventosas con tornillo, garantizando un soporte rígido y adecuado al tipo de superficie con la que hace contacto la caja.

Para la sujeción de las placas empleadas en el sistema, se ha optado por un proceso de atornillo simple, aprovechando en su mayoría los agujeros ya realizados por el fabricante en dichos circuitos. Por último, y para facilitar el aislamiento ante líquidos por la parte superior del acuario, se han colocado unas prensaestopas para los cables de los actuadores más voluminosos (filtro, termocalentador y lámpara).

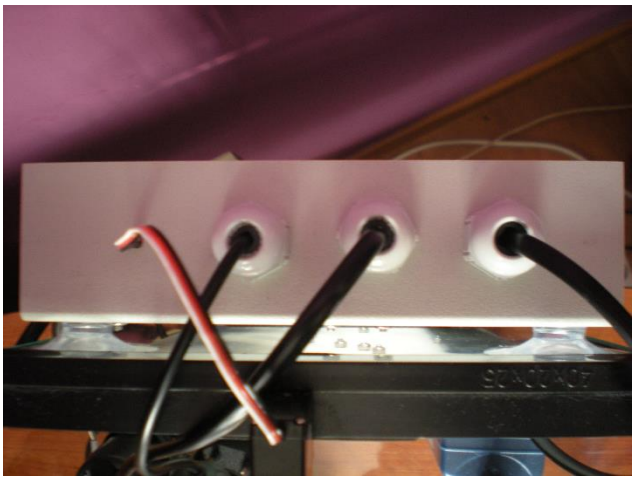


Figura 8.2.2 - Prensaestopas situadas en la parte superior de la caja

El plano correspondiente a los diversos agujeros a realizar en la caja para los tornillos de cada una de las placas y las prensaestopas, así como las dimensiones generales de la misma, puede ser consultado en el documento de planos de este trabajo.

9. ORDEN DE PRIORIDAD ENTRE LOS DOCUMENTOS BÁSICOS

Se establecerá, según lo dictado en la guía para la elaboración y presentación del trabajo de fin de grado, un orden de prioridad para los documentos básicos pertenecientes al consiguiente trabajo.

Por lo tanto, a continuación se mostrará el orden de prioridad elegido para los documentos realizados:

1. Memoria
2. Anexos
3. Planos
4. Pliego de condiciones
5. Estado de mediciones
6. Presupuesto

TÍTULO: **SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB**

ANEXOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBRERO, S/N

15405 - FERROL

FECHA: **FEBRERO DE 2015**

AUTOR: EL ALUMNO

Fdo.: NÉSTOR DE JUAN VÁZQUEZ

Índice

ANEXO I:

DOCUMENTACIÓN DE PARTIDA

ESCUELA UNIVERSITARIA POLITÉCNICA

ASIGNACIÓN DE TRABAJO FIN DE GRADO

En virtud de la solicitud efectuada por:
En virtud da solicitude efectuada por:

APELLIDOS, NOMBRE: *Juan Vázquez,Néstor De*
APELIDOS E NOME:

DNI: [redacted] **Fecha de Solicitud:** OCT2014
DNI: *Fecha de Solicitude:*

Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:

O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:

Título T.F.G:SCADA para control de acuarios mediante Arduino y VB

Número TFG: *770G01A66*

TUTOR:(Titor) *Prieto Guerreiro,Francisco*

COTUTOR/CODIRECTOR: *Jose Maria Cardona Comellas*

La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento*:*

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

Ferrol a Miercoles, 5 de Noviembre del 2014

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

Fdo: Juan Vázquez,Néstor De

DESCRIPCIÓN Y OBJETIVO:El objetivo del proyecto es llevar a cabo la creación de un SCADA para el control del funcionamiento de un acuario, utilizando para ello VB y arduino.

La aplicación deberá controlar y supervisar los elementos habituales en el funcionamiento de un acuario utilizando diversos sensores y actuadores ligados a la plataforma de bajo coste arduino. Dicha información será enviada a un SCADA realizado en VB, que permitira el control y supervision del sistema.

ANEXO II:

CÁLCULOS

Índice

1. ECUACIONES

A continuación se enunciarán, previo paso a todos los cálculos realizados en el trabajo, las ecuaciones empleadas, así como la definición principal del resultado o variable obtenida de cada una:

- Tasa de transferencia/baud rate: número de baudios transmitidos en el medio por unidad de tiempo.

$$baud\ rate = b_r = \frac{nº\ bits}{T}\ (baudios) \quad (1.1)$$

- Error absoluto: diferencia entre el valor medido y el valor exacto en valor absoluto.

$$error\ absoluto = e_a = Valor\ medido - Valor\ real \quad (1.2)$$

- Error relativo: error absoluto dividido entre el valor exacto.

$$error\ relativo = e_r\ (\%) = \frac{Valor\ medido - Valor\ real}{Valor\ medido} \cdot 100 \quad (1.3)$$

- LSB (Bit Menos Significativo): diferencia entre los códigos correspondientes a dos estados adyacentes, cuando la salida es expresada en estados y cada uno de ellos está representado por N códigos digitales.

$$LSB = \frac{rango\ de\ salida}{2^{nº\ bits}} \quad (1.4)$$

- Sensibilidad: pendiente de la curva de calibración.

$$Sensibilidad = \frac{salida}{entrada} \quad (1.5)$$

2. CÁLCULOS

2.1. Tasa de transferencia del proceso

La tasa de transferencia o velocidad de transmisión de la comunicación del sistema de control diseñado se calculará acorde a la expresión 1.1 del primer apartado de este documento.
Teniendo en cuenta las condiciones establecidas en el apartado 7.1.2 de la memoria de este trabajo, la velocidad de transferencia mínima que tendría que tener el sistema sería de:

$$b_r \geq \frac{120\ bits \cdot 1s}{220\ ms} =$$

$$b_r \geq 546\ baudios \qquad (2.1.1)$$

Cualquier tasa de transferencia por encima de la obtenida será válida. Teniendo en cuenta el valor escogido en la memoria (9600 baudios), el tiempo que tardaría en enviar los caracteres sería de:

$$T = \frac{130\ bits}{9600\ baudios} =$$

$$T = 13.5\ ms \qquad (2.1.2)$$

2.2. Error existente en el periodo empleando delay

Conforme a lo descrito en el apartado 7.1.3 del documento Memoria, a continuación se justifican los cálculos correspondientes a la exactitud del intervalo temporal del programa al emplear la función delay.
A continuación se muestra la tabla en la que se recogen los valores de los 10 primeros tiempos de ejecución del programa:

Tiempos de ejecución		
Nº de ciclo	Tiempo (us)	Diferencia (us)
1	1267000	-
2	2046772	779772
3	2826540	779768
4	3606308	779768
5	4386076	779768
6	5165840	779764
7	5945608	779768
8	6725376	779768
9	7505144	779768
10	8284904	779760

Tabla 2.2.1 - Medidas de los tiempos de ejecución del script

Con los datos recogidos, la media del tiempo de ejecución (3ª columna) es de:

$$\Delta T_e = \frac{779768\ \mu s \cdot 6 + 779760\ \mu s + 779764\ \mu s + 779772\ \mu s}{9} \cong\ 779767\ \mu s \quad (2.2.1)$$

El valor de retardo introducido deberá ser:

$$delay = 1 \cdot 10^6 s - 779767\ \mu s =$$

$$delay = 220233\ \mu s \cong 220\ ms \quad (2.2.2)$$

Una vez introducido este valor en el script, se obtiene, tomando dos tiempos cualesquiera al inicio de la ejecución, de:

$$\Delta T_e = 8486368\ \mu s - 7486588\ \mu s = 999780\ \mu s \quad (2.2.3)$$

Ello supondrá unos errores absolutos y relativos de valor igual al retardo introducido (ecuaciones 1.2 y 1.3):

$$e_a = 1 \cdot 10^6\ s - 999780\ \mu s = 220\ \mu s \quad (2.2.4)$$

$$e_r = \frac{1 \cdot 10^6 s - 999780 \mu s}{1 \cdot 10^6 s} \cdot 100 = 0.022\ \% \quad (2.2.5)$$

2.3. Sensibilidad y LSB del sensor DS18B20

Como se menciona en el apartado Análisis de las soluciones del documento Memoria, este sensor de temperatura es capaz de medir valores de temperatura entre los -55ºC y los +125ºC, presentando una resolución programable entre 9 y 12 bits.

Atendiendo a la programación realizada, la resolución empleada para el sensor es de 10 bits de forma predeterminada. El valor del LSB será, partiendo de la fórmula 1.4 descrita más arriba:

$$LSB_{10\ bits} = \frac{125ºC-(-55ºC)}{2^{10}} = 0.178\ ºC \quad (2.3.1)$$

Por otro lado, la sensibilidad del sensor valdrá, a través de la ecuación 1.5:

$$Sensibilidad = \frac{salida}{entrada} = \frac{5V-0V}{125ºC-(-55ºC)} = 27.8\ mV/ºC \quad (2.3.2)$$

2.4. Sensibilidad y LSB del sensor SEN-0161

Recurriendo de nuevo a la ecuación 1.3, es posible calcular el valor del LSB del sensor de acidez empleado en el sistema. Para ello, es necesario tener en cuenta dos consideraciones:

- El rango de valores disponibles se mueve entre 0 y 14 pH.
- La resolución del sensor, a través de su documentación, es de 10 bits.

$$LSB_{10\ bits} = \frac{14pH-0pH}{2^{10}} = 0.014\ pH \quad (2.4.1)$$

La sensibilidad del sensor mencionado será de:

$$Sensibilidad = \frac{salida}{entrada} = \frac{5V-0V}{14pH-0pH} = 0.36\ V/pH \quad (2.4.2)$$

3. BIBLIOGRAFÍA

Referencias a ecuaciones

(1.1) "Tasa de baudios", [en línea]. Noviembre 2014. Disponible en la web: http://es.wikipedia.org/wiki/Tasa_de_baudios.

(1.2) PÉREZ GARCÍA, MA; ÁLVAREZ ANTÓN, JC; CAMPO RODRÍGUEZ, JC. *Instrumentación electrónica.* 1ª ed. Madrid: Ediciones Paraninfo S.A, 2003, p. 20. ISBN: 9788497321662.

(1.3) PÉREZ GARCÍA, MA; ÁLVAREZ ANTÓN, JC; CAMPO RODRÍGUEZ, JC. *Instrumentación electrónica.* 1ª ed. Madrid: Ediciones Paraninfo S.A, 2003, p. 20. ISBN: 9788497321662.

(1.4) PÉREZ GARCÍA, MA; ÁLVAREZ ANTÓN, JC; CAMPO RODRÍGUEZ, JC. *Instrumentación electrónica.* 1ª ed. Madrid: Ediciones Paraninfo S.A, 2003, p. 619. ISBN: 9788497321662.

(1.5) PÉREZ GARCÍA, MA; ÁLVAREZ ANTÓN, JC; CAMPO RODRÍGUEZ, JC. *Instrumentación electrónica.* 1ª ed. Madrid: Ediciones Paraninfo S.A, 2003, p. 16. ISBN: 9788497321662.

ANEXO III:

CÓDIGOS

Índice

1. CÓDIGO DE ARDUINO

A continuación se muestra el código diseñado para el script de la tarjeta Arduino conforme a la aplicación desarrollada:

```
/*INCLUSIÓN DE LIBRERÍAS*/
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Servo.h>
#include <TimerOne.h>

/*DEFINICIÓN DE CONSTANTES*/
#define Pin_Temp 5
#define Pin_Acid 0
#define Pin_Nivel 42
#define termico 6
#define filtro 7
#define lampara 8
#define Offset_temp 0.1
#define Offset_acid 0.16

/*DEFINICIÓN DE VARIABLES PARA LA CONEXIÓN A INTERNET*/
byte mac[] = { 0xDE, 0x3C, 0x05, 0x00, 0x00, 0xF7 };
IPAddress ip(192, 168, 1, 67);
EthernetServer server(80);

/*VARIABLES EMPLEADAS*/
//Envío serie:
String trama;
//Temperatura:
OneWire tempProbe(Pin_Temp);
DallasTemperature sensors(&tempProbe);
float temp;
//Acidez:
unsigned long int lectura;
float acidez;
char ph[4];
String cad_acid;
String cero = "0";
```

```
String espacio = "";
//Nivel:
int nivel;
//Servomotor:
Servo servo;
//Actuadores y testeo:
String cad_test;
String respuesta;
String prueba = "TP00000000";
String prueba_serie = "COMCORRECT";
byte dato_serie;
int angulo;
//Timer1:
int cont;
unsigned int t_serie;
unsigned int t_elegido;
long t_defecto = 1000000;
//Archivo HTML:
File webFile;
unsigned long pos;
char char_in;
String HTTP_req;
//Control web:
const byte pin_rele[] = {6, 7, 8};
byte est_rele[sizeof(pin_rele)] = {0};
boolean activar_alim = false;
boolean activar_ajust = false;
boolean comprobar_alim = false;
boolean comprobar_acid = false;
boolean ult_accion[] = {false, false, false};
byte evitar_comp[] = {0, 0, 0};
//Control de errores:
String errorsd = "ES00000000";
String errorhtm = "EH00000000";

void setup()
{
   Ethernet.begin(mac, ip);
   server.begin();
   sensors.begin();

   pinMode(53, OUTPUT);
   pinMode(Pin_Nivel, INPUT);
```

```
    digitalWrite(Pin_Nivel, HIGH);
    servo.attach(9);
    servo.write(85);

    for (byte i = 0; i < sizeof(pin_rele); i++) {
        pinMode(pin_rele[i], OUTPUT);
        digitalWrite(pin_rele[i], HIGH);
    }

    Serial.begin(9600);

    error_1:
    if (!SD.begin(4)) {
        Serial.println(errorsd);
        goto error_1;
    }
    error_2:
    if (!SD.exists("html_web.htm")) {
        Serial.println(errorhtm);
        goto error_2;
    }

    t_elegido = 0;
    cont = 0;
    Timer1.initialize();
    Timer1.attachInterrupt(timerIsr);
}

/*FUNCIÓN DE INTERRUPCIÓN DEL TIMER1*/
void timerIsr()
{
  if (t_elegido > 0){
    cont += 1;
    if(cont >= t_elegido){
      trama = "";
      capture_acid();
      trama += cad_acid;
      capture_temp();
      trama += temp;
      capture_nivel();
      trama += nivel;
      Serial.println(trama);
      cont = 0;
```

```
    }
  }
}

void loop()
{
   lectura_serie();
   SCADA_web();
}

/*FUNCIÓN PARA EL SCADA WEB*/
void SCADA_web()
{
   EthernetClient client = server.available();

   if (client) {
      boolean currentLineIsBlank = true;
      while (client.connected()) {
         if (client.available()) {
            char_in = client.read();
            HTTP_req += char_in;
            if (char_in == '\n' && currentLineIsBlank) {
               client.println("HTTP/1.1 200 OK");
               client.println("Content-Type: text/html");
               client.println("Connnection: close");
               client.println();
               webFile = SD.open("index_2.htm", FILE_WRITE);
               if (webFile) {
                 muestreo();
                 evaluar_req();
                 esc_variables();
                 pos = 637;
                 webFile.seek(pos);
                 estado_actuadores(0);
                 pos = 694;
                 webFile.seek(pos);
                 estado_actuadores(1);
                 pos = 743;
                 webFile.seek(pos);
                 estado_actuadores(2);
                 pos = 1009;
                 webFile.seek(pos);
                 checkbox_rele(0);
```

```
                    pos = 1234;
                    webFile.seek(pos);
                    checkbox_rele(1);
                    pos = 1453;
                    webFile.seek(pos);
                    checkbox_rele(2);
                    evaluar_servo();
                    pos = 1698;
                    webFile.seek(pos);
                    servo_alim();
                    pos = 1893;
                    webFile.seek(pos);
                    servo_acid();
                    habilitar_checkbox();
                    webFile.close();
                  }
                  webFile = SD.open("html_web.htm");
                  if (webFile) {
                    while(webFile.available()) {
                      client.write(webFile.read());
                    }
                    webFile.close();
                  }
                  HTTP_req = "";
                  break;
                }
                if (char_in == '\n') {
                  currentLineIsBlank = true;
                }
                else if (char_in != '\r') {
                  currentLineIsBlank = false;
                }
              }
            }
            client.stop();
        }
}

/*HABILITACIÓN/INHABILITACIÓN DE LAS CHECKBOX DEL SCADA WEB*/
void habilitar_checkbox()
{
  if (modo == 1) {
    pos = 1105;
```

```
    webFile.seek(pos);
    webFile.print(" disabled ");
    pos = 1341;
    webFile.seek(pos);
    webFile.print(" disabled ");
    pos = 1570;
    webFile.seek(pos);
    webFile.print(" disabled ");
    pos = 1782;
    webFile.seek(pos);
    webFile.print(" disabled ");
    pos = 1987;
    webFile.seek(pos);
    webFile.print(" disabled ");
  }
  else if (modo == 2) {
    pos = 1105;
    webFile.seek(pos);
    webFile.print("          ");
    pos = 1341;
    webFile.seek(pos);
    webFile.print("          ");
    pos = 1570;
    webFile.seek(pos);
    webFile.print("          ");
    pos = 1782;
    webFile.seek(pos);
    webFile.print("          ");
    pos = 1987;
    webFile.seek(pos);
    webFile.print("          ");
  }
}

/*ACTUALIZACIÓN DEL ESTADO DE LOS RELÉS*/
void evaluar_req()
{
    for (int led_num = 0; led_num < sizeof(pin_rele); led_num++) {
    if ((HTTP_req.charAt(9) == (pin_rele[rele_num] + '0')) &&
                (evitar_comp[rele_num] == 1) &&
                (HTTP_req.charAt(16) == (pin_rele[rele_num] + '0'))) {
        est_rele[rele_num] = 0;
        evitar_comp[rele_num] = 0;
```

```
    }
      if ((HTTP_req.charAt(9) == (pin_rele[led_num] + '0')) &&
              (HTTP_req.charAt(16) == (pin_rele[led_num] + '0'))) {
        est_rele[led_num] = 1;
        comprobar_alim = false;
        comprobar_acid = false;
        Serial.print("CA");
        Serial.print(rele_num);
        Serial.println("1000000");
      }
      else if (HTTP_req.charAt(9) == (pin_rele[led_num] + '0')) {
        est_rele[led_num] = 0;
        comprobar_alim = false;
        comprobar_acid = false;
        Serial.print("CA");
        Serial.print(rele_num);
        Serial.println("0000000");
      }
    }
}

/*ACTUALIZACIÓN DEL ESTADO DEL SERVO*/
void evaluar_servo()
{
  if ((HTTP_req.charAt(9) == '9') && (comprobar_alim == false)) {
    activar_alim = true;
    comprobar_alim = true;
    Serial.println("SA00000000");
  }
  else if ((HTTP_req.charAt(9) == '9') && (comprobar_alim == true)) {
    activar_alim = false;
  }

  if ((HTTP_req.charAt(9) == '1') && (comprobar_acid == false)) {
    activar_ajust = true;
    comprobar_acid = true;
    Serial.println("SP00000000");
  }
  else if ((HTTP_req.charAt(9) == '1') && (comprobar_acid == true)) {
    activar_ajust = false;
  }
}
```

```
/*MUESTREO DE SEÑALES PARA EL SCADA WEB*/
void muestreo()

{
  capture_acid();
  capture_temp();
  capture_nivel();
}

/*ESCRITURA DE VARIABLES MUESTREADAS EN EL SCADA WEB*/
void esc_variables()
{
  pos = 433;
  webFile.seek(pos);
  if (temp != -126.84) {
    webFile.print(temp);
  }

  pos = 476;
  webFile.seek(pos);
  if (acidez < 10)
  {
    webFile.print('0' + acidez, 1);
  }
  else {
    webFile.print(acidez, 1);
  }

  pos = 517;
  webFile.seek(pos);
  if (nivel) {
    webFile.print("OFF");
  }
  else {
    webFile.print("ON ");
  }
}

/*ESCRITURA DE ESTADO DE ACTUADORES EN EL SCADA WEB*/
void estado_actuadores(int led_num)
{
  if (est_rele[led_num]) {
    webFile.print("ON ");
```

```
  }
  else {
    webFile.print("OFF");

  }
}

/*CREA LAS CHECKBOXES QUE PERMITEN ACTIVAR/DESACTIVAR LOS
ACTUADORES Y ACTUALIZA SU ESTADO*/
void checkbox_rele(int led_num) {
  webFile.print("<input type=\"hidden\" name=\"DIG");
  webFile.print(pin_rele[rele_num], DEC);
  webFile.print("\" value=\"0\"> <input type=\"checkbox\" name=\"DIG");
  webFile.print(pin_rele[rele_num], DEC);
  webFile.print("\" value=\"");
  webFile.print(pin_rele[rele_num], DEC);
  webFile.print("\"");
  if (est_rele[rele_num]) {
    webFile.print(" checked ");
    digitalWrite(pin_rele[rele_num], LOW);
  }
  else {
    webFile.print("         ");
    digitalWrite(pin_rele[rele_num], HIGH);
  }
  webFile.print(" onclick=\"submit();\">ON  ");
}

/*ACTIVACIÓN DEL SERVO PARA ALIMENTACIÓN POR SCADA WEB*/
void servo_alim()
{
  webFile.print("<input type=\"checkbox\" name=\"SRV1\" value=\"1\"");
  if (activar_alim) {
    webFile.print(" checked ");
    webFile.print(" onclick=\"submit();\">ON  ");
    servo.write(0);
    delay(500);
    servo.write(85);
    pos = 1698;
    webFile.seek(pos);
    webFile.print("<input type=\"checkbox\" name=\"SRV1\" value=\"1\"
onclick=\"submit();\">ON  ");
    activar_alim = false;
```

```
    comprobar_alim = true;
  }
  else {
    pos = 1743;
    webFile.print("          onclick=\"submit();\">ON  ");
  }
}

/*ACTIVACIÓN DEL SERVO PARA AJUSTE DE ACIDEZ POR SCADA WEB*/
void servo_acid()
{
  webFile.print("<input type=\"checkbox\" name=\"SRV2\" value=\"2\"");
  if (activar_ajust) {
    webFile.print(" checked ");
    webFile.print(" onclick=\"submit();\">ON  ");
    servo.write(180);
    delay(500);
    servo.write(85);
    pos = 1893;
    webFile.seek(pos);
    webFile.print("<input type=\"checkbox\" name=\"SRV2\" value=\"2\"
onclick=\"submit();\">ON  ");
    activar_ajust = false;
    comprobar_acid = true;
  }
  else {
    pos = 1938;
    webFile.print("          onclick=\"submit();\">ON  ");
  }
}

/*ACTIVACIÓN/DESACTIVACIÓN DE ACTUADORES Y LLAMADA A TESTEO Y
CAMBIO DE PERIODO*/
void lectura_serie(){
  if (Serial.available()){
    dato_serie = Serial.read();
    switch(dato_serie){
    case 0:
      digitalWrite(termico, LOW);
      est_rele[0] = 1;
      ult_accion[0] = true;
      break;
    case 1:
```

```
    digitalWrite(filtro, LOW);
    est_rele[1] = 1;
    ult_accion[1] = true;
    break;
  case 2:
    digitalWrite(lampara, LOW);
    est_rele[2] = 1;
    ult_accion[2] = true;
    break;
  case 3:
    digitalWrite(termico, HIGH);
    est_rele[0] = 0;
    if (ult_accion[0]) {
      evitar_comp[0] = 0;
      ult_accion[0] = false;
    }
    else {
      evitar_comp[0] = 1;
    }
    break;
  case 4:
    digitalWrite(filtro, HIGH);
    est_rele[1] = 0;
    if (ult_accion[1]) {
      evitar_comp[1] = 0;
      ult_accion[1] = false;
    }
    else {
      evitar_comp[1] = 1;
    }
    break;
  case 5:
    digitalWrite(lampara, HIGH);
    est_rele[2] = 0;
    if (ult_accion[2]) {
      evitar_comp[2] = 0;
      ult_accion[2] = false;
    }
    else {
      evitar_comp[2] = 1;
    }
    break;
  case 6:
```

```
      servo.write(0);
      delay(500);
      servo.write(85);
      break;
    case 7:
      servo.write(180);
      delay(500);
      servo.write(85);
      break;
    case 8:
      testeo();
      break;
    case 9:
      configurar();
      break;
    default:
      digitalWrite(termico, HIGH);
      digitalWrite(filtro, HIGH);
      digitalWrite(lampara, HIGH);
      servo.write(85);
      est_rele[0] = 0;
      est_rele[1] = 0;
      est_rele[2] = 0;
      break;
    }
  }
}

/*FUNCIÓN DE TESTEO*/
void testeo(){
  Timer1.stop();
  Serial.println(prueba);
  cad_test = "T";
  Serial.println(cad_test);
  cad_test = "";
  if (digitalRead(termico) == HIGH)
  {
    digitalWrite(termico, LOW);
  }
  delay(1000);
  if (digitalRead(termico) == LOW)
  {
    digitalWrite(termico, HIGH);
```

```
  respuesta = "YT";
}
else
{
  respuesta = "NT";
}
cad_test += respuesta;
delay(1000);
if (digitalRead(filtro) == HIGH)
{
  digitalWrite(filtro, LOW);
}
delay(1000);
if (digitalRead(filtro) == LOW)
{
  digitalWrite(filtro, HIGH);
  respuesta = "YF";
}
else
{
  respuesta = "NF";
}
cad_test += respuesta;
delay(1000);
if (digitalRead(lampara) == HIGH)
{
  digitalWrite(lampara, LOW);
}
delay(1000);
if (digitalRead(lampara) == LOW)
{
  digitalWrite(lampara, HIGH);
  respuesta = "YL";
}
else
{
  respuesta = "NL";
}
cad_test += respuesta;
delay(1000);
servo.write(45);
delay(500);
angulo = servo.read();
```

```
  if (angulo == 45){
    servo.write(85);
    delay(500);
    angulo = servo.read();
    if (angulo == 85){
      respuesta = "YS";
    }
    else
    {
      respuesta = "NS";
    }
    cad_test += respuesta;
  }
  delay(1000);
  respuesta = "T";
  cad_test += respuesta;
  Serial.println(cad_test);
  Timer1.start();
}

/*CONFIGURACIÓN INICIAL DEL SISTEMA*/
void configurar(){
  Timer1.stop();
  while(Serial.available() <= 0){}
  t_serie = Serial.read();
  while(Serial.available() <= 0){}
  modo = Serial.read();
  t_elegido = t_serie * 60;
  trama = "";
  capture_acid();
  trama += cad_acid;
  volver:
  capture_temp();
  if (temp == -126.84) {
    goto volver;
  }
  trama += temp;
  capture_nivel();
  trama += nivel;
  cont = 0;
  Timer1.initialize(t_defecto);
  Serial.println(prueba_serie);
  Serial.println(trama);
```

```
}

/*LECTURA DEL VALOR DE ACIDEZ*/
void capture_acid()
{
  lectura = analogRead(Pin_Acid);
  acidez = (float)lectura*5.0/1024;
  acidez = 3.5*acidez + Offset_acid;
  dtostrf(acidez, 3, 1, ph);
  if(acidez < 10)
  {
    cad_acid = cero + ph;
  }
  else
  {
    cad_acid = espacio + ph;
  }
}

/*LECTURA DEL VALOR DE TEMPERATURA*/
void capture_temp()
{
  sensors.requestTemperatures();
  temp = sensors.getTempCByIndex(0) + Offset_temp;
}

/*LECTURA DEL VALOR DEL NIVEL*/
void capture_nivel()
{
  nivel = digitalRead(Pin_Nivel);
}
```

2. CÓDIGO DE VISUAL BASIC

En el caso del código diseñado para el SCADA en VB, para facilitar la comprensión del lector se dividirá el mismo en códigos por formulario en forma de apartados. A continuación se muestran las composiciones de dichos formularios.

2.1. Formulario 1 (Inicio.frm)

```
Option Explicit
'Determina el modo de funcionamiento
Dim modo As Integer
'Variables auxiliares
Dim pregunta As Integer

Private Sub Form_Load()
Image1.Picture = ImageList1.ListImages(1).Picture

Form1.Height = 10000
Form1.Width = 13700
modo = 2
End Sub

'Botón de inicio del control
Private Sub Comenzar_Click()
If (modo < 1) Or (modo > 2) Then
   MsgBox "¡No se ha seleccionado ningún modo!", vbOKOnly + vbCritical, "Error en la configuración"
   ElseIf modo = 1 Then
      Comenzar.Rotation = Rotate270
      Form4.Show
      Form1.Hide
      Comenzar.Rotation = Rotate0
   Else
      Comenzar.Rotation = Rotate270
```

```
        Form3.Show
        Form1.Hide
        Comenzar.Rotation = Rotate0
End If
End Sub

'Botón de información
Private Sub Info_Click()
Form2.Show
End Sub

'Elección de modo automático
Private Sub Auto_Click()
modo = 1
End Sub

'Elección de modo manual
Private Sub Manual_Click()
modo = 2
End Sub

'Salida del SCADA
Private Sub Salir_Click()
pregunta = MsgBox("¿Está seguro de que quiere salir?", vbYesNo +
vbInformation, "Salida del programa")
If pregunta = 6 Then
   End
End If
End Sub

'Salida del SCADA (mediante control de Windows)
Private Sub Form_Unload(Cancel As Integer)
pregunta = MsgBox("¿Está seguro de que quiere salir?", vbYesNo +
```

```
vbInformation, "Salida del programa")
If pregunta = 6 Then
   End
End If
End Sub
```

2.2. Formulario 2 (Info.frm)

```
Private Sub Form_Load()
Image1.Picture = ImageList1.ListImages(1).Picture
Image2.Picture = ImageList1.ListImages(2).Picture

Form2.Height = 6000
Form2.Width = 5000
End Sub

'Botón de vuelta al formulario de origen
Private Sub Volver_Click()
Form2.Hide
End Sub
```

2.3. Formulario 3 (Manual.frm)

```
Option Explicit
'Variables para control temporal de temperatura y acidez
Dim cont_temp As Integer, cont_acid As Integer
'Variables para control de alimentación y ajuste de acidez
Dim t_alim As Date, t_acid As Date, alim As Boolean, ajust As Boolean
'Procedimiento para poder ejecutar archivos (abrir los registros de texto)
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal HWnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long

Private Sub Form_Load()
MSComm1.RThreshold = 12
MSComm1.InputLen = 12
MSComm1.InBufferSize = 12
MSComm1.Settings = "9600,n,8,1"
MSComm1.CommPort = 3
MSComm1.DTREnable = True
MSComm1.RTSEnable = True

Form3.Height = 10700
Form3.Width = 18300
Form3.WindowState = 2
Image1.Picture = ImageList1.ListImages(1).Picture
Image1.Stretch = True

s_nivel = "SIN SEÑAL"
s_servo = "OFF"
s_filtro = "OFF"
s_lampara = "OFF"
s_termo = "OFF"
```

```
Xt_1 = 0
YT_1 = 0
YA_1 = 0
temp = 0
acid = 0
cont = 0
i = 1
v_temp(0) = 0
v_acid(0) = 0
cont_temp = 0
cont_acid = 0
lim_t_temp = 0
lim_t_acid = 0
modo_arduino = 2
error = False
nivel = False
valid = False
registro_data = False
registro_event = False
control = False
test_puerto = False
test_lampara = False
test_filtro = False
test_termo = False
test_servo = False
alim = False
ajust = False
tend = False
t_alim = #12:00:00 AM#
t_acid = #12:00:00 AM#

Timer1.Enabled = False
Timer2.Enabled = True
```

```
Timer4.Enabled = False
Timer5.Enabled = False
Timer6.Enabled = False
Timer7.Enabled = False
Timer8.Enabled = False
Timer9.Enabled = False
Timer10.Enabled = False

SFStandard2.Visible = True
SFStandard4.Visible = True
SFStandard6.Visible = True
SFStandard8.Visible = True
SFStandard10.Visible = True
SFStandard2.FillColorMode = Original
SFStandard4.FillColorMode = Hollow
SFStandard6.FillColorMode = Hollow
SFStandard8.FillColorMode = Original
SFStandard10.FillColorMode = Hollow
SFStandard12.FillColorMode = Hollow
SFStandard13.FillColorMode = Hollow
SFStandard14.FillColorMode = Hollow
SFStandard15.FillColorMode = Hollow
SFStandard16.FillColorMode = Hollow
SFStandard3.Visible = False
SFStandard5.Visible = False
SFStandard7.Visible = False
SFStandard9.Visible = False
SFStandard11.Visible = False
SFStandard16.Flip = 0
SFStandard17.FillColor = vbGreen
SFStandard18.FillColor = vbGreen
SFCutaway1.Level = 20
SFCutaway2.Level = 7
```

```
AbrirData.Enabled = False
AbrirEvent.Enabled = False
AbrirTend.Enabled = False
Command7.Caption = "SISTEMA PARADO"
Command7.BackColor = vbYellow
Command9.Caption = "SERVO APAGADO"
Command9.BackColor = vbYellow
Command11.Caption = "SIN SEÑAL"
Command11.BackColor = vbYellow
Validar.Enabled = True
Resetear.Enabled = False

Text4.Text = t_alim
Text5.Text = t_acid
Text6.Text = "OFF"
Text7.Text = "OFF"
Text8.Text = "OFF"
Text9.Text = "OFF"
Text10.Text = UpDown1.Max
Text11.Text = UpDown2.Min
Text12.Text = UpDown3.Max
Text13.Text = UpDown4.Min
Text14.Text = UpDown5.Min
End Sub

'Función para ajustar la imagen de fondo del formulario
Private Sub Form_Resize()
With Image1
    .Left = 0
    .Top = 0
    .Width = Me.Width
    .Height = Me.Height
End With
```

```
End Sub

'Subformulario para control de recepción de datos
Private Sub MSComm1_OnComm()
If MSComm1.CommEvent = comEvReceive Then
   sData = MSComm1.Input
   Text3.Text = sData
   primer_char = Mid$(sData, 1, 1)
   segundo_char = Mid$(sData, 2, 1)
   mensaje = Mid$(sData, 1, 10)
   If mensaje = "COMCORRECT" Then
      control = True
   End If
   If (primer_char = "C") And (segundo_char = "A") Then
      num_actuador = CInt(Val(Mid$(sData, 3, 1)))
      operacion = CInt(Val(Mid$(sData, 4, 1)))
      Call actualizar
   End If
   If primer_char = "E" Then
      error = True
      Call reiniciar
      MSComm1.PortOpen = False
      If segundo_char = "S" Then
         MsgBox "Error: no se detecta la tarjeta SD", vbOKOnly + vbCritical, "Error
de inicialización"
            ElseIf segundo_char = "H" Then
               MsgBox "Error: no se detecta el archivo .htm", vbOKOnly + vbCritical,
"Error de inicialización"
      End If
   Else
      error = False
   End If
   If primer_char = "S" Then
```

```
        If segundo_char = "A" Then
            t_alim = #12:00:00 AM#
            ElseIf segundo_char = "P" Then
                t_acid = #12:00:00 AM#
        End If
    End If
    If (primer_char <> "T") And (primer_char <> "E") And (primer_char <> "C") And
(primer_char <> "S") Then
        lect_acid = Mid$(sData, 1, 4)
        lect_temp = Mid$(sData, 5, 5)
        lect_nivel = Mid$(sData, 10, 1)
        temp = CSng(Val(lect_temp))
        acid = CSng(Val(lect_acid))
        nivel = CBool(Val(lect_nivel))
        If test = False Then
            Text1.Text = temp
            Text2.Text = acid
            SFCutaway1.Level = temp
            SFCutaway2.Level = acid

            Call toma_datos

            'Control del nivel del acuario
            If nivel = False Then
                Command11.Caption = "NIVEL CORRECTO"
                Command11.BackColor = vbGreen
                s_nivel = "CORRECTO"
            Else
                Command11.Caption = "NIVEL INCORRECTO"
                Command11.BackColor = vbRed
                s_nivel = "INCORRECTO"
            End If
```

```
        'Control de la temperatura
        If ((temp >= min_temp) And (temp <= (min_temp + 1.5))) Or ((temp <=
max_temp) And (temp >= (max_temp - 1.5))) Then
            SFStandard17.FillColor = vbYellow
            ElseIf (temp < min_temp) Or (temp > max_temp) Then
                SFStandard17.FillColor = vbRed
                Timer9.Enabled = True
            Else
                SFStandard17.FillColor = vbGreen
        End If

        'Control de la acidez
        If ((acid >= min_acid) And (acid <= (min_acid + 1))) Or ((acid <= max_acid)
And (acid >= (max_acid - 1))) Then
            SFStandard18.FillColor = vbYellow
            ElseIf (acid < min_acid) Or (acid > max_acid) Then
                SFStandard18.FillColor = vbRed
                Timer10.Enabled = True
            Else
                SFStandard18.FillColor = vbGreen
        End If

       'Dibujado de gráficas
        YT = temp
        YA = acid
        If (IsNumeric(YT)) And (IsNumeric(YA)) Then
            If Xt_1 = (48 / toma) Then
                Call Redibujar_temp
                Call Redibujar_acid
                Xt_1 = 0
                i = 1
                v_temp(0) = v_temp(CInt(48 / toma))
                v_acid(0) = v_acid(CInt(48 / toma))
```

```
            End If
            Xt = Xt_1 + toma
            Form5.Picture1.Line (Xt_1, YT_1)-(Xt, YT), vbRed, BF
            Form5.Picture2.Line (Xt_1, YA_1)-(Xt, YA), vbBlue, BF
            Xt_1 = Xt
            YT_1 = YT
            YA_1 = YA
            v_temp(i) = temp
            v_acid(i) = acid
            i = i + 1
        End If
    End If
End If
If primer_char = "T" Then
    If segundo_char = "P" Then
        test_puerto = True
    Else
        str_termo = Mid$(sData, 4, 2)
        str_filtro = Mid$(sData, 6, 2)
        str_lampara = Mid$(sData, 8, 2)
        str_servo = Mid$(sData, 10, 2)

        If str_termo = "YT" Then
            test_termo = True
            ElseIf str_termo = "NT" Then
                test_termo = False
        End If
        If str_filtro = "YF" Then
            test_filtro = True
            ElseIf str_filtro = "NF" Then
                test_filtro = False
        End If
        If str_lampara = "YL" Then
```

```
            test_lampara = True
            ElseIf str_lampara = "NL" Then
                test_lampara = False
        End If
        If str_servo = "YS" Then
            test_servo = True
            ElseIf str_servo = "NS" Then
                test_servo = False
        End If
        Call result_test(test_puerto, test_lampara, test_filtro, test_termo,
test_servo)
      End If
   End If
End If
End Sub

'Funciones de ajuste de acidez por parte del usuario para el control UpDown
Private Sub UpDown3_UpClick()
If Text12.Text >= 14 Then
   Text12.Text = 14
Else
   Text12.Text = Text12.Text + 0.1
End If
End Sub

Private Sub UpDown3_DownClick()
If Text12.Text <= 0 Then
   Text12.Text = 0
Else
   Text12.Text = Text12.Text - 0.1
End If
End Sub
```

```
Private Sub UpDown4_UpClick()
If Text13.Text >= 14 Then
   Text13.Text = 14
Else
   Text13.Text = Text13.Text + 0.1
End If
End Sub

Private Sub UpDown4_DownClick()
If Text13.Text <= 0 Then
   Text13.Text = 0
Else
   Text13.Text = Text13.Text - 0.1
End If
End Sub

'Elección del registro de datos
Private Sub SelecData_Click()
CommonDialog1.InitDir = "C:\Users\Néstor\Documents"
CommonDialog1.DialogTitle = "Abrir archivo para guardar muestras"
CommonDialog1.Filter = "Archivos de texto|*.txt|Documentos de Word|*.docx|Todos los Archivos|*.*"
CommonDialog1.ShowOpen
If CommonDialog1.FileName <> "" Then
   If CommonDialog1.FileName = CommonDialog2.FileName Then
      MsgBox "¡No se puede elegir el mismo fichero para datos y eventos!", vbOKOnly + vbExclamation, "Error de selección de fichero"
      ElseIf registro_data = False Then
         Open CommonDialog1.FileName For Append As #1
         Print #1, "--------------------------------------------------------------------------------"
         Print #1, "Registro de medidas tomadas para control de alarmas"
         Print #1, "Fecha de elección de registro: " & Date & " " & Time
```

```
        Print #1, vbCrLf
        AbrirData.Enabled = True
        registro_data = True
    End If
End If
End Sub

'Elección del registro de eventos
Private Sub SelecEvent_Click()
CommonDialog2.InitDir = "C:\Users\Néstor\Documents"
CommonDialog2.DialogTitle = "Abrir archivo para guardar eventos"
CommonDialog2.Filter = "Archivos de texto|*.txt|Documentos de Word|*.docx|Todos los Archivos|*.*"
CommonDialog2.ShowOpen
If CommonDialog2.FileName <> "" Then
    If CommonDialog2.FileName = CommonDialog1.FileName Then
        MsgBox "¡No se puede elegir el mismo fichero para datos y eventos!", vbOKOnly + vbExclamation, "Error de selección de fichero"
        ElseIf registro_event = False Then
            Open CommonDialog2.FileName For Append As #2
            Print #2, "--------------------------------------------------------------------------------"
            Print #2, "Registro de eventos ocurridos para control de alarmas"
            Print #2, "Fecha de elección de registro: " & Date & " " & Time
            Print #2, vbCrLf
            AbrirEvent.Enabled = True
            registro_event = True
    End If
End If
End Sub

Private Sub UpDown3_UpClick()
If Text12.Text >= 14 Then
```

```
    Text12.Text = 14
Else
    Text12.Text = Text12.Text + 0.1
End If
End Sub

Private Sub UpDown3_DownClick()
If Text12.Text <= 0 Then
    Text12.Text = 0
Else
    Text12.Text = Text12.Text - 0.1
End If
End Sub

Private Sub UpDown4_UpClick()
If Text13.Text >= 14 Then
    Text13.Text = 14
Else
    Text13.Text = Text13.Text + 0.1
End If
End Sub

Private Sub UpDown4_DownClick()
If Text13.Text <= 0 Then
    Text13.Text = 0
Else
    Text13.Text = Text13.Text - 0.1
End If
End Sub

'Botón de validación de las preferencias del usuario
Private Sub Validar_Click()
max_temp = CSng(Val(Text10.Text))
```

```
min_temp = CSng(Val(Text11.Text))
max_acid = CSng(Val(Text12.Text))
min_acid = CSng(Val(Text13.Text))
periodo = CSng(Val(Text14.Text))
toma = periodo / 60

If (periodo < 30) Then
   lim_t_temp = 60
   lim_t_acid = 60
   ElseIf (periodo >= 30) And (periodo < 45) Then
      lim_t_temp = 90
      lim_t_acid = 90
      ElseIf (periodo >= 45) And (periodo <= 60) Then
         lim_t_temp = 130
         lim_t_acid = 130
End If

Validar.Enabled = False
Resetear.Enabled = True
UpDown1.Enabled = False
UpDown2.Enabled = False
UpDown3.Enabled = False
UpDown4.Enabled = False
UpDown5.Enabled = False

If (max_temp <= min_temp) Or (max_acid <= min_acid) Then
   MsgBox "¡Los valores máximos y mínimos escogidos no son correctos!", vbOKOnly + vbCritical, "Error en la personalización"
Else
   valid = True
End If
End Sub
```

```
'Reseteo de las preferencias elegidas por el usuario
Private Sub Resetear_Click()
Text10.Text = UpDown1.Max
Text11.Text = UpDown2.Min
Text12.Text = UpDown3.Max
Text13.Text = UpDown4.Min
Text14.Text = UpDown5.Min
Resetear.Enabled = False
Validar.Enabled = True
UpDown1.Enabled = True
UpDown2.Enabled = True
UpDown3.Enabled = True
UpDown4.Enabled = True
UpDown5.Enabled = True

lim_t_temp = 0
lim_t_acid = 0

If valid = True Then
   valid = False
End If
End Sub

'Botón de puesta en marcha
Private Sub SFStandard2_Click()
If registro_data = False Then
   MsgBox "¡No ha seleccionado un fichero como registro de datos!", vbOKOnly +
vbCritical, "Error en la configuración"
   ElseIf registro_event = False Then
      MsgBox "¡No ha seleccionado un fichero como registro de eventos!",
vbOKOnly + vbCritical, "Error en la configuración"
      ElseIf valid = False Then
         MsgBox "¡No ha validado las variables de personalización!", vbOKOnly +
```

```
vbCritical, "Error en la configuración"
    Else
        SFStandard2.Visible = False
        SFStandard3.Visible = True
        Resetear.Enabled = False
        Form6.Height = 1500
        Form6.Label1.Alignment = 2
        Form6.ProgressBar1.Visible = False
        Form6.Label1.Caption = "Estableciendo comunicación..."
        Form6.Show
        MSComm1.PortOpen = True
        retardo = 2
        Call Espera(retardo)
        If error = False Then
            orden = 9
            MSComm1.Output = Chr(orden)
            MSComm1.Output = Chr(periodo)
            MSComm1.Output = Chr(modo_arduino)
            Form6.Hide
            control = True
            Print #2, "Se activó el botón de puesta en marcha a las " & Time
            Print #2, vbCrLf
            Timer4.Enabled = True
            Timer5.Enabled = True

            AbrirData.Enabled = True
            AbrirEvent.Enabled = True
            AbrirTend.Enabled = True
            SelecData.Enabled = False
            SelecEvent.Enabled = False
            SFStandard4.Visible = True
            SFStandard5.Visible = False
            SFStandard4.FillColorMode = Original
```

```
            SFStandard8.FillColorMode = Hollow
            SFStandard10.FillColorMode = Original
            SFStandard12.FillColorMode = Original
            SFStandard13.FillColorMode = Original
            SFStandard14.FillColorMode = Original
            SFStandard16.FillColorMode = Original
            Text6.Text = "ON"
            Command7.Caption = "SISTEMA ACTIVO"
            Command7.BackColor = vbGreen
        End If
End If
End Sub

'Botón de paro
Private Sub SFStandard4_Click()
orden = 10
MSComm1.Output = Chr(orden)
MSComm1.PortOpen = False
control = False
Print #2, "Se activó el botón de paro a las " & Time
Print #2, vbCrLf

Resetear.Enabled = True
SFStandard2.Visible = True
SFStandard3.Visible = False
SFStandard4.Visible = False
SFStandard5.Visible = True
SFStandard6.Visible = True
SFStandard7.Visible = False
SFStandard8.Visible = True
SFStandard9.Visible = False
SFStandard10.Visible = True
SFStandard11.Visible = False
```

```
SFStandard2.FillColorMode = Original
SFStandard6.FillColorMode = Hollow
SFStandard8.FillColorMode = Original
SFStandard10.FillColorMode = Hollow
SFStandard12.FillColorMode = Hollow
SFStandard13.FillColorMode = Hollow
SFStandard14.FillColorMode = Hollow
SFStandard15.FillColorMode = Hollow
SFStandard16.FillColorMode = Hollow
Text6.Text = "OFF"
Command7.Caption = "SISTEMA PARADO"
Command7.BackColor = vbYellow
Command9.Caption = "SERVO APAGADO"
Command9.BackColor = vbYellow
Command11.Caption = "SIN SEÑAL"
Command11.BackColor = vbYellow
End Sub

'Botón de rearme
Private Sub SFStandard6_Click()
Print #2, "Se activó el botón de rearme a las " & Time
Print #2, vbCrLf
Timer8.Enabled = False

Form6.Height = 2000
Form6.Label1.Alignment = 0
Form6.ProgressBar1.Visible = True
Form6.Hide
Image1.Picture = ImageList1.ListImages(1).Picture
Command7.Caption = "SISTEMA EN REARME"
Command7.BackColor = vbBlue
SFStandard6.Visible = False
SFStandard7.Visible = True
```

```
SFStandard10.Visible = True
SFStandard11.Visible = False
SFStandard10.FillColorMode = Original
Timer6.Enabled = True
End Sub

'Botón de test
Private Sub SFStandard8_Click()
If registro_data = False Then
   MsgBox "¡No ha seleccionado un fichero como registro de datos!", vbOKOnly + vbCritical, "Error en la configuración"
   ElseIf registro_event = False Then
      MsgBox "¡No ha seleccionado un fichero como registro de eventos!", vbOKOnly + vbCritical, "Error en la configuración"
   Else
      Print #2, "Se activó el botón de testeo a las " & Time
      Print #2, vbCrLf

      SFStandard2.Visible = True
      SFStandard3.Visible = False
      SFStandard4.Visible = True
      SFStandard5.Visible = False
      SFStandard6.Visible = True
      SFStandard7.Visible = False
      SFStandard8.Visible = False
      SFStandard9.Visible = True
      SFStandard10.Visible = True
      SFStandard11.Visible = False
      SFStandard2.FillColorMode = Hollow
      SFStandard4.FillColorMode = Hollow
      SFStandard6.FillColorMode = Hollow
      SFStandard10.FillColorMode = Hollow
      Text6.Text = "OFF"
```

```
        Command7.Caption = "TEST EN MARCHA"
        Command7.BackColor = vbYellow
        Form6.Height = 2000
        Form6.Label1.Alignment = 0
        Form6.ProgressBar1.Visible = True
        Form6.ProgressBar1.Value = 0
        Form6.Label1.Caption = "Testeando comunicación..."
        Form6.Show

        MSComm1.PortOpen = True
        retardo = 2
        Call Espera(retardo)

        orden = 8
        MSComm1.Output = Chr(orden)

        Form6.ProgressBar1.Value = 25
        Form6.Label1.Caption = "Testeando lámpara..."
        Call Espera(retardo)
        Form6.ProgressBar1.Value = 50
        Form6.Label1.Caption = "Testeando filtro..."
        Call Espera(retardo)
        Form6.ProgressBar1.Value = 75
        Form6.Label1.Caption = "Testeando termocalentador..."
        Call Espera(retardo)
        Form6.ProgressBar1.Value = 100
        Form6.Label1.Caption = "Testeando servo..."
        Call Espera(retardo)
        Form6.Hide
End If
End Sub
```

```
'Botón de parada de emergencia
Private Sub SFStandard10_Click()
Print #2, "Se activó la seta de EMERGENCIA a las " & Time
Print #2, vbCrLf
Print #1, "Valores de magnitudes en el momento de la parada: " & Time
Print #1, "Temperatura", "Acidez", "Nivel", "Servo", "Filtro", "Lámpara", "Termocalentador"
Print #1, Format(temp, "#00.00"), Format(acid, "##00.0"), s_nivel, s_servo, s_filtro, s_lampara, s_termo

Call Emergencia
End Sub

'Activación/desactivación del relé correspondiente a la lámpara
Private Sub SFStandard12_Click()
If SFStandard12.Flip = 0 Then
   SFStandard12.Flip = 2
   SFStandard15.FillColorMode = Original
   orden = 0
   s_lampara = "ON"
   Print #2, "Se activó la lámpara a las " & Time
   Print #2, vbCrLf
   Text7.Text = "ON"
   ElseIf SFStandard12.Flip = 2 Then
      SFStandard12.Flip = 0
      If (SFStandard13.Flip = 2) Or (SFStandard14.Flip = 2) Then
         SFStandard15.FillColorMode = Original
      Else
         SFStandard15.FillColorMode = Hollow
      End If
      orden = 3
      s_lampara = "OFF"
      Print #2, "Se desactivó la lámpara a las " & Time
```

```
        Print #2, vbCrLf
        Text7.Text = "OFF"
End If
If SFStandard15.Flip = 2 Then
    SFStandard15.Flip = 0
End If
MSComm1.Output = Chr(orden)
End Sub

'Activación/desactivación del relé correspondiente al filtro
Private Sub SFStandard13_Click()
If SFStandard13.Flip = 0 Then
    SFStandard13.Flip = 2
    SFStandard15.FillColorMode = Original
    orden = 1
    s_filtro = "ON"
    Print #2, "Se activó el filtro a las " & Time
    Print #2, vbCrLf
    Text8.Text = "ON"
    Timer7.Enabled = True
    ElseIf SFStandard13.Flip = 2 Then
        SFStandard13.Flip = 0
        If (SFStandard12.Flip = 2) Or (SFStandard14.Flip = 2) Then
            SFStandard15.FillColorMode = Original
        Else
            SFStandard15.FillColorMode = Hollow
        End If
        orden = 4
        s_filtro = "OFF"
        Print #2, "Se desactivó el filtro a las " & Time
        Print #2, vbCrLf
        Text8.Text = "OFF"
        Timer7.Enabled = False
```

```
End If
If SFStandard15.Flip = 2 Then
   SFStandard15.Flip = 0
End If
MSComm1.Output = Chr(orden)
End Sub

'Activación/desactivación del relé correspondiente al calentador
Private Sub SFStandard14_Click()
If SFStandard14.Flip = 0 Then
   MsgBox "El selector del calentador es manual. La elección de la temperatura depende del propio usuario.", vbOKOnly + vbInformation, "Salida del programa"
   SFStandard14.Flip = 2
   SFStandard15.FillColorMode = Original
   s_termo = "ON"
   Print #2, "Se activó el termocalentador a las " & Time
   Print #2, vbCrLf
   Text9.Text = "ON"
   orden = 2
   ElseIf SFStandard14.Flip = 2 Then
      SFStandard14.Flip = 0
      If (SFStandard12.Flip = 2) Or (SFStandard13.Flip = 2) Then
         SFStandard15.FillColorMode = Original
      Else
         SFStandard15.FillColorMode = Hollow
      End If
      s_termo = "OFF"
      Print #2, "Se desactivó el termocalentador a las " & Time
      Print #2, vbCrLf
      Text9.Text = "OFF"
      orden = 5
End If
If SFStandard15.Flip = 2 Then
```

```
    SFStandard15.Flip = 0
End If
MSComm1.Output = Chr(orden)
End Sub

'Desactivación de todos los relés
Private Sub SFStandard15_Click()
orden = 10
MSComm1.Output = Chr(orden)
Print #2, "Se desactivaron los relés a las " & Time
Print #2, vbCrLf

SFStandard15.Flip = 2
SFStandard12.Flip = 0
SFStandard13.Flip = 0
SFStandard14.Flip = 0
Text7.Text = "OFF"
Text8.Text = "OFF"
Text9.Text = "OFF"
retardo = 1
Call Espera(retardo)
SFStandard15.Flip = 0
SFStandard15.FillColorMode = Hollow
End Sub

'Activación del servo
Private Sub SFStandard16_Click()
pregunta = MsgBox("¿Ejecutar proceso de alimentación?", vbYesNo + vbExclamation, "Elección de acción")
If pregunta = 6 Then
    SFStandard16.Flip = 2
    s_servo = "ON"
    Print #2, "Se activó el servo a las " & Time
```

```
    Print #2, vbCrLf
    Command9.Caption = "SERVO ACTIVO"
    Command9.BackColor = vbGreen
    alim = True
    Timer1.Enabled = True
    orden = 6
    MSComm1.Output = Chr(orden)
Else
    pregunta = MsgBox("¿Ejecutar ajuste de acidez?", vbYesNo + vbExclamation,
"Elección de acción")
    If pregunta = 6 Then
        SFStandard16.Flip = 2
        s_servo = "ON"
        Print #2, "Se activó el servo a las " & Time
        Print #2, vbCrLf
        Command9.Caption = "SERVO ACTIVO"
        Command9.BackColor = vbGreen

        ajust = True
        Timer1.Enabled = True
        orden = 7
        MSComm1.Output = Chr(orden)
    End If
End If
End Sub

'Temporizador de espera del botón del servo
Private Sub Timer1_Timer()
cont = cont + 1
If cont >= 2 Then
    cont = 0
    s_servo = "OFF"
    Command9.Caption = "SERVO APAGADO"
```

```
    Command9.BackColor = vbYellow
    SFStandard16.Flip = 0
    Timer1.Enabled = False
End If
End Sub

'Reloj para fecha y hora del sistema
Private Sub Timer2_Timer()
Label7.Caption = Date
Label6.Caption = Time
End Sub

'Timer para control de tiempo desde que se realizó la última alimentación
Private Sub Timer4_Timer()
t_alim = t_alim + #12:00:01 AM#
Text4.Text = t_alim
If alim = True Then
    t_alim = #12:00:00 AM#
    alim = False
End If
End Sub

'Timer para control de tiempo desde que se realizó el último ajuste de acidez
Private Sub Timer5_Timer()
t_acid = t_acid + #12:00:01 AM#
Text5.Text = t_acid
If ajust = True Then
    t_acid = #12:00:00 AM#
    ajust = False
End If
End Sub
```

```
'Reloj encargado del proceso de rearme
Private Sub Timer6_Timer()
cont = cont + 1
If cont < 2 Then
Form6.Show
Form6.Label1.Caption = "Rearmando el sistema..."
Form6.ProgressBar1.Value = 0
End If
If cont = 2 Then
    Form6.ProgressBar1.Value = 25
    Form6.Label1.Caption = "Reiniciando los actuadores..."
    MSComm1.PortOpen = True
    orden = 10
    MSComm1.Output = Chr(orden)
    MSComm1.PortOpen = False
    ElseIf cont = 4 Then
        Form6.ProgressBar1.Value = 50
        Form6.Label1.Caption = "Reiniciando variables..."
        s_nivel = "SIN SEÑAL"
        s_servo = "OFF"
        s_filtro = "OFF"
        s_lampara = "OFF"
        s_termo = "OFF"
        Xt_1 = 0
        YT_1 = 0
        YA_1 = 0
        temp = 0
        acid = 0
        cont_temp = 0
        cont_acid = 0
        lim_t_temp = 0
        lim_t_acid = 0
        i = 1
```

```
error = False
nivel = False
control = False
test_puerto = False
test_lampara = False
test_filtro = False
test_termo = False
test_servo = False
alim = False
ajust = False
t_alim = #12:00:00 AM#
t_acid = #12:00:00 AM#
ElseIf cont = 6 Then
   Form6.ProgressBar1.Value = 75
   Form6.Label1.Caption = "Reiniciando apariencia..."
   SFStandard2.Visible = True
   SFStandard4.Visible = True
   SFStandard6.Visible = True
   SFStandard8.Visible = True
   SFStandard10.Visible = True
   SFStandard2.FillColorMode = Original
   SFStandard4.FillColorMode = Hollow
   SFStandard6.FillColorMode = Hollow
   SFStandard8.FillColorMode = Original
   SFStandard10.FillColorMode = Hollow
   SFStandard12.FillColorMode = Hollow
   SFStandard13.FillColorMode = Hollow
   SFStandard14.FillColorMode = Hollow
   SFStandard15.FillColorMode = Hollow
   SFStandard16.FillColorMode = Hollow
   SFStandard3.Visible = False
   SFStandard5.Visible = False
   SFStandard7.Visible = False
```

```
SFStandard9.Visible = False
SFStandard11.Visible = False
SFStandard16.Flip = 0
SFStandard17.FillColor = vbGreen
SFStandard18.FillColor = vbGreen
SFCutaway1.Level = 20
SFCutaway2.Level = 7
Command7.Caption = "SISTEMA REARMADO"
Command9.Caption = "SERVO APAGADO"
Command9.BackColor = vbYellow
Command11.Caption = "SIN SEÑAL"
Command11.BackColor = vbYellow
Text1.Text = ""
Text2.Text = ""
Text4.Text = #12:00:00 AM#
Text5.Text = #12:00:00 AM#
Text6.Text = "OFF"
Text7.Text = "OFF"
Text8.Text = "OFF"
Text9.Text = "OFF"
Text10.Text = UpDown1.Max
Text11.Text = UpDown2.Min
Text12.Text = UpDown3.Max
Text13.Text = UpDown4.Min
Text14.Text = UpDown5.Min

Call Redibujar_temp
Call Redibujar_acid

Form6.ProgressBar1.Value = 100
ElseIf cont = 7 Then
   Form6.Hide
   cont = 0
```

```
            Timer6.Enabled = False
End If
End Sub

'Timer para el movimiento del gráfico del filtro
Private Sub Timer7_Timer()
If SFStandard19.Flip = 0 Then
   SFStandard19.Flip = 1
   ElseIf SFStandard19.Flip = 1 Then
      SFStandard19.Flip = 0
End If
End Sub

'Timer para situación de emergencia
Private Sub Timer8_Timer()
Beep
If Image1.Picture = ImageList1.ListImages(1).Picture Then
   Image1.Picture = LoadPicture("")
   Form3.BackColor = vbRed
   ElseIf Form3.BackColor = vbRed Then
      Image1.Picture = ImageList1.ListImages(1).Picture
End If
End Sub

'Temporizador para el control de temperatura
Private Sub Timer9_Timer()
cont_temp = cont_temp + 1
If (cont_temp >= lim_t_temp) Then
   If (temp < min_temp) Or (temp > max_temp) Then
      Form6.Height = 1500
      Form6.Label1.Alignment = 2
      Form6.ProgressBar1.Visible = False
      Form6.Label1.Caption = "¡LA TEMPERATURA ESTÁ FUERA DE LOS
```

```
LÍMITES!"
        Form6.Show

        Call emergencia
        cont_temp = 0
    Else
        cont_temp = 0
        Timer9.Enabled = False
    End If
End If
End Sub

'Timer para el control de la acidez
Private Sub Timer10_Timer()
cont_acid = cont_acid + 1
If (cont_acid >= lim_t_acid) Then
    If (acid < min_acid) Or (acid > max_acid) Then
        Form6.Height = 1500
        Form6.Label1.Alignment = 2
        Form6.ProgressBar1.Visible = False
        Form6.Label1.Caption = "¡LA ACIDEZ ESTÁ FUERA DE LOS LÍMITES!"
        Form6.Show

        Call emergencia
        cont_acid = 0
    Else
        cont_acid = 0
        Timer10.Enabled = False
    End If
End If
End Sub
```

```
'Función preparada para ejecutar una situación de emergencia
Function emergencia()
MSComm1.PortOpen = False
Timer1.Enabled = False
Timer4.Enabled = False
Timer5.Enabled = False
Timer7.Enabled = False
Timer9.Enabled = False
Timer10.Enabled = False

SFStandard10.Visible = False
SFStandard11.Visible = True
SFStandard2.Visible = True
SFStandard3.Visible = False
SFStandard4.Visible = True
SFStandard5.Visible = False
SFStandard2.FillColorMode = Hollow
SFStandard4.FillColorMode = Hollow
SFStandard6.FillColorMode = Original
SFStandard8.FillColorMode = Hollow
SFStandard12.FillColorMode = Hollow
SFStandard13.FillColorMode = Hollow
SFStandard14.FillColorMode = Hollow
SFStandard15.FillColorMode = Hollow
SFStandard16.FillColorMode = Hollow

Text6.Text = "OFF"
Command7.Caption = "PARADO POR EMERGENCIA"
Command7.BackColor = vbRed
Command9.Caption = "PARADO POR EMERGENCIA"
Command9.BackColor = vbRed
Command11.Caption = "PARADO POR EMERGENCIA"
Command11.BackColor = vbRed
```

```
Timer8.Enabled = True
End Function

'Función para la obtención de resultados del proceso de testeo
Function result_test(ByVal a As Boolean, ByVal b As Boolean, ByVal c As Boolean, ByVal d As Boolean, ByVal e As Boolean)
If a = True Then
   c_puerto = "correcta"
Else
   c_puerto = "incorrecta"
End If
If b = True Then
   c_lampara = "correcto"
Else
   c_lampara = "incorrecto"
End If
If c = True Then

   c_filtro = "correcto"
Else
   c_filtro = "incorrecto"
End If
If d = True Then
   c_termo = "correcto"
Else
   c_termo = "incorrecto"
End If
If e = True Then
   c_servo = "correcto"
Else
   c_servo = "incorrecto"
End If
```

```
MsgBox "Comunicación serie: " & c_puerto & Chr(13) & _
" Funcionamiento de lámpara: " & c_lampara & Chr(13) & _
" Funcionamiento de filtro: " & c_filtro & Chr(13) & _
" Funcionamiento de termoc.: " & c_termo & Chr(13) & _
" Funcionamiento de servo: " & c_servo, vbOKOnly, "Resultados del testeo"

Print #2, "Resultados del test:"
Print #2, "Comunicación serie: " & c_puerto & vbCrLf & _
"Funcionamiento de lámpara: " & c_lampara & vbCrLf & _
"Funcionamiento de filtro: " & c_filtro & vbCrLf & _
"Funcionamiento de termoc.: " & c_termo & vbCrLf & _
"Funcionamiento de servo: " & c_servo
Print #2, vbCrLf

a = False
b = False
c = False
d = False
e = False

SFStandard8.Visible = True
SFStandard9.Visible = False
SFStandard2.FillColorMode = Original
Command7.Caption = "SISTEMA PARADO"
Command7.BackColor = vbYellow
MSComm1.PortOpen = False
End Function

'Actualización del estado gráfico de los actuadores
Function actualizar()
If (num_actuador = 1) And (operacion = 1) Then
   SFStandard13.Flip = 2
   ElseIf (num_actuador = 1) And (operacion = 0) Then
```

```
        SFStandard13.Flip = 0
End If
If (num_actuador = 2) And (operacion = 1) Then
    SFStandard12.Flip = 2
    ElseIf (num_actuador = 2) And (operacion = 0) Then
        SFStandard12.Flip = 0
End If
If (num_actuador = 0) And (operacion = 1) Then
    SFStandard14.Flip = 2
    ElseIf (num_actuador = 0) And (operacion = 0) Then
        SFStandard14.Flip = 0
End If
If (SFStandard12.Flip = 0) And (SFStandard13.Flip = 0) And (SFStandard14.Flip =
0) Then
    SFStandard15.FillColorMode = Hollow
Else
    SFStandard15.FillColorMode = Original
End If

End Function

'Función para el reinicio del sistema
Function reiniciar()
s_nivel = "SIN SEÑAL"
s_servo = "OFF"
s_filtro = "OFF"
s_lampara = "OFF"
s_termo = "OFF"
Xt_1 = 0
YT_1 = 0
YA_1 = 0
temp = 0
acid = 0
```

```
cont = 0
i = 1
v_temp(0) = 0
v_acid(0) = 0
cont_temp = 0
cont_acid = 0
lim_t_temp = 0
lim_t_acid = 0
error = False
nivel = False
valid = False
registro_data = False
registro_event = False
control = False
test_puerto = False
test_lampara = False
test_filtro = False
test_termo = False
test_servo = False
alim = False
ajust = False
tend = False
t_alim = #12:00:00 AM#
t_acid = #12:00:00 AM#

Timer1.Enabled = False
Timer2.Enabled = True
Timer4.Enabled = False
Timer5.Enabled = False
Timer6.Enabled = False
Timer7.Enabled = False
Timer8.Enabled = False
Timer9.Enabled = False
```

```
Timer10.Enabled = False

SFStandard2.Visible = True
SFStandard4.Visible = True
SFStandard6.Visible = True
SFStandard8.Visible = True
SFStandard10.Visible = True
SFStandard2.FillColorMode = Original
SFStandard4.FillColorMode = Hollow
SFStandard6.FillColorMode = Hollow
SFStandard8.FillColorMode = Original
SFStandard10.FillColorMode = Hollow
SFStandard12.FillColorMode = Hollow
SFStandard13.FillColorMode = Hollow
SFStandard14.FillColorMode = Hollow
SFStandard15.FillColorMode = Hollow
SFStandard16.FillColorMode = Hollow
SFStandard3.Visible = False
SFStandard5.Visible = False
SFStandard7.Visible = False
SFStandard9.Visible = False
SFStandard11.Visible = False
SFStandard16.Flip = 0
SFStandard17.FillColor = vbGreen
SFStandard18.FillColor = vbGreen
SFCutaway1.Level = 20
SFCutaway2.Level = 7
AbrirData.Enabled = False
AbrirEvent.Enabled = False
AbrirTend.Enabled = False
SelecData.Enabled = True
SelecEvent.Enabled = True
Command7.Caption = "SISTEMA PARADO"
```

```
Command7.BackColor = vbYellow
Command9.Caption = "SERVO APAGADO"
Command9.BackColor = vbYellow
Command11.Caption = "SIN SEÑAL"
Command11.BackColor = vbYellow
Validar.Enabled = True
Resetear.Enabled = False
Text4.Text = #12:00:00 AM#
Text5.Text = #12:00:00 AM#
Text6.Text = "OFF"
Text7.Text = "OFF"
Text8.Text = "OFF"
Text9.Text = "OFF"
Text10.Text = UpDown1.Max
Text11.Text = UpDown2.Min
Text12.Text = UpDown3.Max
Text13.Text = UpDown4.Min
Text14.Text = UpDown5.Min
End Function

'Función encargada de ejecutar operaciones para cerrar el SCADA
Function terminar_control()
If MSComm1.PortOpen = False Then
   MSComm1.PortOpen = True
End If
orden = 10
MSComm1.Output = Chr(orden)
MSComm1.PortOpen = False
Close #1
Close #2
End
End Function
```

```
'Botón de menú encargado de abrir el fichero de datos
Private Sub AbrirData_Click()
ret_data = ShellExecute(Me.HWnd, "Open", CommonDialog1.FileName, "", "", 1)
End Sub

'Botón de menú encargado de abrir el fichero de eventos
Private Sub AbrirEvent_Click()
ret_event = ShellExecute(Me.HWnd, "Open", CommonDialog2.FileName, "", "", 1)
End Sub

'Botón de menú encargado de abrir el formulario de tendencias
Private Sub AbrirTend_Click()
Form5.Show
End Sub

'Botón de menú encargado de abrir la ventana de información
Private Sub Acercade_Click()
Form2.Show
End Sub

'Permite volver a la pantalla inicial del sistema (Formulario 1)
Private Sub Volver_Click()
If control = True Then
   MsgBox "¡Debe parar antes el control!", vbOKOnly + vbCritical, "Error en la configuración"
Else
   pregunta = MsgBox("¿Está seguro de que quiere volver?", vbYesNo + vbInformation, "Salida del programa")
   If pregunta = 6 Then
      If MSComm1.PortOpen = False Then
         MSComm1.PortOpen = True
      End If
      orden = 10
```

```
        MSComm1.Output = Chr(orden)
        MSComm1.PortOpen = False
        Close #1
        Close #2

        Call reiniciar

        Form1.Show
        Form3.Hide
    End If
End If
End Sub

'Salida del SCADA y cierre del programa mediante el control "Salir"
Private Sub Salir_Click()
If control = True Then
    MsgBox "¡Debe parar antes el control!", vbOKOnly + vbCritical, "Error en la configuración"
Else
    pregunta = MsgBox("¿Está seguro de que quiere salir?", vbYesNo + vbInformation, "Salida del programa")
    If pregunta = 6 Then
        Call terminar_control
    End If
End If
End Sub

'Salida del SCADA y cierre del programa por control de Windows
Private Sub Form_Unload(Cancel As Integer)
If control = True Then
    MsgBox "¡Debe parar antes el control!", vbOKOnly + vbCritical, "Error en la configuración"
Else
```

```
    pregunta = MsgBox("¿Está seguro de que quiere salir?", vbYesNo + vbInformation, "Salida del programa")
    If pregunta = 6 Then
        Call terminar_control
    End If
End If
End Sub
```

2.4. Formulario 4 (Auto.frm)

```
Option Explicit
'Variables para control temporal de temperatura y acidez
Dim cont_temp As Integer, cont_ph As Integer, cont_envio As Integer
'Variables auxiliares específicas del modo automático
Dim cont_alim As Integer, cont_acid As Integer
'Variables de personalización del sistema
Dim t_aux As Date, t_lampara_on As Date, t_lampara_off As Date
'Flag auxiliar empleado para control automático de la lámpara
Dim lampara As Boolean
'Variables para control de alimentación y ajuste de acidez
Dim t_alim As Date, t_acid As Date, h_alim As Date, elec_alim As Date, elec_acid
As Boolean, total_alim As Date, total_acid As Boolean, alim As Boolean, ajust As
Boolean
'Procedimiento para poder ejecutar archivos (abrir los registros de texto)
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA"
(ByVal HWnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal
lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long)
As Long

Private Sub Form_Load()
MSComm1.RThreshold = 12
MSComm1.InputLen = 12
MSComm1.InBufferSize = 12
MSComm1.Settings = "9600,n,8,1"
MSComm1.CommPort = 3
MSComm1.DTREnable = True
MSComm1.RTSEnable = True

Form4.Height = 10700
Form4.Width = 18300
Form4.WindowState = 2
```

```
Image1.Picture = ImageList1.ListImages(1).Picture
Image1.Stretch = True

s_nivel = "SIN SEÑAL"
s_servo = "OFF"
s_filtro = "OFF"
s_lampara = "OFF"
s_termo = "OFF"
Xt_1 = 0
YT_1 = 0
YA_1 = 0
temp = 0
acid = 0
cont = 0
i = 1
v_temp(0) = 0
v_acid(0) = 0
cont_alim = 0
cont_acid = 0
cont_temp = 0
cont_ph = 0
cont_envio = 0
lim_t_temp = 0
lim_t_acid = 0
modo_arduino = 1
error = False
nivel = False
lampara = False
termo = False
filtro = False
valid = False
registro_data = False
registro_event = False
```

control = False
test_puerto = False
test_lampara = False
test_filtro = False
test_termo = False
test_servo = False
alim = False
ajust = False
tend = False
DTPicker1.Value = #12:00:00 AM#
DTPicker2.Value = #12:00:00 AM#
DTPicker3.Value = #12:00:00 AM#
t_alim = #12:00:00 AM#
t_acid = #12:00:00 AM#
h_alim = #12:00:00 AM#

Timer1.Enabled = False
Timer2.Enabled = True
Timer4.Enabled = False
Timer5.Enabled = False
Timer6.Enabled = False
Timer7.Enabled = False
Timer8.Enabled = False
Timer9.Enabled = False
Timer10.Enabled = False
Timer11.Enabled = False
Timer12.Enabled = False
Timer13.Enabled = False
Timer14.Enabled = False

AbrirData.Enabled = True
AbrirEvent.Enabled = True
AbrirTend.Enabled = True

```
SFStandard2.Visible = True
SFStandard4.Visible = True
SFStandard6.Visible = True
SFStandard8.Visible = True
SFStandard10.Visible = True
SFStandard3.Visible = False
SFStandard5.Visible = False
SFStandard7.Visible = False
SFStandard9.Visible = False
SFStandard11.Visible = False
SFStandard4.FillColorMode = Hollow
SFStandard6.FillColorMode = Hollow
SFStandard8.FillColorMode = Original
SFStandard10.FillColorMode = Hollow
SFStandard17.FillColor = vbGreen
SFStandard18.FillColor = vbGreen
SFCutaway1.Level = 20
SFCutaway2.Level = 7
AbrirData.Enabled = False
AbrirEvent.Enabled = False
AbrirTend.Enabled = False
Command7.Caption = "SISTEMA PARADO"
Command7.BackColor = vbYellow
Command9.Caption = "SERVO APAGADO"
Command9.BackColor = vbYellow
Command11.Caption = "SIN SEÑAL"
Command11.BackColor = vbYellow

Text4.Text = t_alim
Text5.Text = t_acid
Text6.Text = "OFF"
Text7.Text = "OFF"
Text8.Text = "OFF"
```

```
Text9.Text = "OFF"
Text10.Text = UpDown1.Max
Text11.Text = UpDown2.Min
Text12.Text = UpDown3.Max
Text13.Text = UpDown4.Min
Text14.Text = UpDown5.Min
End Sub

Private Sub Form_Resize()
With Image1
    .Left = 0
    .Top = 0
    .Width = Me.Width
    .Height = Me.Height
End With
End Sub

'(Subformulario definido en el anterior apartado)
Private Sub MSComm1_OnComm()
If MSComm1.CommEvent = comEvReceive Then
    sData = MSComm1.Input
    longitud = Len(sData)
    primer_char = Mid$(sData, 1, 1)
    segundo_char = Mid$(sData, 2, 1)
    mensaje = Mid$(sData, 1, 10)
    If mensaje = "COMCORRECT" Then
        control = True
    End If
    If primer_char = "E" Then
        error = True
        Call reiniciar
        MSComm1.PortOpen = False
        If segundo_char = "S" Then
```

```
        MsgBox "Error: no se detecta la tarjeta SD", vbOKOnly + vbCritical, "Error de inicialización"
            ElseIf segundo_char = "H" Then
                MsgBox "Error: no se detecta el archivo .htm", vbOKOnly + vbCritical, "Error de inicialización"
      End If
   Else
      error = False
   End If
   If primer_char = "S" Then
      If segundo_char = "A" Then
         t_alim = #12:00:00 AM#
         ElseIf segundo_char = "P" Then
            t_acid = #12:00:00 AM#
      End If
   End If
   If (primer_char <> "T") And (primer_char <> "E") And (primer_char <> “S”) Then
      lect_acid = Mid$(sData, 1, 4)
      lect_temp = Mid$(sData, 5, 5)
      lect_nivel = Mid$(sData, 10, 1)
      temp = CSng(Val(lect_temp))
      acid = CSng(Val(lect_acid))
      nivel = CBool(Val(lect_nivel))
      If test = False Then
         Text1.Text = temp
         Text2.Text = acid
         Text3.Text = sData
         SFCutaway1.Level = temp
         SFCutaway2.Level = acid

         Call toma_datos
```

```
'Control del nivel del acuario
If nivel = False Then
   Command11.Caption = "NIVEL CORRECTO"
   Command11.BackColor = vbGreen
   s_nivel = "CORRECTO"
Else
   Command11.Caption = "NIVEL INCORRECTO"
   Command11.BackColor = vbRed
   s_nivel = "INCORRECTO"
End If

'Control de la temperatura
If ((temp >= min_temp) And (temp <= (min_temp + 1.5))) Or ((temp <= max_temp) And (temp >= (max_temp - 1.5))) Then
   SFStandard17.FillColor = vbYellow
   ElseIf (temp < min_temp) Or (temp > max_temp) Then
      SFStandard17.FillColor = vbRed
      Timer15.Enabled = True
   Else
      SFStandard17.FillColor = vbGreen
End If

'Control de la acidez
If ((acid >= min_acid) And (acid <= (min_acid + 1))) Or ((acid <= max_acid) And (acid >= (max_acid - 1))) Then
   SFStandard18.FillColor = vbYellow
   ElseIf (acid < min_acid) Or (acid > max_acid) Then
      SFStandard18.FillColor = vbRed
      Timer16.Enabled = True
   Else
      SFStandard18.FillColor = vbGreen
End If
```

```
        'Dibujado de las gráficas
        YT = temp
         YA = acid
         If (IsNumeric(YT)) And (IsNumeric(YA)) Then
            If Xt_1 = (48 / toma) Then
               Call Redibujar_temp
               Call Redibujar_acid
               Xt_1 = 0
               i = 1
               v_temp(0) = v_temp(CInt(48 / toma))
               v_acid(0) = v_acid(CInt(48 / toma))
            End If
            Xt = Xt_1 + toma
            Form5.Picture1.Line (Xt_1, YT_1)-(Xt, YT), vbRed, BF
            Form5.Picture2.Line (Xt_1, YA_1)-(Xt, YA), vbBlue, BF
            Xt_1 = Xt
            YT_1 = YT
            YA_1 = YA
            v_temp(i) = temp
            v_acid(i) = acid
            i = i + 1
         End If
      End If
   End If
   If primer_char = "T" Then
      If segundo_char = "P" Then
         test_puerto = True
      Else
         str_termo = Mid$(sData, 4, 2)
         str_filtro = Mid$(sData, 6, 2)
         str_lampara = Mid$(sData, 8, 2)
         str_servo = Mid$(sData, 10, 2)
         If str_termo = "YT" Then
```

```
            test_termo = True
            ElseIf str_termo = "NT" Then
                test_termo = False
        End If
        If str_filtro = "YF" Then
            test_filtro = True
            ElseIf str_filtro = "NF" Then
                test_filtro = False
        End If
        If str_lampara = "YL" Then
            test_lampara = True
            ElseIf str_lampara = "NL" Then
                test_lampara = False
        End If
        If str_servo = "YS" Then
            test_servo = True
            ElseIf str_servo = "NS" Then
                test_servo = False
        End If
        Call result_test(test_puerto, test_lampara, test_filtro, test_termo,
test_servo)
      End If
   End If
End If
End Sub

'Elección de registro para datos
Private Sub SelecData_Click()
CommonDialog1.InitDir = "C:\Users\Néstor\Documents"
CommonDialog1.DialogTitle = "Abrir archivo para guardar muestras"
CommonDialog1.Filter = "Archivos de texto|*.txt|Documentos de
Word|*.docx|Todos los Archivos|*.*"
CommonDialog1.ShowOpen
```

```
If CommonDialog1.FileName <> "" Then
   If CommonDialog1.FileName = CommonDialog2.FileName Then
      MsgBox "¡No se puede elegir el mismo fichero para datos y eventos!", vbOKOnly + vbExclamation, "Error de selección de fichero"
   ElseIf registro_data = False Then
         Open CommonDialog1.FileName For Append As #1
         Print #1, "--------------------------------------------------------------------------------"
         Print #1, "Registro de medidas tomadas para control de alarmas"
         Print #1, "Fecha de elección de registro: " & Date & " " & Time
         Print #1, vbCrLf
         AbrirData.Enabled = True
         registro_data = True
   End If
End If
End Sub

'Elección de registro para eventos
Private Sub SelecEvent_Click()
CommonDialog2.InitDir = "C:\Users\Néstor\Documents"
CommonDialog2.DialogTitle = "Abrir archivo para guardar eventos"
CommonDialog2.Filter = "Archivos de texto|*.txt|Documentos de Word|*.docx|Todos los Archivos|*.*"
CommonDialog2.ShowOpen
If CommonDialog2.FileName <> "" Then
   If CommonDialog2.FileName = CommonDialog1.FileName Then
      MsgBox "¡No se puede elegir el mismo fichero para datos y eventos!", vbOKOnly + vbExclamation, "Error de selección de fichero"
      ElseIf registro_event = False Then
         Open CommonDialog2.FileName For Append As #2
         Print #2, "--------------------------------------------------------------------------------"
         Print #2, "Registro de eventos ocurridos para control de alarmas"
```

```
            Print #2, "Fecha de elección de registro: " & Date & " " & Time
            Print #2, vbCrLf
            AbrirEvent.Enabled = True
            registro_event = True
        End If
    End If
End Sub

'Control de la variable de acidez para los objetos UpDown correspondientes
Private Sub UpDown3_UpClick()
If Text12.Text >= 14 Then
    Text12.Text = 14
Else
    Text12.Text = Text12.Text + 0.1
End If
End Sub

Private Sub UpDown3_DownClick()
If Text12.Text <= 0 Then
    Text12.Text = 0
Else
    Text12.Text = Text12.Text - 0.1
End If
End Sub

Private Sub UpDown4_UpClick()
If Text13.Text >= 14 Then
    Text13.Text = 14
Else
    Text13.Text = Text13.Text + 0.1
End If
End Sub
```

```
Private Sub UpDown4_DownClick()
If Text13.Text <= 0 Then
   Text13.Text = 0
Else
   Text13.Text = Text13.Text - 0.1
End If
End Sub

Private Sub Validar_Click()
max_temp = CInt(Val(Text10.Text))
min_temp = CInt(Val(Text11.Text))
max_acid = CInt(Val(Text12.Text))
min_acid = CInt(Val(Text13.Text))
periodo = CInt(Val(Text14.Text))
t_lampara_on = DTPicker2.Value
t_lampara_off = DTPicker3.Value
t_aux = DTPicker1.Value
toma = periodo / 60

If (periodo < 30) Then
   lim_t_temp = 60
   lim_t_acid = 60
   ElseIf (periodo >= 30) And (periodo < 45) Then
      lim_t_temp = 90
      lim_t_acid = 90
      ElseIf (periodo >= 45) And (periodo <= 60) Then
         lim_t_temp = 130
         lim_t_acid = 130
End If

Validar.Enabled = False
Resetear.Enabled = True
UpDown1.Enabled = False
```

```
UpDown2.Enabled = False
UpDown3.Enabled = False
UpDown4.Enabled = False
UpDown5.Enabled = False

DTPicker1.Enabled = False
DTPicker2.Enabled = False
DTPicker3.Enabled = False

If (max_temp <= min_temp) Or (max_acid <= min_acid) Then
   MsgBox "¡Los valores máximos y mínimos escogidos no son correctos!",
vbOKOnly + vbCritical, "Error en la personalización"
   ElseIf t_lampara_on = t_lampara_off Then
      MsgBox "¡Los tiempos de encendido y apagado deben ser distintos!",
vbOKOnly + vbCritical, "Error en la personalización"
   Else
      valid = True
End If
End Sub

Private Sub Resetear_Click()
DTPicker1.Value = #12:00:00 AM#
DTPicker2.Value = #12:00:00 AM#
DTPicker3.Value = #12:00:00 AM#
Text10.Text = UpDown1.Max
Text11.Text = UpDown2.Min
Text12.Text = UpDown3.Max
Text13.Text = UpDown4.Min
Text14.Text = UpDown5.Min

Validar.Enabled = True
Resetear.Enabled = False
UpDown1.Enabled = True
```

```
UpDown2.Enabled = True
UpDown3.Enabled = True
UpDown4.Enabled = True
UpDown5.Enabled = True
DTPicker1.Enabled = True
DTPicker2.Enabled = True
DTPicker3.Enabled = True

lim_t_temp = 0
lim_t_acid = 0

If valid = True Then
   valid = False
End If
End Sub

'Botón de puesta en marcha
Private Sub SFStandard2_Click()
If registro_data = False Then
   MsgBox "¡No ha seleccionado un fichero como registro de datos!", vbOKOnly + vbCritical, "Error en la configuración"
   ElseIf registro_event = False Then
      MsgBox "¡No ha seleccionado un fichero como registro de eventos!", vbOKOnly + vbCritical, "Error en la configuración"
      ElseIf valid = False Then
         MsgBox "¡No ha validado las variables de personalización!", vbOKOnly + vbCritical, "Error en la configuración"
      Else
         SFStandard2.Visible = False
         SFStandard3.Visible = True
         Resetear.Enabled = False
         MSComm1.PortOpen = True
         retardo = 2
```

```
Call espera(retardo)
If error = False Then
   orden = 9
   MSComm1.Output = Chr(orden)
   MSComm1.Output = Chr(periodo)
   MSComm1.Output = Chr(modo_arduino)
   orden = 1
   MSComm1.Output = Chr(orden)
   control = True
   h_alim = Time + t_aux
   Print #2, "Se activó el botón de puesta en marcha a las " & Time
   Print #2, vbCrLf

   Timer4.Enabled = True
   Timer5.Enabled = True
   Timer6.Enabled = True
   Timer8.Enabled = True
   Timer10.Enabled = True
   Timer12.Enabled = True
   Timer13.Enabled = True

   AbrirData.Enabled = True
   AbrirEvent.Enabled = True
   AbrirTend.Enabled = True
   SelecData.Enabled = False
   SelecEvent.Enabled = False
   SFStandard4.Visible = True
   SFStandard5.Visible = False
   SFStandard4.FillColorMode = Original
   SFStandard6.FillColorMode = Hollow
   SFStandard8.FillColorMode = Hollow
   SFStandard10.FillColorMode = Original
   Text6.Text = "ON"
```

```
            Text8.Text = "ON"
            Command7.Caption = "SISTEMA FUNCIONANDO"
            Command7.BackColor = vbGreen
        End If
End If
End Sub

'Botón de paro
Private Sub SFStandard4_Click()
MSComm1.PortOpen = False
control = False
Print #2, "Se activó el botón de paro a las " & Time
Print #2, vbCrLf

Timer4.Enabled = False
Timer5.Enabled = False
Timer6.Enabled = False
Timer7.Enabled = False
Timer8.Enabled = False
Timer10.Enabled = False
Timer12.Enabled = False
Timer13.Enabled = False

Resetear.Enabled = True
SFStandard2.Visible = True
SFStandard3.Visible = False
SFStandard4.Visible = False
SFStandard5.Visible = True
SFStandard6.Visible = True
SFStandard7.Visible = False
SFStandard8.Visible = True
SFStandard9.Visible = False
SFStandard10.Visible = True
```

```
SFStandard11.Visible = False
SFStandard2.FillColorMode = Original
SFStandard6.FillColorMode = Hollow
SFStandard8.FillColorMode = Original
SFStandard10.FillColorMode = Hollow
Text6.Text = "OFF"
Command7.Caption = "SISTEMA PARADO"
Command7.BackColor = vbYellow
Command9.Caption = "SERVO APAGADO"
Command9.BackColor = vbYellow
Command11.Caption = "SIN SEÑAL"
Command11.BackColor = vbYellow
End Sub

'Botón de rearme
Private Sub SFStandard6_Click()
Print #2, "Se activó el botón de rearme a las " & Time
Print #2, vbCrLf

Form6.Height = 2000
Form6.Label1.Alignment = 0
Form6.ProgressBar1.Visible = True
Form6.Hide
Image1.Picture = ImageList1.ListImages(1).Picture
Command7.Caption = "SISTEMA EN REARME"
Command7.BackColor = vbBlue
SFStandard6.Visible = False
SFStandard7.Visible = True
SFStandard10.Visible = True
SFStandard11.Visible = False
SFStandard10.FillColorMode = Original

Timer9.Enabled = True
```

```
End Sub

'Botón de test
Private Sub SFStandard8_Click()
If registro_data = False Then
   MsgBox "¡No ha seleccionado un fichero como registro de datos!", vbOKOnly + vbCritical, "Error en la configuración"
   ElseIf registro_event = False Then
      MsgBox "¡No ha seleccionado un fichero como registro de eventos!", vbOKOnly + vbCritical, "Error en la configuración"
   Else
      Print #2, "Se activó el botón de testeo a las " & Time
      Print #2, vbCrLf

      SFStandard2.Visible = True
      SFStandard3.Visible = False
      SFStandard4.Visible = True
      SFStandard5.Visible = False
      SFStandard6.Visible = True
      SFStandard7.Visible = False
      SFStandard8.Visible = False
      SFStandard9.Visible = True
      SFStandard2.FillColorMode = Hollow
      SFStandard4.FillColorMode = Hollow
      SFStandard6.FillColorMode = Hollow
      SFStandard10.FillColorMode = Hollow
      Text6.Text = "OFF"
      Command7.Caption = "TEST EN MARCHA"
      Command7.BackColor = vbYellow

      Form6.Height = 2000
      Form6.Label1.Alignment = 0
      Form6.ProgressBar1.Visible = True
```

```
        Form6.ProgressBar1.Value = 0
        Form6.Label1.Caption = "Testeando comunicación..."
        Form6.Show

        MSComm1.PortOpen = True
        retardo = 2
        Call Espera(retardo)

        orden = 8
        MSComm1.Output = Chr(orden)

        Form6.ProgressBar1.Value = 25
        Form6.Label1.Caption = "Testeando lámpara..."
        Call Espera(retardo)
        Form6.ProgressBar1.Value = 50
        Form6.Label1.Caption = "Testeando filtro..."
        Call Espera(retardo)
        Form6.ProgressBar1.Value = 75
        Form6.Label1.Caption = "Testeando termocalentador..."
        Call Espera(retardo)
        Form6.ProgressBar1.Value = 100
        Form6.Label1.Caption = "Testeando servo..."
        Call Espera(retardo)
        Form6.Hide
End If
End Sub

'Botón de parada de emergencia
Private Sub SFStandard10_Click()
Print #2, "Se activó la seta de EMERGENCIA a las " & Time
Print #2, vbCrLf
Print #1, "Valores de magnitudes en el momento de la parada: " & Time
Print #1, "Temperatura", "Acidez", "Nivel", "Servo", "Filtro", "Lámpara",
```

```
"Termocalentador"
Print #1, Format(temp, "#00.00"), Format(acid, "##00.0"), s_nivel, s_servo, s_filtro,
s_lampara, s_termo
Call Emergencia
End Sub

'Temporizador de espera de la alimentación
Private Sub Timer1_Timer()
cont_alim = cont_alim + 1
If cont_alim >= 2 Then
    cont_alim = 0
    s_servo = "OFF"
    Command9.Caption = "SERVO APAGADO"
    Command9.BackColor = vbYellow
    Timer1.Enabled = False
End If
End Sub

'Reloj para fecha y hora del sistema
Private Sub Timer2_Timer()
Label20.Caption = Date
Label21.Caption = Time
End Sub

'Reloj para control de funcionamiento de la lámpara
Private Sub Timer4_Timer()
If (Time >= t_lampara_on) And (Time < t_lampara_off) Then
    If lampara = False Then
        orden = 0
        MSComm1.Output = Chr(orden)
        lampara = True
        s_lampara = "ON"
        Print #2, "Se activó la lámpara a las " & Time
```

```
        Print #2, vbCrLf
        Text7.Text = "ON"
    End If
    ElseIf (Time >= t_lampara_off) Then
        If lampara = True Then
            orden = 3
            MSComm1.Output = Chr(orden)
            lampara = False
            s_lampara = "OFF"
            Print #2, "Se desactivó la lámpara a las " & Time
            Print #2, vbCrLf
            Text7.Text = "OFF"
        End If
End If
End Sub

'Reloj para control de funcionamiento del termocalentador
Private Sub Timer5_Timer()
If temp < min_temp Then
    orden = 2
    MSComm1.Output = Chr(orden)
    Print #2, "Se activó el termocalentador a las " & Time
    Print #2, vbCrLf
    Timer7.Enabled = True
    Timer5.Enabled = False
End If
End Sub

'Reloj para control de funcionamiento de la alimentación
Private Sub Timer6_Timer()
If Time >= h_alim Then
    orden = 6
    MSComm1.Output = Chr(orden)
```

```
    s_servo = "ON"
    Print #2, "Se activó el servo para alimentación a las " & Time
    Print #2, vbCrLf
    Command9.Caption = "SERVO ACTIVO"
    Command9.BackColor = vbGreen
    alim = True
    h_alim = Time + t_aux
    Timer1.Enabled = True
End If
End Sub

'Reloj para control de funcionamiento del termocalentador
Private Sub Timer7_Timer()
If temp > max_temp Then
    orden = 5
    MSComm1.Output = Chr(orden)
    Print #2, "Se desactivó el termocalentador a las " & Time
    Print #2, vbCrLf
    Timer5.Enabled = True
    Timer7.Enabled = False
End If
End Sub

'Reloj para control de ajuste de acidez
Private Sub Timer8_Timer()
cont_ph = cont_ph + 1
If (cont_ph >= lim_t_acid) Then
    If (acid < min_acid) Or (acid > max_acid) Then
        If num_ajustes < 8 Then
            orden = 7
            MSComm1.Output = Chr(orden)
            Print #2, "Se activó el servo para ajustar la acidez a las " & Time
            Print #2, vbCrLf
```

```
            Command9.Caption = "SERVO ACTIVO"
            Command9.BackColor = vbGreen
            num_ajustes = num_ajustes + 1
            cont_ph = 0
            espera_ajust = True
            Timer3.Enabled = True
        Else
            Form6.Height = 1500
            Form6.Label1.Alignment = 2
            Form6.ProgressBar1.Visible = False
            Form6.Label1.Caption = "¡LA ACIDEZ ESTÁ FUERA DE LOS LÍMITES!"
            Form6.Show

            Call emergencia

            cont_ph = 0
            num_ajustes = 0
        End If
    End If
End If
End Sub

'Temporizador empleado para la ejecución de la función de rearme
Private Sub Timer9_Timer()
cont = cont + 1
If cont < 2 Then
Form6.Show
Form6.Label1.Caption = "Rearmando el sistema..."
Form6.ProgressBar1.Value = 0
End If
If cont = 2 Then
    Form6.ProgressBar1.Value = 25
    Form6.Label1.Caption = "Reiniciando los actuadores..."
```

```
MSComm1.PortOpen = True
orden = 10
MSComm1.Output = Chr(orden)
MSComm1.PortOpen = False
ElseIf cont = 4 Then
    Form6.ProgressBar1.Value = 50
    Form6.Label1.Caption = "Reiniciando variables..."
    s_nivel = "SIN SEÑAL"
    s_servo = "OFF"
    s_filtro = "OFF"
    s_lampara = "OFF"
    s_termo = "OFF"
    Xt_1 = 0
    YT_1 = 0
    YA_1 = 0
    temp = 0
    acid = 0
    i = 1
    v_temp(0) = 0
    v_acid(0) = 0
    cont_alim = 0
    cont_acid = 0
    cont_envio = 0
    lim_t_temp = 0
    lim_t_acid = 0
    error = False
    nivel = False
    lampara = False
    termo = False
    filtro = False
    valid = False
    control = False
    test_puerto = False
```

```
test_lampara = False
test_filtro = False
test_termo = False
test_servo = False
alim = False
ajust = False
ElseIf cont = 6 Then
   Form6.ProgressBar1.Value = 75
   Form6.Label1.Caption = "Reiniciando apariencia..."
   SFStandard2.Visible = True
   SFStandard4.Visible = True
   SFStandard6.Visible = True
   SFStandard8.Visible = True
   SFStandard10.Visible = True
   SFStandard3.Visible = False
   SFStandard5.Visible = False
   SFStandard7.Visible = False
   SFStandard9.Visible = False
   SFStandard11.Visible = False
   SFStandard4.FillColorMode = Hollow
   SFStandard6.FillColorMode = Hollow
   SFStandard8.FillColorMode = Original
   SFStandard10.FillColorMode = Hollow
   SFStandard17.FillColor = vbGreen
   SFStandard18.FillColor = vbGreen
   SFCutaway1.Level = 20
   SFCutaway2.Level = 7
   Command7.Caption = "SISTEMA REARMADO"
   Command9.Caption = "SERVO APAGADO"
   Command9.BackColor = vbYellow
   Command11.Caption = "SIN SEÑAL"
   Command11.BackColor = vbYellow
```

```
        Text1.Text = ""
        Text2.Text = ""
        Text4.Text = #12:00:00 AM#
        Text5.Text = #12:00:00 AM#
        Text6.Text = "OFF"
        Text7.Text = "OFF"
        Text8.Text = "OFF"
        Text9.Text = "OFF"
        Text10.Text = UpDown1.Max
        Text11.Text = UpDown2.Min
        Text12.Text = UpDown3.Max
        Text13.Text = UpDown4.Min
        Text14.Text = UpDown5.Min
        DTPicker1.Value = #12:00:00 AM#
        DTPicker2.Value = #12:00:00 AM#
        DTPicker3.Value = #12:00:00 AM#

        Call Redibujar_temp
        Call Redibujar_acid

        Form6.ProgressBar1.Value = 100
        ElseIf cont = 7 Then
           Form6.Hide
           cont = 0
           Timer9.Enabled = False
End If
End Sub

'Timer para el movimiento del filtro
Private Sub Timer10_Timer()
If SFStandard19.Flip = 0 Then
   SFStandard19.Flip = 1
   ElseIf SFStandard19.Flip = 1 Then
```

```
        SFStandard19.Flip = 0
End If
End Sub

'Timer para situación de emergencia
Private Sub Timer11_Timer()
Beep
If Image1.Picture = ImageList1.ListImages(1).Picture Then
    Image1.Picture = LoadPicture("")
    Form4.BackColor = vbRed
    ElseIf Form4.BackColor = vbRed Then
        Image1.Picture = ImageList1.ListImages(1).Picture
End If
End Sub

'Timer para control de tiempo desde que se realizó la última alimentación
Private Sub Timer12_Timer()
t_alim = t_alim + #12:00:01 AM#
Text4.Text = t_alim
If alim = True Then
    t_alim = #12:00:00 AM#
    alim = False
End If
End Sub

'Timer para control de tiempo desde que se realizó el último ajuste de acidez
Private Sub Timer13_Timer()
t_acid = t_acid + #12:00:01 AM#
Text5.Text = t_acid
If ajust = True Then
    t_acid = #12:00:00 AM#
    ajust = False
End If
```

```
End Sub

Reloj de espera para comprobación de la temperatura en el agua
Private Sub Timer14_Timer()          '
cont_temp = cont_temp + 1
If (cont_temp >= lim_t_temp) Then
   If (temp < min_temp) Or (temp > max_temp) Then
      Form6.Height = 1500
      Form6.Label1.Alignment = 2
      Form6.ProgressBar1.Visible = False
      Form6.Label1.Caption = "¡LA TEMPERATURA ESTÁ FUERA DE LOS
LÍMITES!"
      Form6.Show
      Call emergencia
      cont_temp = 0
   Else
      cont_temp = 0
      Timer14.Enabled = False
   End If
End If
End Sub

'Función preparada para ejecutar una situación de emergencia
Function Emergencia()
MSComm1.PortOpen = False
Timer1.Enabled = False
Timer4.Enabled = False
Timer5.Enabled = False
Timer6.Enabled = False
Timer7.Enabled = False
Timer8.Enabled = False
Timer10.Enabled = False
Timer12.Enabled = False
```

```
Timer13.Enabled = False
Timer14.Enabled = False

SFStandard2.Visible = True
SFStandard3.Visible = False
SFStandard4.Visible = True
SFStandard5.Visible = False
SFStandard6.Visible = True
SFStandard7.Visible = False
SFStandard8.Visible = True
SFStandard9.Visible = False
SFStandard10.Visible = False
SFStandard11.Visible = True
SFStandard2.FillColorMode = Hollow
SFStandard4.FillColorMode = Hollow
SFStandard6.FillColorMode = Original
SFStandard8.FillColorMode = Hollow

Text6.Text = "OFF"
Command7.Caption = "PARADO POR EMERGENCIA"
Command7.BackColor = vbRed
Command9.Caption = "PARADO POR EMERGENCIA"
Command9.BackColor = vbRed
Command11.Caption = "PARADO POR EMERGENCIA"
Command11.BackColor = vbRed
Timer11.Enabled = True
End Function

'Función para la obtención de resultados del proceso de testeo
Function result_test(ByVal a As Boolean, ByVal b As Boolean, ByVal c As Boolean, ByVal d As Boolean, ByVal e As Boolean)
If a = True Then
   c_puerto = "correcta"
```

```
Else
    c_puerto = "incorrecta"
End If
If b = True Then
    c_lampara = "correcto"
Else
    c_lampara = "incorrecto"
End If
If c = True Then
    c_filtro = "correcto"
Else
    c_filtro = "incorrecto"
End If
If d = True Then
    c_termo = "correcto"
Else
    c_termo = "incorrecto"
End If
If e = True Then
    c_servo = "correcto"
Else
    c_servo = "incorrecto"
End If

MsgBox "Comunicación serie: " & c_puerto & Chr(13) & _
" Funcionamiento de lámpara: " & c_lampara & Chr(13) & _
" Funcionamiento de filtro: " & c_filtro & Chr(13) & _
" Funcionamiento de termoc.: " & c_termo & Chr(13) & _
" Funcionamiento de servo: " & c_servo, vbOKOnly, "Resultados del testeo"

Print #2, "Resultados del test:" & Chr(13) & _
"Comunicación serie: " & c_puerto & Chr(13) & _
"Funcionamiento de lámpara: " & c_lampara & Chr(13) & _
```

```
"Funcionamiento de filtro: " & c_filtro & Chr(13) & _
"Funcionamiento de termoc.: " & c_termo & Chr(13) & _
"Funcionamiento de servo: " & c_servo
Print #2, vbCrLf

a = False
b = False
c = False
d = False
e = False

SFStandard8.Visible = True
SFStandard9.Visible = False
SFStandard2.FillColorMode = Original
Command7.Caption = "SISTEMA PARADO"
Command7.BackColor = vbYellow

MSComm1.PortOpen = False
End Function

'Función para el reinicio del sistema
Function reiniciar()
s_nivel = "SIN SEÑAL"
s_servo = "OFF"
s_filtro = "OFF"
s_lampara = "OFF"
s_termo = "OFF"
Xt_1 = 0
YT_1 = 0
YA_1 = 0
temp = 0
acid = 0
cont = 0
```

```
i = 1
v_temp(0) = 0
v_acid(0) = 0
cont_alim = 0
cont_acid = 0
cont_temp = 0
cont_ph = 0
cont_envio = 0
lim_t_temp = 0
lim_t_acid = 0
error = False
nivel = False
lampara = False
termo = False
filtro = False
valid = False
registro_data = False
registro_event = False
control = False
test_puerto = False
test_lampara = False
test_filtro = False
test_termo = False
test_servo = False
alim = False
ajust = False
tend = False
DTPicker1.Value = #12:00:00 AM#
DTPicker2.Value = #12:00:00 AM#
DTPicker3.Value = #12:00:00 AM#
t_alim = #12:00:00 AM#
t_acid = #12:00:00 AM#
h_alim = #12:00:00 AM#
```

SFStandard2.Visible = True
SFStandard4.Visible = True
SFStandard6.Visible = True
SFStandard8.Visible = True
SFStandard10.Visible = True
SFStandard3.Visible = False
SFStandard5.Visible = False
SFStandard7.Visible = False
SFStandard9.Visible = False
SFStandard11.Visible = False
AbrirData.Enabled = False
AbrirEvent.Enabled = False
AbrirTend.Enabled = False
SelecData.Enabled = True
SelecEvent.Enabled = True
SFStandard4.FillColorMode = Hollow
SFStandard6.FillColorMode = Hollow
SFStandard8.FillColorMode = Original
SFStandard10.FillColorMode = Hollow
SFStandard17.FillColor = vbGreen
SFStandard18.FillColor = vbGreen
SFCutaway1.Level = 20
SFCutaway2.Level = 7
Command7.Caption = "SISTEMA PARADO"
Command7.BackColor = vbYellow
Command9.Caption = "SERVO APAGADO"
Command9.BackColor = vbYellow
Command11.Caption = "SIN SEÑAL"
Command11.BackColor = vbYellow
Validar.Enabled = True
Resetear.Enabled = False

Text4.Text = t_alim

```
Text5.Text = t_acid
Text6.Text = "OFF"
Text7.Text = "OFF"
Text8.Text = "OFF"
Text9.Text = "OFF"
Text10.Text = UpDown1.Max
Text11.Text = UpDown2.Min
Text12.Text = UpDown3.Max
Text13.Text = UpDown4.Min
Text14.Text = UpDown5.Min
End Function

'Función encargada de ejecutar operaciones para cerrar el SCADA
Function terminar_control()
If MSComm1.PortOpen = False Then
   MSComm1.PortOpen = True
End If
orden = 10
MSComm1.Output = Chr(orden)
MSComm1.PortOpen = False
If Timer6.Enabled = True Then
   Timer6.Enabled = False
End If
Close #1
Close #2
End
End Function

'Botón de menú encargado de abrir el fichero de datos
Private Sub AbrirData_Click()
ret_data = ShellExecute(Me.HWnd, "Open", CommonDialog1.FileName, "", "", 1)
End Sub
```

```
'Botón de menú encargado de abrir el fichero de eventos
Private Sub AbrirEvent_Click()
ret_event = ShellExecute(Me.HWnd, "Open", CommonDialog2.FileName, "", "", 1)
End Sub

'Botón de menú encargado de abrir el formulario de tendencias
Private Sub AbrirTend_Click()
Form5.Show
End Sub

'Botón de menú encargado de abrir la ventana de información
Private Sub Acercade_Click()
Form2.Show
End Sub

'Permite volver a la pantalla inicial del sistema (Formulario 1)
Private Sub Volver_Click()
If control = True Then
   MsgBox "¡Debe parar antes el control!", vbOKOnly + vbCritical, "Error en la configuración"
Else
   pregunta = MsgBox("¿Está seguro de que quiere volver?", vbYesNo + vbInformation, "Salida del programa")
   If pregunta = 6 Then
      If MSComm1.PortOpen = False Then
         MSComm1.PortOpen = True
      End If
      orden = 10
      MSComm1.Output = Chr(orden)
      MSComm1.PortOpen = False
      If Timer6.Enabled = True Then
         Timer6.Enabled = False
      End If
```

```
        Close #1
        Close #2

        Call reiniciar

        Form1.Show
        Form4.Hide
    End If
End If
End Sub

'Salida del SCADA y cierre del programa mediante el control "Salir"
Private Sub Salir_Click()
If control = True Then
    MsgBox "¡Debe parar antes el control!", vbOKOnly + vbCritical, "Error en la configuración"
Else
    pregunta = MsgBox("¿Está seguro de que quiere salir?", vbYesNo + vbInformation, "Salida del programa")
    If pregunta = 6 Then
        Call terminar_control
    End If
End If
End Sub

'Salida del SCADA y cierre del programa por control de Windows
Private Sub Form_Unload(Cancel As Integer)
If control = True Then
    MsgBox "¡Debe parar antes el control!", vbOKOnly + vbCritical, "Error en la configuración"
Else
    pregunta = MsgBox("¿Está seguro de que quiere salir?", vbYesNo + vbInformation, "Salida del programa")
```

```
    If pregunta = 6 Then
        Call terminar_control
    End If
End If
End Sub
```

2.5. Formulario 5 (Tend.frm)

```
Private Sub Form_Load()
Form5.Height = 11500
Form5.Width = 14500

'Tendencia para la temperatura
Picture1.DrawWidth = 1
Picture1.BackColor = vbBlack
Picture1.Scale (0, 30)-(48, -5)
Picture1.Line (0, 0)-(48, 0), vbGreen
Picture1.Line (0, 0)-(0, 30), vbGreen
Picture1.Font.Size = 8

'Tendencia para la acidez
Picture2.DrawWidth = 1
Picture2.BackColor = vbBlack
Picture2.Scale (0, 14)-(48, -3)
Picture2.Line (0, 0)-(48, 0), vbGreen
Picture2.Line (0, 0)-(0, 14), vbGreen
Picture2.Font.Size = 8

Call Dibujar
End Sub

'Cierre de la ventana de tendencias
Private Sub Form_Unload(Cancel As Integer)
tend = True
Form5.Hide
End Sub
```

2.6. Formulario 6 (Adapt.frm)

```
Private Sub Form_Load()
Form6.Height = 2000
Form6.Width = 3000
End Sub
```

2.7. Módulo 1 (Funciones.bas)

```
Option Explicit
'Variables para creación de ejes (temperatura)
Dim X1 As Single, Y1 As Single
'Variables para creación de ejes (acidez)
Dim X2 As Single, Y2 As Single

'Dibuja inicialmenta los ejes de las gráficas de temperatura y acidez
Function Dibujar()
'Marcas de ejes X e Y
For X1 = 0 To 48 Step 4
   Form5.Picture1.Line (X1, 1)-(X1, -1), vbGreen
   Form5.Picture1.Print X1
Next X1
For Y1 = 0 To 30 Step 5
   Form5.Picture1.Line (1, Y1)-(-1, Y1), vbGreen
   Form5.Picture1.PSet (5, Y1), vbGreen
   If (Y1 <> 0) Then
      Form5.Picture1.Print Y1
   End If
Next Y1
'Rejillas en X e Y
For X1 = 0 To 48 Step 4
   Form5.Picture1.Line (X1, 30)-(X1, -1), vbGreen
   Form5.Picture1.Print X1
Next X1
For Y1 = 0 To 30 Step 5
   Form5.Picture1.Line (48, Y1)-(0, Y1), vbGreen
   Form5.Picture1.PSet (5, Y1), vbGreen
   If (Y1 <> 0) Then
      Form5.Picture1.Print Y1
   End If
```

```
Next Y1
'Marcas de ejes X e Y
For X2 = 0 To 48 Step 4
   Form5.Picture2.Line (X2, 1)-(X2, -1), vbGreen
   Form5.Picture2.Print X2
Next X2
For Y2 = 0 To 14 Step 3
   Form5.Picture2.Line (1, Y2)-(-1, Y2), vbGreen
   Form5.Picture2.PSet (5, Y2), vbGreen
   If (Y2 <> 0) Then
      Form5.Picture2.Print Y2
   End If
Next Y2
'Rejillas en X e Y
For X2 = 0 To 48 Step 4
   Form5.Picture2.Line (X2, 14)-(X2, -1), vbGreen
   Form5.Picture2.Print X2
Next X2
For Y2 = 0 To 14 Step 3
   Form5.Picture2.Line (48, Y2)-(0, Y2), vbGreen
   Form5.Picture2.PSet (5, Y2), vbGreen
   If (Y2 <> 0) Then
      Form5.Picture2.Print Y2
   End If
Next Y2
End Function

'Redibuja la gráfica de temperatura a partir de los valores almacenados
Function Redibujar_temp()
Form5.Picture1.Picture = LoadPicture("")
'Marcas de ejes X e Y
For X1 = 0 To 48 Step 4
   Form5.Picture1.Line (X1, 1)-(X1, -1), vbGreen
```

```
    Form5.Picture1.Print X1
Next X1
For Y1 = 0 To 30 Step 5
    Form5.Picture1.Line (1, Y1)-(-1, Y1), vbGreen
    Form5.Picture1.PSet (5, Y1), vbGreen
    If (Y1 <> 0) Then
        Form5.Picture1.Print Y1
    End If
Next Y1
'Rejillas en X e Y
For X1 = 0 To 48 Step 4
    Form5.Picture1.Line (X1, 30)-(X1, -1), vbGreen
    Form5.Picture1.Print X1
Next X1
For Y1 = 0 To 30 Step 5
    Form5.Picture1.Line (48, Y1)-(0, Y1), vbGreen
    Form5.Picture1.PSet (5, Y1), vbGreen
    If (Y1 <> 0) Then
        Form5.Picture1.Print Y1
    End If
Next Y1
End Function

'Redibuja la gráfica de acidez a partir de los valores almacenados
Function Redibujar_acid()
Form5.Picture2.Picture = LoadPicture("")
'Marcas de ejes X e Y
For X2 = 0 To 48 Step 4
    Form5.Picture2.Line (X2, 1)-(X2, -1), vbGreen
    Form5.Picture2.Print X2
Next X2
For Y2 = 0 To 14 Step 3
    Form5.Picture2.Line (1, Y2)-(-1, Y2), vbGreen
```

```
    Form5.Picture2.PSet (5, Y2), vbGreen
    If (Y2 <> 0) Then
        Form5.Picture2.Print Y2
    End If
Next Y2
'Rejillas en X e Y
For X2 = 0 To 48 Step 4
    Form5.Picture2.Line (X2, 14)-(X2, -1), vbGreen
    Form5.Picture2.Print X2
Next X2
For Y2 = 0 To 14 Step 3
    Form5.Picture2.Line (48, Y2)-(0, Y2), vbGreen
    Form5.Picture2.PSet (5, Y2), vbGreen
    If (Y2 <> 0) Then
        Form5.Picture2.Print Y2
    End If
Next Y2
End Function

'Función de espera temporal ante eventos
Function espera(ByVal Segundos As Long)
On Local Error Resume Next
hora = Timer
Do Until Timer >= hora + Segundos
    DoEvents
Loop
End Function

'Toma de datos en cada muestreo realizado
Function toma_datos()
Print #1, "Hora de toma: " & Time
Print #1, "Temperatura", "Acidez", "Nivel", "Servo", "Filtro", "Lámpara",
"Termocalentador"
```

```
Print #1, Format(temp, "#00.00"), Format(acid, "##00.0"), s_nivel, s_servo, s_filtro, s_lampara, s_termo
Print #1, vbCrLf    'Antes de imprimir las siguientes, provocamos un salto de línea
End Function
```

2.8. Módulo 2 (Variables.bas)

```
Option Explicit
'Variables para la lectura de los datos del puerto serie
Public sData As String, lect_temp As String, lect_acid As String, lect_nivel As String
'Variables para el fraccionamiento de los datos recibidos por el puerto serie
Public primer_char As String, segundo_char As String, mensaje As String
'Variables auxiliares
Public pregunta As Integer, control As Boolean, error As Boolean, test As Boolean, valid As Boolean, cont As Integer, hora As Double, retardo As Long
'Variables numéricas de los datos recibidos
Public temp As Single, acid As Single, nivel As Boolean
'Variables de personalización del sistema
Public max_temp As Single, min_temp As Single, max_acid As Single, min_acid As Single
'Límites temporales establecidos para temperatura y acidez
Public lim_t_temp As Integer, lim_t_acid As Integer
'Variables empleadas en la escritura del registro
Public s_nivel As String, s_filtro As String, s_lampara As String, s_termo As String, s_servo As String
'Variables de control en el testeo del sistema
Public test_puerto As Boolean, test_lampara As Boolean, test_filtro As Boolean, test_termo As Boolean, test_servo As Boolean
'Cadenas empleadas para mostrar los resultados del testeo
Public c_puerto As String, c_lampara As String, c_filtro As String, c_termo As String, c_servo As String
'Variables en forma de byte o integer que son transmitidas a través del puerto serie
Public orden As Byte, periodo As Integer
'Cadenas de control en el testeo del sistema
Public str_lampara As String, str_filtro As String, str_termo As String, str_servo As String
```

```
'Vectores empleados para guardar los valores de temperatura y acidez
Public v_temp(2880) As Single, v_acid(2880) As Single
'Índices empleados en el redibujo de los valores de las gráficas
Public i As Integer, j As Integer
'Variable que indica que el dibujo de tendencias ya ha comenzado
Public tend As Boolean, toma As Single
'Variables para evolución temporal de las gráficas de tendencia
Public Xt As Single, Xt_1 As Single
'Variables para creación de puntos de tendencias (temperatura)
Public YT As String, YT_1 As String
'Variables para creación de puntos de tendencias (acidez)
Public YA As String, YA_1 As String
'Variables auxiliares (para registros)
Public registro_data As Boolean, registro_event As Boolean, ret_data As Long,
ret_event As Long
'Variable relacionada con el SCADA web, que indica a la TAD Arduino el modo de
funcionamiento elegido
Public modo_arduino As Integer
```

3. CÓDIGO DE HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>SCADA para acuario</title>
    <meta http-equiv="refresh" content="300 ">
  </head>
  <body bgcolor="aqua">
    <h1>SCADA en línea para control de acuarios a través de la plataforma
Arduino.</h1>
    <hr />
    <div style="padding: 10px; float: left; width: 45%; text-align: justify;">
    <p>Variables a controlar:</p>
      <ul>
        <li>Temperatura:       </li>
          <br />
          <li>Acidez:      </li>
          <br />
          <li>Nivel:     </li>
      </ul>
    <br />
    <p>Estado de los acuadores:</p>
      <ul>
          <li>Filtro de agua:     </li>
          <br />
        <li>Termocalentador:     </li>
          <br />
        <li>Lámpara:     </li>
        <br />
      </ul>
    </div>
    <div style="padding: 10px; float: right; width: 45%; text-align: justify;">
    <p>Activación de los actuadores:</p>
      <ul>
        <li>Activación del filtro:
              <form method="get\"> <input type="hidden" name="DIG7"
value="0"> <input type="checkbox" name="DIG7" value="7"
onclick="submit();">ON
              </form></li>
        <br />
          <li>Activación del termocalentador:
```

```
                <form method="get\"> <input type="hidden" name="DIG6"
value="0"> <input type="checkbox" name="DIG6" value="6"
onclick="submit();">ON
                </form></li>
          <br />
            <li>Activación de la lámpara:
                <form method="get\"> <input type="hidden" name="DIG8"
value="0"> <input type="checkbox" name="DIG8" value="8"
onclick="submit();">ON
                </form></li>
          <br />
            <li>Activación del servo (para ajuste de alimentación):
                <form method="get\"> <input type="checkbox" name="SRV1"
value="1"                 onclick="submit();">ON
                </form></li>
          <br />
            <li>Activación del servo (para ajuste de acidez):
                <form method="get\"> <input type="checkbox" name="SRV2"
value="2"                 onclick="submit();">ON
                </form></li>
          <br />
          </ul>
     </div>
   </body>
</html>
```

TÍTULO: **SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB**

PLANOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBRERO, S/N

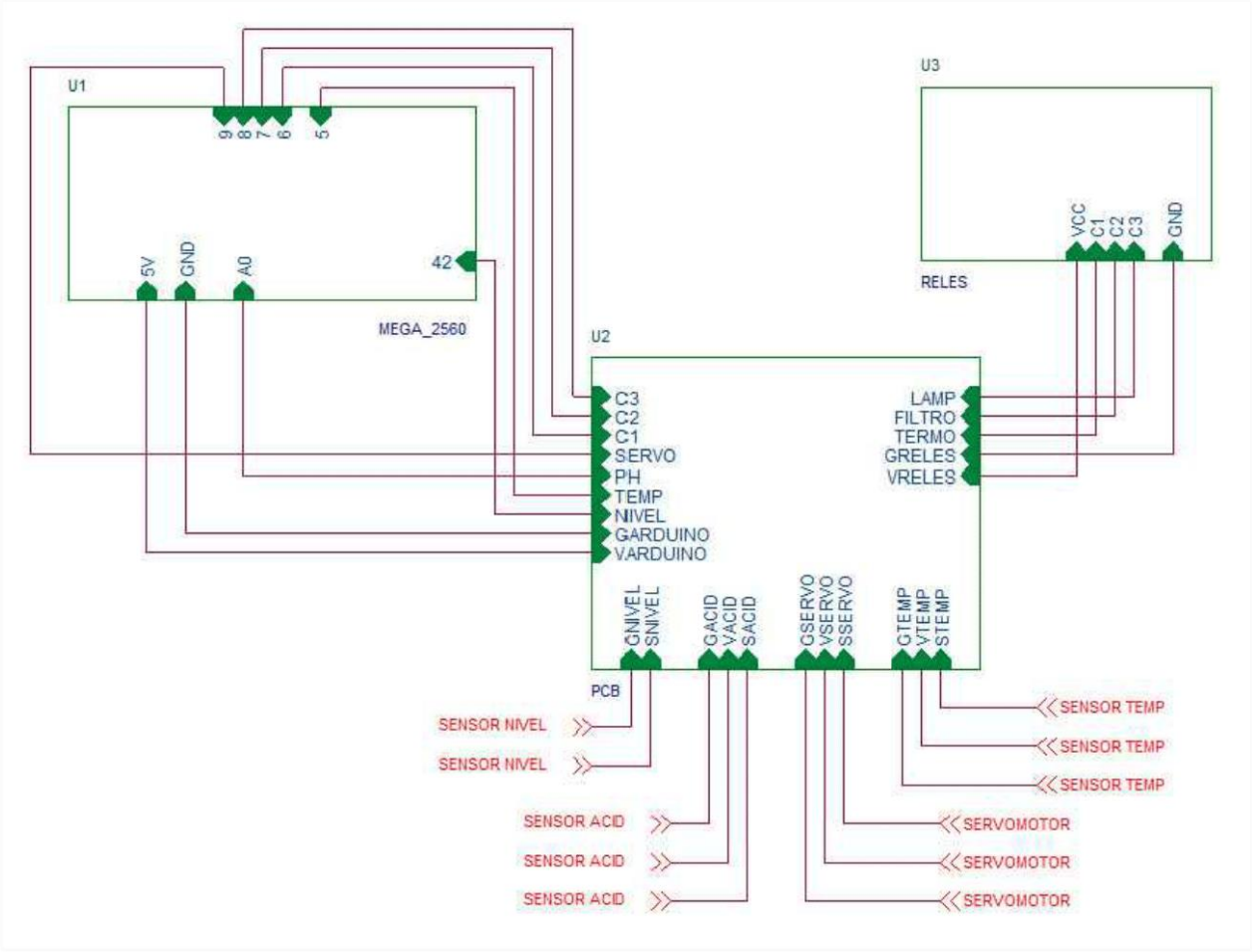
15405 - FERROL

FECHA: **FEBRERO DE 2015**

AUTOR: EL ALUMNO

Fdo.: NÉSTOR DE JUAN VÁZQUEZ

Índice



UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A66

TÍTULO DEL TFG:

SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB

TÍTULO DEL PLANO:

CONEXIONADO ELEMENTOS ACUARIO

FECHA: FEBRERO-2015

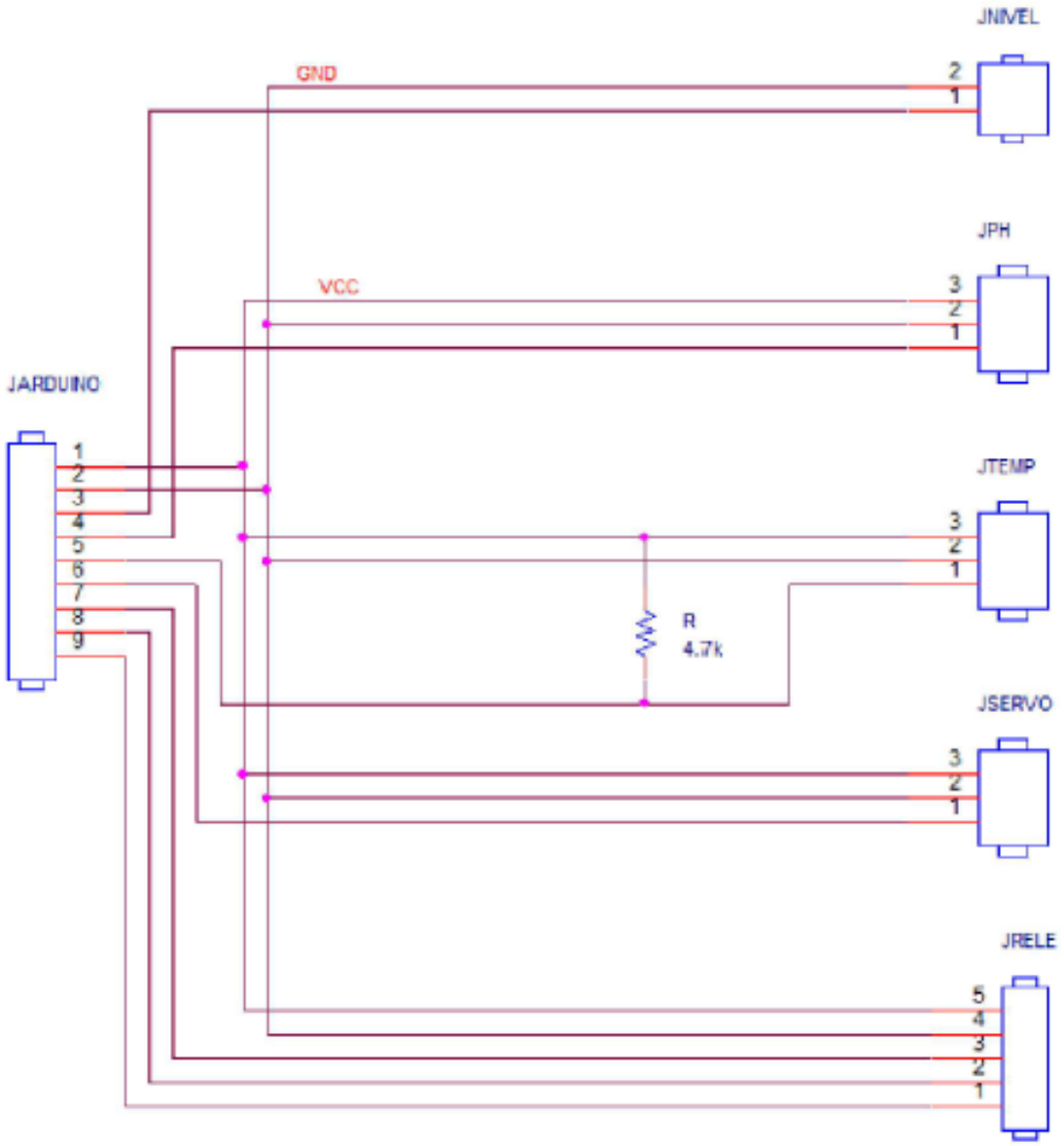
ESCALA: --

AUTOR:

NÉSTOR DE JUAN VÁZQUEZ

FIRMA:

PLANO Nº: 01



UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A66

TÍTULO DEL TFG:

SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB

TÍTULO DEL PLANO:

ESQUEMA ELÉCTRICO PLACA CONEXIONES

FECHA: FEBRERO-2015

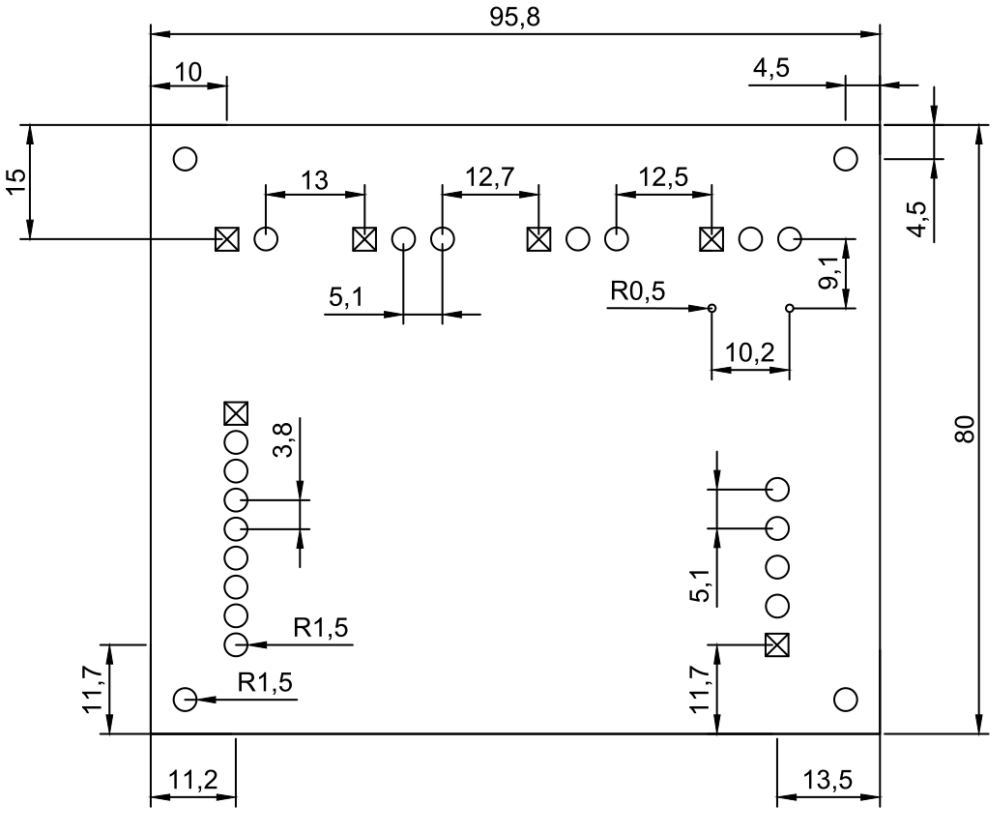
ESCALA: --

AUTORES:

NÉSTOR DE JUAN VÁZQUEZ

FIRMA:

PLANO Nº: 02



UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A66

TÍTULO DEL TFG:

SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB

TÍTULO DEL PLANO:

DIMENSIONES PCB DISEÑADA

FECHA: FEBRERO-2015

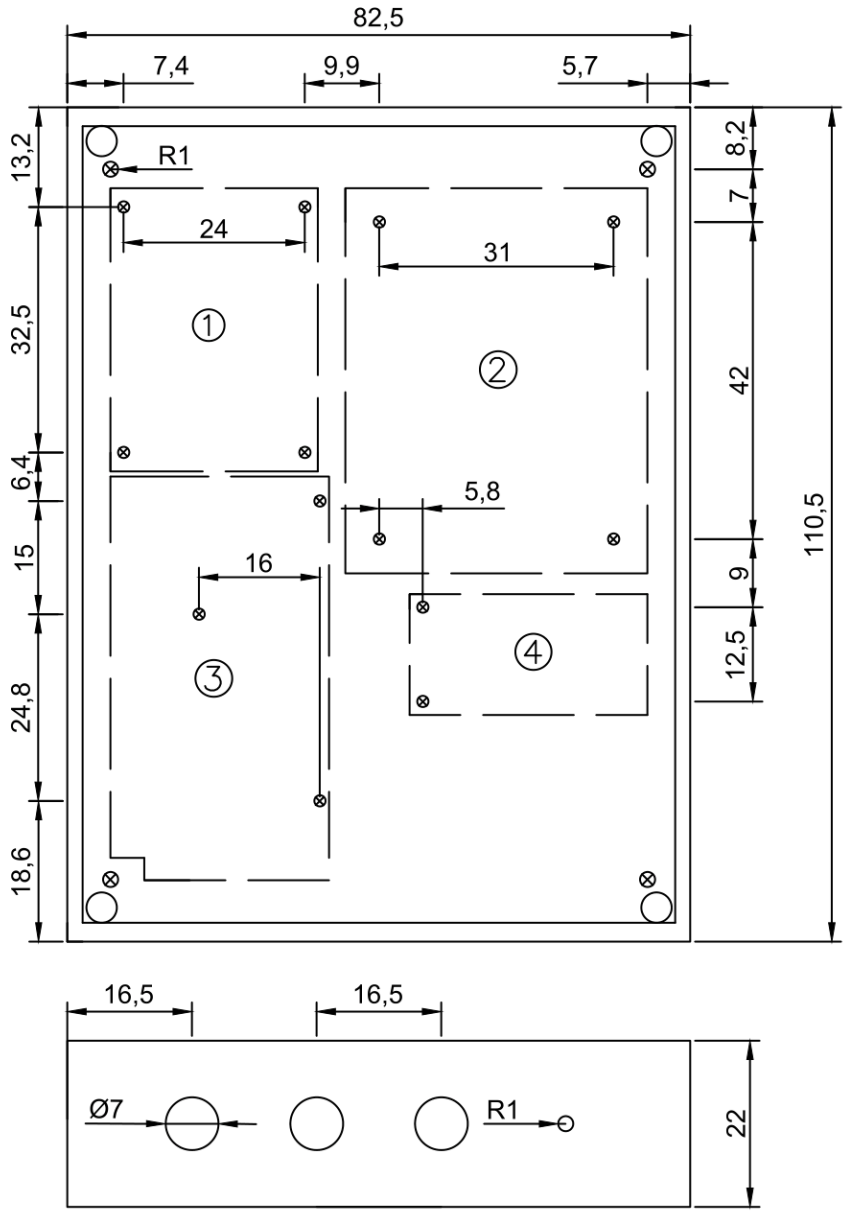
ESCALA: 1:1

AUTOR:

NÉSTOR DE JUAN VÁZQUEZ

FIRMA:

PLANO Nº: 03



① AGUJEROS CORRESPONDIENTES A PLACA DE RELÉS
② AGUJEROS CORRESPONDIENTES A PCB DISEÑADA
③ AGUJEROS CORRESPONDIENTES A TARJETA ARDUINO
④ AGUJEROS CORRESPONDIENTES A PCB DEL SENSOR DE ACIDEZ

UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A66

TÍTULO DEL TFG:

SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB

TÍTULO DEL PLANO:

AGUJEROS PARA TORNILLOS Y PRENSAESTOPAS

FECHA: FEBRERO-2015

ESCALA: 1:2

AUTOR:

NÉSTOR DE JUAN VÁZQUEZ

FIRMA:

PLANO Nº: 04

TÍTULO: **SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB**

PLIEGO DE CONDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBRERO, S/N

15405 - FERROL

FECHA: **FEBRERO DE 2015**

AUTOR: EL ALUMNO

Fdo.: NÉSTOR DE JUAN VÁZQUEZ

Índice

1. ESPECIFICACIONES DE MATERIALES

Las especificaciones de los materiales y elementos constitutivos asociados al proyecto realizado quedan descritos a continuación:

- *Placa Arduino MEGA 2560 Rev3, basada en microcontrolador ATMega2560. Cable de alimentación USB incluido.* Homologado con marcado de conformidad CE. Acorde con la directiva RoHS (2011/65/UE).
- *Módulo de 4 relés de 5V. Preparado para el trabajo con placas Arduino. Dimensiones: 7.7x5.0x1.8 cm.* Homologado con marcado de conformidad CE.
- *Sensor de temperatura sumergible DS18B20*. Homologado con marcado de conformidad CE. Grado de protección IP68.
- *Sensor de medición de acidez SEN-0161*. Homologado con marcado de conformidad CE. Grado de protección IP68.
- *Sensor flotador de nivel C-7236*. Homologado con marcado de conformidad CE.
- *Termocalentador XiLONG XL-025A*. Potencia de 25W. Homologado con marcado de conformidad CE. Grado de protección IP68.
- *Filtro de agua Elite JET-FLO 50*. Potencia de 3.6W. Homologado con marcado de conformidad CE.
- *Lámpara LED*. Potencia de 9W. Homologada con marcado de conformidad CE. Ref.: GL-18T.
- *Servomotor de ángulo de rotación 180º*. Homologado con marcado de conformidad CE.
- *Pica de titanio para eliminación de corriente residual*. Homologada con marcado de conformidad CE. Acorde con la directiva RoHS (2011/65/UE).
- *Nano Router inalámbrico TP-LINK, modelo TL-WR702N, velocidad máxima de 150Mbps.* Homologado con marcado de conformidad CE.
- *Caja para montajes eléctricos y electrónicos Retex, gama Elbox. Dimensiones de 232.5x77x180.5 mm. Tapa y base fabricadas en ABS de alta resistencia autoextinguible según UL 94-V0. Panel frontal y posterior en aluminio.* Homologada con marcado de conformidad CE. Grado de protección IP65.

2. CALIBRACIONES

Para garantizar el correcto funcionamiento del sistema diseñado, es de necesaria importancia que los resultados de las magnitudes leídas a través de los sensores sean lo más fieles posibles a las reales, garantizando que no se llevan a cabo medidas falsas.

En el trabajo realizado, para evitar que este hecho se produzca será necesario llevar a cabo la calibración de los sensores empleados, describiendo:

- El procedimiento seguido.
- Las condiciones de dichas pruebas.
- Las modificaciones a realizar según los resultados obtenidos.

Para el sensor de nivel no será necesario llevar a cabo ningún tipo de proceso de calibración, prueba o ensayo debido a que no necesita de los mismos.

2.1. Sensor de temperatura DS18B20

Para la calibración del sensor de temperatura del acuario, se seguirá el procedimiento definido a continuación. Dicho procedimiento no sigue ninguna norma específica ni ninguna recomendación específica del fabricante.

Los pasos del proceso a seguir serán:

1. En el interior de un recipiente lleno de agua, se introducirá el sensor en cuestión. Junto a dicho sensor, se colocarán dos termómetros de carácter sumergible.
2. Se tomarán medidas en intervalos de tiempo fijos de 20 segundos.
3. Tras esperar a que se establezca la temperatura, se observarán y compararán los valores obtenidos a través de Arduino y las lecturas anotadas de los termómetros. Todo lo recogido queda mostrado en la tabla siguiente:

Tiempo (seg)	Temperatura (ºC)		
	DS18B20	Termómetro 1	Termómetro 2
0	33,38	34	34
20	33,31	34,2	33,8
40	33,25	34,1	34
60	33,19	34,2	34
80	33,13	34	34
100	33	34,1	34
120	32,94	34,1	34
600	29,25	30,5	30
900	29,19	29,5	29,2

Tabla 2.1.1 - Lecturas recogidas de temperatura

Teniendo en cuenta la última muestra, que se corresponde a un instante de tiempo en el cual la temperatura está establecida en torno a un valor constante, tal y como se calcula en el apartado de cálculos el error absoluto entre las muestras de los termómetros y del sensor es de:

$$e_a = 0.16\ ºC \tag{3.1.1}$$

Será necesario por lo tanto añadir un offset al resultado de temperatura obtenido en el muestreo. Una vez calibrado el sensor ajustando el error existente, se ha llevado a cabo un seguimiento de la evolución de temperatura para observar la respuesta del sensor. Se ha planteado una evolución desde una temperatura inicial de 31.5 ºC hasta los 34 ºC finales.

Las tomas se han realizado en intervalos de tiempo de medio minuto. Dichas tomas están recogidas en la siguiente tabla:

Tiempo (min)	Temperatura (ºC)	Tiempo (min)	Temperatura (ºC)
0	31,5	10,5	33,31
0,5	31,69	11	33,38
1	31,75	11,5	33,5
1,5	31,75	12	33,63
2	31,81	12,5	33,69
2,5	31,87	13	33,81
3	31,94	13,5	33,88
3,5	32	14	34
4	32,06	14,5	34
4,5	32,13	15	34,06
5	32,25	15,5	34,13
5,5	32,36	16	34,13
6	32,52	16,5	34,13
6,5	32,68	17	34,06
7	32,81	17,5	34,06
7,5	33,06	18	34,06
8	33,13	18,5	34
8,5	33,13	19	34
9	33,13	19,5	33,94
9,5	33,13	20	34
10	33,19		

Tabla 2.1.2 - Medidas de temperatura tomadas

Aprovechando estos datos, se ha elaborado un gráfico en el que se recoge la respuesta temporal del sensor ante las evoluciones de temperatura. La gráfica en cuestión se muestra abajo, incluyendo una línea de tendencia de la evolución de la magnitud medida:

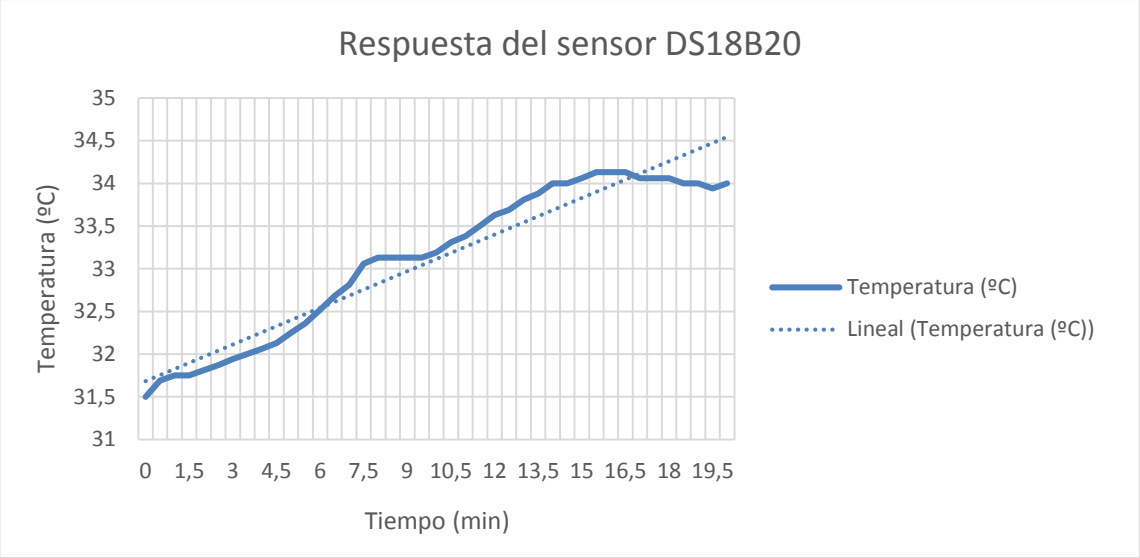


Figura 2.1.1 - Respuesta temporal del sensor de temperatura

2.2. Sensor de acidez SEN-0161

Para calibrar el sensor de acidez del acuario, se seguirá el procedimiento descrito en la documentación del producto, el cual se menciona a continuación y que puede ser consultado en la bibliografía descrita en la memoria:

1. Poner el electrodo de pH dentro de la solución estándar cuyo pH es de 7.00, o bien directamente cortocircuitar la entrada del conector BNC. Observar el valor imprimido en el puerto serie, y ver cuál es la desviación respecto al valor mencionado. La diferencia debe ser plasmada en el "Offset" en el código.

   En este caso, el valor obtenido en el IDE de Arduino es de 6.9. Teniendo en cuenta que el valor determinado en el procedimiento es de 7.00, el valor del offset será:

   $$Offset = 7.0 - 6.9 = 0.1 \quad (3.2.1)$$

2. Poner el electrodo en la muestra cuyo valor de pH es 4.00. Esperar un minuto, ajustar la ganancia potencial de la placa de conexión y dejar el valor estabilizarse en torno a 4.00.

3. Atendiendo a las características lineales del electrodo, después de la anterior calibración se permite la posibilidad de medir directamente el valor de pH de la muestra de solución alcalina, pudiendo recalibrar el sensor para una mayor precisión si así se prefiere. La calibración alcalina usa la solución estándar cuyo valor de pH es de 9.18.

Una vez realizados estos tres pasos, y tras comprobar que los resultados de la norma se cumplen, se puede considerar que el sensor está calibrado y preparado para su funcionamiento. Con el objetivo de asegurar el correcto funcionamiento del sensor para cuando el sistema de control esté montado al completo, tras acabar la fase de montaje y conexionado pueden repetirse los pasos (2) y (3) para asegurarse así que la respuesta sigue siendo la adecuada.

3. CONDICIONES DE HARDWARE Y SOFTWARE

El hardware empleado en el sistema estará compuesto por la tarjeta de adquisición de datos (TAD), todas las placas y sensores pertenecientes al sistema y el computador empleado para el control del acuario. Dichos elementos deberán encontrarse en buen estado, estar conectados entre sí de forma adecuada mediante los enlaces cableados correspondientes y presentar la calidad mínima exigida en el apartado 1 de este documento. Todas las pruebas y ensayos que puedan ser necesarios para la puesta en marcha del sistema son los determinados en el apartado 3 de este mismo documento.

Por parte del software, y como ha quedado determinado de forma concisa en la memoria, se empleará el programa Microsoft Visual Studio 6.0 y el IDE o Entorno de desarrollo oficial de Arduino. Si el usuario desea además poder realizar el control remoto del sistema vía web, será necesario disponer de un navegador web en el equipo que el usuario tenga pensado emplear para dicho control. Además, en el ordenador empleado deberá estar instalado un sistema operativo que permita ejecutar ambos problemas sin incidencias (recomendable emplear versiones de Windows de XP en adelante).

La garantía del sistema diseñado será de dos años desde el momento de la adquisición. Cualquier modificación fuera de lo establecido en los ámbitos de hardware y software del sistema diseñado eliminará cualquier tipo de garantía existente.

TÍTULO: **SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB**

ESTADO DE MEDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBRERO, S/N

15405 - FERROL

FECHA: **FEBRERO DE 2015**

AUTOR: EL ALUMNO

Fdo.: NÉSTOR DE JUAN VÁZQUEZ

Índice

1. DEFINICIONES

A continuación, y como paso previo al estado de mediciones propiamente dicho, se definirán a fin de aclarar los siguientes conceptos para una mejor comprensión de la posterior lista de materiales empleada en el presente trabajo.

Las definiciones son las siguientes:

- Unidad: cada uno de los componentes integrantes de una construcción; en este caso, de la planta creada. Se pueden dividir en tres tipos:
    - Elementos unitarios: unidades elementales de carácter indivisible.
    - Elementos complejos: elementos formados por un conjunto de elementos unitarios.
    - Elementos funcionales: agrupación de elementos simples o complejos con función propia.
    - Partidas alzadas: aquellas unidades de obra de las cuales no sea posible desglosar, en forma razonable, el detalle de las mismas.
- Capítulos de obra: conjunto de partidas bajo las cuales se agrupan las unidades en grupos en los cuales compartan alguna característica o función común.

2. ESTADO DE MEDICIONES

En la siguiente tabla se muestra el estado de mediciones correspondientes al trabajo realizado:

Capítulo de obra	Unidad	Función	Tipo	Uds.
Instrumentación	DS18B20	Sensor	Elemental	1
	SEN-0161	Sensor	Elemental	1
	C-7236	Sensor	Elemental	1
	MicroServo	Actuador	Elemental	1
	EliteJet Flo 50	Actuador	Elemental	1
	XiNGA	Actuador	Elemental	1
	GT-138T	Actuador	Elemental	1
	Pica titanio	Aislamiento	Elemental	1
Circuitos impresos	Arduino MEGA R2560	TAD	Elemental	1
	Ethernet Shield W5100	PCB	Elemental	1
	Placa conexiones	PCB	Compleja	1
	Placa relés x4	PCB	Elemental	1
	Placa SEN-0161	PCB	Elemental	1
Conexión a Internet	Nano Router TL-WR702N	Conexión	Elemental	1
	Cable Ethernet	Cableado	Elemental	1
Materiales de construcción	Cables macho-hembra	Cableado	Elemental	5
	Cables placa perforada	Cableado	Elemental	17
	Regletas de conexión	Cableado	Elemental	3
	Ventosas atornilladas	Sujeción	Elemental	4
	Prensaestopas	Aislamiento	Elemental	3
	Caja para montaje	Presentación	Elemental	1

Tabla 2.1 - Estado de mediciones del trabajo

A continuación se muestra una segunda tabla con el desglose de los materiales que han sido necesarios para la realización de la PCB empleada como placa de conexiones:

Ud. Compleja	Ud. Elemental	Función	Uds.
Placa conexiones	Placa fotosensible 10X16cm	Base	1
	Regleta atornillada (2 tornillos)	Conexionado	1
	Regleta atornillada (3 tornillos)	Conexionado	3
	Regleta atornillada (2 tornillos)	Conexionado	1
	Regleta atornillada (9 tornillos)	Conexionado	1

Tabla 2.2 - Estado de mediciones desglosado de unidad compleja

TÍTULO: **SCADA PARA CONTROL DE ACUARIOS MEDIANTE ARDUINO Y VB**

PRESUPUESTO

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBRERO, S/N

15405 - FERROL

FECHA: **FEBRERO DE 2015**

AUTOR: EL ALUMNO

Fdo.: NÉSTOR DE JUAN VÁZQUEZ

Índice

1. CUADRO DE PRECIOS

Previa realización del presupuesto necesario para materializar el trabajo, se muestra un cuadro de precios con cada uno de los precios de las correspondientes unidades de obra, ya sean elementales, funcionales, o complejas, necesarias para dicha materialización:

Capítulo de obra	Unidad	Tipo	Precio/Ud.
Instrumentación	DS18B20	Elemental	13,14 €
	SEN-0161	Elemental	24,90 €
	C-7236	Elemental	16,70 €
	MicroServo	Elemental	7,24 €
	EliteJet Flo 50	Elemental	11,77 €
	XiNGA	Elemental	10,40 €
	GT-138T	Elemental	8,69 €
	Pica titanio	Elemental	12,00 €
Circuitos impresos	Arduino MEGA R2560	Elemental	47,19 €
	Ethernet Shield W5100	Elemental	10,00 €
	Placa conexiones***	Compleja	15,31 €
	Placa relés x4	Elemental	14,90 €
	Placa SEN-0161	Elemental	*
Conexión a Internet	Nano Router TL-WR702N	Elemental	23,90 €
	Cable Ethernet	Elemental	**
Materiales de instalación	Cable macho-hembra (Ud)	Elemental	0,25 €
	Cable placa perforada (m)	Elemental	0,70 €
	Regleta de conexión (Ud)	Elemental	0,06 €
	Ventosa atornillada (Ud)	Elemental	2,50 €
	Prensaestopa (Ud)	Elemental	0,67 €
	Caja para montaje	Elemental	19,60 €
Mano de obra	Hora/Operario	Elemental	2,50 €
	Hora/Programador	Elemental	1,50 €

Tabla 1.1 - Cuadro de precios del trabajo

*No se incluye ningún precio para esta unidad, ya que se encuentra incluida en el precio del sensor de acidez SEN-0161.

**El cable, así como su precio, están incluidos en el router mostrado en la celda superior.

***Como unidad compleja, se puede desglosar el precio de la placa de conexiones en su presupuesto particular, recogido en la siguiente tabla:

Ud. Compleja	Unidad	Tipo	Precio/Ud.
Placa conexiones	Placa fotosensible 10x16cm	Elemental	4,24 €
	Regleta atornillada (2 tor.)	Elemental	0,82 €
	Regleta atornillada (3 tor.)	Elemental	1,44 €
	Regleta atornillada (5 tor.)	Elemental	1,98 €
	Regleta atornillada (9 tor.)	Elemental	3,95 €

Tabla 1.2 - Cuadro de precios desglosado de unidad compleja

2. PRESUPUESTO

A continuación se puede visualizar el presupuesto total del trabajo:

Capítulo de obra	Unidad	Precio/Ud.	Uds.	Total
Instrumentación	DS18B20	13,14 €	1	**13,14 €**
	SEN-0161	24,90 €	1	**24,90 €**
	C-7236	16,70 €	1	**16,70 €**
	MicroServo	7,24 €	1	**7,24 €**
	EliteJet Flo 50	11,77 €	1	**11,77 €**
	XiNGA	10,40 €	1	**10,40 €**
	GT-138T	8,69 €	1	**8,69 €**
	Pica titanio	12,00 €	1	**12,00 €**
Circuitos impresos	Arduino MEGA R2560	47,19 €	1	**47,19 €**
	Ethernet Shield W5100	10,00 €	1	**10,00 €**
	Placa conexiones**	15,31 €	1	**15,31 €**
	Placa relés x4	14,90 €	1	**14,90 €**
	Placa SEN-0161	*	1	*****
Conexión a Internet	Nano Router TL-WR702N	23,90 €	1	**23,90 €**
	Cable Ethernet	**	1	**
Materiales de instalación	Cable macho-hembra (Ud)	0,25 €	5	**1,25 €**
	Cable placa perforada (Ud)	0,70 €	17	**11,90 €**
	Regleta de conexión (Ud)	0,06 €	3	**0,18 €**
	Ventosa atornillada (Ud)	2,50 €	4	**10,00 €**
	Prensaestopa (Ud)	0,67 €	3	**2,01 €**
	Caja para montaje	19,60 €	1	**19,60 €**
Mano de obra	Hora/Operario	2,50 €	10	**25,00 €**
	Hora/Programador	1,50 €	50	**75,00 €**

Total	**271,08 €**

Tabla 2.1 - Presupuesto del trabajo

*No se incluye ningún precio para esta unidad, ya que se encuentra incluida en el precio del sensor de acidez SEN-0161.

**El cable, así como su precio, están incluidos en el router mostrado en la celda superior.

***Como unidad compleja, se puede desglosar el precio de la placa de conexiones en su presupuesto particular, recogido en la siguiente tabla:

Ud. Compleja	Ud. Elemental	Precio/Ud.	Uds.	Total
Placa conexiones	Placa fotosensible 10x16cm	4,24 €	1	**4,24 €**
	Regleta atornillada (2 tor.)	0,82 €	1	**0,82 €**
	Regleta atornillada (3 tor.)	1,44 €	3	**4,32 €**
	Regleta atornillada (5 tor.)	1,98 €	1	**1,98 €**
	Regleta atornillada (9 tor.)	3,95 €	1	**3,95 €**

Total	15,31 €

Tabla 2.2 - Presupuesto desglosado de unidad compleja

Todos y cada uno de los precios definidos en las tablas arriba mostradas incluyen en los precios de las distintas unidades el pertinente Impuesto sobre el Valor Añadido o IVA correspondiente al 21%.

No se han tenido en consideración los posibles gastos de envío u otros tipos de costes asociados a cada uno de los productos, debido a su carácter variable o en algunos casos incluso nulo.