INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal:
“PROTOTIPO DE PANTALLA DE AGUA PARA EXHIBIR PEQUEÑOS ANUNCIOS”
TT2012-A038

Que para cumplir con la opción de titulación curricular en la carrera de:
“Ingeniería en Sistemas Computacionales”

Presenta

Néstor Carlos González De La Rosa

Directores:

M. en C. Araujo Díaz David

M. en C. Méndez Segundo Laura

México D.F. a 24 de mayo de 2013.

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

SUBDIRECCIÓN ACADÉMICA

N° de Registro: TT2012-A0038 Serie: Amarilla México D.F. a 24 de mayo de 2013

Documento técnico

PROTOTIPO DE PANTALLA DE AGUA PARA EXHIBIR PEQUEÑOS ANUNCIOS

Presenta

González De La Rosa Nestor Carlos[1]

Directores

M. en C. Araujo Díaz David
M. en C. Méndez Segundo Laura

Resumen

El desarrollo de este proyecto muestra el diseño y construcción de un prototipo de pantalla de agua. Para lograr esto, se toma una serie de caracteres y/o figuras simples, se transforman en un formato apto para el sistema, se envían a través de un puerto USB a un dispositivo controlador el cual la representa físicamente en una cortina de agua. El sistema cuenta con capacidades para guardar representaciones, controlar la velocidad de presentación, así como poder calibrar la salida de chorro de agua del sistema con el fin de que la secuencia sea legible.

PALABRAS CLAVE: electrónica, programación, traficación, procesamiento digital de señales

[1]E-mail: nestorcarlos@hotmail.com

ESCUELA SUPERIOR DE CÓMPUTO

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE FORMCIÓN INTEGRAL E INSTITUCIONAL

ESCOM

COMISIÓN ACADÉMICA DE TRABAJO TERMINAL

México D.F. a 22 de mayo de 2012.

ING. APOLINAR FCO. CRUZ LAZARO
PRESIDENTE DE LA COMISIÓN ACADEMICA
DE TRABAJO TERMINAL
PRESENTE

Por medio del presente, informamos que el alumno que integra el **TRABAJO TERMINAL 2012- A0038** titulado **"Pantalla De Agua Para Exhibir Pequeños Anuncios"**, concluyo satisfactoriamente su trabajo.

El empastado del Reporte Técnico Final y el Disco Compacto (CD) fueron revisados ampliamente por sus servidores y corregidos, cubriendo el alcance y el objetivo planeados en el protocolo original y de acuerdo a los requisitos establecidos por la Comisión que Usted preside.

ATENTAMENTE

Araujo Díaz David
M. en C. Araujo Díaz David

M. en C. Méndez Segundo Laura

Directores

Advertencia

“Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan.”

La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

En La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000 Extensión 52000

Agradecimientos

A mi hermano Luis por haberme apoyado durante años, por confiar en mí, por sus consejos y por su ejemplo.

A mi familia por brindarme su afecto y cariño incondicional, por el ánimo que siempre me han infundido.

A Dios por que ha sido fiel, por darme fuerza para continuar.

Ing. Nestor Carlos González De La Rosa.

ÍNDICE

ÍNDICE DE FIGURAS

RESUMEN

El desarrollo de este proyecto muestra el diseño y construcción de un prototipo de pantalla de agua. Para lograr esto, se toma una serie de caracteres y/o figuras simples, se transforman en un formato apto para el sistema, se envían a través de un puerto USB a un dispositivo controlador el cual la representa físicamente en una cortina de agua. El sistema cuenta con capacidades para guardar representaciones, controlar la velocidad de presentación, así como poder calibrar la salida de chorro de agua del sistema con el fin de que la secuencia sea legible.

INTRODUCCIÓN

Seguramente algunos hemos visto las famosas fuentes danzantes, orgullo de muchas figuras y centros públicos, sin embargo en la evolución de como el hombre se ha fascinado por el dinamismo del agua admirando sus multiformes maneras de impresionarnos a través de las cataratas en diferentes partes del mundo, geiseres, ríos, fenómenos atmosféricos, y por supuesto el mismo océano. Durante el curso dela historia siempre se ha intentado duplicar los sentimientos, emociones e impresiones que nos causa el agua, así creando obras de arte dinámicas las cuales se han desarrollado desde clásicas fuentes, clepsidra lugares de recreación hasta lo que actualmente se conoce como pantallas de agua.

Figura 1. Clepsidra.

En cualquier ciudad se vive en un contraste muy acentuado con un entorno natural, en donde podemos percibir un exacerbado culto a lo artificial, a vivir apresuradamente, saturando de publicidad los medios en los que nos desarrollamos, comercios artículos personales, cualquier tipo de exagerada sofisticación dedicado a una sociedad consumista; dejando muy aparte o por lo menos dejando de lado un estilo de vida tranquilo y sencillo olvidándonos de un ambiente más humano, más amable para nuestro organismo y que por lo tanto puede llegar a causar variedad de afecciones en la salud pública.

Tomando en cuenta estas cosas considero que la inserción de lugares de esparcimiento, en donde se pueda aminorar la carga de estrés diaria son muy necesarios teniendo la oportunidad de sustituir el bombardeo mercadológico con espacios para la expresión donde el proyecto de pantallas de agua que este a la vista de todos dé un mejoramiento a la imagen urbana.

CAPÍTULO 1: MARCO TEÓRICO

1.1 Estado del arte.

BIT.FALL[1].- Las pantallas de agua el producto de la imaginación y la tecnología desarrolladas primeramente como un proyecto del artista alemán Julius Popp en Artexpo, New York, llamado BIT.FALL en 2004. El cual tiene el primer antecedente a lo que parece se esta popularizando a nivel mundial. En el cual según el artista. La velocidad a la que se obtiene información, intercambio y actualización en nuestra sociedad moderna es casi inconcebible, y más efímero que nunca. El trabajo BIT.FALL traduce este proceso abstracto en una experiencia para los sentidos y es una metáfora de estas corrientes contemporáneas de la información. En BIT.FALL, la información es representada por palabras generadas por un programa de ordenador, en base a un algoritmo estadístico. El programa filtra términos relevantes de la secuencia actual de las noticias en el Internet, y transmite los valores de la unidad de control de BIT.FALL. En una fracción de segundo, BIT.FALL libera cientos de gotas a intervalos específicos, la creación de una "cascada" de las palabras. Cada gota de agua se convierte en un líquido y transitorio 'pixel' o 'bit', la unidad más pequeña de información. BIT.FALL combina dos sistemas de circulación distintas - la circulación en la naturaleza (a través de sus propias leyes, tales como la gravedad) y la circulación de la cultura (a través de los grados de atención social según lo registrado por las estadísticas). El agua, un medio amorfo, se convierte en un portador de la información cultural que sólo es perceptible por un instante y luego desaparece de nuevo. Este aspecto de BIT.FALL se refiere a la naturaleza efímera de la información cultural y de valores: mientras las percibimos, realmente somos incapaces de entenderlo.

Figura 2. Bit fall.

KOEI Co.,Ltd[2].- Posteriormente esta fascinante pantalla de agua fue creada por la compañía japonesa "KOEI CO. Industria" en 2008 y está ubicado en el edificio de la Puerta del Sur en el centro comercial de "Osaka Station City", Japón. Y también en "Trade Center Kaliningrado (Kaliningrad, Rusia)", "Canal City Hakata (Fukuoka)" en 2008, " Kyoto Meitengai Kintetsu" en la estación de Kyoto. En el cual fluyen hojas de aguas como un lienzo brillante, ya que la pantalla es la cascada y parte impresora digital. Las corrientes de agua se iluminan para mostrar un conjunto siempre cambiante de los patrones y diseños. Y se utiliza el agua para mostrar el tiempo, temperatura incluso obras de arte.

Figura 3. Pantalla de agua en Osaka, Japón.

Este concepto ha estado siendo aceptado y comercializado por diversas empresas como: "AQUA REIGN", "Smeraldo 3000" por mencionar algunas.

CAPÍTULO 2: OBJETIVO

2.1 Objetivo general.

Diseñar e implementar un sistema prototipo de pantalla de agua, el cual represente una serie de caracteres introducidos a través de un programa en la computadora para trasmitir un mensaje visualmente atractivo.

Para alcanzar el objetivo general, es necesario cumplir con los siguientes objetivos específicos.

2.2 Objetivos específicos.

- Análisis y selección del controlador adecuado, que cubran las necesidades del sistema.
- Análisis y selección del hardware que controlan el flujo de líquido.
- Análisis de las herramientas de software, lenguajes de programación.
- Desarrollo e implementación del sistema de software
- Desarrollo e implementación de la interfaz USB PC-dispositivo de control.
- Desarrollo e implementación de dispositivo de control.
- Desarrollo e implementación de la parte mecánica del control del fluido
- Integración y pruebas al sistema.
- Generación de documentación: manual de usuario, manual técnico y especificaciones

CAPÍTULO 3: JUSTIFICACIÓN

Los espacios urbanos están saturados de contaminación visual ya que se genera basura, ocasionando estrés, además de ser en texto plano y estático, que deteriora la imagen de las áreas públicas por lo tanto propongo un sistema de pantalla de agua la cual es agradable y atractivo a la vista y el sonido de agua es tranquilizador además de proporciona un espacio donde se trasmitirá un mensaje y de esparcimiento y diversión, la cual podrá transmitir cualquier mensaje visual de forma creativa y dinámica además de proporcional otros usos como son: el de una barrera pasiva, que puede abrirse o cerrarse, una fuente interactivo, un espacio de expresión artística.

Durante mucho tiempo la publicidad se ha basado en tradicionales pancartas folletos anuncios, espectaculares que además de contaminar y ser desagradable saturan los espacios públicos no dejando lugar a los espacios de expresión y esparcimiento y producir mucho desperdicio de materias primas

Diseñar un sistema capaz de revolucionar en el ámbito de la publicidad combinando tecnologías de última generación y aplicando conocimientos de diversas ramas de la ingeniería, siendo el control del flujo del agua y los efectos visuales que esta produzcan los principales elementos utilizados dentro del sistema.

Este sistema será desarrollado con trabajo de ingeniería mexicana del Instituto Politécnico Nacional contribuyendo con la innovación tecnológica

Este sistema es de interés para: el ramo turístico, hotelero, los centros comerciales, los espacios públicos, los restaurantes, convenciones, exposiciones, conciertos, conferencias, área de publicidad de las empresas, las estancias de los edificios, los balnearios, estaciones del sistema de transporte, monumentos, esculturas.

CAPÍTULO 4: ANÁLISIS

Durante el desarrollo de este capítulo se describe la metodología a seguir y se expone las características de las herramientas de software y hardware que se han elegido y se explica brevemente sus ventajas que ayudaran a la elaboración de este proyecto

4.1 Metodología.

Durante la realización de este proyecto se usara el modelo incremental retro alimentado, y la metodología estructurada orientada a procesos, el cual está constituido de las siguientes fases:

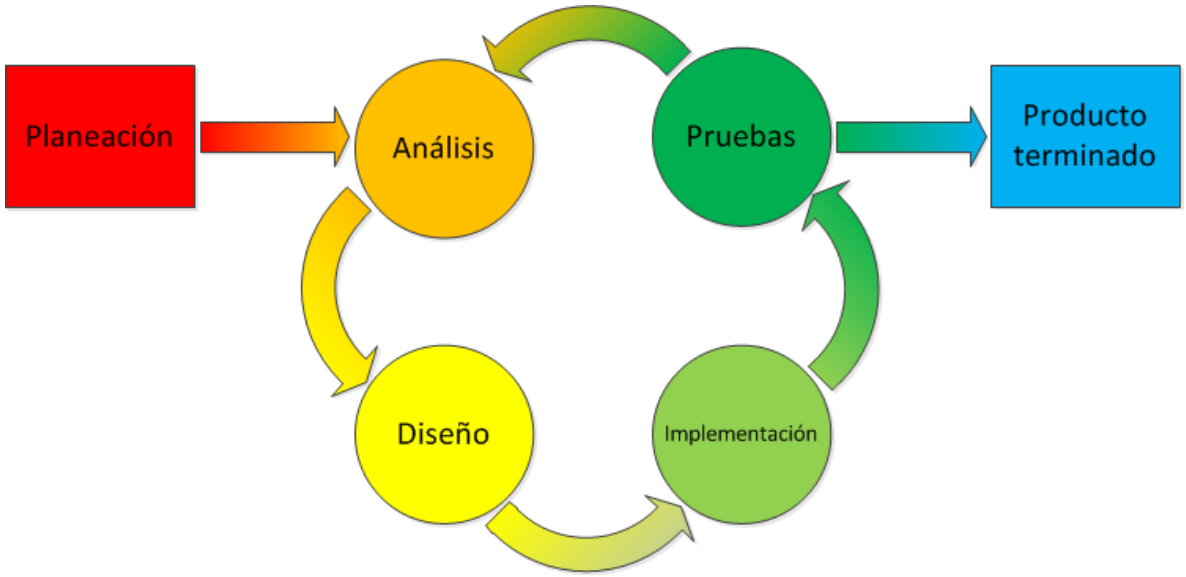

Figura 4. Metodología incremental retroalimentada.

I. Planeación
   Durante esta etapa se determinaron los siguientes puntos:
   a. Realizar el Trabajo Terminal con el proyecto pantalla de agua.
   b. Se usa la metodología incremental retroalimentada.
   c. Se determina el cronograma de actividades.

II. Análisis
   Durante esta etapa se racionaliza la forma en que se debe resolver los problemas

detectados así como la especificación de herramientas que me ayudaran al correcto desarrollo del proyecto también como su mejora constante.

a. Se utilizan para el desarrollo de software el lenguaje c#, apoyándome en los siguientes IDEs:
    i. Microsoft Visual C# Express en Windows.
    ii. MonoDevelop en Linux

b. Para la programación de la plataforma Arduino:
    i. open-source Arduino environment.

c. Para la etapa de control se necesita una configuración de dispositivos con las cuales pueda satisfacer algunos requerimientos como poder implementar una conexión USB, dispones de memoria RAM y memoria ROM, disponer de buen procesamiento se utilizara el dispositivo siguiente:
    i. Plataforma Arduino[6] Leonardo

d. Para la etapa de potencia se utiliza una configuración de triac- optoacoplador y debido a que cada electroválvula debe actuar independientemente se asignara esta configuración a cada elemento actuador:
    i. Triac 2n6344[4] el cual tiene características suficientes para los requerimientos de corriente y voltaje de las electroválvulas, y tiene un coste medio.
    ii. Moc 3011[5] el cual está conformado por un emisor y un receptor internos que permiten aislar la parte de control de la parte de potencia.

e. Electroválvulas convencionales ya que las electroválvulas comerciales tienen un costo demasiado alto, y en este proyecto se está formando un prototipo.
    i. UNIVALVULA WASHING MABE AMAZONAZ: VW-40245C: 323B1472P001

III. Diseño

Durante esta etapa se diseñaron los diferentes programas, mediante los diagramas de flujo, diagramas de proceso, diagramas de bloques, diagramas de casos. Esta parte se estará retroalimentado progresivamente, para la corrección de errores y el incremento de nuevas funcionalidades.

IV. Implementación

Durante esta etapa se puso en marcha los diagramas especificados en la etapa

anterior llevándolos a ver su funcionamiento real y quizá reformar el diagrama de procesos. Durante esta parte también interactuaran los diferentes sistemas internos del proyecto para realizar la interconexión y la trasmisión de información por medio de los diferentes protocolos que se vayan a utilizar. También se armara el cancel que sostendrá las electroválvulas

V. Pruebas

Durante esta etapa se expondrá el sistema diferentes test de calidad y de funcionamiento en el cual de tomara nota de su desempeño y sus debilidades los cuales permitirán cerciorarnos de que partes necesitan mayor atención. Además de que en esta parte se propondrán mejoras constantes que permitan incorporar mejores y nuevas funcionalidades que aumenten el valor del sistema y permitan al proyecto ser escalable y reconfigurable.

VI. Producto terminado

Durante esta etapa se entregara un producto terminado con todas sus partes funcionales y la documentación necesaria. Y se evaluara el proyecto para confirmar el estado de su calidad y comprobar su funcionamiento

CAPÍTULO 5: DISEÑO

5.1 Diseño de mecanismo hidráulico-mecánico.

En esta parte del proyecto se utilizo principalmente una electroválvula comercial económica de aproximadamente $56 pesos cada una, la cual a pesar de no ser la más adecuada, es la que entra dentro de los límites del presupuesto tomando en cuenta que otros dispositivo más adecuados son excesivamente caros como se presenta en el ejemplo de la electroválvula DSVP12N que cuesta $87 dólares aproximadamente. Las electroválvulas serán alimentadas por una estructura de tuberías de PVC la cual se describe enseguida.

Figura 5. Electroválvula común.

Figura 6. Electroválvula DSVP12N.

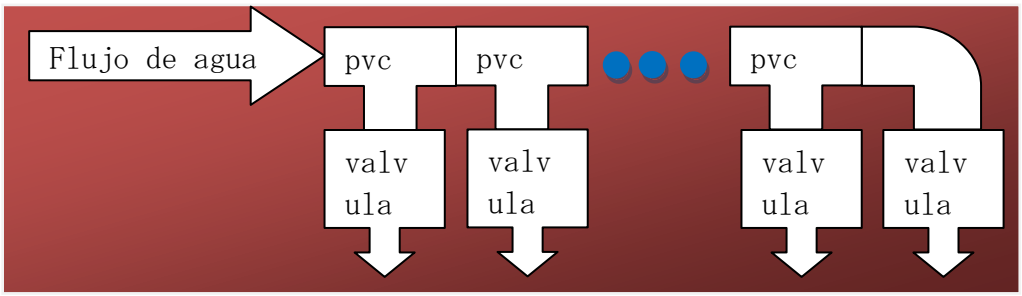

Figura 7. Distribuidor de agua.

5.2 Diseño de la etapa de potencia.

Esta parte del diseño realizara implementando un optoacoplador y un triac los cuales se especifican a continuación.

El diagrama del dispositivo MOC 3011[5] es el siguiente:

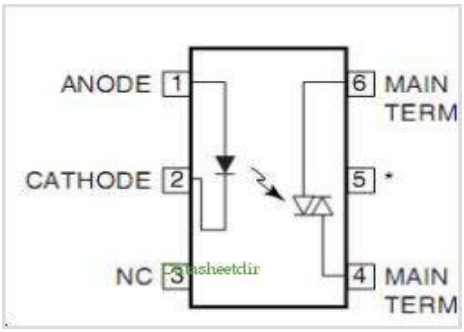

Figura 8. Diagrama del MOC3011.

El diagrama del dispositivo triac 2n6344[4] es el siguiente

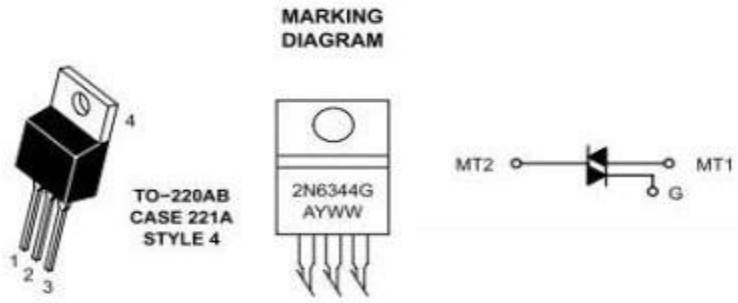

Figura 9. Diagrama del Triac 2n6344.

En conjunto se conectaran de la siguiente forma:

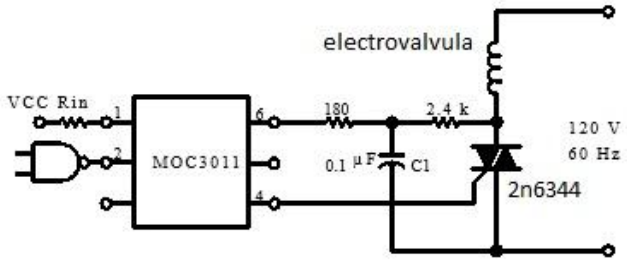

Figura 10: Diagrama de la etapa de potencia.

5.3 Diseño de la etapa de control.

La arquitectura de hardware que se utilizara será la de la plataforma de desarrollo Arduino[6] Leonardo el cual cuenta con una interfaz USB la cual resulta muy conveniente para la aplicación de este proyecto. En el mundo de la electrónica digital, el Arduino ha sido uno de los sistemas más populares debido a su costo y por ser hardware y software de código abierto, lo cual quita muchas de las trabas de esquemas electrónicos comerciales. Arduino, Leonardo, ofrece nuevas posibilidades. Esta versión puede simular un dispositivo USB. El sistema está basado en un Atmega32U4[7] de 16 MHz, con un diseño simple, debido a que el CPU toma a cargo la parte de la interfaz USB. Esto significa menos chips y el USB está implementado en software. Esto significa que Leonardo no solamente trabaja como el Arduino estándar, sino que puede actuar como HID (Human Interface Device), lo que significa es que podemos hacer que el sistema pretenda ser un ratón, un teclado o tal vez otros dispositivos de entrada, todo dependiendo del software que se escriba. Cuenta con 20 pines de entradas/salidas digitales (de los cuales 7 se pueden utilizar como salidas PWM y 12 como entradas analógicas), un oscilador de cristal de 16 MHz, una conexión micro USB, un conector de alimentación, un puerto ICSP, y un botón de reset.

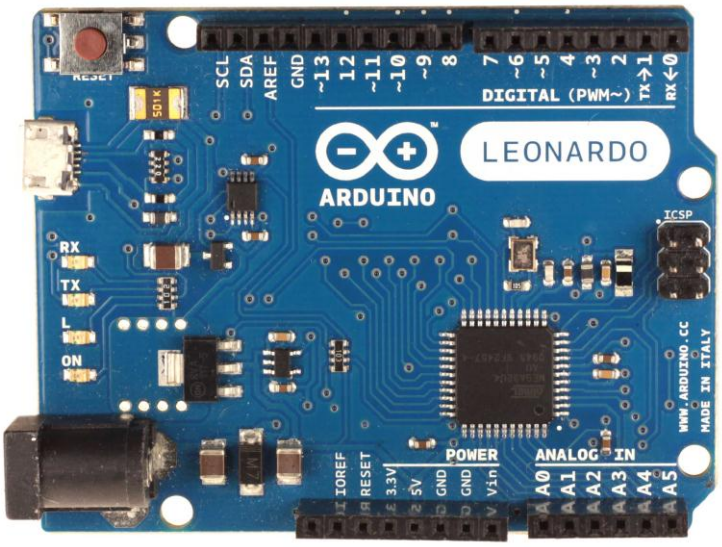

Figura 11. Módulo Arduino Leonardo.

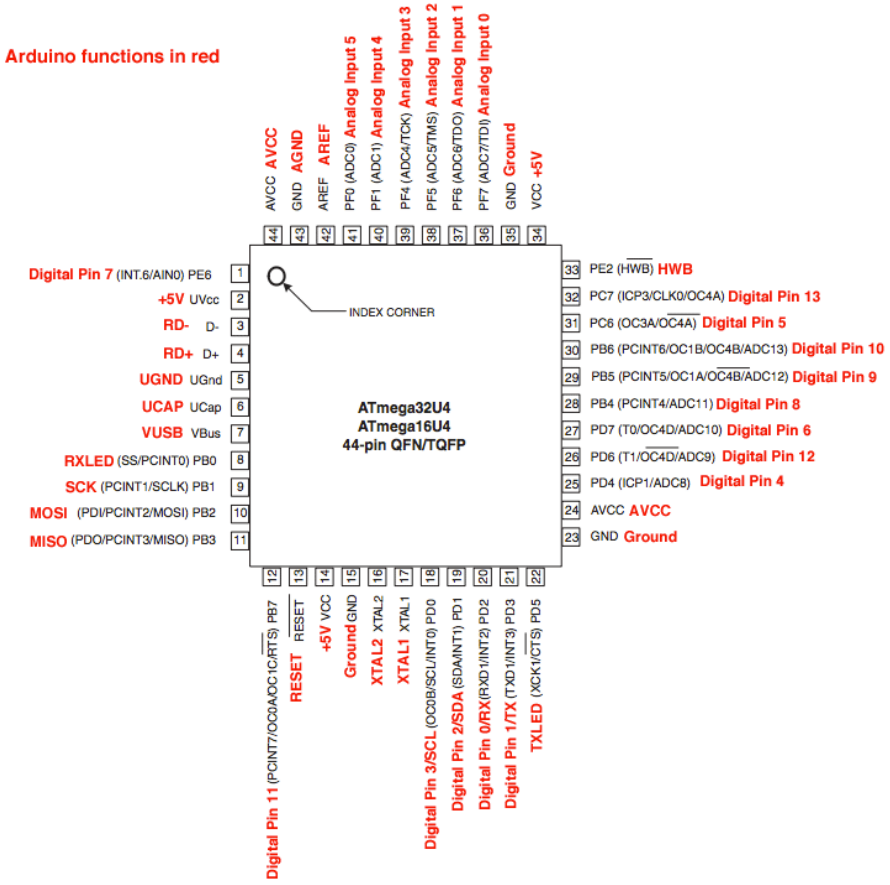

Figura 12. Mapeo entre los pines Arduino Leonardo y los puertos del ATmega32u4.

5.4 Diseño de la aplicación.

La aplicación será implementado en el lenguaje C# en el IDE Visual C# ambos de Microsoft. Esto porque proporcionan las bibliotecas necesarias para realizar la transmisión de datos de forma transparente y flexible además de proporcionar herramientas de diseño que permiten presentar una aplicación visualmente más agradable. Este lenguaje se ejecuta sobre la plataforma *.NET Framework.* Lo cual permite al programa ser ejecutado en cualquier computadora que tenga instalada dicha plataforma, actualmente se están desarrollando equivalentes de esta plataforma para Linux, MacOS. A continuación se mostraran los diagramas para el programa que se ejecutara en la computadora.

Figura 13. Diagrama de la estructura de software.

5.5 Diseño de Programa del Arduino.

En esta parte del diagrama de flujo incluyen las bibliotecas, se inicializan las variables que se usan en el resto del programa, posteriormente se pregunta si hay datos en el puerto si es así primero se identifica si es un dato especificado como comando y se levanta una bandera que puntualiza que tipo de comando se introdujo y se reincorpora al ciclo principal.

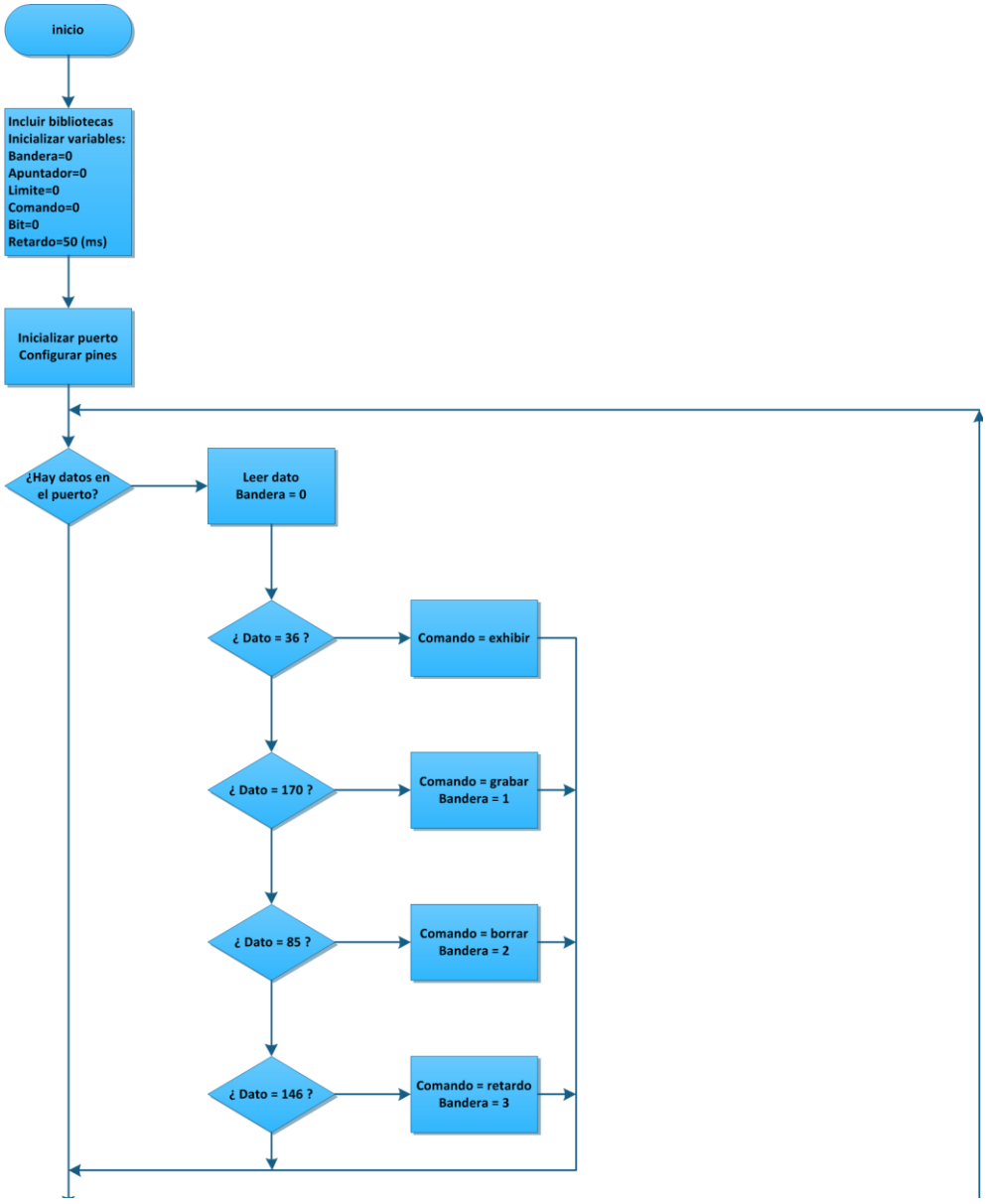

Figura 15. Diagrama de flujo del programa en Arduino. Parte 1

En esta parte del diagrama primero se pregunta si la bandera de exhibir está levantada si es cierto y si el apuntador de memoria no se encuentra en el límite entonces lee el byte de la dirección actual y lo recorre enviando bit por bit a su respectivo pin de salida y espera un tiempo de retardo e incrementa el apuntador y reinicia este proceso.

Cuando avanza al siguiente bucle se pregunta si la bandera de grabar esta levantada, entonces todos los siguientes datos se grabaran en la dirección del apuntador que se incrementara en uno y posteriormente se bajara la bandera de grabar.

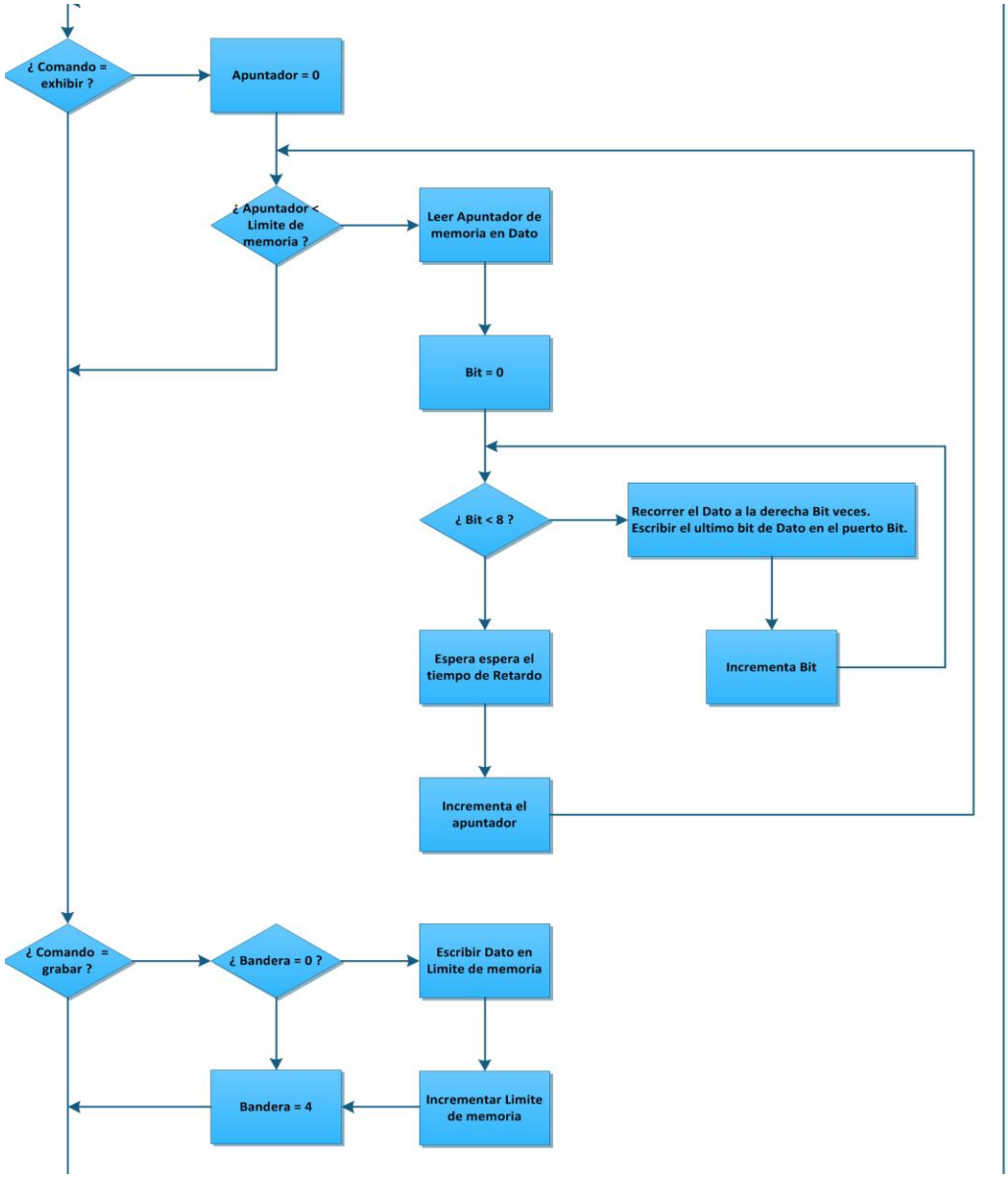

Figura 16. Diagrama de flujo del programa en Arduino. Parte 2

En esta parte del diagrama primero se pregunta si el comando es el de borrar y si su bandera está levantada si es cierto se graban unos en la dirección del apuntador y se decrementa en uno sucesivamente hasta llegar a la primera localidad de memoria, se borra lo que este en los pines de salida y se baja la bandera de borrar.

Después se pregunta si el comando es de retardo y está levantada su bandera, si es cierto se recibe el dato como un número que se multiplica por 50 milisegundos y se guarda en una variable, levanta la bandera de exhibir y sale del bucle.

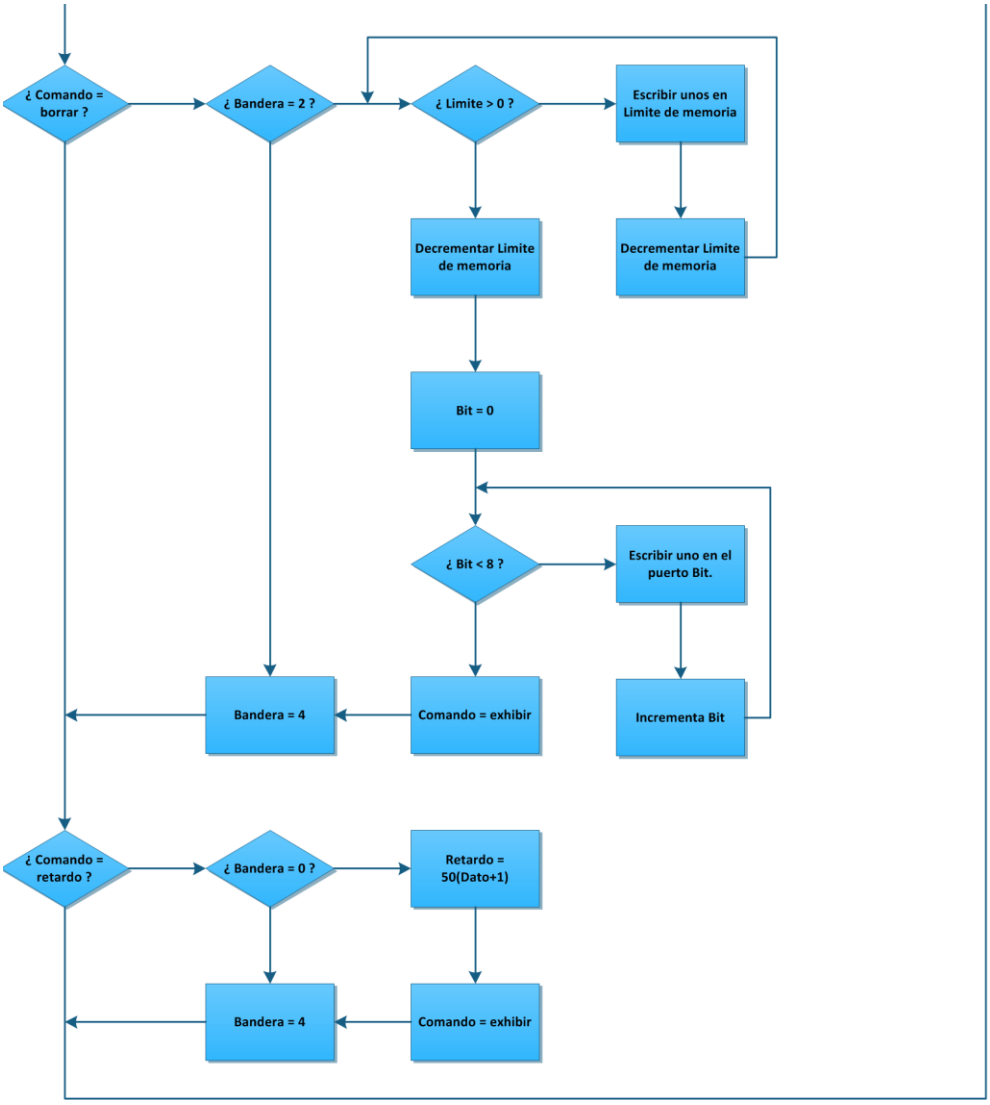

Figura 17. Diagrama de flujo del programa en Arduino. Parte 3

CAPÍTULO 6: DESARROLLO

Durante el desarrollo de este proyecto se fueron solucionados los diferentes problemas, que fueron surgiendo, de forma gradual permitiendo la mejora constante y la apreciación de diversos aspectos acerca de las posibles alternativas que se podrían elegir para incrementos y trabajos posteriores. Para realizar la implementación de este proyecto se dividió en tres tareas que se describen a continuación.

6.1 Descripción de programas.

APLICACIÓN C#.

La aplicación fue llevada a cabo utilizando el IDE Microsoft Visual C# 2010[6]. Empezando una aplicación de formulario de ventanas. Diseñando la interfaz de usuario y posteriormente programando el comportamiento de sus elementos. Permitiendo controlar el puerto al que se conectaría la aplicación, la tasa de baudios, la velocida de exposición, introducir los datos a grabar en la memoria del dispositivo, borrar la memoria del dispositivo, limpiar el área de introducción. Y para transmitir los datos elegí caracteres especiales para diferenciar entre comandos de control y datos de la siguiente manera:

‘*’ = exibir.
‘+’ = agregar.
‘-‘ = borrar.
‘_’ = velocidad.

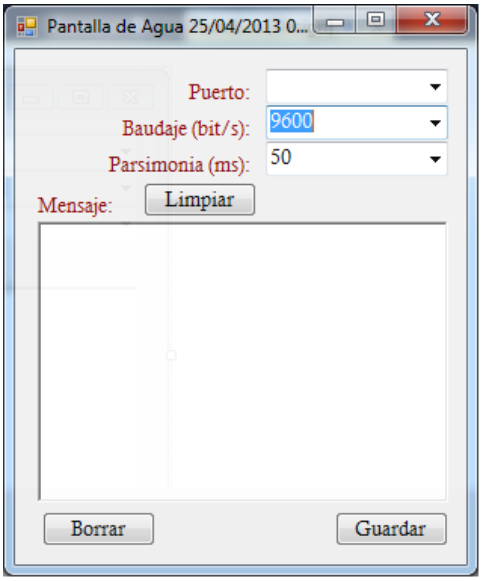

Figura 18. Aplicación C#: interfaz de usuario.

Código de la aplicación C#

```
//librerias que se usaran para formar la ventana
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

//librerias para la usar al puerto
using System.IO.Ports;
using System.Threading;

//inicia la aplicacion
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        //inicializa la ventana
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //titulo y dimensiones de la ventana
            this.Text += DateTime.Now.ToString();
            this.Top = 60;
            this.Left = 90;

            //obtiene la lista de los puertos com conectados
            String[] com = System.IO.Ports.SerialPort.GetPortNames();

            //para cada puerto com conectado se agrega al a lista del combo box
            foreach (string porto in com)   comboBox2.Items.Add(porto);
            comboBox2.Text = com[0];
        }
        //limpia la caja de texto
        private void button1_Click(object sender, EventArgs e)
        {
            richTextBox1.Text = "";
        }

        //boton para enviar informacion al dispocitivo
        private void button2_Click(object sender, EventArgs e)
        {
            //enviar comando para cambiar velocidad
            Pixelica('_');
            //envia el numero de indice al dispocitivo
            byte[] b = { Convert.ToByte(comboBox3.SelectedIndex.ToString()) };
            S_Serial(b);
```

```
            //comando para agregar bytes en la memoria del dispocitivo
            Pixelica('+');

            //lo que esta el la caja de texto enviarlo a el dispocitivo
            Granularidad();

            //comando para exhibir lo que este en la memoria del dispocitivo
            Pixelica('*');
        }

        //boton para borrar
        private void button3_Click(object sender, EventArgs e)
        {
            //comando para borrar toda la memoria del dispocitivo
            Pixelica('-');
        }

        //seccionar todo el texto en caracteres
        public void Granularidad()
        {
            //enviando cada caracter a ser codificado
            foreach (char caracter in richTextBox1.Text) Pixelica(caracter);
        }

  //cada caracter se combierte en conbinacion de bytes para poder ser apreciados
        public void Pixelica(char caracter)
        {
            byte[] fuente ,b = new byte[1];
            switch (caracter)
            {
                    //cada caracter tiene una codificacion para ser representado
                case '0': fuente = new byte[] { 126, 131, 129, 193, 126 }; break;
                case '1': fuente = new byte[] { 64, 255 }; break;
                case '2': fuente = new byte[] { 99, 133, 137, 145, 97 }; break;
                case '3': fuente = new byte[] { 130, 129, 145, 177, 206 }; break;
                case '4': fuente = new byte[] { 24, 40, 8, 255, 8 }; break;
                case '5': fuente = new byte[] { 242, 145, 145, 145, 142 }; break;
                case '6': fuente = new byte[] { 126, 137, 145, 145, 78 }; break;
                case '7': fuente = new byte[] { 128, 143, 144, 160, 192 }; break;
                case '8': fuente = new byte[] { 110, 145, 145, 145, 110 }; break;
                case '9': fuente = new byte[] { 96, 145, 145, 145, 126 }; break;
                case 'A': fuente = new byte[] { 127, 136, 136, 136, 127 }; break;
                case 'B': fuente = new byte[] { 255, 145, 145, 145, 110 }; break;
                case 'C': fuente = new byte[] { 126, 129, 129, 129, 98 }; break;
                case 'D': fuente = new byte[] { 255, 129, 129, 129, 126 }; break;
                case 'E': fuente = new byte[] { 255, 145, 145, 145, 129 }; break;
                case 'F': fuente = new byte[] { 255, 144, 144, 144, 128 }; break;
                case 'G': fuente = new byte[] { 126, 129, 129, 137, 78 }; break;
                case 'H': fuente = new byte[] { 255, 16, 16, 16, 255 }; break;
                case 'I': fuente = new byte[] { 129, 129, 255, 129, 129 }; break;
                case 'J': fuente = new byte[] { 6, 1, 129, 129, 254 }; break;
                case 'K': fuente = new byte[] { 255, 24, 36, 66, 129 }; break;
                case 'L': fuente = new byte[] { 255, 1, 1, 1, 1 }; break;
                case 'M': fuente = new byte[] { 255, 64, 32, 64, 255 }; break;
                case 'N': fuente = new byte[] { 255, 96, 48, 24, 255 }; break;
```

```
                case 'O': fuente = new byte[] { 126, 129, 129, 129, 126 }; break;
                case 'P': fuente = new byte[] { 255, 136, 136, 136, 112 }; break;
                case 'Q': fuente = new byte[] { 126, 129, 133, 131, 127 }; break;
                case 'R': fuente = new byte[] { 255, 136, 140, 138, 113 }; break;
                case 'S': fuente = new byte[] { 98, 145, 145, 145, 78 }; break;
                case 'T': fuente = new byte[] { 128, 128, 255, 128, 128 }; break;
                case 'U': fuente = new byte[] { 254, 1, 1, 1, 254 }; break;
                case 'V': fuente = new byte[] { 252, 2, 1, 2, 252 }; break;
                case 'W': fuente = new byte[] { 255, 2, 4, 2, 255 }; break;
                case 'X': fuente = new byte[] { 199, 40, 16, 40, 199 }; break;
                case 'Y': fuente = new byte[] { 240, 8, 15, 8, 240 }; break;
                case 'Z': fuente = new byte[] { 135, 137, 145, 161, 193 }; break;
                case 'a': fuente = new byte[] { 70, 137, 137, 138, 127 }; break;
                case 'b': fuente = new byte[] { 255, 10, 9, 9, 6 }; break;
                case 'c': fuente = new byte[] { 62, 65, 65, 65, 34 }; break;
                case 'd': fuente = new byte[] { 6, 9, 9, 10, 255 }; break;
                case 'e': fuente = new byte[] { 62, 73, 73, 73, 50 }; break;
                case 'f': fuente = new byte[] { 63, 72, 72, 64, 32 }; break;
                case 'g': fuente = new byte[] { 98, 145, 145, 145, 254 }; break;
                case 'h': fuente = new byte[] { 255, 16, 32, 32, 31 }; break;
                case 'i': fuente = new byte[] { 191, 1 }; break;
                case 'j': fuente = new byte[] { 6, 1, 1, 1, 190 }; break;
                case 'k': fuente = new byte[] { 127, 8, 12, 18, 33 }; break;
                case 'l': fuente = new byte[] { 255, 1 }; break;
                case 'm': fuente = new byte[] { 127, 32, 31, 32, 31 }; break;
                case 'n': fuente = new byte[] { 127, 16, 32, 32, 63 }; break;
                case 'o': fuente = new byte[] { 14, 17, 17, 17, 14 }; break;
                case 'p': fuente = new byte[] { 255, 72, 72, 72, 48 }; break;
                case 'q': fuente = new byte[] { 48, 72, 74, 74, 255 }; break;
                case 'r': fuente = new byte[] { 191, 1, 1 }; break;
                case 's': fuente = new byte[] { 50, 73, 73, 73, 38 }; break;
                case 't': fuente = new byte[] { 32, 32, 255, 32, 32 }; break;
                case 'u': fuente = new byte[] { 126, 1, 1, 2, 127 }; break;
                case 'v': fuente = new byte[] { 120, 4, 2, 1, 126 }; break;
                case 'w': fuente = new byte[] { 62, 1, 62, 1, 62 }; break;
                case 'x': fuente = new byte[] { 35, 20, 8, 20, 35 }; break;
                case 'y': fuente = new byte[] { 224, 17, 17, 33, 254 }; break;
                case 'z': fuente = new byte[] { 35, 39, 45, 57, 49 }; break;

                case '!': fuente = new byte[] { 1}; break;//
                case '"': fuente = new byte[] { 2}; break;//
                case '#': fuente = new byte[] { 4 }; break;//
                case '$': fuente = new byte[] { 8 }; break;//
                case '%': fuente = new byte[] { 16 }; break;//
                case '&': fuente = new byte[] { 32 }; break;//
                case '/': fuente = new byte[] { 64 }; break;//
                case '(': fuente = new byte[] { 128 }; break;//
case '?': fuente = new byte[] { 128, 192, 96, 48, 24, 12, 6, 3 }; break;//zig
case '¡': fuente = new byte[] { 1, 3, 6, 12, 24, 48, 96, 192 }; break;//zag
case '@': fuente = new byte[] { 16, 56, 124, 254, 127, 62, 28, 8 }; break;//diamante
case ',': fuente = new byte[] { 85, 170 }; break;//01010101,10101010
case '.': fuente = new byte[] { 240, 15 }; break;//11110000,00000000,00001111
case ';': fuente = new byte[] { 17, 34, 68, 136 }; break;//00010001
case ':': fuente = new byte[] { 129, 66, 36, 24}; break;//A
```

```
case '=': fuente = new byte[] { 1, 1, 2, 12, 48, 64, 128, 128, 64, 48, 48, 12, 2 };
break;//A

                case '*': fuente = new byte[] { 253 }; break;//mostrar
                case '+': fuente = new byte[] { 251 }; break;//grabar
                case '-': fuente = new byte[] { 250 }; break;//limpiar
                case '_': fuente = new byte[] { 249 }; break;//velocidad-_
                default: fuente = new byte[] { 0 }; break;
            }

            //para cada linea en la que se divide un caracter
            foreach(byte linea in fuente)
            {
                //niega el byte actual
                b[0]=(byte)(linea ^ 0xff);

                //un byte a ser enviado
                S_Serial(b);
            }
        }

        //transmicion de la informacion
        public void S_Serial(byte[] b)
        {
            //configura el puerto con sus parametros
            SerialPort puerto = new SerialPort(comboBox2.SelectedItem.ToString(),
Convert.ToInt32(comboBox1.SelectedItem.ToString()));
            if (!puerto.IsOpen)
            {
                puerto.Open();//abreel puerto
                puerto.Write(b, 0, 1);//envia la informacio
                puerto.Close();//cierra el puerto
            }
        }
    }
}
```

Cuadro de texto1: Programa de interfaz de usuario

ARDUINO[7].

El programa que se ejecuta en el módulo Arduino fue desarrollada gradual mente y mejorado para realizar las funciones que el proyecto necesita. Las tareas que realizan son: inicialización de variables, realizar la conexión con la computadora, especificar los pines de salida, esperar el evento de dato transmitido, distinguir si es un comando u es un carácter, mostrar a travez de los pines de salida los datos guardados en le memoria del dispositivo, guardar los datos recibidos en la memoria del dispositivo, borrar la memoria del dispositivo, cambiar la velocidad de exhibición de los datos como se muestra en la figura.

```
#include <EEPROM.h>
byte outPin[] = {0, 1 ,2, 3, 4, 5, 6, 7}, i=0,j=0,k=0, b,comm=0,flg=0;
// 0, 1 ,2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
int reta=100;
void setup(){

  //baudaje de 9600
  Serial.begin(9600);

  //continua hasta que se realice la conexion
  while (!Serial) { }

  //configura los pines como salida
  for ( i = 0; i < 8; i++)    pinMode(outPin[i], OUTPUT);
}

void loop(){
  //bucle permanente
    while(true){

      //si hay un dato en el puerto
      if(Serial.available()){

        //lee el puerto
        b=Serial.read();
        flg=0;

        //comandos
        if (b==2)       {comm=0;}//exhibir
        else if(b==4)    {comm=1;flg=1;}//grabar
        else if(b==5)   {comm=2;flg=2;}//borrar
        else if(b==6)   {comm=3;flg=3;}//retardo
      }

      //exhibir
      if (comm==0){

        //recorre la memoria usada
          for(j=0;j<k;j++){
              b = EEPROM.read(j);

              //enciende pin por pin
              for( i=0; i<8; i++)
                  digitalWrite(outPin[i], b>>i&1);
```

```
            delay(reta);
        }
    }

    //agregar datos a la memoria
    else if (comm==1){
        if(flg==0){

          //grabar datos en la memoria
            EEPROM.write(k++, b);
        }
        flg=4;
    }

    //borrar memoria
    else if (comm==2){
        if(flg==2){

          //desde la ultima direccion utilizada
            for ( ; k >0; k--)
                EEPROM.write(k, 255);

            //la primera direccion de memoria
            EEPROM.write(k, 255);

            //lo que se muestra en los pines
            for( i=0; i<8; i++)  digitalWrite(outPin[i],HIGH);
            comm=0;
        }
        flg=4;
    }

    //cambia el retardo en la exhibicion
    else if (comm==3){
        if(flg==0){
            reta=(1+b)*50;
            comm=0;
        }
         flg==4;
    }
  }
}
```

Figura 19. Programa del módulo Arduino.

6.2 Descripción de circuitos.

Etapa de potencia.- Se utilizan dos circuitos, el circuito de potencia integrado principalmente por el Triac 2n6344 y el optoacoplador Moc3011; y el circuito que es propiamente el módulo Arduino.

Se realizó la siguiente configuración del circuito para la etapa de potencia que permitiera su uniformidad con ocho conjuntos de circuitos para las ocho elecrovalvulas.

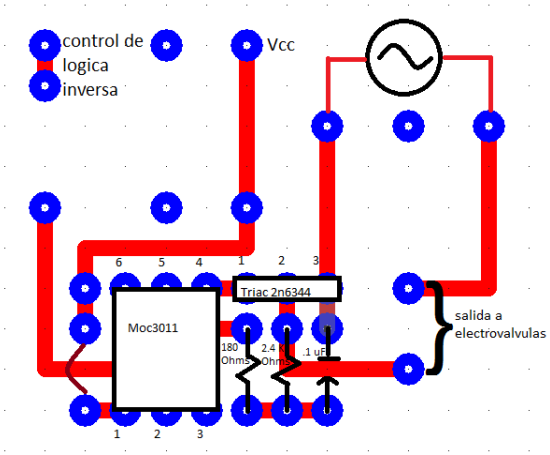

Figura 20. Configuración del circuito de control.

Quedando el circuito impreso de la siguiente manera.

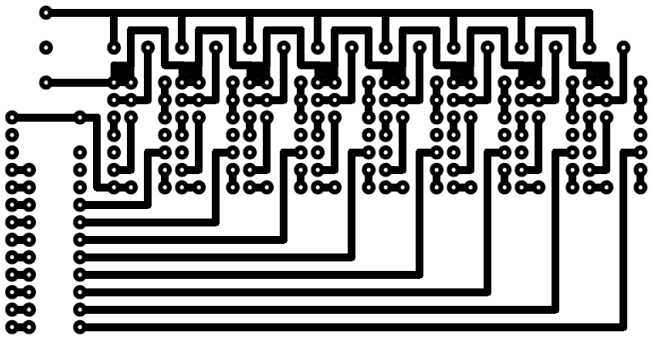

Figura 21. Circuito impreso de la etapa de potencia.

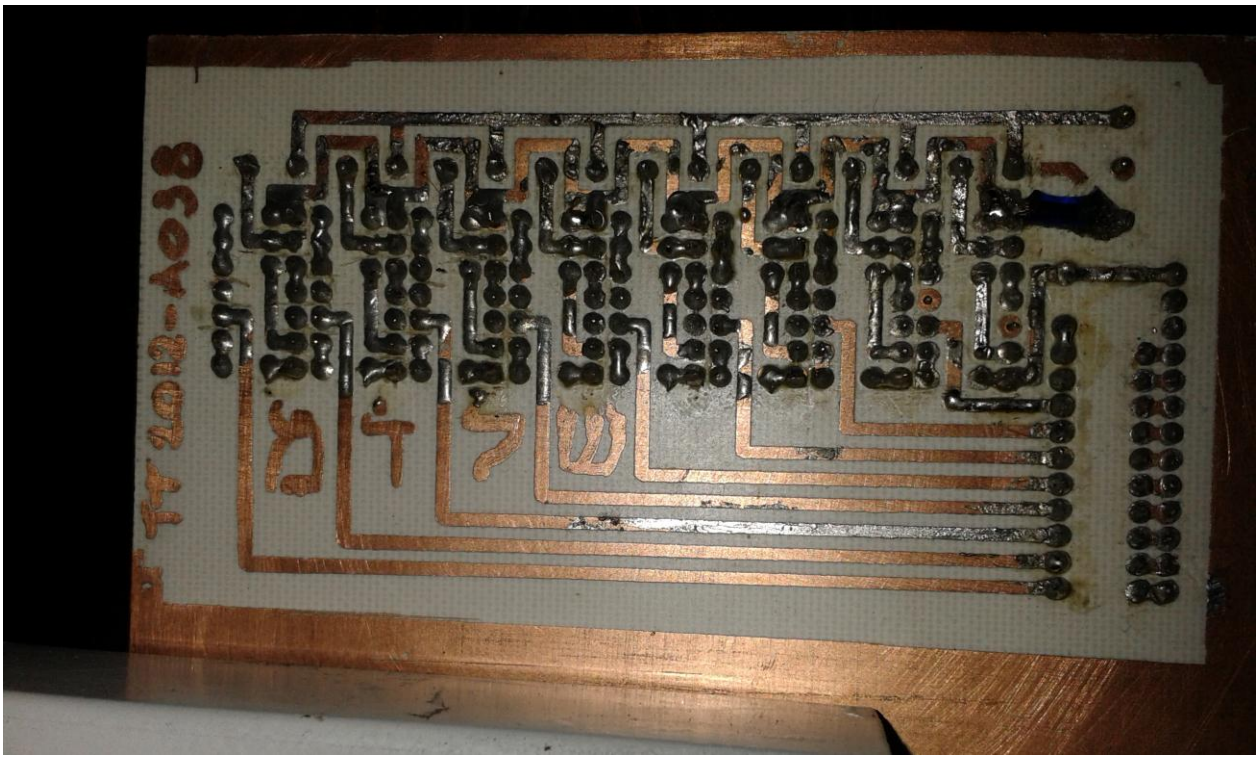

Figura 22. Circuito impreso, soldado

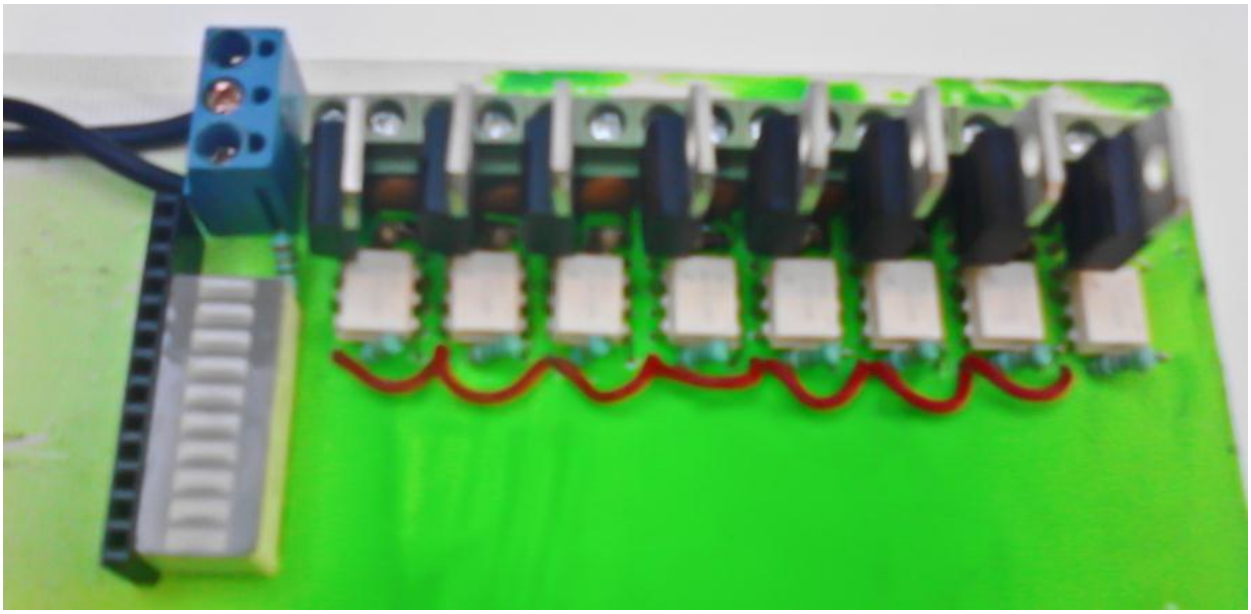

Figura 23. Circuito terminado.

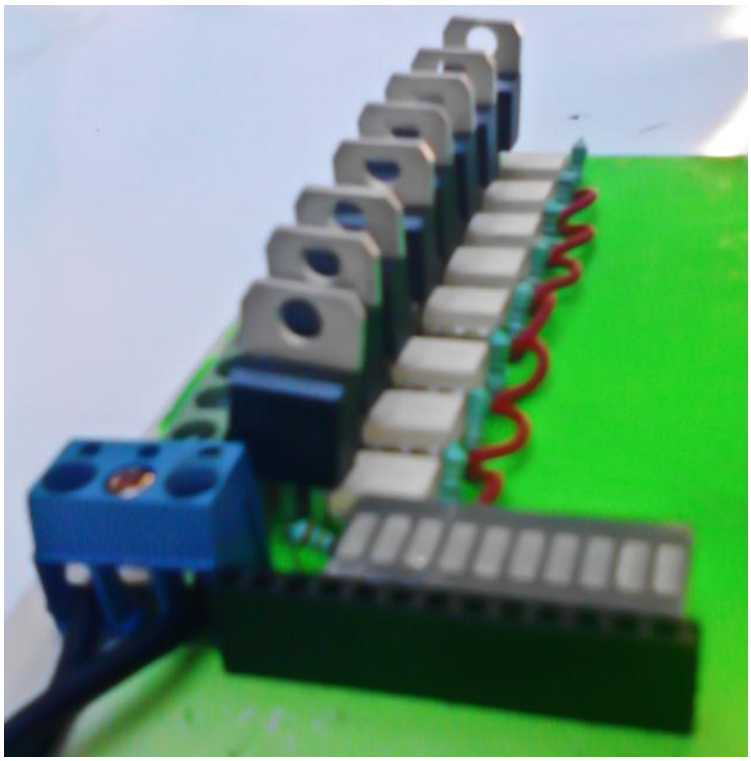

Figura 24. Circuito terminado en perspectiva.

Etapa de control.- para la etapa de control se utilizara el módulo Arduino, en el cual se hicieron pruebas de recepción de datos, de tiempos de exhibición, de escritura, lectura de memoria del dispositivo.

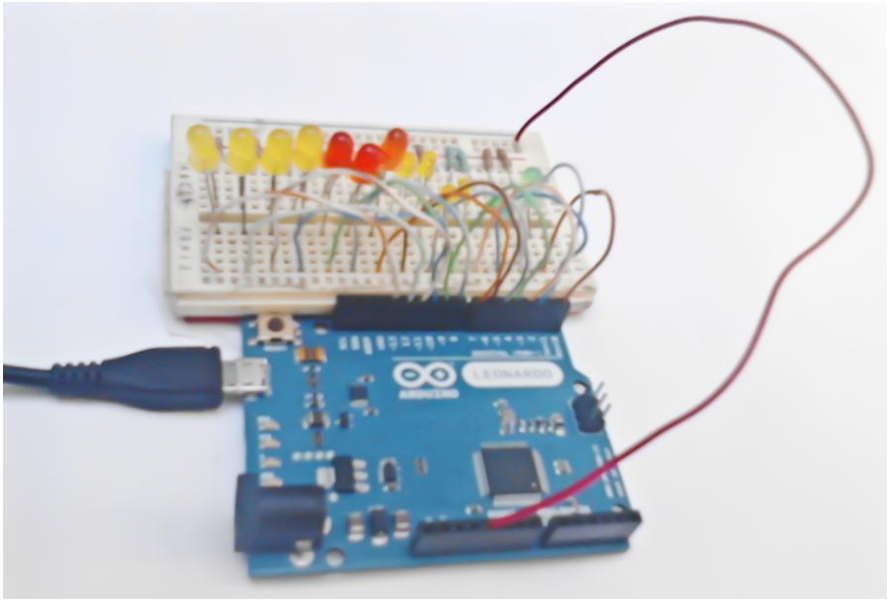

Figura 25. Pruebas con el módulo Arduino.

6.3 Descripción de mecanismo.

A través de varias pruebas se construyó la estructura de distribución de agua y se ensamblo las electroválvulas a las cuales se comprobó que no tuvieran fugas y trabajaran correctamente.

Se inició probando con diferentes piezas de tubería PVC

Figura 26. Piezas de tubería de PVC.

De esta manera se llegó construir la estructura que satisface las necesidades de distribución de agua para las electroválvulas. La cual consta de 7 Tees lisas de ¾”, un codo liso de ¾”, ocho uniones liso a rosca hembra de ¾”, una unión liso a rosca macho de ¾”, ocho adaptadores rosca a manguera de ¾”, diecisiete abracaderas de ¾”, ocho adaptadores de manguera de ¾” a rosca 5/8”, tres metros de manguera transparente de ¾”

Figura 27. Estructura de distribución de agua.

Figura 28. Piezas de tubería ensambladas a detalle.

Posteriormente se montaron las electroválvulas en un armazón metálico con la separación mínima entre cada una de ellas. Para evitar la deformación de las figuras a la hora de hacer las pruebas.

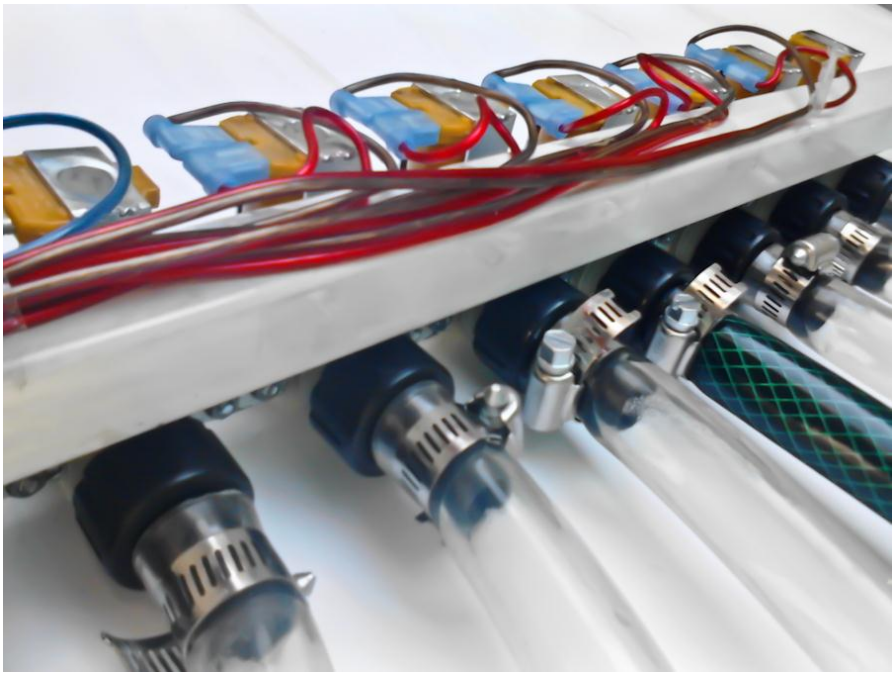

Figura 29. Armazón metálico.

Figura 30. Separación mínima entre electroválvulas.

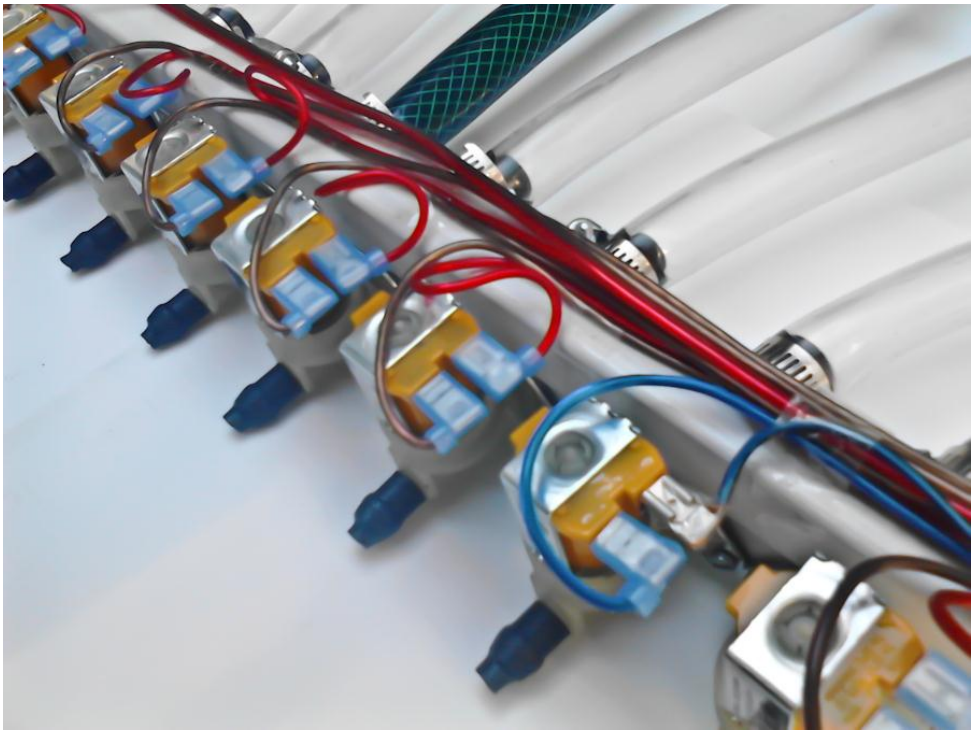

Figura 31. Electroválvulas y cableado.

Figura 32. Estructura completa (vista frontal.)

Figura 33. Estructura armada (vista trasera.)

6.4 Fases de prueba.

Se procedió con pruebas acerca de cómo se forman las imágenes de agua. De lo cual se apreció que debido a la alta presión y a la salida turbia del flujo de agua se tienen dificultades para distinguir la forma de los caracteres pero es más fácil distinguir figuras y símbolos más simples

Se realizaron pruebas de control, donde se pudo apreciar el correcto acoplamiento de las válvulas, del controlador de la etapa de potencia y el módulo de control. Se hicieron pruebas a cada pin de salida y en diversos conjuntos siendo siempre exitosos los intentos de dichas pruebas

Figura 34: pruebas de control.

Se realizaron pruebas de velocidad de reacción de las válvulas, de donde se apreció que con la disminución de la cantidad de fluido por medio de las boquillas re redujo la velocidad de cierre de las válvulas que fue de 500 milisegundos a 200 milisegundos

Figura 35: pruebas de velocidad.

CONCLUSIONES

- Se considera que se realizo el cumplimiento de los objetivos al cien por ciento ya que se diseñó e implemento el sistema prototipo de pantalla de agua y en efecto es capaz de representar figuras introducidos a través de un programa en la computadora.
- Se cumplieron los objetivos específicos:
    - Análisis y selección del controlador adecuado, que cubran las necesidades del sistema.
    - Análisis y selección del hardware que controlan el flujo de líquido.
    - Análisis de las herramientas de software, lenguajes de programación.
    - Desarrollo e implementación del sistema de software
    - Desarrollo e implementación de la interfaz USB PC-dispositivo de control.
    - Desarrollo e implementación de dispositivo de control.
    - Desarrollo e implementación de la parte mecánica del control del fluido
    - Integración y pruebas al sistema.
    - Generación de documentación: manual de usuario, manual técnico y especificaciones
- Se resolvieron los problemas de comunicación PC-modulo usando la biblioteca serial del Arduino
- Se resolvieron los problemas de comportamiento de la etapa de control al recibir la información implementando banderas en el programa.
- Se resolvieron los problemas de tiempo de apertura y cierre de las válvulas al agregarles un aboquilla.
- Se resolvió el problema de calentamiento de la etapa de potencia con una nueva configuración del dispositivos electrónicos
- Este proyecto aporta un prototipo que puede ser mejorado y se le pueden añadir más funcionalidades.
- Este proyecto aporta el diseño y programas que pueden ser tomados como referencia para posteriores proyectos

REFERENCIAS

1. Julius Pop. [fecha de consulta: 24 de mayo del 2013] Página disponible en: http://en.wikipedia.org/wiki/Julius_Popp.
2. Julius Pop. [fecha de consulta: 24 de mayo del 2013] Página disponible en: http://www.cccb.org/es/autor-julius_popp-12703
3. KOEI.co.ltd. [fecha de consulta: 24 de mayo del 2013] Página disponible en: http://www.koeiaquatec.co.jp/
4. ON Semiconductor. [fecha de consulta: 24 de mayo del 2013] Página disponible en: http://www.onsemi.com/pub_link/Collateral/2N6344-D.PDF
5. Motorola. [fecha de consulta: 24 de mayo del 2013] Página disponible en: http://www.datasheetcatalog.org/datasheet/motorola/MOC3010.pdf
6. Massimo Banzi, David Cuartielles. [fecha de consulta: 24 de mayo del 2013] Página disponible en: http://es.wikipedia.org/wiki/Arduino
7. Atmel corporation. [fecha de consulta: 24 de mayo del 2013] Página disponible en: http://www.atmel.com/Images/7766s.pdf