

PROYECTO FINAL
EQUIPO 1
ALDO BARRIOS GARCIA
CARLOS MELENDEZ MEJIA
DAVID PEREZ XOLOCOTZI
ERICK ARTURO VAZQUEZ MACARIO

Introducción

Objetivo

El objetivo del cual nace este proyecto es el de optimizar una tarea de la vida real por medio de la implementación en conjunto de distintos componentes de Hardware y de Software, funcionando por medio de un microcontrolador PIC16F887, programado en C y que a su vez éste nos permite tener comunicación serial con una computadora para mostrar el estado de nuestro circuito en tiempo real, mediante una interfaz de usuario diseñada para optimizar la comunicación y de la misma manera ajustar el funcionamiento de circuito en tiempo real de acuerdo a nuestras necesidades.

La tarea que escogimos optimizar como equipo es la del enfriamiento de una casa por medio de la medición de la temperatura ambiental en tiempo real y que de la misma manera el sistema se encargará del enfriamiento del lugar donde se implementará este circuito.

Funcionalidad

La funcionalidad del circuito es la siguiente, se tienen dos sensores de temperatura LM35, los cuales funcionan como nuestras entradas analógicas, estos sensores se encuentran midiendo en tiempo real la temperatura y dando un voltaje en una señal análoga.

Después estas señales entran por el conversor Análogo/Digital de nuestro microcontrolador PIC16F887. Este microcontrolador desempeña diversas funciones en el circuito.

La primera función es la conversión del ADC, el cual regresa un valor digital para que podamos manipularlo de acuerdo con nuestras necesidades, la PIC realiza esta operación por cada uno de los sensores. Después por medio de una conversión matemática, dichos valores de nuestro ADC son convertidos al valor de la temperatura en grados Celsius.

La segunda función que desempeña es la de la configuración y posteriormente la escritura de nuestra pantalla LCD, en dicha pantalla se desplegarán cuatro valores en tiempo real, los primeros dos corresponden a la temperatura ya convertida registrada por nuestros sensores en tiempo real. Mientras que los otros dos valores corresponden al estado de nuestros motores, es decir si se encuentran encendidos o apagados.

La tercera función es manejada en conjunto con un puente H L293D y esa es la del manejo de nuestros actuadores, los cuales son 2 motores de 5 volts que se encargan del enfriamiento por medio de 2 ventiladores. El puente H se encarga de administrar el funcionamiento de estos motores y de su alimentación.

La cuarta función que desempeña es la comunicación serial con nuestra computadora por medio del puerto serial del PIC y nuestro convertidor USB Serial TTL CP2102.

La comunicación entre nuestra PIC y la computadora es bidireccional, lo que significa que la PIC puede enviar información a la computadora y viceversa. Esta tarea se realiza en conjunto con la interfaz de usuario que diseñamos para que el usuario pueda manipular la PIC con mayor facilidad.

Dicha interfaz de usuario cuenta con diversas opciones como la selección, conexión y desconexión del puerto de comunicación de nuestra computadora que más nos favorezca. Así como la lectura en tiempo real del estado de nuestra temperatura y nuestros motores.

La última función que desempeña la PIC, la realiza en conjunto con nuestra interfaz de usuario. Se trata de la selección de los diversos modos de operación de nuestro circuito, por defecto el circuito opera en modo automático. En el modo automático, la PIC se encarga de encender y apagar los ventiladores de acuerdo con el valor de la temperatura registrado por los sensores, es decir, al alcanzar una temperatura elevada, los motores se accionan para comenzar el enfriamiento del lugar y al nivelarse la temperatura, los ventiladores se apagarán nuevamente.

De la misma manera, contamos con un modo manual, al que el usuario podrá acceder en cualquier momento por medio de nuestra interfaz, en dicho modo, el usuario podrá determinar qué ventiladores encender o apagar a voluntad, independientemente de los valores de temperatura que registran los sensores.

Alcance

Al tratarse de un prototipo, nuestro circuito solo funciona a pequeña escala, nosotros decidimos ponerlo en práctica en una pequeña maqueta, con la finalidad de simular la implementación a gran escala.

De igual manera, los componentes y su distribución sólo nos permiten operar a cierta distancia de la PC, además de que este aún requiere una conexión física al puerto de comunicaciones de dicho elemento.

El circuito requiere de una alimentación constante de 5 volts para que desempeñe un funcionamiento óptimo, sobre todo por la alimentación que requieren los motores, que se podría decir, son los componentes más demandantes de nuestro circuito.

Diagrama de Bloques del Sistema

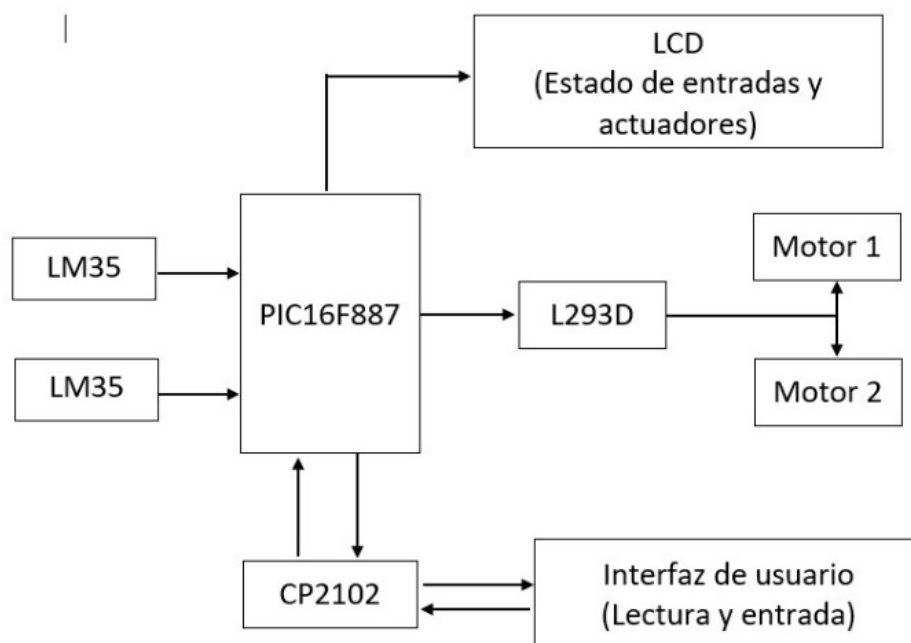
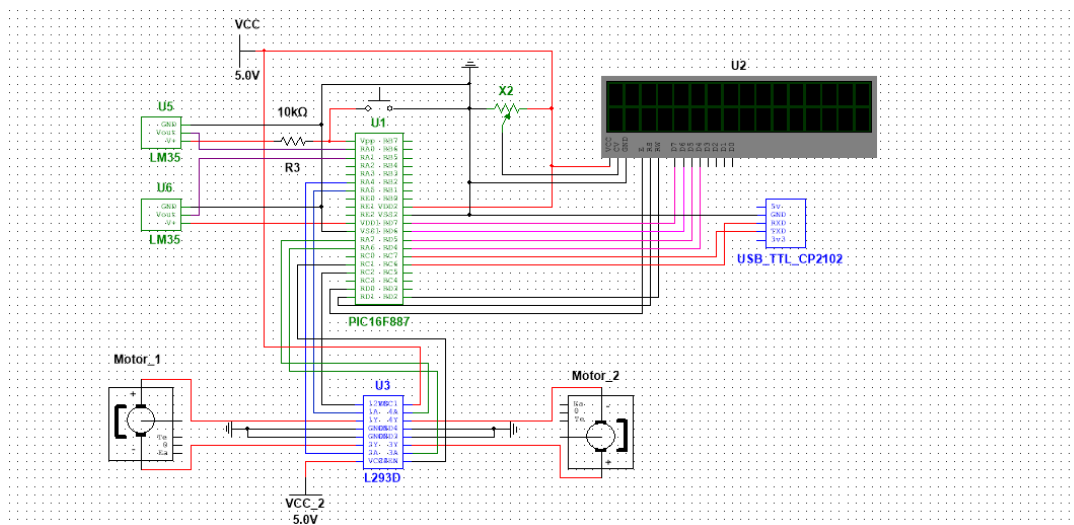


Diagrama eléctrico del Circuito



Código Fuente del Firmware Comentado

```
#include <16f887.h>
#define ADC=10
#define fuses INTRC_IO, NOWDT, PROTECT, NOLVP, MCLR, NOBROWNOUT
#define use delay(INTERNAL=4000000)
#include <lcd.c>
#include <math.h>
#include <stdbool.h>
//Configuración para enviar y recibir mensaje en terminal serial
#define RS232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, timeout=100)

char automatico(int16 temp1, int16 temp2){
    //Variables de temperatura , modo y ciclo
    bool x=true;
    char a;
    float temperatura1, temperatura2;

    //Se limpia pantalla y se imprime el modo
    printf(lcd_putc, "\f");
    lcd_gotoxy(1,1);
    printf(lcd_putc, "MODO AUTOM.");
    delay_ms(1500);

    //Inicio de bucle automatico
    while(x){

        //Lectura de los canales analogicos y almacenamiento
        set_adc_channel(0);
        temp1=read_adc();
        set_adc_channel(1);
        temp2=read_adc();
        delay_ms(1000);
        printf(lcd_putc, "\f");

        //Conversion de ADC a temperatura
        temperatura1=(((float)temp1)*5/1023)*100;
        temperatura2=(((float)temp2)*5/1023)*100;

        if(temperatura1>=30.0){ //Rango de encendido motor 1
            //Se activa el motor y se imprime el estado
            output_low(PIN_A4);
            output_high(PIN_A5);
            set_pwm1_duty(1000);
        }
    }
}
```

```

        lcd_gotoxy(1,1);
        printf(lcd_putc,"T1:%fC V1:E", temperatura1);
        printf("T1:%fC V1:E M:A \n\r", temperatura1);
    }
    else if(temperatura1<30.0){    //Rango de apagado motor 1
        //Se apaga el motor y se imprime el estado
        output_low(PIN_A4);
        output_low(PIN_A5);
        set_pwm1_duty(0);    //Fijado de ciclo de trabajo en 0%
        lcd_gotoxy(1,1);
        printf(lcd_putc,"T1:%fC V1:A", temperatura1);

        printf("T1:%fC V1:A M:A\n\r", temperatura1);
    }

    if(temperatura2>=30.0){    //Rango de encendido motor 2
        //Se activa el motor y se imprime el estado
        output_low(PIN_A6);
        output_high(PIN_A7);
        set_pwm2_duty(1000);

        lcd_gotoxy(1,2);
        printf(lcd_putc,"T2:%fC V2:E", temperatura2);
        printf("T2:%fC V2:E M:A\n\r", temperatura2);
    }
    else if(temperatura2<30.0){    //Rango de apagado motor 2
        //Se apaga el motor y se imprime el estado
        output_low(PIN_A6);
        output_low(PIN_A7);
        set_pwm2_duty(0);

        lcd_gotoxy(1,2);
        printf(lcd_putc,"T2:%fC V2:A", temperatura2);

        printf("T2:%fC V2:A M:A\n\r", temperatura2);
    }

    //Lectura de entrada para cambio de modo
    a=getch();
    //Caracter para cambio de modo a manual
    if(a=='M'){

```

```

        //Regresa caracter M
        return a;
        x=false;
    }
}

char manual(int16 temp1, int16 temp2){
    //Variables de temperatura , modos y ciclo
    bool x=true;
    char op;
    char m1='A';
    char m2='A';
    float temperatura1, temperatura2;

    //Se apagan los motores por defecto
    output_low(PIN_A4);
    output_low(PIN_A5);

    output_low(PIN_A6);
    output_low(PIN_A7);

    //Se imprime el modo actual
    printf(lcd_putc, "\f");
    lcd_gotoxy(1,1);
    printf(lcd_putc, "MODO MANUAL");
    delay_ms(1500);

    while(x){
        //Lectura de ADC de los dos sensores
        set_adc_channel(0);
        temp1=read_adc();
        set_adc_channel(1);
        temp2=read_adc();
        //Transformacion a temperatura
        temperatura1=(((float)temp1)*5/1023)*100;
        temperatura2=(((float)temp2)*5/1023)*100;

        //Se imprime en la terminal y en la LCD el estado de las variables
        printf(lcd_putc, "\f");
        lcd_gotoxy(1,1);
        printf(lcd_putc, "T1:%fC V1:%c", temperatura1, m1);
        lcd_gotoxy(1,2);
        printf(lcd_putc, "T2:%fC V2:%c", temperatura2, m2);
    }
}

```

```

printf("T1:%fC  V1:%c  M:M  \n\r", temperatura1, m1);
printf("T2:%fC  V2:%c  M:M\n\r", temperatura2, m2);

delay_ms(1000);

//Se lee caracter para poder saber que elementos encender o como
operarlos
op=getch();

switch(op){
//Caso 1 todo apagado
case '1':

output_low(PIN_A4);
output_low(PIN_A5);
set_pwm1_duty(0);
output_low(PIN_A6);
output_low(PIN_A7);
set_pwm2_duty(0);
m1='A';
m2='A';

break;

//Caso 2 se enciende solamente el motor 1
case '2':

output_low(PIN_A4);
output_high(PIN_A5);
set_pwm1_duty(1020);
output_low(PIN_A6);
output_low(PIN_A7);
set_pwm2_duty(0);
m1='E';
m2='A';
break;

//Caso 3 se enciende solamente el motor 2
case '3':
output_low(PIN_A4);
output_low(PIN_A5);
set_pwm1_duty(0);
output_low(PIN_A6);
output_high(PIN_A7);
set_pwm2_duty(1020);

```



```

    m1='A';
    m2='E';
    break;

    //Caso 4 se encienden ambos motores
    case '4':
        output_low(PIN_A4);
        output_high(PIN_A5);
        set_pwm1_duty(1020);
        output_low(PIN_A6);
        output_high(PIN_A7);
        set_pwm2_duty(1020);
        m1='E';
        m2='E';
        break;

    //Caso A se encarga de cambiar al modo automatico
    case 'A':
        //Actualiza y retorna variables
        op='A';
        return op;
        x=false;
        break;
    //Si se da otro caso, no se hace nada
    default:
        break;
    }
}

}

void main()
{
    //Variable de ADC y caracter de protocolo
    int16 temp1, temp2;
    char modo = 'A';

    //char protocolo;
    lcd_init();

    setup_ccp1(CCP_PWM);    //Se configura ccp1 para usar pwm
    setup_ccp2(CCP_PWM);    //Se configura ccp2 para usar pwm

    setup_timer_2(T2_DIV_BY_16,1023,1); // Timer a usar
    set_pwm1_duty(0);    //Fijado de ciclo de trabajo en 0%
    set_pwm2_duty(0);

```

```

//Puerto analogico
setup_adc_ports(sAN0, sAN1);
setup_adc(ADC_CLOCK_INTERNAL);

while(TRUE)
{
    switch(modos){

        //Modo automatico
        case 'A':
            printf("-----MODO AUTOMATICO-----\n\r");
            modos=automatico(temp1,temp2);
            break;

        //Modo manual
        case 'M':
            printf("-----MODO MANUAL-----\n\r");
            modos>manual(temp1, temp2);
            break;

        //Nunca entra aqui
        default:

            break;
    }
}
}

```

Código Fuente de la Interfaz Comentado

```
namespace MicroContr1
{
    partial class Frame
    {
        /// <summary>
        /// Variable del diseñador necesaria.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Limpiar los recursos que se estén usando.
        /// </summary>
        /// <param name="disposing">true si los recursos administrados se
        /// deben desechar; false en caso contrario.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Código generado por el Diseñador de Windows Forms

        /// <summary>
        /// Método necesario para admitir el Diseñador. No se puede
        /// modificar
        /// el contenido de este método con el editor de código.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new
            System.ComponentModel.ComponentResourceManager(typeof(Frame));
            this.botonConectar = new System.Windows.Forms.Button();
            this.botonDesconectar = new System.Windows.Forms.Button();
            this.cbPuertos = new System.Windows.Forms.ComboBox();
            this.botonAutomatico = new System.Windows.Forms.Button();
            this.botonManual = new System.Windows.Forms.Button();
            this.boton_pt1 = new System.Windows.Forms.Button();
            this.boton_pt4 = new System.Windows.Forms.Button();
        }
    }
}
```

```

        this.boton_pt2 = new System.Windows.Forms.Button();
        this.boton_pt3 = new System.Windows.Forms.Button();
        this.tbSerial = new System.Windows.Forms.TextBox();
        this.puertoSerial = new
System.IO.Ports.SerialPort(this.components);
        this.SuspendLayout();
        //
        // botonConectar
        //
        this.botonConectar.Location = new System.Drawing.Point(12, 35);
        this.botonConectar.Name = "botonConectar";
        this.botonConectar.Size = new System.Drawing.Size(75, 23);
        this.botonConectar.TabIndex = 0;
        this.botonConectar.Text = "Conectar";
        this.botonConectar.UseVisualStyleBackColor = true;
        this.botonConectar.Click += new
System.EventHandler(this.botonConectar_Click);
        //
        // botonDesconectar
        //
        this.botonDesconectar.Location = new System.Drawing.Point(93,
35);

        this.botonDesconectar.Name = "botonDesconectar";
        this.botonDesconectar.Size = new System.Drawing.Size(85, 23);
        this.botonDesconectar.TabIndex = 1;
        this.botonDesconectar.Text = "Desconectar";
        this.botonDesconectar.UseVisualStyleBackColor = true;
        this.botonDesconectar.Click += new
System.EventHandler(this.botonDesconectar_Click);
        //
        // cbPuertos
        //
        this.cbPuertos.FormattingEnabled = true;
        this.cbPuertos.Location = new System.Drawing.Point(184, 37);
        this.cbPuertos.Name = "cbPuertos";
        this.cbPuertos.Size = new System.Drawing.Size(128, 21);
        this.cbPuertos.TabIndex = 3;
        //
        // botonAutomatico
        //
        this.botonAutomatico.Enabled = false;
        this.botonAutomatico.Location = new System.Drawing.Point(31,
86);

        this.botonAutomatico.Name = "botonAutomatico";
        this.botonAutomatico.Size = new System.Drawing.Size(95, 23);

```

```

        this.botonAutomatico.TabIndex = 4;
        this.botonAutomatico.Text = "Modo Auto.";
        this.botonAutomatico.UseVisualStyleBackColor = true;
        this.botonAutomatico.Click += new
System.EventHandler(this.botonAutomatico_Click);
        //
        // botonManual
        //
        this.botonManual.Enabled = false;
        this.botonManual.Location = new System.Drawing.Point(184, 86);
        this.botonManual.Name = "botonManual";
        this.botonManual.Size = new System.Drawing.Size(95, 23);
        this.botonManual.TabIndex = 5;
        this.botonManual.Text = "Modo Manual";
        this.botonManual.UseVisualStyleBackColor = true;
        this.botonManual.Click += new
System.EventHandler(this.botonManual_Click);
        //
        // boton_pt1
        //
        this.boton_pt1.Enabled = false;
        this.boton_pt1.Location = new System.Drawing.Point(57, 133);
        this.boton_pt1.Name = "boton_pt1";
        this.boton_pt1.Size = new System.Drawing.Size(95, 23);
        this.boton_pt1.TabIndex = 6;
        this.boton_pt1.Text = "Apagar todo";
        this.boton_pt1.UseVisualStyleBackColor = true;
        this.boton_pt1.Click += new
System.EventHandler(this.boton_pt1_Click);
        //
        // boton_pt4
        //
        this.boton_pt4.Enabled = false;
        this.boton_pt4.Location = new System.Drawing.Point(57, 162);
        this.boton_pt4.Name = "boton_pt4";
        this.boton_pt4.Size = new System.Drawing.Size(95, 23);
        this.boton_pt4.TabIndex = 7;
        this.boton_pt4.Text = "Encender todo";
        this.boton_pt4.UseVisualStyleBackColor = true;
        this.boton_pt4.Click += new
System.EventHandler(this.boton_pt4_Click);
        //
        // boton_pt2
        //
        this.boton_pt2.Enabled = false;

```

```

        this.boton_pt2.Location = new System.Drawing.Point(153, 134);
        this.boton_pt2.Name = "boton_pt2";
        this.boton_pt2.Size = new System.Drawing.Size(98, 22);
        this.boton_pt2.TabIndex = 8;
        this.boton_pt2.Text = "Solo V1";
        this.boton_pt2.UseVisualStyleBackColor = true;
        this.boton_pt2.Click += new
System.EventHandler(this.boton_pt2_Click);
        //
        // boton_pt3
        //
        this.boton_pt3.Enabled = false;
        this.boton_pt3.Location = new System.Drawing.Point(153, 162);
        this.boton_pt3.Name = "boton_pt3";
        this.boton_pt3.Size = new System.Drawing.Size(98, 23);
        this.boton_pt3.TabIndex = 9;
        this.boton_pt3.Text = "Solo V2";
        this.boton_pt3.UseVisualStyleBackColor = true;
        this.boton_pt3.Click += new
System.EventHandler(this.boton_pt3_Click);
        //
        // tbSerial
        //
        this.tbSerial.Location = new System.Drawing.Point(12, 208);
        this.tbSerial.Multiline = true;
        this.tbSerial.Name = "tbSerial";
        this.tbSerial.ReadOnly = true;
        this.tbSerial.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
        this.tbSerial.Size = new System.Drawing.Size(300, 132);
        this.tbSerial.TabIndex = 10;
        //
        // puertoSerial
        //
        this.puertoSerial.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.puertoSerial_DataReceive
d);
        //
        // Frame
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(321, 349);
        this.Controls.Add(this.tbSerial);
        this.Controls.Add(this.boton_pt3);

```

```

        this.Controls.Add(this.boton_pt2);
        this.Controls.Add(this.boton_pt4);
        this.Controls.Add(this.boton_pt1);
        this.Controls.Add(this.botonManual);
        this.Controls.Add(this.botonAutomatico);
        this.Controls.Add(this.cbPuertos);
        this.Controls.Add(this.botonDesconectar);
        this.Controls.Add(this.botonConectar);
        this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
        this.Name = "Frame";
        this.Text = "Veintilación";
        this.Load += new System.EventHandler(this.Form1_Load);
        this.ResumeLayout(false);
        this.PerformLayout();

    }

    #endregion

    private System.Windows.Forms.Button botonConectar;//Se declara el
    boton conectar
    private System.Windows.Forms.Button botonDesconectar;//Se declara el
    boton desconectar
    private System.Windows.Forms.ComboBox cbPuertos;//Se declara el
    combobox donde se almacenan los puertos
    private System.Windows.Forms.Button botonAutomatico;//Se declara el
    boton automatico
    private System.Windows.Forms.Button botonManual;//Se declara el
    boton manual
    private System.Windows.Forms.Button boton_pt1;//Se declara el boton
    "Apagar todo"
    private System.Windows.Forms.Button boton_pt4;//Se declara el boton
    "Encender todo"
    private System.Windows.Forms.Button boton_pt2;//Se declara el boton
    "Solo V1"
    private System.Windows.Forms.Button boton_pt3;//Se declara el boton
    "Solo V2"
    private System.Windows.Forms.TextBox tbSerial;//Se declara el
    textbox donde se imprime la salida del dispositivo
    private System.IO.Ports.SerialPort puertoSerial;//Se declara la
    lectura de los puertos de comunicacion
    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MicroContr1 //Se usa para declarar un ámbito que contiene un
conjunto de objetos relacionados
{
    public partial class Frame : Form//Se crea la clase parcial del frame
    {
        public Frame()//Se crea el metodo Frame
        {
            InitializeComponent();//Se inicializa el Frame
        }

        private void Form1_Load(object sender, EventArgs e)//Se declara el
metodo para cargar los puertos
        {
            String[] puertos =
System.IO.Ports.SerialPort.GetPortNames();//Variable donde se almacenan los
puertos de la PC
            cbPuertos.DataSource = puertos;//Se guardan los puertos de la pc
en la ComboBox
            CheckForIllegalCrossThreadCalls = false;//Se iguala a falso la
verificacion de llamadas de hilos
        }

        private void botonDesconectar_Click(object sender, EventArgs e)//Se
declara el evento para el boton para desconectar el puerto deseado
        {
            if (puertoSerial.IsOpen)//Se pregunta si el puerto esta abierto
            {
                puertoSerial.Close();//Se cierra el puerto de comunicacion
                MessageBox.Show("Puerto CERRADO");//Se despliega un mensaje
de confirmación
                boton_pt1.Enabled = false;//Se deshabilita el boton "Apagar
todo" de la interfaz
                boton_pt2.Enabled = false;//Se deshabilita el boton "Solo
V1" de la interfaz
            }
        }
    }
}

```



```

        boton_pt3.Enabled = false;//Se deshabilita el "Solo V2" de
la interfaz
        boton_pt4.Enabled = false;//Se deshabilita el boton
"Encender todo" de la interfaz
        botonAutomatico.Enabled = false;//Se deshabilita el boton
del modo automatico de la interfaz
        botonManual.Enabled = false;//Se deshabilita el boton del
modo manual de la interfaz
    }
}

private void botonConectar_Click(object sender, EventArgs e)//Se
declara el evento para el boton para conectar el puerto deseado
{
    string nombrepuerto = cbPuertos.SelectedItem.ToString();//Se
almacena el nombre del puerto seleccionado por el usuario

    if (!puertoSerial.IsOpen)//Se pregunta si el puerto NO esta
abierto
    {
        puertoSerial.PortName = nombrepuerto;//Se le asigna el valor
al puerto
        puertoSerial.Open();//Se abre el puerto de comunicacion
        MessageBox.Show("Puerto " + nombrepuerto + " abierto");//Se
despliega un mensaje de confirmación
        botonAutomatico.Enabled = true;//Se habilita el boton del
modo automatico de la interfaz
        botonManual.Enabled = true;//Se habilita el boton del modo
manual de la interfaz
    }
}

private void botonAutomatico_Click(object sender, EventArgs e)//Se
declara el evento para el boton automatico
{
    if (puertoSerial.IsOpen)//Se pregunta si el puerto esta abierto
    {
        puertoSerial.WriteLine("A");//Se envia el caracter al
dispositivo para ejecutar su programacion predefinida
        boton_pt1.Enabled = false;//Se deshabilita el boton "Apagar
todo" de la interfaz
        boton_pt2.Enabled = false;//Se deshabilita el boton "Solo
V1" de la interfaz
    }
}

```

```

        boton_pt3.Enabled = false; //Se deshabilita el "Solo V2" de
la interfaz
        boton_pt4.Enabled = false; //Se deshabilita el boton
"Encender todo" de la interfaz
        botonManual.Enabled = true; //Se habilita el boton del modo
manual de la interfaz
        tbSerial.AppendText(System.Environment.NewLine); //Se imprime
en el textbox la salida del dispositivo
        tbSerial.AppendText(System.Environment.NewLine); //Se imprime
en el textbox la salida del dispositivo
    }

}

private void botonManual_Click(object sender, EventArgs e) //Se
declara el evento para el boton manual
{
    if (puertoSerial.IsOpen) //Se pregunta si el puerto esta abierto
    {
        puertoSerial.WriteLine("M");
        boton_pt1.Enabled = true; //Se habilita el boton "Apagar
todo" de la interfaz
        boton_pt2.Enabled = true; //Se habilita el boton "Solo V1" de
la interfaz
        boton_pt3.Enabled = true; //Se habilita el "Solo V2" de la
interfaz
        boton_pt4.Enabled = true; //Se habilita el boton "Encender
todo" de la interfaz
        botonManual.Enabled = false; //Se deshabilita el boton del
modo manual de la interfaz
        tbSerial.AppendText(System.Environment.NewLine); //Se imprime
en el textbox la salida del dispositivo
        tbSerial.AppendText(System.Environment.NewLine); //Se imprime
en el textbox la salida del dispositivo
    }
}

//Se declara el evento para la confirmación del dato recibido
private void puertoSerial_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
{
    if (puertoSerial.IsOpen) //Se pregunta si el puerto esta abierto
    {
        try {

```

```
        string recibido = puertoSerial.ReadLine();//Se almacena
la lectura del puerto serial
        tbSerial.AppendText(recibido);//Se imprime en el textbox
la salida del dispositivo
        tbSerial.AppendText(System.Environment.NewLine);//Se
imprime en el textbox la salida del dispositivo
```

```
    } catch (Exception ) {
    }
}
}
```

```
private void boton_pt1_Click(object sender, EventArgs e)//Se declara
el evento para el boton "Apagar todo"
```

```
{
    if (puertoSerial.IsOpen)//Se pregunta si el puerto esta abierto
    {
        puertoSerial.WriteLine("1");//Se envia el caracter al
dispositivo para ejecutar su programacion predefinida
    }
}
```

```
private void boton_pt2_Click(object sender, EventArgs e)//Se declara
el evento para el boton "Solo V1"
```

```
{
    if (puertoSerial.IsOpen)//Se pregunta si el puerto esta abierto
    {
        puertoSerial.WriteLine("2");//Se envia el caracter al
dispositivo para ejecutar su programacion predefinida
    }
}
```

```
private void boton_pt3_Click(object sender, EventArgs e)//Se declara
el evento para el boton "Solo V2"
```

```
{
    if (puertoSerial.IsOpen)//Se pregunta si el puerto esta abierto
    {
        puertoSerial.WriteLine("3");//Se envia el caracter al
dispositivo para ejecutar su programacion predefinida
    }
}
```

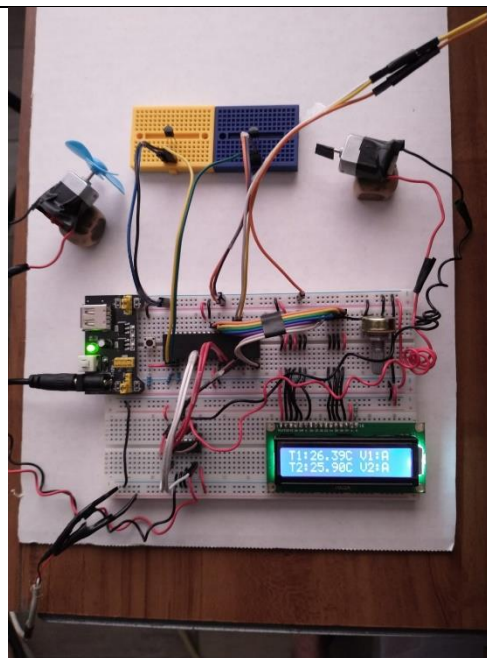
```
private void boton_pt4_Click(object sender, EventArgs e)//Se declara
el evento para el boton "Encender todo"
```

```
{
```

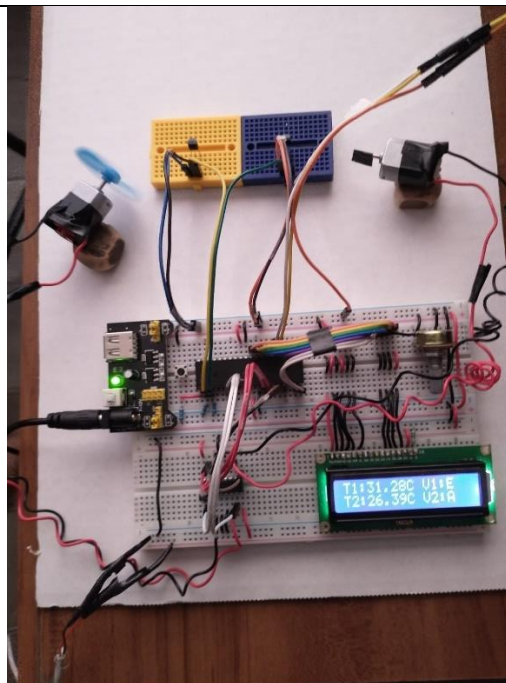
```
        if (puertoSerial.IsOpen)//Se pregunta si el puerto esta abierto
        {
            puertoSerial.WriteLine("4");//Se envia el caracter al
dispositivo para ejecutar su programacion predefinida
        }
    }
}
```

Fotografías del circuito físico

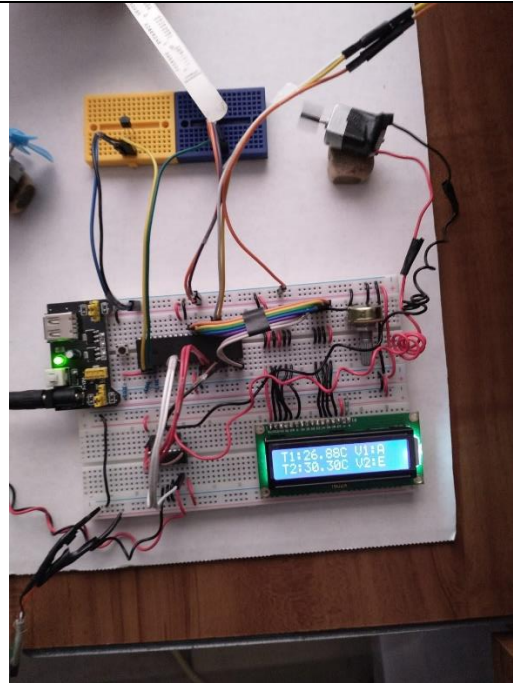
Modo Automático con los ventiladores apagados.



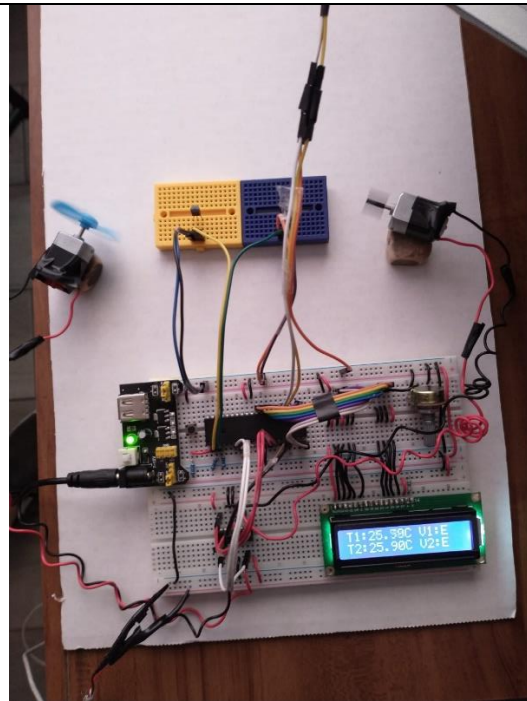
Modo Automático con el ventilador 1 encendido.



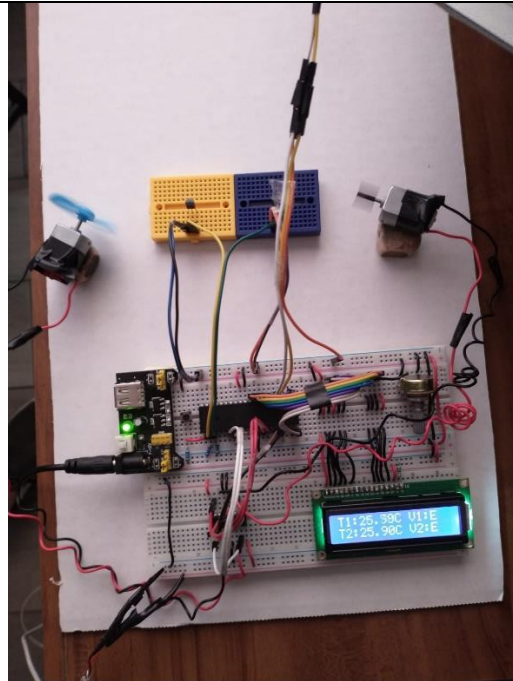
Modo Automático con el ventilador 2 encendido



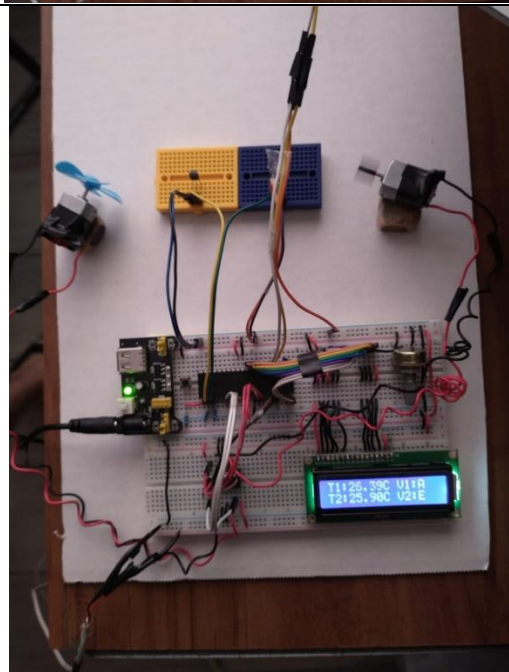
Manual con los ventiladores encendidos



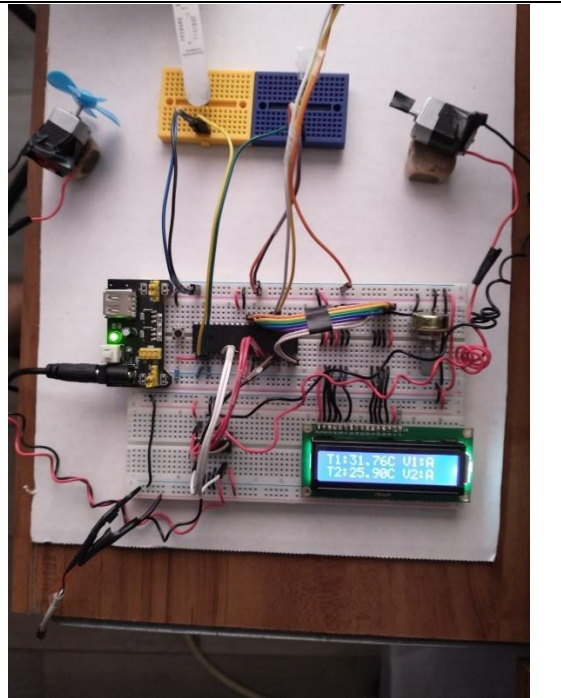
Manual con el ventilador 1 encendido




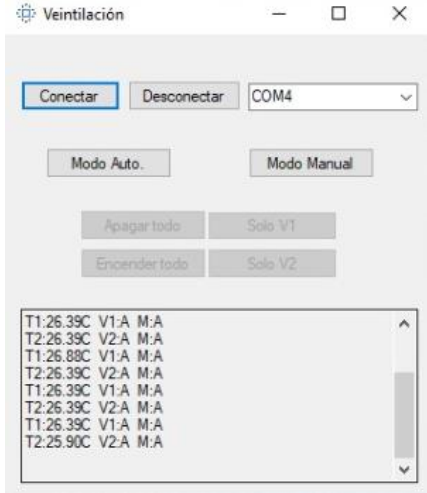
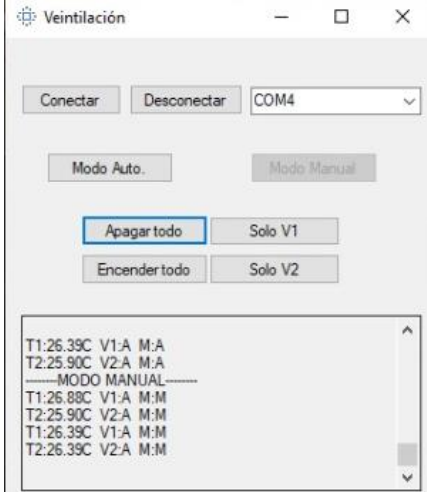
Manual con el ventilador 2 encendido



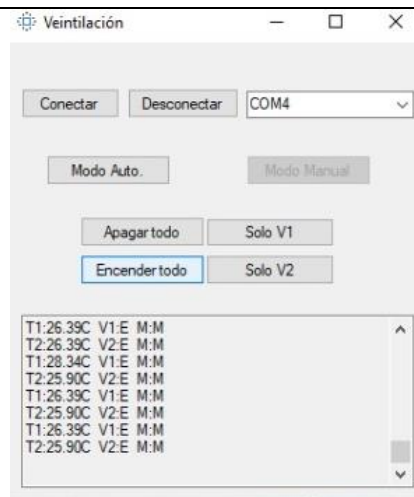
Manual con los ventiladores apagados



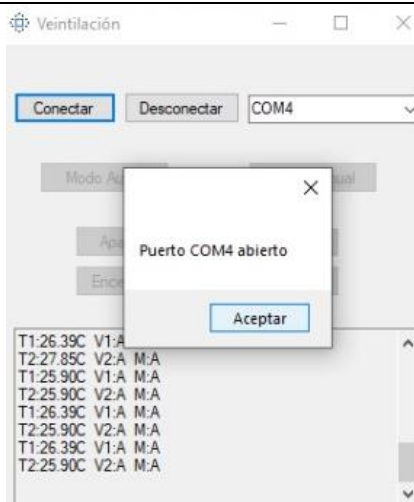
Capturas de la Interfaz

Vista inicial de la interfaz.	
Modo automático.	
Modo manual con todos los ventiladores apagados.	

Modo Manual con todos los ventiladores encendidos.



Ventana emergente del puerto de comunicaciones.



Lista de Materiales (Bill of materials)

COMPONENTE	CANTIDAD	DESCRIPCION	PROVEEDOR	COSTO POR UNIDAD	COSTO TOTAL
Protoboard 830 puntos	2	Tabla con 830 puntos interconectados que permite realizar conexiones entre componentes electronicos	UNIT Electronics	\$36.00	\$72.00
Protoboard 170 puntos	2	Tabla con 170 puntos interconectados que permite realizar conexiones entre componentes electronicos	Geek Factory	\$10.00	\$20.00
Sensor de temperatura LM35DZ	2	El Sensor de temperatura LM35DZ (Analógico) con salida analógica de 10 mV/C. El voltaje de salida del sensor lineal y directamente proporcional a la temperatura en grados Celsius (centígrados).	Geek Factory	\$42.50	\$85.00
Potenciometro 10K Ohms	1	Resistencia variable que funciona por medio de una perilla con un rango de resistencia de 0 - 10k Ohms	UNIT Electronics	\$12.00	\$12.00
Pantalla LCD 16X2	1	Pantalla LCD 16x2 compatible con el controlador HD44780. Su presentación es en color azul con letras en color blanco.	UNIT Electronics	\$66.00	\$66.00
USB Serial TTL CP2102	1	Conversor capaz de conectar y permitir la comunicación entre un microcontrolador y una PC	UNIT Electronics	\$71.00	\$71.00
Jumper Macho/Macho	20	Cables para protoboard con terminales Macho / Macho	UNIT Electronics	\$27.00	\$27.00
Jumper Macho/Hembra	10	Cables para protoboard con terminales Macho / Hembra	UNIT Electronics	\$27.00	\$27.00
Resistencia 1K Ohms	3	Resistencia variable que funciona por medio de una perilla con un rango de resistencia de 0 - 1k Ohms	UNIT Electronics	\$0.20	\$0.60
PIC16F887	1	El Microcontrolador PIC16F887 cuenta con 8 Bits, 14KB, 368 RAM, 20MHz, 40 Pines. Cuenta con un oscilador interno de precisión con frecuencia seleccionable entre 31 kHz y 8 MHz.	UNIT Electronics	\$101.00	\$101.00
Fuente 5 Volts	2	La fuente para protoboard B102 con mini usb es un módulo que permite alimentar protoboard's de 400 y 830, puede suministrar una alimentación de 5 o 3.3V	UNIT Electronics	\$49.00	\$98.00
L293D	1	Circuito integrado capaz de producir corrientes bidireccionales, es útil en aplicaciones que requieran controlar la dirección de giro y velocidad de motores de DC.	UNIT Electronics	\$28.00	\$28.00
Motor R150 5 Volts	2	El Motor R150 es un motor de corriente directa (DC) que funciona en un rango de voltaje desde 1V hasta 5V lo que le permite alcanzar una velocidad máxima (sin carga) de 5000 revoluciones por minuto (RPM) en su eje circular de 4.5 mm de largo y de 2 mm de diámetro.	UNIT Electronics	\$10.00	\$20.00
Push Button	1	Interruptor que permite el paso del voltaje cada vez que se pulsa	UNIT Electronics	\$2.00	\$2.00
Alambre AWG 22	1	Alambre calibre 22 AWG con material conductor de cobre estañado de un solo núcleo y materia aislante de PVC	Steren	\$6.00	\$6.00
COSTO FINAL:					\$635.60

Planes a Futuro

El proyecto que hemos presentado hasta ahora es solo un prototipo y una idea conceptual de lo que, a nuestro parecer, podría llegar a ser un sistema de enfriamiento inteligente a gran escala, para ser precisos, un sistema de enfriamiento doméstico. Creemos que la mejor implementación para un sistema doméstico es la simpleza, para no limitar al usuario en tareas complicadas.

En este caso se requeriría de una tecnología más avanzada y de una mejor infraestructura. La mejor opción para auxiliarnos en este caso sería la del IoT (Internet of Things).

Como primer paso, deberíamos implementar a nuestros componentes una forma de que operen de manera inalámbrica y esto se logra con puertos de comunicación WiFi, para que todo esto sea manejado dentro de una red LAN.

Posteriormente podríamos implementar sensores de temperatura más avanzados, que nos permitan medir también la humedad, esto con la finalidad de que el usuario pueda tener un monitoreo constante del ambiente de su hogar.

Continuando con esta visualización del futuro, los ventiladores podrían ser sustituidos por sistemas reguladores de humedad y sistemas de aire acondicionado, los cuales se encargan de regular el clima de la habitación a un valor predefinido o al que el usuario desee, tal y como lo hace nuestro prototipo con el manejo de su modo manual y su modo automático.

Finalmente, el monitoreo en tiempo real podría realizarse a través de una aplicación para móviles que le permita al usuario regular los modos de operación, así como tener un reporte del estado del sistema en tiempo real, desde la comodidad que le provee un dispositivo móvil.

Lo mas importante en esto, además de la optimización de dicho proceso, es observar el beneficio que se puede obtener de esta tarea o incluso el beneficio monetario que nos puede traer, ya sea para uso propio o para su monetización y posible comercialización.