

**MATERIA**

Sistemas Programables

**CARRERA**

Ing. Sistemas Computacionales.

**PRESENTA:**

**NOMBRE DEL ALUMNO**

Luis Eduardo Hernández Magaña.

**NOMBRE DEL MAESTRO:**

Ing. Levy Rojas Carlos Rafael

**LEÓN, GUANAJUATO Periodo: Enero-Julio 201**

**CONTENIDO**

**1.Marco Teórico**

**a. Datos sobre la placa Arduino Uno**

**datos sobre el Proyecto**

**datos sobre el Sensor de Temperatura**

**2.Materiales y Costo**

**3.Código Fuente**

**a. Código fuente del programa comentado (java)**

**4.Diagrama del Circuito**

**5.Capturas y Fotografías Trabajo**

**6.Bibliografía**

**MARCO TEORICO**

1. **Datos sobre la placa Arduino UNO**

Arduino es una placa con un micro controlador de la marca Atmel y con toda la circuitería de soporte, que incluye, reguladores de tensión, un puerto USB (En los últimos modelos, aunque el original utilizaba un puerto serie) conectado a un módulo adaptador USB-Serie que permite programar el micro controlador desde cualquier PC de manera cómoda y también hacer pruebas de comunicación con el propio chip.

Un Arduino dispone de 14 pines que pueden configurarse como entrada o salida y a los que puede conectarse cualquier dispositivo que sea capaz de transmitir o recibir señales digitales de 0 y 5 V.

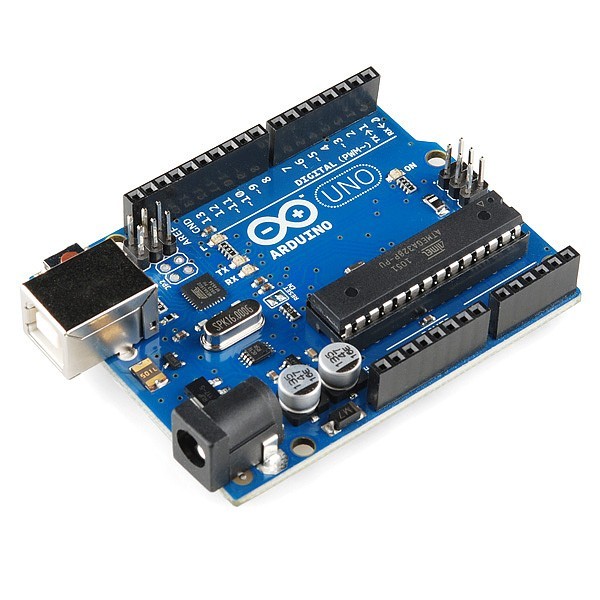
También dispone de entradas y salidas analógicas. Mediante las entradas analógicas podemos obtener datos de sensores en forma de variaciones continuas de un voltaje. Las salidas analógicas suelen utilizarse para enviar señales de control en forma de señales PWM.

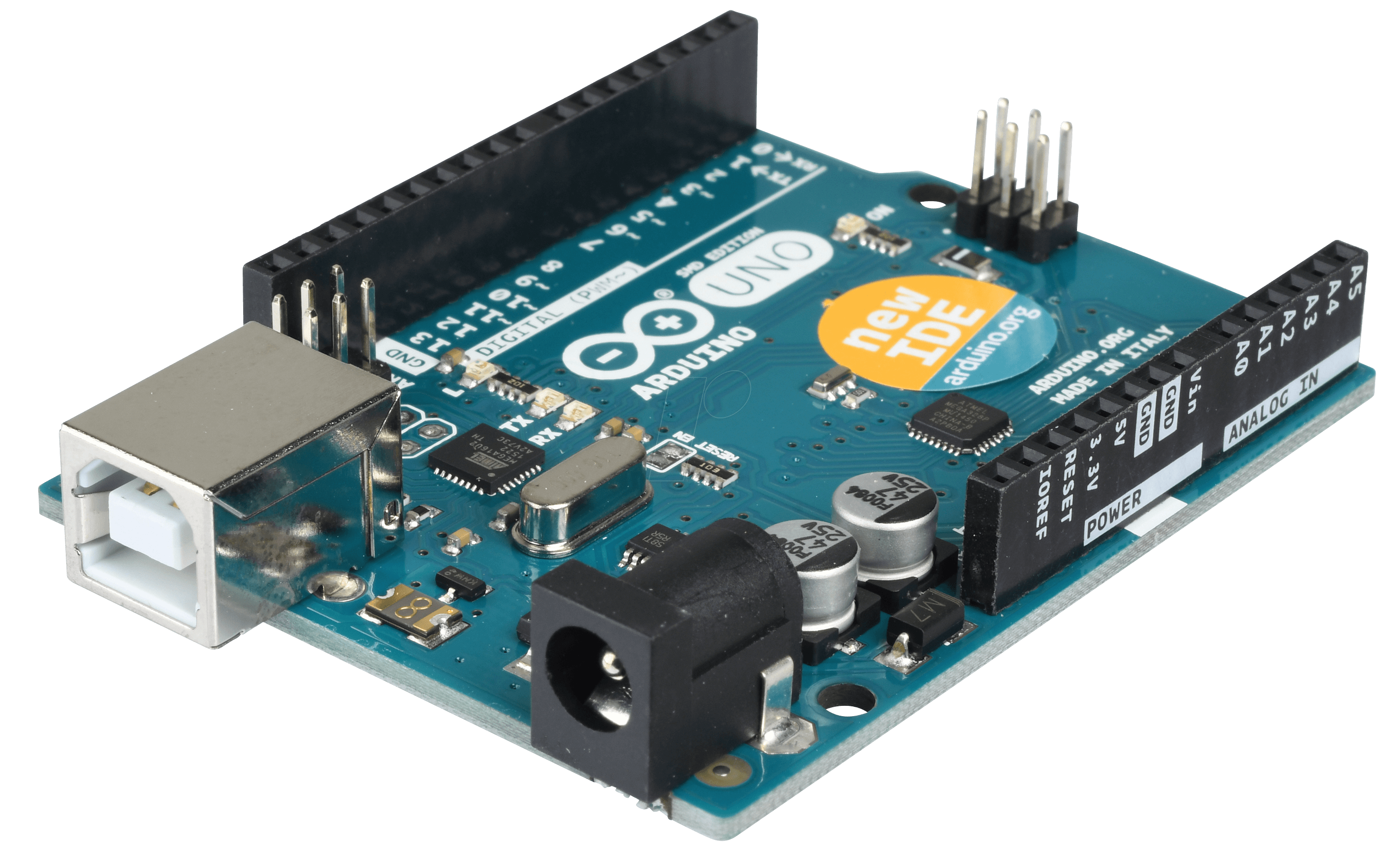
Arduino UNO es la última versión de la placa, existen dos variantes, la Arduino UNO convencional y la Arduino UNO SMD. La única diferencia entre ambas es el tipo de micro controlador que montan.

• La primera es un micro controlador Atmega en formato DIP.

• Y la segunda dispone de un micro controlador en formato SMD.

Nosotros nos decantaremos por la primera porque nos permite programar el chip sobre la propia placa y después integrarlo en otros montajes.





Arduino con microcontrolador en formato SMD

Arduino UNO con controlador en formato DIP

Arduino UNO con micro controlador en formato DIP Arduino UNO con micro controlador en formato SMD Entradas y salidas:

Cada uno de los 14 pines digitales se puede usar como entrada o como salida. Funcionan a 5V, cada pin puede suministrar hasta 40 mA. La intensidad máxima de entrada también es de 40 mA.

Cada uno de los pines digitales dispone de una resistencia de pull-up interna de entre 20KΩ y 50 KΩ que está desconectada, salvo que nosotros indiquemos lo contrario.

Arduino también dispone de 6 pines de entrada analógicos que trasladan las señales a un conversor analógico/digital de 10 bits. Pines especiales de entrada y salida:

* RX y TX: Se usan para transmisiones serie de señales TTL.
* Interrupciones externas: Los pines 2 y 3 están configurados para generar una interrupción en el atmega. Las interrupciones pueden dispararse cuando se encuentra un valor bajo en estas entradas y con flancos de subida o bajada de la entrada.
* PWM: Arduino dispone de 6 salidas destinadas a la generación de señales PWM de hasta 8 bits.
* SPI: Los pines 10, 11, 12 y 13 pueden utilizarse para llevar a cabo comunicaciones SPI, que
* Permiten trasladar información full dúplex en un entorno Maestro/Esclavo
* I2C: Permite establecer comunicaciones a través de un bus I2C. El bus I2C es un producto de Phillips para interconexión de sistemas embebidos. Actualmente se puede encontrar una gran diversidad de dispositivos que utilizan esta interfaz, desde pantallas LCD, memorias EEPROM, sensores...

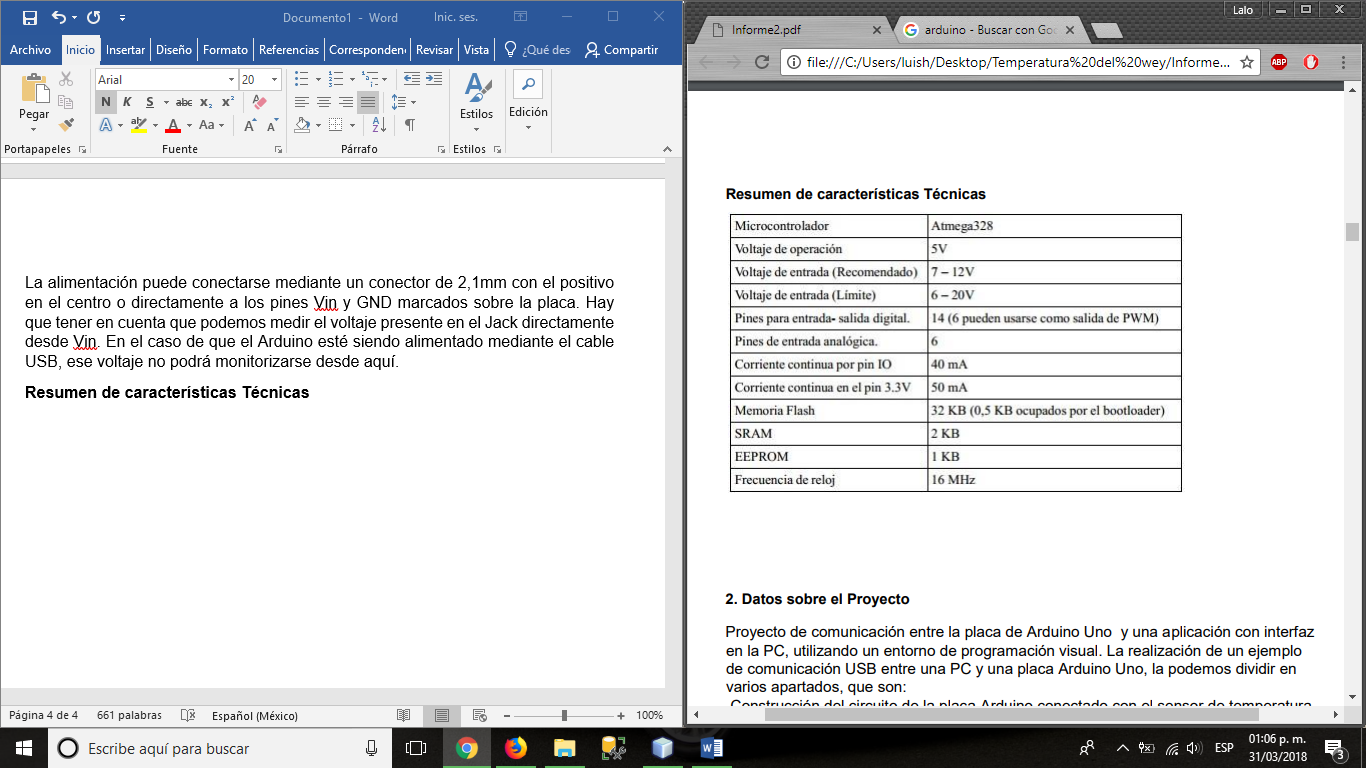
**Alimentación de un Arduino**

Puede alimentarse directamente a través del propio cable USB o mediante una fuente de alimentación externa, como puede ser un pequeño transformador o, por ejemplo, una pila de 9V.

Los límites están entre los 6 y los 12 V. Como única restricción hay que saber que, si la placa se alimenta con menos de 7V, la salida del regulador de tensión a 5V puede dar menos que este voltaje y si sobrepasamos los 12V, probablemente dañaremos la placa.

La alimentación puede conectarse mediante un conector de 2,1mm con el positivo en el centro o directamente a los pines Vin y GND marcados sobre la placa. Hay que tener en cuenta que podemos medir el voltaje presente en el Jack directamente desde Vin. En el caso de que el Arduino esté siendo alimentado mediante el cable USB, ese voltaje no podrá monitorizarse desde aquí.

**Resumen de características Técnicas**



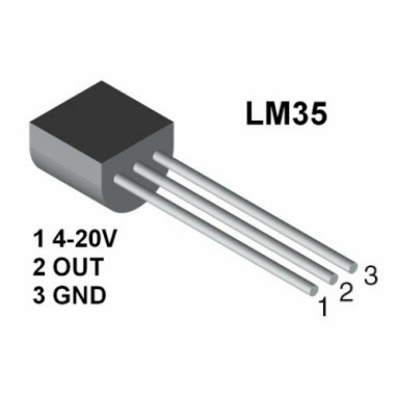
1. **Datos sobre el Proyecto**

* Proyecto de comunicación entre la placa de Arduino Uno y una aplicación con interfaz en la PC, utilizando un entorno de programación visual. La realización de un ejemplo de comunicación USB entre una PC y una placa Arduino Uno, la podemos dividir en varios apartados, que son:
* Construcción del circuito de la placa Arduino conectado con el sensor de temperatura y el LCD respectivamente desarrollado en el Proto-Board.
* Desarrollo del programa que conectara el Arduino (IDE Arduino).
* Instalación de los respectivos drivers del Arduino
* Desarrollo de la aplicación de escritorio que servirá de interfaz de comunicación entre el usuario y la placa Arduino (Lenguaje de programación utilizado java NetBeans).

1. Datos sobre el Sensor de Temperatura

LM35 -- Sensor de temperatura de precisión

El LM35 es un sensor de temperatura integrado de precisión, cuya tensión de salida es linealmente proporcional a temperatura en ºC (grados centígrados). El LM35 por lo tanto tiene una ventaja sobre los sensores de temperatura lineal calibrada en grados Kelvin: que el usuario no está obligado a restar una gran tensión constante para obtener grados centígrados. El LM35 no requiere ninguna calibración externa o ajuste para proporcionar una precisión típica de ± 1.4 ºC a temperatura ambiente y ± 3.4 ºC a lo largo de su rango de temperatura (de -55 a 150 ºC). El dispositivo se ajusta y calibra durante el proceso de producción. La baja impedancia de salida, la salida lineal y la precisa calibración inherente, permiten la creación de circuitos de lectura o control especialmente sencillos. El LM35 puede funcionar con alimentación simple o alimentación doble (+ y -)

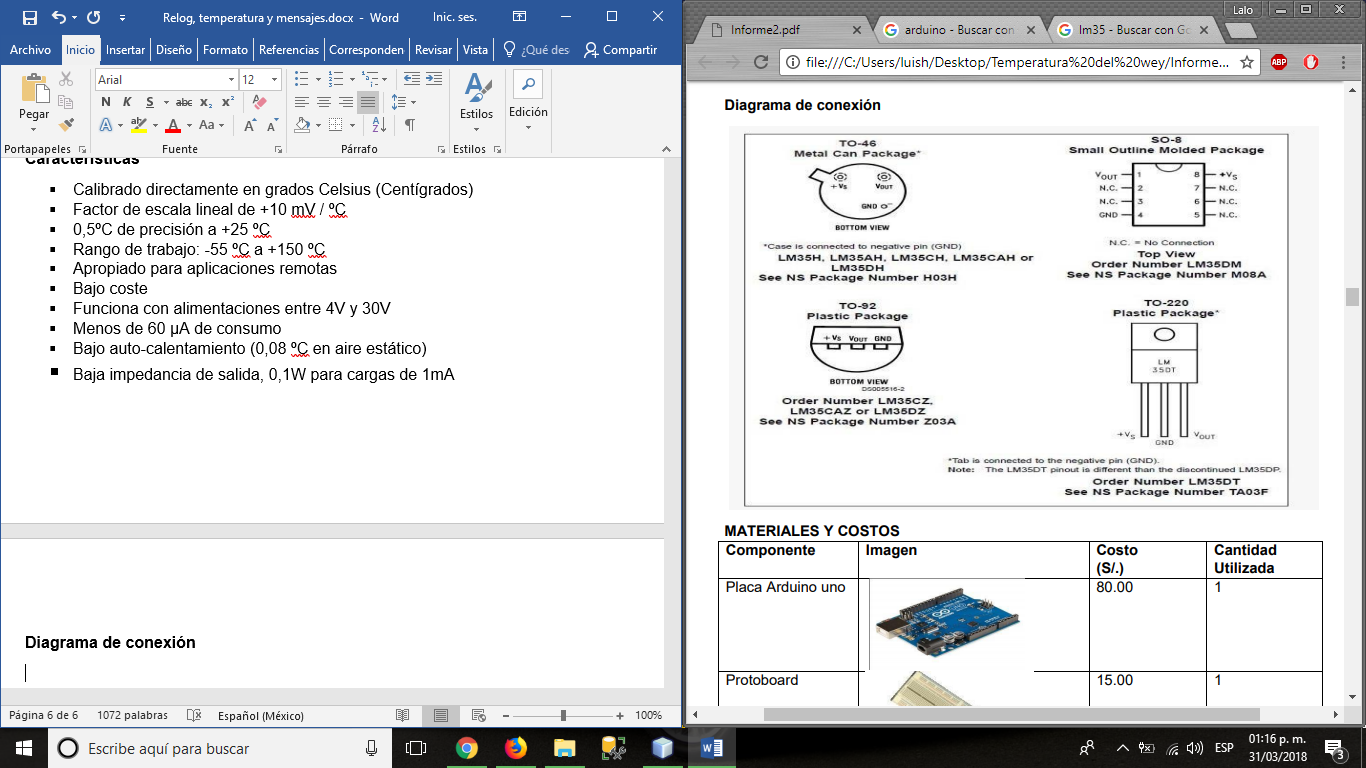


Requiere sólo 60 µA para alimentarse, y bajo factor de auto-calentamiento, menos de 0,1 ºC en aire estático. El LM35 está preparado para trabajar en una gama de temperaturas que abarca desde los- 55 ºC bajo cero a 150 ºC, mientras que el LM35C está preparado para trabajar entre -40 ºC y 110 ºC (con mayor precisión).

**Características**

* Calibrado directamente en grados Celsius (Centígrados)
* Factor de escala lineal de +10 mV / ºC
* 0,5ºC de precisión a +25 ºC
* Rango de trabajo: -55 ºC a +150 ºC
* Apropiado para aplicaciones remotas
* Bajo coste
* Funciona con alimentaciones entre 4V y 30V
* Menos de 60 µA de consumo
* Bajo auto-calentamiento (0,08 ºC en aire estático)
* Baja impedancia de salida, 0,1W para cargas de 1mA

**Diagrama de conexión**



**Materiales y Costos**

|  |  |  |  |
| --- | --- | --- | --- |
| Componentes | Imagen | Costo | Cantidad Utilizada |
| Placa Arduino UNO | Resultado de imagen para arduino | 220 | 1 |
| Protoboard | Resultado de imagen para protoboard | 150 | 1 |
| LCD |  | 75 | 1 |
| Potenciometro |  | 5 | 1 |
| Cables |  | 1.50 | 20 |
| Sensor de Temperatura LM35 |  | 35 | 1 |

Código Fuente

1. Programa y Código fuente del IDE Arduino

**Clase RelogArduino**

#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2);

byte PIN\_SENSOR = A0;

int dato\_serial = 0;

float C;

int temp;

void setup() {

lcd.begin(16,2);

Serial.begin(9600);

Serial.setTimeout(50);

pinMode(8, OUTPUT);

pinMode(9, OUTPUT);

pinMode(10, OUTPUT);

}

void loop() {

C = (5.0 \* analogRead(PIN\_SENSOR) \* 100.0)/ 1024;

temp = C;

if ( Serial.available() > 0){

//lectura\_dato();

dato\_serial = Serial.read();

comparacion\_dato();

if(C < 15)

{

digitalWrite(8,HIGH);

digitalWrite(9,LOW);

digitalWrite(10,LOW);

}

if(C > 20)

{

digitalWrite(8,LOW);

digitalWrite(9,LOW);

digitalWrite(10,HIGH);

}

if(C < 20 )

{

if(C > 15)

{

digitalWrite(8,LOW);

digitalWrite(9,HIGH);

digitalWrite(10,LOW);

}

}

}

String text = Serial.readString();

String line1 = text.substring(0, 16);

String line2 = text.substring(16, 32);

if(text.length() > 0){

lcd.setCursor(0,0);

lcd.print(" ");

lcd.setCursor(0,1);

lcd.print(" ");

}

lcd.setCursor(0,0);

lcd.print(line1);

lcd.setCursor(0,1);

lcd.print(line2);

}

void lectura\_dato (void ){

dato\_serial = Serial.read();

}

void comparacion\_dato (void){

if(dato\_serial == 'T'){

Serial.write(temp);

}

}

1. Programa y Código fuente de la interfaz java NetBeans

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.PrintWriter;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JPanel;

import com.fazecast.jSerialComm.SerialPort;

import javax.swing.JTextField;

public class RelogArduino {

static SerialPort chosenPort;

public static void main(String[] args) {

// Crea y configura la ventana

JFrame window = new JFrame();

window.setTitle("Arduino Hora, mensajes y temperatura");

window.setSize(800, 100);

window.setLayout(new BorderLayout());

window.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Crea el drop box y el boton conectar, y los coloca hasta arriba

JComboBox<String> portList = new JComboBox<String>();

JButton connectButton = new JButton("Connect");

JPanel topPanel = new JPanel();

JTextField txtcampo = new JTextField ("",20);

JButton btnenviar = new JButton("Enviar");

JButton btncancelar = new JButton("Desconectar");

JButton btntemperatura = new JButton("Temperatura");

topPanel.add(portList);

topPanel.add(connectButton);

topPanel.add(txtcampo);

topPanel.add(btnenviar);

topPanel.add(btncancelar);

topPanel.add(btntemperatura);

window.add(topPanel, BorderLayout.NORTH);

// Se asigna el drop box

SerialPort[] portNames = SerialPort.getCommPorts();

for(int i = 0; i < portNames.length; i++)

portList.addItem(portNames[i].getSystemPortName());

// Configura el boton conectar y el hilo para enviar al serial

connectButton.addActionListener(new ActionListener(){

@Override public void actionPerformed(ActionEvent arg0) {

if(connectButton.getText().equals("Connect")) {

// Se adjunta el numero de puerto al serial

chosenPort = SerialPort.getCommPort(portList.getSelectedItem().toString());

chosenPort.setComPortTimeouts(SerialPort.TIMEOUT\_SCANNER, 0, 0);

if(chosenPort.openPort()) {

connectButton.setEnabled(false);

btnenviar.setEnabled(false);

btntemperatura.setEnabled(false);

portList.setEnabled(false);

// Crea un nuevo hilo y enviar al arduino

Thread thread = new Thread(){

@Override public void run() {

// Espera el booteo

try {Thread.sleep(100); } catch(Exception e) {}

//Ingresa al loop del arduino

PrintWriter output = new PrintWriter(chosenPort.getOutputStream());

while(true) {

output.print(new SimpleDateFormat("hh:mm:ss a MMMMMMM dd, yyyy").format(new Date()));

output.flush();

try {Thread.sleep(100); } catch(Exception e) {}

}

}

};

thread.start();

}

} else {

// Desconecta el puerto serial

chosenPort.closePort();

portList.setEnabled(true);

}

}

});

btnenviar.addActionListener(new ActionListener(){

@Override public void actionPerformed(ActionEvent arg0) {

if(btnenviar.getText().equals("Enviar")) {

// Empareja el serial puerto

chosenPort = SerialPort.getCommPort(portList.getSelectedItem().toString());

chosenPort.setComPortTimeouts(SerialPort.TIMEOUT\_SCANNER, 0, 0);

if(chosenPort.openPort()) {

btnenviar.setEnabled(false);

btntemperatura.setEnabled(false);

connectButton.setEnabled(false);

portList.setEnabled(false);

// Crea un nuevo hilo y lo envia al arduino

Thread thread = new Thread(){

@Override public void run() {

// Espera la conexion

try {Thread.sleep(100); } catch(Exception e) {}

// Ingresa al loop del arduino

PrintWriter output = new PrintWriter(chosenPort.getOutputStream());

while(true) {

output.print(txtcampo.getText());

output.flush();

try {Thread.sleep(100); } catch(Exception e) {}

}

}

};

thread.start();

}

}

}

});

btncancelar.addActionListener(new ActionListener(){

@Override public void actionPerformed(ActionEvent arg0) {

// Desconecta los puertos seriales

chosenPort.closePort();

portList.setEnabled(true);

btnenviar.setEnabled(true);

btntemperatura.setEnabled(true);

connectButton.setEnabled(true);

}

});

btntemperatura.addActionListener(new ActionListener(){

@Override public void actionPerformed(ActionEvent arg0) {

Tempreatura tm = new Tempreatura();

tm.setVisible(true);

}

});

// Muestra la ventana

window.setVisible(true);

}

}

**Clase Temperatura**

package proyectotemperatura;

import gnu.io.CommPortIdentifier;

import gnu.io.PortInUseException;

import gnu.io.SerialPort;

import gnu.io.UnsupportedCommOperationException;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.util.Enumeration;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextField;

import javax.swing.border.EmptyBorder;

/\*\*

\*

\* @author Kaiser

\*/

//public class Ventana2 extends javax.swing.JFrame {

public class Ventana2 extends JFrame {

private JPanel contentPane;

private JTextField textField;

Enumeration puertos\_libres =null;

CommPortIdentifier port=null;

SerialPort puerto\_ser = null;

OutputStream out = null;

InputStream in = null;

int temperatura=10;

Thread timer;

JLabel lblNewLabel;

JButton btnNewButton,btnNewButton\_1;

public Ventana2() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 636, 365);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

timer = new Thread(new ImplementoRunnable());

timer.start();

timer.interrupt();

btnNewButton = new JButton("Conectar");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

puertos\_libres = CommPortIdentifier.getPortIdentifiers();

int aux=0;

while (puertos\_libres.hasMoreElements())

{

port = (CommPortIdentifier) puertos\_libres.nextElement();

int type = port.getPortType();

if (port.getName().equals(textField.getText()))

{

try {

puerto\_ser = (SerialPort) port.open("puerto serial", 2000);

int baudRate = 9600; // 9600bps

puerto\_ser.setSerialPortParams(

baudRate,

SerialPort.DATABITS\_8,

SerialPort.STOPBITS\_1,

SerialPort.PARITY\_NONE);

puerto\_ser.setDTR(true);

out = puerto\_ser.getOutputStream();

in = puerto\_ser.getInputStream();

btnNewButton\_1.setEnabled(true);

btnNewButton.setEnabled(false);

timer.resume();

} catch ( IOException e1) {

} catch (PortInUseException e1) {

e1.printStackTrace();

} catch (UnsupportedCommOperationException e1) {

e1.printStackTrace();

}

break;

}

}

}

});

btnNewButton.setBounds(38, 63, 89, 23);

contentPane.add(btnNewButton);

btnNewButton\_1 = new JButton("Desconectar");

btnNewButton\_1.setEnabled(false);

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

timer.interrupt();

puerto\_ser.close();

btnNewButton\_1.setEnabled(false);

btnNewButton.setEnabled(true);

}

});

btnNewButton\_1.setBounds(205, 63, 128, 23);

contentPane.add(btnNewButton\_1);

textField = new JTextField();

textField.setBounds(38, 32, 86, 20);

contentPane.add(textField);

textField.setColumns(10);

JLabel lblPuertoCom = new JLabel("Puerto COM");

lblPuertoCom.setBounds(37, 11, 90, 14);

contentPane.add(lblPuertoCom);

lblNewLabel = new JLabel("Temperatura");

lblNewLabel.setBounds(80, 124, 128, 24);

lblNewLabel .setFont(new java.awt.Font("Arial", 0, 20));

lblNewLabel .setForeground(Color.blue);

contentPane.add(lblNewLabel);

}public void paint(Graphics g) {

super.paint(g);

g.setColor(Color.blue);

g.fillRect(38, 250-temperatura, 20, temperatura);

try {

Thread.sleep(200);

} catch (InterruptedException e) {

e.printStackTrace();

}}

private class ImplementoRunnable implements Runnable{

int aux;

public void run() {

while(true){

try {

out.write('T');

Thread.sleep(100);

aux = in.read();

if (aux!=2){

temperatura = aux;

lblNewLabel.setText(""+temperatura+" ºC");

System.out.println(aux);

}repaint();

} catch (Exception e1) {

}}}}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 400, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 300, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Ventana2.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Ventana2.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Ventana2.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Ventana2.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Ventana2().setVisible(true);

}

});

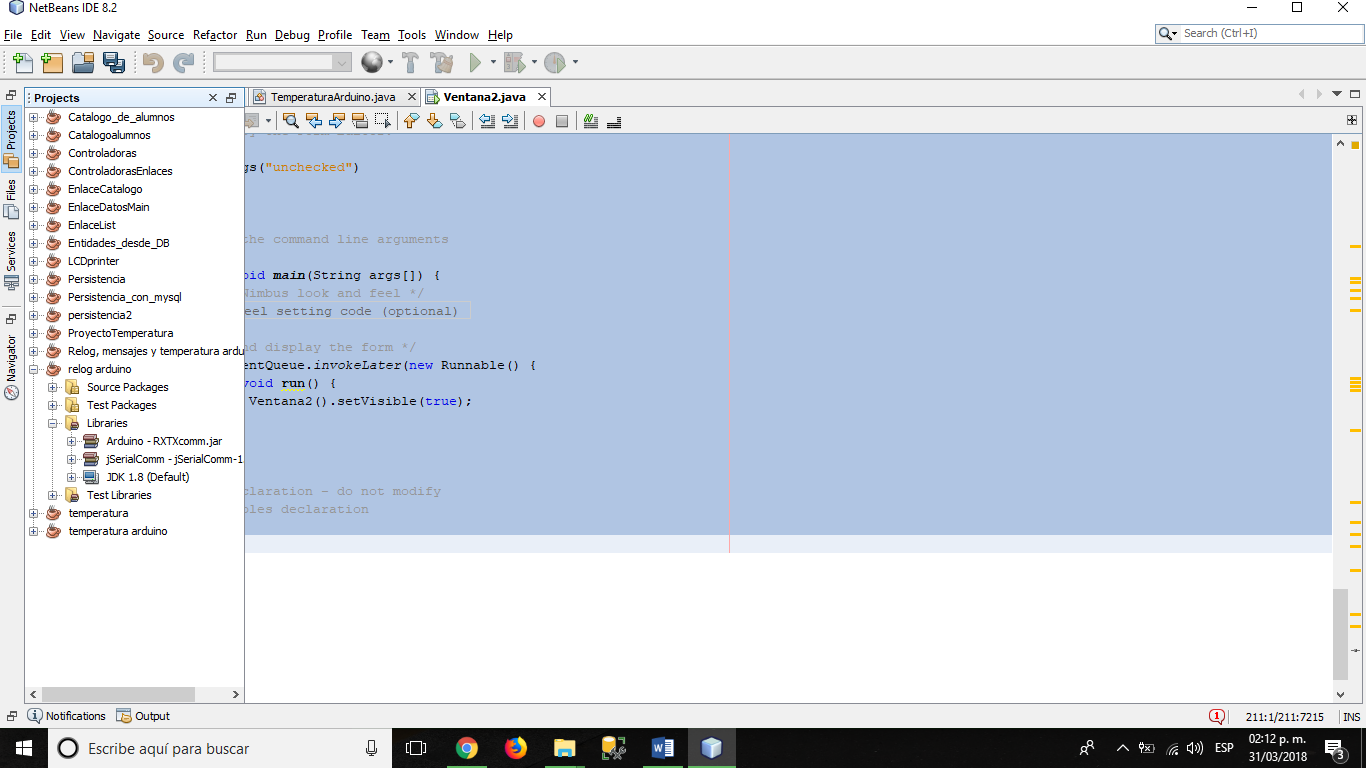
}

// Variables declaration - do not modify

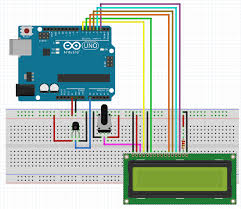
// End of variables declaration

}

Para que los programas puedan funcionar deben de tener estas librerias para su funcionamiento.



1. Diagrama del Circuito



1. Capturas y armado del trabajo

