Nombre:	Matricula	Fecha
Gerardo Naranjo	A01209499	29-10-2018
Roberto Figueroa	A01209689	
Alejandro Ossio	A01209122	

Numero de práctica:

Práctica 1, parcial 2.

Nombre de la práctica:

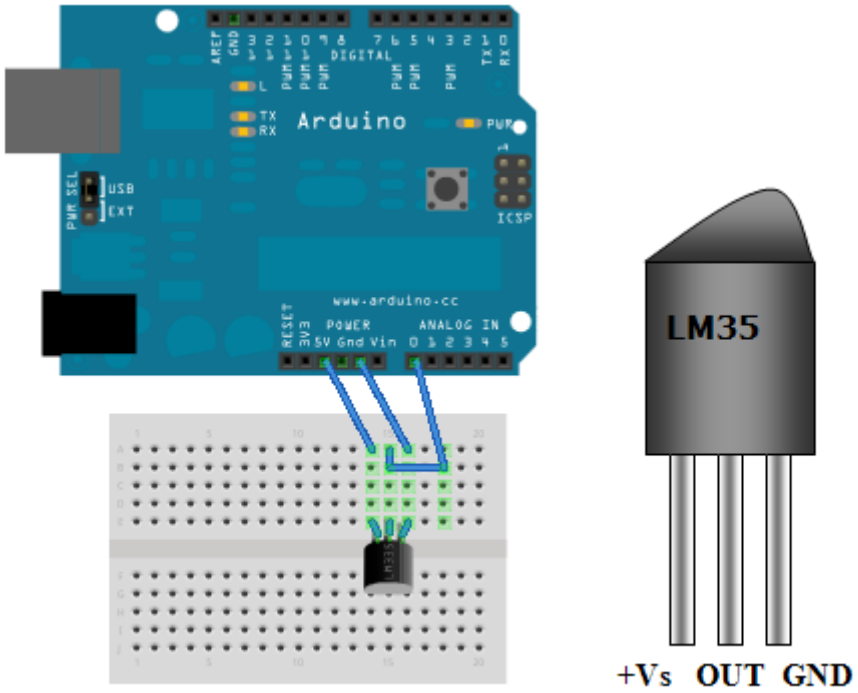
Termómetro con LM35 y Arduino.

Descripción de la práctica:

Utilizando un Arduino y un LM35 se planea medir la temperatura ambiente. El objetivo es lograr la comunicación serial.

El LM35 mide la temperatura y, en base a ella, regresa un valor de voltaje al Arduino. Posteriormente, el voltaje es convertido a grados centígrados para finalmente ser enviado a la computadora y ser mostrada en la interfaz creada en LabWindows.

Esquemático:



Material Utilizado:

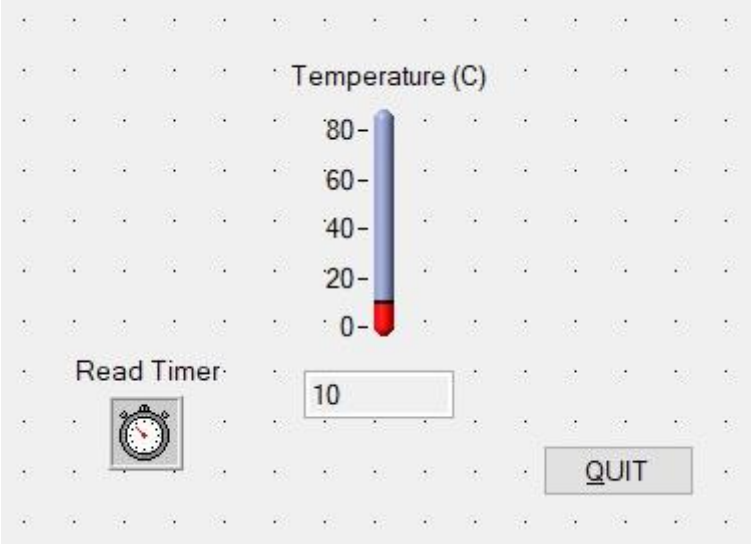
- 1 Arduino.
- 1 LM35.
- 1 protoboard
- Un cable USB para el Arduino-PC.
- Jumpers.

Interfaz generada en LabWindows:

Con la ayuda del software LabWindows, se creó una interfaz GUI mediante la cual se puede apreciar la temperatura ambiente, obtenida con ayuda del LM35. Se adjunta una imagen de la interfaz:

Resultados obtenidos:

Después de conectar el LM35 a los puertos correspondientes del Arduino (establecidos en el código), se pudo obtener la temperatura ambiente desplegada en la interfaz de LabWindows. A continuación, se adjunta una imagen del resultado:



Conclusiones (aprendizaje obtenido):

Esta práctica representó un circuito bastante sencillo, gracias a la utilización del Arduino. El objetivo es, más bien, el lograr la comunicación serial con LabWindows y el poder crear la interfaz funcional.

Por lo tanto, nos ayudó a entender cómo crear la interfaz serial en LabWindows (UIR y código) y cómo representar los datos visualmente; como preparación para el proyecto del clúster de instrumentos.

Resultó ser una práctica sencilla pero con gran aprendizaje.

Problemas enfrentados y como fueron resueltos:

- El principal problema fue la comunicación para la interfaz GUI con el IO14.
- El segundo problema enfrentado fue la saturación al tratar de desplegar el valor de la temperatura. Esto, a su vez, depende del problema tres; pero, también, generado por el tipo de variable utilizado.
- El tercer problema enfrentado fue la conversión a grados centígrados.

Código utilizado en Arduino:

```
char temp;
int tempPin = 0;
void setup()
{
  Serial.begin(9600);
}void loop()
{
  temp = analogRead(tempPin);
  temp = temp * 0.48828125;
  Serial.print(temp);
  delay(1000);
}
```

Código utilizado en LabWindows:

```
//==============================================================================
===
//
// Title:       IO14
// Purpose:           A short description of the application.
//
// Created on:  10/19/2016 at 4:23:29 PM by Rick Swenson.
// Copyright:   ITESM. All Rights Reserved.
//
//==============================================================================
===

//==============================================================================
===
// Include files

#include <rs232.h>
#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
#include "IO14.h"
#include "toolbox.h"

//==============================================================================
===
// Constants
// COMM Port assigned by Windows when IO-14 is plugged in
// Obtain this number from device manager
#define  COM_PORT 17

//==============================================================================
===
// Types

//==============================================================================
===
// Static global variables

static int panelHandle = 0;

//==============================================================================
===
// Static functions

//==============================================================================
===
// Function prototypes. User defined functions
// Initialize the communications port and do the initial
// Handshaking
int initialize(void);
```

```
//==============================================================================
===
// Global variables
// Flag thar informes if the Initialization has taken place
// at the beginning. When 0, no initialization has ocurred.
// When 1, initialization took place.
unsigned char Enable = 0;

//==============================================================================
===
// Global functions

/// HIFN The main entry-point function.
int main (int argc, char *argv[])
{
      int error = 0;

      /* initialize and load resources */
      nullChk (InitCVIRTE (0, argv, 0));
      errChk (panelHandle = LoadPanel (0, "IO14.uir", PANEL));

      /* display the panel and run the user interface */
      errChk (DisplayPanel (panelHandle));
      errChk (RunUserInterface ());

Error:
      /* clean up */
      if (panelHandle > 0)
            DiscardPanel (panelHandle);
      return 0;
}

//==============================================================================
===
// UI callback function prototypes

/// HIFN Exit when the user dismisses the panel.
int CVICALLBACK panelCB (int panel, int event, void *callbackData,
            int eventData1, int eventData2)
{
      if (event == EVENT_CLOSE)
            QuitUserInterface (0);
      return 0;
}

// User defined funtion to initialize Comm port and
// perform initial handshaking, only done once!
int initialize(void)
{
      // Declare local variables
      // Setting up the IO-14 handshaking commands
/* char init[7] = {0x2B,     // Flash LED
                           0x63,     // Select Channel 12 (LM35 is connected
here
```

```
                              0x68,     // Set binary communications
                              0x69,     // Internal 5V reference
                              0x2A,0x09,// 5V reference for ADC
                              '\0'     // Character array in C end will null
                            };
      int bytes_sent,i;   */

      // Open communications port
      OpenComConfig (COM_PORT, "", 9600, 0, 8, 1, 10, 10);

      //Send the initialization comands to the IO-14
/*    for (i=0; i<strlen(init); i++){
        bytes_sent = ComWrtByte (COM_PORT, init[i]);
        Delay(0.1);
      }           */


      // It occure once, so flag it
      Enable = 1;

      return 0;

}

// This function will be called every Timer Click
int CVICALLBACK TimerCallBack (int panel, int control, int event,
                                          void *callbackData, int eventData1,
int eventData2)
{
      // Define local variables
      //char ReadTemp [2] = {0x76,'\0'}; // Command to request a temperature
reading
      unsigned int Temp[2] = {0,0};
      int bytes_sent,i;
      float Tmp;
      char Value;

      // Disable NON FATAL RUN TIME Library ERROR
      SetBreakOnLibraryErrors(0);

      // Check wheter we have to open COMMs port and Initialize it
      if (Enable == 0)
        initialize();
      // Comm port is open and initialized
      else {

        // Request a temperature reading
        /*for (i=0; i<strlen(ReadTemp); i++){
             bytes_sent = ComWrtByte (COM_PORT, ReadTemp[i]);
             Delay(0.1);
        }  */

        // Read the temperature sent by the IO-14
        for (i=0; i<2; i++) {
```

```
            Temp[i] = ComRdByte (COM_PORT);
            Delay(0.1);
        }

        // Convert the binary values obtained from the IO-14
        // and convert them using the formula found in the
        // pdf file
        /*Tmp = (float)Temp[1] * 255.0 + (float)Temp[0];
        Value = Tmp/1023.0 * 5.0;
        Value = Value * 100.0;   */
        Value = Temp[0];

        // Sent Value to the Thermometer indicator
        SetCtrlVal(panelHandle,PANEL_TEMPERATURE,Value);
    }

    // Enable NON FATAL RUN TIME Library ERRORS
    SetBreakOnLibraryErrors(1);

    return 0;
}


int CVICALLBACK QuitCallback (int panel, int control, int event,
                                        void *callbackData, int eventData1,
int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            // Close Communication port
            CloseCom (COM_PORT);
            QuitUserInterface (0);
            break;
        case EVENT_RIGHT_CLICK:

            break;
    }
    return 0;
}
```