UNIVERSIDAD FAVALORO

24/11/2020

Sistema de monitoreo de temperatura para pacientes con COVID-19

Campo Francisco, Galmarini Felipe Ignacio, Stanganelli Ezequiel, Vazquez Constanza.
Electrónica Digital II, FICEN, Universidad Favaloro.

Resumen: En este informe mostraremos los códigos y esquemáticos para la reproducción de un sistema de monitoreo de temperatura para pacientes con COVID-19 utilizando el microcontrolador PIC16F877, un módulo de bluetooth y un LCD Graficador.

I. Introducción:

El PIC16F877 es un microcontrolador que tiene un circuito integrado que puede ser programado de muchas formas diferentes.

El sensor de temperatura es un dispositivo que transforma una señal analogica a una señal eléctrica que puede ser procesada en el microcontrolador y luego ser transformada en una señal digital.

Las temperaturas que maneja el sensor LM35 van desde -55ºC (-550mV) a 150ºC (1500 mV) con una precisión de 1°C que equivale a 10mV, entonces para poder convertir el valor analógico que devuelve el sensor se debe hacer el siguiente cálculo:

Temperatura = Valor * 5 * 100 / 1024

Ante la necesidad de mantener un monitoreo constante de la temperatura de pacientes infectados con este virus, ideamos un sistema para suplir con esta demanda, utilizando un sensor de temperatura, un microcontrolador, una pantalla GLCD y un módulo de bluetooth para transferir los datos a una computadora y así poder realizar un manejo de estos dependiendo de la necesidad de quien los reciba.

II. Materiales:

- PIC16F877
- Sensor de temperatura LM35
- Display LCD LGM12641BS1R
- Resistencias: 100 Ω, 2x10 kΩ
- Potenciometro 20 kΩ
- Transistor bipolar PNP
- Buzzer
- Módulo Bluetooth HC-05
- Fuente regulada de 5V y 12V
- Cables
- 2 Pulsadores

III. Procedimiento:

Utilizamos un software de simulación de circuitos electrónicos ISIS de Proteus 8 para diseñar el circuito de medición de temperatura a partir del sensor LM35 y el microcontrolador PIC16F877 que se puede observar en la Figura 1. Se realizan las conexiones correspondientes entre el PIC, los materiales y los módulos previamente enumerados. Una vez conectados correctamente lo siguiente es programar en el Pic C Compiler y luego simularlo en Proteus. Estas conexiones se pueden observar más detalladamente en el Anexo.

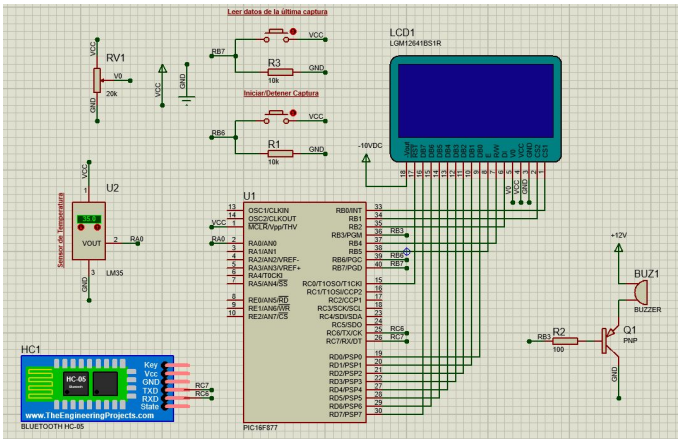

Figura 1. Vista general de todo el circuito.

A medida que se están realizando las mediciones se envían las temperaturas, mediante el uso del protocolo RS232, a través del módulo bluetooth HC-05. Para emular la comunicación COM, se utilizó un programa llamado *Virtual Serial Ports Tools* de la compañía *HHD Software Ltd.* Dentro de este software se configuró el COM1 y COM2 como puentes locales (Local Bridges) para que se efectuará la transmisión de los datos del PIC16F877 al software que los recibía. (**Ver Figura 10 Anexo**)

Una vez finalizada la captura de datos se procede a guardar los 80 datos finales (registrados cada 3 segundos) en la memoria EEPROM del PIC.

Además se programa una de las interrupciones, disparada por el botón conectado al pin B7 (**Ver Figura 9 Anexo**), para hacer la lectura de los datos almacenados en la EEPROM y graficarlos en el display GLCD.

IV. Resultados:

A continuación se muestran las líneas de código programadas en el programa PIC C Compiler.

Inicio del Código

"main.h"

```
#include <16F877.h>
#device ADC=10
#use delay(crystal=4000000)
#use rs232(baud=9600,
xmit=PIN_C6,rcv=PIN_C7, bits=8, parity=N)
```

"main.c"

```
#include <main.h>
#fuses XT, NOWDT,NOPROTECT,NOLVP
#include <HDM64GS12.c> //Manejo del
display grafico
#include <graphics.c> //Funciones para
dibujar y escribir en el display
#include <srcomLIB.c> //Funciones para el
envio datos por serial

#byte trisb=0x86

///Defines

#define escala -2 //Escala negativa para
que crezca hacia arriba.
#define limpiarGrafico
limpiarPorcion(0,20,128,44) //Limpia el
area del grafico
#define offset 120 //es el offset para la
altura de la grafica
#define TMR1 0x0BDC //Este es el valor
para 500ms en el timer1
(0,5s=(4/4000000)8(65536-TMR1))=>TMR1=303
6
#define TMR0 0x40 //Este es el valor para
50ms en el timer0
#define nMediciones 80 //Numero de
mediciones

///Fin Defines

///Variables Globales

int x=0; //Posicion inicial de x para
graficar la temperatura
float y; //Temperatura anterior
float t; //Temperatura actual
```

```
int16 iAn; //Lectura del sensor de temperatura
int registro[nMediciones]; //registro de las ultimas nMediciones mediciones
int1 habilitarLectura=0; //Variable para habilitar o deshabilitar la captura de datos del sensor
int ciclos=10; //Variable que cuenta los ciclos que debe hacer el timer0 antes de ejecutar su codigo
int ciclosT1=6; //Variable que cuenta los ciclos que debe hacer el timer1 antes de ejecutar su codigo
short int btST = 1;
char r_;

///Fin Variables Globales

///Funciones

void limpiarPorcion(int x1, int y1, int x2, int y2){ //(x1, y1) = posicion del primer pixel. (x2, y2) = cantididad de pixeles hacia la derecha y hacia abajo
   for(int i=x1;i<x1+x2;i++)
      for(int j=y1;j<y1+y2;j++)
         glcd_pixel(i, j, OFF);
//Apagamos el pixel.
}

void nuevaLinea(float temp){ //Funcion para graficar las nuevas lineas de temperatura
   int x1=x+1;
   if(x1>=128){ //Revisamos si sobrepasamos el tamaño de la pantalla
      limpiarGrafico; //Limpiamos el area del grafico
      x=0;
      x1=1;
   }
   glcd_line(x, (y*escala)+offset, x1, (temp*escala)+offset, ON);
//multiplicamos por la escala para que la grafica entre en el area del grafico. El offset esta explicado arriba
   y=temp;
   x=x1;
}

void toggleBT(char c_) {
   if(c_ == 'a') {
      btST = !ON;

glcd_text57(110,0,(char*)"BT",1,ON);
//Mostramos BT
   }
   else {
      btST = !OFF;
      glcd_text57(110,0,(char*)"BT",1,OFF); //Apagamos BT
   }
}

///Fin Funciones

///Interrupciones

#INT_TIMER0
void TIMER0_isr(){ //Timer para hacer sonar el buzzer cada 500ms
   ciclos--;
   if(ciclos==0){
      ciclos=10;
      output_low(PIN_B3);
      delay_ms(50);
      output_high(PIN_B3);
   }
   set_timer0(TMR0);
}

#INT_TIMER1
```

```
void TIMER1_isr(){ //Timer para almacenar, en un array, la temepratura actual cada 3 segundos
   ciclosT1--;
   if(ciclosT1==0){
      ciclosT1=6;
      //Aca hay que hacer el guardado en sd
      //Con este bloque de codigo guardamos en un array las ultimas N mediciones
      //Las cuales podemos guardar despues donde querramos
      for(int i=0; i<nMediciones-1;i++)
         registro[i]=registro[i+1];
      registro[nMediciones-1]=iAn;
   }
   set_timer1(TMR1);
}

#INT_RB
void RB_isr(){ //Prueba de interrupciones
   if(input(PIN_B6)){ //Habilita o deshabilita la captura de datos
      habilitarLectura=~habilitarLectura;
      disable_interrupts(INT_TIMER0); //Desabilitamos el timer0 para que no siga funcionando el buzzer si nos quedamos en temperaturas fuera de los aceptable
      if(habilitarLectura){
         glcd_text57(128/2-30, 0, (char*)"Capturando", 1, ON);
         sprintf(str, "%3.2fC", y);
         glcd_text57(12*6, 10, str, 1, ON); //Escribimos la temperatura.
         enable_interrupts(INT_TIMER1); //Habilitamos el timer1 para el almacenamiento de los valores de temperatura
         set_timer1(TMR1); //Seteamos el timer1
      }
      else {
         glcd_text57(128/2-30, 0, (char*)"Capturando", 1, OFF);
         glcd_text57(128/2-30, 0, (char*)"Limpiando", 1, ON);
         limpiarGrafico; //Limpiamos el grafico
         glcd_text57(128/2-30, 0, (char*)"Limpiando", 1, OFF);
         sprintf(str, "%3.2fC", t);
         glcd_text57(12*6, 10, str, 1, OFF);
         x=0; //Volvemos a posicionar x al inicio del GLCD
         disable_interrupts(INT_TIMER1);
         glcd_text57(128/2-30, 0, (char*)"Guardando", 1, ON);
         for(int pos=0; pos<nMediciones; pos++){
            write_eeprom(pos, registro[pos]); //Escritura de datos en la EEPROM
         }
         glcd_text57(128/2-30, 0, (char*)"Guardando", 1, OFF);
      }
   }
   if(input(PIN_B7)){ //Leectura de datos de la EEPROM
      habilitarLectura=0;
      glcd_text57(128/2-30, 0, (char*)"Capturando", 1, OFF);
      disable_interrupts(INT_TIMER1);
      disable_interrupts(INT_TIMER0);
      for(int pos=0; pos<nMediciones; pos++){ //Leemos los datos de la EEPROM y graficamos los mismos.
         registro[pos]=read_eeprom(pos); //Lectura de la EEPROM
         float dato=(5.0*registro[pos]*100.0)/1024.0;
```

```
//Convertimos los datos almacenados a
flotantes
         if(dato>=28 && dato<=50)
            nuevaLinea(dato);
         if(dato<28)
            nuevaLinea(29);
         if(dato>50)
            nuevaLinea(50);
         registro[pos]=0; //Limpiamos el
registro de las temperaturas
      }
      glcd_text57(128/2-30, 0,
(char*)"Guardando", 1, ON);
      for(int pos=0; pos<nMediciones;
pos++){
         write_eeprom(pos,
registro[pos]); //Escritura de datos en
la EEPROM
      }
      glcd_text57(128/2-30, 0,
(char*)"Guardando", 1, OFF);
   }
}

#INT_RDA
void serial_interrupt() {
   r_ = getc();
   toggleBT(r_);
}

///Fin Interrupciones

void main()
{
   trisb=0b01000000;

   setup_adc_ports(AN0); //seteamos el
pin A0 como analogico
   setup_adc(ADC_CLOCK_INTERNAL);
//Establecemos el reloj interno

setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256)
; //setup del timer0

setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
//setup del timer1
   set_timer0(TMR0); //seteamos el timer0
   set_timer1(TMR1); //seteamos el timer1

   glcd_init(ON); //Inicializamos el lcd

   enable_interrupts(INT_RB);
//Habilitamos las interrupciones del
RB4-7

enable_interrupts(INT_RDA);//Habilitamos
las interrupciones del RDA
   enable_interrupts(GLOBAL);
//Habilitamos las interrupciones globales


glcd_text57(0,10,(char*)"Temperatura:",1,
ON); //Escribimos el texto "Tempreatura:"
en la posicion 0,10
   glcd_line(0, 8, 128, 8, ON);
//Pintamos una linea por debajo del barra
de notificaciones
   glcd_line(0, 19, 128, 19, ON);
//Pintamos una linea por debajo de la
temperatura

   set_adc_channel(0); //Seteamos el
canal que vamos a leer
   delay_ms(250); //Esperamos 250 ms para
tener una captura mas real de la primera
temperatura
   iAn=read_adc(); //Levantamos el dato
   y=(5.0*iAn*100.0)/1024.0; //Lo
convertimos a temperatura. Esta es la
primera captura
   delay_ms(250);
```

```
   set_SERIALsr();

   while(TRUE) {
      if(habilitarLectura){
         set_adc_channel(0); //Seteamos el canal que vamos a leer
         delay_us(10); //Esperamos 10 us
         iAn=read_adc(); //Levantamos el dato
         t=(5.0*iAn*100.0)/1024.0; //Lo convertemos a temperatura

         if(btST)
            sendINT('t', iAn, SEP); //Funcion para enviar datos via el Modulo Bt

         if(t!=y){ //Si t es != al dato anterior refrescamos la temperatura y la enviamos al bluetooth
            sprintf(str, "%3.2fC", y); //Convertimos la temperatura float en un char*
            glcd_text57(12*6, 10, str, 1, OFF); //borramos del lcd la temperatura anterior
            sprintf(str, "%3.2fC", t); //Convertimos la temperatura float en un char*
            glcd_text57(12*6, 10, str, 1, ON); //Escribimos la temperatura.
         }
         if(t<=40 && t>=35)

disable_interrupts(INT_TIMER0);
         if(t>40 || t<35)

enable_interrupts(INT_TIMER0);
         if(t<=50 && t>=28){ //Si t es menor o igual a 50 la agregamos al grafico
            nuevaLinea(t); //Dibujamos la nueva linea en el grafico.
            glcd_text57(3, 0, (char*)"T>50", 1, OFF); //ocultamos la notificacion de t>50
            glcd_text57(3, 0, (char*)"T<28", 1, OFF); //ocultamos la notificacion de t>50
         }
         if(t>50){
            glcd_text57(3, 0, (char*)"T>50", 1, ON); //Esto significa que no graficamos temperaturas superiores a 50
            y=t; //igualamos el dato anterior al valor de temperatura para no refrescar otra vez el valor de la temperatura
         }
         if(t<28){
            glcd_text57(3, 0, (char*)"T<28", 1, ON); //Esto significa que no graficamos temperaturas superiores a 50
            y=t; //igualamos el dato anterior al valor de temperatura para no refrescar otra vez el valor de la temperatura
         }
         delay_ms(250);
      }
   }
}
```

“srcomLIB.c”

```
#include <stdlibm.h>
#include <string.h>
#define SEP '\n'
#define END '\0'

struct SERIAL_LIB {
    char* sSerial;
```

```
}; typedef struct SERIAL_LIB SERIALsr;

SERIALsr serialData;
char str[9] = "\0";

// SEND SYSTEM
//Any of this functions, send data char by char
void set_SERIALsr() { //INIT Declaration
    serialData.sSerial = malloc(sizeof(char));
}
void sendFT(char t , float v, char a) {
//t: Temp, Humity, etc || v: Float value || a: SEP or END
    sprintf(str, "%c%f%c", t, v, a);
    for(int i = 0; str[i] != a; i++)
        printf("%c", str[i]);
    printf("%c", a);
}
void sendINT(char t , int v, char a) {
//t: Temp, Humity, etc || v: Int value || a: SEP or END
    sprintf(str, "%c%d%c", t, v, a);
    for(int i = 0; str[i] != a; i++)
        printf("%c", str[i]);
    printf("%c", a);
}
void sendSTR(char t , char v[], char a) {
//t: Temp, Humity, etc || v: String value || a: SEP or END
    sprintf(str, "%c%s%c", t, v, a);
    for(int i = 0; str[i] != a; i++)
        printf("%c", str[i]);
    printf("%c", a);
}
void sendDATA(char s_[], char a) { //v: Int value || a: SEP or END
    for(int i = 0; s_[i] != a; i++)
        printf("%c", s_[i]);
    printf("%c", a);
}
```

```
// END SEND SYSTEM
// READ SERIAL
int getArrSize() { return strlen(serialData.sSerial); } //Get Size of array when reading COM Serial
int insert(char c_) { //Insert read value into a dynamic array
    int i = 0;
    serialData.sSerial = realloc(serialData.sSerial, sizeof(char)*(i + 1));
    if(serialData.sSerial != NULL) {
        i++;
        *(serialData.sSerial + getArrSize() - 1) = c_;
        return TRUE; //Data was able to save
    } return FALSE; //No RAM able to increase array size
}
char* getSTR() { return serialData.sSerial; } //Return String from COM reading
// END READ SERIAL
```

Fin del Código

En la Figura 2. podemos observar el resultado de la gráfica de la temperatura a medida que avanza el tiempo.

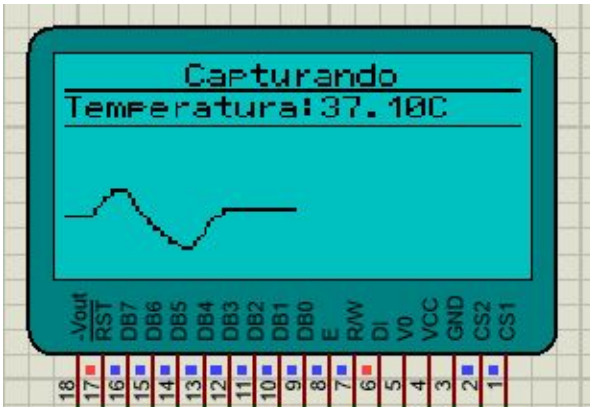

Figura 2. Display graficando temperatura en tiempo real.

Ante temperaturas mayores a 40 °C y menores a 35 °C se dispara una alerta sonora a través del Buzzer. Además si las temperaturas superan los 50 °C o está por debajo de los 28°C se mostrará una notificación (**Ver Figura 3**) en el margen superior izquierdo del display indicando que se detuvo la gráfica pero la medición continúa.

Figura 3. Notificaciones de temperaturas.

A continuación se muestra el diagrama de bloques (**Figura 4)** que representa esquemáticamente el proceso que realiza el dispositivo al realizar las mediciones.

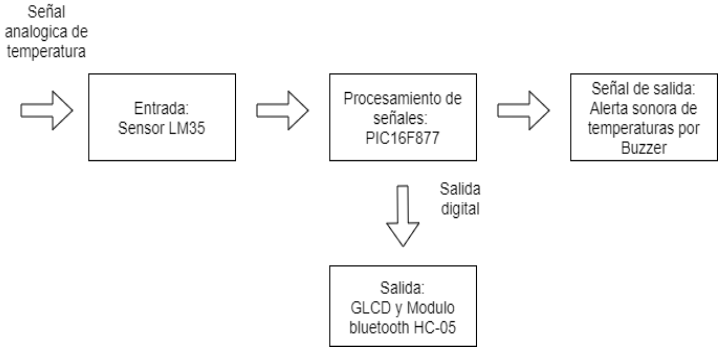

Figura 4. Diagrama de bloques del circuito.

V. Conclusiones

Para finalizar con este proyecto podemos concluir que con la utilización de un PIC de mayor memoria RAM el dispositivo podría funcionar de manera más fluida, así como también podríamos haber guardado un mayor volumen de datos en una memoria externa (microSD). Así mismo, el uso de los puertos virtuales con una simulación del PIC, no nos presenta la misma efectividad que una conexión real entre el microcontrolador con otro dispositivo (smartphone, notebook, etc), ya que nos da más latencia y dificultad para conectar entre sí. Ignorando lo recientemente mencionado, pudimos realizar el proyecto de una forma eficiente y que cumple con los requisitos esperados para solucionar esta problemática.

VI. Anexo

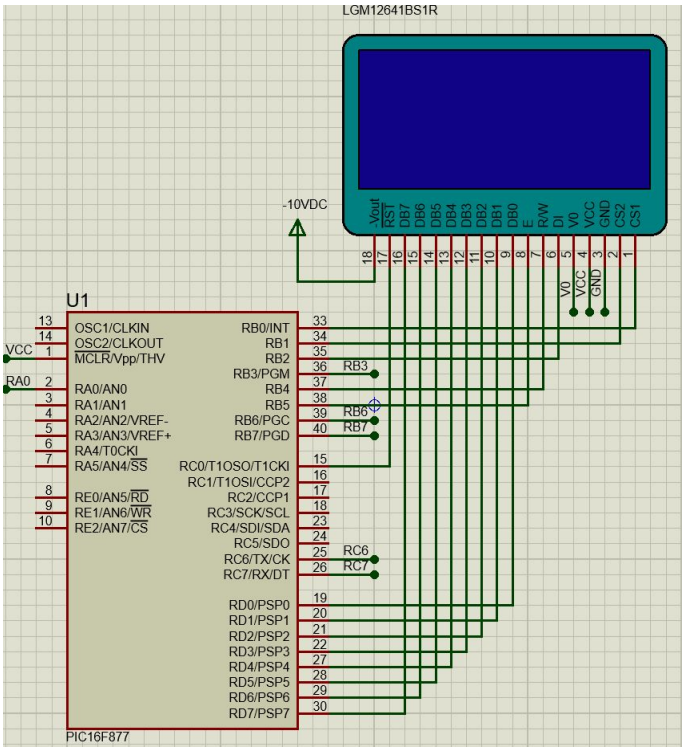

Figura 5. Conexión entre PIC y Display.

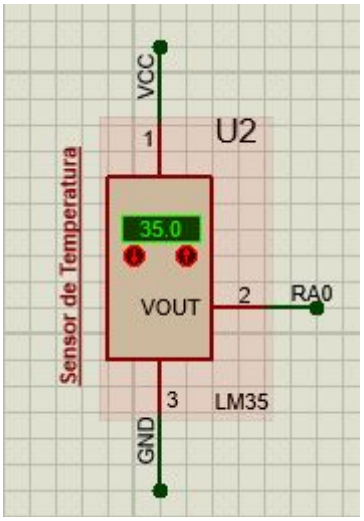

Figura 6. Conexión del sensor de temperatura con el PIC.

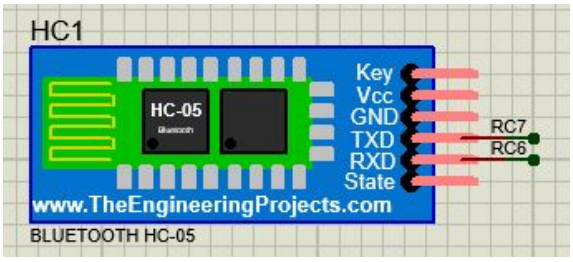

Figura 7. Conexión entre Módulo Bluetooth y PIC.

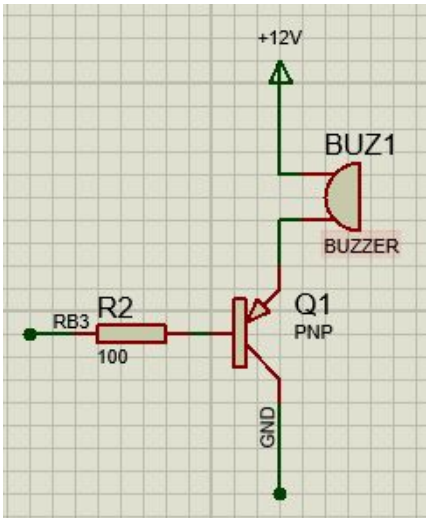

Figura 8. Conexión entre Buzzer y PIC.

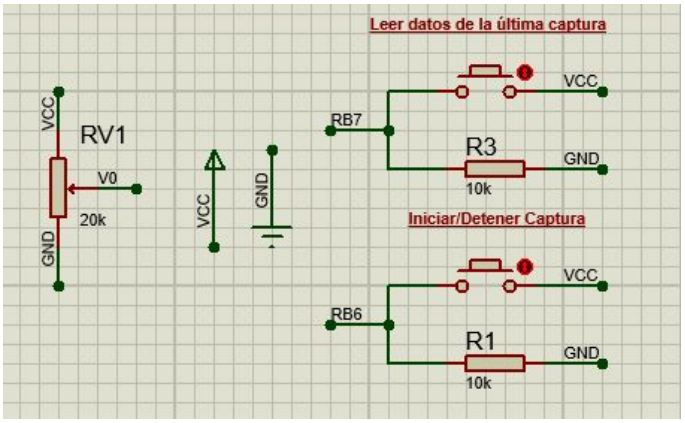

Figura 9. Conexión entre pulsadores y la fuente con el PIC.

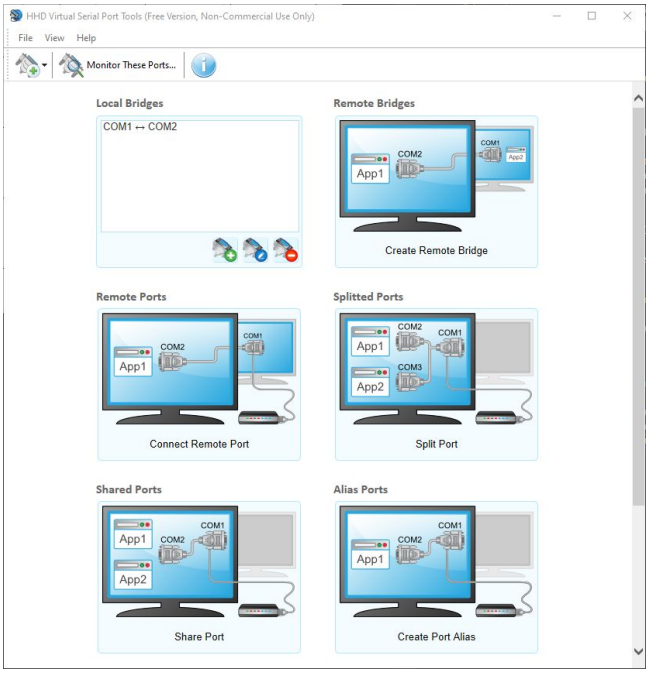

Figura 10. Software utilizado para emular los Serial Ports en la computadora.

- https://gitlab.com/pofbattousai/serial-data-sender-and-receiver-for-pic/-/blob/master/PICxxF/srcomLIB.c (Programación de la libreria "srcomLIB" por Ezequiel A. Stanganelli)

VII. Bibliografía:

- https://www.instructables.com/LM35-Temperature-Sensor-Simulation/ (Conexión del sensor de temperatura LM35)
- https://cifpn1sjd2.wordpress.com/2011/01/20/ejercicio-con-lcd-grafico-usando-el-pic18f4550/ (Conexión del display LGM12641BS1R)
- https://www.electronicslovers.com/2017/01/Add-Bluetooth-Module-Library-into-Proteus-Software.html (Libreria de proteus con el Modulo Bluetooth HC-05)
- https://freevirtualserialports.com/ (Programa para virtualizar el puerto COM)
- https://controlautomaticoeducacion.com/microcontroladores-pic/14-conversion-analogodigital-ad/ (Información del conversor analógico/digital)