Manual rápido de integración de c++ con Proteus

Elaborado por Mag. Juan M. Chau

1 Instalación y uso de Virtual Serial Port Driver

Descargar el instalador desde este enlace. Ejecutar el programa y seguir todas las indicaciones, empezando con la de la Figura 1.

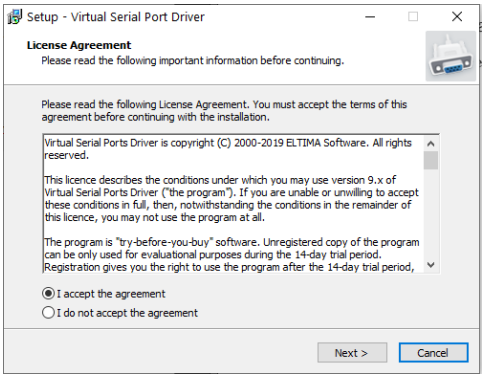

Figure 1: Ventana de instalación de Virtual Serial Port Driver.

Ejecutar el programa Virtual Serial Port Driver. Seleccionar ambos puertos que se desean utilizar para realizar la comunicacin como se puede apreciar en la Figura 2, luego añadir el par virtual a la lista. Recordar estos valores para uso futuro.

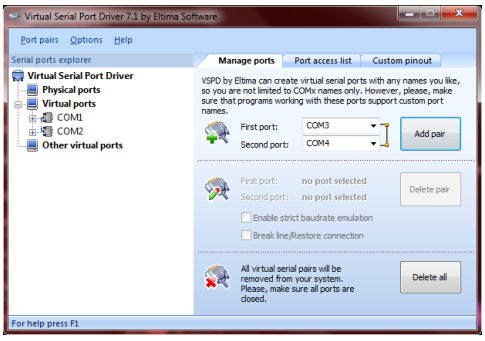

Figure 2: Ventana de configuración de Virtual Serial Port Driver. Se puede apreciar que el par virtual COM1-COM2 ya ha sido agregado.

2 Conexión RS232 utilizando c++

Para poder utilizar la comunicación de los puertos unidos basta con utilizar una simulación básica en Proteus. La Figura 3 muestra un modelo sencillo que utiliza Arduino para encender un LED cuando se recibe la instrucción.

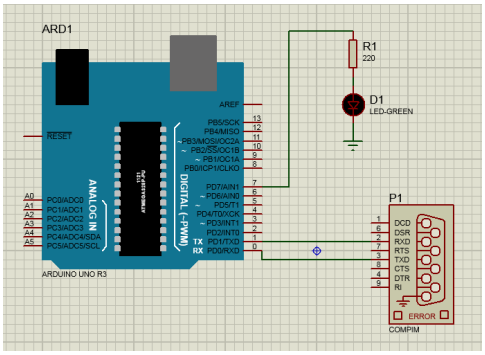

Figure 3: Modelo básico de Arduino en Proteus.

Se debe tener en cuenta que uno de los puertos seleccionados anteriormente para la comunicación (COM2) debe estar asignado dentro de la simulación, como se muestra en la Figura 4.

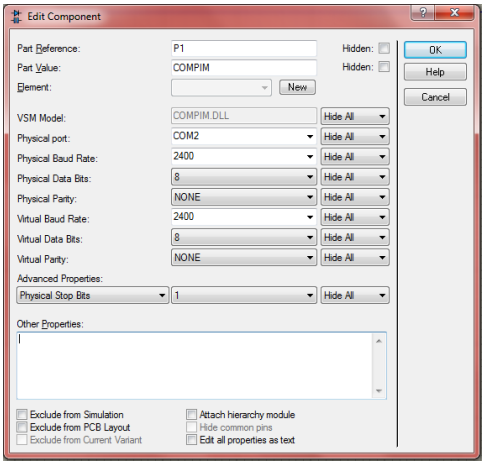

Figure 4: Configuración de puerto serial dentro de Proteus.

De igual manera, se requiere crear un objeto de tipo HANDLE en c++, que se asignará al segundo puerto seleccionado también anteriormente (COM1). Un ejemplo de configuración se puede apreciar en el bloque de código 1. El

objeto configurado permitirá leer y escribir datos utilizando las funciones *ReadFile()* y *WriteFile()*.

Listing 1: Configuración básica de puerto RS232

```
#include <iostream>
#include "Windows.h"

// Handle para el puerto
HANDLE serialHandle;

// Reemplazar COM1 con el nombre del puerto a utilizar
std::wstring portName = L"\\\\.\\COM1";

// Funcion para abrir el puerto
void openSerialPort( std::wstring portName,
    int baudRate = 9600, int byteSize = 8,
    int stopBits = TWOSTOPBITS, int parity = NOPARITY ) {

    // Asignacion de puerto a handle
    serialHandle = CreateFile( portName.c_str(),
        GENERIC_READ | GENERIC_WRITE, 0, 0, OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL, 0);

    // Configuracion basica
    DCB serialParams = { 0 };
    serialParams.DCBlength = sizeof(serialParams);
    GetCommState(serialHandle, &serialParams);
    serialParams.BaudRate = baudRate;
    serialParams.ByteSize = byteSize;
    serialParams.StopBits = stopBits;
    serialParams.Parity = parity;
    SetCommState(serialHandle, &serialParams);

    // Configuracion de timeouts
    COMMTIMEOUTS timeout = { 0 };
    timeout.ReadIntervalTimeout = 50;
    timeout.ReadTotalTimeoutConstant = 50;
    timeout.ReadTotalTimeoutMultiplier = 50;
    timeout.WriteTotalTimeoutConstant = 50;
    timeout.WriteTotalTimeoutMultiplier = 10;
```

```
37      SetCommTimeouts(serialHandle, &timeout);
38  }
39
40  // Funcion para cerrar el puerto
41  void closeSerialPort() {
42      CloseHandle(serialHandle);
43  }
```

Finalmente, se recomienda tener en cuenta que los parámetros de configuración de ambos puertos RS232 sean los mismos: paridad, tamaño de bits, velocidad de transferencia, etc., además de asegurarse de que el puerto se cierre antes de que el programa termine su ejecución.