UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Comunicaciones serie en Qt usando QextSerialPort.

Apellidos, nombre	Perles Ivars, Àngel (aperles@disca.upv.es)
Departamento	Informática de Sistemas y Computadores
Centro	Universitat Politècnica de València

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Resumen de las ideas clave

En este artículo se va a introducir el uso de una biblioteca que aporta la capacidad de realizar comunicaciones serie con Qt [1]. A continuación se resumen las ideas clave que se van a introducir:

- Cómo se obtiene la biblioteca.
- Cómo se configuran los proyectos para su uso.
- Cómo son las primitivas básicas de envío y recepción de datos.
- Ejemplos básicos de uso.

2 Introducción

Las normas de comunicación serie, como la RS-232 y la RS-485, permiten aprovechar una enorme cantidad dispositivos industriales que se pueden conectar a un computador industrial.

Para aprovechar este potencial desde una aplicación desarrollada en Qt es necesario incorporar una biblioteca de terceras partes y saber emplear las primitivas típicas de configuración de un puerto de comunicaciones y de envío/recepción de datos.

Este artículo presenta el ciclo completo de obtención, integración y desarrollo básico de una aplicación que emplee la biblioteca QextSerialPort [2] para hacer comunicaciones serie multiplataforma en Qt. Debe funcionar en Microsoft Windows, Linux, Mac OS X y FreeBSD.

Este artículo se ha preparado usando solo herramientas de programación libres. El tutorial se ha probado en Linux y en Microsoft Windows.

3 Objetivos

Una vez leído y practicado con este documento, el lector será capaz de:

- Emplear comunicaciones serie en sus aplicaciones Qt.

4 Desarrollo

4.1 Requisitos previos

Este artículo solo se puede seguir sí:

- Tienes nociones básicas de programación en Qt.
- Tienes nociones básicas sobre la interfaz serie RS-232.

Para el desarrollo de la experiencia será necesario, como mínimo:

- Qt 5.0.2 32 bits para Windows con compilador mingw [3]

- Biblioteca QextSerialPort [2]
- Un puerto serie en el computar, ya sea real o virtual (Bluetooth, USB, infrarrojos, …). Otra opción es un emulador de conexión null-modem, por ejemplo, com0com [4]

4.2 Lo primero: conseguir la biblioteca

La biblioteca está disponible como código fuente en un repositorio GIT de "googlecode". Si queremos descargar la última versión de desarrollo es necesario utilizar alguna herramienta de control de código. Por ejemplo, en Linux bastaría con abrir un terminal y hacer:

```
git clone https://code.google.com/p/qextserialport/
```

y se creará el directorio qextserialport con el código fuente del componente, los manuales y los ejemplos. En Windows, deberemos instalar antes alguna de las utilidades para Windows de control de versiones para GIT.

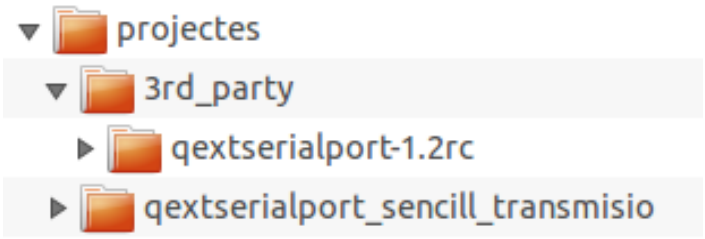
Una opción más sencilla que se nos ofrece es descargar una versión empaquetada.

En el área de descargas ("Downloads") de la página del proyecto podemos descargar la ***versión 1.2rc***, descomprimirla y colocarla en el directorio que nos interese.

4.3 Incorporando la biblioteca a la aplicación Qt

La manera más sencilla consiste en incluir el componente QextSerialPort dentro de nuestro proyecto. Para ello procederemos así:

- Dejar la carpeta QextSerialPort en un subdirectorio relativo a nuestra aplicación. Por ejemplo, como se indica en la siguiente imagen



- Incorporar el "Project include" de QextSerialPort en el archivo de proyecto de nuestra aplicación. Por ejemplo, el siguiente listado resalta la inclusión necesaria:

```
# Proyecto sencillo para qextserialport
# Configuracion para obtener una aplicacion de consola
QT       += core
QT       -= gui

TARGET = qextserialport_sencill_transmisio
CONFIG   += console
```

```
CONFIG   -= app_bundle
TEMPLATE = app
SOURCES += main.cpp

# inclusion del proyecto QextSerialPort dentro del nuestro
include(../3rd_party/qextserialport-1.2rc/src/qextserialport.pri)
```

- En cualquier módulo de nuestra aplicación que desee emplear el componente, realizar la inclusión de la siguiente manera:

```
#include <qextserialport.h>
```

4.4 Usando el componente

4.4.1 Creando una instancia

QextSerialPort es una clase Qt que se utilizará de la misma manera que cualquier otra.

Aunque en el manual se indican distintos tipos de constructores, por claridad, se empleará el constructor que menos configuraciones iniciales hace. Su prototipo es:

```
QextSerialPort::QextSerialPort();
```

Por ejemplo, se puede crear una instancia de la clase de la siguiente manera:

```
QextSerialPort *puerto; // puntero al objeto que maneja el puerto
void funcion(void)
{
    QextSerialPort *puerto = new QextSerialPort();
}
```

4.4.2 Configurando la conexión

El siguiente paso es configurar el comportamiento del objeto y el puerto a utilizar. Para ello, se pueden utilizar los siguientes métodos:

```
void setQueryMode (QueryMode mode);
void setPortName (const QString &name);
```

Con setQueryMode() se establece la manera en que se atenderá la información enviada/recibida por nuestro programa. Se tienen dos opciones: Polling (por encuesta) y EventDriven (por eventos).

Con setPortName() se establece el nombre del dispositivo serie a utilizar.

Por ejemplo, se podría hacer:

```
puerto->setQueryMode(QextSerialPort::Polling);
puerto->setPortName("COM1");
```

A continuación se deberán configurar los parámetros típicos de una conexión serie. Para ello, se pueden utilizar los siguientes métodos:

```
void setBaudRate (BaudRateType);
void setDataBits (DataBitsType);
void setParity (ParityType);
void setStopBits (StopBitsType);
void setFlowControl (FlowType);
```

En el manual del componente se pueden consultar los posible parámetros. Como ejemplo, para una configuración 9600-8N1 sin control de flujo se podría hacer:

```
puerto->setBaudRate(BAUD9600);
puerto->setDataBits(DATA_8);
puerto->setStopBits(STOP_1);
puerto->setParity(PAR_NONE);
puerto->setFlowControl(FLOW_OFF);
```

4.4.3 "Abriendo" y "cerrando" la conexión

Para que una aplicación pueda utilizar un determinado puerto serie, primero debe solicitarlo al sistema operativo para que se lo asigne. A esto se le suele denominar "abrir" el puerto serie.

De la misma manera, cuando ya no se usa una conexión serie, se debe devolver al SO para que otra aplicación pueda disponer de ella. A esto se le llama "cerrar" el puerto.

Para gestionar estas actividades se pueden utilizar los siguientes métodos:

```
bool open (OpenMode mode);
void close ();
```

La función open() permite "abrir" el puerto y saber si se ha logrado abrir. Como parámetro admite los valores ReadOnly (solo lectura/recibir), WriteOnly (solo escritura/enviar), ReadWrite (lectura/recibir y escritura/enviar).

Por ejemplo, se podría hacer:

```
printf("Abriendo el puerto ... ");
if (puerto->open(QextSerialPort::WriteOnly)) {
    printf("abierto!!!\n");
} else {
    printf("vaya, esto CASCA\n");
}
// ahora se usa el puerto
...
printf("Cerrando el puerto\n");
puerto->close();
```

4.4.4 Enviando y recibiendo datos

Una vez abierta la conexión, se pueden enviar datos empleando distintos métodos. Por ejemplo :

```
putChar (char c);
write (const QByteArray &byteArray);
write (const char *data, qint64 maxSize);
```

El método putChar() permite enviar un dato y el método write() permite enviar al buffer de salida una cadena o maxSize datos apuntados por el puntero data. Por ejemplo:

```
char *mensaje = "Bon dia pel mati";
char datos[] = {125,42,37,9};
//...
puerto->putChar(34);
puerto->putChar('A');
puerto->putChar(0xF3);
puerto->write(mensaje);
puerto->write(datos,4);
```

Para recibir datos se dispone también de varios métodos. Por ejemplo:

```
qint64 bytesAvailable ();
getChar (char *c);
read (qint64 maxSize);
read (char *data, qint64 maxSize);
```

Se debe tener en cuenta que las funciones de recepción son bloqueantes, es decir, si no se ha recibido nada, la función se queda a la espera de que llegue algo.

Esto no es adecuado en el caso de que se quiera que nuestra aplicación atienda a otras cosas simultáneamente, así que, para evitar el bloqueo, se puede usar el método bytesAvailable(), que devuelve el número de datos disponibles en el buffer de entrada o un valor <= 0 si no hay nada o hay problemas.

Con los métodos getChar() y read() podremos recoger uno o varios datos respectivamente del buffer de entrada. Por ejemplo:

```
while (1) {
    int num_datos;
    char dato;

    num_datos = puerto->bytesAvailable();
    if (num_datos > 0) {    // hay algo
        puerto->getChar(&dato);
        printf("Recibido el ascii %d (%c)\n",(int)dato, dato);
```

```
        }
    }
```

4.5 Ejemplos completos

4.5.1 Transmisión sencilla

```
/*  Proyecto sencillo para mostrar la funcionalidad de
qextserialport
   Ejemplo que transmite algo
*/

#include <QtCore/QCoreApplication>
#include <stdio.h>

#ifdef Q_OS_UNIX
#include <unistd.h> /* anyadir esta cabecera en los *nix para la
función sleep() */
#endif

// anyadimos la cabecera con la informacion sobre la clase
#include <qextserialport.h>

// nombre del puerto que deseamos abrir
#define PUERTO_NOMBRE "/dev/ttyS0"
//#define PUERTO_NOMBRE "/dev/ttyUSB0"      // adaptador USB-serie
//#define PUERTO_NOMBRE "COM1"
//#define PUERTO_NOMBRE "COM3"

QextSerialPort *puerto; // puntero al objeto que maneja el puerto

int main(int argc, char *argv[])
{
    //QCoreApplication a(argc, argv);       // no eliminar esto en
aplicaciones serias

    printf ("Creando instancia de QextSerialPort para abrir %s\n",
PUERTO_NOMBRE);
    QextSerialPort *puerto = new QextSerialPort();
    puerto->setPortName(PUERTO_NOMBRE);
```

```
    puerto->setQueryMode(QextSerialPort::Polling);    // modo mas
sencillo

    printf("Configurando puerto a 9600-8N1 sin control de
flujo\n");
    puerto->setBaudRate (BAUD9600);
    puerto->setDataBits(DATA_8);
    puerto->setStopBits(STOP_1);
    puerto->setParity(PAR_NONE);
    puerto->setFlowControl(FLOW_OFF);

    printf("Abriendo el puerto ... ");
    if (puerto->open(QextSerialPort::WriteOnly)) {
        printf("abierto!!!\n");
    } else {
        printf("vaya, esto CASCA\n");
    }

    printf("Enviar \"Hola, puerto serie\" al puerto serie\n");
    puerto->write("Hola, puerto serie");

    //esperar un poco a que salga todo por el puerto
#ifdef Q_OS_WIN
    Sleep(1);
#else
    sleep(1);
#endif

    printf("Cerrando el puerto\n");
    puerto->close();

    printf("Saliendo\n");

    //return a.exec();

    return(0);
}
```

4.5.2 Recepción sencilla

```
/*  Proyecto sencillo para mostrar la funcionalidad de qextserialport
   Ejemplo que recibe cosas
*/

#include <QCoreApplication>
#include <stdio.h>

// anyadimos la cabecera con la informacion sobre la clase
#include <qextserialport.h>

// nombre del puerto que deseamos abrir
#define PUERTO_NOMBRE "/dev/ttyS0"
//#define PUERTO_NOMBRE "/dev/ttyUSB0"      // adaptador USB-serie
//#define PUERTO_NOMBRE "/dev/rfcomm3"        // Bluettoth SPP
//#define PUERTO_NOMBRE "COM1"

QextSerialPort *puerto; // puntero al objeto que maneja el puerto

int main(int argc, char *argv[])
{
    //QCoreApplication a(argc, argv);       // no eliminar esto en aplicacione serias

    printf ("Creando instancia de QextSerialPort para abrir %s\n", PUERTO_NOMBRE);
    QextSerialPort *puerto = new QextSerialPort();
    puerto->setPortName(PUERTO_NOMBRE);
    puerto->setQueryMode(QextSerialPort::Polling);   // modo mas sencillo

    printf("Configurando puerto a 9600-8N1 sin control de flujo\n");
    puerto->setBaudRate (BAUD9600);
    puerto->setDataBits(DATA_8);
    puerto->setStopBits(STOP_1);
    puerto->setParity(PAR_NONE);
```

```
    puerto->setFlowControl(FLOW_OFF);

    printf("Abriendo el puerto ... ");
    if (puerto->open(QextSerialPort::ReadOnly)) {
        printf("abierto!!!\n");
    } else {
        printf("vaya, esto CASCA\n");
        exit(1);
    }

    printf("Entrando al bucle que mira si se van recibiendo
cosas:\n");
    while (1) {
        int num_datos;
        char dato;

        num_datos = puerto->bytesAvailable();
        if (num_datos > 0) {    // hay algo
            puerto->getChar(&dato);
            printf("Recibido el ascii %d (%c)\n",(int)dato, dato);
        }
    }

    // aqui no llegamos nunca, otro dia sera
    printf("Cerrando el puerto\n");
    puerto->close();

    printf("Saliendo\n");

    //return a.exec();

    return(0);
}
```

5 Cierre

A lo largo de este objeto de aprendizaje se ha visto cómo proporcionar la capidad de emplear una conexión serie mediante Qt.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

6 Bibliografía

[1] The Qt project. The Qt project foundation official page. 2013. Disponible en: URL: http://qt-project.org/.

[2] QextSerialPort libraries for serial communications using Qt. Disponible en: http://code.google.com/p/qextserialport/

[3] Qt 5.0.2 for Windows 32-bit (MinGW 4.7, 650 MB). 2013. Disponible en: http://qt-project.org/downloads

[4] com0com virtual null-modem emulator for Microsoft Windows. http://com0com.sourceforge.net/