

Manual de QPort

by

Sebastian Oldani

www.sebest.com.ar

Documento GPL

Indice de Contenidos

Que es QPort?.....	3
Como compilar QPort.....	3
Instalación en Linux.....	3
Instalación en Windows.....	3
Configurar QPort.....	4
Listas de Funciones.....	5
Listado de SIGNALS.....	5
Detalles de Funciones.....	6
Detalle de SIGNALS.....	10

Que es QPort?

QPort es un widgets para QT4, que permite realizar una comunicación a través del puerto serie de la PC. Lo positivo de este widgets es que no es necesario programar nada para realizar la comunicación, simplemente se puede arrastrar desde el QTDesigner y configurar desde allí todos sus parámetros. Además cuenta con unos slots que pueden ser conectados gráficamente.

QPort se distribuye con este manual y además el ejemplo que aquí mismo se trata. Para conocer más sobre QPort dirijase a www.sebest.com.ar o por mail sebastianoldani@gmail.com

Como compilar QPort:

QPort esta listo para ser compilado, no es necesario modificar nada, simplemente se tiene que descomprimir el archivo, entrar a la carpeta y ejecutar los comandos "qmake" y a continuación "make".

Un punto importante QPort no tiene un autoinstall porque esta pensado para utilizarse en aplicaciones muy particulares y no me parecio apropiado instalarlo en la PC, sin embargo voy a explicar los pasos a seguir para que el widgets este correctamente instalado.

Para poder instalar QPort es necesario bajar el archivador del código fuente. Hay que descomprimirlo y luego ejecutar dentro de la carpeta que se descomprimió las sentencias:

```
qmake  
make
```

No es necesario ser administrador para ejecutar estas sentencias. Esto sólo compila el widget.

Instalación en Linux:

Copiar el archivo "libqport.so" en la ubicación de las librerías "/usr/lib" y crear un enlace simbólico en los plugins del QTdesigner, que en ubuntu linux esta por defecto en la ubicación "/usr/lib/qt4/plugins/designer".

Además es necesario copiar los archivos de cabeceras "ManageSerialPort.h", "posix_qextserialport.h", "qextserialbase.h", "qextserialport.h", "ui_qport.h" en la carpeta de tus includes "/usr/include".

Una vez copiados estos archivos ya tienes la librería instalada, ahora para poder utilizarla es necesario incluir unos parámetros al archivo del proyecto.

Instalación en Windows:

Copiar el archivo "qport.dll" y por un bug copiar el mismo archivo, pero con el nombre "libqport.dll" en la ubicación de las librerías "X:\Windows\System32" y copiarlo en los plugins del QTdesigner, que depende de donde hayas instalado QT "X:\XX\qt4\plugins\designer".

Además es necesario copiar los archivos de cabeceras "ManageSerialPort.h", "posix_qextserialport.h", "qextserialbase.h", "qextserialport.h", "ui_qport.h" en la carpeta de tus includes dentro de qt4 o en cualquier ubicación que sea accesible desde el path.

Una vez copiados estos archivos ya tienes la librería instalada, ahora para poder utilizarla es necesario incluir unos parámetros al archivo del proyecto.

Crear una aplicación con QPort:

Primero unas definiciones que son necesarias para poder compilar el proyecto tanto en Linux como en Windows. Y además que agregues la librería "qport" a tu proyecto.

Acá se muestra el código que hay que incluir, tener en cuenta que si ya tienes algunas

librerías incluidas tendrás que agregarlas sobre las que ya tienes para no crear ningún conflicto.

```
////////////////////////////////código a incluir////////////////////////////////
win32 {
DEFINES += _TTY_WIN_QWT_DLL QT_DLL
HEADERS += win_qextserialport.h
SOURCES += win_qextserialport.cpp
}
unix {
DEFINES += _TTY_POSIX_
HEADERS += posix_qextserialport.h
SOURCES += posix_qextserialport.cpp
}
LIBS += -lqport
////////////////////////////////////////////////////////////////
```

Configurar QPort:

Como ya dije QPort es un widget para manejar el puerto serie de la PC, por ello es necesario configurarlo con algunos valores para poder realizar la comunicación que necesitamos.

Una vez instalado puedes empezar a utilizar QPort, para ello crea un proyecto con una ventana de dialogo como lo harías normalmente, ve a QTDesigner y arrastra el widget de QPort, que se encuentra en el menú con nombre “Sebest Widget” y de ahí arrastra el que se llama “QPort” a la ventana que estas creando, haz clic sobre él y modifica sus propiedades para que se adecuen a tu conveniencia.

Tienes que decir cual puerto vas a usar, puede ser si estas en linux “/dev/ttyS0” este es el primer serie, o “/dev/ttyS1” si es el segundo serie y así siguiendo ese orden. Si tienes un cable USB – SERIE deberas usar el puerto con nombre “/dev/ttyUSB0” o “/dev/ttyUSB1” dependiendo de cuantos tengas conectados y cual quieras usar.

En Windows la cosa cambia y es un poco más simple solo tienes que poner que Com es y listo, desde el administrador de dispositivos puedes ver como se llama el que quieras usar y listo, pueden ser “Com1” o “Com2” y así según cual uses.

Otro parametro es la velocidad de comunicación, esta velocidad debe ser la misma para los dos dispositivos que se van a comunicar, al igual que los demás parametros. No voy a explicar que son los demás parametros porque no viene al caso.

Además puedes colocar en el dialogo un checkbox para abrir y cerrar el puerto mediante signals y slots sin tener que codificar nada. Arrastras el checkbox, abris el editor de signals y slots creas una nueva con la siguiente información. Sender “checkBox”, Signal “toggled(bool)”, Receiver “qport”, Slot “open(bool)”.

Listas de Funciones:

```
bool open();
bool open(const QString &name, const BaudRateType baudRate, const DataBitsType dataBits, \
const ParityType parity, const StopBitsType stopBits, \
const FlowType flowControl, ulong seconds, ulong milliseconds);
void setBaudRate(const Speed baudRate);
void setDataBits(const DataBits dataBits);
void setParity(const Parity parity);
void setStopBits(const BitsStop stopBits);
void setFlowControl(const Flow flowControl);
bool isOpen();
void close();
void setPort(const QString &name);
QString getPort();
Speed getBaudRate(void);
DataBits getDataBits(void);
Parity getParity(void);
BitsStop getStopBit(void);
Flow getFlowControl(void);
int getBuffer(void);
void enableSending();
void disableSending();
bool isSendingEnabled();
char sendData(QByteArray &dataToSend);
QString FormatData(QByteArray data,char format,QString space=" ");
void stopSending();
void enableReceiving();
void disableReceiving();
bool isReceivingEnabled();
void stopReceiving();
```

Listado de SIGNALS:

```
void newData(const QByteArray &DatoR);
```

Detalles de Funciones:

bool open();

Abre el puerto con los parámetros por defecto o los cargados mediante las funciones de carga.

bool open(const QString &name, const BaudRateType baudRate, const DataBitsType dataBits, const ParityType parity, const StopBitsType stopBits, const FlowType flowControl, ulong seconds, ulong milliseconds);

Función sobre cargada, donde es posible determinar los parámetros antes de iniciar el puerto

void setBaudRate(const Speed baudRate);

Setea la el baud rate del puerto.

```
BAUD50, //POSIX ONLY
BAUD75, //POSIX ONLY
BAUD110,
BAUD134, //POSIX ONLY
BAUD150, //POSIX ONLY
BAUD200, //POSIX ONLY
BAUD300,
BAUD600,
BAUD1200,
BAUD1800, //POSIX ONLY
BAUD2400,
BAUD4800,
BAUD9600,
BAUD14400, //WINDOWS ONLY
BAUD19200,
BAUD38400,
BAUD56000, //WINDOWS ONLY
BAUD57600,
BAUD76800, //POSIX ONLY
BAUD115200,
BAUD128000, //WINDOWS ONLY
BAUD256000
```

void setDataBits(const DataBits dataBits);

Setea los bits de datos.

```
DATA_5,
DATA_6,
DATA_7,
DATA_8
```

void setParity(const Parity parity);

Setea la paridad

PAR_NONE,
PAR_ODD,
PAR_EVEN,
PAR_MARK, //WINDOWS ONLY
PAR_SPACE

void setStopBits(const BitsStop stopBits);

Setea los bits de Stop.

STOP_1,
STOP_1_5, //WINDOWS ONLY
STOP_2

void setFlowControl(const Flow flowControl);

Setea el control de flujo.

FLOW_OFF,
FLOW_HARDWARE,
FLOW_XONXOFF

bool isOpen();

Responde "true" si el puerto esta abierto

void close();

Cierra el puerto

void setPort(const QString &name);

Setea el nombre del puerto.

QString getPort();

Devuelve el nombre del puerto.

Speed getBaudRate(void);

Devuelve los baud rate del puerto

DataBits getDataBits(void);

Devuelve los bits de datos.

Parity getParity(void);

Devuelve la paridad.

BitsStop getStopBit(void);

Devuelve los bits de stop.

Flow getFlowControl(void);

Devuelve el tipo de control de flujo.

void setBuffer(int nBuff);

Setea la cantidad de caracteres que tienen que llegar antes de que se ejecute la signal. Si se ingresa "0" se ejecuta la signal ni bien llegue cualquier cantidad de bytes.

int getBuffer(void);

Devuelve la cantidad de caracteres que deben llegar al puerto antes de ejecutarse la signal.

void enableSending();

Activa la transmisión de datos.

void disableSending();

Desactiva la transmisión de datos.

bool isSendingEnabled();

Responde "true" si esta la transmisión esta activada.

char sendData(QByteArray &dataToSend);

Envia la cadena de datos *dataToSend* y responde:

return 1 : add OK
return 2 : port is not open
return 3 : sending is not enable

QString FormatData(QByteArray data,char format,QString space=" ");

Convierte la cadena *data*, según el formato *format* y lo separa con la cadena *space*,

format:

t	Devuelve la cadena en formato de texto, el valor de <i>space</i> no altera el resultado
h	Devuelve la cadena como hexadecimal separando los valores mediante <i>space</i>
o	Devuelve la cadena como octal separando los valores mediante <i>space</i>
b	Devuelve la cadena como binario separando los valores mediante <i>space</i>
d	Devuelve la cadena como decimal separando los valores mediante <i>space</i>

Ej:

QString str;
str=port->FormatData(dato,'h',' '); //dato=123

```
// str = 31 32 33
```

```
str=qport->FormatData(dato,'t," "); //dato=123  
// str = 123
```

void stopSending();

Deshabilita la transmisión de datos.

void enableReceiving();

Activa la recepción de datos.

void disableReceiving();

Deshabilita la recepción de datos.

bool isReceivingEnabled();

Responde “true” si la recepción esta activada.

void stopReceiving();

Detiene la recepción de datos.

Detalle de SIGNALS:

void newData(const QByteArray &DatoR);

Esta señal se activa cuando un dato llega al puerto serie. El dato es *DatoR*.