Fetch files from remote device via Bluetooth FTP

This article explains how to create a Bluetooth FTP connection to a target device, browse it's folders, select and retrieve its files.

Introduction

This article is a tutorial on how to connect to a remote device via Bluetooth using FTP protocol (OBEXFtp) using the QBluetoothZero library. Enables you to see the remote file system, select the files you want and receive them. It is very convenient in cases you use a computer as a master to control various things where in that case the computer connects to the device at will, receive the files it wants and later on operates on them.

The catch is that this functionality is implemented only for Windows platform. These extra functionalities that will be used in this article are part of the version 1.2.0 of the library.

Initial info we need to know

OK, lets see the whole picture. We have a Windows machine and we want to connect to an other remote device via Bluetooth. At first this requires the UID address of the remote device which is in a format as follows: 00:00:00:00:00:00. This UID (as the name suggests) is a unique identifier for each device. It can either be set manual to the program if someone knows it in advance or it can acquired through a Device discovery.

Example on how to initiate and handle a device discovery can be found here: Discovering Bluetooth devices with the QBluetooth library

Next thing is the Bluetooth service that is going to be used. Its name is OBEX File Transfer and its UID is 0x1106. Its support is a prerequisite in order to continue with the rest of the tutorial. This information can be acquired either from the device specifications that can be found in http://www.developer.nokia.com/Devices/Device_specifications/ or by initiating a Service Discovery on the remote device. For the later technique you can advice the following article: Discovering nearby Bluetooth services with the QBluetooth library.

New functions that will be used

As mentioned above, in version 1.2.0 there are a few additional functions regarding FTP and are implemented for the Windows platform. They are contained in the class QBtObjectExchangeClient.

- getWorkingPath() / setPath (const QString& newPath) : Set the remote path. Can either be absolute or relative to the current working directory
- initiateFolderBrowsing (const QString & path). Browser the selected folder. Returns a list of QBtRemoteFileInfo.
- locateFiles(QRegExp* regex, QString folder, bool enableRecursive) *: Select the files needed. The function takes arguments the regex through which the files will be selected, the starting remote folder from which to search and the option to operate recurscively in the child folders of the defined folder. Returns a list of QBtRemoteFileInfo.*
- batchFileRetrieval(const QList<QBtRemoteFileInfo*>& files, const QString destinationFolder, bool retrieveOnlyNewFiles); : Downloads the selected files from the remote device, stores them in the local folder passed as argument. Finally you can specify to download only files that haven't yet been fetched (default is to download all).

Get in the code

The following code can be found here: File:RemoteBrowserBluetoothFTP.zip

Supported commands:

- cd <DIR>
- ls <DIR>
- pwd
- select <REGEX> {<SRC_DIR>}
- selectImages <SRC_DIR>
- receive
- help

First create an instance of the browser and set manually the UID address of the device (or as described above it can be done through a device discovery). Then the *initializeConnectionData* will search the device for the specific service needed OBEX File Transfer. Then assuming the service is found we initiate the connection. If there was a problem with the address or the service then no connection will be made.

```
RemoteBrowser browser = RemoteBrowser(QBtAddress(QString("00:26:69:4F:F9:5E")));
browser.initializeConnectionData();
browser.connectToDevice();
```

Then the code continues with the main loop which in every loop requests input just like a command shell would do.

Operations `pwd`, `ls` are simple function calls so they are not going to be explained. *cd* supports both absolute and relative paths as well as the very well known "..". Now I would like to take a closer look at the `select` and `receive` functions.

Using the select function you can define a regex through which the files will be searched and the target folder of the search. If not folder is defined then it will search in the current working directory. This gives us the ability to either get one specific file entering as regex the specific name of the file (**select log.txt**) or a group of files which comply to the regex (**select .jpg**).

(C) Copyright Nokia 2013. All rights reserved.

```
if (command.contains("select "))
{
    QStringList commands = command.split(" ");
    if(commands.length() != 3 && commands.length()!=2)
    {
        cout << "Usage: select <REGEX> {<SRC_DIR>}" << endl;
        continue;
    }
    QString currentFolder = (commands.length() == 3) ? commands[2] : browser.pwd();
    QRegExp expression(commands[1]);
    expression.setCaseSensitivity(Qt::CaseInsensitive);
    selectedFiles = browser.locateFiles(currentFolder, &expression, false);
    browser.printFileListInfo(selectedFiles);
}
```

There is an already implemented operation as well called selectImages that selects all the images found in the target folder and all of its children.

Once the files are selected we can proceed in receiving them. The 3rd argument (boolean) specifies if all the files will be fetched or just the files that don't exist in the target folder (in this case C:\\receivedFiles). Hence this could be used if you want to retrieve only the non-fetched photos from your mobile phone to that folder.

```
browser.batchFileRetrieval(selectedFiles, "C:\\receivedFiles", true);
```

And you are done!

Hit **exit**, it gets disconnected and the program closes

Run it

After the successful compilation it is important not to forget to copy the QBluetoothZeroZero.dll in the directory where the .exe is placed. Be careful to copy the correct version of the QBluetoothZero.dll. The debug version for a Debug configuration and so on. (The QBluetoothZero.dlls are placed under the QBluetoothZero/bin folder in the projects folder) Here is a screenshot of all the above in action:

```
Starting Service Discovery on: 00:26:69:4f:f9:5e
Service discovery started
Service found: OBEX File Transfer, Uuid: 1106
Service discovery stopped
Connected to FTP server
\> ls
        \C:     D       0       20/11/1601
        \E:     D       0       20/11/1601
\> cd C:
\C:> ls
        \C:\Data        D       0       20/11/1601
        \C:\log.txt     _       0       20/11/1601
\C:> cd Data
\C:\Data> ls
        \C:\Data\.config        D       0       20/11/1601
        \C:\Data\Activenotes    D       0       20/11/1601
        \C:\Data\d2ddump        D       0       20/11/1601
        \C:\Data\Documents      D       0       20/11/1601
        \C:\Data\Games  D       0       20/11/1601
        \C:\Data\Images D       0       20/11/1601
        \C:\Data\Installs       D       0       20/11/1601
        \C:\Data\Juice  D       0       20/11/1601
        \C:\Data\lm     D       0       20/11/1601
        \C:\Data\logs   D       0       20/11/1601
        \C:\Data\MailForExchange        D       0       20/11/1601
        \C:\Data\Media Player Sync files        D       0       20/11/1601
        \C:\Data\My Videos      D       0       20/11/1601
        \C:\Data\Nokia  D       0       20/11/1601
        \C:\Data\Others D       0       20/11/1601
        \C:\Data\Sounds D       0       20/11/1601
        \C:\Data\SportsTracker2 D       0       20/11/1601
        \C:\Data\system D       0       20/11/1601
        \C:\Data\Videos D       0       20/11/1601
        \C:\Data\HXMDEngine_3_2.cfg     _       203     20/11/1601
        \C:\Data\MdfPluginArchive.txt   _       11585   20/11/1601
        \C:\Data\plugin_archive.txt     _       4369    20/11/1601
        \C:\Data\R1_Mobile_4_0.cfg      _       636     20/11/1601
        \C:\Data\xmlfile.xml    _       2600    20/11/1601
\C:\Data> select .xml
        \C:\Data\xmlfile.xml    _       2600    20/11/1601
\C:\Data> receive
File: xmlfile.xml queued for download.
*******It is transfering file xmlfile.xml, and current size = 0****
*******It is transfering file , and current size = 2600****
*******It has finished transfering file ,and total size = 2600 ****
File received: C:\receivedFiles\xmlfile.xml
File received: C:\receivedFiles\xmlfile.xml
\C:\Data> exit
Disconnected from FTP server
```

Future work

Here it could be easily added functions such as making directories, making files or "touching" but are not yet implemented.

Don't forget!

The Windows part of the library is depended upon the Bluesoleil drivers. The standard Microsoft Bluetooth Stack is not supported so you need to have the Bluesoleil drivers installed. The download page can be found here: http://www.bluesoleil.com/products/S0001201005190001.html So please don't forget to do that.

(C) Copyright Nokia 2013. All rights reserved.

Known issues

As you will see, the modification dates are not correct (it would be funny though if they were, since we would witness many Medieval files from the 1601 :P...interesting). Any way, this issue is the reason why the binary files in the project's page are not updated yet. It should better get fixed before updating the binaries. Until then make your self comfortable with the binaries contained in the repository.

favoritas37 14:17, 13 June 2012 (EEST)

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.