Discovering Bluetooth devices with the QBluetoothZero library

This article shows how to discover Bluetooth devices using using QBluetoothZero, a third-party library for using Bluetooth in Qt on Symbian. This article assumes that the library has been already compiled. If not, please see the project site for instructions on how to do this. This example uses Qt Creator.

Demo overview

This demo has an UI with two buttons: one for starting the discovery (start button) and another to stop it (disconnect button). The events are posted on a text box so that you can see what's going on. This UI is created with Qt Creator, and it looks like the picture below:

Widget class

The widget class declaration looks like this:

```
#include <QWidget>
#include <QBluetoothZero.h>

class Widget : public QWidget
{
    Q_OBJECT

public:
    explicit Widget(QWidget *parent = 0);
    ~Widget();

private:
    void put (const QString & s);

private slots:
    void startButtonPressed();
    void stopButtonPressed();

    void discoveryStopped ();
    void discoveryStarted ();
    void newDeviceFound (const QBtDevice & device);

private:
    Ui::Widget *ui;
    QBtDeviceDiscoverer deviceDiscoverer;

};
```

The `startButtonPressed` and `stopButtonPressed` are event handlers for button clicks, the first one if for the start button, and the other one is for the disconnect button. The remaining slots are handlers for events coming from `QBtDeviceDiscoverer`. The `put()` method is a convenience method to print text to the text box.

The global file `<QBluetoothZero.h>` includes all the necessary files for the QBluetoothZero library.

Signals for QBtDeviceDiscoverer

void discoveryStarted ()
This signal is emitted when the discovery starts.

void discoveryStopped ()
This signal is emitted when the discovery is finished.

void newDeviceFound (const QBtDevice & device)
This signal is emitted when a device is discovered. Notice that the service list for the device is not up to date when this signal is emitted. To get the service list for a specific device, it's necessary to use the `QBtServiceDiscoverer` class.

(C) Copyright Nokia 2013. All rights reserved.

Widget implementation

The convenience method put is implemented as this:

```
void Widget::put(const QString & s)
{
    ui->plainTextEdit->appendPlainText (s);
}
```

In the widget constructor, we wire the signals and slots:

```
Widget::Widget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Widget)
{
    ui->setupUi(this);

    // signals-slots for device discovery
    deviceDiscoverer = new QBtDeviceDiscoverer (this);
    connect (deviceDiscoverer, SIGNAL(discoveryStopped ()), SLOT (discoveryStopped ()) );
    connect (deviceDiscoverer, SIGNAL(discoveryStarted ()), SLOT (discoveryStarted ()) );
    connect (deviceDiscoverer, SIGNAL(newDeviceFound (const QBtDevice &)), SLOT (newDeviceFound (const QBtDevice &)) );


    // signals-slots for buttons
    connect (ui->startButton, SIGNAL(clicked()), SLOT(startButtonPressed()));
    connect (ui->stopButton, SIGNAL(clicked()), SLOT(stopButtonPressed()));

    // turn on bluetooth without bothering the user
    QBtLocalDevice::setBluetoothPowerState(true);
}
```

Starting and stopping discovery

Discovery starts when the user clicks the start button:

```
void Widget::startButtonPressed()
{
    // do not start if the discovery is still busy finding devices
    if (deviceDiscoverer.isBusy() )
      return;

    // clear the text box
    ui->plainTextEdit->clear();

    // let's go!
    deviceDiscoverer.startDiscovery();
}
```

Likewise, the user stops discovery by pressing the other button:

```
void Widget::stopButtonPressed()
{
  if (deviceDiscoverer.isBusy() )
    deviceDiscoverer.stopDiscovery();
}
```

Handling discovery events

Here's the implementation of the discovery event handling (which basically logs the action):

```
void Widget::discoveryStopped()
{
    put ("device discovery stopped");
}

void Widget::discoveryStarted()
{
    put ("device discovery started");
}

void Widget::newDeviceFound (const QBtDevice & device)
{
    put (QString("found device: %1 address: %2").arg(device.getName()).arg (device.getAddress().toString()  );
}
```

The .pro file

The .pro file looks like this:

```
QT       += core gui

TARGET = discoverer
TEMPLATE = app

# include this for Qt Creator to find the Symbian SDK and QBluetoothZero headers
INCLUDEPATH += $$EPOCROOT\epoc32\include
INCLUDEPATH += $$EPOCROOT\epoc32\include\QBluetoothZero

SOURCES += main.cpp\
        widget.cpp

HEADERS  += widget.h

FORMS    += widget.ui

# this says that we are going to use the QBluetoothZero library
LIBS += -lQBluetoothZero

symbian {
```


(C) Copyright Nokia 2013. All rights reserved.

```
    # select an UID from the protected range as the app cannot be self-signed
    TARGET.UID3 = 0x2003328F

    # the capabilities must match the ones used to build the QBluetoothZero library.
    # it required at least:  self-signed capabilities plus ReadDeviceData and WriteDeviceData.
    TARGET.CAPABILITY += LocalServices NetworkServices ReadUserData UserEnvironment WriteUserData ReadDeviceData WriteDeviceData



    # add deployment if you want to embed the library dll with the app installer. This way yo do not have to provide a separate .sis
    # for the library. However, if another application had already installed the dll with the same name, your installation will fail
    addFiles.sources = QBluetoothZero.dll
    addFiles.path = \sys\bin
    DEPLOYMENT += addFiles
}
```


(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.