Discovering nearby Bluetooth services with the QBluetoothZero library

This article shows how to discover Bluetooth services 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 was written and tested with Qt Creator.

Demo overview

05 Sep 2010

This demo has an UI with two buttons: one for starting the discovery (start button) and another to stop it (stop 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 discoveryStarted ();
    void discoveryEnded ();
    void newDeviceFound (const QBtDevice & device);

private:
    Ui::Widget *ui;
    QBtServiceDiscovererForAll discoverer;

};
```

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 stop button. The remaining slots are handlers for events coming from `QBtServiceDiscovererForAll`. There are more signals related to error handling, please see the library documentation for those. 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 QBtServiceDiscovererForAll

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 the service discovery is finished for the specified device, and there are services to report.

(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);

    // signal-slots for the service discoverer
    connect (&discoverer, SIGNAL(discoveryEnded ()), SLOT (discoveryEnded ()) );
    connect (&discoverer, SIGNAL(discoveryStarted ()), SLOT (discoveryStarted ()) );
    connect (&discoverer, SIGNAL(newDeviceFound (const QBtDevice &)), SLOT (newDeviceFound (const QBtDevice &)) );

    // signal-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::buttonPressed()
{
    // do not start if the discovery is still busy finding devices
    if (discoverer.isBusy() )
      return;

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

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

The `startDiscovery()`discovery process searches for OBEX and RFCOMM services. There are variations to customize the search, as this:

- `startObexDiscovery()` - Searches for OBEX services only.
- `startRfcommDiscovery()` - Searches for RFCOMM services only.
- `startDiscovery (const QBtUuid & uuid)` - This searches for a service with the specified UUID. This UUID can also be a protocol UUID, in this case the class searches for all services based on that protocol.

```
// my custom service uuid
QBtUuid uuid ("10FF");

discoverer.startDiscovery (uuid);
```

Please see the `QBtUuid` for alternate ways to specify UUIDs for it.

- `startDiscovery (const QList <QBtUuid> & uuidList)` - This searches for services based on an UUID list.

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

```
void Widget::disconnectPressed()
{
  if (discoverer.isBusy() )
    discoverer.stopDiscovery();
}
```

Handling discovery events

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

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

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

    // let's list all services
    put ("listing services");

    const QBtService::List & services = device.getSupportedServices();

    foreach (const QBtService & s, services)
    {
       put (s.getName());
    }

}

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

 (C) Copyright Nokia 2013. All rights reserved.

The .pro file

The .pro file looks like this:

```
QT       += core gui

TARGET = serviceDiscoverer
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 {

    # 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 you do not have to provide a separate .si
    # for the library. However, if another application had already installed the dll with the same name, your installation will fail
    addFiles.sources = QBluetoothZero
    addFiles.path = \sys\bin
    DEPLOYMENT += addFiles
}
```


(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.