Qt BluetoothLE UART library

Generated by Doxygen 1.9.4

Chapter 1

Qt BluetoothLE UART library

1.1 Introduction

This library can be used to talk to BLE devices via UART in Qt. It was designed to make talking to devices like the ESP32 from Qt easier.

1.2 Installation

```
cd yourRepo
git submodule add https://itsblue.dev/itsblue-development/QBluetoothLeUart.git
git submodule update --init --recursive
```

And in your MyProject.pro include the .pri file:

```
# Optional: enable QML stuff
CONFIG += QBluetoothLeUart_QML
# Include library
include($$PWD/QBluetoothLeUart/QBluetoothLeUart.pri)
```

To enable the QML module you need to call

```
QBluetoothLeUart::init();
```

somewhere before app.exec(); in your main.cpp.

1.3 Getting_started

This library currently supports: BluetoothLE UART client

1.3.1 UART client

To get started with the BLE client, see the docs of QBluetoothLeUartClient.

Chapter 2

Qt BluetoothLE UART library

A library for Qt to make using Bluetooth LE UART easier. Documentation is available here

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Class Documentation

6.1 QBluetoothLeUartClient Class Reference

The QBluetoothLeUartClient class can be used to talk to BluetoothLE devices via UART effordlessly. It can be used via C++ and QML.

`#include <qbluetoothleuartclient.h>`

Inheritance diagram for QBluetoothLeUartClient:

6.2 QBluetoothLeUartDevice Class Reference

The QBluetoothLeUartDevice class is a helper class for use with QBluetoothLeUart. It is used to get device details and connect to devices.

`#include <qbluetoothleuartdevice.h>`

Inheritance diagram for QBluetoothLeUartDevice:

Collaboration diagram for QBluetoothLeUartDevice:

Signals

- void **deviceChanged** ()

Public Member Functions

- **QBluetoothLeUartDevice** (QBluetoothDeviceInfo device, QObject *parent=nullptr)
- Q_INVOKABLE QString getName ()

  Function to get the name of the device.
- Q_INVOKABLE QString getAddress ()

  Function to get the address of the device.

Protected Member Functions

- QBluetoothDeviceInfo **getDevice** ()

Properties

- QString **name**
- QString **address**

Friends

- class **QBluetoothLeUartClient**

6.2.1 Detailed Description

The QBluetoothLeUartDevice class is a helper class for use with QBluetoothLeUart. It is used to get device details and connect to devices.

6.2.2 Member Function Documentation

6.2.2.1 getAddress()

```
QString QBluetoothLeUartDevice::getAddress ( )
```

Function to get the address of the device.

Returns

address of the device

6.2.2.2 getName()

```
QString QBluetoothLeUartDevice::getName ( )
```

Function to get the name of the device.

Returns

The name of the device

The documentation for this class was generated from the following files:

- /drone/src/qbluetoothleuartdevice.h
- /drone/src/qbluetoothleuartdevice.cpp

6.3 QBluetoothLeUartDeviceModel Class Reference

The QBluetoothLeUartDeviceModel class can be used to display available devices in a QML ListView.

`#include <qbluetoothleuartdevicemodel.h>`

Inheritance diagram for QBluetoothLeUartDeviceModel:

Collaboration diagram for QBluetoothLeUartDeviceModel:

Public Types

- enum **QBluetoothLeUartDeviceModelRole** { **NameRole** = Qt::DisplayRole , **IdRole** , **AddressRole** , **DeviceRole** }

Signals

- void **rowCountChanged** ()

Public Member Functions

- int **rowCount** (const QModelIndex &=QModelIndex()) const
- QVariant **data** (const QModelIndex &index, int role=Qt::DisplayRole) const
- QHash< int, QByteArray > **roleNames** () const

Protected Member Functions

- **QBluetoothLeUartDeviceModel** (QList< QBluetoothLeUartDevice * > availableDevices, QObject *parent=nullptr)
- void **append** (QBluetoothLeUartDevice *device)
- void **remove** (int row)
- void **clear** ()

Properties

- int **rowCount**

Friends

- class **QBluetoothLeUartClient**

6.3.1 Detailed Description

The QBluetoothLeUartDeviceModel class can be used to display available devices in a QML ListView.

Example implementation:

```
import de.itsblue.bluetoothleuart 1.0
QBluetoothLeUartClient {
    id: ble
    Component.onCompleted: {
        ble.startScanningForDevices()
    }
}
ListView {
    model: ble.availableDevicesModel
    delegate: ItemDelegate {
        width: parent.width
        text: name
        onClicked: backend.bleController.connectToDevice(device)
    }
}
```

The documentation for this class was generated from the following files:

- /drone/src/qbluetoothleuartdevicemodel.h
- /drone/src/qbluetoothleuartdevicemodel.cpp

Chapter 7

File Documentation

7.1 /drone/src/qbluetoothleuartclient.h

```
1 #ifndef BLUETOOTHLEUART_H
2 #define BLUETOOTHLEUART_H
3
4 #include <QBluetoothDeviceDiscoveryAgent>
5 #include <QBluetoothDeviceInfo>
6 #include <QLowEnergyController>
7 #include <QLowEnergyService>
8
9 #ifdef Q_OS_ANDROID
10 #include <QtAndroidExtras>
11 #endif
12
13 #ifdef QBluetoothLeUart_QML
14 #include <QQmlApplicationEngine>
15 #endif
16
17 #include <qbluetoothleuartdevice.h>
18 #include <qbluetoothleuartdevicemodel.h>
19
128 class QBluetoothLeUartClient : public QObject
129 {
130     Q_OBJECT
131     Q_PROPERTY(QVariantList availableDevices READ getAvailableDevicesDetailList NOTIFY
      avaliableDevicesChanged)
132     Q_PROPERTY(QBluetoothLeUartDeviceModel* availableDevicesModel READ getAvailableDevicesModel NOTIFY
      avaliableDevicesModelChanged)
133     Q_PROPERTY(QBluetoothLeUartDevice* currentDevice READ getCurrentDevice NOTIFY currentDeviceChanged)
134     Q_PROPERTY(BluetoothLeUartClientState state READ getState NOTIFY stateChanged)
135
136 public:
140     enum BluetoothLeUartClientState {
141         Idle = 0,
142         AdapterTurnedOff,
143         LocationPermissionDenied,
144         Scanning,
145         ScanFinished,
146         Connecting,
147         ScanningForService,
148         ServiceFound,
149         Connected
150     };
151     Q_ENUM(BluetoothLeUartClientState)
152
153     enum BluetoothScanError {
154         UnknownError,
155         AdapterTurnedOffError,
156         InputOutputError,
157         LocationPermissionDeniedError
158     };
159     Q_ENUM(BluetoothScanError);
160
161     QBluetoothLeUartClient(QObject *parent = nullptr);
162     ~QBluetoothLeUartClient();
163
167     static void init();
168
175     void setUUIDs(const char uartServiceUUID[36], const char txUUID[36], const char rxUUID[36]);
176
```

```
177 private:
178
179     // The UUIDs
180     QString uartServiceUUID;
181     QString txUUID;
182     QString rxUUID;
183
184     // current state
185     QBluetoothLeUartClient::BluetoothLeUartClientState state;
186
187     // QBluetooth controllers
188     QBluetoothDeviceDiscoveryAgent *bluetoothDeviceDiscoveryAgent;
189     QLowEnergyController *bluetoothController;
190     QLowEnergyService *bluetoothService;
191
192     // Bluetooth device
193     QBluetoothLeUartDevice *currentBluetoothDevice;
194     QList<QBluetoothLeUartDevice*> availableDevices;
195     QLowEnergyDescriptor bluetoothTransmissionDescriptor;
196     bool foundValidUARTService;
197
198     // for QML
199     QBluetoothLeUartDeviceModel* availableDevicesModel;
200
201 public slots:
202
203     Q_INVOKABLE bool requestLocationPermission();
204
205     Q_INVOKABLE bool isLocationPermissionGranted();
206
224     Q_INVOKABLE bool startScanningForDevices();
225
231     Q_INVOKABLE bool stopScanningForDevices();
232
242     Q_INVOKABLE QList<QBluetoothLeUartDevice*> getAvailableDevices();
243
257     Q_INVOKABLE QVariantList getAvailableDevicesDetailList();
258
266     Q_INVOKABLE QBluetoothLeUartDeviceModel* getAvailableDevicesModel();
267
272     Q_INVOKABLE QBluetoothLeUartDevice* getCurrentDevice();
273
290     Q_INVOKABLE bool connectToDevice(int deviceId);
291
308     Q_INVOKABLE bool connectToDevice(QBluetoothLeUartDevice *device);
309
315     Q_INVOKABLE bool disconnectFromDevice();
316
322     Q_INVOKABLE bool sendData(QString data, bool asynchronous = true);
323
329     Q_INVOKABLE QBluetoothLeUartClient::BluetoothLeUartClientState getState() const;
330
331 private slots:
332     void setState(QBluetoothLeUartClient::BluetoothLeUartClientState newState);
333
334     // Slots for QBluetothDeviceDiscoveryAgent
335     void handleDeviceDiscovered(const QBluetoothDeviceInfo&);
336     void handleScanFinished();
337     void handleDeviceScanError(QBluetoothDeviceDiscoveryAgent::Error);
338
339     // Slots for QLowEnergyController
340     void handleServiceDiscovered(const QBluetoothUuid & uuid);
341     void handleServiceScanDone();
342     void handleControllerError(QLowEnergyController::Error);
343     void handleDeviceConnected();
344     void handleDeviceDisconnected();
345
346     // Slots for QLowEnergyService
347     void handleServiceStateChange(QLowEnergyService::ServiceState s);
348     void handleServiceCharacteristicChange(const QLowEnergyCharacteristic &c, const QByteArray &value);
349     void handleServiceDescriptorWritten(const QLowEnergyDescriptor &d, const QByteArray &value);
350
351 signals:
352     void stateChanged(QBluetoothLeUartClient::BluetoothLeUartClientState newState);
353
354     void foundNewDevice(QBluetoothLeUartDevice* device);
355     void avaliableDevicesChanged(QList<QBluetoothLeUartDevice*> avaliableDevices);
356     void avaliableDevicesModelChanged();
357     void currentDeviceChanged();
358     void scanFinished(QList<QBluetoothLeUartDevice*> availableDevices);
359     void scanningErrorOccured(QBluetoothLeUartClient::BluetoothScanError error);
360
361     void connectedToDevice();
362
363     void dataReceived(QString data);
364
365 };
```

```
366
367 #endif // BLUETOOTHLEUART_H
```

7.2 /drone/src/qbluetoothleuartdevice.h

```
1 #ifndef DEVICEINFO_H
2 #define DEVICEINFO_H
3
4 #include <QString>
5 #include <QObject>
6 #include <qbluetoothdeviceinfo.h>
7 #include <qbluetoothaddress.h>
8 #include <qbluetoothuuid.h>
9
14 class QBluetoothLeUartDevice: public QObject
15 {
16     Q_OBJECT
17     Q_PROPERTY(QString name READ getName NOTIFY deviceChanged)
18     Q_PROPERTY(QString address READ getAddress NOTIFY deviceChanged)
19
20 public:
21     QBluetoothLeUartDevice(QBluetoothDeviceInfo device, QObject *parent = nullptr);
22
23     friend class QBluetoothLeUartClient;
24
29     Q_INVOKABLE QString getName();
30
35     Q_INVOKABLE QString getAddress();
36
37 protected:
38     QBluetoothDeviceInfo getDevice();
39
40 private:
41     void setDevice(QBluetoothDeviceInfo device);
42     QBluetoothDeviceInfo bluetoothDeviceInfo;
43
44 signals:
45     void deviceChanged();
46 };
47
48 #endif // DEVICEINFO_H
```

7.3 /drone/src/qbluetoothleuartdevicemodel.h

```
1 #ifndef QBLUETOOTHLEUARTDEVICEMODEL_H
2 #define QBLUETOOTHLEUARTDEVICEMODEL_H
3
4 #include <QAbstractListModel>
5 #include <QObject>
6 #include <qbluetoothleuartdevice.h>
7
35 class QBluetoothLeUartDeviceModel : public QAbstractListModel
36 {
37     Q_OBJECT
38     Q_PROPERTY(int rowCount READ rowCount NOTIFY rowCountChanged)
39 public:
40     friend class QBluetoothLeUartClient;
41
42     enum QBluetoothLeUartDeviceModelRole {
43         NameRole = Qt::DisplayRole,
44         IdRole,
45         AddressRole,
46         DeviceRole
47     };
48     Q_ENUM(QBluetoothLeUartDeviceModelRole)
49
50     int rowCount(const QModelIndex & = QModelIndex()) const;
51     QVariant data(const QModelIndex &index, int role = Qt::DisplayRole) const;
52     QHash<int, QByteArray> roleNames() const;
53
54 protected:
55     QBluetoothLeUartDeviceModel(QList<QBluetoothLeUartDevice*> availableDevices, QObject *parent =
      nullptr);
56
57     void append(QBluetoothLeUartDevice* device);
58     void remove(int row);
59     void clear();
60
61 private:
```

```
62     QList<QBluetoothLeUartDevice*> availableDevices;
63
64 signals:
65     void rowCountChanged();
66
67 };
68
69 #endif // QBLUETOOTHLEUARTDEVICEMODEL_H
```

Index