Bazinga CameraDriver

0.1

Wygenerowano przez Doxygen 1.6.2

Sun Feb 28 15:45:51 2010

Spis treści

Rozdział 1

Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 2

Dokumentacja klas

2.1 Dokumentacja klasy BConnection

Klasa reprezentujaca polaczenie.

`#include <bconnection.h>`

Sygnały

- void connected (quint32 sessid)

  Emitowany, gdy serwer wysle info o polaczeniu.

- void disconnected ()

  Emitowany, gdy serwer wysle informacje o zerwaniu polaczenia.

Metody publiczne

- BConnection (unsigned char clientType)

  Konstruuje obiekt z ustawionym typem klienta.

- bool isSessionAlive ()

  Sprawda czy sesja jest aktywna.

- void connect (const QString &serverAddress, quint16 serverPort, const QString &login, const QString &password, quint16 listeningPort, const QString &token=QString())

  Laczy z serwerem.

- void disconnectFromHost ()

  Rozlacz sie z serwerem.

- int sendData (BDatagram &data)

  Wysyla przygotowany wczesniej datagram.

- int sendData (unsigned char command, QByteArray &data)

Wysyla datagram z danymi.

- int sendData (unsigned char command)

  Wysyla pusty datagram.

- BDatagram * getData ()

  Odbierz datagram od serwera.

2.1.1 Opis szczegółowy

Klasa reprezentujaca polaczenie. Obiekt po polaczeniu wysyla do serwera okresowo pakiet informujacy, ze jeszcze zyje. Odbieranie informacji wykonuje sie zawsze jawnie odpalajac getData().

2.1.2 Dokumentacja konstruktora i destruktora

2.1.2.1 CLIENTLIBSHARED_EXPORT BConnection::BConnection (unsigned char *clientType*)

Konstruuje obiekt z ustawionym typem klienta.

Parametry:

clientType B_SOURCE_DRIVER albo B_SOURCE_APP

2.1.3 Dokumentacja funkcji składowych

2.1.3.1 void BConnection::connect (const QString & *serverAddress*, quint16 *serverPort*, const QString & *login*, const QString & *password*, quint16 *listeningPort*, const QString & *token* = `QString()`)

Laczy z serwerem. Binduje podany port (listeningPort), laczy sie z serwerem na podanym porcie, loguje na podstawie podanych danych (token moze byc pusty i nie podany)

Wysyla serwerowi komunikat SREQ o tresci login:haslo:port:token lub login:haslo:port w zaleznosci od tego, czy token zostal podany.

Ostrzeżenie:

Nie czeka na potwierdzenie serwera. Trzeba wywolac getData(), wtedy, jesli zostanie odebrane od serwera potwierdzenie zalogowania, obiekt polaczenia wyemituje sygnal connected(quint32 sessid).

Parametry:

serverAddress adres serwera

serverPort port serwera

login login

password haslo

listeningPort port na ktorym driver bedzie nasluchiwal potwierdzen z serwera

token token gry, parametr opcjonalny

2.1.3.2 void BConnection::connected (quint32 *sessid*) [signal]

Emitowany, gdy serwer wysle info o polaczeniu.

Parametry:

sessid id sesji

2.1.3.3 void BConnection::disconnectFromHost ()

Rozlacz sie z serwerem. Wysyla do serwera CLOSE i wylacza nasluchiwanie na porcie lokalnym.

2.1.3.4 BDatagram * BConnection::getData ()

Odbierz datagram od serwera. Metoda wstepnie przetwarza pobrany pakiet. W zaleznosci od jego zawartosci moze:

- zwrocic datagram.
- rzucic wyjatkiem BConnectionException z zawartoscia pakietu badz nie.
- zapisac sobie id sesji, wyemitowac sygnal connected(quint32 sessid) i zwrocic NULL. Od tego momentu zakladane jest, ze polaczenie zostalo nawiazane.

2.1.3.5 bool BConnection::isSessionAlive ()

Sprawda czy sesja jest aktywna. Technicznie sprowadza sie to do sprawdzenia czy sessid != 0.

Zwraca:

true jesli sesja jest aktywna, false w przeciwnym przypadku.

2.1.3.6 int BConnection::sendData (unsigned char *command*)

Wysyla pusty datagram. jak w sendData(unsigned char command, QByteArray & data) tylko ze wysyla sam naglowek.

Zwraca:

status jak w QUDPSocket::writeDatagram();

2.1.3.7 int BConnection::sendData (unsigned char *command*, QByteArray & *data*)

Wysyla datagram z danymi. Ustawia odpowiednio pole command i dane. Reszta jest pobierana z wlasciwosci tego obiektu.

Zwraca:

status jak w QUDPSocket::writeDatagram();

2.1.3.8 int BConnection::sendData (BDatagram & *data*)

Wysyla przygotowany wczesniej datagram.

Zwraca:

status jak w QUDPSocket::writeDatagram();

Dokumentacja dla tej klasy została wygenerowana z plików:

- bconnection.h
- bconnection.cpp

2.2 Dokumentacja klasy BConnectionBlocker

Blokuje albo zezwala na nawiazanie polaczenia w BConnectionWidget.

`#include <bconnectionblocker.h>`

Metody publiczne

- virtual bool canConnect ()

  Zezwala/zabrania nawiazanie polaczenia.

- virtual bool canDisconnect ()

  Zezwala/zabrania zerwanie polaczenia.

- virtual QString toString ()

  Zwraca komunikat ostatniego bledu.

2.2.1 Opis szczegółowy

Blokuje albo zezwala na nawiazanie polaczenia w BConnectionWidget. Klasa dziedziczaca moze zaimplementowac metody canConnect() i canDisconnect() zeby zablokowac laczenie/rozlaczenie w wybranych przypadkach.

Przydaje sie to, jesli chcemy, by widget nie pozwalal na polaczenie, jesli np. nie zostaly ustawione wszystkie potrzebne parametry gry.

2.2.2 Dokumentacja funkcji składowych

2.2.2.1 bool BConnectionBlocker::canConnect () [virtual]

Zezwala/zabrania nawiazanie polaczenia.

Zwraca:

domyslna implementacja zwraca true

2.2.2.2 bool BConnectionBlocker::canDisconnect () [virtual]

Zezwala/zabrania zerwanie polaczenia.

Zwraca:

domyslna implementacja zwraca true

Dokumentacja dla tej klasy została wygenerowana z plików:

- bconnectionblocker.h
- bconnectionblocker.cpp

2.3 Dokumentacja klasy BConnectionException

Klasa opisujaca wyjatek polazenia z serwerem Bazinga.

`#include <bconnectionexception.h>`

Metody publiczne

- BConnectionException (char *cause, BDatagram *datagram=NULL)

  Utworz wyjatek z podana przyczyna i datagramem.

- BConnectionException (const QString &cause, BDatagram *datagram=NULL)

  Utworz wyjatek z podana przyczyna i datagramem.

- BConnectionException (BConnectionException &old)

  Konstruktor kopiujacy. Przejmuje datagram po starym. W old datagram ustawiany jest na NULL.

- ~BConnectionException ()

  Destruktor, kasuje ew. datagram.

- const QString toString ()

  Zwraca opis wyjatku.

- BDatagram * getDatagram ()

  Zwraca datagram.

2.3.1 Opis szczegółowy

Klasa opisujaca wyjatek polazenia z serwerem Bazinga. Moze zawierac w sobie datagram powodujacy problem.

2.3.2 Dokumentacja funkcji składowych

2.3.2.1 BDatagram * BConnectionException::getDatagram ()

Zwraca datagram. Ewentualnego datagramu nie nalezy usuwac. Zrobi to destruktor wyjatku.

Zwraca:

Jesli nie ma datagramu, zwraca NULL.

2.3.2.2 const QString BConnectionException::toString ()

Zwraca opis wyjatku.

Zwraca:

powod + informacja czy jest w wyjatku zawarty datagram.

Dokumentacja dla tej klasy została wygenerowana z plików:

- bconnectionexception.h
- bconnectionexception.cpp

2.4 Dokumentacja klasy BConnectionWidget

Widget z wszystkimi polami potrzebnymi do logowania.

`#include <bconnectionwidget.h>`

Sloty publiczne

- void serverConnected (quint32 sessid)

  Informuj o nawiazaniu polaczenia.

- void serverDisconnected ()

  Informuj o rozlaczeniu.

- void pushConnectButton ()

  Programowo przydus przycisk polaczenia.

Sygnały

- void connectButtonClicked ()

  Emitowany zaraz po wcisniecu przycisku polaczenia.

- void throwException (BConnectionException *)

  Sygnal, ktory jest emitowany, jesli wewnatrz zostanie wykryty jakis wyjatek.

Metody publiczne

- BConnectionWidget (QWidget *parent=0, BConnection *connection=NULL, QSettings *settings=NULL, bool useToken=true, BConnectionBlocker *blocker=NULL)

  konstruktor

- void setSettingsObject (QSettings *settings)

  Ustaw obiekt do skladowania ustawien.

- void setBlockerObject (BConnectionBlocker *blocker)

  Ustaw obiekt blokujacy.

- void setConnectionObject (BConnection *connection)

  Ustaw obiekt polaczenia.

- void setUseToken (bool use)

  Uzywaj pola na token albo nie.

Metody chronione

- void changeEvent (QEvent ∗e)

  wygenerowane przez QtDesignera

2.4.1 Opis szczegółowy

Widget z wszystkimi polami potrzebnymi do logowania.

- Widget moze zapisac ustawienia w obiekcie typu QSettings, jesli podano.
- Moze sie odwolywac do obiektu powstrzymujacego logowanie typu BConnectionBlocker.

2.4.2 Dokumentacja konstruktora i destruktora

2.4.2.1 BConnectionWidget::BConnectionWidget (QWidget ∗ *parent* = `0`, BConnection ∗ *connection* = `NULL`, QSettings ∗ *settings* = `NULL`, bool *useToken* = `true`, BConnectionBlocker ∗ *blocker* = `NULL`)

konstruktor

Zobacz również:

setSettingsObject(QSettings ∗ settings), setBlockerObject(BConnectionBlocker ∗ blocker), setUseToken(bool use)

Parametry:

parent widget-rodzic

connection obiekt polaczenia, bardzo potrzebny IMHO ;)

settings Jesli podano, ustawienia zostana odczytane i zapisane z/do tego obiektu.

useToken Uzywaj pola token, wylaczenie spowoduje zniknięcie pola tokenu z formularza

blocker Ustaw obiekt blokujacy

2.4.3 Dokumentacja funkcji składowych

2.4.3.1 void BConnectionWidget::changeEvent (QEvent ∗ *e*) `[protected]`

wygenerowane przez QtDesignera Chyba chodzi o dynamiczna zmiane jezyka

2.4.3.2 void BConnectionWidget::connectButtonClicked () `[signal]`

Emitowany zaraz po wcisniecu przycisku polaczenia. Na wszelki przypadek, gdyby programista chcial wiedziec, kiedy uzytkownik wciska przycisk.

Emitowane tuz po wcisniecu polaczenia i jeszcze przed podjeciem akcji polaczenia/rozlaczenia/rezygnacji z proby nawiazania polaczenia.

2.4.3.3 void BConnectionWidget::serverConnected (quint32 *sessid*) [slot]

Informuj o nawiazaniu polaczenia. Widget sam dba o odpowienie podpiecie tu sygnalu z obiektu polaczenia.

2.4.3.4 void BConnectionWidget::serverDisconnected () [slot]

Informuj o rozlaczeniu.

Zobacz również:

serverConnected(quint32 sessid);

2.4.3.5 void BConnectionWidget::setBlockerObject (BConnectionBlocker * *blocker*)

Ustaw obiekt blokujacy. Obiekt blokujacy jest odpytywany przed proba polaczenia (BConnectionBlocker::canConnect()) i rozlaczenia (BConnectionBlocker::canDisconnect()) czy mozna nawiazywac polaczenie.

Moze to byc przydatne dla uzytkownika tej klasy, jesli chce okreslic swoje warunki nawiazania/przerwania polaczenia.

2.4.3.6 void BConnectionWidget::setConnectionObject (BConnection * *connection*)

Ustaw obiekt polaczenia. Bez tego nie ma za bardzo sensu uzycie tej klasy.

Widget sam sobie ustawia polaczenia do BConnection::connected(quint32 sessid) do serverConnected(quint32 sessid)) i analogicznie BConnection::disconnected() do serverDisconnected().

2.4.3.7 void BConnectionWidget::setSettingsObject (QSettings * *settings*)

Ustaw obiekt do skladowania ustawien. W podanym obiekcie zostana zapisane ustawienia logowania w odpowiednich kluczach:

- user/login
- user/save_password - true/false
- user/password - jesli zaznaczono zapisywanie hasla
- user/token - jesli useToken
- user/port - lokalny port do nasluchiwania
- server/address
- server/port

Pierwszy odczyt ustawien odbywa sie w wywolaniu tej funkcji

Zapisywanie odbywa sie przy kazdej probie nawiazania polaczenia.

2.4.3.8 void BConnectionWidget::setUseToken (bool *use*)

Uzywaj pola na token albo nie. Jesli use, pokaz pole na token. Spowoduje rowniez to, ze token nie bedzie zapisny w ustawieniach.

2.4.3.9 void BConnectionWidget::throwException (BConnectionException ∗) `[signal]`

Sygnal, ktory jest emitowany, jesli wewnatrz zostanie wykryty jakis wyjatek. Przydatne do wylapywania wyjatkow przez programiste uzywajacego tej klasy.

Ostrzeżenie:

Nie kasuj rzucanego obiektu na wlasna reke. Zostanie on skasowany po obsludze przez wszystkich obiorcow tego sygnalu.

Dokumentacja dla tej klasy została wygenerowana z plików:

- bconnectionwidget.h
- bconnectionwidget.cpp

2.5 Dokumentacja klasy BDatagram

Datagram zawierajacy generajuacy odpowiedenie naglowki.

`#include <bdatagram.h>`

Metody publiczne

- BDatagram (quint32 sessid, unsigned char source, quint32 timestamp, unsigned char type, QByteArray &data)

  Tworzy obiekt na podstawie podanych parametrow.

- BDatagram (char *dataIn, int size)

  Tworzy obiekt z surowej tablicy bajtow o podanym rozmiarze.

- QByteArray * getAllData ()

  Pobiera datagram gotowy do wyslania na serwer.

Atrybuty publiczne

- quint32 sessid

  ID sesji, dopoki nie jest znany mozna wstawic 0.

- unsigned char source

  Typ zrodla: B_SOURCE_APP, B_SOURCE_DRIVER lub B_SOURCE_SERVER.

- quint32 timestamp

  timestamp uniksowy.

- unsigned char type

  typ komunikatu.

- QByteArray data

  tablica danych

2.5.1 Opis szczegółowy

Datagram zawierajacy generajuacy odpowiedenie naglowki. Oferuje bardzo podstawowa funkcjonalnosc. Obiekt ma dwa konstruktory dla wygody i metode getAllData() do wyciagania tablic typu QByteArray gotowych do wyslania (z odpowiednim naglowkiem).

Ostrzeżenie:

Koduje liczby w kolejnosci bajtow little endian. Serwer jest na to przygotowany.

Naglowek sklada sie z:

- 4 bajty unsinged little endian - ID sesji.

- 1 bajt unsinged - typ zrodla.
- 4 bajty unsinged little endian - timestamp.
- 1 bajt unsigned - typ komunikatu, jedno z B_TYPE_∗
- 2 bajty unsinged little endian - rozmiar danych.

2.5.2 Dokumentacja konstruktora i destruktora

2.5.2.1 BDatagram::BDatagram (quint32 *sessid*, unsigned char *source*, quint32 *timestamp*, unsigned char *type*, QByteArray & *data*)

Tworzy obiekt na podstawie podanych parametrow.

Parametry:

sessid id sesji

source typ zrolda

timestamp timestamp uniksowy.

type typ komunikatu.

data tablica bajtow z danymi.

Dokumentacja dla tej klasy została wygenerowana z plików:

- bdatagram.h
- bdatagram.cpp

2.6 Dokumentacja klasy BOb

Klasa reprezenujaca polozenie obiektu (BOb(ject)) i jego wielkosc.

`#include <bob.h>`

Metody publiczne

- BOb (quint16 x=0, quint16 y=0, quint16 width=0, quint16 height=0)
  Tworzy obiekt z podanych liczb.
- BOb (QByteArray &arr, int offset)
  Tworzy obiekt z danych w tablicy.
- void appendToArray (QByteArray &arr) const
  Zapisuje obiekt na koniec podanej tablicy.

Atrybuty publiczne

- quint16 x
  Wspolrzedna X.
- quint16 y
  Wspolrzedna Y.
- quint16 width
  Szerokosc obiektu.
- quint16 height
  Wysokosc obiektu.

2.6.1 Opis szczegółowy

Klasa reprezenujaca polozenie obiektu (BOb(ject)) i jego wielkosc. Poniewaz dosc standardowa rozdzielczoscia kamer internetowych jest 640x480:

- x i width musza byc rzutowane na przedzial 0-639
- y i height musza byc rzutowane na przedzial 0-479

Technicznie kazda liczba reprezentowana jest jako 10 bitow unsigned.

2.6.2 Dokumentacja konstruktora i destruktora

2.6.2.1 BOb::BOb (QByteArray & *arr*, int *offset*)

Tworzy obiekt z danych w tablicy. Obiekt tworzac sie przeczyta kolejnych 5 bajtow tablicy. Traktujac je nastepujaco:

- Z pierwszych 4 bajtow czyta po 8 mlodszych bitow kazdej liczby (x,y...)
- Z ostatniego bajtu czyta po 2 starsze bity z kazdej liczby

Parametry:

arr tablica, z ktorej ma byc czytane

offset przesuniecie w tabeli od miejsca w ktorym ma czytac.

2.6.3 Dokumentacja funkcji składowych

2.6.3.1 void BOb::appendToArray (QByteArray & *arr*) const

Zapisuje obiekt na koniec podanej tablicy. Zapis jest prowadzony zgodnie z formatem opisanym w konstruktorze BOb(QByteArray & arr, int offset) .

Parametry:

arr Tablica do ktorej ma sie dodac.

2.6.4 Dokumentacja atrybutów składowych

2.6.4.1 quint16 BOb::height

Wysokosc obiektu. Wartosc musi byc z przedzialu 0-479.

2.6.4.2 quint16 BOb::width

Szerokosc obiektu. Wartosc musi byc z przedzialu 0-639.

2.6.4.3 quint16 BOb::x

Wspolrzedna X. Wartosc musi byc z przedzialu 0-639.

2.6.4.4 quint16 BOb::y

Wspolrzedna Y. Wartosc musi byc z przedzialu 0-479.

Dokumentacja dla tej klasy została wygenerowana z plików:

- bob.h
- bob.cpp

2.7 Dokumentacja klasy BObList

Lista obiektow BOb.

`#include <boblist.h>`

Metody publiczne

- BObList (QByteArray &arr)

  Czyta pakiety z tablicy bajtow.

- BObList ()

  Tworzy pusta tablice BOb-ow.

- QByteArray * pack ()

  Pakuje obiekty do tablicy bajtow i zwraca te tablice.

2.7.1 Opis szczegółowy

Lista obiektow BOb. Pakuje i wypakowuje obiekty BOb z tablicy bajtow. Dziedzy po QList, mozna wiec uzywac wszystkich metod/iteratorow dla QList.

2.7.2 Dokumentacja konstruktora i destruktora

2.7.2.1 BObList::BObList (QByteArray & *arr*)

Czyta pakiety z tablicy bajtow. Kazdy obiekt ma po 5 bajtow, wiec zakladana ilosc obiektow jest arr.size() / 5.

2.7.3 Dokumentacja funkcji składowych

2.7.3.1 QByteArray * BObList::pack ()

Pakuje obiekty do tablicy bajtow i zwraca te tablice. Tablica ma rozmiar ilosc obiektow * 5 bajtow.

Dokumentacja dla tej klasy została wygenerowana z plików:

- boblist.h
- boblist.cpp