МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

"ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

Кафедра "Обчислювальна техніка та програмування"

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТП

\_\_\_\_\_\_\_\_\_\_ проф.Семенов С. Г.

"\_\_\_"\_\_\_\_\_\_\_\_\_\_2018 р.

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОТРИМАННЯ ЗОБРАЖЕННЯ З IP-КАМЕРИ З ВИКОРИСТАННЯ ПРОТОКОЛУ ПЕРЕДАЧІ TCP

Текст програми

Лист затвердження

КІТ-26В.15389-01 12 01-1-ЛЗ

Розробники:

Керівник

\_\_\_\_\_\_\_\_\_\_ Шевердін І.В.

"\_\_\_"\_\_\_\_\_\_\_\_\_\_2018 р.

Виконавець

\_\_\_\_\_\_\_\_\_\_Гатанюк Н.С.

"\_\_\_"\_\_\_\_\_\_\_\_\_\_2018 р.

Харків 2018

ЗАТВЕРДЖЕНО

КІТ-26В.15389-01 12 01-1-ЛЗ

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОТРИМАННЯ ЗОБРАЖЕННЯ З IP-КАМЕРИ З ВИКОРИСТАННЯ ПРОТОКОЛУ ПЕРЕДАЧІ TCP

Текст програми

КІТ-26В.15389-01 12 01-1

Аркушів 9

Харків 2018

АНОТАЦІЯ

Даний документ входить до комплекту документів курсового проекту на тему "Розробка програмного забезпечення для отримання зображення з IP-камери з використанням протоколу передачі TCP".

Програма розроблена як проект для середовища Qt Creator .

У процесі розробки використовувалася мова програмування С++

Текст модулів та функції, що були згенеровані автоматично або незначно змінювались, цілком не приводиться.

ЗМІСТ

[1   Файл client.h 4](#_Toc406601793)

[2   Файл mainwindow.h 4](#_Toc406601794)

[3   Файл client.cpp 5](#_Toc406601795)

[4   Файл main.cpp 8](#_Toc406601796)

5 Файл mainwindow.cpp……………………………………………………………9

1.Файл client.h

#pragma once

#include <QWidget>

#include <QTcpSocket>

#include <QLabel>

class QTextEdit;

class QLineEdit;

/\*\*

\*@class MyClient Класс в котором храняться все переменные и данные необходимые для работы с камерой

\*

\*Данный класс состоит из многих переменных и методов необходимых для подключения, получения и вывода на экран стрима.

\*/

class MyClient : public QWidget

{

Q\_OBJECT

private:

QTcpSocket\* tcpSocket;///< Brief QTcpSocket Сокет для управления клиентом

QTextEdit\* txtInfo;///< Brief QTextEdit Виджет для вывода информации о состоянии процесса.

QLabel \*lb;///< Brief QLabel Виджет для вывода на экран стрима

QLineEdit\* txtIp;///< Brief QLineEdit Виджет для ввода адреса камеры

QLineEdit\* txtPort;///< Brief QLineEdit Виджет для ввода порта камеры

quint16 nextBlockSize;///< Brief quint16 Размер блока

QByteArray txt;

int Size = 0;///< Brief int Размер полученых данных

void getPicture(); ///<Функция получения изображения

public:

MyClient(const QString & strHost,int nPort,QWidget\* pwgt = 0); ///Конструктор

private slots:

void slotReadyRead();///Слот для чтения данных

void slotError(QAbstractSocket::SocketError);///слот соединен с сигналом отправляющем ощибку

void slotSendToServer();///Слот для отправки сообщения камере

void slotConnected(); ///Слот для подключения

//void slotReadChannelFinished();

};

2.Файл “mainwindow.h”

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

3.Файл “client.cpp”

#include <QtNetwork>

#include <QtGui>

#include <QWidget>

#include <QMessageBox>

#include <QTextEdit>

#include <QtWidgets>

#include <stdio.h>

#include "client.h"

/\*!

Копирует содержимое из исходной области памяти в целевую область память

\param strHost Адрес камеры в сети

\param nPort Порт камеры

\param pwgt Указатель на Класс Виджетов

\*/

MyClient::MyClient(const QString& strHost,int nPort, QWidget\* pwgt):QWidget(pwgt),nextBlockSize(0)

{

tcpSocket = new QTcpSocket(this); ///Создание сокета

//==============================================================================================

connect(tcpSocket,SIGNAL(connected()),SLOT(slotConnected()));///Соединяем сигнал и слот при создании соединения

connect(tcpSocket,SIGNAL(readyRead()),SLOT(slotReadyRead())); ///Соединяет слот и синал при готовности к передачи данных

//connect(tcpSocket,SIGNAL(readChannelFinished()),SLOT(slotReadChannelFinished()));

connect(tcpSocket,SIGNAL(error(QAbstractSocket::SocketError)),this,SLOT(slotError(QAbstractSocket::SocketError)));///Слот и сигнал при выдачи ощибки

//==============================================================================================

txtInfo = new QTextEdit; ///Виджет для отображения текста

txtIp = new QLineEdit;///Виджет для вставки и редактирования текста

txtPort = new QLineEdit;///Виджет для вставки и редактирования текста

txtInfo->setReadOnly(true);

QPushButton\* pcmd = new QPushButton("&GO!");///Виджет-кнопка для запуски стрима

//============================================================================

connect(pcmd,SIGNAL(clicked()),SLOT(slotSendToServer()));///Соединение слота и кнопки, передаеться сигнал о начале стрима и запуск метода

//============================================================================

QHBoxLayout\* res = new QHBoxLayout;

lb = new QLabel;///Рамка для Видео Изображения

lb->setFixedSize(1080,720);

//QImage img("C:\\Users\\Nata\\Desktop\\1.jpg");

//lb->setPixmap(QPixmap::fromImage(img));

QVBoxLayout\* pvbxLayout = new QVBoxLayout;

pvbxLayout->addWidget(new QLabel("<H1>Client</H1>"));

pvbxLayout->addWidget(txtIp);

pvbxLayout->addWidget(txtPort);

pvbxLayout->addWidget(txtInfo);

pvbxLayout->addWidget(pcmd);

res->addWidget(lb);

res->*addItem*(pvbxLayout);

setLayout(res);

txtIp->setText(strHost);

QString strPort = "";

strPort.setNum(nPort);

txtPort->setText(strPort);

}

void MyClient::slotReadyRead()

{

QByteArray arr = "";

if(Size == 0)

{

char search1[] = "Content-Length: ";

int pol1 = 0;

QByteArray size;

do

{

arr = tcpSocket->readLine();

pol1 = arr.indexOf(search1);

if(pol1 != -1)

{

break;

}

arr = "";

}

while(true);

pol1 =+ 16;

while(arr.at(pol1)!='\r')

{

size.append(arr.at(pol1));

pol1++;

}

Size = size.toInt();

txt = "";

arr = tcpSocket->readLine();

}

arr = "";

arr = tcpSocket->read(Size);

txt.append(arr);

int i = arr.size();

Size = Size - i;

if(Size == 0)

{

QImage img;

img.loadFromData(txt,"jpg");

lb->setPixmap(QPixmap::fromImage(img));

lb->setScaledContents(true);

}

}

void MyClient::slotError(QAbstractSocket::SocketError err)

{

QString strError = "Error: " + (err == QAbstractSocket::HostNotFoundError ? "The host was not found." : err == QAbstractSocket::RemoteHostClosedError ? "The remote host is closed." : err == QAbstractSocket::ConnectionRefusedError ? "The connection was refused. " : QString(tcpSocket->errorString()));

txtInfo->append(strError);

}

void MyClient::slotSendToServer()

{

QString ip = txtIp->text();

int port = (txtPort->text()).toInt();

tcpSocket->*connectToHost*(ip,port);

//const char request[] = "GET /axis-cgi/jpg/image.cgi?resolution=320x240 HTTP/1.1\r\nHost: 10.168.0.185\r\n\r\n";

QByteArray request;

request.append("GET /axis-cgi/mjpg/video.cgi?resolution=320x240 HTTP/1.1\r\nHost: " + ip + "\r\n\r\n");

//const char request[] = "GET /axis-cgi/mjpg/video.cgi?resolution=320x240 HTTP/1.1\r\nHost: 10.168.0.185\r\n\r\n";

//tcpSocket->write(request, strlen(request));

tcpSocket->write(request);

txtInfo->append("Send!");

//->setText("Send!");

}

void MyClient::slotConnected()

{

txtInfo->append("Received the connected() signals");

}

4.Файл “main.cpp”

#include "mainwindow.h"

#include <QApplication>

#include "client.h"

#include <QtWidgets>

/\*!

\author Gatanyuk N.S.

\version 1.0.1

\date December 2018 year

\*/

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MyClient client("169.254.69.224", 80);

QPalette pal;

pal.setBrush(client.backgroundRole(),QBrush(QPixmap("C:\\Users\\Nata\\Desktop\\camera2.jpg")));

QCursor cur = QCursor(QPixmap("C:\\Users\\Nata\\Desktop\\c3.png"));

client.setCursor(cur);

client.setPalette(pal);

client.setAutoFillBackground(true);

client.show();

return a.exec();

}

5.Файл “ mainwindow.cpp”

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~*MainWindow*()

{

delete ui;

}