МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение

Высшее профессиональное образование

Национальный исследовательский университет “МИЭТ”

Кафедра “Вычислительная техника”

**ПРОЕКТ**

Тема: “Программная часть программно-аппаратного комплекса тестирования блока аналого-цифрового преобразователя интегральных схем на основе базовых матричных кристаллов”

Проверяющий Тельминов О. А.

Выполнили студенты факультета

МПиТК группы МП-21

Петров А. С.

Клюшин С.М.

Москва 2015

**Содержание**

1. Постановка задачи, описание задачи………………………………………………..3
2. Платформа, среда разработки и язык программирования………………4
3. UML диаграммы………………………………………………………………………………….5
4. Описание программы…………………………………………………………………………8
5. Тестирование…………………………………………………………………………………….10
6. Заключение……………………………………………………………………………………….17
7. Приложение………………………………………………………………………………………18

**Постановка задачи**

Необходимо разработать ПО для проведения тестирования блока АЦП опытного образца микросхемы. Тестирование заключается в анализе зависимости результатов измерений от фактического значения входного напряжения. Наиболее наглядная форма представления подобных результатов – график.

**Описание задачи**

Для тестирования имеется аппаратная часть: тестовый модуль, выполняющий все необходимые аппаратные задачи. Модуль подключается к ПЭВМ по интерфейсу USB. Затем происходит последовательная установка напряжения, канала и блока АЦП на устройстве через COM-порт. После этого каждую секунду считываем данные с устройства. Получив преобразованное значение напряжения строим по ним различные графики зависимости. Существует три графика зависимости, по которым мы определяем правильность работы блоков АЦП. Также мы можем сохранить график в png формат для дальнейшего анализа. Работа осуществляется с устройством, реализованным по данной ниже схеме.

АЦП

SPI

Микроконтроллер

UARTRT

вам

RT230

USB

ЭВМ

dssd

ЦАП

АЦП

SPI

UART

Аналоговая схема

**Платформа, среда разработки и язык программирования**

Предоставленное приложение является кроссплатформенным (работает на всех платформах)

Среда разработки: Qt Creator 3.5.1

Была выбрана по причине наличия большого числа библиотек, удобного графического интерфейса, готовых шаблонов проектов и кроссплатформенности создаваемых приложений.

Язык С++ был выбран по следующим причинам:

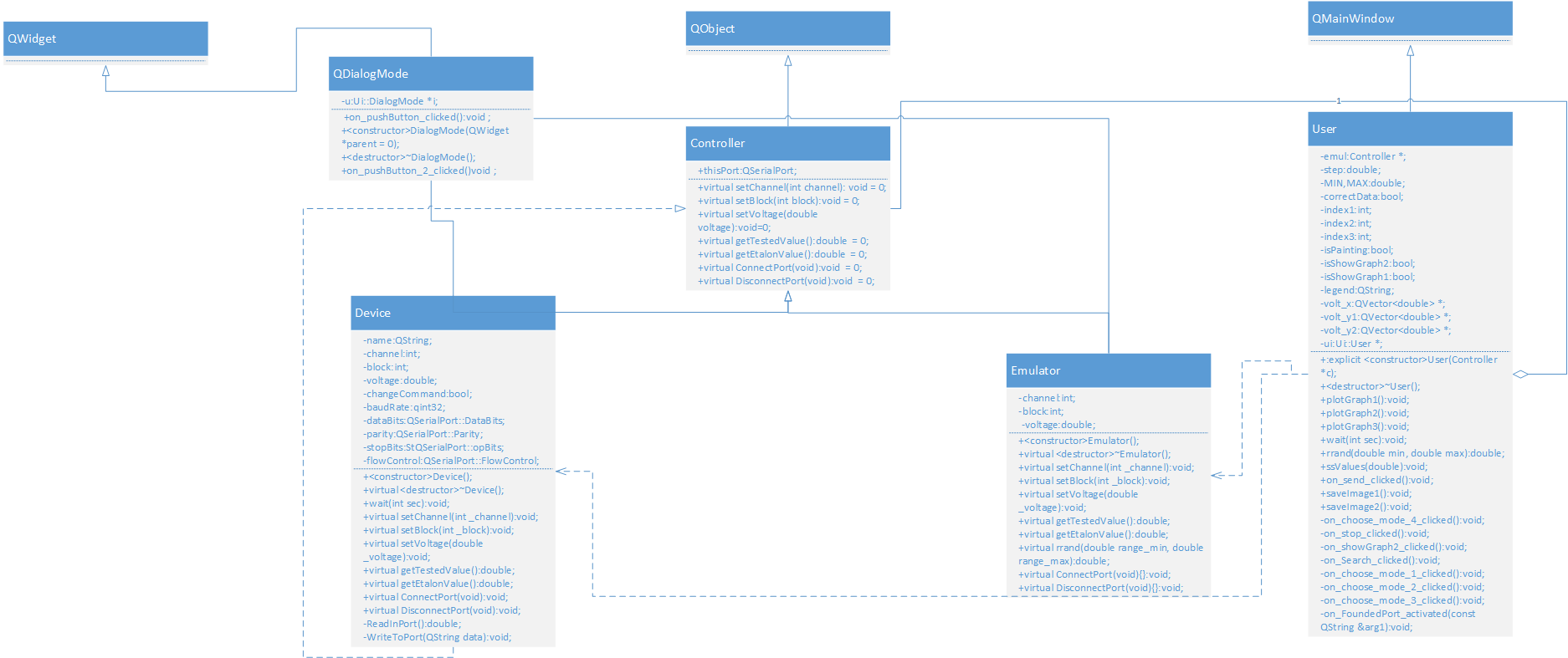
1. Компилируемый язык со статической типизацией
2. Сочетание высокоуровневых и низкоуровневых средств
3. Возможность реализации ООП
4. STL
5. Подробное изучение данного языка в рамках курса

Используемые библиотеки:

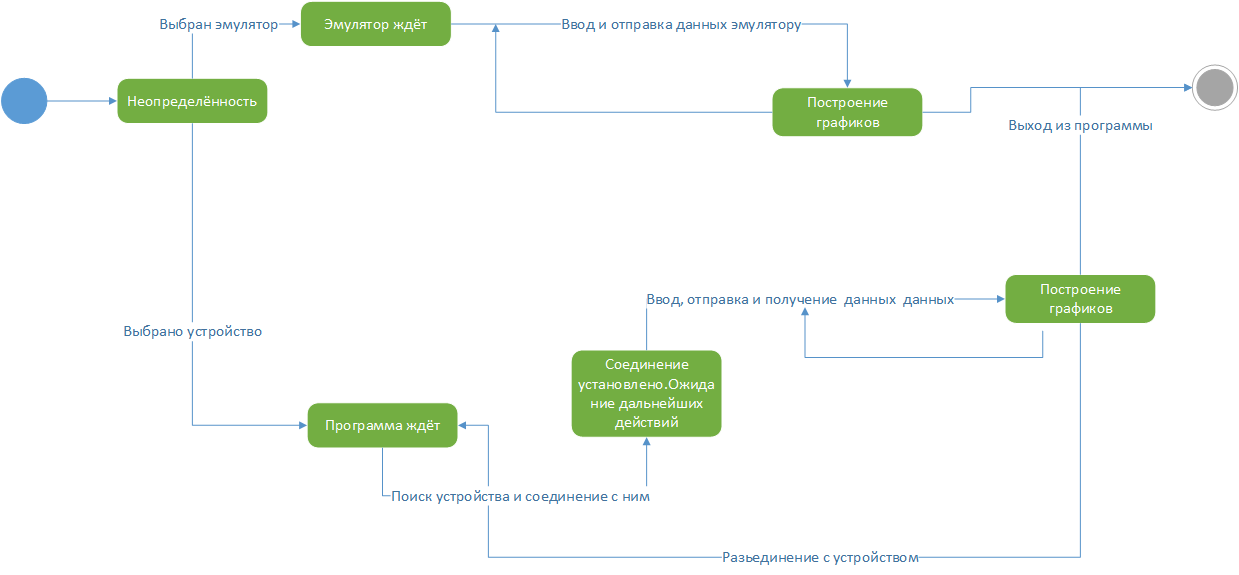
* QDebug
* QCustomPlot
* QTime
* QApplication
* QMainWindow
* QButtonGroup
* QVector
* QMouseEvent
* QString

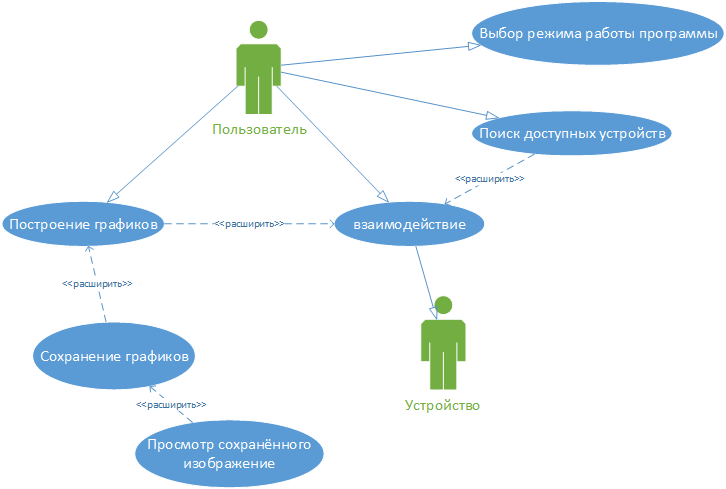
**UML – диаграммы**

**Диаграмма классов**



**Диаграмма состояний**



**Диаграмма Use-Case**

**Описание программы**

**Controller** – абстрактный класс, от которого наследуются классы Device и Emulator, содержащий чистые виртуальные функции.

**Device** – класс, реализующий методы абстрактного класса Controller. Содержит методы для обеспечения информационного обмена с устройством через COM-порт.

Стр. 3-10: описание конструктора, в котором инициализируются параметры

порта

Стр. 16-32: установка напряжения на устройстве, запоминание номера блока и канала в класс

Стр. 34-56: установка канала, блока АЦП и получение напряжения с измерителей

Стр. 58-75: установка параметров порта, соединение с портом

Стр. 77-82: разъединение с портом

Стр. 84-92: отправка команды в порт

Стр. 94-127: считывание данных с порта

Стр. 129 -136: установка таймера (используется для задержки между запросами)

**Emulator** – класс, реализующий методы абстрактного класса Controller.

Содержит методы для имитации работы с устройством без COM – порта.

Стр. 12-28: установка канала, блока АЦП и напряжения в эмуляторе устройства

Стр. 30-45: получение значений напряжений на тестируемом и эталонном измерителях

Стр. 47-49: установка рандомного значения в заданном диапазоне (используем для задания погрешности измерителей)

В emulator.h реализованы две пустые виртуальные функции для соединения и разъединения с портом

**User** – основной класс для коммуникации с устройством через класс Controller, для графического представления полученных с устройства данных и их последующего анализа оператором.

Стр. 6-64: в конструкторе инициализация полей класса, создание группы элементов формы, работа с графиками, регулярными выражениями, создание connect’ов

Стр. 66-70: в деструкторе высвобождение памяти

Стр. 71-194: установка канала и блока АЦП, проверка корректности введённых данных, вывод комментариев

Стр. 200-209: установка напряжения, получение напряжения с измерителей, построение графиков

Стр. 219-235: работа с элементами формы

Стр. 237-317: работа с графиками, остановка построения графиков, отображение и скрытие графика

Стр. 319-332: поиск доступных портов и вывод их в выпадающий список

Стр. 334-372: сохранение графиков по выбранному пути

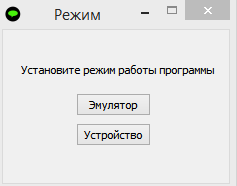
Стр. 373-403: установка соединения и разъединения с устройством и вывод соответствующих сообщений

**DialogMode** – класс для выбора режима работы программы

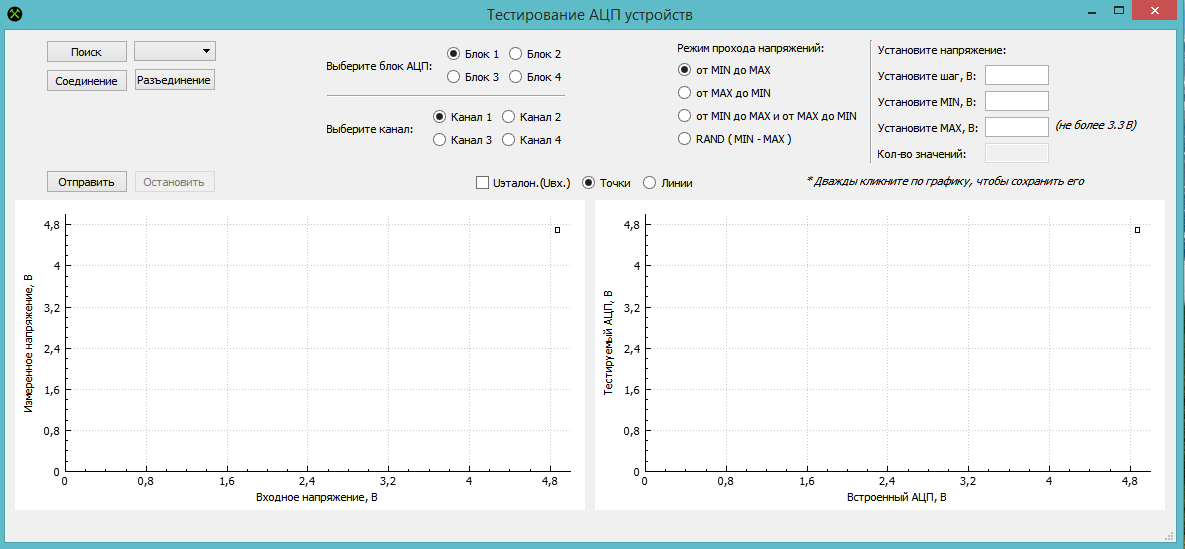
**Тестирование**

Технические требования программы:

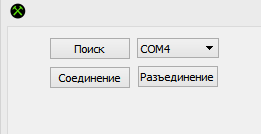
1. Выбор режима работы программы
2. Наличие графического интерфейса
3. Поиск, выбор доступных COM-портов на ПК, соединение и разъединение с ними
4. Установка канала и блока АЦП
5. Режим прохода напряжения в выбранном режиме по заданному диапазоне
6. Установка значений, не нарушающих логику
7. Отправка данных на устройство
8. Построение графиков по значениям, полученных с устройства
9. Интерактивная работа с графиками
10. Сохранение графиков по выбранному пользователем пути с уникальным именем, которое содержит информацию о канале, блоке, дату и время сохранения графиков
11. В этом окне мы можем выбрать режим работы устройства или режим работы эмулятора



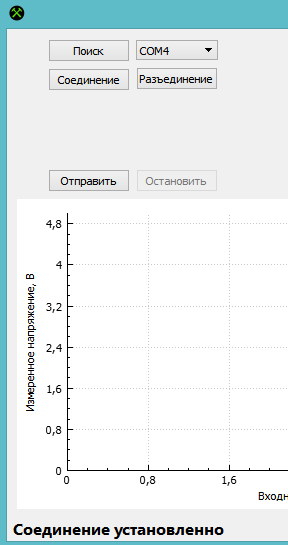
1. Основное окно программы



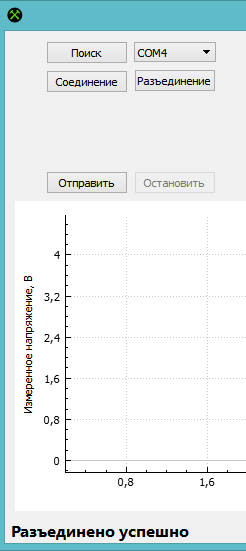
1. Поиск доступных COM-портов



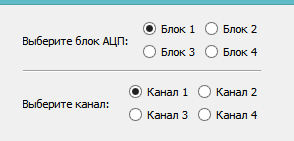
Соединение с устройством COM4



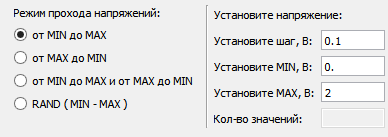
Разъединение с устройством COM4

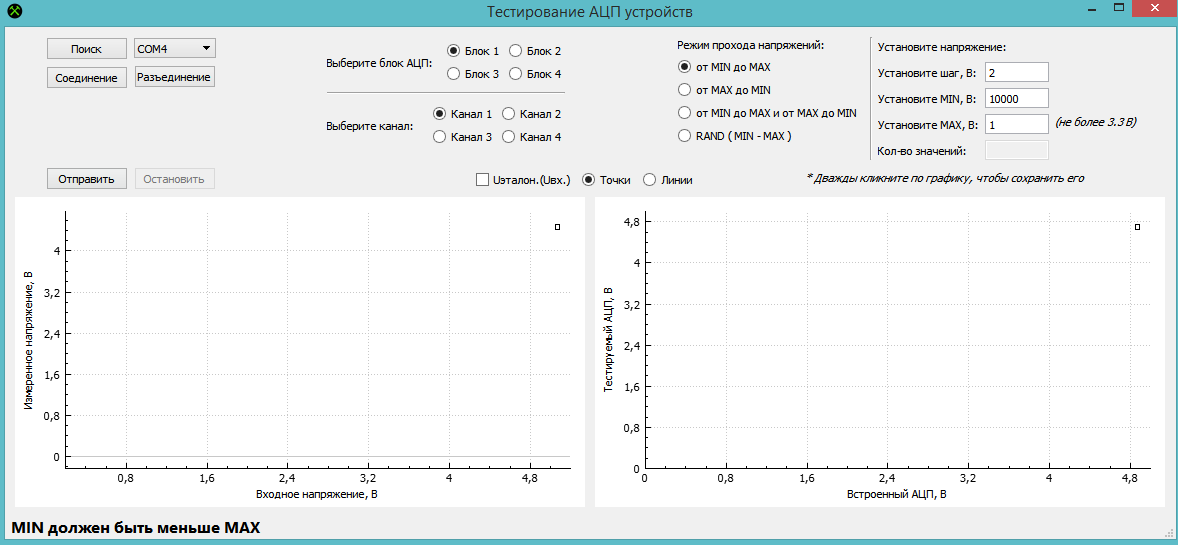


1. Выбор блока АЦП и канала

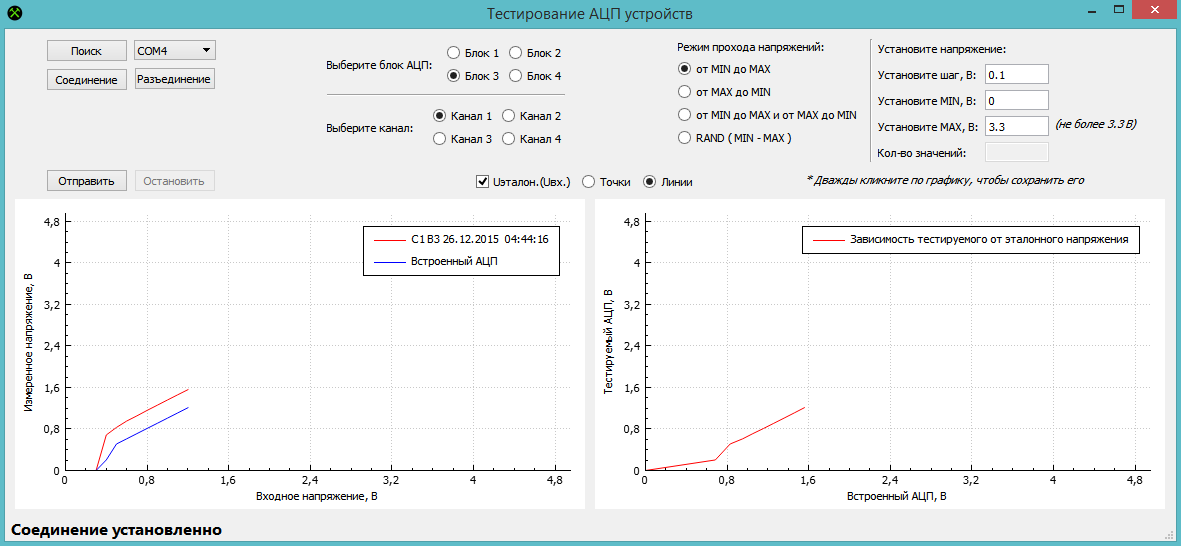


5. Выбор режима прохода напряжения по заданному диапазону с заданным шагом

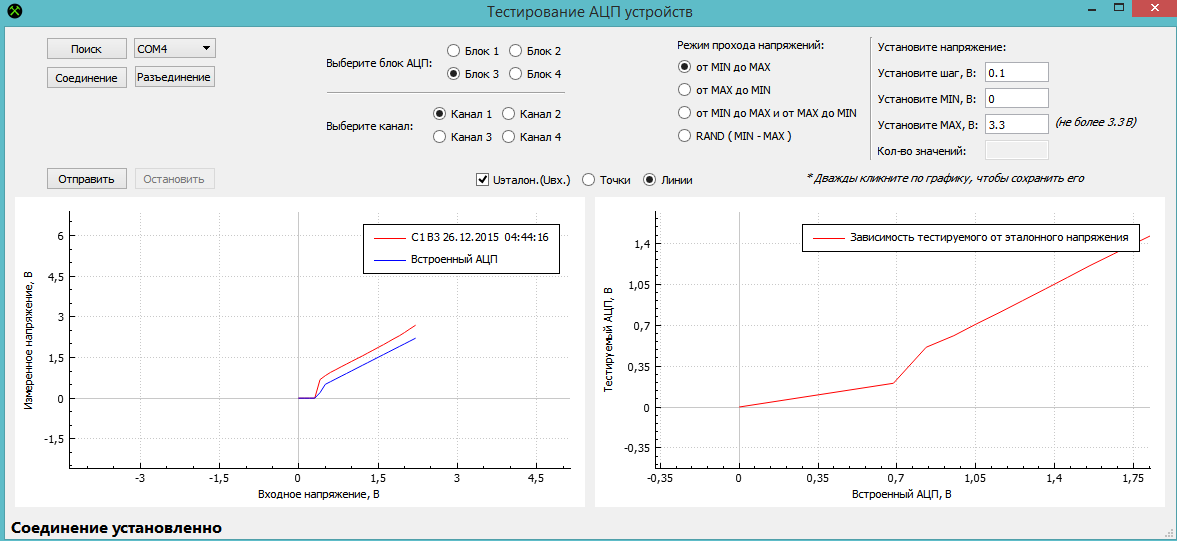
6. Проверка корректности введенных данных, с помощью регулярных выражений.



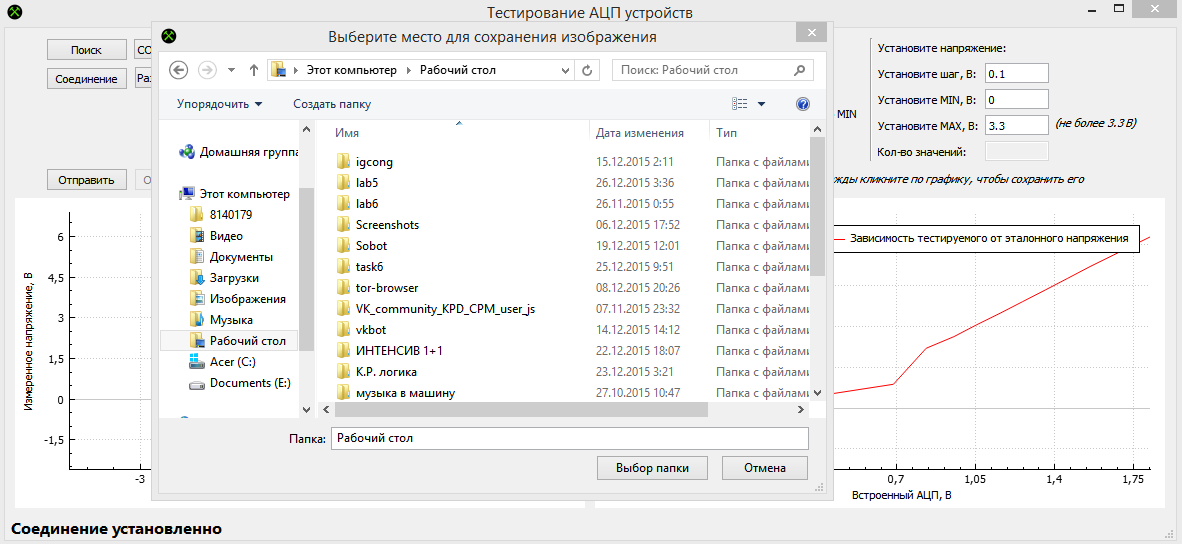
8. Построение графиков



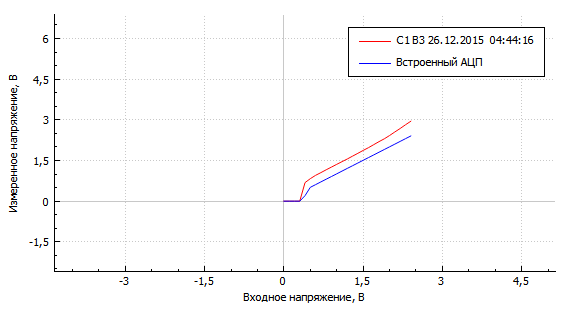
9. На левом графике мы увеличили масштаб, а на правом уменьшили.



10. Сохранение графиков в формате png.



Сохраненный график.



**Заключение**

Поставленная задача выполнена. Подзадачи, поставленные во время работы над проектом достигнуты и сделаны соответствующе выводы. Технические требования программы выполнены.  
  
В процессе работы над проектом, мы научились:  
1. Работать с реальными устройствами через COM- порт.  
2. Углубили свои знания в области микроэлектроники.  
3. Научились работать с библиотеками QCustomPlot и QSerialPort.  
4. Закрепили свои знания в области ООП на языке С++  
  
Дальнейшее развитие проекта видим в следующем:  
1. Добавить возможность выгрузки данных в файл, для дальнейшей работы.  
2. Добавить возможность настройки внутренних команд устройства пользователем.  
3. Добавить индикатор состояния соединения.  
4. Добавить программный анализ блоков АЦП, на основе полученных данных, в виде выявления наилучшего и наихудшего блоков АЦП.

**Приложение**

**controller.h**

#ifndef CONTROLLER\_H

#define CONTROLLER\_H

#include <QtSerialPort/QSerialPort>

#include <QtSerialPort/QSerialPortInfo>

#include "QDebug"

class Controller : public QObject

{

public:

QSerialPort thisPort;

virtual void *setChannel*(int channel) = 0;

virtual void *setBlock*(int block) = 0;

virtual void *setVoltage*(double voltage) = 0;

virtual double *getTestedValue*() = 0;

virtual double *getEtalonValue*() = 0;

public slots:

virtual void *ConnectPort*(void) = 0;

virtual void *DisconnectPort*(void) = 0;

};

#endif // CONTROLLER\_H

**device.h**

#ifndef DEVICE\_H

#define DEVICE\_H

#include "controller.h"

#include <QTime>

#include <QApplication>

#include <string.h>

class Device : public Controller

{

Q\_OBJECT

public:

Device();

virtual ~*Device*();

void wait(int sec);

virtual void *setChannel*(int \_channel);

virtual void *setBlock*(int \_block);

virtual void *setVoltage*(double \_voltage);

virtual double *getTestedValue*();

virtual double *getEtalonValue*();

private:

QString name;

int channel;

int block;

double voltage;

bool changeCommand;

qint32 baudRate;

QSerialPort::DataBits dataBits;

QSerialPort::Parity parity;

QSerialPort::StopBits stopBits;

QSerialPort::FlowControl flowControl;

public slots:

virtual void *ConnectPort*(void);

virtual void *DisconnectPort*(void);

private slots:

double ReadInPort();

void WriteToPort(QString data);

};

#endif // DEVICE\_H

**device.cpp**

#include "device.h"

Device::Device()

{

this->baudRate = 9600;

this->dataBits = QSerialPort::Data8;

this->parity = QSerialPort::NoParity;

this->stopBits = QSerialPort::OneStop;

this->flowControl = QSerialPort::NoFlowControl;

}

Device::~*Device*(){

}

void Device::*setChannel*(int \_channel){

channel = \_channel;

}

void Device::*setBlock*(int \_block){

block = \_block;

}

void Device::*setVoltage*(double \_voltage){

QString sendLine = "DAC ";

double DACValue\_ = (4095.\*\_voltage)/3.3;

unsigned int DACValue = (unsigned int)DACValue\_;

sendLine.append(QString::number(DACValue, 16));

WriteToPort(sendLine);

ReadInPort();

}

double Device::*getTestedValue*(){

QString sendLine = "CONVCH ";

sendLine.append(QString::number(channel-1));

WriteToPort(sendLine);

ReadInPort();

sendLine = "READ 0 ";

sendLine.append(QString::number(block-1));

WriteToPort(sendLine);

return ReadInPort();

}

double Device::*getEtalonValue*(){

QString sendLine = "ADC";

WriteToPort(sendLine);

changeCommand = true;

return ReadInPort();

}

void Device::*ConnectPort*(void){

if (thisPort.*open*(QIODevice::ReadWrite)){

if (

thisPort.setBaudRate(this->baudRate) &&

thisPort.setDataBits(this->dataBits) &&

thisPort.setParity(this->parity) &&

thisPort.setStopBits(this->stopBits) &&

thisPort.setFlowControl(this->flowControl)){

thisPort.setTextModeEnabled(true);

if (thisPort.isOpen()){

qDebug()<<"Открыт";

}

}

}

}

void Device::*DisconnectPort*(){

if(thisPort.isOpen()){

thisPort.*close*();

qDebug()<< "Отключились от порта";

}

}

void Device::WriteToPort(QString data){

if(thisPort.isOpen()){

thisPort.clear();

QString line = data;

line.append("\r");

QByteArray data = line.toLocal8Bit();

thisPort.write(data);

}

}

double Device::ReadInPort(){

wait(1);

if (thisPort.isOpen()){

QString readed;

int MaxLines=20;

double val = 4.;

while (--MaxLines)

{

readed = thisPort.readLine();

readed = readed.trimmed();

// qDebug() << "[" << readed << "]";

if (readed == "Done.")

break;

QStringList list = readed.split(':');

for (int i=0; i< list.count(); i++)

list[i] = list[i].trimmed();

if (list.count()>=2)

{

bool ok;

if (list[0]=="MADCVAL")

{

val = list[1].toInt(&ok, 16);

val = val/4095\*3.3;

}

if (list[0]=="ADCVOLT")

{

val = list[1].toDouble(&ok);

}

}

}

return val;

}

else return 0;

}

void Device::wait(int sec){

QTime time;

time.start();

for(;time.elapsed() < sec\*1000;){

qApp->processEvents();

}

}

**dialogmode.h**

#ifndef DIALOGMODE\_H

#define DIALOGMODE\_H

#include "emulator.h"

#include "device.h"

#include <QWidget>

#include <QDir>

namespace Ui {

class DialogMode;

}

class DialogMode : public QWidget

{

Q\_OBJECT

public:

DialogMode(QWidget \*parent = 0);

~*DialogMode*();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

private:

Ui::DialogMode \*ui;

};

#endif // DIALOGMODE\_H

**dialogmode.cpp**

#include "dialogmode.h"

#include "ui\_dialogmode.h"

#include "user.h"

DialogMode::DialogMode(QWidget \*parent) :

QWidget(parent),

ui(new Ui::DialogMode)

{

ui->setupUi(this);

this->window()->setWindowTitle("Режим");

this->setWindowIcon(QIcon(QDir::currentPath()+"/icon1.png"));

}

DialogMode::~*DialogMode*()

{

delete ui;

}

void DialogMode::on\_pushButton\_clicked()

{

this->close();

Controller \*emulator = new Emulator();

User \*user = new User(emulator);

user->show();

}

void DialogMode::on\_pushButton\_2\_clicked()

{

this->close();

Controller \*emulator = new Device();

User \*user = new User(emulator);

user->show();

}

**emulator.h**

#ifndef EMULATOR\_H

#define EMULATOR\_H

#include "controller.h"

#include <iostream>

class Emulator : public Controller

{

Q\_OBJECT

public:

Emulator();

virtual ~*Emulator*();

virtual void *setChannel*(int \_channel);

virtual void *setBlock*(int \_block);

virtual void *setVoltage*(double \_voltage);

virtual double *getTestedValue*(); //add random -10%-+10%

virtual double *getEtalonValue*(); //add random -2%-+2%

virtual double *rrand*(double range\_min, double range\_max);

private:

int channel;

int block;

double voltage;

public slots:

virtual void *ConnectPort*(void){}

virtual void *DisconnectPort*(void){}

};

#endif // EMULATOR\_H

**emulator.cpp**

#include "emulator.h"

#include <qmath.h>

using namespace std;

Emulator::Emulator(){

}

Emulator::~*Emulator*(){

}

void Emulator::*setChannel*(int \_channel){

qDebug()<< "Установлен канал №";

channel = \_channel;

qDebug()<<channel;

}

void Emulator::*setVoltage*(double \_voltage){

qDebug()<< "Установлено напряжение:";

voltage = \_voltage;

qDebug()<<voltage;

}

void Emulator::*setBlock*(int \_block){

qDebug()<< "Установлен блок №";

block = \_block;

qDebug()<<block;

}

double Emulator::*getTestedValue*(){

double rand = *rrand*(-voltage\*0.1,voltage\*0.1);

qDebug()<<"На тестируемом измерителе:";

double testVoltage = voltage+rand;

qDebug() << testVoltage;

return testVoltage;

}

double Emulator::*getEtalonValue*(){

double rand = *rrand*(-voltage\*0.02,voltage\*0.02);

qDebug()<<" На эталонном измерителе:";

double etalonVoltage = voltage+rand;

qDebug() << etalonVoltage;

return etalonVoltage;

}

double Emulator::*rrand*(double range\_min, double range\_max) {

return double((double)rand() / (RAND\_MAX + 1) \* (range\_max - range\_min) + range\_min);

}

**user.h**

#ifndef USER\_H

#define USER\_H

#include "controller.h"

#include <QMainWindow>

#include <QButtonGroup>

#include <QVector>

#include <QMouseEvent>

namespace Ui {

class User;

}

class User : public QMainWindow

{

Q\_OBJECT

public:

explicit User(Controller \*c);

~*User*();

void plotGraph1();

void plotGraph2();

void plotGraph3();

void wait(int sec);

double rrand(double min, double max);

void ssValues(double);

private:

Controller \*emul;

double step;

double MIN,MAX;

bool correctData;

int index1,index2,index3;

bool isPainting;

bool isShowGraph2;

bool isShowGraph1;

QVector<double> \*volt\_x,\*volt\_y1,\*volt\_y2;

QString legend;

public slots:

void on\_send\_clicked();

void saveImage1();

void saveImage2();

private slots:

void on\_choose\_mode\_4\_clicked();

void on\_stop\_clicked();

void on\_showGraph2\_clicked();

void on\_Search\_clicked();

void on\_choose\_mode\_1\_clicked();

void on\_choose\_mode\_2\_clicked();

void on\_choose\_mode\_3\_clicked();

void on\_FoundedPort\_activated(const QString &arg1);

// void

void on\_Connect\_clicked();

void on\_Disconnect\_clicked();

private:

Ui::User \*ui;

};

#endif // USER\_H

**user.cpp**

#include "user.h"

#include "ui\_user.h"

#include <QString>

#include <iostream>

User::User(Controller \*c): QMainWindow(), ui(new Ui::User)

{

emul = c;

isShowGraph2 = false;

isShowGraph1 = false;

index1 = 0;

index2 = 0;

index3 = 0;

legend = "";

ui->setupUi(this);

this->setWindowTitle("Тестирование АЦП устройств");

this->setWindowIcon(QIcon(QDir::currentPath()+"/icon.png"));

QButtonGroup \*group = new QButtonGroup();

group->addButton(ui->channel\_1);

group->addButton(ui->channel\_2);

group->addButton(ui->channel\_3);

group->addButton(ui->channel\_4);

group = new QButtonGroup();

group->addButton(ui->block\_1);

group->addButton(ui->block\_2);

group->addButton(ui->block\_3);

group->addButton(ui->block\_4);

group = new QButtonGroup();

group->addButton(ui->dotOnGraph);

group->addButton(ui->lineOnGraph);

ui->widget->xAxis->setLabel("Входное напряжение, В");

ui->widget->yAxis->setLabel("Измеренное напряжение, В");

ui->widget->legend->setVisible(true);

ui->widget->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom | QCP::iSelectPlottables);

ui->widget\_2->xAxis->setLabel("Встроенный АЦП, В");

ui->widget\_2->yAxis->setLabel("Тестируемый АЦП, В");

ui->widget\_2->legend->setVisible(true);

ui->widget\_2->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom | QCP::iSelectPlottables);

ui->dotOnGraph->setChecked(true);

ui->block\_1->setChecked(true);

ui->channel\_1->setChecked(true);

ui->choose\_mode\_1->setChecked(true);

ui->num->setDisabled(true);

ui->stop->setDisabled(true);

QRegExp rx("^[0-9]\*[.]{1}[0-9]\*$");

QValidator \*validator = new QRegExpValidator(rx, this);

ui->step->setValidator( validator );

ui->min->setValidator(validator);

ui->max->setValidator(validator);

connect(ui->widget, SIGNAL(mouseDoubleClick(QMouseEvent\*)), this, SLOT(saveImage1()));

connect(ui->widget\_2, SIGNAL(mouseDoubleClick(QMouseEvent\*)), this, SLOT(saveImage2()));

}

User::~*User*()

{

delete ui;

}

void User::on\_send\_clicked()

{

legend = "";

ui->statusbar->clearMessage();

if (!emul->thisPort.isOpen()){

ui->statusbar->showMessage("Нет соединения. Установите соединение");

}

if (ui->showGraph2->isChecked()) isShowGraph2 = true;

else isShowGraph2 = false;

volt\_x = new QVector<double>;

volt\_y1 = new QVector<double>;

volt\_y2 = new QVector<double>;

ui->widget->clearGraphs();

ui->widget\_2->clearGraphs();

index1 = 0;

index2 = 0;

index3 = 0;

isPainting = true;

if (ui->channel\_1->isChecked()){

emul->*setChannel*(1);

legend.append("C1 ");

}

else if (ui->channel\_2->isChecked()){

emul->*setChannel*(2);

legend.append("C2 ");

}

else if (ui->channel\_3->isChecked()){

emul->*setChannel*(3);

legend.append("C3 ");

}

else if (ui->channel\_4->isChecked()){

emul->*setChannel*(4);

legend.append("C4 ");

}

if (ui->block\_1->isChecked()){

emul->*setBlock*(1);

legend.append("B1 ");

}

else if (ui->block\_2->isChecked()){

emul->*setBlock*(2);

legend.append("B2 ");

}

else if (ui->block\_3->isChecked()){

emul->*setBlock*(3);

legend.append("B3 ");

}

else if (ui->block\_4->isChecked()){

emul->*setBlock*(4);

legend.append("B4 ");

}

if (ui->step->text().length() > 0 && ui->min->text().length() > 0 && ui->max->text().length() > 0){

if (ui->min->text().toDouble() < ui->max->text().toDouble()){

if (ui->step->text().toDouble() <= (ui->max->text().toDouble() - ui->min->text().toDouble())){

step = ui->step->text().toDouble();

MIN = ui->min->text().toDouble();

MAX = ui->max->text().toDouble();

}

else{

ui->statusbar->showMessage("Шаг неверный. Введите корректный шаг");

return;

}

}

else{

ui->statusbar->showMessage("MIN должен быть меньше MAX");

return;

}

}

else {

ui->statusbar->showMessage("Введите данные");

return;

}

ui->widget->xAxis->setRange(0, MAX\*1.5);

ui->widget->yAxis->setRange(0, MAX\*1.5);

ui->widget\_2->xAxis->setRange(0, MAX\*1.5);

ui->widget\_2->yAxis->setRange(0, MAX\*1.5);

if (emul->thisPort.isOpen()){

ui->stop->setEnabled(true);

}

if(ui->choose\_mode\_1->isChecked() ){

for (double i = MIN; i <= MAX; i+=step){

if (!isPainting) break;

ssValues(i);

}

}

else if(ui->choose\_mode\_2->isChecked()){

for (double i = MAX; i >=MIN; i-=step){

if (!isPainting) break;

ssValues(i);

}

}

else if(ui->choose\_mode\_3->isChecked()){

for (double i = MIN; i <= MAX; i+=step){

if (!isPainting) break;

ssValues(i);

}

for (double j = MAX; j >=MIN; j-=step){

if (!isPainting) break;

ssValues(j);

}

}

else if(ui->choose\_mode\_4->isChecked()){

int kol = ui->num->text().toInt();

for (double i = 0; i < kol;i++){

if (!isPainting) break;

ssValues(rrand(MIN,MAX));

}

}

if (emul->thisPort.isOpen()){

ui->statusbar->showMessage("Построение графиков завершено");

ui->stop->setDisabled(true);

}

}

double User::rrand(double range\_min, double range\_max) {

return double((double)rand() / (RAND\_MAX + 1) \* (range\_max - range\_min) + range\_min);

}

void User::ssValues(double value){

emul->*setVoltage*(value); // 1 DAC

volt\_x->append(value);

volt\_y1->append(emul->*getTestedValue*()); // CONVCH Затем READ

volt\_y2->append(emul->*getEtalonValue*()); // ADC

plotGraph1();

plotGraph3();

if (ui->showGraph2->isChecked()) plotGraph2();

}

void User::wait(int sec){

QTime time;

time.start();

for(;time.elapsed() < sec\*1000;){

qApp->processEvents();

}

}

void User::on\_choose\_mode\_1\_clicked(){

ui->step->setEnabled(true);

ui->num->setDisabled(true);

}

void User::on\_choose\_mode\_2\_clicked(){

User::on\_choose\_mode\_1\_clicked();

}

void User::on\_choose\_mode\_3\_clicked(){

User::on\_choose\_mode\_1\_clicked();

}

void User::on\_choose\_mode\_4\_clicked(){

ui->step->setDisabled(true);

ui->num->setEnabled(true);

}

void User::plotGraph1(){

if (index1 == 0){

ui->widget->addGraph();

ui->widget->graph(0)->setPen(QColor(Qt::red)); //задаем цвет точки

if (!ui->lineOnGraph->isChecked()){

ui->widget->graph(0)->setLineStyle(QCPGraph::lsNone); //убираем линии

ui->widget->graph(0)->setScatterStyle(QCPScatterStyle(QCPScatterStyle::ssCircle, 4)); //формируем вид точек

}

ui->widget->graph(0)->setName(legend+QDateTime::currentDateTime().toString("dd.MM.yyyy HH:mm:ss"));

}

ui->widget->graph(0)->setData(\*volt\_x,\*volt\_y1);

ui->widget->replot();

index1++;

isShowGraph1 = true;

}

void User::plotGraph2(){

if (index2 == 0){

ui->widget->addGraph();

ui->widget->graph(1)->setPen(QColor(Qt::blue)); //задаем цвет точки

if (!ui->lineOnGraph->isChecked()){

ui->widget->graph(1)->setLineStyle(QCPGraph::lsNone); //убираем линии

ui->widget->graph(1)->setScatterStyle(QCPScatterStyle(QCPScatterStyle::ssCircle, 4)); //формируем вид точек

}

ui->widget->graph(1)->setName("Встроенный АЦП");

index2++;

}

ui->widget->graph(1)->setData(\*volt\_x,\*volt\_y2);

ui->widget->replot();

isShowGraph2 = true;

}

void User::plotGraph3(){

if (index3 == 0){

ui->widget\_2->addGraph();

ui->widget\_2->graph(0)->setPen(QColor(Qt::red)); //задаем цвет точки

if (!ui->lineOnGraph->isChecked()){

ui->widget\_2->graph(0)->setLineStyle(QCPGraph::lsNone); //убираем линии

ui->widget\_2->graph(0)->setScatterStyle(QCPScatterStyle(QCPScatterStyle::ssCircle, 4)); //формируем вид точек

}

ui->widget\_2->graph(0)->setName("Зависимость тестируемого от эталонного напряжения");

index3++;

}

ui->widget\_2->graph(0)->setData(\*volt\_y1,\*volt\_y2);

ui->widget\_2->replot();

}

void User::on\_stop\_clicked()

{

isPainting = false;

ui->stop->setDisabled(true);

ui->statusbar->showMessage("Построение графиков остановлено");

}

void User::on\_showGraph2\_clicked()

{

if (!isShowGraph1) return;

if (ui->showGraph2->isChecked() && !isShowGraph2 && isShowGraph1){

plotGraph2();

ui->widget->replot();

}

else{

if (ui->widget->graph(1)->visible() == false){

ui->widget->graph(1)->setVisible(true);

ui->widget->graph(1)->*addToLegend*();

ui->widget->replot();

return;

}

if (ui->widget->graph(1)->visible() == true){

ui->widget->graph(1)->setVisible(false);

ui->widget->graph(1)->*removeFromLegend*();

ui->widget->replot();

}

}

}

void User::on\_Search\_clicked()

{

ui->FoundedPort->clear();

foreach (const QSerialPortInfo &info, QSerialPortInfo::availablePorts()){

ui->FoundedPort->addItem(info.portName());

}

emul->thisPort.setPortName(ui->FoundedPort->itemText(ui->FoundedPort->currentIndex()));

}

void User::on\_FoundedPort\_activated(const QString &arg1)

{

emul->thisPort.setPortName(arg1);

}

void User::saveImage1(){

QString directory = QFileDialog::getExistingDirectory(this,

tr("Выберите место для сохранения изображения"), QDir::currentPath());

if (!directory.isEmpty()) {

QString name = legend+QDateTime::currentDateTime().toString("dd\_MM\_yyyy HH\_mm\_ss");

if (!legend.isEmpty()) directory.append("/"+name+".png");

else {

ui->statusbar->showMessage("Постройте график для сохранения");

return;

}

if (ui->widget->savePng(directory)) {

ui->statusbar->showMessage("График успешно сохранен");

}

else ui->statusbar->showMessage("График не сохранен. Повторите попытку");

}

}

void User::saveImage2(){

QString directory = QFileDialog::getExistingDirectory(this,

tr("Выберите место для сохранения изображения"), QDir::currentPath());

if (!directory.isEmpty()) {

QString name = legend+QDateTime::currentDateTime().toString("dd\_MM\_yyyy HH\_mm\_ss");

if (!legend.isEmpty()) directory.append("/"+name+".png");

else {

ui->statusbar->showMessage("Постройте график для сохранения");

return;

}

if (ui->widget\_2->savePng(directory)) {

ui->statusbar->showMessage("График успешно сохранен");

}

else ui->statusbar->showMessage("График не сохранен. Повторите попытку");

}

}

void User::on\_Connect\_clicked()

{

if (!emul->thisPort.isOpen()) {

emul->*ConnectPort*();

if (emul->thisPort.isOpen()){

ui->statusbar->showMessage("Соединение установленно");

}

else{

ui->statusbar->showMessage("Соединение не установленно");

}

}

else {

ui->statusbar->showMessage("Соединение уже установленно");

}

}

void User::on\_Disconnect\_clicked()

{

if (emul->thisPort.isOpen()) {

emul->*DisconnectPort*();

if (!emul->thisPort.isOpen()){

ui->statusbar->showMessage("Разъединено успешно");

}

else{

ui->statusbar->showMessage("Разъединение не удалось");

}

}

else {

ui->statusbar->showMessage("Уже разъединено");

}

}