ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ГАЗА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
ИМЕНИ И.М. ГУБКИНА»

ФАКУЛЬТЕТ КОМПЛЕКСНОЙ БЕЗОПАСНОСТИ ТЭК
КАФЕДРА БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Лабораторная работа №9

по дисциплине «Специализированные языки и технологии программирования»

на тему «QIODevice, работа с внешними устройствами»

Выполнил студент:
группы КА-22-06
Воронин Алексей Дмитриевич

Преподаватель:
Греков Владимир Сергеевич

Москва, 2025

Оглавление

Цель работы 3

Задание 3

ЧАСТЬ 1 - Детальные инструкции к выполнению 4

ЧАСТЬ 2 - Тестирование приложения 15

Самостоятельная работа 17

ЗАКЛЮЧЕНИЕ 22

Контрольные вопросы 23

Цель работы

Целью данной лабораторной работы является разработка приложения на Qt, которое взаимодействует с внешними устройствами через различные интерфейсы (например, Bluetooth). Студенты изучат классы QIODevice и связанные с ними инструменты для обмена данными с устройствами. Это даст понимание принципов работы с внешними устройствами и коммуникационными протоколами.

Задание

Разработать приложение для обмена файлами по Bluetooth, которое: - Эмулирует Bluetooth-соединение между двумя устройствами через TCP-сокеты. - Обеспечивает двусторонний обмен. - Реализует графический интерфейс для управления подключением и передачи данных.

ЧАСТЬ 1 - Детальные инструкции к выполнению

Добавьте в заголовочный файл deviceemulator.cpp следующее:

```
#include "deviceemulator.h"
#include <QMessageBox>
#include <QDebug>

DeviceEmulator::DeviceEmulator(const QString &userType, QObject *parent)
    : QObject(parent), userType(userType), server(new QTcpServer(this)), socket(nullptr)
{
    uuidToPortMap[QUuid("550e8400-e29b-41d4-a716-446655440000")] = 12345;
    uuidToPortMap[QUuid("550e8400-e29b-41d4-a716-446655440001")] = 12346;

    localUuid = (userType == "A")
        ? QUuid("550e8400-e29b-41d4-a716-446655440000")
        : QUuid("550e8400-e29b-41d4-a716-446655440001");

    if (!server->listen(QHostAddress::LocalHost, uuidToPortMap[localUuid])) {
        qCritical() << "Не удалось запустить сервер:" << server->errorString();
    } else {
        qDebug() << "Сервер запущен на порту" << uuidToPortMap[localUuid];
    }

    connect(server, &QTcpServer::newConnection, this, &DeviceEmulator::onNewConnection);
}

DeviceEmulator::~DeviceEmulator() {
    disconnect();
    server->close();
}

bool DeviceEmulator::connectToDevice(const QString &uuid) {
    QUuid targetUuid(uuid);
    if (!uuidToPortMap.contains(targetUuid) || targetUuid == localUuid) {
        return false;
    }
```

```
        if (socket) {
            socket->disconnectFromHost();
            socket->deleteLater();
        }

        socket = new QTcpSocket(this);
        connect(socket, &QTcpSocket::readyRead, this,
&DeviceEmulator::onReadyRead);
        connect(socket, &QTcpSocket::disconnected, this,
&DeviceEmulator::onDisconnected);
        connect(socket, &QTcpSocket::errorOccurred, this,
&DeviceEmulator::onSocketError);

        socket->connectToHost(QHostAddress::LocalHost,
uuidToPortMap[targetUuid]);

        if (!socket->waitForConnected(3000)) {
            qWarning() << "Не удалось подключиться:" << socket->errorString();
            return false;
        }

        remoteUuid = targetUuid;
        emit connectionEstablished();
        return true;
    }

    void DeviceEmulator::sendData(const QString &data) {
        if (socket && socket->state() == QAbstractSocket::ConnectedState) {
            socket->write(data.toUtf8());
            socket->flush();
        }
    }

    void DeviceEmulator::disconnect() {
        if (socket) {
            socket->disconnectFromHost();
        }
    }

    bool DeviceEmulator::isConnected() const {
        return socket && socket->state() == QAbstractSocket::ConnectedState;
```

```
}

void DeviceEmulator::onNewConnection() {
    if (socket) {
        socket->disconnectFromHost();
        socket->deleteLater();
    }

    socket = server->nextPendingConnection();
    connect(socket, &QTcpSocket::readyRead, this,
&DeviceEmulator::onReadyRead);
    connect(socket, &QTcpSocket::disconnected, this,
&DeviceEmulator::onDisconnected);
    connect(socket, &QTcpSocket::errorOccurred, this,
&DeviceEmulator::onSocketError);

    quint16 peerPort = socket->peerPort();
    remoteUuid = (peerPort == 12345)
        ? QUuid("550e8400-e29b-41d4-a716-446655440000")
        : QUuid("550e8400-e29b-41d4-a716-446655440001");

    emit connectionEstablished();
}

void DeviceEmulator::onReadyRead() {
    if (socket) {
        QByteArray data = socket->readAll();
        emit dataReceived(QString::fromUtf8(data));
    }
}

void DeviceEmulator::onDisconnected() {
    emit connectionLost();
}

void DeviceEmulator::onSocketError(QAbstractSocket::SocketError error) {
    qWarning() << "Ошибка сокета:" << error;
    emit connectionLost();
}
```

Добавьте в deviceemulator.h следующее:

```
#ifndef DEVICEEMULATOR_H
#define DEVICEEMULATOR_H

#include <QObject>
#include <QTcpServer>
#include <QTcpSocket>
#include <QUuid>

class DeviceEmulator : public QObject
{
    Q_OBJECT

public:
    explicit DeviceEmulator(const QString &userType, QObject *parent =
nullptr);
    ~DeviceEmulator();

    bool connectToDevice(const QString &uuid);
    void sendData(const QString &data);
    void disconnect();
    bool isConnected() const;

signals:
    void dataReceived(const QString &data);
    void connectionEstablished();
    void connectionLost();

private slots:
    void onNewConnection();
    void onReadyRead();
    void onDisconnected();
    void onSocketError(QAbstractSocket::SocketError error);

private:
    QTcpServer *server;
    QTcpSocket *socket;
    QString userType;
    QUuid localUuid;
    QUuid remoteUuid;
    QMap<QUuid, quint16> uuidToPortMap;
};
```

```
#endif // DEVICEEMULATOR_H
```

Добавьте в mainwindow.cpp следующее:

```
#include "mainwindow.h"

MainWindow::MainWindow(const QString &userType, QWidget *parent)
    : QMainWindow(parent), currentUser(userType),
connectedDeviceName("")
{
    QWidget *centralWidget = new QWidget(this);
    setCentralWidget(centralWidget);

    QVBoxLayout *mainLayout = new QVBoxLayout(centralWidget);
    QHBoxLayout *messageLayout = new QHBoxLayout();
    QHBoxLayout *buttonLayout = new QHBoxLayout();

    userLabel = new QLabel(QString("Пользователь %1").arg(userType));
    chatDisplay = new QTextEdit();
    chatDisplay->setReadOnly(true);
    messageEdit = new QLineEdit();
    sendButton = new QPushButton("Отправить");
    scanButton = new QPushButton("Поиск устройств");
    disconnectButton = new QPushButton("Отключиться");
    connectionStatus = new QLabel("Не подключено");

    statusBar = new QStatusBar();
    setStatusBar(statusBar);

    messageLayout->addWidget(messageEdit);
    messageLayout->addWidget(sendButton);

    buttonLayout->addWidget(scanButton);
    buttonLayout->addWidget(disconnectButton);

    mainLayout->addWidget(userLabel);
    mainLayout->addWidget(chatDisplay);
    mainLayout->addLayout(messageLayout);
    mainLayout->addLayout(buttonLayout);
    mainLayout->addWidget(connectionStatus);
```

```
        setWindowTitle(QString("Bluetooth эмулятор - Пользователь %1").arg(userType));
        resize(500, 400);

        deviceEmulator = new DeviceEmulator(userType, this);

        connect(scanButton, &QPushButton::clicked, this, &MainWindow::on_scanButton_clicked);
        connect(sendButton, &QPushButton::clicked, this, &MainWindow::on_sendButton_clicked);
        connect(disconnectButton, &QPushButton::clicked, this, &MainWindow::on_disconnectButton_clicked);
        connect(deviceEmulator, &DeviceEmulator::dataReceived, this, &MainWindow::onDataReceived);
        connect(deviceEmulator, &DeviceEmulator::connectionEstablished, this, &MainWindow::onConnectionEstablished);
        connect(deviceEmulator, &DeviceEmulator::connectionLost, this, &MainWindow::onConnectionLost);

        sendButton->setEnabled(false);
        disconnectButton->setEnabled(false);
    }

    MainWindow::~MainWindow()
    {
        delete deviceEmulator;
    }

    void MainWindow::on_scanButton_clicked()
    {
        DeviceSelectionDialog dialog(currentUser, this);
        connect(&dialog, &DeviceSelectionDialog::deviceSelected, this, &MainWindow::onDeviceSelected);
        dialog.exec();
    }

    void MainWindow::on_sendButton_clicked()
    {
        QString message = messageEdit->text();
        if (!message.isEmpty()) {
```

```
            deviceEmulator->sendData(message);
            chatDisplay->append(QString("[Вы]: %1").arg(message));
            messageEdit->clear();
        }
    }

    void MainWindow::on_disconnectButton_clicked()
    {
        deviceEmulator->disconnect();
    }

    void MainWindow::onDeviceSelected(const QString &uuid)
    {
        if (deviceEmulator->connectToDevice(uuid)) {
            connectedDeviceName = (uuid == "550e8400-e29b-41d4-a716-
446655440000") ? "Устройство A" : "Устройство B";
            statusBar->showMessage(QString("Подключение к
%1...").arg(connectedDeviceName));
        } else {
            statusBar->showMessage("Ошибка подключения", 3000);
        }
    }

    void MainWindow::onDataReceived(const QString &data)
    {
        chatDisplay->append(QString("[%1]:
%2").arg(connectedDeviceName).arg(data));
    }

    void MainWindow::onConnectionEstablished()
    {
        connectionStatus->setText(QString("Подключено к
%1").arg(connectedDeviceName));
        sendButton->setEnabled(true);
        scanButton->setEnabled(false);
        disconnectButton->setEnabled(true);
        statusBar->showMessage("Подключение установлено", 3000);
    }

    void MainWindow::onConnectionLost()
    {
```

```
    connectionStatus->setText("Не подключено");
    sendButton->setEnabled(false);
    scanButton->setEnabled(true);
    disconnectButton->setEnabled(false);
    statusBar->showMessage("Соединение разорвано", 3000);
    chatDisplay->append("> Соединение потеряно");
}
```

Добавьте в mainwindow.h следующее:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTextEdit>
#include <QLineEdit>
#include <QPushButton>
#include <QLabel>
#include <QStatusBar>
#include <QVBoxLayout>
#include <QHBoxLayout>
#include "deviceemulator.h"
#include "deviceselectiondialog.h"

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(const QString &userType, QWidget *parent =
nullptr);
    ~MainWindow();

private slots:
    void on_scanButton_clicked();
    void on_sendButton_clicked();
    void on_disconnectButton_clicked();
    void onDeviceSelected(const QString &uuid);
    void onDataReceived(const QString &data);
    void onConnectionEstablished();
    void onConnectionLost();
```

```
private:
    QTextEdit *chatDisplay;
    QLineEdit *messageEdit;
    QPushButton *sendButton;
    QPushButton *scanButton;
    QPushButton *disconnectButton;
    QLabel *userLabel;
    QLabel *connectionStatus;
    QStatusBar *statusBar;

    DeviceEmulator *deviceEmulator;
    QString currentUser;
    QString connectedDeviceName;
};
#endif // MAINWINDOW_H
```

Добавьте в main.cpp следующее:

```
#include <QApplication>
#include "mainwindow.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QString userType = (argc > 1) ? argv[1] : "A";

    MainWindow w(userType);
    w.show();

    return a.exec();
}
```

Добавьте в deviceselectiondialog.cpp следующее:

```
#include "deviceselectiondialog.h"

DeviceSelectionDialog::DeviceSelectionDialog(const QString &userType,
QWidget *parent)
    : QDialog(parent), userType(userType)
{
    QVBoxLayout *mainLayout = new QVBoxLayout(this);
```

```
        devicesList = new QListWidget(this);
        connectButton = new QPushButton("Подключиться", this);

        if (userType != "A") {
            devicesList->addItem("Устройство A (550e8400-e29b-41d4-a716-
446655440000)");
        }
        if (userType != "B") {
            devicesList->addItem("Устройство B (550e8400-e29b-41d4-a716-
446655440001)");
        }

        mainLayout->addWidget(devicesList);
        mainLayout->addWidget(connectButton);

        setLayout(mainLayout);
        setWindowTitle("Выбор устройства");
        resize(400, 200);

        connect(connectButton, &QPushButton::clicked, this,
&DeviceSelectionDialog::on_connectButton_clicked);
    }

    DeviceSelectionDialog::~DeviceSelectionDialog()
    {

    }

    void DeviceSelectionDialog::on_connectButton_clicked()
    {
        if (devicesList->currentItem()) {
            QString selectedDevice = devicesList->currentItem()->text();
            QString uuid;

            if (selectedDevice.contains("Устройство A")) {
                uuid = "550e8400-e29b-41d4-a716-446655440000";
            } else {
                uuid = "550e8400-e29b-41d4-a716-446655440001";
            }
```

```
            emit deviceSelected(uuid);
            accept();
        }
    }
```

Добавьте в `deviceselectiondialog.h` следующее:

```
#ifndef DEVICESELECTIONDIALOG_H
#define DEVICESELECTIONDIALOG_H

#include <QDialog>
#include <QListWidget>
#include <QPushButton>
#include <QVBoxLayout>
#include <QUuid>

class DeviceSelectionDialog : public QDialog
{
    Q_OBJECT

public:
    explicit DeviceSelectionDialog(const QString &userType, QWidget *parent
= nullptr);
    ~DeviceSelectionDialog() override;

signals:
    void deviceSelected(const QString &uuid);

private slots:
    void on_connectButton_clicked();

private:
    QListWidget *devicesList;
    QPushButton *connectButton;
    QString userType;
};

#endif // DEVICESELECTIONDIALOG_H
```

Для корректной работы в CMakeLists.txt нужно добавить определенную конфигурацию:

```
cmake_minimum_required(VERSION 3.5)
project(BluetoothEmulator LANGUAGES CXX)

set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_AUTOMOC ON)

find_package(Qt6 REQUIRED COMPONENTS Core Gui Widgets Network)

add_executable(BluetoothEmulator
  main.cpp
  mainwindow.cpp
  deviceemulator.cpp
  deviceselectiondialog.cpp
)

target_link_libraries(BluetoothEmulator PRIVATE
  Qt6::Core
  Qt6::Gui
  Qt6::Widgets
  Qt6::Network
)
```

ЧАСТЬ 2 - Тестирование приложения

Нужно запустить два экземпляра, для этого в qtcreator соберем проект и перейдем в терминал где выполним следующие команды:

./BluetoothEmulator A

./BluetoothEmulator B

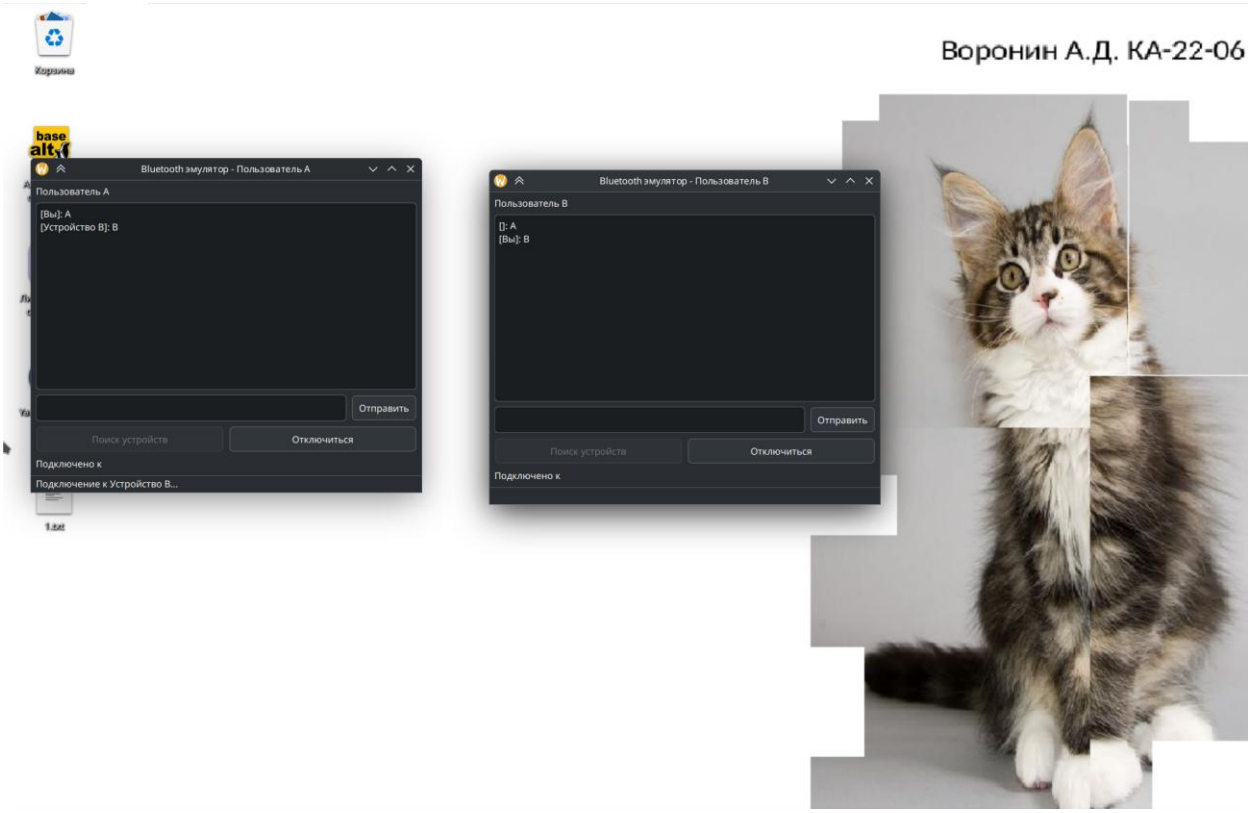

Рисунок 1 - Тестирование

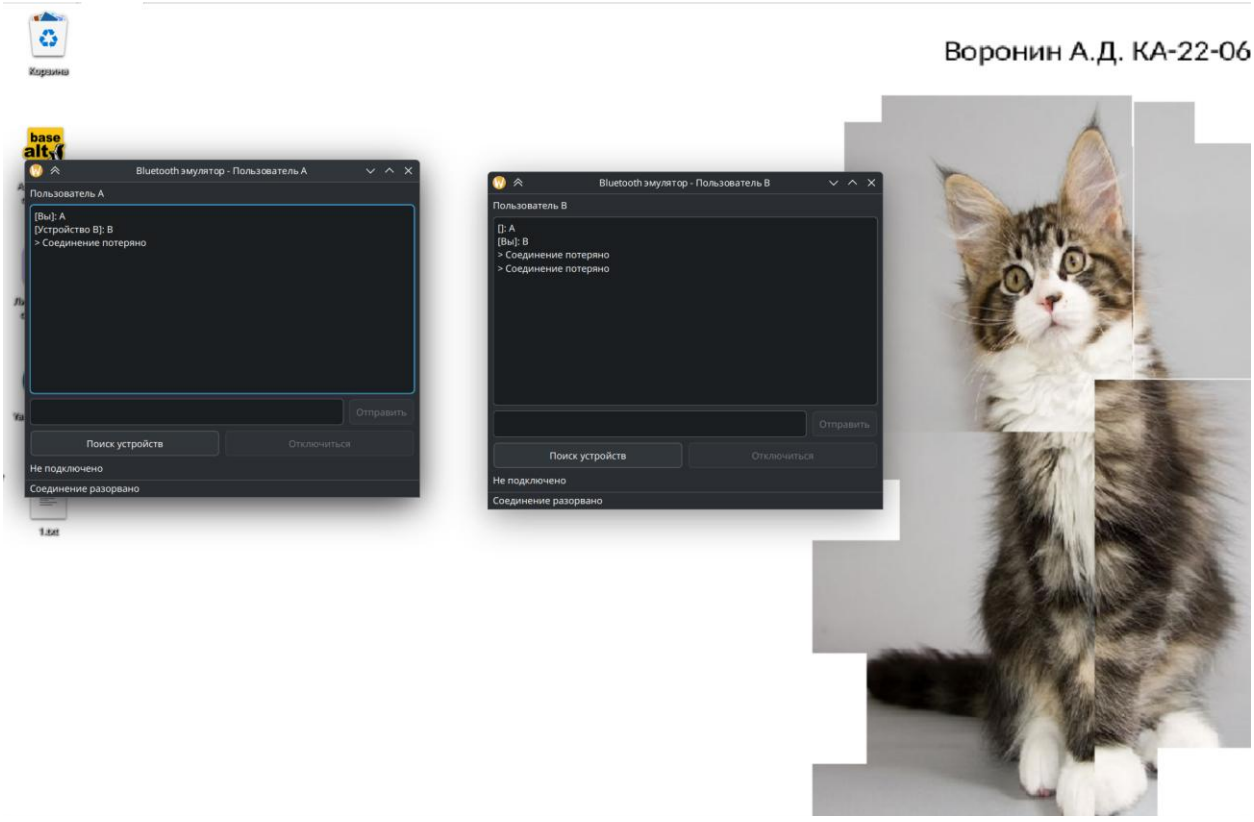

Рисунок 2 – Разрыв соединения

Самостоятельная работа

Задание для самостоятельной работы

- Реализуйте передачу файлов между устройствами.
- Реализуйте ГОСТ Р 34.12-2015 "Магма" (российский стандарт блочного шифрования с длиной блока 64 бита) для защиты данных

В mainwindow добавлена кнопка "Отправить файл", слот для выбора и отправки файла через QFileDialog, и обработка получения файлов с уведомлением в чате. В deviceemulator реализованы методы sendFile для отправки файлов с шифрованием и onReadyRead для получения файлов с расшифровкой.

Также был создан новый класс Magma, который реализует шифрование 64-битных блоков.

Magma.cpp:

```
#include "magma.h"
#include <QDebug>

const uint8_t Magma::sboxes[8][16] = {
   {12, 4, 6, 2, 10, 5, 11, 9, 14, 8, 13, 7, 0, 3, 15, 1},
   {6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15},
   {11, 3, 5, 8, 2, 15, 10, 13, 14, 1, 7, 4, 12, 9, 6, 0},
   {12, 8, 2, 1, 13, 4, 15, 6, 7, 0, 10, 5, 3, 14, 9, 11},
   {7, 15, 5, 10, 8, 1, 6, 13, 0, 9, 3, 14, 11, 4, 2, 12},
   {5, 13, 15, 6, 9, 2, 12, 10, 11, 7, 8, 1, 4, 3, 14, 0},
   {8, 14, 2, 5, 6, 9, 1, 12, 15, 4, 11, 13, 13, 10, 7, 0},
   {1, 7, 14, 13, 0, 5, 8, 3, 4, 15, 10, 6, 9, 12, 11, 2}
};

Magma::Magma() {}

void Magma::setKey(const QByteArray &key)
{
   if (key.size() != 32) {
      qWarning() << "Invalid key size:" << key.size();
      return;
   }
```

```
    subkeys.resize(32);
    for (int i = 0; i < 8; i++) {
        uint32_t k = 0;
        for (int j = 0; j < 4; j++) {
            k |= static_cast<uint32_t>(static_cast<uint8_t>(key[i*4 + j])) << ((3-
j)*8);
        }
        subkeys[i] = k;
    }
    // Первые 24 раунда: ключи K1..K8 повторяются 3 раза
    for (int i = 0; i < 24; i++) {
        subkeys[i] = subkeys[i % 8];
    }
    // Последние 8 раундов: ключи K8..K1
    for (int i = 24; i < 32; i++) {
        subkeys[i] = subkeys[31 - i];
    }
}

uint32_t Magma::t(uint32_t a)
{
    uint32_t result = 0;
    for (int i = 0; i < 8; i++) {
        uint8_t nibble = (a >> (i*4)) & 0xF;
        uint8_t sbox_val = sboxes[7-i][nibble];
        result |= static_cast<uint32_t>(sbox_val) << (i*4);
    }
    return result;
}

uint32_t Magma::G(uint32_t a, uint32_t k)
{
    uint32_t temp = (a + k) & 0xFFFFFFFF; // Модуль 2^32
    temp = t(temp);
    temp = (temp << 11) | (temp >> 21);
    return temp;
}

QByteArray Magma::encrypt(const QByteArray &block)
{
    if (block.size() != 8) {
```

```
            qWarning() << "Invalid block size for encryption:" << block.size();
            return QByteArray();
        }

        // Разбиваем блок на две 32-битные части (big-endian)
        uint32_t left = 0, right = 0;
        for (int i = 0; i < 4; i++) {
            left |= static_cast<uint32_t>(static_cast<uint8_t>(block[i])) << ((3-i)*8);
            right |= static_cast<uint32_t>(static_cast<uint8_t>(block[i+4])) << ((3-
i)*8);
        }

        // 32 раунда шифрования
        for (int i = 0; i < 32; i++) {
            uint32_t temp = right;
            right = left ^ G(right, subkeys[i]);
            left = temp;
        }

        // Формируем выходной блок
        QByteArray result(8, 0);
        for (int i = 0; i < 4; i++) {
            result[i] = static_cast<char>((right >> ((3-i)*8)) & 0xFF);
            result[i+4] = static_cast<char>((left >> ((3-i)*8)) & 0xFF);
        }
        return result;
    }

    QByteArray Magma::decrypt(const QByteArray &block)
    {
        if (block.size() != 8) {
            qWarning() << "Invalid block size for decryption:" << block.size();
            return QByteArray();
        }

        // Разбиваем блок на две 32-битные части (big-endian)
        uint32_t left = 0, right = 0;
        for (int i = 0; i < 4; i++) {
            left |= static_cast<uint32_t>(static_cast<uint8_t>(block[i])) << ((3-i)*8);
            right |= static_cast<uint32_t>(static_cast<uint8_t>(block[i+4])) << ((3-
i)*8);
```

```
    }

    // 32 раунда дешифрования (обратный порядок ключей)
    for (int i = 0; i < 32; i++) {
        uint32_t temp = right;
        right = left ^ G(right, subkeys[31-i]);
        left = temp;
    }

    // Формируем выходной блок
    QByteArray result(8, 0);
    for (int i = 0; i < 4; i++) {
        result[i] = static_cast<char>((right >> ((3-i)*8)) & 0xFF);
        result[i+4] = static_cast<char>((left >> ((3-i)*8)) & 0xFF);
    }
    return result;
}
```

Magma.h:

```
#ifndef MAGMA_H
#define MAGMA_H

#include <QByteArray>
#include <vector>

class Magma {
public:
    Magma();
    void setKey(const QByteArray &key);
    QByteArray encrypt(const QByteArray &block);
    QByteArray decrypt(const QByteArray &block);

private:
    std::vector<uint32_t> subkeys;
    static const uint8_t sboxes[8][16];
    uint32_t G(uint32_t a, uint32_t k);
    uint32_t t(uint32_t a);
};

#endif // MAGMA_H
```

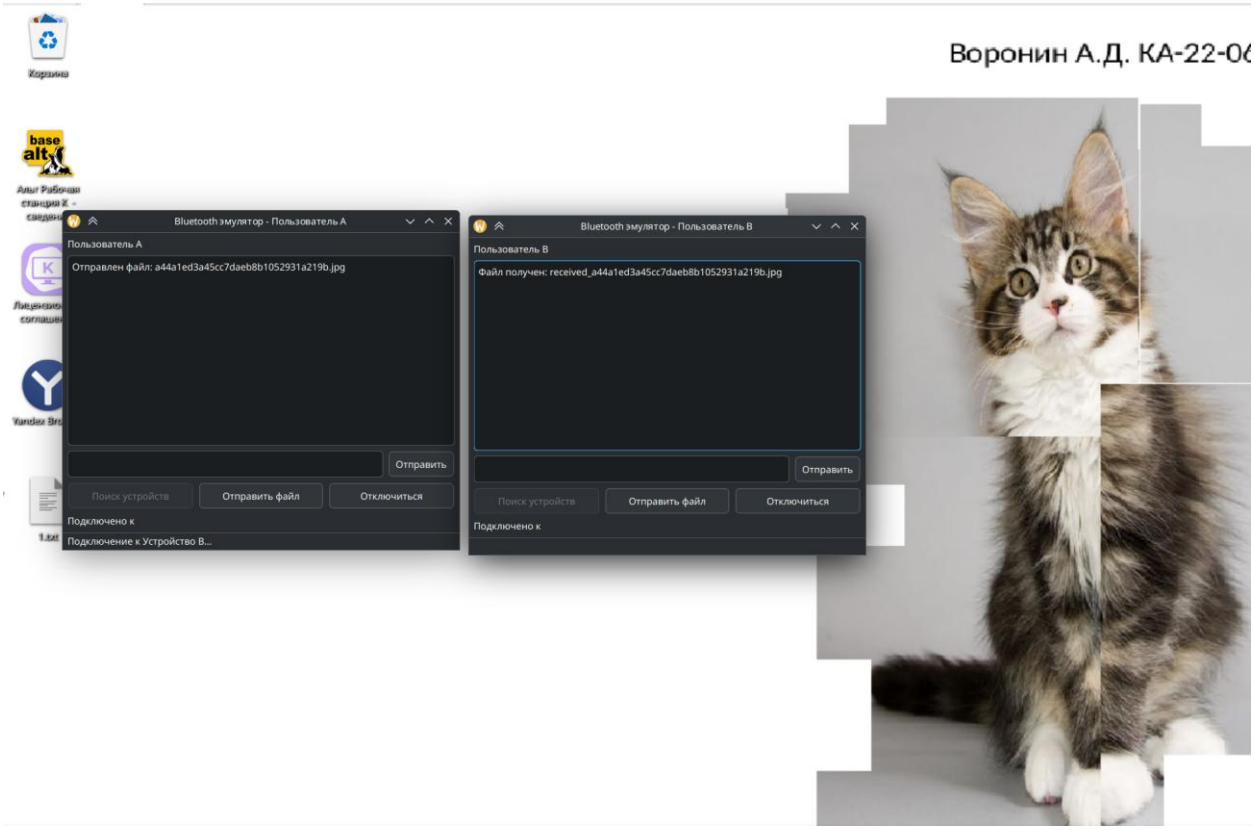

Рисунок 3 – Отправка файлов

ЗАКЛЮЧЕНИЕ

В заключение, подводя итог проделанной работе, все поставленные в начале цели и задания данной лабораторной работы были достигнуты в полном объеме.

Контрольные вопросы

1. Какой класс в Qt является базовым для работы с устройствами ввода-вывода?

QIODevice

2. Для чего используются UUID и как они связаны с портами?

UUID (Universally Unique Identifier) используются для уникальной идентификации устройств в эмуляторе Bluetooth, а их связь с портами обеспечивает привязку каждого устройства к определённому TCP-порту для установления соединения.

3. Какие основные методы предоставляет QIODevice?

Чтение read()

Запись write()

Управление open()close()

4. Как можно модифицировать код для работы с реальным Bluetooth (без эмуляции через TCP)?

Qt Bluetooth использует другие классы и подходы для работы с Bluetooth, в частности, QBluetoothSocket и QBluetoothServer вместо QTcpSocket и QTcpServer.

В CMakeLists.txt нужно добавить модуль Qt Bluetooth, заменив Qt6::Network на Qt6::Bluetooth. Это обеспечит доступ к классам Qt Bluetooth API. А также Класс DeviceEmulator нужно переработать, заменив TCP-сокеты на Bluetooth.