Министерство образования и науки Российской Федерации

федеральное государственное автономное образовательное учреждение
высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»

Институт дополнительного образования
Высшая инженерная школа

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Разработка программного адаптера джойстика для системы беспроводного управления квадрокоптером

по программе профессиональной переподготовки:
«Разработчик прикладного программного обеспечения (Языки C и C++)»

Исполнитель: Пономарчук Денис Александрович

Руководитель: ст. преподаватель Полубенцева Марина Игоревна

Постановка цели и задач

Целью работы является разработка программного адаптера для системы беспроводного управления квадрокоптером, обеспечивающего прием данных от устройство ввода (джойстика), и передачу данных другим компонентам системы управления.

Задачи работы:

- Обзор предметной области;
- Формирование требований к программе;
- Разработка архитектуры программы;
- Программная реализация;
- Тестирование.

Предметная область

Рисунок 1 – Структурная схема системы управления

Основная функциональность:

- Прием данных от джойстика;
- Визуализация положения джойстика;
- Документирование в базу данных приходящих данных от джойстика;
- Передача данных по UDP с возможностью настройки частоты передачи.

- Входом для адаптера является поток данных, описывающих положение рукоятки джойстика. Положение рукоятки джойстика – выражает желаемое значения скоростей по крену, тангажу и рысканию.
- Выходные данные – это обработанные значения, которые призваны уменьшить погрешности при управлении и снизить влияние шумов при передаче. Это необходимо для обеспечения устойчивости аппарата.

Формирование требований к программе

К разрабатываемому программному адаптеру предъявляются следующие требования:

- Функционирования на ОС Linux;
- Разработка на языке C++ с применением Qt 4.8;
- Передача данных по сетевому протоколу UDP с возможностью задания частоты передачи;
- Документирование принимаемых данных.

Архитектура программы

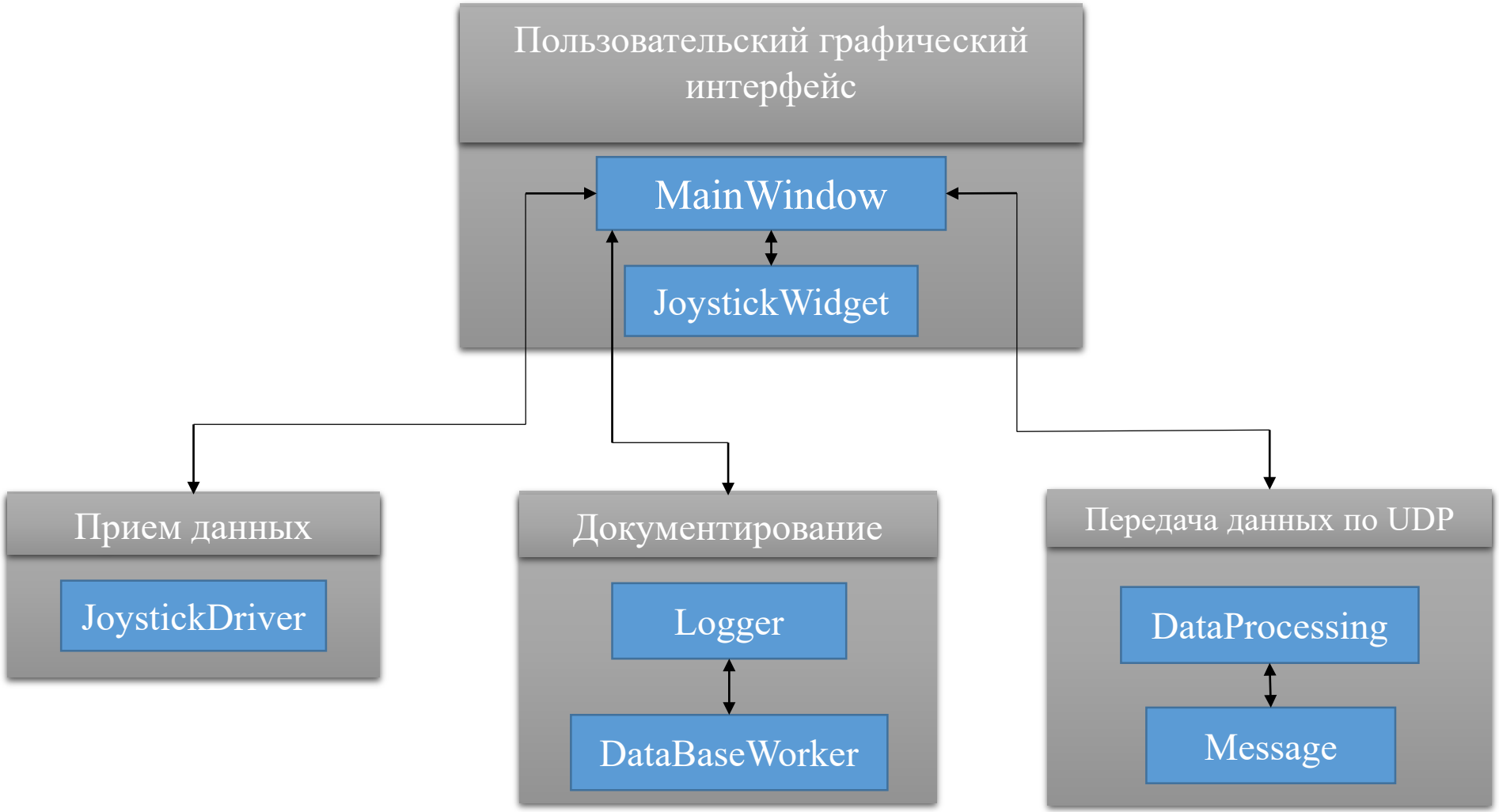


Рисунок 2 – Архитектура программы

Прием данных

Класс JoystickDriver	
Переменные:	
JoystickEvent *m_jsData	Указатель на структуру для принятия данных.
Методы:	
bool start();	Запуск приема данных
void stop();	Остановка приема данных
bool setPath(const QString&);	Установка пути к устройству
void slotReadData();	Чтение данных, соединен с сигналом readyRead()
void signalJsEvent (const JoystickDriver::JoystickEvent&);	Передача структуры для документирования.
void signalValueAxis (unsigned char, short);	Сигнал значения по оси, передает номер оси, значение по оси.
Всего: 16 методов, 6 переменных.	

```
    //! Возможные значения типа данных от джойстика.
    enum typeJsEvent : unsigned char
    {
        button = 0x01,  //!< Event associated with the button.
        axis   = 0x02,  //!< Event associated with the axis.
        init   = 0x81   //!< Initializaton.
    };

struct JoystickEvent
{
    unsigned int    time;
    short           value;
    typeJsEvent     type;
    unsigned char   number;
};
```

Рисунок 3 – Структура принимаемых данных

Документирование

Класс DataBaseWorker	
Переменные:	
QSqlDatabase m_dataBase;	Объект базы данных.
QString m_typeDB;	Тип базы данной.
QString m_nameDB;	Имя базы данной.
Методы:	
void openDataBase();	Метод открытия базы данной (или создание).
void createTableDataBase();	Создание таблицы в БД.
void slotInsertToBase(const JoystickDriver::JoystickEvent &msg);	Слот для вставки msg структуры в БД.
void slotConnectToDB();	Вызывает методы открытия БД и создания таблицы.
void closeDB();	Закртыие БД.
Всего: 11 методов, 4 переменных.	

```
Logger::Logger(QObject *parent) : QObject(parent)
{
    m_DBworker  = new DataBaseWorker;
    m_thread    = new QThread(this);
    m_DBworker->moveToThread(m_thread);

    //! По запуску потока, происходит соединение с базой данных.
    connect(m_thread, SIGNAL(started()), m_DBworker, SLOT(slotConnectToDB()));

    //! По сигналу остановки потока удалить объект в этом потоке.
    connect(m_thread, SIGNAL(finished()), m_DBworker, SLOT(deleteLater()));

    //! Соединение для передачи данных для документирования.
    connect(this, SIGNAL(signalSendMsgToWorker(JoystickDriver::JoystickEvent)),
                        m_DBworker,
                        SLOT(slotInsertToBase (JoystickDriver::JoystickEvent)),
                        Qt::QueuedConnection);

    m_thread->start();
}
```

Рисунок 4 – Конструктор Logger

Передача данных по UDP

Класс DataProcessing	
Переменные:	
QUdpSocket *m_socketUdpSend;	Указатель на сокет UDP.
QTimer *m_timer;	Указатель на таймер.
unsigned short m_destPort;	Порт для передачи.
QString m_destIP;	IP-адрес для передачи.
Методы:	
void startUdp();	Запуск передачи (таймера).
void stopUdp();	Остановка передачи (таймера).
void slotSendDatagram();	Отправка датаграммы.
Всего: 12 методов, 8 переменных.	

Класс Message	
Переменные:	
QByteArray m_data;	Данные для передачи.
Методы:	
void setType (uint8_t type);	Установка типа (отправителя сообщения).
void setData (const QByteArray &data);	Установка данных.
operator const QByteArray &() const;	Оператор приведения к константной ссылке на QBytreArray.

```
struct msg_hdr
{
    uint16_t    count;
    uint16_t    length;
    uint8_t     type;

};

struct msg
{
    msg_hdr hdr;
    uint8_t data[];
};
```

Рисунок 5 – Протокол обмена по UDP

Реализация графического интерфейса

На рисунке 6 представлено главное окно программы. В области 1 расположена визуализация положения джойстика, в области 2 находятся настройки передачи данных по UDP и включение (остановка) передачи. В 3 области находятся функциональные клавиши начала и остановки приема входных данных от джойстика. В области 4 отображается состояние приема и передачи.

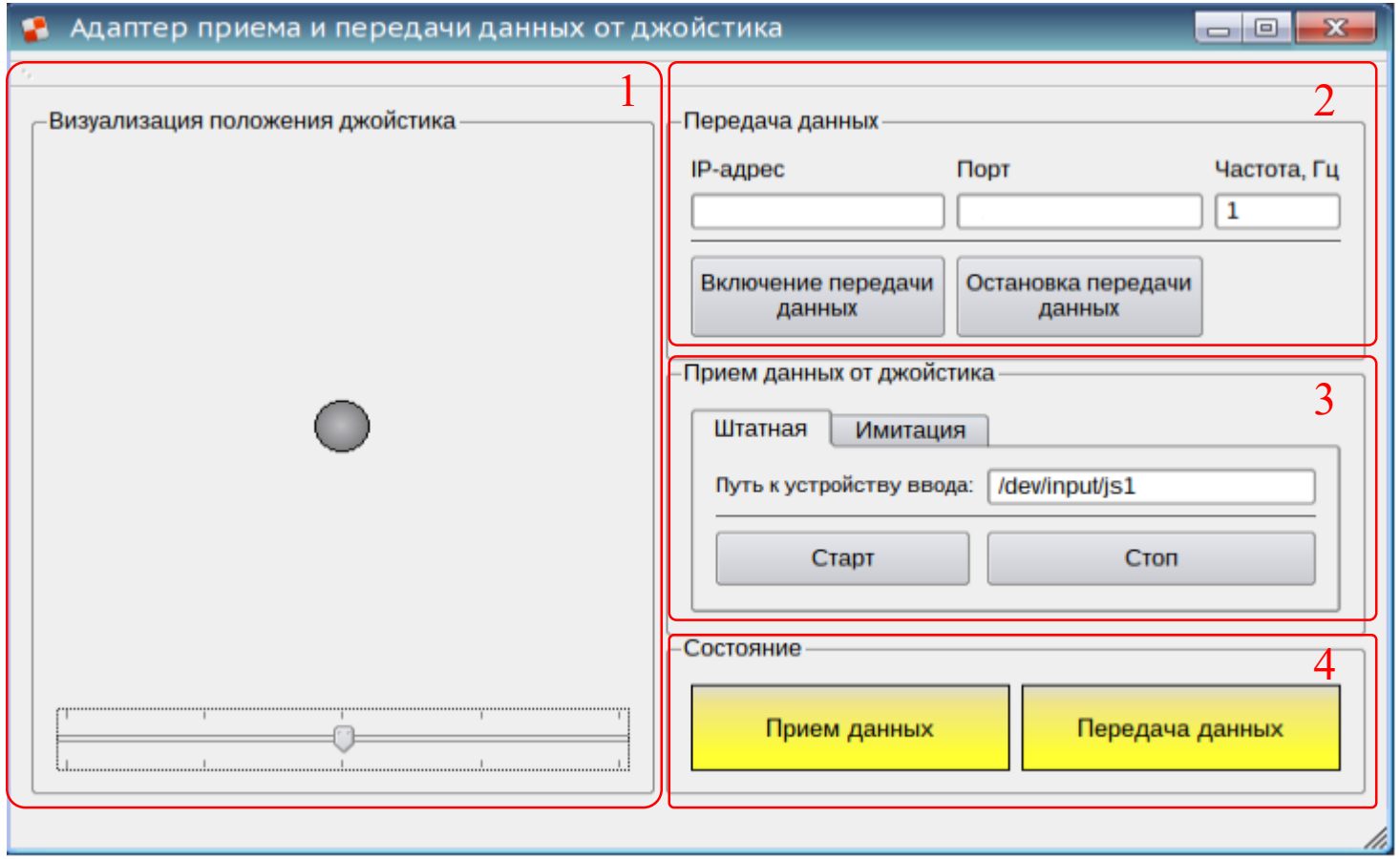
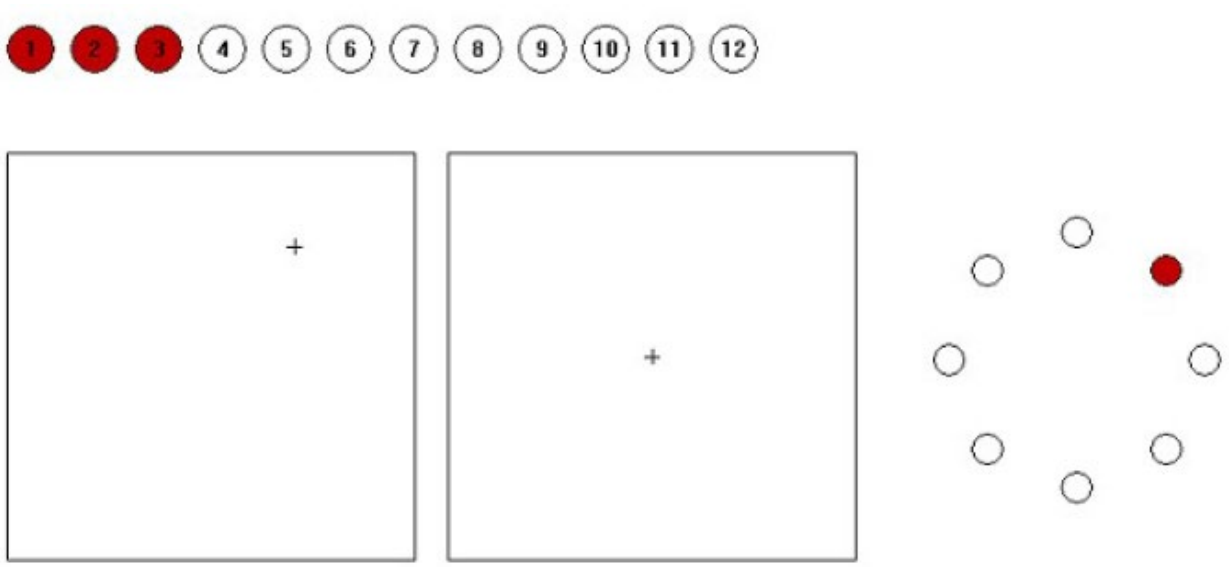


Рисунок 6 – Виджет представления данных вводимых джойстиком

Тестирование

- Тест принятия данных от джойстика;
- Тест передачи данных UDP.



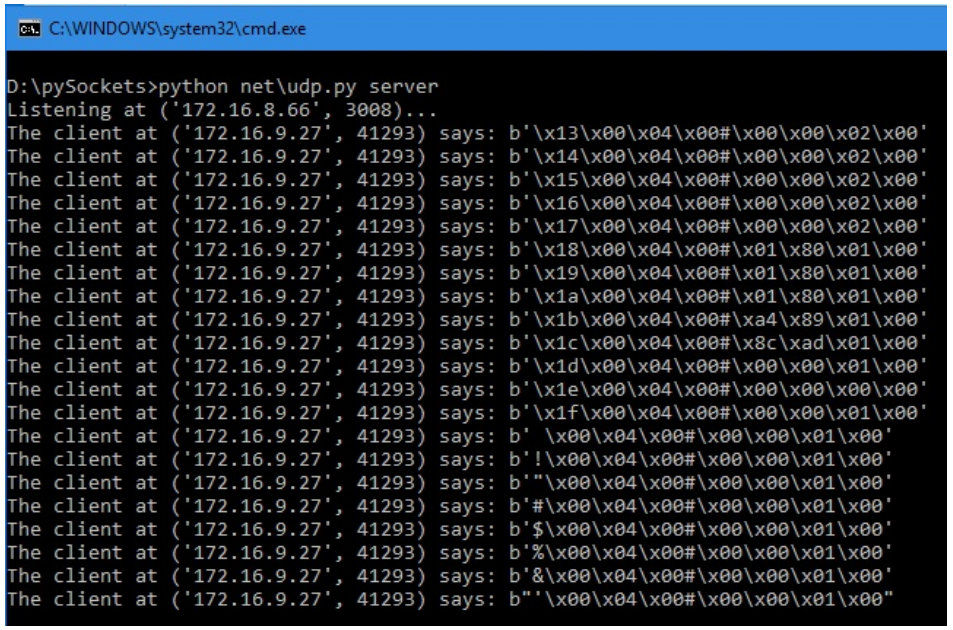


Рисунок 7 – Окна тестирования работоспособности

Заключение

В результате выполнение ВКР был разработан программный адаптер, являющийся компонентом системы управления квадрокоптером, который обеспечивает следующую функциональность:

- Прием данных от устройства ввода с последующей обработкой и передачей данных;
- Документирование принятых данных;
- Визуализация положения устройства ввода.

Произведено тестирование работоспособности применением двух тестов: на принятие данных и передачу данных по UDP.