МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет» Факультет автоматики и вычислительной техники Кафедра электронных вычислительных машин

Отчет

по производственной практике на АО «Научно-производственное предприятие «Знак», г. Киров

Выполнил студент группы ИВТ-31 /Птахова А.М. /

Руководитель практики от ВятГУ /Исупов К.С. / Руководитель практики от предприятия / /

Киров 2023

|  |  |
| --- | --- |
| Дата | Вид деятельности |
| 27.06.2023 | Получение задания на практику |
| 28.06.2023 – 19.07.2023 | Выполнение задания |
| 20.07.2023 – 21.07.2023 | Подготовить и оформить отчет |

Введение

Производственная практика является неотъемлемой и важной частью учебного процесса. В ходе неё не только закрепляются теоретические знания, полученные во время обучения, но и отрабатываются необходимые практические умения для эффективной работы в профессиональной сфере, осваивается инструментарий для работы и взаимодействия в команде, оттачиваются коммуникативные навыки для общения с коллегами.

Данный документ представляет собой отчет по производственной практике, проходившей в период с 26 июня по 23 июля АО «Научно-производственное предприятие «Знак», г. Киров.

1. Общие сведения о предприятии

АО «НПП «Знак» был создан 19 сентября 1991 года.

Физический адрес: 610000, Российская Федерация, Кировская область, г. Киров, ул. Спасская, д. 30.

Акционерное общество (АО) – это хозяйственное общество, уставный капитал которого разделён на определённое число акций. Акционерное общество несет ответственность по всем своим обязательствам.

Сегодня это предприятие занимается производством радиолокационной, радионавигационной аппаратуры и радиоаппаратуры дистанционного управления.

2. Описание выполненной работы

В соответствии с заданием от организации во время прохождения практики необходимо выполнить подготовку основы для последующего выполнения расчета радио - трассы.

В подготовку основы входят следующие этапы:

- отображение карты;

- получение координат точки;

- отображение статистики с использованием средств графического

интерфейса;

- выполнение перевода пользовательского интерфейса на русский язык;

- выполнение сборки программы под Astra Linux

2.1 Отображение карты

Отображение карты осуществляется при помощи загрузки тайтлов с сервера и отображения их на стороне клиента.

Для реализации этой функции необходимо соединить сигнал, вырабатываемый сервером в результате выполнения запроса о загрузке тайтла, с функцией, которая обработает этот сигнал.

Фрагмент кода для отображения карты приведен ниже:

netFactory = new qmapcontrol::MapNetworkFactory(this);

connect(netFactory, &qmapcontrol::MapNetworkFactory::mapConnected, this, [=](QString mapName, QString mapNameUi, int layerCount)

В результате выполнения функции, в окне программы будет выведена карта. Пример отображаемой карты представлен на рисунке 1.

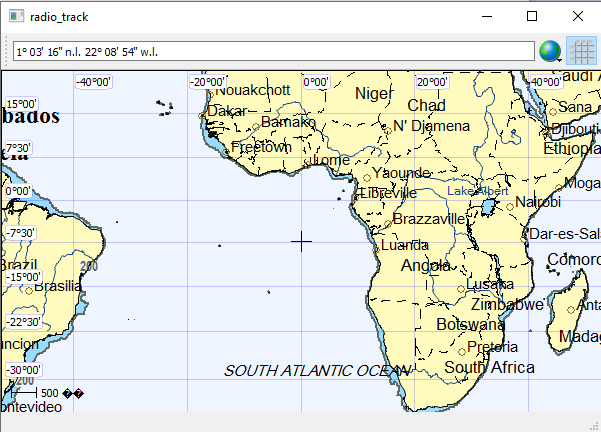


Рисунок 1 – Пример отображаемой карты

2.2 Получение координат точки

Для того, чтобы получить географические координаты точки необходимо определить положение курсора, полученные координаты преобразовать в географические.

2.2.1 Определение положения курсора

Осуществляется при помощи связывания внутреннего сигнала и внешнего сигнала об изменении положения курсора. В свой очередь последний необходимо реализовать самостоятельно. Фрагмент кода, описывающего создание собственного сигнала об изменении позиции курсора приведен ниже:

signals:

void mouseMoveEvent(QMouseEvent\* evnt, QPointF coordinate);

Что касается связывания двух сигналов, то оно происходит при помощи функции connect. Фрагмент кода приведен ниже:

connect(mapControl, qOverload<QMouseEvent\*, QPointF>(

&qmapcontrol::MapControl::mouseMoveEvent),

this, &MapWidget::mouseMoveEvent)

2.2.2 Преобразование координат в географические

Преобразование координат в географические осуществляется при помощи преобразования координат экрана в градусы и представления в виде строки. Фрагмент кода приведен ниже:

auto latitude = Helper::fromDegree(cursor.y());

auto longitude = Helper::fromDegree(cursor.x());

auto coordiante = PelengCaption::placeString(latitude, longitude);

В свою очередь функция fromDegree имеет следующее описание:

static int64\_t fromDegree(float degree)

{

return degree \* 3600000;

}

А функция placeString описана следующим образом:

static QString placeString(int64\_t latitude\_ms, int64\_t longitude\_ms)

{

earth\_math::earth\_point point{

earth\_math::degree\_latitude::from\_milliseconds(latitude\_ms),

earth\_math::degree\_longitude::from\_milliseconds(longitude\_ms)

};

return QString::fromStdString(

point.to\_string(earth\_math::show\_seconds));

}

Отображение координат курсора представлено на рисунке 2.

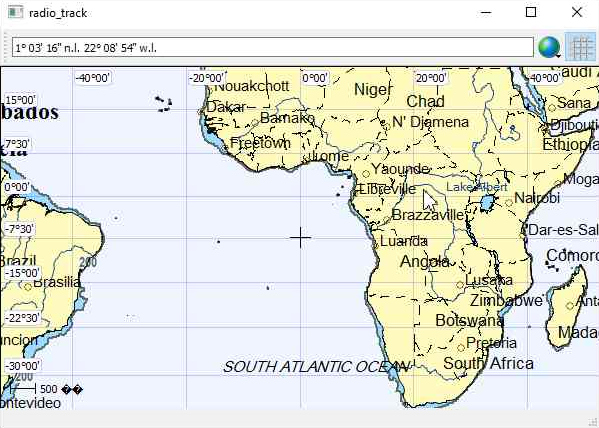


Рисунок 2 – Отображение координат курсора

2.3 Отображение статистики при помощи средств графического интерфейса

В качестве такого средства была выбрана библиотека qwt, а для формы отображения статистики была выбрана гистограмма.

Код, реализующий рисование гистограммы, приведен в приложении А.

Окно приложения с нарисованной гистограммой представлено на рисунке 3.

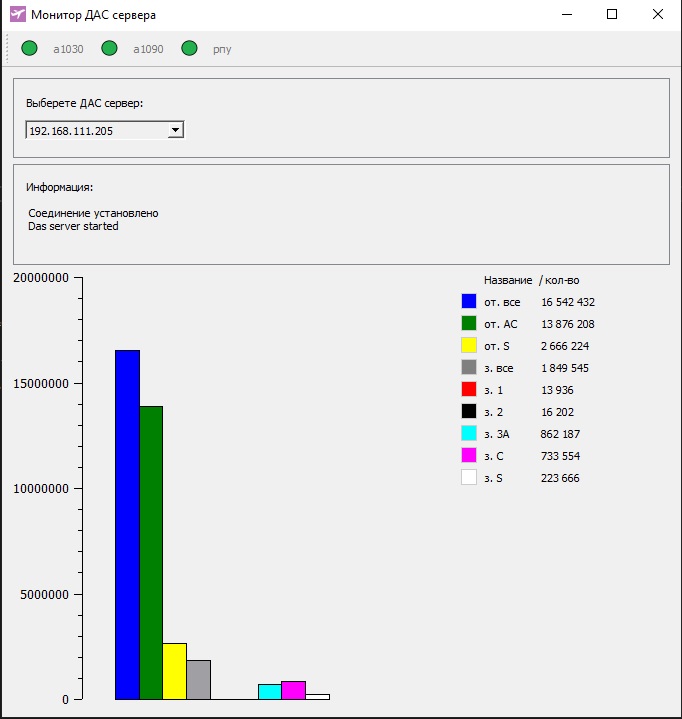


Рисунок 3 – Окно программы с нарисованной гистограммой

2.4 Выполнение перевода пользовательского интерфейса на русский язык

Перевод осуществляется при помощи Qt Linguist. В случае, если необходимо сделать перевод названия компонентов формы, то необходимо:

- создать / обновить файл переводов (.ts);

- запустить qt Linguist и задать перевод для каждого компонента;

- сгенерировать qm файл;

- подключить его к проекту.

Подключение к проекту следует выполнять через ресурсы для проекта. Для этого нужно:

- создать префикс в файле ресурсов;

- включить сгенерированный файл переводов к этому префиксу.

Фрагмент кода, реализующий подключение файла переводов приведен ниже:

QTranslator myTrans;

if (myTrans.load(":/translation/das\_server\_monitor\_ru.qm"))

{

a.installTranslator(&myTrans);

}

a.installTranslator(&myTrans);

Если же необходимо сделать перевод строки, выводимой в одном из компонентов формы, то необходимо написать QObject::tr. Фрагмент кода приведен ниже:

QString translate\_message = QObject::tr("connection estabilished");

Пример перевода интерфейса представлен на рисунке 4.

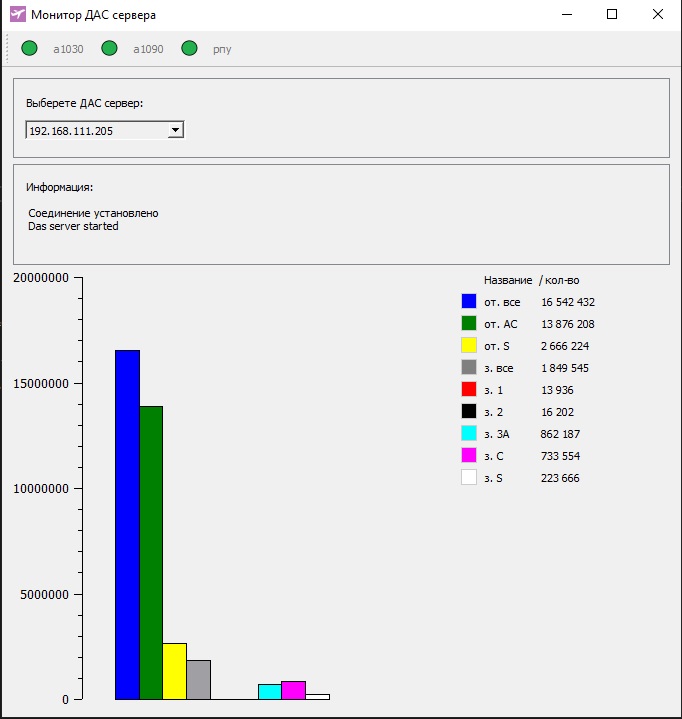


Рисунок 4 – Пример перевода интерфейса

2.5 Сборка проекта под Astra Linux

Код программы, выполняющий сборку под Astra Linux приведен в приложении Б.

Заключение

В ходе выполнения производственной практики были получены навыки работы с библиотекой qwt, инструментом Qt Linguist. Кроме того, были разработаны функции, позволяющие выполнять отображение карты и взаимодействие с ней. А также были получены навыки сборки приложения под другую операционную систему, точнее под Astra Linux.

В результате выполнения задания была подготовлена основа для последующего написания приложения, рассчитывающего радио - трассу.

Приложение А

Листинг кода для отображения гистограммы

QwtPlotHistogram\* histogram = new QwtPlotHistogram();

histogram->setStyle(QwtPlotHistogram::Columns);

QVector <QwtIntervalSample> t;

if (item\_statistic.first == das\_server\_monitoring\_protocol::stat\_answer\_all\_name)

{

histogram->setBrush(Qt::blue);

QString message = QString::number(item\_statistic.second);

QString format\_message = change\_format\_str(message);

ui.answer\_all\_count->setText(format\_message);

}

int pos;

QVector <QString>::iterator it = std::find(var\_statistic\_parameters.begin(), var\_statistic\_parameters.end(), column\_name);

if (it != var\_statistic\_parameters.end())

{

//нахождение позиции имени запроса

pos = var\_statistic\_parameters.end()- it;

}

//(10 - pos) - для отображения диаграммы слева направо

t.append(QwtIntervalSample(value, 10-pos, 10-pos+1));

QString message = column\_name + " : " + QString::number(value);

Приложение Б

Листинг кода для сборки приложения под Astra Linux

**rules**

#!/usr/bin/make -f

# See debhelper(7) (uncomment to enable)

# output every command that modifies files on the build system.

#export DH\_VERBOSE = 1

# see FEATURE AREAS in dpkg-buildflags(1)

#export DEB\_BUILD\_MAINT\_OPTIONS = hardening=+all

# see ENVIRONMENT in dpkg-buildflags(1)

# package maintainers to append CFLAGS

#export DEB\_CFLAGS\_MAINT\_APPEND = -Wall -pedantic

# package maintainers to append LDFLAGS

#export DEB\_LDFLAGS\_MAINT\_APPEND = -Wl,--as-needed

DEB\_HOST\_MULTIARCH := $(shell dpkg-architecture -qDEB\_HOST\_MULTIARCH)

GIT\_VERSION := $(shell dpkg-parsechangelog -SVersion)

DEBTRG=debian/das-server-monitor/opt/znak

DEBTRGBIN=$(DEBTRG)/bin

DEBTRICO=$(DEBTRG)/share/icons

DEBTRGDESKTOP=$(DEBTRG)/share/applications

DEBTRGSYSTEMD=$(DEBTRG)/etc/systemd/system

DEBTRGETC=$(DEBTRG)/etc/das-server-monitor

DEBTRGDOC=$(DEBTRG)/share/doc/das-server-monitor

%:

dh $@

override\_dh\_auto\_build:

qmake ./solution.pro -spec linux-g++

make -j4

override\_dh\_auto\_install:

# РЅР°СЃС‚СЂРѕР№РєР° РїР°РєРµС‚Р° das-server-monitor

mkdir -p $(DEBTRGBIN)

cp build/das\_server\_monitor $(DEBTRGBIN)/das-server-monitor.$(shell cat

./.version.tmp)

ln -s das-server-monitor.$(shell cat ./.version.tmp)

$(DEBTRGBIN)/das-server-monitor

mkdir -p $(DEBTRICO)

cp das\_server\_monitor/resources/icons/main.ico

$(DEBTRICO)/das-server-monitor.ico

mkdir -p $(DEBTRGETC)

mkdir -p $(DEBTRGDESKTOP)

cp debian/das-server-monitor.desktop $(DEBTRGDESKTOP)/

/bin/sed -i -e "s/\_VERSION\_/$(shell cat ./.version.tmp)/g"

$(DEBTRGDESKTOP)/das-server-monitor.desktop

override\_dh\_installdocs:

mkdir -p $(DEBTRGDOC)

cp -P debian/das-server-monitor.README.Debian $(DEBTRGDOC)/

override\_dh\_installchangelogs:

mkdir -p $(DEBTRGDOC)

cp -P debian/changelog $(DEBTRGDOC)/

**das-server-monitor.postinst**

#!/bin/sh

# postinst script for event\_bus

#

# see: dh\_installdeb(1)

set -e

# summary of how this script can be called:

# \* <postinst> `configure' <most-recently-configured-version>

# \* <old-postinst> `abort-upgrade' <new version>

# \* <conflictor's-postinst> `abort-remove' `in-favour' <package>

# <new-version>

# \* <postinst> `abort-remove'

# \* <deconfigured's-postinst> `abort-deconfigure' `in-favour'

# <failed-install-package> <version> `removing'

# <conflicting-package> <version>

# for details, see https://www.debian.org/doc/debian-policy/ or

# the debian-policy package

. /usr/share/debconf/confmodule

case "$1" in

configure)

db\_version 2.0

#db\_get znak/event-bus-replic-transmitter-gui/ip

#\_IP="${RET}"

db\_stop

chmod ugo+w /opt/znak/etc/das-server-monitor

#/bin/sed -i -e "s/gui.\*:7788/gui "$\_IP":7788/g"

/opt/znak/share/applications/event-bus-replic-transmitter-gui.desktop;;

abort-upgrade|abort-remove|abort-deconfigure)

;;

\*)

echo "postinst called with unknown argument \`$1'" >&2

exit 1

;;

esac

# dh\_installdeb will replace this with shell code automatically

# generated by other debhelper scripts.

#DEBHELPER#

exit 0

**control**

Source: das-server-monitor

Section: app/znak

Priority: optional

Maintainer: Ptakhova A.M. <ptaxovaa@inbox.ru>

Build-Depends: debhelper (>= 9)

Standards-Version: 3.9.8

Homepage: http://znak.ru

Vcs-Git: http://serverznak:3000/ptaxova/das-server-monitor.git

Vcs-Browser: http://serverznak:3000/ptaxova/das-server-monitor.git

Package: das-server-monitor

Architecture: any

Multi-arch: same

Section: app/znak

Depends: ${shlibs:Depends}, ${misc:Depends}

Recommends: libreoffice-astra, radar-location

Description: DAS server monitor

.