МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра Математического обеспечения и суперкомпьютерных технологий**

Направление подготовки: «Фундаментальная информатика и информационные технологии»

Профиль подготовки: «Инженерия программного обеспечения»

**ОТЧЕТ**

**По лабораторной работе №6**

по дисциплине «Технология программирования мобильных систем»

**Выполнила:** студентка группы 381906-3

Сорокина Екатерина Вячеславовна

**Руководитель:**

Доцент, кандидат технических наук

Борисов Николай Анатольевич

Нижний Новгород  
2022

**Содержание**

[1. Цель 3](#_Toc123141216)

[2. Постановка задачи 4](#_Toc123141217)

[3. Решение поставленной задачи 5](#_Toc123141218)

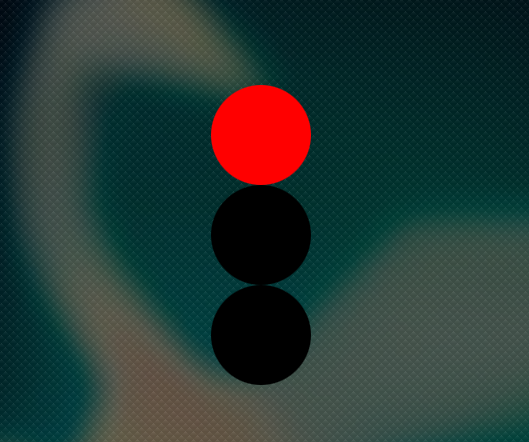
[4. Руководство пользователя 10](#_Toc123141219)

[Приложение 13](#_Toc123141220)

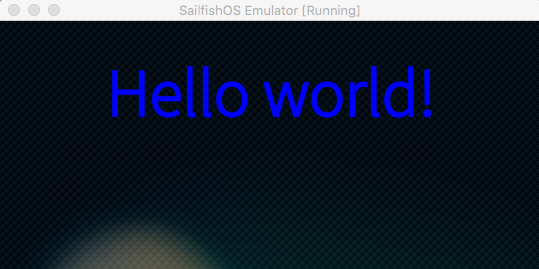
1. Цель

Цель: научиться создавать пользовательский интерфейс конфигурируемый состояниями, реализовывать анимированные переходы при смене состояний и создавать собственные QML компоненты

1. Постановка задачи
2. Создать приложение, отображающее светофор. На экране должно присутствовать 3 разноцветных сигнала, которые загораются и гаснут в том же порядке, что и сигналы светофора. Сделать автоматическую смену состояний.



1. Доработать задание 1 так, чтобы во время зеленого сигнала светофора из одного конца экрана в другой плавно двигалась иконка человечка.
2. Создать приложение, отображающее строку текста вверху экрана. При нажатии на текст он должен плавно перемещаться вниз экрана, поворачивать на 180 градусов и менять цвет. Когда нажатие прекращается, он должен так же плавно возвращаться в исходное положение.



1. Выделить сигналы светофора из задания 1 в отдельный компонент и использовать его.
2. Создать QML компонент со свойством по умолчанию, который берет значение свойства text любого объявленного внутри него объекта и создает Button с тем же текстом. Добавить возможность задавать цвет кнопки при объявлении компонента.
3. Создать приложение-секундомер. На экране должны отображаться значения часов, минут и секунд. Секундомер запускается по сигналу кнопки, при повторном нажатии секундомер останавливается. Для отображения часов, минут и секунд использовать собственные QML компоненты.
4. Решение поставленной задачи

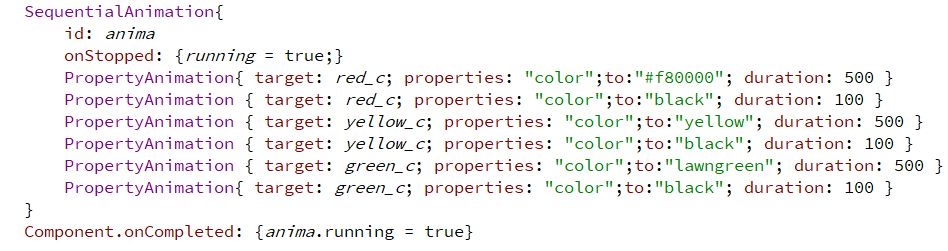
Создаём проект со стандартной заготовкой приложения.

**Задание 1.**

Круги для сигналов светофора



Анимация изменения цвета:

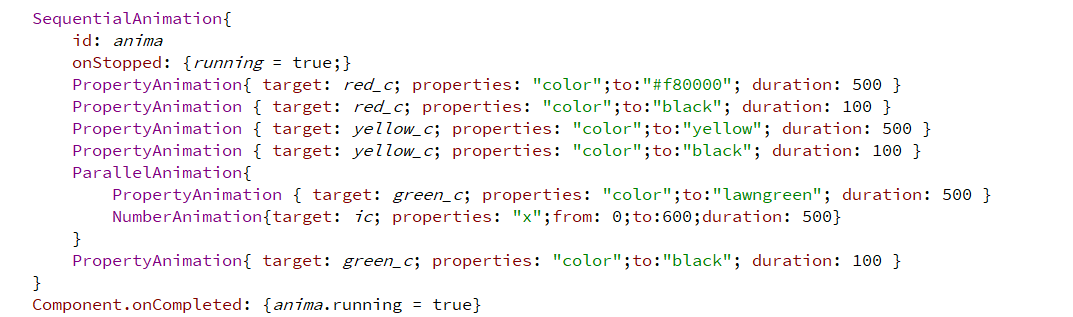


**Задание 2.**

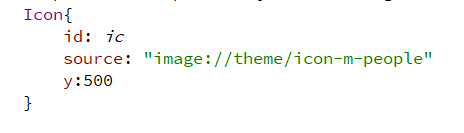
Круги для сигналов светофора



Анимация изменения цвета и перемещения иконки

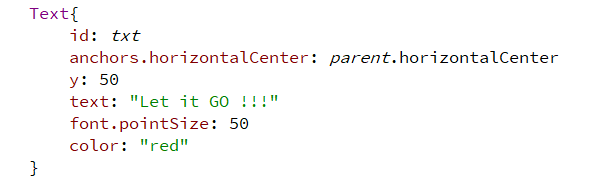


Иконка

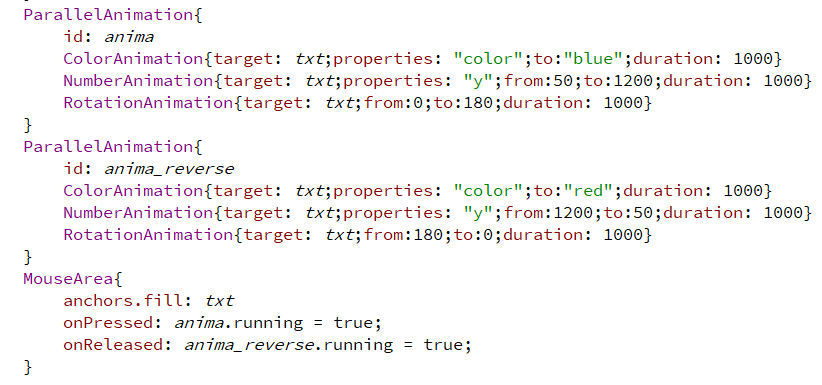


**Задание 3.**

Надпись для анимации

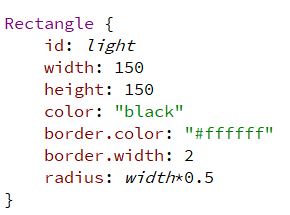


Анимация перемещения, поворота и изменения цвета

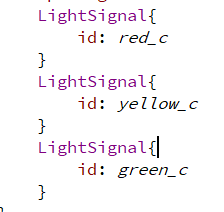


**Задание 4.**

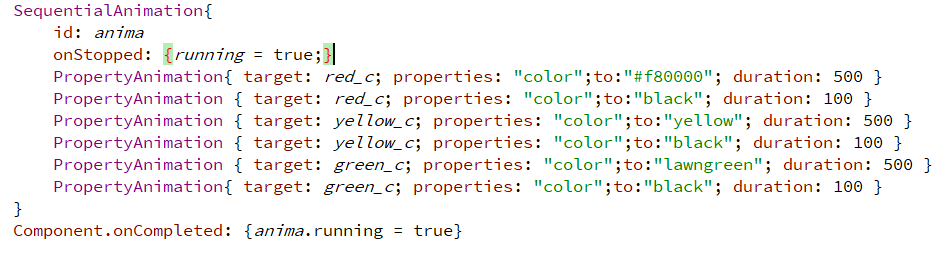
Компонента



Вызов компонент для сигналов

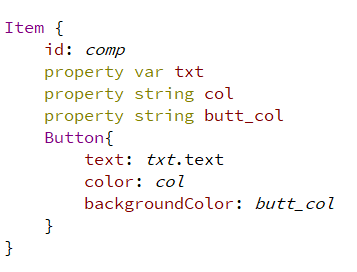


Анимация изменения цвета

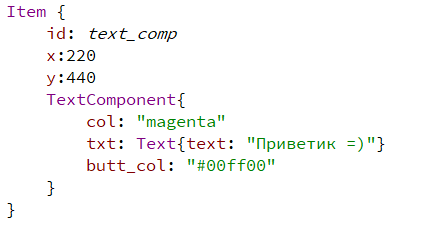


**Задание 5.**

Текстовая компонента

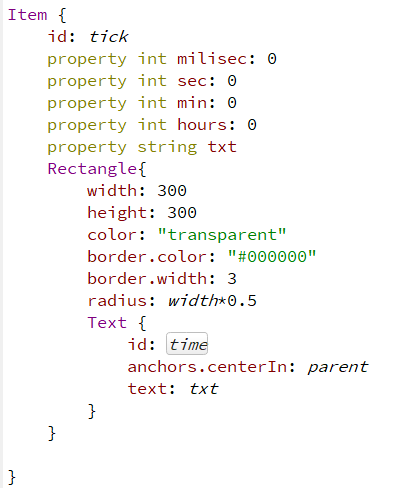


Вызов текстовой компоненты и изменение её свойств

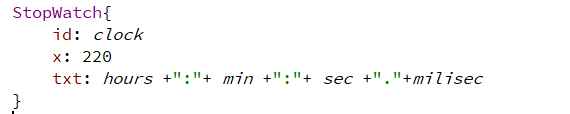


**Задание 6.**

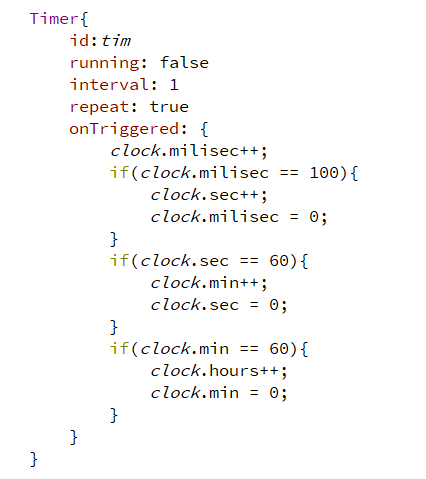
Компонента часов



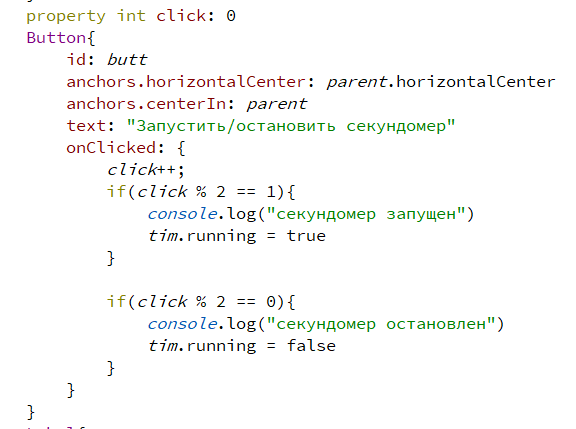
Вызов компоненты



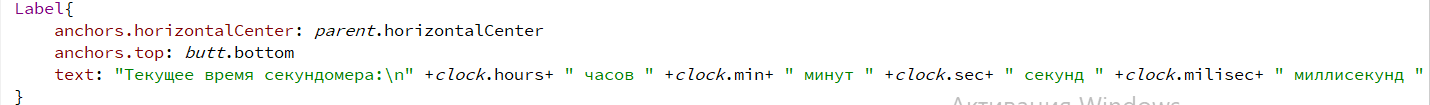
Подсчет времени



Кнопка для запуска/остановки



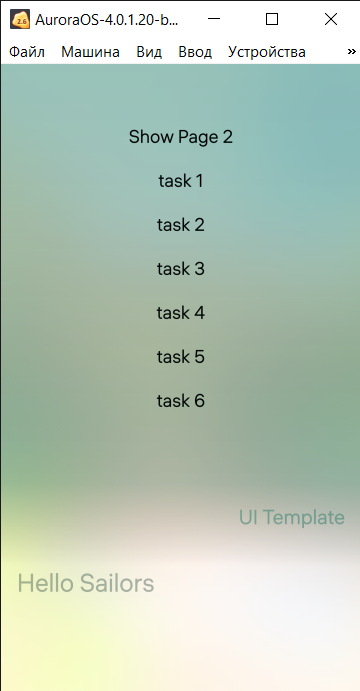
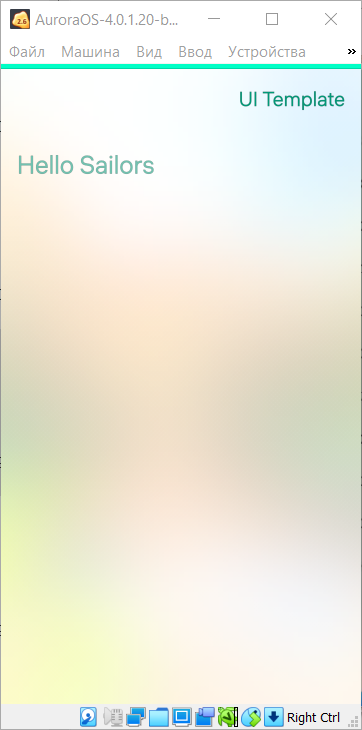
Вывод времени



Осталось только сохранить все изменения и запустить эмулятор.

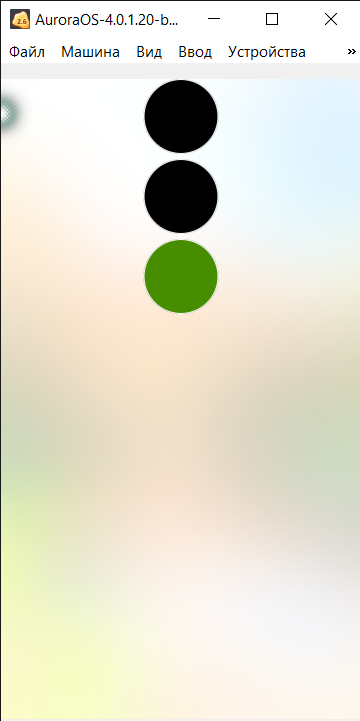
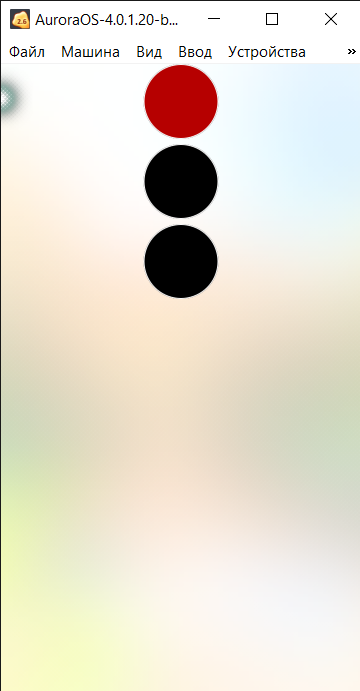
1. Руководство пользователя

При запуске приложения пользователь попадает на страницу FirstPage. Потянув вниз, открывается меню с нашими страницами.

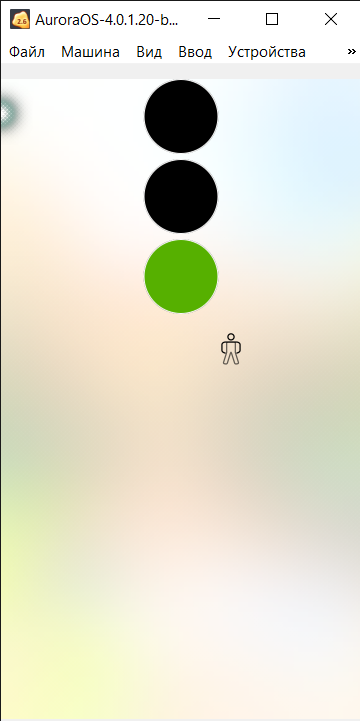


При нажатии на каждую из страниц, она загрузит нам наше задание.

Задание 1 и 4:



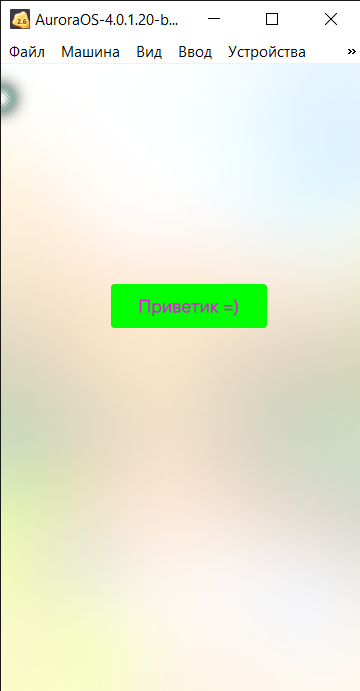
Задание 2:



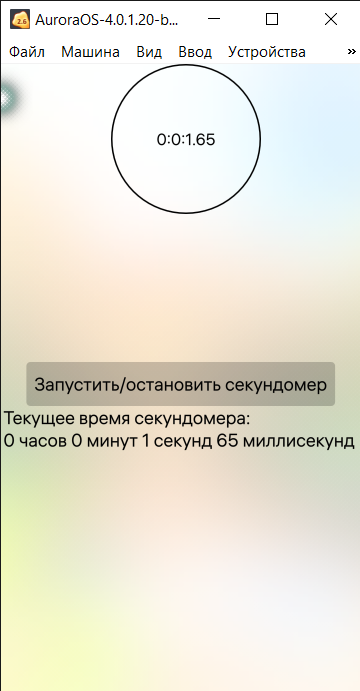
Задание 3:



Задание 5:



Задание 6:



Приложение

// FirstPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

// The effective value will be restricted by ApplicationWindow.allowedOrientations

allowedOrientations: Orientation.All

// To enable PullDownMenu, place our content in a SilicaFlickable

SilicaFlickable {

anchors.fill: *parent*

// PullDownMenu and PushUpMenu must be declared in SilicaFlickable, SilicaListView or SilicaGridView

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 2")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("SecondPage.qml"))

}

MenuItem{

text: *qsTr*("task 1")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_1.qml"))

}

MenuItem{

text: *qsTr*("task 2")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_2.qml"))

}

MenuItem{

text: *qsTr*("task 3")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_3.qml"))

}

MenuItem{

text: *qsTr*("task 4")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_4.qml"))

}

MenuItem{

text: *qsTr*("task 5")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_5.qml"))

}

MenuItem{

text: *qsTr*("task 6")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_6.qml"))

}

}

// Tell SilicaFlickable the height of its content.

contentHeight: *column*.height

// Place our content in a Column. The PageHeader is always placed at the top

// of the page, followed by our content.

Column {

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("UI Template")

}

Label {

x: Theme.horizontalPageMargin

text: *qsTr*("Hello Sailors")

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeExtraLarge

}

}

}

}

// task\_1.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page1*

Column{

id: *column*

spacing: 10

anchors.horizontalCenter: *parent*.horizontalCenter

Rectangle {

id: *red\_c*

width: 150

height: 150

color: "black"

border.color: "#ffffff"

border.width: 2

radius: *width*\*0.5

}

Rectangle {

id: *yellow\_c*

width: 150

height: 150

color: "black"

border.color: "#ffffff"

border.width: 2

radius: *width*\*0.5

}

Rectangle {

id: *green\_c*

width: 150

height: 150

color: "black"

border.color: "#ffffff"

border.width: 2

radius: *width*\*0.5

}

}

SequentialAnimation{

id: *anima*

onStopped: {*running* = true;}

PropertyAnimation{ target: *red\_c*; properties: "color";to:"#f80000"; duration: 500 }

PropertyAnimation { target: *red\_c*; properties: "color";to:"black"; duration: 100 }

PropertyAnimation { target: *yellow\_c*; properties: "color";to:"yellow"; duration: 500 }

PropertyAnimation { target: *yellow\_c*; properties: "color";to:"black"; duration: 100 }

PropertyAnimation { target: *green\_c*; properties: "color";to:"lawngreen"; duration: 500 }

PropertyAnimation{ target: *green\_c*; properties: "color";to:"black"; duration: 100 }

}

Component.onCompleted: {*anima*.running = true}

}

//task\_2.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page2*

Column{

id: *column*

spacing: 10

anchors.horizontalCenter: *parent*.horizontalCenter

Rectangle {

id: *red\_c*

width: 150

height: 150

color: "black"

border.color: "#ffffff"

border.width: 2

radius: *width*\*0.5

}

Rectangle {

id: *yellow\_c*

width: 150

height: 150

color: "black"

border.color: "#ffffff"

border.width: 2

radius: *width*\*0.5

}

Rectangle {

id: *green\_c*

width: 150

height: 150

color: "black"

border.color: "#ffffff"

border.width: 2

radius: *width*\*0.5

}

}

SequentialAnimation{

id: *anima*

onStopped: {*running* = true;}

PropertyAnimation{ target: *red\_c*; properties: "color";to:"#f80000"; duration: 500 }

PropertyAnimation { target: *red\_c*; properties: "color";to:"black"; duration: 100 }

PropertyAnimation { target: *yellow\_c*; properties: "color";to:"yellow"; duration: 500 }

PropertyAnimation { target: *yellow\_c*; properties: "color";to:"black"; duration: 100 }

ParallelAnimation{

PropertyAnimation { target: *green\_c*; properties: "color";to:"lawngreen"; duration: 500 }

NumberAnimation{target: *ic*; properties: "x";from: 0;to:600;duration: 500}

}

PropertyAnimation{ target: *green\_c*; properties: "color";to:"black"; duration: 100 }

}

Component.onCompleted: {*anima*.running = true}

Icon{

id: *ic*

source: "image://theme/icon-m-people"

y:500

}

}

//task\_3.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page3*

Text{

id: *txt*

anchors.horizontalCenter: *parent*.horizontalCenter

y: 50

text: "Let it GO !!!"

font.pointSize: 50

color: "red"

}

ParallelAnimation{

id: *anima*

ColorAnimation{target: *txt*;properties: "color";to:"blue";duration: 1000}

NumberAnimation{target: *txt*;properties: "y";from:50;to:1200;duration: 1000}

RotationAnimation{target: *txt*;from:0;to:180;duration: 1000}

}

ParallelAnimation{

id: *anima\_reverse*

ColorAnimation{target: *txt*;properties: "color";to:"red";duration: 1000}

NumberAnimation{target: *txt*;properties: "y";from:1200;to:50;duration: 1000}

RotationAnimation{target: *txt*;from:180;to:0;duration: 1000}

}

MouseArea{

anchors.fill: *txt*

onPressed: *anima*.running = true;

onReleased: *anima\_reverse*.running = true;

}

}

//task\_4.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page4*

Column{

anchors.horizontalCenter: *parent*.horizontalCenter

spacing: 10

LightSignal{

id: *red\_c*

}

LightSignal{

id: *yellow\_c*

}

LightSignal{

id: *green\_c*

}

}

SequentialAnimation{

id: *anima*

onStopped: {*running* = true;}

PropertyAnimation{ target: *red\_c*; properties: "color";to:"#f80000"; duration: 500 }

PropertyAnimation { target: *red\_c*; properties: "color";to:"black"; duration: 100 }

PropertyAnimation { target: *yellow\_c*; properties: "color";to:"yellow"; duration: 500 }

PropertyAnimation { target: *yellow\_c*; properties: "color";to:"black"; duration: 100 }

PropertyAnimation { target: *green\_c*; properties: "color";to:"lawngreen"; duration: 500 }

PropertyAnimation{ target: *green\_c*; properties: "color";to:"black"; duration: 100 }

}

Component.onCompleted: {*anima*.running = true}

}

//LightSignal.qml

import QtQuick 2.0

Rectangle {

id: *light*

width: 150

height: 150

color: "black"

border.color: "#ffffff"

border.width: 2

radius: *width*\*0.5

}

//task\_5.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page5*

Item {

id: *text\_comp*

x:220

y:440

TextComponent{

col: "magenta"

txt: Text{text: "Приветик =)"}

butt\_col: "#00ff00"

}

}

}

//TextComponent.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Item {

id: *comp*

property var txt

property string col

property string butt\_col

Button{

text: *txt*.text

color: *col*

backgroundColor: *butt\_col*

}

}

//task\_6.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page6*

property int click: 0

Timer{

id:*tim*

running: false

interval: 1

repeat: true

onTriggered: {

*clock*.milisec++;

if(*clock*.milisec == 100){

*clock*.sec++;

*clock*.milisec = 0;

}

if(*clock*.sec == 60){

*clock*.min++;

*clock*.sec = 0;

}

if(*clock*.min == 60){

*clock*.hours++;

*clock*.min = 0;

}

}

}

StopWatch{

id: *clock*

x: 220

txt: *hours* +":"+ *min* +":"+ *sec* +"."+*milisec*

}

Button{

id: *butt*

anchors.horizontalCenter: *parent*.horizontalCenter

anchors.centerIn: *parent*

text: "Запустить/остановить секундомер"

onClicked: {

*click*++;

if(*click* % 2 == 1){

*console*.log("секундомер запущен")

*tim*.running = true

}

if(*click* % 2 == 0){

*console*.log("секундомер остановлен")

*tim*.running = false

}

}

}

Label{

anchors.horizontalCenter: *parent*.horizontalCenter

anchors.top: *butt*.bottom

text: "Текущее время секундомера:\n" +*clock*.hours+ " часов " +*clock*.min+ " минут " +*clock*.sec+ " секунд " +*clock*.milisec+ " миллисекунд "

}

}

//StopWatch.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Item {

id: *tick*

property int milisec: 0

property int sec: 0

property int min: 0

property int hours: 0

property string txt

Rectangle{

width: 300

height: 300

color: "transparent"

border.color: "#000000"

border.width: 3

radius: *width*\*0.5

Text {

id: *time*

anchors.centerIn: *parent*

text: *txt*

}

}

}