

Магистерская диссертация на тему

Исследование программного интерфейса радиочастотной части мобильного мультирадиотерминала

Руководитель: доцент, к.т.н. Иванов В. Н.

Студент: гр. 1740М, Шатунов Л.В.

Цель и задачи исследования

Цель: Исследование, имплементация и оптимизация программного интерфейса радиочастотной части реконфигурируемых мобильных устройств.

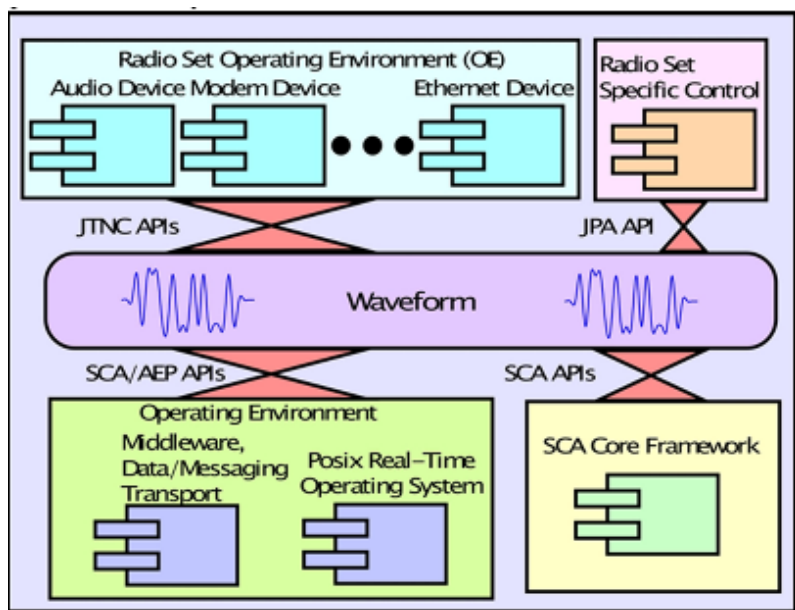
Задачи:

1. Изучить возможность реализации программного интерфейса радиочастотной части мобильных терминалов
2. Имплементировать и оптимизировать информационную модель интерфейса RRFI

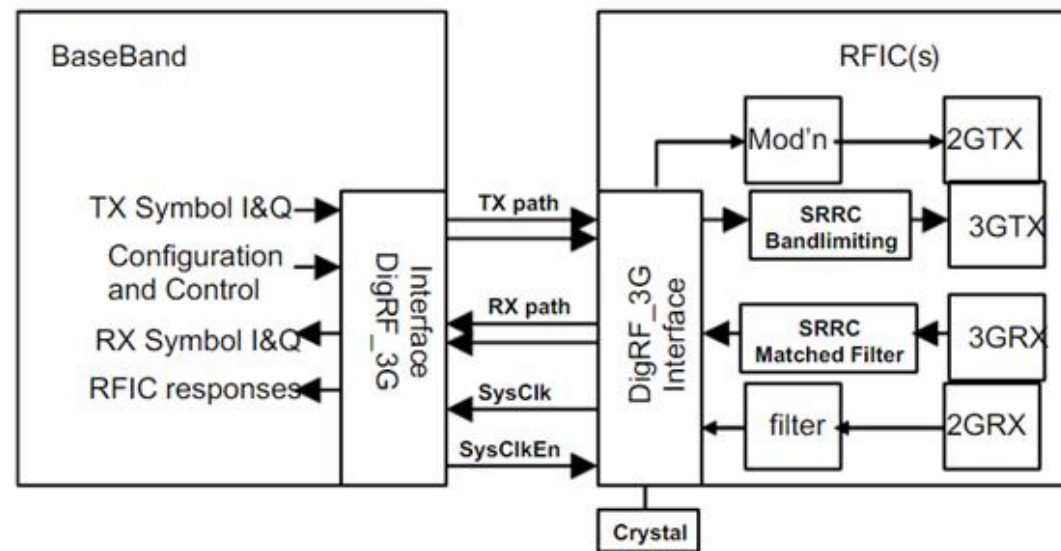
Результат обзора существующих технологий



SDR – Software Defined Radio



SCA – Software Communication Architecture



DigRF

Результат обзора существующих технологий

SDR – реализация программирования для «стандартного» компьютера (машина фон Неймана)

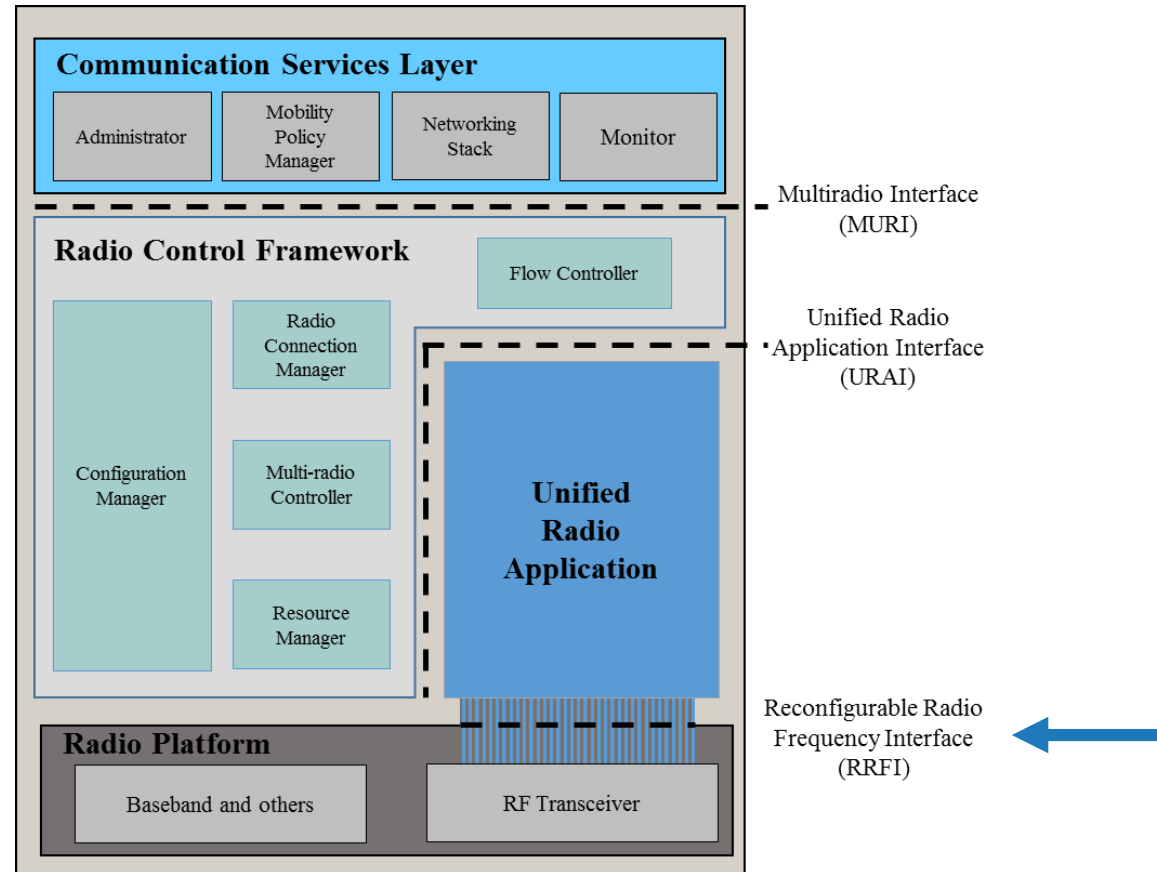
- Налагает определенные ограничения на аппаратную архитектуру
- Проблемы с переносимостью кода
- Взаимодействие функциональных компонент происходит через программную шину



RRS – реализация программирования для виртуальной машины, независимой от аппаратной платформы

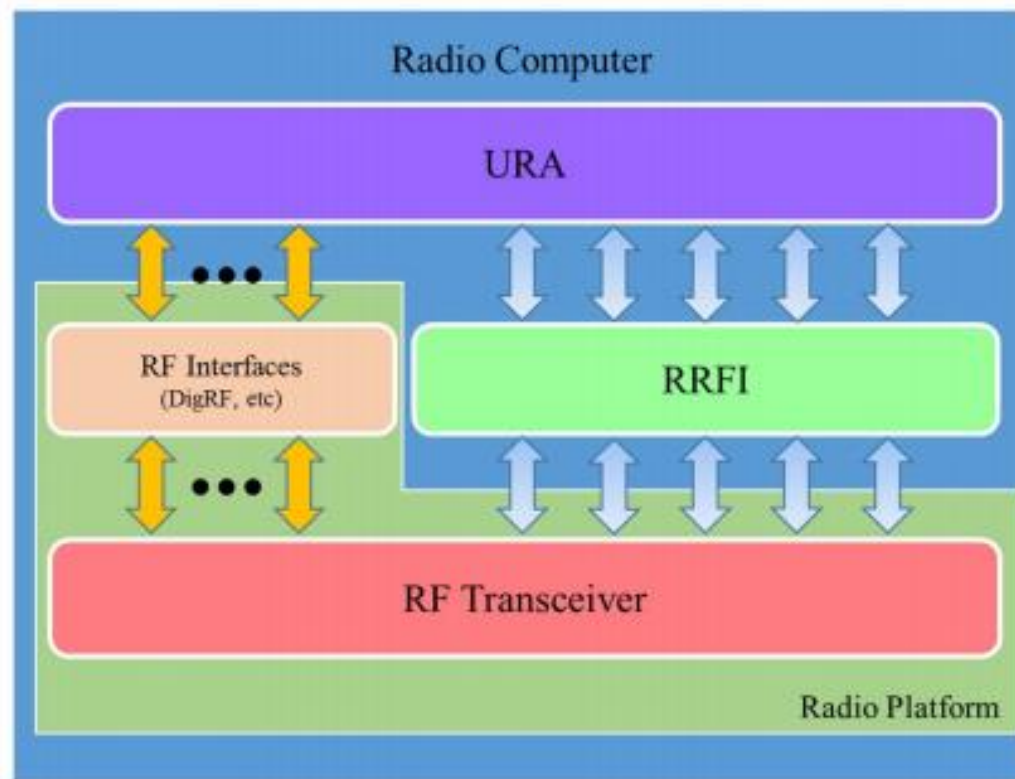
- Снимает ограничения аппаратной архитектуры
- Обеспечивает переносимость кода
- Взаимодействие компонент описывается на уровне программного кода

Reconfigurable Radio System



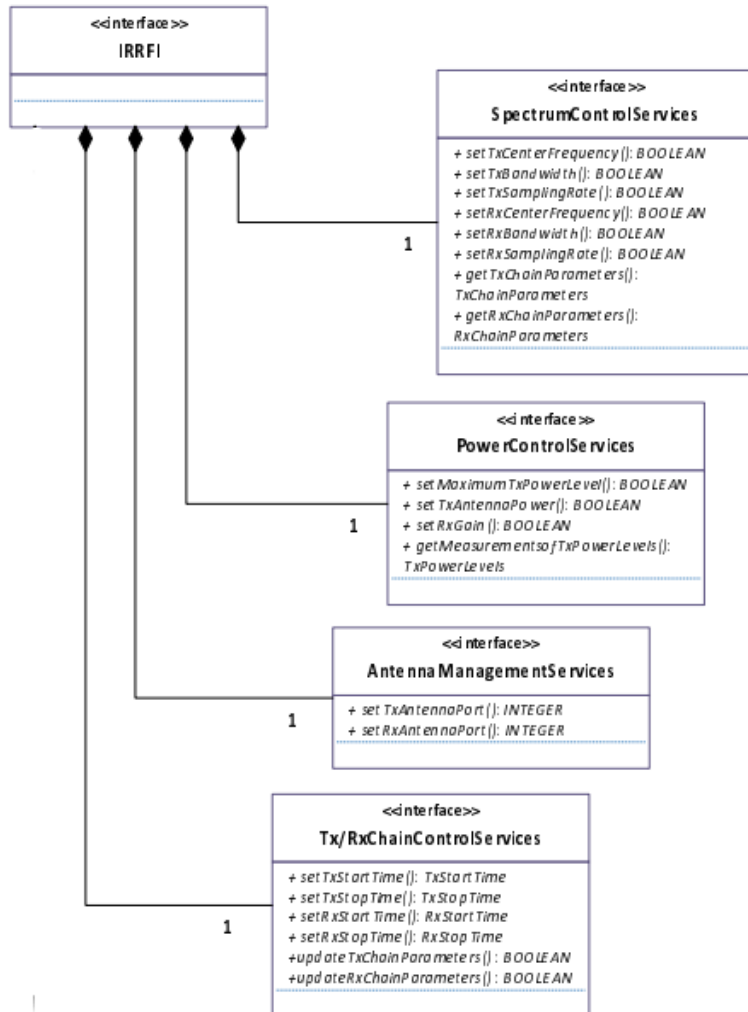
RMD – Reconfigurable Mobile Device

Reconfigurable Radio Frequency Interface



RRFI – Reconfigurable Radio Frequency Interface

Reconfigurable Radio Frequency Interface



Для конфигурации радиотрансивера используется 4 сервиса предоставляющих различные функции:

1. Spectrum Control Services – управление радиоспектром
2. Power Control Services – управление мощностью
3. Antenna Management Services – управление антеннами
4. Tx/Rx Chain Control Services – управление потоками приема/передачи

Программная реализация интерфейса



USRP B210

```
bool ETSI_RRS_PowerControlServices::set_txPowerLevel(double actualTxGain, int channel) {
    if (this->usrpDevice->min_tx_gain > actualTxGain) {
        usrpDevice->usrp->set_tx_gain(actualTxGain, size_t(channel));
        cout << "tx_powerlevel is less than min value, tx_powerlevel is min value" << endl;
        return false;
    } else {
        if (this->usrpDevice->max_tx_gain < actualTxGain) {
            usrpDevice->usrp->set_tx_gain(actualTxGain, size_t(channel));
            cout << "tx_powerlevel is larger than max value, tx_powerlevel is max value" << endl;
            return false;
        } else {
            usrpDevice->usrp->set_tx_gain(actualTxGain, size_t(channel));
            cout << "tx_powerlevel changed successfully" << endl;
            return true;
        }
    }
}
```

```
String ETSI_RRS_AntennaManagementServices::get_txAntennaPort(int channel) {
    if (this->txAntennaPort == usrpDevice->usrp->get_tx_antenna(size_t(channel))) {
        return txAntennaPort;
    } else {
        cout << "error" << endl;
    }
}
```

```
bool ETSI_RRS_TxRxChainControlServices::tx_from_buff(std::vector<short> buff, size_t samps_per_buff) {
    usrpDevice->tx_samps_per_buff = samps_per_buff;
    cout << "Sending bytes..." << endl;
    if (usrpDevice->tx_stream->send(&buff.front(), samps_per_buff, usrpDevice->tx_md) != samps_per_buff) {
        return false;
    }
    else {
        cout << "\e[1m" << "Bytes was sent successfully" << "\e[0m" << endl;
        return true;
    }
}
```

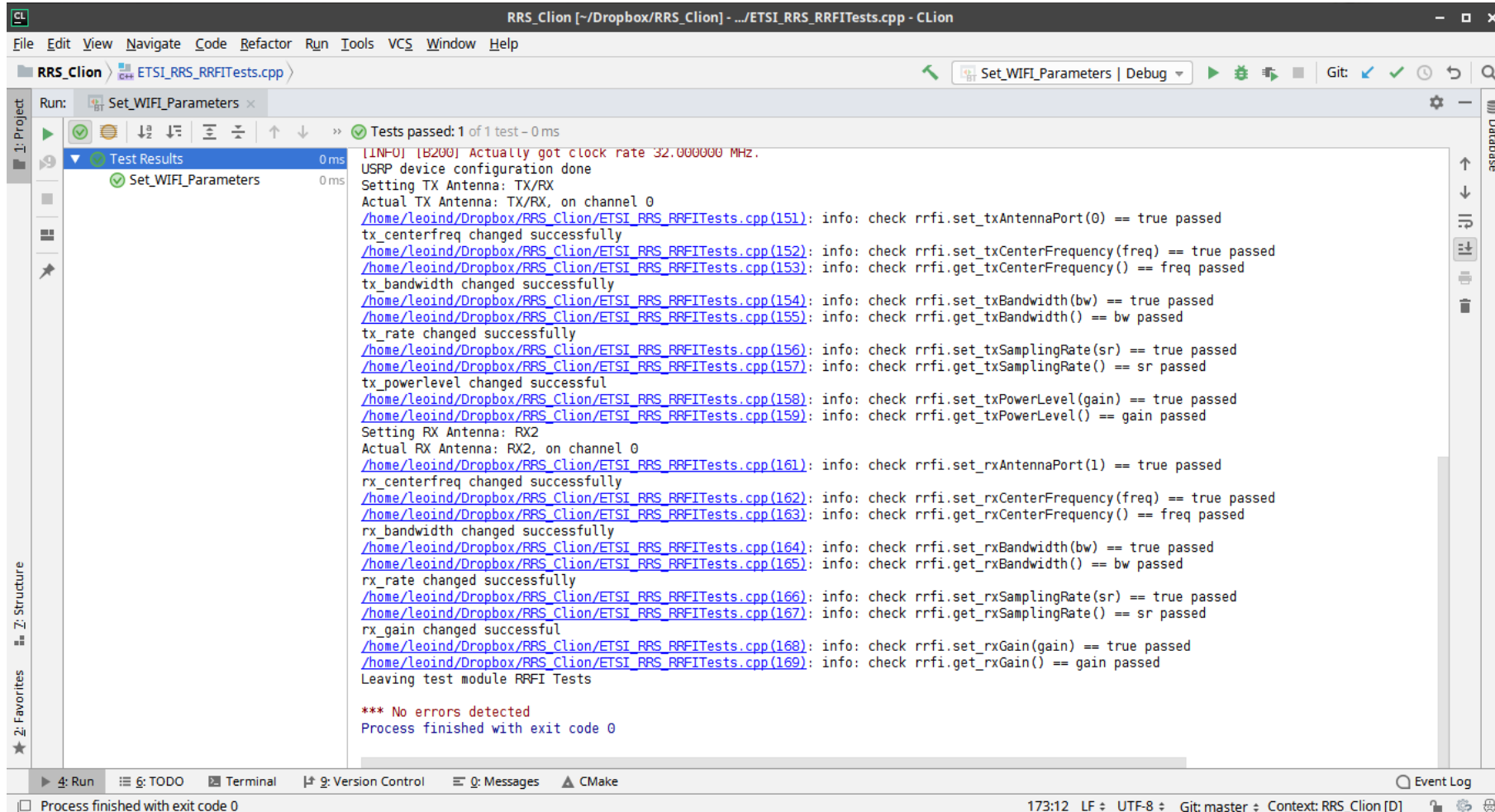

Проверка работоспособности интерфейса

Интерфейс не имеет
исполняемого файла



Работоспособность
проверяется
наборами тестов для
всех функций и
параметров
различных
протоколов связи

Проверка работоспособности интерфейса



```

RRS_Client [~/Dropbox/RRS_Client] - .../ETSI_RRS_RRFITests.cpp - CLion
File Edit View Navigate Code Refactor Run Tools VCS Window Help
RRS_Client > ETSI_RRS_RRFITests.cpp
Run: Set_WIFI_Parameters x
Tests passed: 1 of 1 test - 0 ms
Test Results
Set_WIFI_Parameters 0 ms
[INFO] [B200] Actually got clock rate 32.000000 MHz.
USRP device configuration done
Setting TX Antenna: TX/RX
Actual TX Antenna: TX/RX, on channel 0
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(151): info: check rrfi.set_txAntennaPort(0) == true passed
tx_centerfreq changed successfully
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(152): info: check rrfi.set_txCenterFrequency(freq) == true passed
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(153): info: check rrfi.get_txCenterFrequency() == freq passed
tx_bandwidth changed successfully
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(154): info: check rrfi.set_txBandwidth(bw) == true passed
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(155): info: check rrfi.get_txBandwidth() == bw passed
tx_rate changed successfully
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(156): info: check rrfi.set_txSamplingRate(sr) == true passed
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(157): info: check rrfi.get_txSamplingRate() == sr passed
tx_powerlevel changed successful
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(158): info: check rrfi.set_txPowerLevel(gain) == true passed
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(159): info: check rrfi.get_txPowerLevel() == gain passed
Setting RX Antenna: RX2
Actual RX Antenna: RX2, on channel 0
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(161): info: check rrfi.set_rxAntennaPort(1) == true passed
rx_centerfreq changed successfully
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(162): info: check rrfi.set_rxCenterFrequency(freq) == true passed
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(163): info: check rrfi.get_rxCenterFrequency() == freq passed
rx_bandwidth changed successfully
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(164): info: check rrfi.set_rxBandwidth(bw) == true passed
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(165): info: check rrfi.get_rxBandwidth() == bw passed
rx_rate changed successfully
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(166): info: check rrfi.set_rxSamplingRate(sr) == true passed
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(167): info: check rrfi.get_rxSamplingRate() == sr passed
rx_gain changed successful
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(168): info: check rrfi.set_rxGain(gain) == true passed
/home/leoind/Dropbox/RRS_Client/ETSI_RRS_RRFITests.cpp(169): info: check rrfi.get_rxGain() == gain passed
Leaving test module RRFI Tests

*** No errors detected
Process finished with exit code 0

```

Заключение

Были выполнены задачи:

- Исследование технологии реконфигурируемой системы
- Программная реализация интерфейса RRFI и его оптимизация
- Проверка возможности встраивания и работоспособности интерфейса в реконфигурируемую радиосистему

Перспективное развитие:

- Функционал интерфейса имеет возможность дополнения
- Интерфейс планируется применять для дальнейшей разработки общей реконфигурируемой радиосистемы

Спасибо за внимание!