MobileSDR:
A Mobile Programmable Platform for Wireless Field Tests and Diagnostics

AIMS-2025

Zesen Zhang, Rohith Reddy Vennam, Maiyun Zhang, Yunxiang Chi,
Dinesh Bharadia, Aaron Schulman

Infrastructure Components in Access Network

Last Miles Components

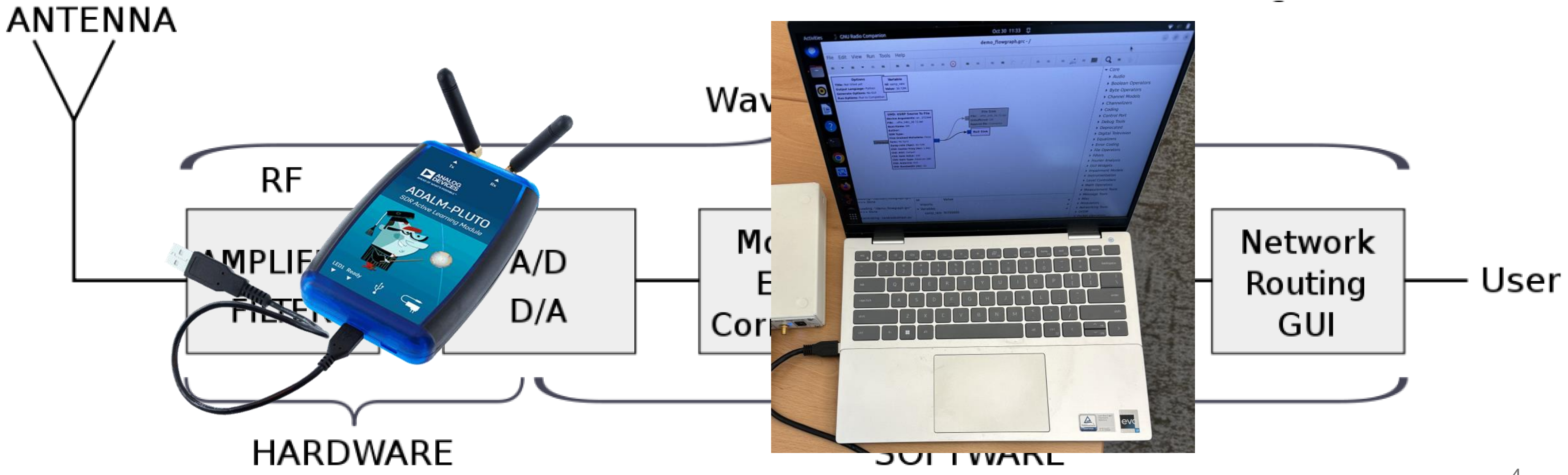
What is wireless measurement

Using sensors to collect analog signal and process with program to capture key information in the signal

- **Sensors**: Collect raw data, such as sound, FM, LoRa, cellular signal
- **Analog-to-digital converter:** Converts the raw data to a digital format if needed.
- **Key feature capture:** FFT, demodulation.
- **Further analysis:** Packet decoding, decryption...

What is SDR

- **Software-defined radio** (**SDR**) is a radio communication system who uses software to implement components which is conventionally implemented by analog hardware.

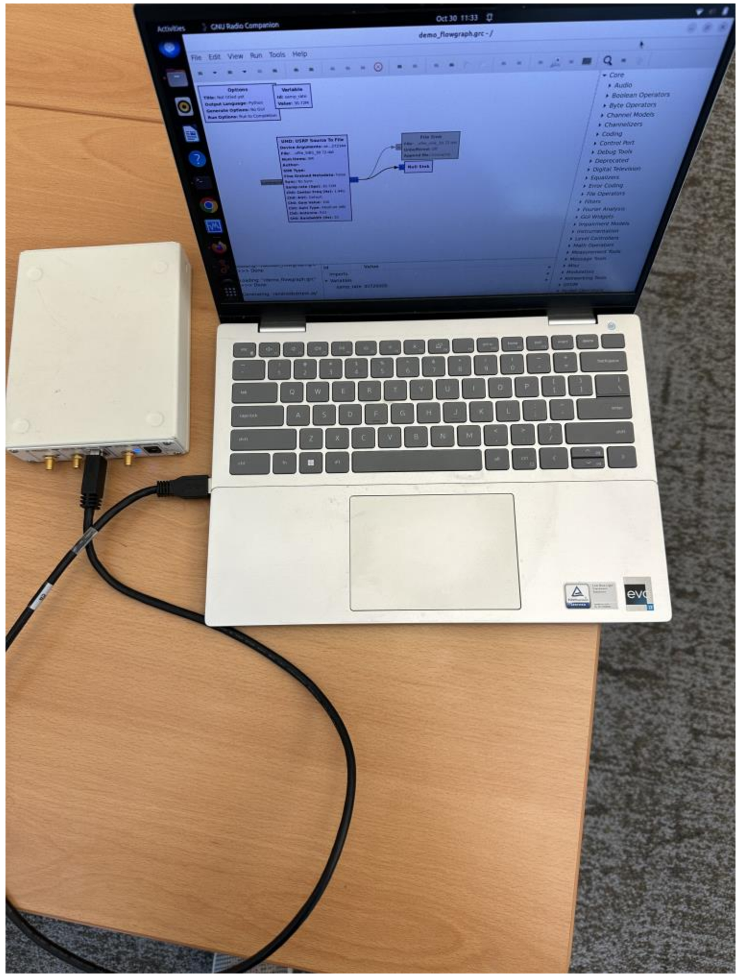


Collecting Wireless Data in the field is not easy...

Design program in lab

Bring a laptop with SDR and Drive/Walk around to collect data

What if...



Connect SDR to laptop

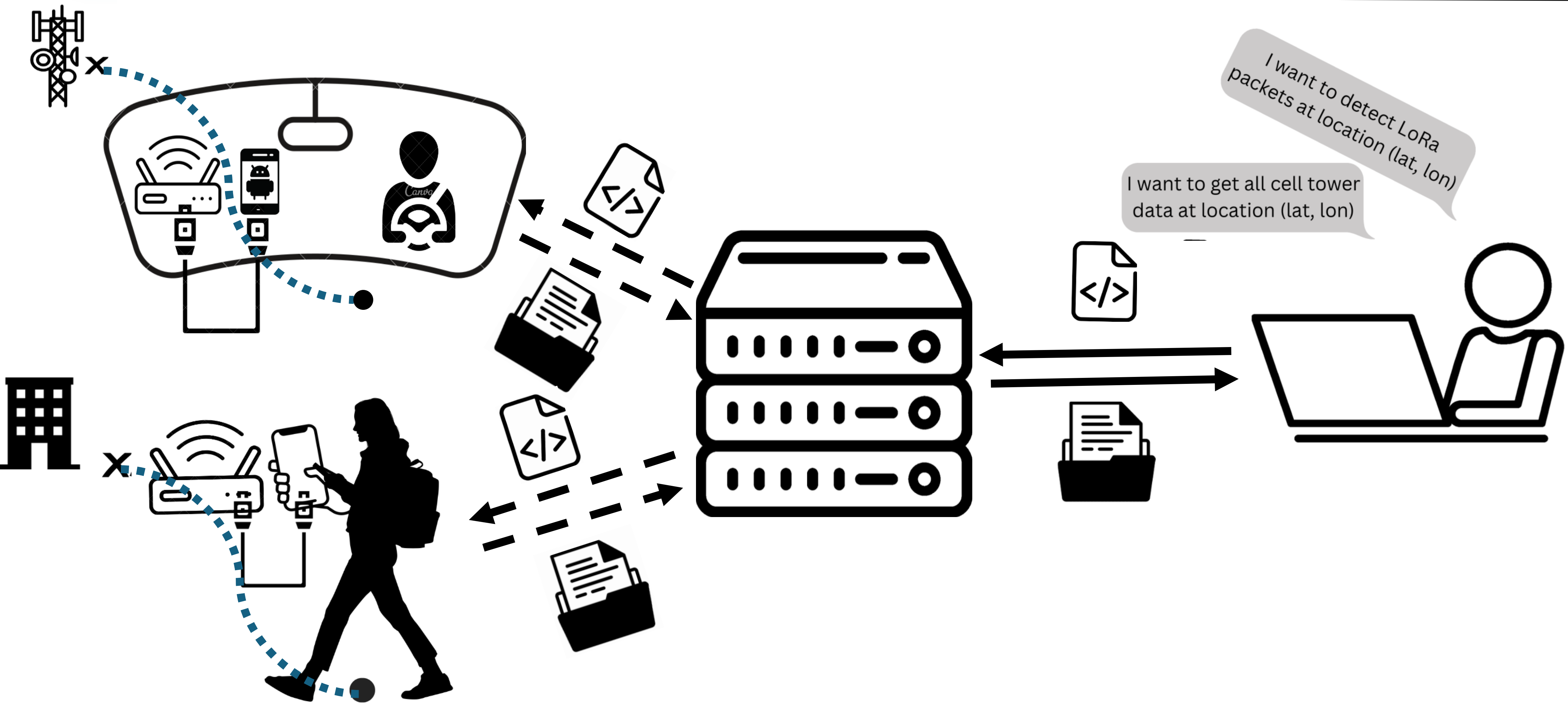
Connect SDR to Phone

So that...

Design program in lab

Bring a phone with SDR and Drive/Walk around to collect data

What if…


I want to detect LoRa packets at location (lat, lon)
I want to get all cell tower data at location (lat, lon)
Canva

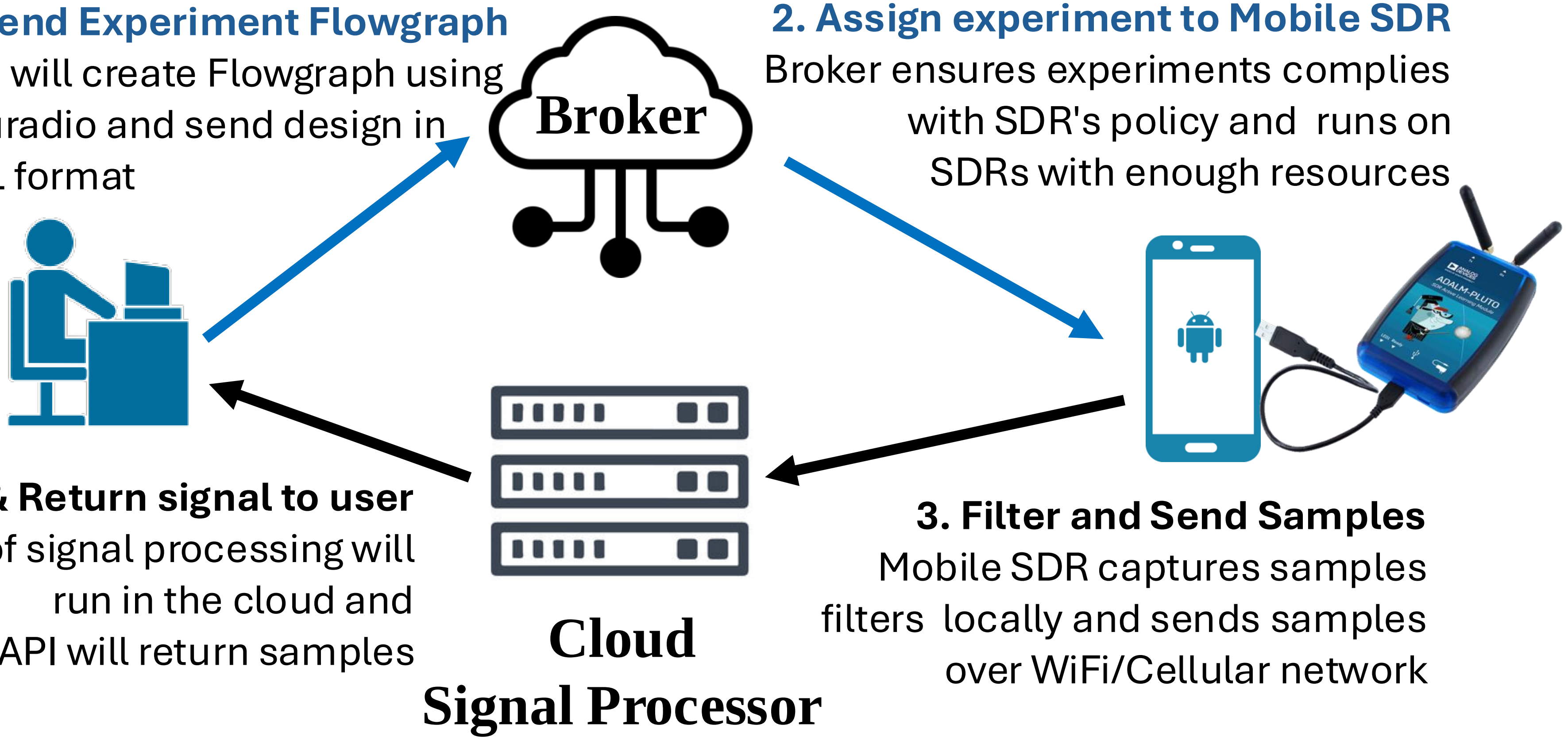
MobileSDR Goal

Separate wireless data collection from program design

Developing a crowdsourced platform

- **Researchers design programs in the lab and publish them on the platform**
- **Users download tasks to their phones and gather data via SDR**
- **Users then upload the collected data to the platform, making it available for everyone**

MobileSDR architecture



1. Send Experiment Flowgraph
This will create Flowgraph using Gnuradio and send design in XML format

2. Assign experiment to Mobile SDR
Broker ensures experiments complies with SDR's policy and runs on SDRs with enough resources

3. Filter and Send Samples
Mobile SDR captures samples filters locally and sends samples over WiFi/Cellular network

4. Store & Return signal to user
The rest of signal processing will run in the cloud and an API will return samples

To achieve this Goal we need to..

Easy for endpoints to collect data and join the ecosystem

Ensure program security running on endpoint's device

Ensure data integrity

Save battery on the phone

Go through limited computation resource on the phone

To achieve this Goal we need to..

Easy for endpoints to collect data and join the ecosystem

Ensure program security running on endpoint's device

Ensure data integrity

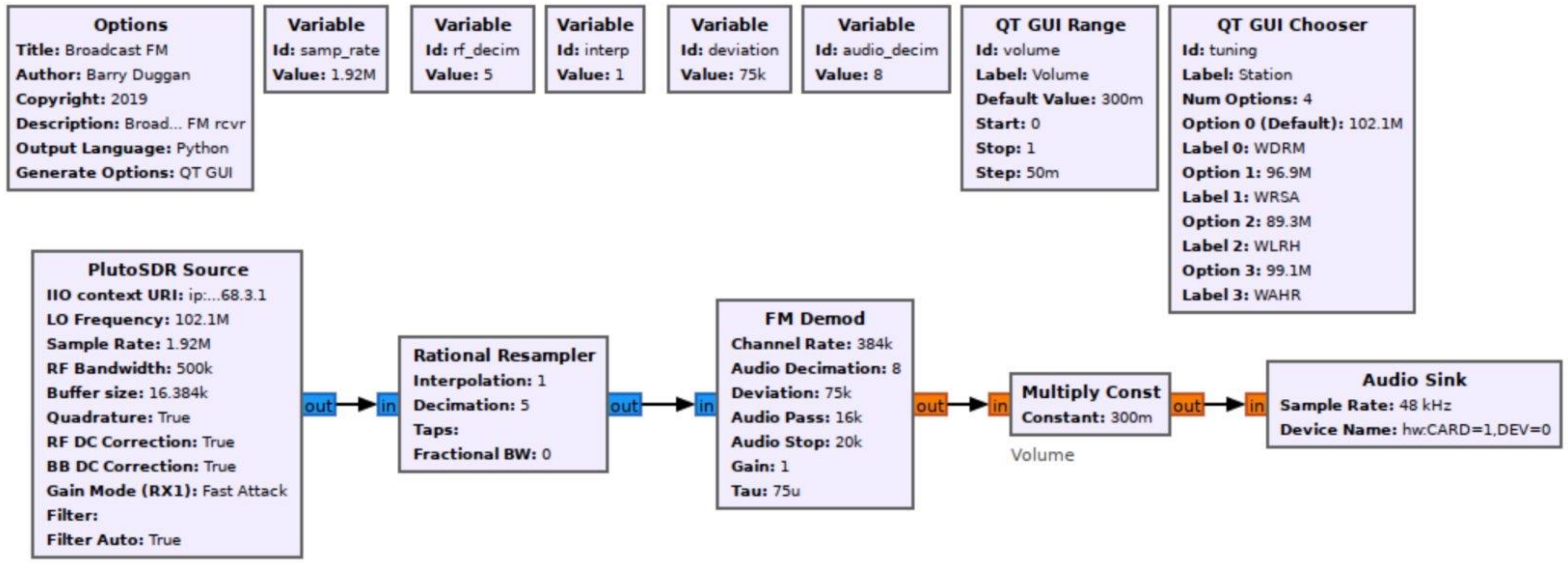
Save battery on the phone

Go through limited computation resource on the phone

Ensure program security running on endpoint

Use domain specific language to restrict what researchers can run on the platform while providing enough flexibility

A **domain-specific language** (**DSL**) is a computer language specialized to a particular application domain.



Options
Title: Broadcast FM
Author: Barry Duggan
Copyright: 2019
Description: Broad... FM rcvr
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 1.92M

Variable
Id: rf_decim
Value: 5

Variable
Id: interp
Value: 1

Variable
Id: deviation
Value: 75k

Variable
Id: audio_decim
Value: 8

QT GUI Range
Id: volume
Label: Volume
Default Value: 300m
Start: 0
Stop: 1
Step: 50m

QT GUI Chooser
Id: tuning
Label: Station
Num Options: 4
Option 0 (Default): 102.1M
Label 0: WDRM
Option 1: 96.9M
Label 1: WRSA
Option 2: 89.3M
Label 2: WLRH
Option 3: 99.1M
Label 3: WAHR

PlutoSDR Source
IIO context URI: ip:...68.3.1
LO Frequency: 102.1M
Sample Rate: 1.92M
RF Bandwidth: 500k
Buffer size: 16.384k
Quadrature: True
RF DC Correction: True
BB DC Correction: True
Gain Mode (RX1): Fast Attack
Filter:
Filter Auto: True

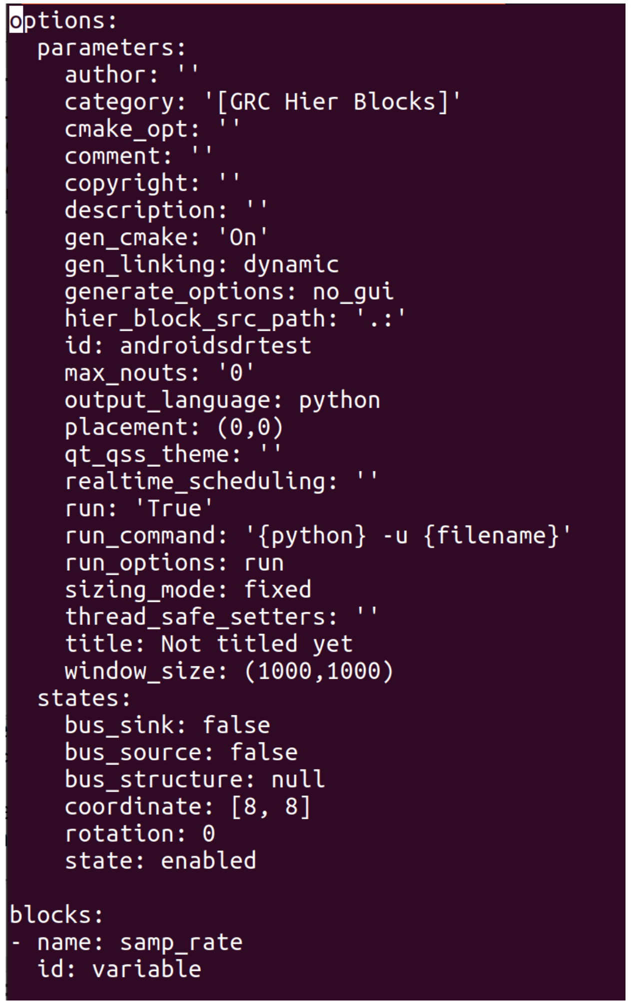
Rational Resampler
Interpolation: 1
Decimation: 5
Taps:
Fractional BW: 0

FM Demod
Channel Rate: 384k
Audio Decimation: 8
Deviation: 75k
Audio Pass: 16k
Audio Stop: 20k
Gain: 1
Tau: 75u

Multiply Const
Constant: 300m
Volume

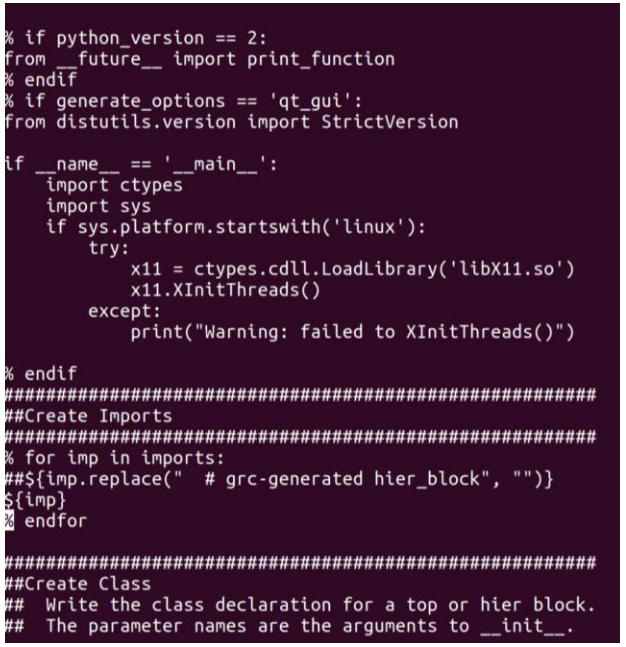
Audio Sink
Sample Rate: 48 kHz
Device Name: hw:CARD=1,DEV=0

Only allow sending XML file instead of binary

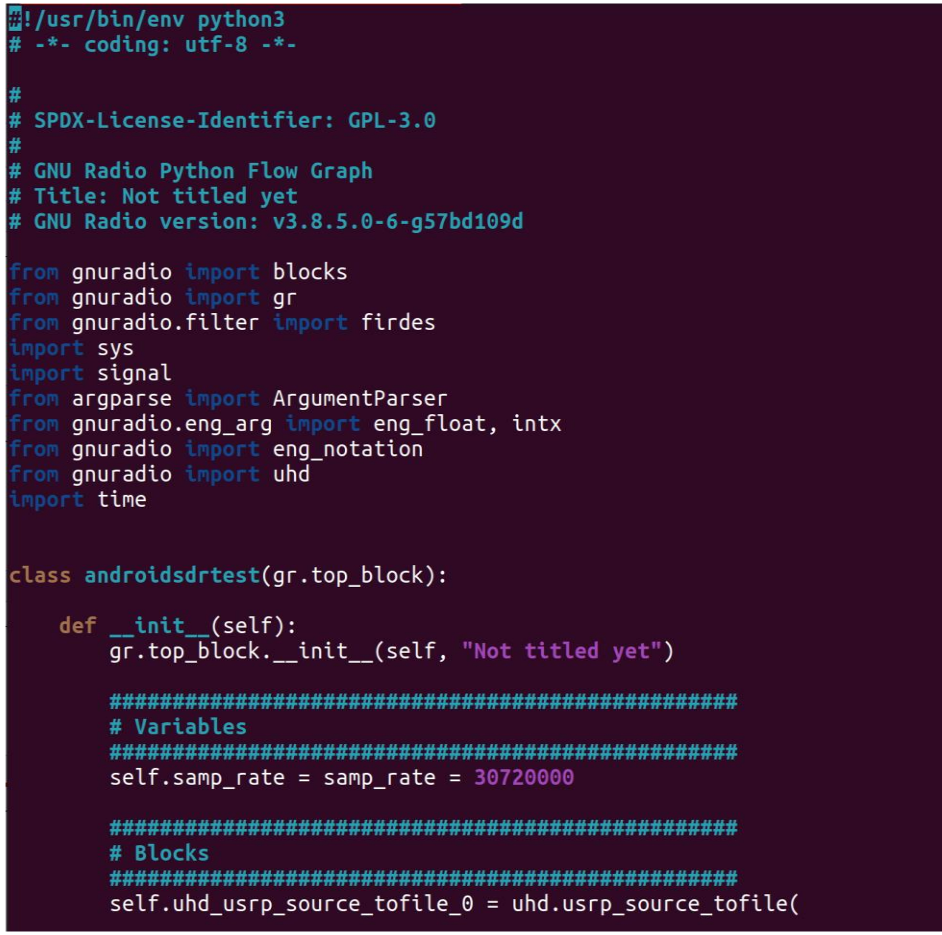


```
options:
  parameters:
    author: ''
    category: '[GRC Hier Blocks]'
    cmake_opt: ''
    comment: ''
    copyright: ''
    description: ''
    gen_cmake: 'On'
    gen_linking: dynamic
    generate_options: no_gui
    hier_block_src_path: '.:'
    id: androidsdrtest
    max_nouts: '0'
    output_language: python
    placement: (0,0)
    qt_qss_theme: ''
    realtime_scheduling: ''
    run: 'True'
    run_command: '{python} -u {filename}'
    run_options: run
    sizing_mode: fixed
    thread_safe_setters: ''
    title: Not titled yet
    window_size: (1000,1000)
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [8, 8]
    rotation: 0
    state: enabled

blocks:
- name: samp_rate
  id: variable
```

Mako template

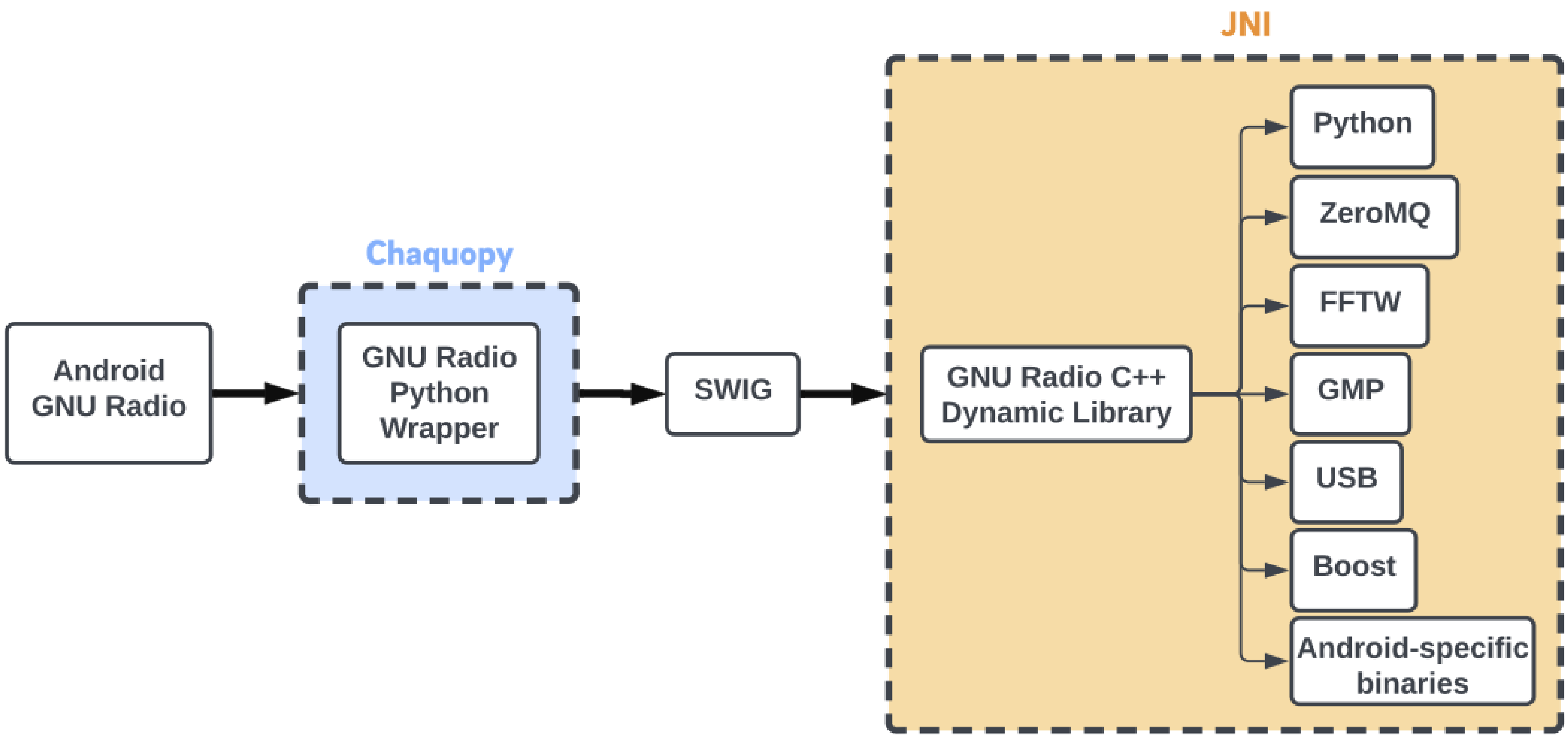


```
% if python_version == 2:
from __future__ import print_function
% endif
% if generate_options == 'qt_gui':
from distutils.version import StrictVersion

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print("Warning: failed to XInitThreads()")

% endif
########################################################
##Create Imports
########################################################
% for imp in imports:
##${imp.replace("  # grc-generated hier_block", "")}
${imp}
% endfor

########################################################
##Create Class
##  Write the class declaration for a top or hier block.
##  The parameter names are the arguments to __init__.
```

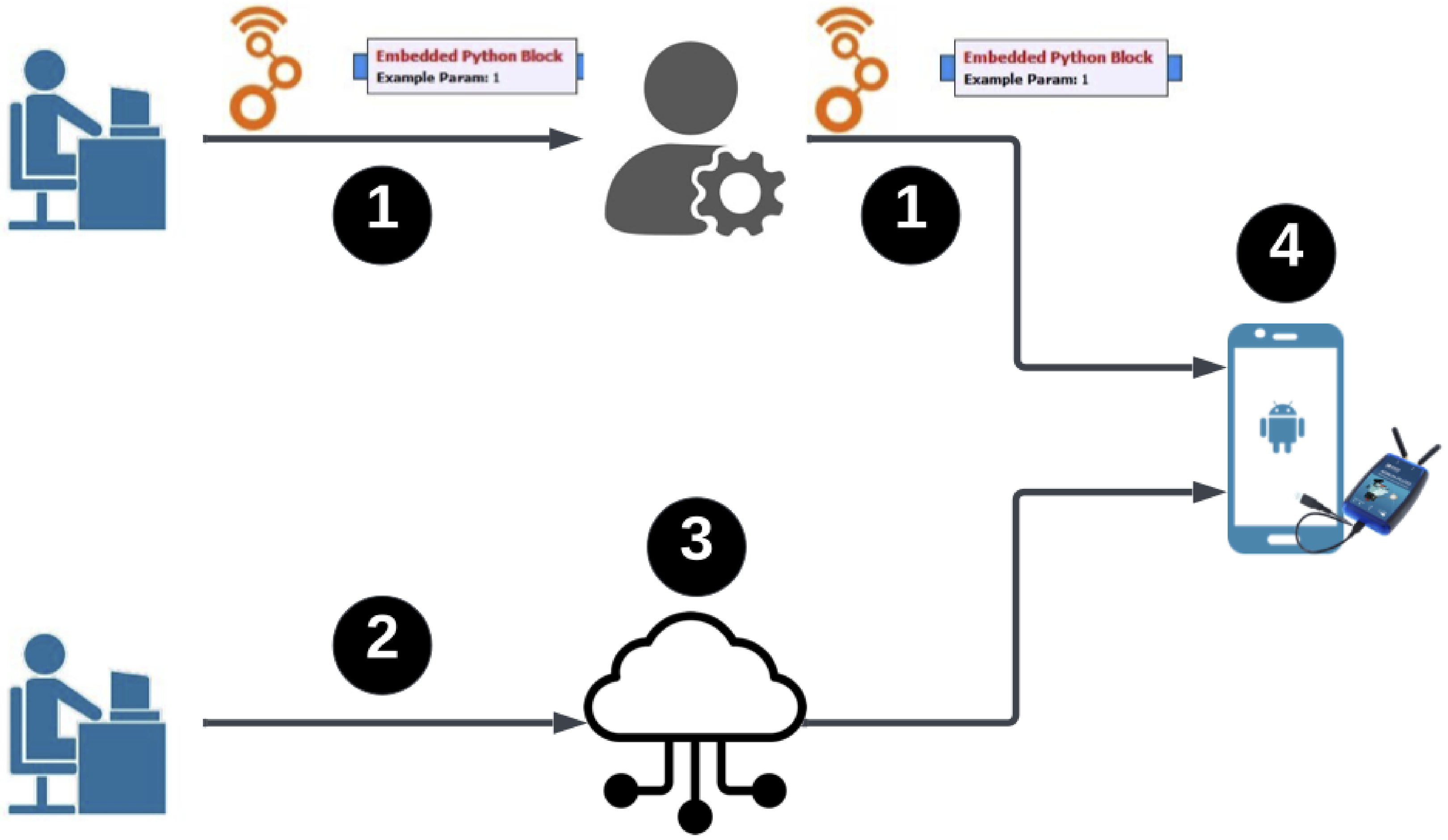


```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# SPDX-License-Identifier: GPL-3.0
#
# GNU Radio Python Flow Graph
# Title: Not titled yet
# GNU Radio version: v3.8.5.0-6-g57bd109d

from gnuradio import blocks
from gnuradio import gr
from gnuradio.filter import firdes
import sys
import signal
from argparse import ArgumentParser
from gnuradio.eng_arg import eng_float, intx
from gnuradio import eng_notation
from gnuradio import uhd
import time


class androidsdrtest(gr.top_block):

    def __init__(self):
        gr.top_block.__init__(self, "Not titled yet")

        ##################################################
        # Variables
        ##################################################
        self.samp_rate = samp_rate = 30720000

        ##################################################
        # Blocks
        ##################################################
        self.uhd_usrp_source_tofile_0 = uhd.usrp_source_tofile(
```

Architecture on the endpoint side



Access control



To achieve this Goal we need to..

Easy for endpoints to collect data and join the ecosystem

Ensure program security running on endpoint's device

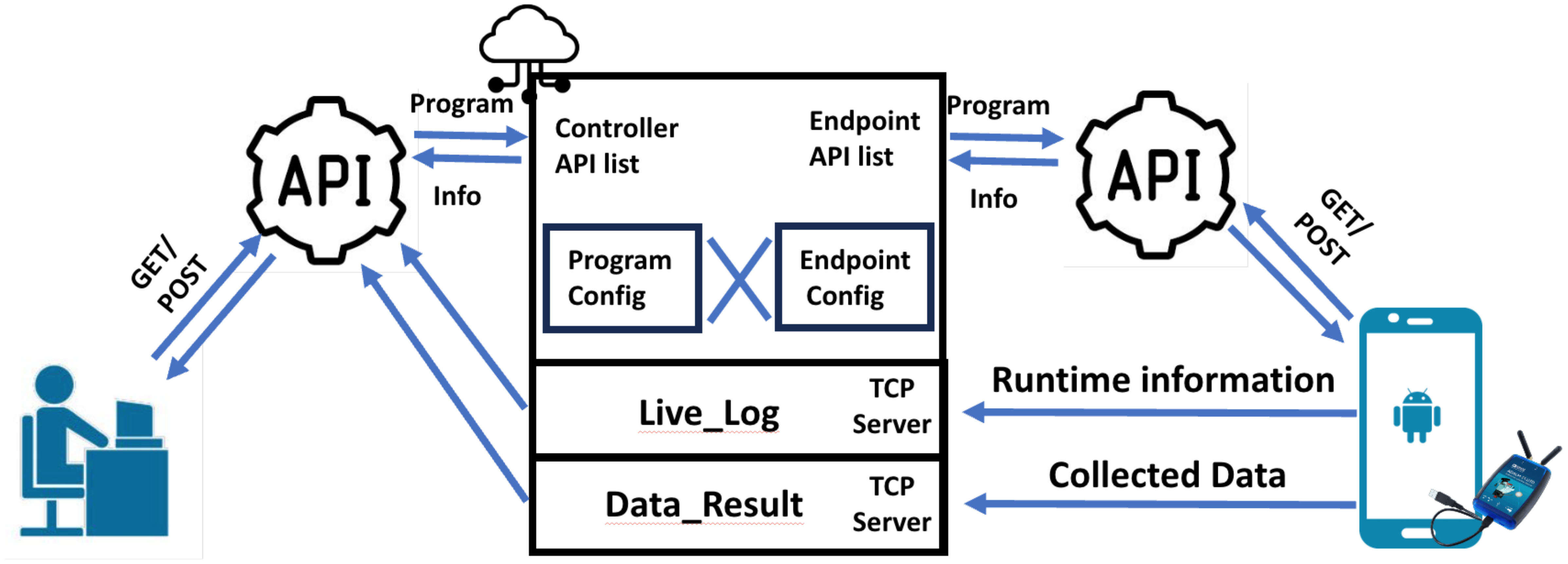
Ensure data integrity

Save battery on the phone

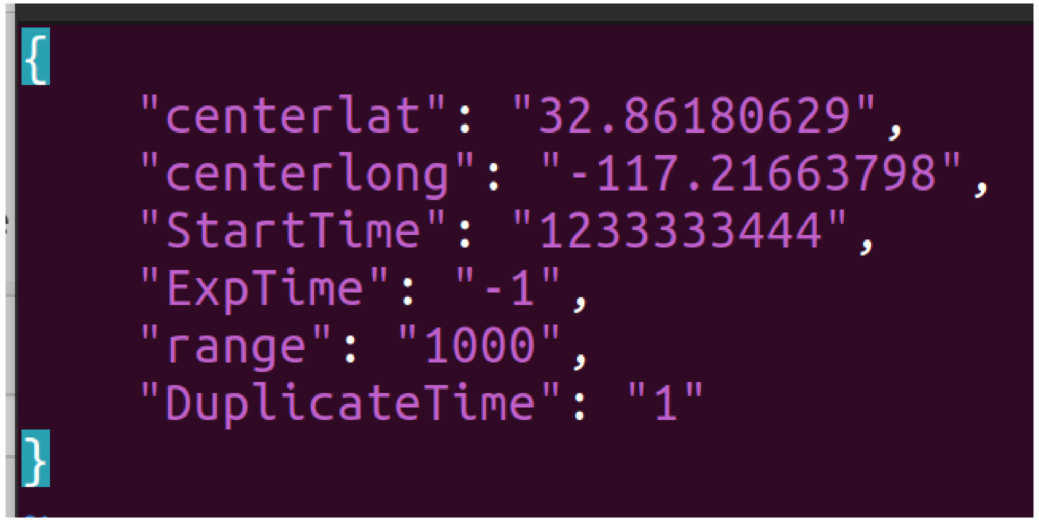
Go through limited computation resource on the phone

Broker automatically sign tasks to endpoint

Broker in the middle control task assignment



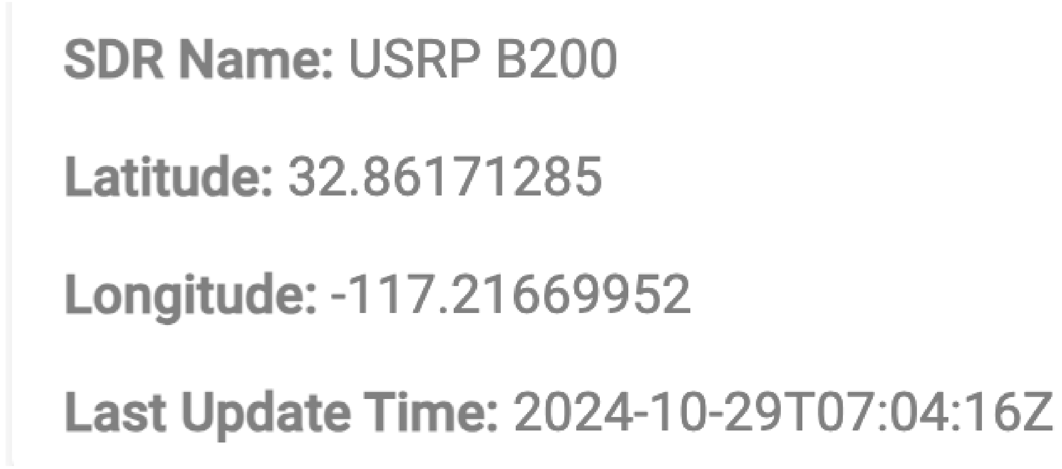
Configuration file



```
{
    "centerlat": "32.86180629",
    "centerlong": "-117.21663798",
    "StartTime": "1233333444",
    "ExpTime": "-1",
    "range": "1000",
    "DuplicateTime": "1"
}
```

Experiment configuration

X



SDR Name: USRP B200

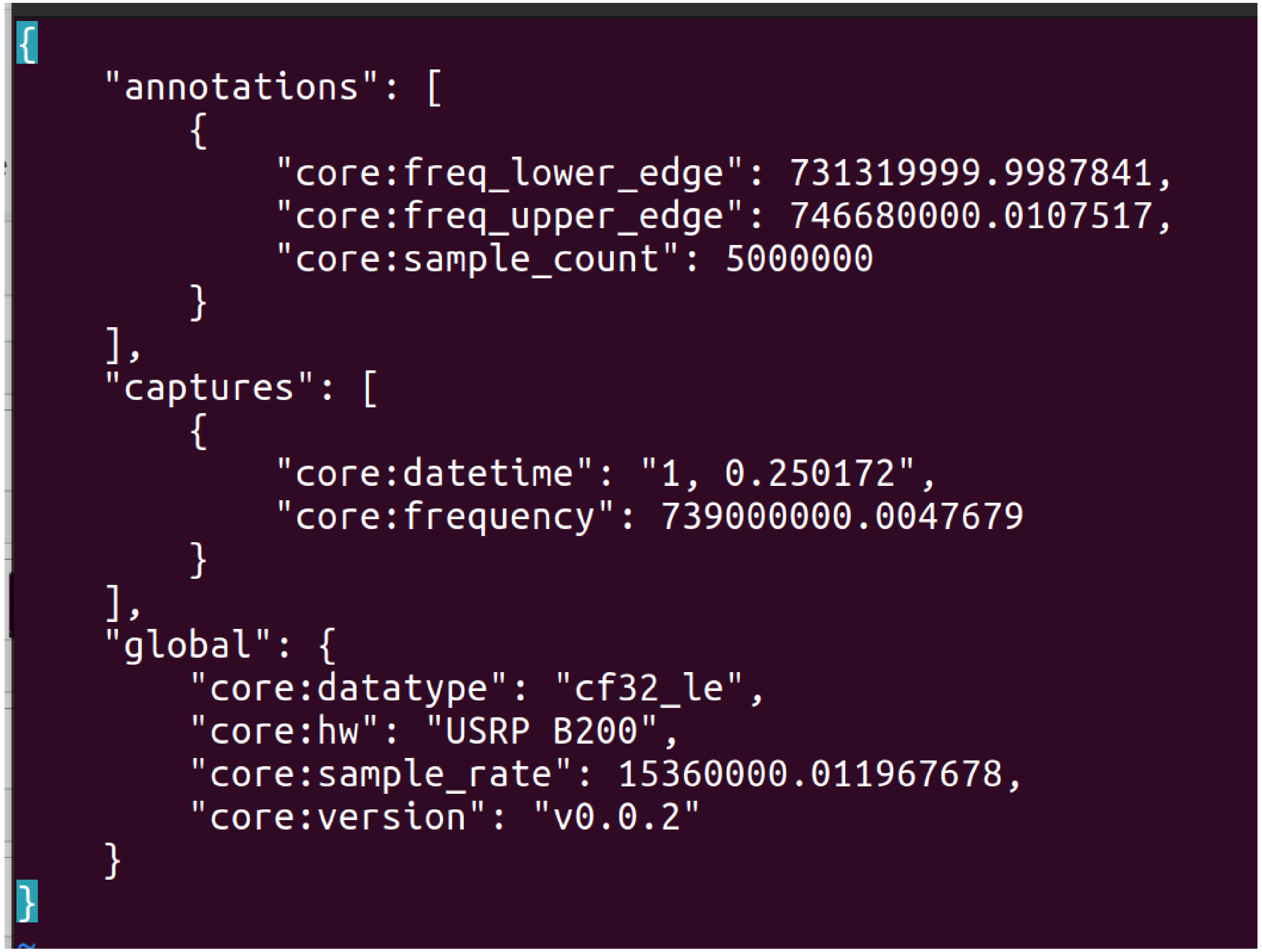
Latitude: 32.86171285

Longitude: -117.21669952

Last Update Time: 2024-10-29T07:04:16Z

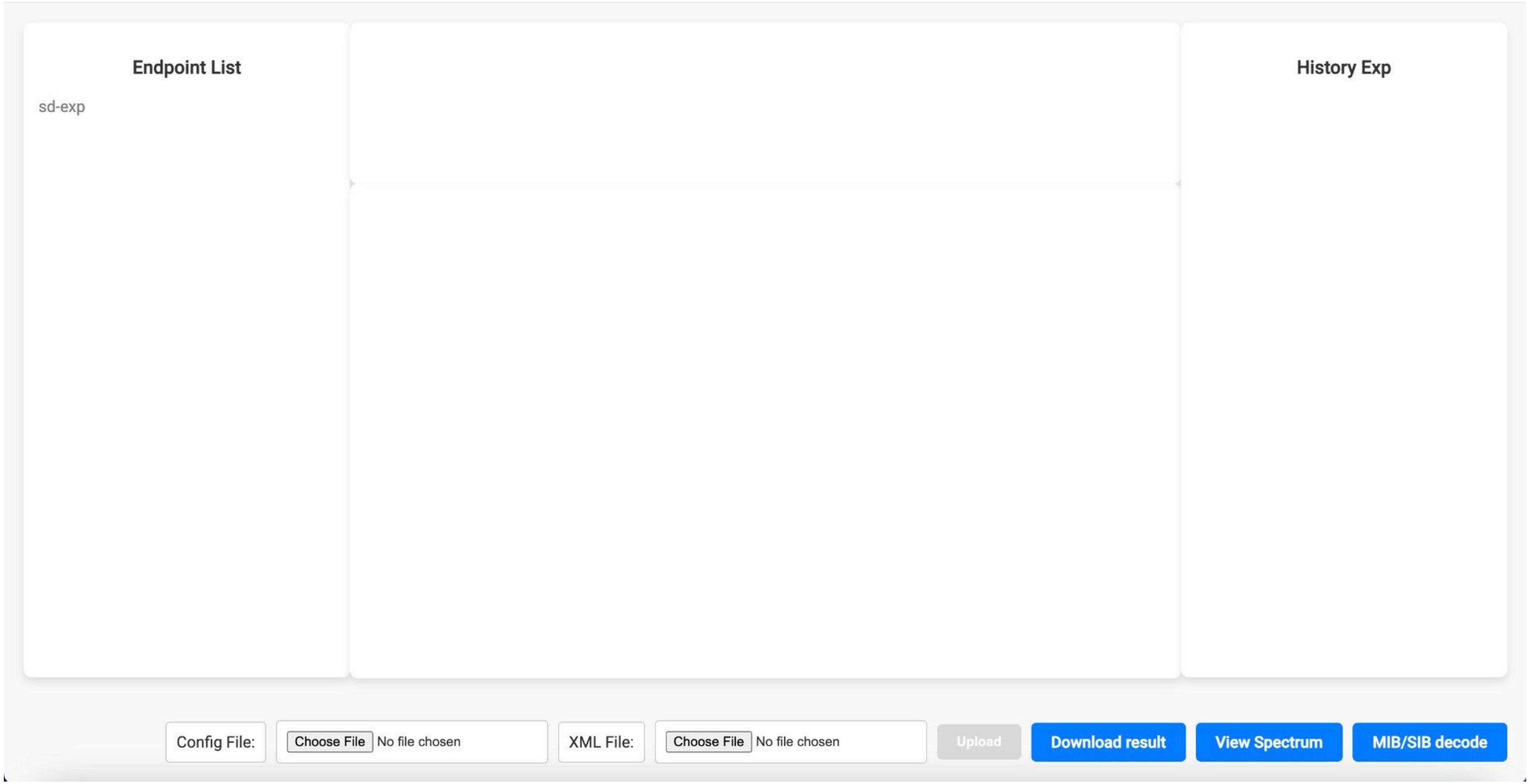
Endpoint configuration

Metadata



```
{
    "annotations": [
        {
            "core:freq_lower_edge": 731319999.9987841,
            "core:freq_upper_edge": 746680000.0107517,
            "core:sample_count": 5000000
        }
    ],
    "captures": [
        {
            "core:datetime": "1, 0.250172",
            "core:frequency": 739000000.0047679
        }
    ],
    "global": {
        "core:datatype": "cf32_le",
        "core:hw": "USRP B200",
        "core:sample_rate": 15360000.011967678,
        "core:version": "v0.0.2"
    }
}
```

MobileSDR demo with MIB/SIB decoder



Thanks for your attention

To achieve this Goal we need to..

Easy for endpoints to collect data and join the ecosystem

Ensure program security running on endpoint's device

Ensure data integrity

Save battery on the phone

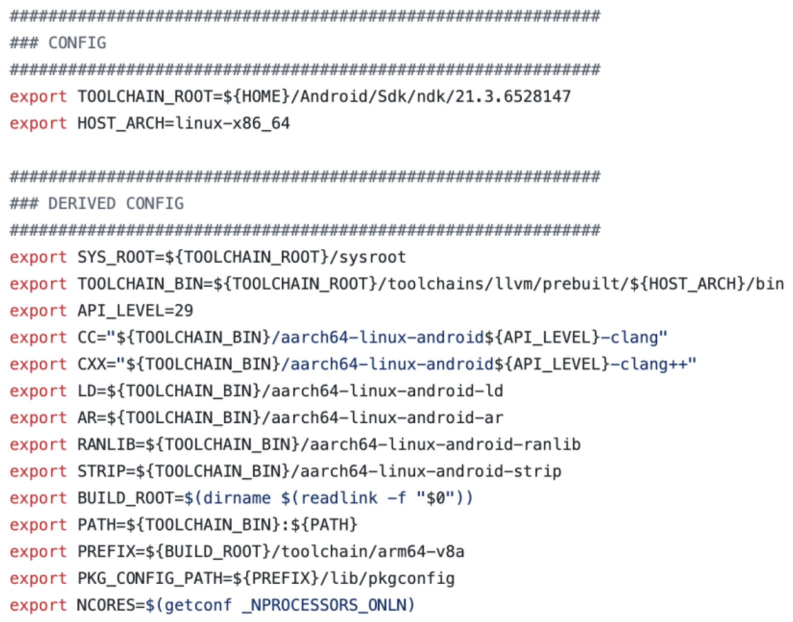
Go through limited computation resource on the phone

Easy for endpoints to collect data

Allow SDR connected with portable devices (Android phone) to collect data

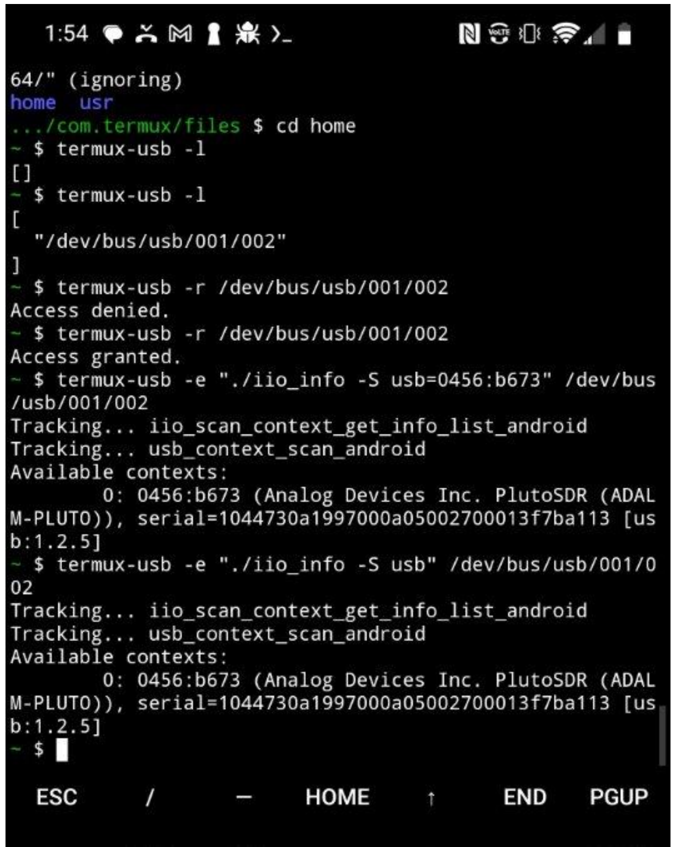
Cross compile SDR's framework on the phone

1. Cross compile with Android environment on Linux:



```
###################################################################
### CONFIG
###################################################################
export TOOLCHAIN_ROOT=${HOME}/Android/Sdk/ndk/21.3.6528147
export HOST_ARCH=linux-x86_64

###################################################################
### DERIVED CONFIG
###################################################################
export SYS_ROOT=${TOOLCHAIN_ROOT}/sysroot
export TOOLCHAIN_BIN=${TOOLCHAIN_ROOT}/toolchains/llvm/prebuilt/${HOST_ARCH}/bin
export API_LEVEL=29
export CC="${TOOLCHAIN_BIN}/aarch64-linux-android${API_LEVEL}-clang"
export CXX="${TOOLCHAIN_BIN}/aarch64-linux-android${API_LEVEL}-clang++"
export LD=${TOOLCHAIN_BIN}/aarch64-linux-android-ld
export AR=${TOOLCHAIN_BIN}/aarch64-linux-android-ar
export RANLIB=${TOOLCHAIN_BIN}/aarch64-linux-android-ranlib
export STRIP=${TOOLCHAIN_BIN}/aarch64-linux-android-strip
export BUILD_ROOT=$(dirname $(readlink -f "$0"))
export PATH=${TOOLCHAIN_BIN}:${PATH}
export PREFIX=${BUILD_ROOT}/toolchain/arm64-v8a
export PKG_CONFIG_PATH=${PREFIX}/lib/pkgconfig
export NCORES=$(getconf _NPROCESSORS_ONLN)
```

2. Compile on Termux:



```
1:54
64/" (ignoring)
home  usr
.../com.termux/files $ cd home
~ $ termux-usb -l
[]
~ $ termux-usb -l
[
  "/dev/bus/usb/001/002"
]
~ $ termux-usb -r /dev/bus/usb/001/002
Access denied.
~ $ termux-usb -r /dev/bus/usb/001/002
Access granted.
~ $ termux-usb -e "./iio_info -S usb=0456:b673" /dev/bus
/usb/001/002
Tracking... iio_scan_context_get_info_list_android
Tracking... usb_context_scan_android
Available contexts:
        0: 0456:b673 (Analog Devices Inc. PlutoSDR (ADAL
M-PLUTO)), serial=1044730a1997000a05002700013f7ba113 [us
b:1.2.5]
~ $ termux-usb -e "./iio_info -S usb" /dev/bus/usb/001/0
02
Tracking... iio_scan_context_get_info_list_android
Tracking... usb_context_scan_android
Available contexts:
        0: 0456:b673 (Analog Devices Inc. PlutoSDR (ADAL
M-PLUTO)), serial=1044730a1997000a05002700013f7ba113 [us
b:1.2.5]
~ $ 
```

To achieve this Goal we need to..

Easy for endpoints to collect data and join the ecosystem

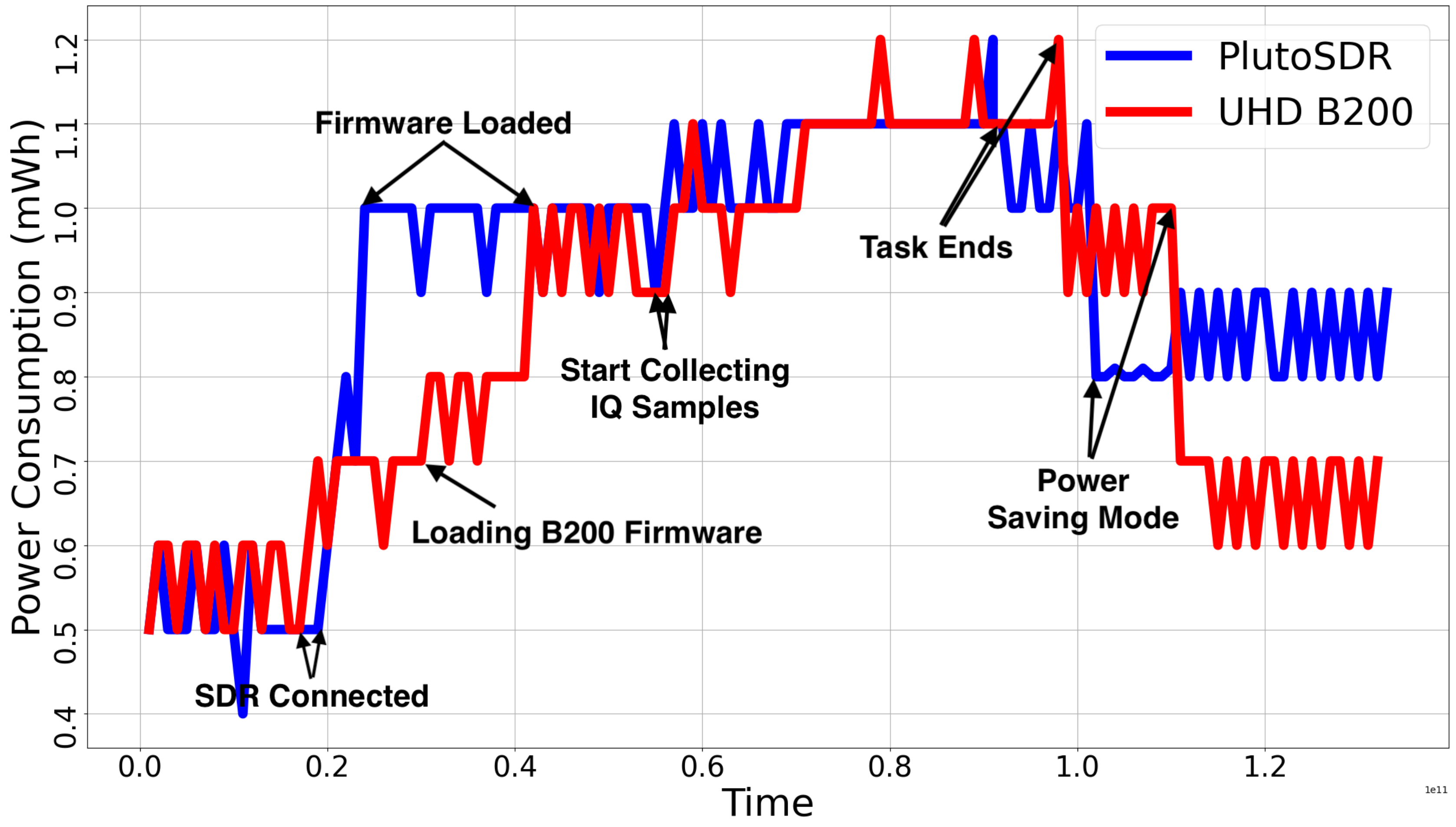
Ensure program security running on endpoint's device

Ensure data integrity

Save battery on the phone

Go through limited computation resource on the phone

Duty Cycle


Power Consumption (mWh)
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e11
Time
PlutoSDR
UHD B200
Firmware Loaded
Start Collecting IQ Samples
Task Ends
Loading B200 Firmware
Power Saving Mode
SDR Connected

To achieve this Goal we need to..

Easy for endpoints to collect data and join the ecosystem

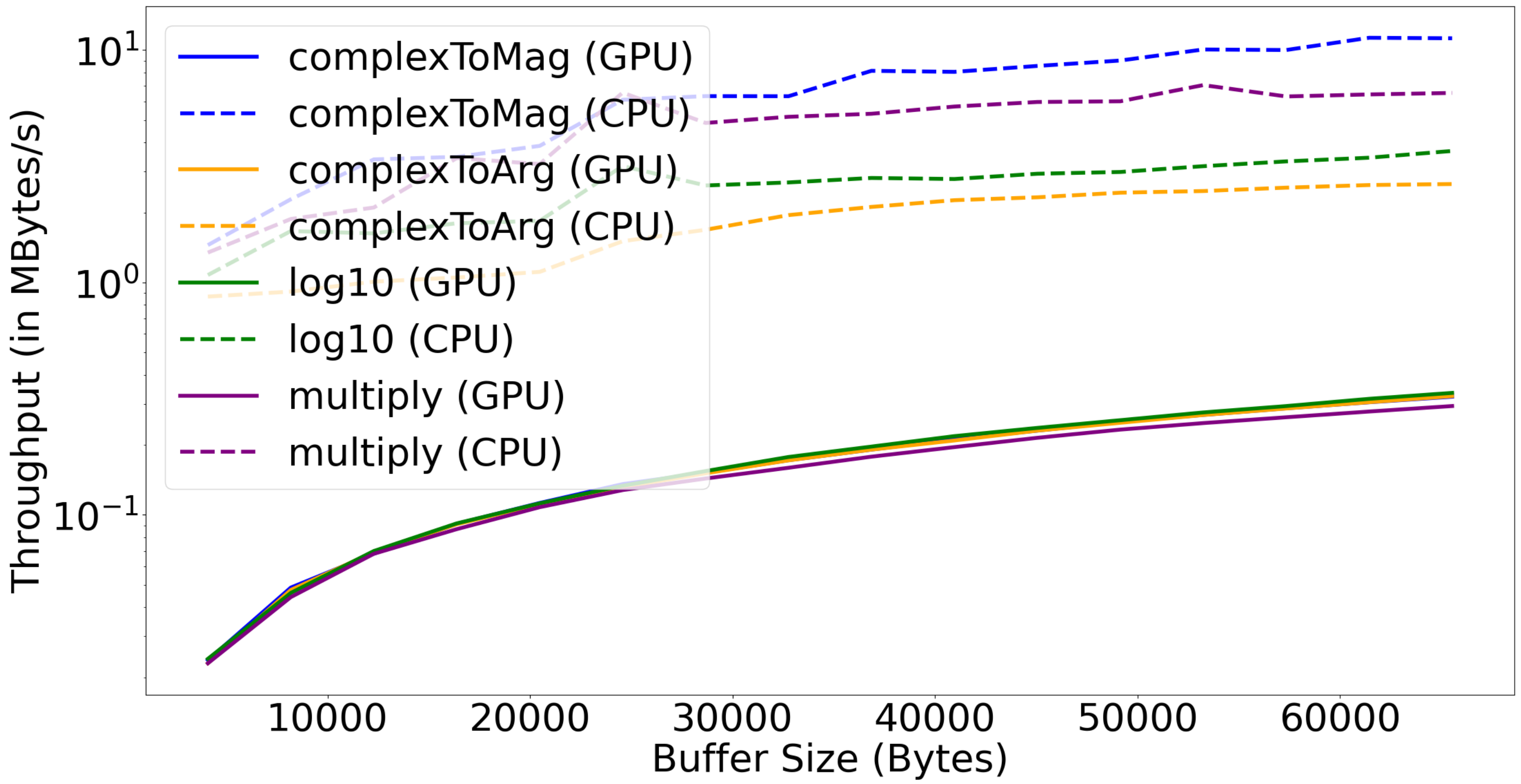
Ensure program security running on endpoint's device

Ensure data integrity

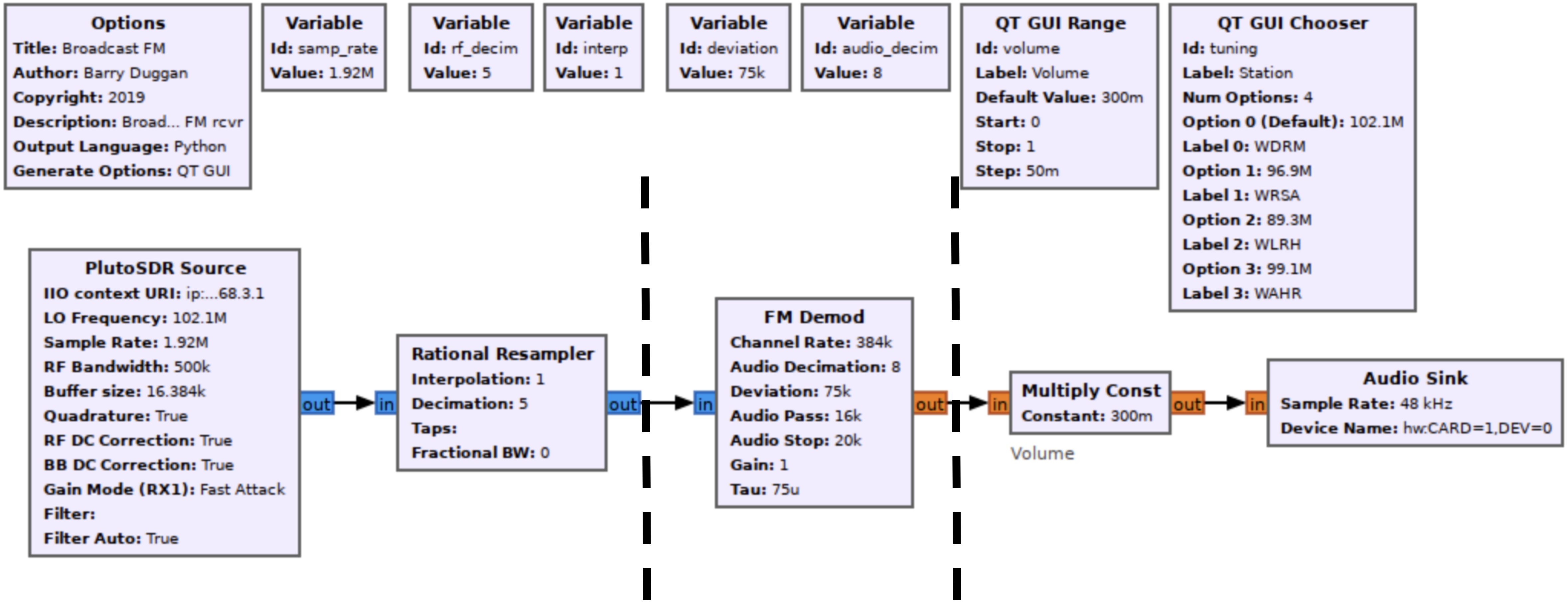
Save battery on the phone

Go through limited computation resource on the phone

Go through computation limitation on the phone



Load tasks on different devices



Options
Title: Broadcast FM
Author: Barry Duggan
Copyright: 2019
Description: Broad... FM rcvr
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 1.92M

Variable
Id: rf_decim
Value: 5

Variable
Id: interp
Value: 1

Variable
Id: deviation
Value: 75k

Variable
Id: audio_decim
Value: 8

QT GUI Range
Id: volume
Label: Volume
Default Value: 300m
Start: 0
Stop: 1
Step: 50m

QT GUI Chooser
Id: tuning
Label: Station
Num Options: 4
Option 0 (Default): 102.1M
Label 0: WDRM
Option 1: 96.9M
Label 1: WRSA
Option 2: 89.3M
Label 2: WLRH
Option 3: 99.1M
Label 3: WAHR

PlutoSDR Source
IIO context URI: ip:...68.3.1
LO Frequency: 102.1M
Sample Rate: 1.92M
RF Bandwidth: 500k
Buffer size: 16.384k
Quadrature: True
RF DC Correction: True
BB DC Correction: True
Gain Mode (RX1): Fast Attack
Filter:
Filter Auto: True

Rational Resampler
Interpolation: 1
Decimation: 5
Taps:
Fractional BW: 0

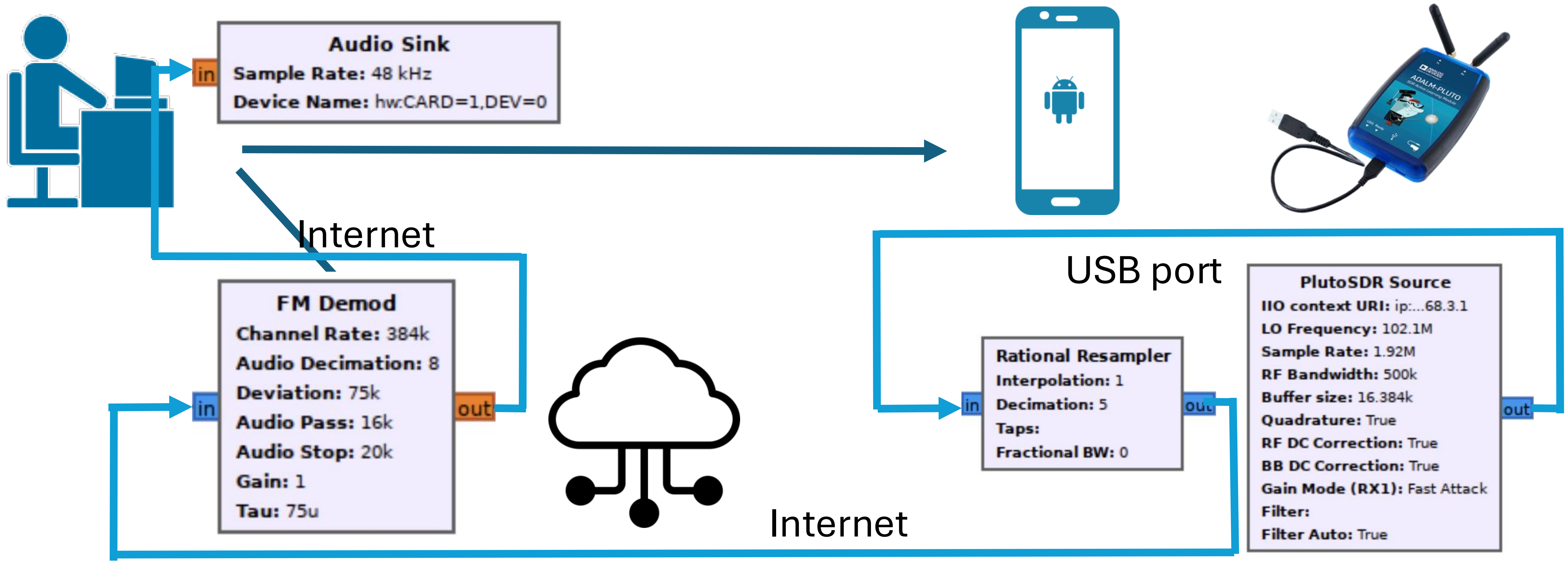
FM Demod
Channel Rate: 384k
Audio Decimation: 8
Deviation: 75k
Audio Pass: 16k
Audio Stop: 20k
Gain: 1
Tau: 75u

Multiply Const
Constant: 300m

Volume

Audio Sink
Sample Rate: 48 kHz
Device Name: hw:CARD=1,DEV=0

Load tasks on different devices



Audio Sink
Sample Rate: 48 kHz
Device Name: hw:CARD=1,DEV=0

FM Demod
Channel Rate: 384k
Audio Decimation: 8
Deviation: 75k
Audio Pass: 16k
Audio Stop: 20k
Gain: 1
Tau: 75u

Rational Resampler
Interpolation: 1
Decimation: 5
Taps:
Fractional BW: 0

PlutoSDR Source
IIO context URI: ip:...68.3.1
LO Frequency: 102.1M
Sample Rate: 1.92M
RF Bandwidth: 500k
Buffer size: 16.384k
Quadrature: True
RF DC Correction: True
BB DC Correction: True
Gain Mode (RX1): Fast Attack
Filter:
Filter Auto: True