Drone 5G Integration

Jay & Jason

August 15, 2022

Geon Technologies, LLC

Drone Project Goals

- RB5 Platform offers a mobile platform for data collection and deployment
- Application
  - **5G Reconnaissance**
  - **5G Spoofing**

Outline

Drone Infrastructure

- UHD Container Installation
  - Provides a **low risk**, generic **collection system**, and potential jammer deployment
- OAI Installation
  - **5G reception and transmission** capability on drone

Application

- 5G Reconnaissance
  - Collection from **UHD allows for signal detection** and **data collection**

Drone Infrastructure

UHD

CONTAINER APPLICATION

OAI INSTALLATION

Drone UHD Container

UHD Container Installation

Needed to install dependencies:

- apt-utils
- libboost-all-dev
- Cmake
- libusb-1.0-0-dev
- git python3
- python3-dev
- python3-pip

Created bash script to install UHD git repository:

- Tested locally on x86
  - Needed privileged mode and a shared volume of "/dev/bus/usb:/dev/bus/usb"
- Needed the 'buildx' command to build on ARM for drone

Objective

Drone:

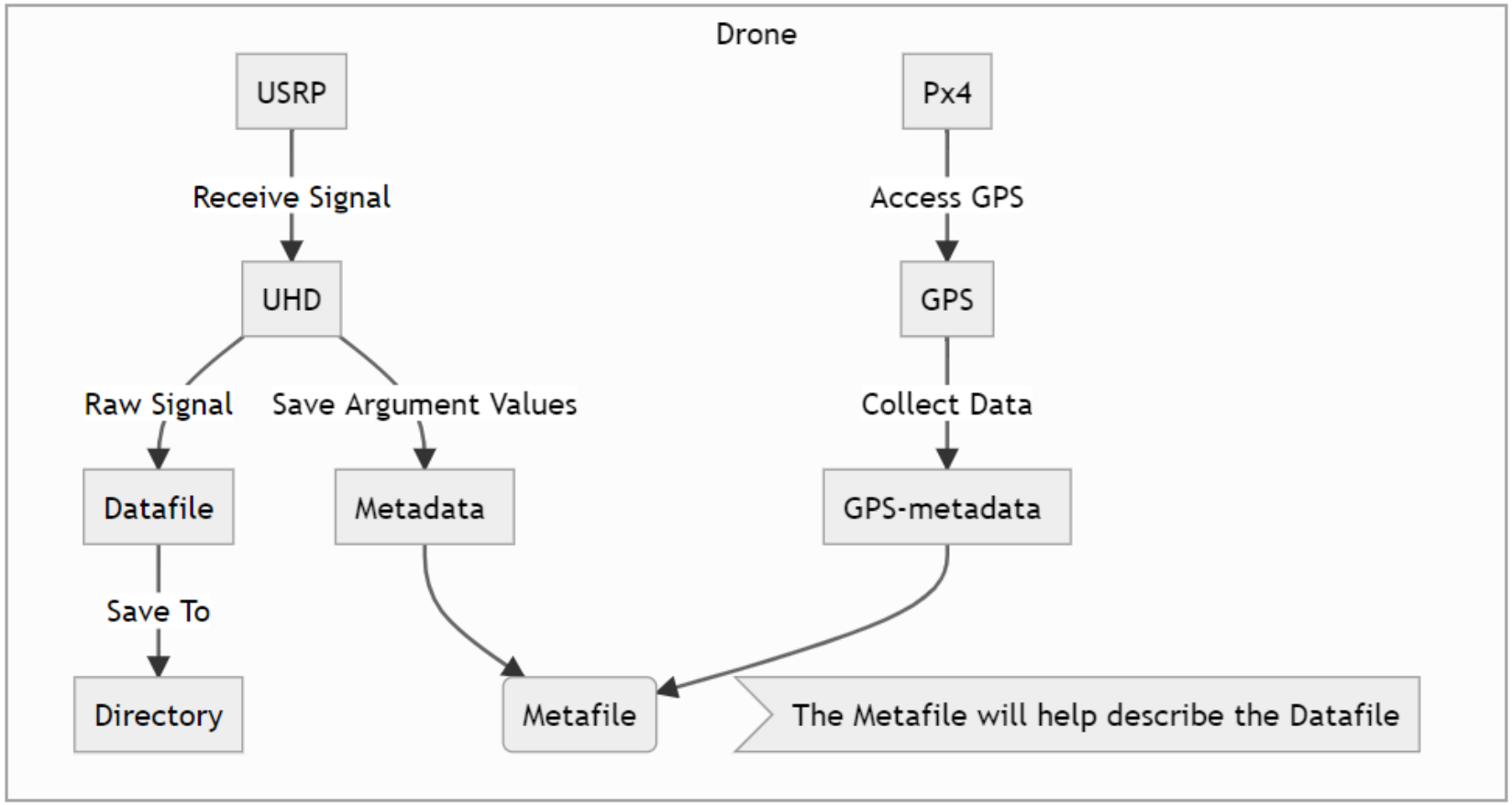
- **Use drone to capture information (raw signals, metadata)**

How:

Drone:

- Use UHD to capture signal
- Access GPS information
- Save UHD and GPS information (metadata) into a file

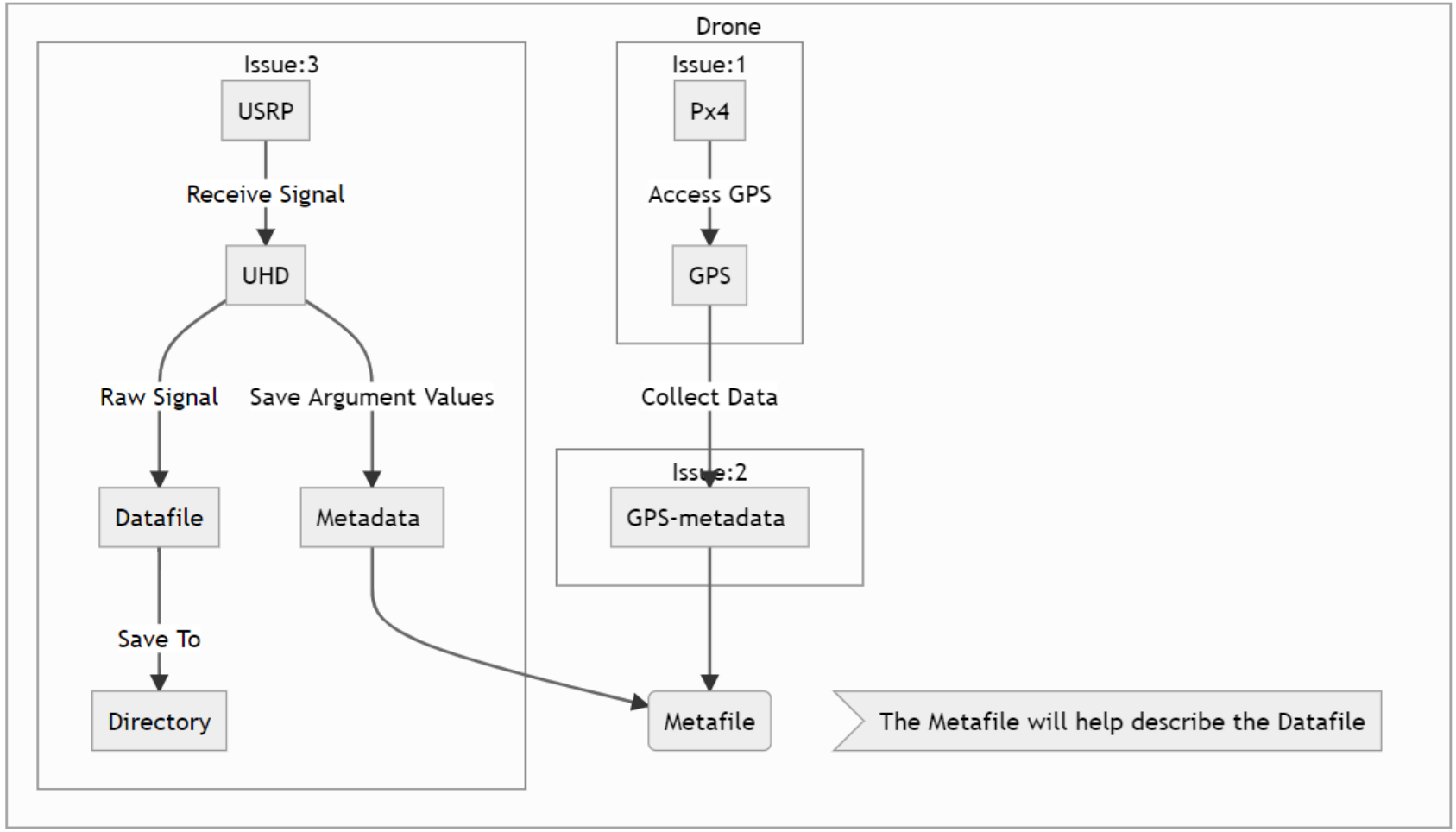
Objective: Use drone to capture information (raw signals, metadata)



Milestone 1

Major Requirements (Issues):

- Access GPS on the drone for location
- Use GPS for timing
- Use SigMF to create a metafile from meta data



Access GPS on Drone

- How: Utilize the Px4 on the drone to start the GPS and check the status of the GPS information (latitude, longitude, altitude, time in EPOCH)

```
root@qrb5165-rb5:~# px4-gps stop
root@qrb5165-rb5:~# px4-gps start -d /dev/ttyHS2 -b 115200
root@qrb5165-rb5:~# px4-gps status
```


Output	Value	Final Answer
time_utc_usec:	1658236383399603	Tuesday, July 19, 2022 9:13:03.399 AM [1]
lat:	391674603	39.16774603
lon:	-768094451	-76.8094451
alt:	52313	52.313

GPS Status Information

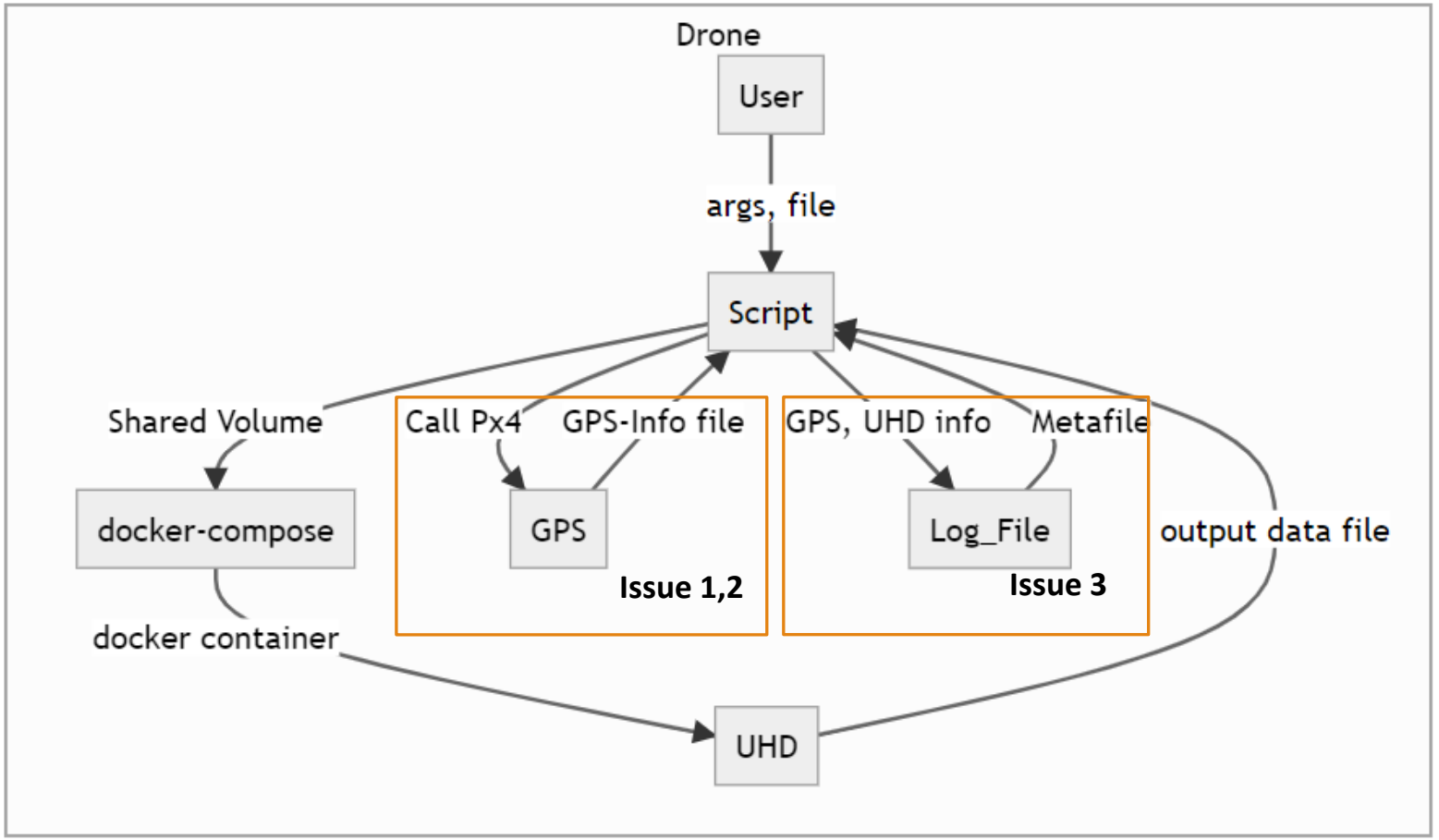
```
timestamp:
time_utc_usec:
lat:
lon:
alt:
alt_ellipsoid:
s_variance_m_s:
c_variance_rad:
eph:
epv:
hdop:
vdop:
noise_per_ms:
jamming_indicator:
vel_m_s:
vel_n_m_s:
vel_e_m_s:
vel_d_m_s:
cog_rad:
timestamp_time_relative:
heading:
heading_offset:
fix_type:
jamming_state:
vel_ned_valid:
satellites_used:
```

Use SigMF to Create a Metafile

- How: Use SigMF to create a metafile that contains the metadata collected from the GPS and UHD
- Obstacle: Unable to install SigMF on drone (pip issue), created docker compose with shared volumes for the GPS to be shared in container of UHD
- Update to Design: Created a log file that contains the information from the UHD and GPS
- **Test:**
- ./Interactive_Python_GPS_UHD.py --input rx_samples_to_udp --rate 10000000 --freq 900000000 --gain 1 --nsamps 100000 --port 80 --addr 10.3.4.2

```
root@qrb5165-rb5:~/rx_file_output# cat output_data.log
The file chosen is: rx_samples_to_udp
The sampling rate is: 10000000
Total number of samples collected is: 100000
Port value selected is: 80
The center frequency is: 900000000
Gain chosen: 1
Address chosen is: 10.3.4.2
The longitude is: 0.0
The latitude is: 0.0
The altitude is: -17.0
The time in EPOCH is: 0
```

Approach: Script



Demo

Commands to Use:

1. **root@qrb5165-rb5:~/jay#:** docker compose up –d
2. **root@qrb5165-rb5:~/jay#:** ./Interactive_Python_GPS_UHD.py --input rx_samples_to_file --rate 10000000 --freq 900000000 --gain 1 --duration 3 --filename example
3. **root@qrb5165-rb5:~/jay#:** docker compose down

What to expect:

- Output log file “metafile”: ~/rx_file_output
- Datafile: ~/rx_file_output
- Overall GPS Information: /tmp
- There may be an inconsistency in altitude if so unplug and replug in drone
- More information Here

OAI Installation

OAI Installation

- Cross-compilation necessary for OAI use on ARM-based drone
  - `docker buildx` reference: Drone.md

- Verification of build on drone
  - Test existence of all necessary packages using python script
  - Run UE on drone and gNB on workstation and test connection
  - Run gNB on drone and UE on workstation and test connection

OAI Installation

ran-base

Latest source files

Necessary packages and compilers to run an OAI RAN executable

Compiles target images for ARM64

ran-build

Builds all target images using ‘ran-base’

Target Images (gNB & nr-UE)

Only contains generated executable, generated shared libraries, necessary libraries and packages to run generated binaries

Goal: Install on Drone

Image Construction

- ./base_command
  - Runs `docker buildx build` to create the 'ran-base' from my edited Dockerfile
- ./build_command
  - Builds 'ran-build' using ran-base's installed source files & libraries
  - Creates 'gNB' & 'NR-uE' images to be installed on drone

- Location

```
if defined(__x86_64__) || defined(__i386__)
 vect128 realPart = _mm_madd_epi16(*a,*b);
 realPart = _mm_srai_epi32(realPart,output_shift);
 vect128 imagPart = _mm_shufflelo_epi16(*b,_MM_SHUFFLE(2,
 imagPart = _mm_shufflehi_epi16(imagPart,_MM_SHUFFLE(2,3,
 imagPart = _mm_sign_epi16(imagPart,*(vect128 *)minusConj
 imagPart = _mm_madd_epi16(imagPart,*a);
 imagPart = _mm_srai_epi32(imagPart,output_shift);
 vect128 lowPart = _mm_unpacklo_epi32(realPart,imagPart);
 vect128 highPart = _mm_unpackhi_epi32(realPart,imagPart)
 return ( _mm_packs_epi32(lowPart,highPart));
elif defined(__arm__)
 AssertFatal(false, "not developped\n");
endif
```

Challenges

- Required creation of script for manual installation of necessary libraries
  - LAPACK – linear algebra package
  - OpenJDK – open source implementation of Java SE
- **Lack of Development for ARM Intrinsics in OAI**
  - sse_intrin.h

Future Work

- sse2neon – open-source C++ header file that performs intrinsics translation from x86 to arm64
- SIMDe – open source intrinsics translation library which builds on sse2neon
  - Potential future use for any Geon project that requires intrinsic translation

- Mobile 5G Reconnaissance & DoS attacks

- Link to Report
  - Major issues
  - Preliminary implementation of sse2neon
- Link to Journal Documenting Troubleshooting
  - Week 5 Journal Entry
  - Week 6 Journal Entry

Outline

Drone Infrastructure

- UHD Container Installation
  - Provides a **low risk**, generic **collection system**, and potential jammer deployment
- OAI Installation
  - Use of **5G reception and transmission** capability on drone

Application

- 5G Reconnaissance
  - Collection from **UHD allows for signal detection** and **data collection**

Objective

5G Reconnaissance

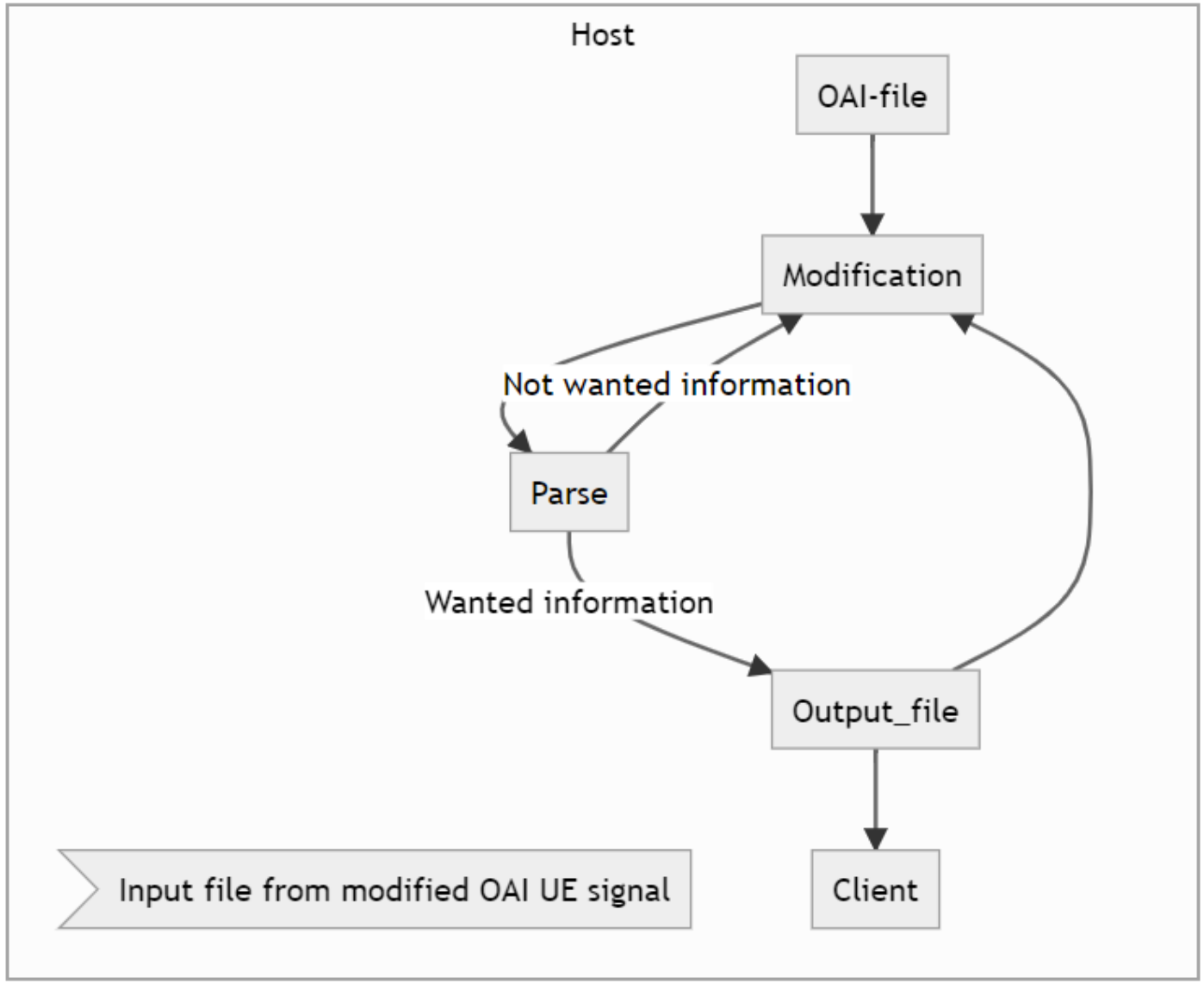
- **Parse OAI receiver log file for cell information**

How:

Host Platform:

- Continuously search for updates to a OAI log file
- Parse Updated Log File for wanted cell information
- Export cell information to client

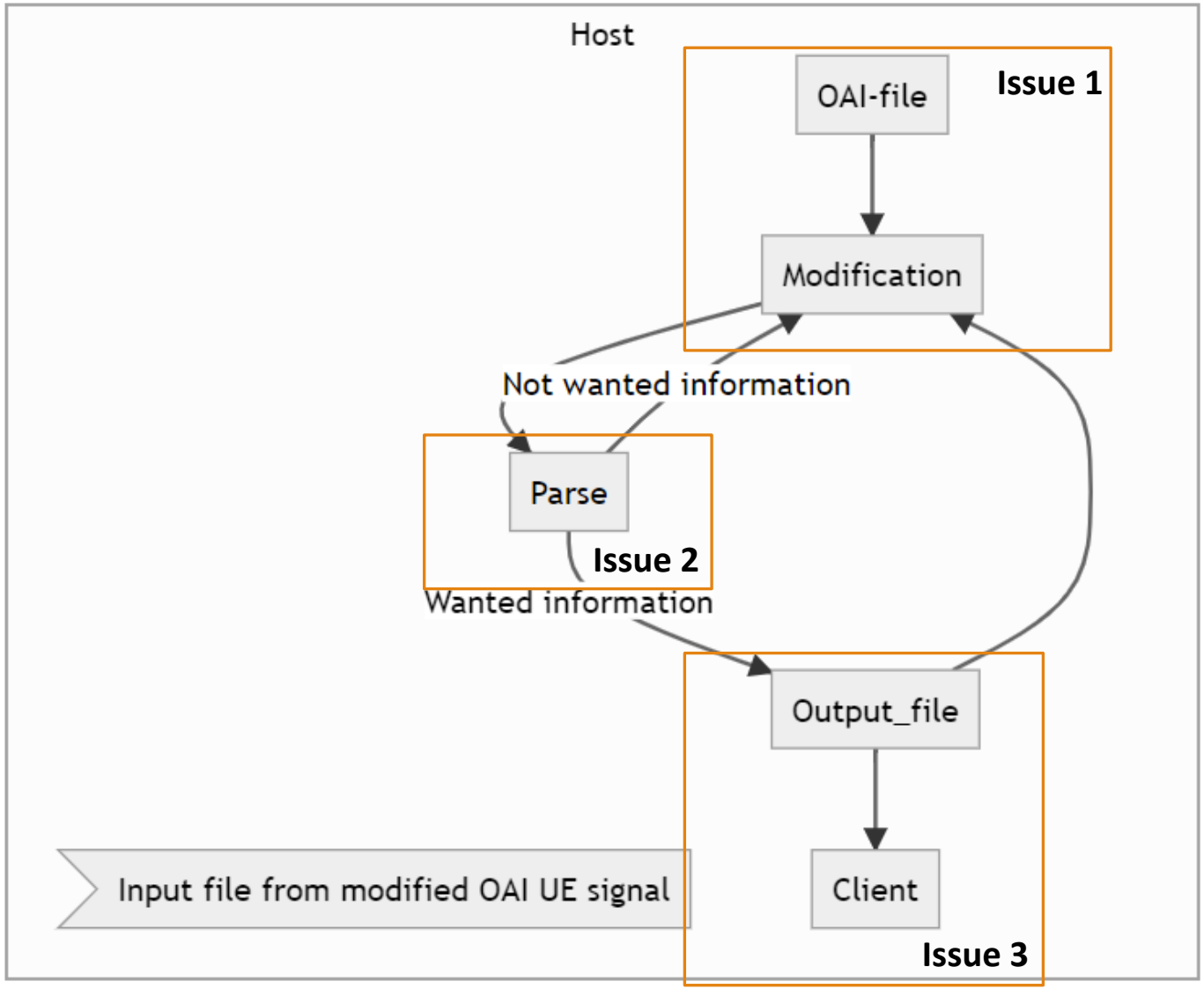
Objective: Parse OAI receiver log file for cell information



Milestone 2

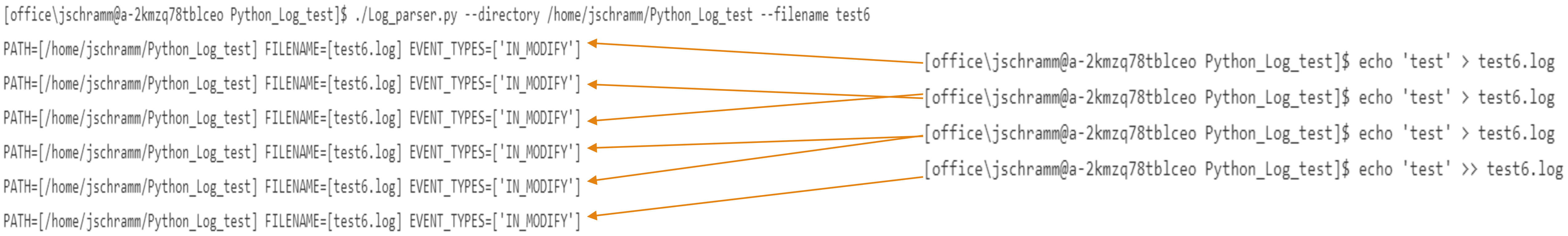
Major Requirements (Issues):

- Use Python Inotify to scan for updates to a log file
- Parse the new lines and detect specific messages
- Update detections to database or use messaging protocol (zmq) to forward to next application
  - (Export messages to client)



Use Python Inotify to Scan for Updates

- How: Inotify will continuously wait and check in a wanted file for modifications made
  - With a given directory and filename of the wanted file given



Parse New Lines and Detect Specific Messages

- How: Parse the modified log file starting from the last line of the previous modification and search for special token ($i+@ware)
  - This will avoid parsing already parsed lines from the previous modifications
  - Save output messages to file
- Example messages:
  - MIB Spec = Frame, SCS common, Type A Pos, Cell Barred, PDCCH CORESET0, pdcch_SS0
  - SIB1 FreqInfoDL.SCS_SpecificCarrierList = Offset to Carrier, Subcarrier Spacing, carrier BW
  - SIB1 PLMN Info = TAC, RANAC, CellResOpUse, MNC, MCC, CellId
  - SIB1 Paging Info = Paging Cycle, ns
  - SIB1 ServingCellConfig Info = SSB Period, PBCH Block Power, Offset to Point A
  - SIB1 TDD_UL_DL_ConfigCommon = Ref SCS, pattern1, pattern2
  - PhysCellId, SCS, N_RB_DL, NRB_UL, SSB SC Offset
  - Common (freq offset, eNB ID)

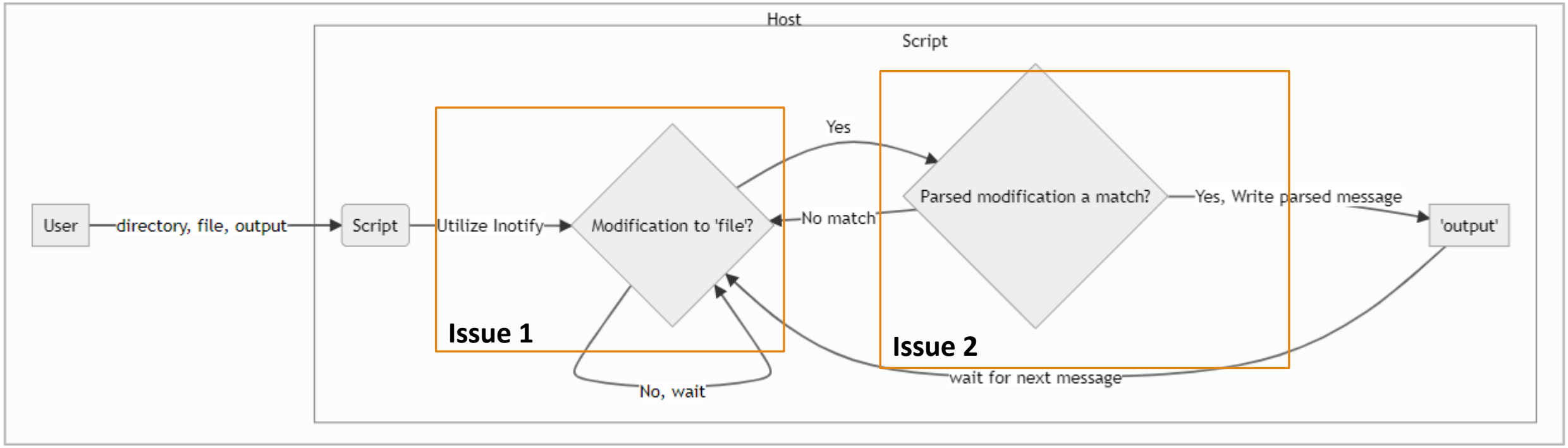
Input to terminal with Token

```
[office\jschramm@a-2kmzq78tblceo Python_Log_test]$ echo 'm3m[NR_RRC]   SIB1 ServingCel
lConfig Info($i+@ware) = SSB Period, PBCH Block Power, Offset to Point A
> m3m[NR_RRC]   SIB1 ServingCellConfig Info($i+@ware) = 2, -25, 86
> m3m[NR_RRC]   SIB1 FreqInfoDL.frequencyBandList = 78
> m3m[NR_RRC]   SIB1 FreqInfoDL.SCS_SpecificCarrierList = Offset to Carrier, Subcarrie
r Spacing, carrier BW
> m3m[NR_RRC]   SIB1 FreqInfoDL.SCS_SpecificCarrierList($i+@ware) = 0, 1, 106
> m3m[NR_RRC]   SIB1 Paging Info includes monitoring occasions...not extracted yet
> m3m[NR_RRC]   SIB1 Paging Info($i+@ware) = Paging Cycle, ns
> m3m[NR_RRC]   SIB1 Paging($i+@ware) = 3, 2
> m3m[NR_RRC]   SIB1 TDD_UL_DL_ConfigCommon($i+@ware) = Ref SCS, pattern1, pattern2
> m3m[NR_RRC]   SIB1 TDD_UL_DL_ConfigCommon($i+@ware) = 1, 6, -1
> m3m[NR_RRC]   Set PLMN Id Info List, has 1 items
> m3m[NR_RRC]   SIB1 PLMN Info($i+@ware) = TAC, RANAC, CellResOpUse, MNC, MCC, CellId
> m3m[NR_RRC]   SIB1 PLMN ($i+@ware) = 321, -1, 1, 99, 208, 57344' >> example2.log
```

Output Parsed Message File

```
[office\jschramm@a-2kmzq78tblceo inotify_result]$ cat example2_output.log
   SIB1 ServingCellConfig Info = SSB Period, PBCH Block Power, Offset to Point A
   SIB1 ServingCellConfig Info = 2, -25, 86
   SIB1 FreqInfoDL.SCS_SpecificCarrierList = 0, 1, 106
   SIB1 Paging Info = Paging Cycle, ns
   SIB1 Paging = 3, 2
   SIB1 TDD_UL_DL_ConfigCommon = Ref SCS, pattern1, pattern2
   SIB1 TDD_UL_DL_ConfigCommon = Ref SCS, pattern1, pattern2
   SIB1 TDD_UL_DL_ConfigCommon = 1, 6, -1
   SIB1 TDD_UL_DL_ConfigCommon = 1, 6, -1
   SIB1 PLMN Info = TAC, RANAC, CellResOpUse, MNC, MCC, CellId
   SIB1 PLMN  = 321, -1, 1, 99, 208, 57344
```

Parse New Lines and Detect Specific Messages



Approach: Script

How to Use

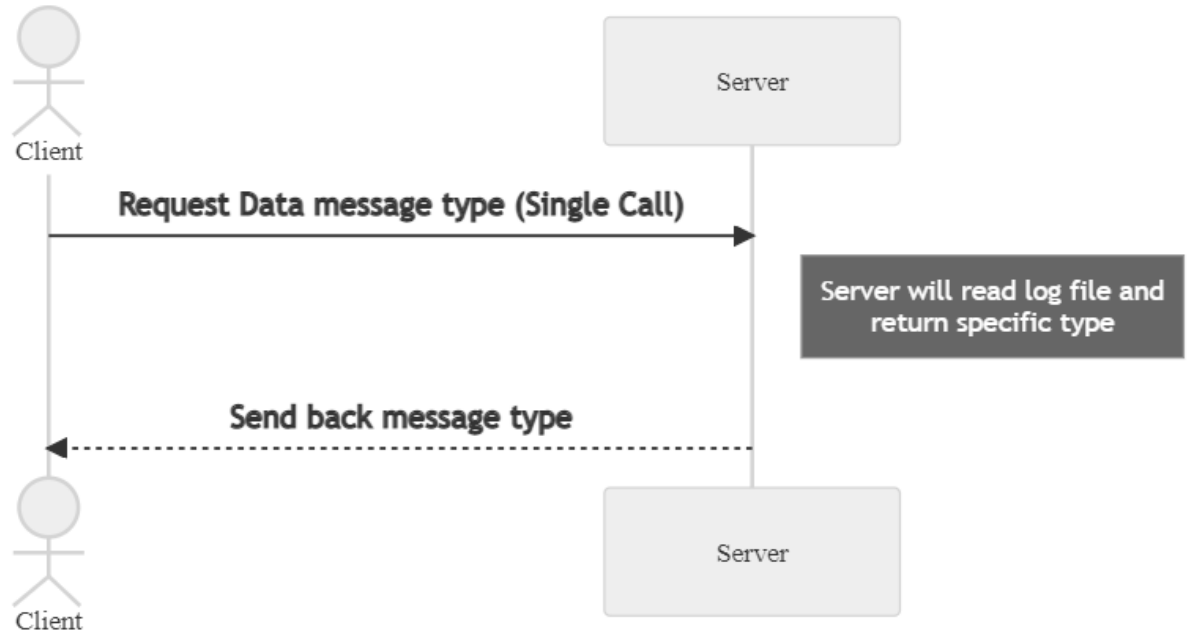
Commands to Use:

1. **jay@5g-lab-01:~/Documents$:** ./Drone_Parser.py --directory /tmp --filename sit_aware --output /home/jay/Documents
2. One terminal: Output information to the filename using output redirection

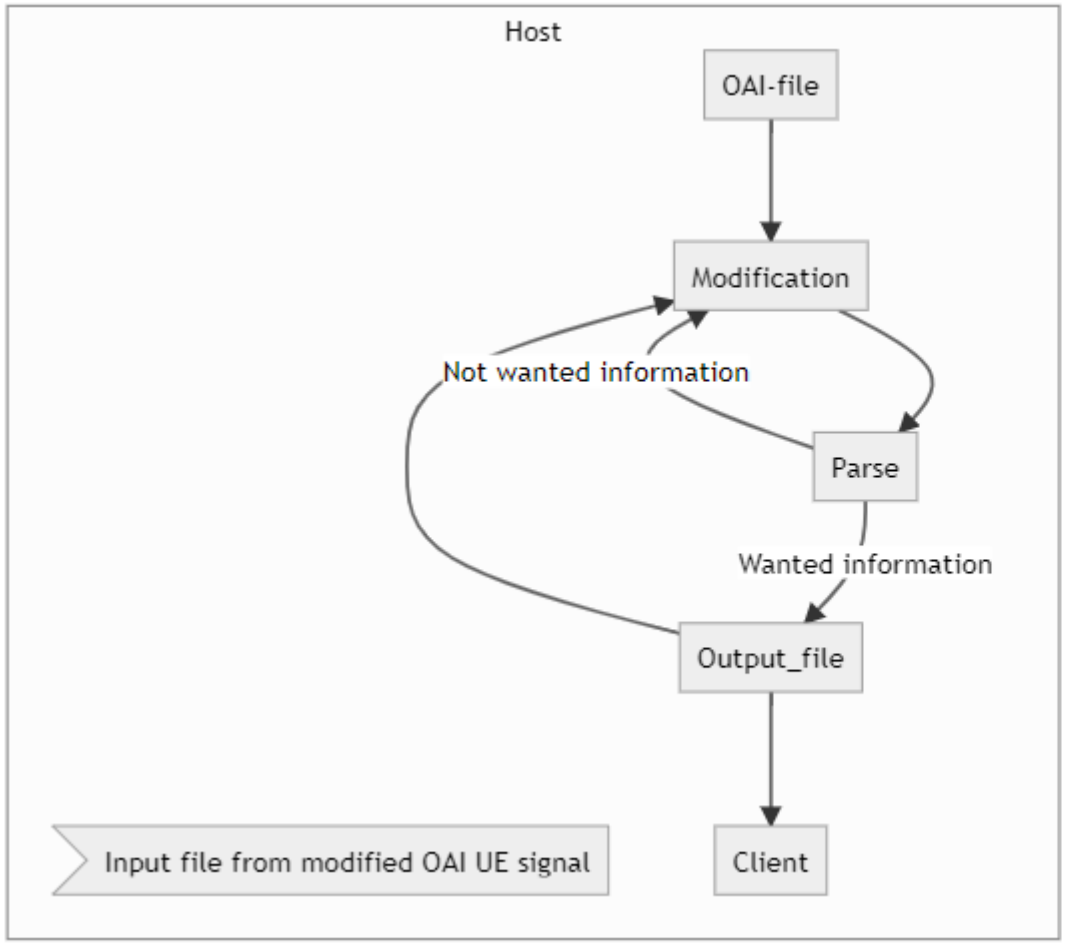
What to expect:

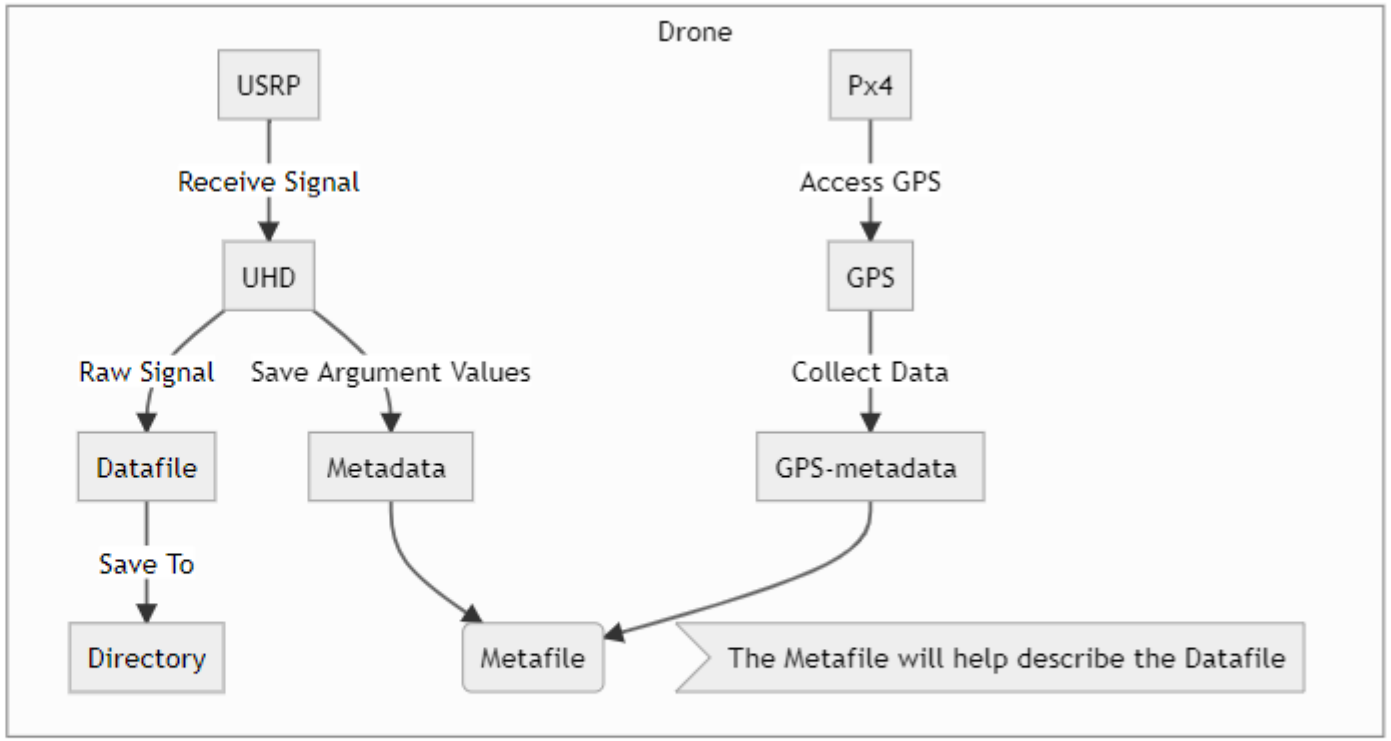
- directory: Where the wanted file to be parsed is found
- filename: Name of the wanted parsed file
- output: Path to where the log file that contains the messages
- **Note: The wanted file must be already created before the script is ran (touch name.log) in /tmp**
- More information Here

Unary RPC Diagram (GRPC)



Export Messages to Client – Future Development





5G Reconnaissance Deployment Diagram

Questions?