Open-Source Prototyping of 5G Wireless Systems for Unmanned Aerial Vehicles

FINAL REPORT

Team 13

Prof. Hongwei Zhang

Daniel Shaw, Ryan Ingram, Suraj Karn, Tyler Hutchinson

sdmay21-13@iastate.edu

https://sdmay21-13.sd.ece.iastate.edu

Executive Summary

Development Standards & Practices Used

- 3rd Generation Partnership Project (3GPP) Specifications
- Small Cell Forum FAPI and nFAPI Specifications
- IEEE 802 family of standards
- Open-source software utilization
    - Open Air Interface (OAI)
        - Cisco Open-nFAPI
    - New Paparazzi Simulator (NPS)
        - JSBsim Flight Dynamics Model (FDM)

Summary of Requirements

- Network simulator should support a multi-cell environment with many base stations (eNB) and many user equipments (UE)
- Integrated simulator should be able to successfully simulate communication among UAVs on a 5G wireless system
- Simulation must be easy to run
- Simulator should be open-source and free to use among the research community

Applicable Courses from Iowa State University Curriculum

- ENGL 314 - Helped in the creation of reports, checking for grammatical errors, and outlining of reports
- SE/COM S 309 - Early project experience helped facilitate working and operating within a team
- COM S 352/CPRE 308 - Understanding of operating systems necessary for navigating Linux
- CPRE 489 - Computer Networking and Data Communications helped in understanding client server programming, data and routing protocols and local area networks.

New Skills/Knowledge acquired that was not taught in courses

- 5G/4G infrastructure
- UAV/drone simulation
- Linux (Ubuntu) installation and kernel configuration
- Virtual machine network configuration

Table of Contents

1 Introduction 6

1.1 Acknowledgement 6

1.2 Problem and Project Statement 6

1.3 Operational Environment 6

1.4 Requirements 6

1.5 Intended Users and Uses 7

1.6 Assumptions and Limitations 7

2 Project Plan 8

2.1 Task Decomposition 8

2.2 Risks And Risk Management/Mitigation 8

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria 8

2.4 Project Timeline/Schedule 9

2.5 Project Tracking Procedures 12

2.6 Personnel Effort Requirements 12

2.7 Other Resource Requirements 13

2.8 Financial Requirements 14

3 Design 14

3.1 Previous Work And Literature 14

3.2 Design Thinking 14

3.3 Proposed Design 15

3.4 Technology Considerations 16

3.5 Design Analysis 16

3.6 Development Process 17

3.7 Design Plan 17

4 Testing 18

4.1 Unit Testing 18

4.2 Interface Testing 18

4.3 Acceptance Testing 18

4.4 Results 18

5 Implementation 19

6 Closing Material 21

6.1 Conclusion 21

6.2 References 21

6.3 Appendix I 22

6.3.1 OpenAirInterface 22
6.3.1.1 L2NFAPI_NOS1 22
6.3.1.2 Basic Simulator 23
6.3.1.3 T Tracer 23
6.3.1.4 Other Simulators 24
6.3.2 PaparazziUAV 24
6.3.2.1 Paparazzi Center 24
6.3.2.2 Configuration 24
6.3.2.3 Ground Control Station 25
6.3.2.4 Integration 25

List of figures/tables/symbols/definitions

- Figure 1 - High level sketch of project
- Figure 2 - Assumptions and limitations table
- Figure 3 - Project timeline
- Figure 4 - Gantt chart
- Figure 5 - Personnel effort table
- Figure 6 - Paparazzi and OAI integration scheme
- Figure 7 - Basic Simulator Structure and Components
- Figure 8 - L2 Simulator Structure and Components
- Figure 9 - OpenAirInterface simulation screenshot
- Figure 10 - Paparazzi Flight Simulation
- UAV - Unmanned Aerial Vehicle
- GCS - Ground Control Station
- OAI - OpenAirInterface (5G/LTE simulator)

- NPS - New Paparazzi Simulator
- C-A2X - Cellular to Aerial Vehicle to Everything Communication
- C2A - Cellular to Aerial Vehicle Communication
- A2A - Aerial to Aerial Vehicle Communication
- UE - User Equipment
- eNB - eNodeB, Base Station
- PDCP - Packet Data Convergence Protocol
- RLC - Radio Link Control
- MAC - Media Access Control
- PHY - Physical Layer
- nFAPI - Network Functional Application Platform Interface

1 Introduction

1.1 Acknowledgement

We thank Professor Zhang for all the assistance in the guidance of this project, as well as the provision of other sources of information for 5G Infrastructure, Drone Simulation, OAI setup, and remote access to computers. We also thank the OAI community with the help that has been provided throughout the project.

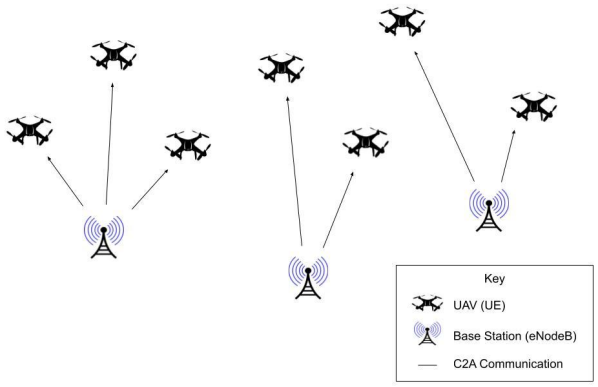


Figure 1: High level sketch of project

1.2 Problem and Project Statement

Networking is a constantly evolving field and with the newest fifth generation technology there is a lot of room for expansion. Specifically talking about unmanned aerial vehicles (UAVs), 5G networks will improve the issues of current 4G implementations such as latency and capacity limitations. With reduced latency and ultra-reliable connection, drones will make huge advancements towards true autonomy and can serve in applications such as drone delivery which rely on functionality beyond the line of sight. Our group plans to use open-source projects to simulate such application scenarios. To do so we plan to integrate a 5G network simulator with a UAV simulator to allow us to model cellular connectivity of drones in a 5G network.

1.3 Operational Environment

We set up our simulations on virtual machines running Ubuntu 16.04 as OAI and Paparazzi UAV simulators are only compatible together on Ubuntu 16.04. OAI is terminal based and is built and run on Ubuntu's terminal. Paparazzi UAV has a GUI and is much more user friendly allowing for configuration and settings to be adjusted with its control center.

1.4 Requirements

The primary functional requirement for our project is for our software simulation tool to simulate communication between UAVs in a multi-cell 5G environment. Our simulation can support the use of different networking algorithms. The simulator is open-source and free to access by others in the research community and beyond. We need it to be easy to expand upon in the future.

1.5 Intended Users and Uses

Our project is intended to be used in scholarly formats for research and extension purposes. With our project being open source it allows users to access and to build, and extended upon our project.

1.6 Assumptions and Limitations

Assumptions	Limitations
• Used by research professionals and network developers • Small scale development versus a large, ready-to-ship version	• Limited knowledge of networking • 2-semesters to finish project • Lack of previous knowledge with 5G • Open-source code for 5G simulation is still being developed

Figure 2: Assumptions and Limitations Table

2 Project Plan

2.1 Task Decompositions

For this project, we separated our tasks into three categories: OAI simulation, PaparazziUAV simulation, and algorithms research and implementation. The OAI simulation was the main component of our project as it was the simulation of 5G infrastructure. OAI involved testing different types of systems and working to get the multiple base stations/multiple user equipment drone setup working. PaparazziUAV involved running the simulation and understanding the custom functionalities to paparazzi. We also needed to work with OAI and Paparazzi for integration strategies. Finally, networking algorithms were researched and, unfortunately, not implemented into the current project, as we were not able to get to a position where we were comfortable, or able, to implement some of our algorithms.

2.2 Risks And Risk Management/Mitigation

Many risks were present in this project, especially when it came to OAI. As an open-source project, OAI is constantly evolving and documentation often becomes outdated or obsolete. This presents a risk as understanding how to run the simulation becomes challenging, especially when there are strict hardware/software requirements. To mitigate this we planned to use a scrum-based approach to stay on top of getting familiar with OAI. However, we ended up taking a different approach of using Trello as a place for storing resources for different project components and creating tasks.

Later on in the project, we discovered that successfully running OAI by the end of the project might not be feasible. To attempt to mitigate this risk we researched other network simulation software such as OMNeT++, but decided to stick with OAI because it has greater potential for future development since it can be mapped to hardware. After deciding to stick to OAI, we attempted to try running other simulations within OAI as another mitigation strategy, but were unsuccessful in running any of the other simulations such as the Basic Simulator. We also reached out to the OAI developer community to try to debug issues with running simulators.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Some key milestones for the project are as follows:

- Fall 2020:
    - Choose software Simulation/Platform
    - Refine OAI to support integrated OAI and drone simulation
    - Understand existing networking algorithms widely used in industry
- Spring 2021:
    - Simulate multiple eNB's and UE's
    - Compare different A2X Algorithms with simulation
    - Write a magazine article to summarize results (software/system development, research findings)
    - Field testing and drone delivery system

As we progress through our project, we will be measuring these test cases against ourselves, and our advisor/client. We'll also be evaluating this through three different processes: where the team will be going through any milestones hit, reviewing if it was a "success" or a "failure", then passing it onto the advisor. If the advisor clears it, the team will then review it once more with the advisor for the final yes or no. At that point, if all 3 cases are passed, the milestone will be considered a success.

2.4 Project Timeline/Schedule

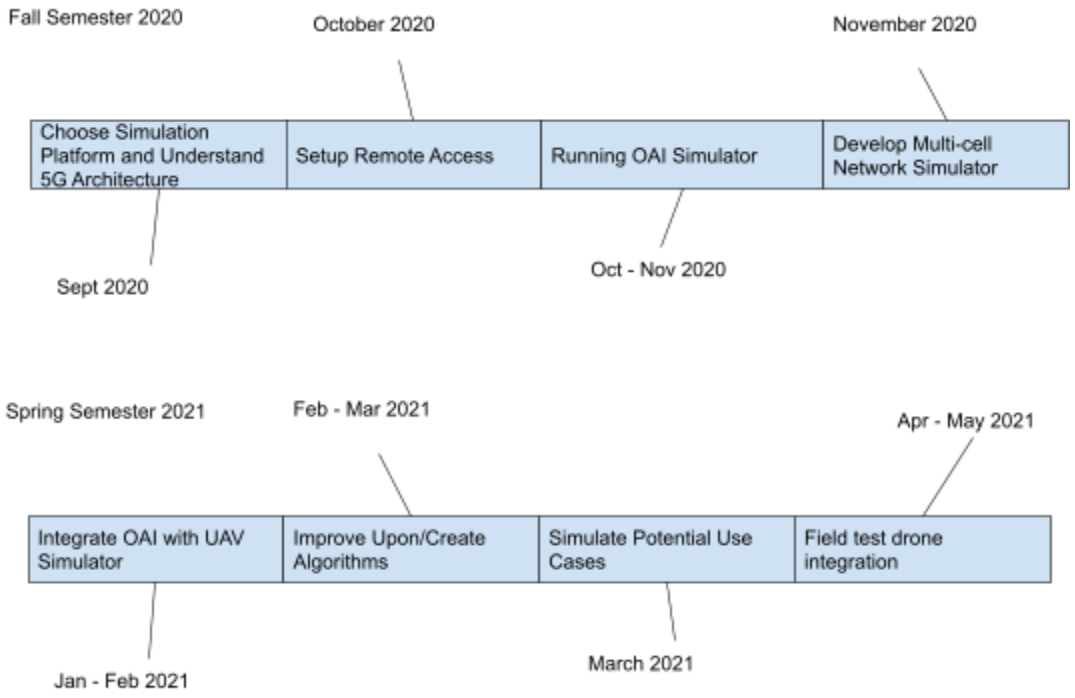


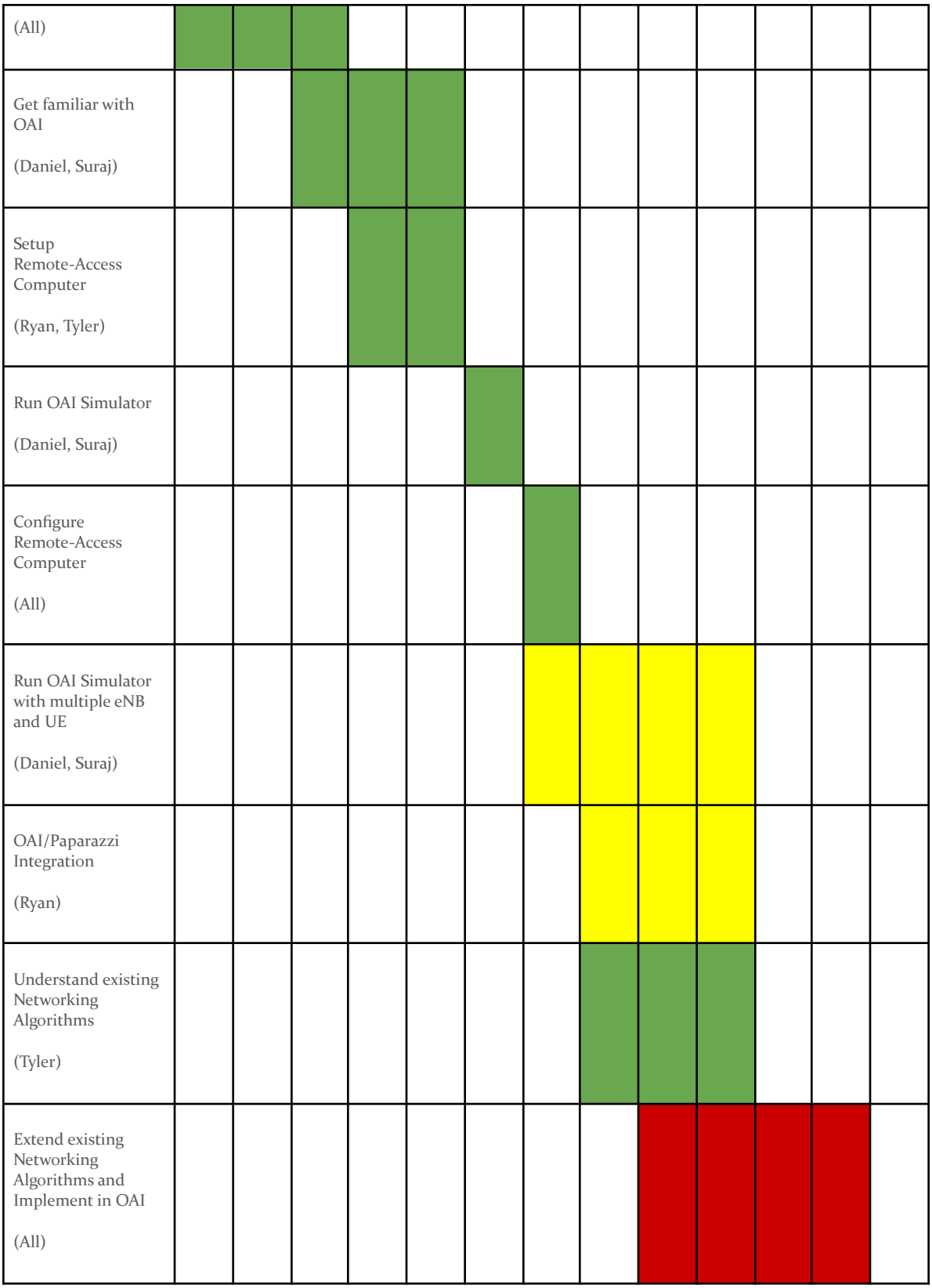
Figure 3: Project Timeline

- Choose simulation platform and understand 5G architecture
    - Experiment with different simulations and choose which best suits our project.
    - Read Wireless Networked Articles and 5G and LTE architecture concepts.
- Setup remote access
    - Gained access to a computer in Coover. Install Ubuntu 16.04.
    - Work with installing OAI and PaparazziUAV.
- Running OAI simulator
    - Setup operating system (Ubuntu) in low-latency kernel mode
    - Build the executables

    - Run the existing single-cell network simulation
- Develop multi-cell network simulator
    - Find different efficient ways to solve drone communication and path mapping using 5G network.
- Integrate OAI with UAV simulator
    - Experiment with different solutions to use OAI and Paparazzi.
    - Simulation will be able to set up 5G networks using multiple drones.
- Improve upon/create algorithms
    - Determine if we must begin the creation of our own algorithms to solve our simulation problems
    - After determination of algorithms we will use, begin the process of refining them to be used for specific purposes
- Simulate potential use cases
    - Using our simulation, be able to simulate different routes for potential use cases
    - Be able to communicate with other objects (C-A2X) using 5G.
- Field test drone integration
    - Reach out to a professor doing related work with drone systems about conducting a field test.

Color Key: Completed, In-progress, Incomplete

Task Name	8/17-8/30	8/31-9/13	9/14-9/27	9/28-10/11	10/12-10/18	10/19-11/1	11/2-11/15	1/25-2/7	2/8-2/21	2/22-3/7	3/8-3/21	3/22-4/4	4/5-4/18
Read 4G/5G Textbook (All)													
Install OAI (All)													
Install Paparazzi UAV													


(All)
Get familiar with OAI
(Daniel, Suraj)
Setup Remote-Access Computer
(Ryan, Tyler)
Run OAI Simulator
(Daniel, Suraj)
Configure Remote-Access Computer
(All)
Run OAI Simulator with multiple eNB and UE
(Daniel, Suraj)
OAI/Paparazzi Integration
(Ryan)
Understand existing Networking Algorithms
(Tyler)
Extend existing Networking Algorithms and Implement in OAI
(All)

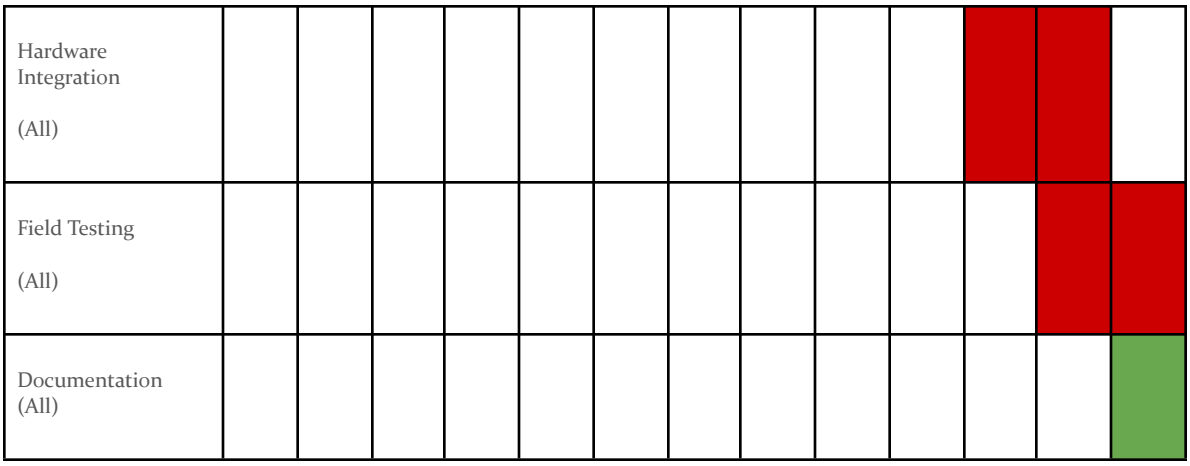


Figure 4: Gantt chart

2.5 Project Tracking Procedures

We used Trello to track our work for each section of the project. Trello was used to create meeting agendas, share resources and links, and used as a visual communication platform with our advisor. Furthermore, we used a shared Google Drive folder to compile and collaborate on the course aspect of our project. Microsoft Teams was used by the group to effectively communicate and share information.

2.6 Personnel Effort Requirements

Tasks	Hours
Research	Expect to spend about 30-40 hours on research, alone, for the project. This is because none of us have prior experience with 5G simulation, thus all of us need time to comprehend the subject. Total: 40 hours
Software Integration	This will be the bread and butter of the project. Thus, we expect to spend much of our time with the integration of simulation and development. Thus, throughout 2 semesters, we expect to spend around 5-10 hours per week on the integration.
Algorithm Design	Algorithm design is another large part and concept of our project. So, this will be very similar and hand-in-hand with our software integration. As such, we expect to spend around 5-10 hours on algorithmic design.
Field Testing	We will spend as much time as we can on this portion of the project. It is dependent on when we

	finish the project and we will spend the remaining time in the semester testing and using our software.
Documentation	Documentation will be done throughout the two semesters. We should spend roughly 2 hours each week working on scheduling, reports, and the design document.

Figure 5: Personnel Effort Table

In terms of time spent on the project our team has decided that we will individually spend roughly 5-6 hours per week on our tasks. This is divided up into categories in the table above. The beginning will mostly consist of researching topics and getting more familiar with the tools/simulations that we will be using. First step will be the research portion of the project. We expect to spend 10 hours a week on this and it will end up being around 40 hours total. After this we will begin our design. This will be done in a parallel manner spending roughly 10 hours on either the software simulation or algorithm design. We expect to complete this portion of the project by the end of the Fall 2020 semester. After we are completed we will spend the remaining time testing our simulation and possibly setting up a real field test. We will also be spending about 10 hours a week on this. Finally, we have our documentation and we expect to spend roughly 2 hours a week making sure everything is organized and up to date.

As we reached the end of our project, we were pretty spot-on for the amount of time we planned to spend on this project, outside of the time planned for our design. Since we did have some troubles getting the simulators to work the way we had expected to, we had spent a bit more time attempting to understand how to run and configure the simulators rather than integration. Overall, we were pretty spot on in our early assessments for time spent.

2.7 Other Resource Requirements

Professor Zhang, our advisor/client, has been very kind in the lending of a possible group computer to use for this project, as well as providing a sizable list of academic resources used to get a better grasp on the project and what is necessary for us to be successful.

2.8 Financial Requirements

All simulation software is open source and free to the public. We did not have any need for financial requests in this project.

3 Design

3.1 Previous Work And Literature

Previous research has been done in the field of cellular vehicle-to-everything (C-V2X) technology, where vehicles maintain a cellular connection and communicate with everything around them. 3GPP standards describe C-V2X as consisting of 4 types of communication: vehicle-to-vehicle (V2V), vehicle-to-pedestrian (V2P), vehicle-to-infrastructure (V2I), and vehicle-to-network (V2N). This communication technology would help enable full autonomy by enhancing safety features through enhanced situational awareness. Some open-source projects have already begun simulating this technology. OpenAirInterface has been working on a project with OrangeLabs to simulate LTE V2X communication using SUMO for vehicular mobility (Harri, Jerome & Villeforceix, Bernadette).

However, these vehicles are traditionally considered to be ground vehicles rather than aerial vehicles. In comparison, there is not as much research or work that has been done on cellular connected aerial vehicles since it is a newer technology, although the application of cellular connected vehicles in the air would yield similar benefits as their ground counterparts. There are few open-source projects that exist for simulating this technology which presents more challenges and opportunities for growth in the field. Similar to how OpenAirInterface is working on integrating vehicular mobility into their simulation stacks, we would like to expand this work to include aerial vehicles.

3.2 Design Thinking

Our work focuses on simulating Unmanned Aerial Vehicles (UAV's) in a 5G wireless space. Our design is shaped by the usability of our simulations and effectiveness of it. Our goal is to provide industry professionals and researchers with key insights on how OAI and UAV simulators interact with each other through our work. We also hope to expand and further develop networking algorithms to allow for better functionality of UAV's, and other electronic devices in the 5G network space. We hope to implement this by being able to integrate a chosen UAV simulator inside of OAI, in order to better simulate UAVs in a stable environment. Additionally, we also hope to implement 5G algorithms inside of the OAI and UAV simulation in order to best utilize the budding new network, while also increasing the speed at which UAVs can communicate and operate seamlessly.

3.3 Proposed Design

Our design will consist of integrating two simulators for co-simulating cellular networks and UAV mobility. This means our design will consist of two main components, the network simulator and the UAV simulator.

For our network simulator we will be using OpenAirInterface. OpenAirInterface is open-source and consists of two main projects, the Radio Access Network (RAN) and the Core Network (CN). For simplicity sake we will only be working with the RAN project, specifically the L2NFAPI_NOS1 simulator, which simulates the evolved node base station (eNB) and user equipment (UE) stacks without the need for an Evolved Packet

Core (EPC). This simulator uses Cisco's Open-nFAPI implementation of nFAPI specifications to functionally split the eNB and support multiple UE connections. We decided to use OpenAirInterface because it satisfies the open-source requirement for our project, has hardware support for future development, and supports a wide range of functionalities. However, OpenAirInterface is still developing their 5G simulators, so we will have to work with their most up-to-date LTE simulators.

As for our UAV simulator, we will be using Paparazzi. Paparazzi is an open-source drone simulator designed for simulating autonomous flight plans and includes support for multiple airframes and ground station software. Paparazzi has a graphical user interface (GUI) that can display Google Satellite images and uses an open-source flight dynamics model called JSBSim which dictates the movement of the aerial vehicle during flight. We decided to use Paparazzi because it is open-source and offers robust simulation of UAV mobility.

3.4 Technology Considerations

While open-source software offers freedom and flexibility, it also has its limitations and weaknesses. Users are often at the mercy of how well the code is documented and have to rely on forums for support. With OAI, many of the tutorials were out-dated or obsolete which made it difficult to work with. It was also difficult to find help as it was often unclear who to reach out to for questions on documentation or support.

Aside from the difficulties met with documentation, we found that OAI only supports single-cell environments. This means that to meet the requirement of simulating a multi-cell environment we would need to build upon the current functionality that OAI supports.

Another limitation of OAI is the strict hardware/software requirements to run the simulators. It is highly recommended that OAI is run on Ubuntu 16.04 with a low-latency kernel. Many of the simulators also require multiple machines to run different hosts. We were advised early in the project to run the simulation hosts on one machine, however we later learned that the Linux Kernel Policy will not allow this as it will drop locally sourced packets.

A solution to OAI could potentially be ensuring that strict hardware/software requirements are met. This likely means that the eNB and UE will need separate physical machines to run on and be connected by a physical ethernet cable. We have tried running the two hosts on separate virtual machines and were not able to verify a successful connection. The issue also could potentially be the need for an EPC which we were advised against using. The EPC will likely need its own machine to run on as well.

An alternative solution would be to use a different network simulator than OAI. One network simulator that we looked into was OMNeT++. OMNeT++ is a discrete event simulator which can be used to create network simulation models. It has already been successfully integrated into similar projects for modeling vehicular mobility, such as Veins (https://veins.car2x.org/). However, the downside of OMNeT++ is that there is no support for real hardware as there is with OAI.

3.5 Design Analysis

Unfortunately, we were unable to fully implement and test our proposed design because of early challenges met with OAI. We had originally planned to be much further in the development stage of our project by the second semester but that was curbed by complications with verifying functionality of OAI's simulators.

We were advised early on to use the L2NFAPI_NOS1 simulator on a single machine, although the documentation says to use two separate machines. Later we found out that this is because the linux kernel will not allow traffic between locally sourced IP addresses. Therefore, two separate machines are required for hosting the eNB and hosting the UE.

After learning about the linux kernel policy for locally sourced packets we wanted to test this with virtual machines. So, we created two separate virtual machines and configured them to be on the same network. One virtual machine hosted the eNB and the other hosted the UE however we were still unable to verify connection by pinging the eNB from the UE interface and vice versa.

Another thought that we had was that since the L2NFAPI_NOS1 documentation is only under the develop branch, maybe it is not finished or not fully supported yet. Perhaps a separate machine would be needed for hosting the EPC as specified in the L2NFAPI documentation under the master branch, which uses the S1 interface to communicate between the eNB and EPC. We also received a suggestion by a EURECOM developer to follow a tutorial on how to run the rf simulator, which may be better supported than the L2NFAPI simulator.

3.6 Development Process

For this project, we used Trello to organize our tasks. In our Trello board, we had bins for OAI, UAV simulation, integration, networking algorithm, and meeting notes. We were originally going to take an Agile approach with biweekly sprints, however we found that it was easier to keep track of links and resources this way. In each of our bins, we have cards for tasks that need to be done and the deadline associated with that task along with any helpful resources found along the way. This was also a nice way of creating meeting agendas and meeting minutes all in one place.

As for communication, our team uses Microsoft Teams to organize and hold virtual meetings. We usually meet at 3 pm on Wednesday of each week and update the team on any progress we have made on tasks and develop new tasks to accomplish for next week. For our advisor meetings, we use zoom and create meeting agendas in Trello.

3.7 Design Plan

To integrate these two simulators, we plan to call Paparazzi functions for getting the location coordinates of the drones and base stations during OAI's simulation and assigning the location data to the appropriate eNB and UE storage locations. For the L2NFAPI_NOS1 simulator, the place where location coordinates of the eNB and UE appear to be stored in the virtual network function (VNF) and physical network function (PNF). The VNF lies in the eNB host and the PNF lies in the UE host. Once this interaction is established we will be able to simulate cellular connection between drones and base stations.

Figure 6, shows the relationship between the Paparazzi simulator and the OAI simulator.

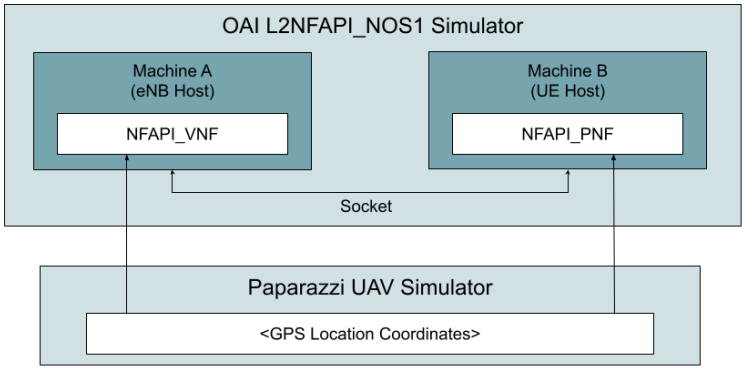


Figure 6: Paparazzi and OAI integration scheme

4 Testing

Testing for our project was split into the OAI and Paparazzi Simulator components of our project. Functional testing was done by running simulations and making sure that our components had the desired output. The non-functional tests were performed by making sure that the platform and builds required to run the simulations were functional before running our simulations. We had hoped to use the divide and conquer strategy to run simulations on OAI and Paparazzi independently and run tests to meet our standards before integrating these components.

4.1 Unit Testing

Unit Testing was primarily done on the OAI simulator as configuration files needed to be set before simulations were performed. Various configuration files needed to be edited and built properly before they could run on our machines. The configuration file setups make the connection between the eNB and UE possible. Our hope was to combine the OAI and Paparazzi components together by transmitting location data points from Paparazzi in real time and then using it to simulate UAV in 5G network space.

Again, we were unable to get the two programs configured and combined for our final simulation to test and ensure that we had properly configured the simulation. Thankfully, we were able to thoroughly setup and configure OAI and Paparazzi separately, so that in the future, combination between the two would be much easier and seamless.

4.2 Interface Testing

With the split of our project interface testing for OAI is done with the successful connection of eNB and UE components. The Basic and L2nFAPI_NOS1 simulators rely on the proper system setup and configuration to make eNB and UE connection possible. Per the simulator documentation the eNB and UE connection is confirmed by a ping test. Pings are sent and received from the newly generated eNB and UE's. Upon the successful completion of the ping test the integration of OAI to paparazzi was to take place so a UAV could be simulated on a 5G network space using location data points.

4.3 Acceptance Testing

Acceptance testing was to be done once the OAI and Paparazzi components of our project functioned independently. Passing the ping test would allow OAI to meet its independent acceptance testing requirements. Paparazzi met its acceptance testing requirements by simulating various flight patterns. The

main criteria for acceptance testing was to combine OAI and Paparazzi simulators and use location data points for the UAV simulation.

4.4 Results

Ultimately, we were not able to test as much, or as in in-depth, as we had wanted to in the project. Due to the difficulties we had with OAI and getting a connection between two nodes of the simulation. We had difficulties moving past this component because the OAI simulator was an integral part of our project, and not much could be done without the eNB and UE connection.

At the end of our project's lifespan, we had found that the graduate students, who previously worked on this project with Dr. Zhang understood and sympathized with the problems that we had faced in the project, as they had faced a similar battle with the OAI simulator. Instead of having a final end product to show Dr. Zhang and to show off for others, we came up with a way to identify the trials and tribulations that we faced in our project. This will be documented and given to Dr. Zhang so that other students, in future groups, will be able to quickly identify and ideally remove any obstacles that they may run into while figuring out and going through this project.

5 Implementation

The implementation of our project was to be done by integrating the OAI with the Paparazzi simulator. Location data points were to be used to combine these simulators and would allow our team to view the UAVs move and communicate in network space. We hoped to simulate eNB and UE using the Basic Simulator or L2 simulators. We intended to use the L2 simulator as directed by our advisor but shifted towards the Basic simulator as the L2 simulator connection could not be established and it did not pass the ping test. Our focus then shifted to getting the Basic simulator running, however, this also proved difficult to do and provided us with challenges of its own.

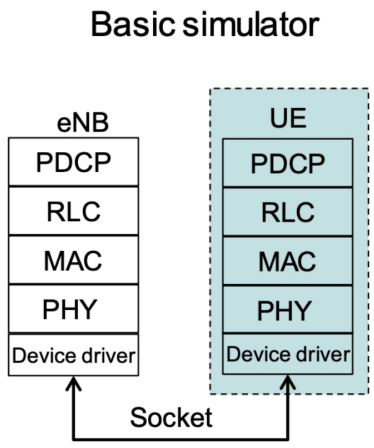


Figure 7: Basic Simulator Structure and Components

The Basic simulator allows for the eNB and UE connection through network interface. To build the Basic simulator the LTE and NAS Tools configuration files need to be set. Then the simulator is run to connect the one eNB with one UE.

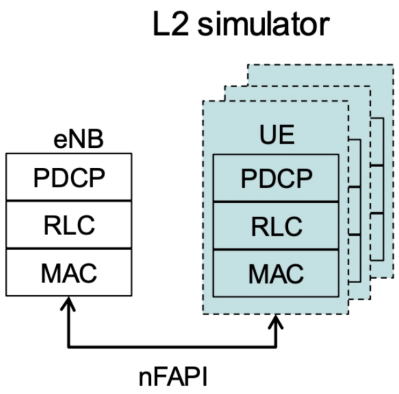


Figure 8: L2 Simulator Structure and Components

The eNB and UE components of the L2 simulator are connected by the nFAPI interface. The nFAPI is used in the 5G network to extend the functional split between the MAC and Physical layers. The L2 simulator needs to be run on two different machines one functions as the eNB and the other as the UE component. Upon the correct configuration and build of the eNB and UE files a connection is established between them.

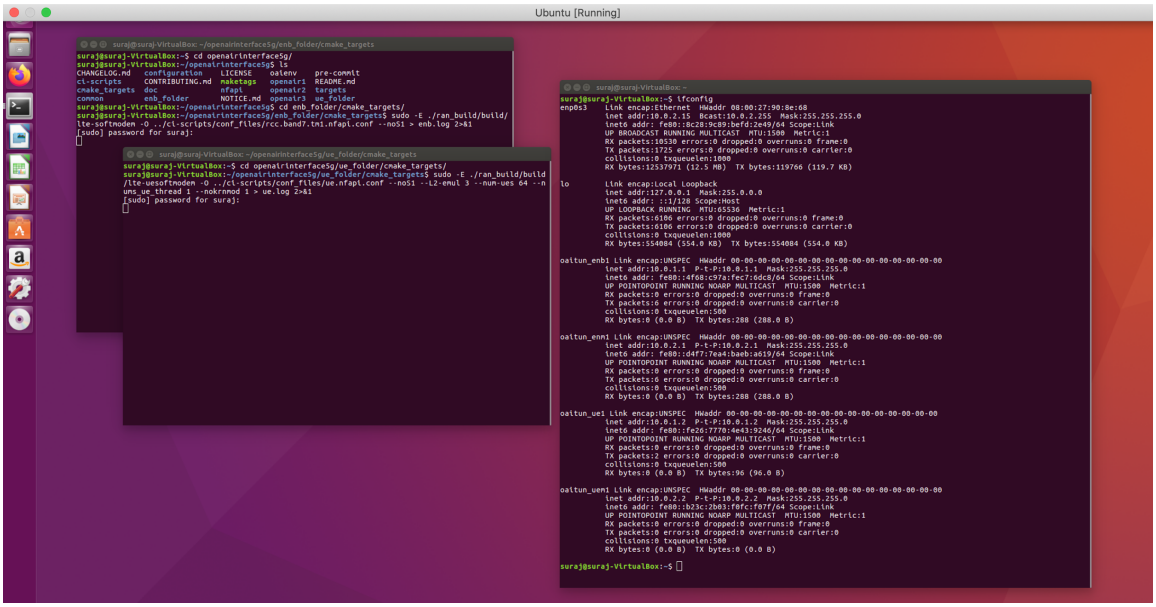


Figure 9: OpenAirInterface simulation screenshot

Figure 9 shows the L2 simulator simulation of one eNB and UE on the Ubuntu console.

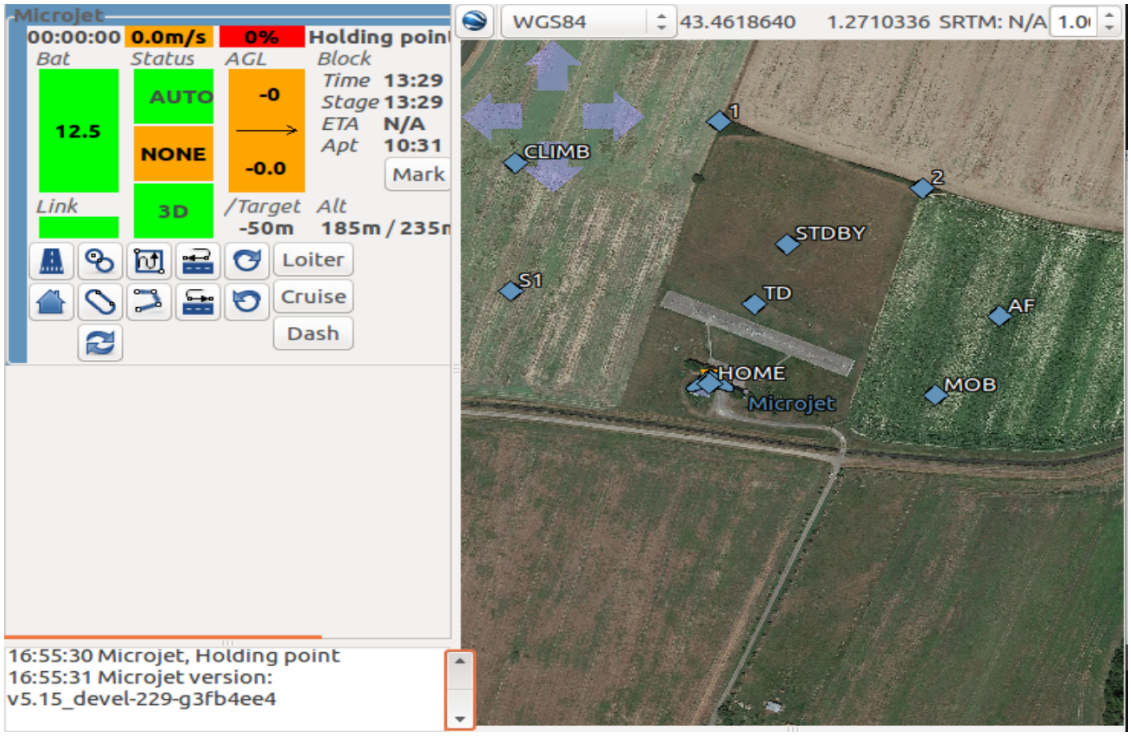


Figure 10: Paparazzi Flight Simulation

Various flight patterns with different configurations were simulated using the Paparazzi UAV simulator.

6 Closing Materials

6.1 Conclusion

Despite not being able to overcome challenges with running OpenAirInterface, we feel that we have learned a lot in the process. We now have a better understanding of 5G/LTE cellular network components and were able to analyze our problems with running the simulator and make conclusions as to why they might not be working. We also got the chance to engage with the 5G community by interacting with some members of the OAI community when trying to troubleshoot our issues. Although we were not able to finish our project, we hope to pass our knowledge on to future teams with our detailed project documentation. We hope that the future teams taking up this project will be able to look at our work and have a better idea of starting and getting OAI up quickly.

6.2 References

1) C. Li, H. Zhang, J. Rao, L. Y. Wang and G. Yin, “Cyber-Physical Scheduling for Predictable Reliability of Inter-Vehicle Communications,” *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, Orlando, FL, 2018, pp. 267-27

2) Y. Chen, H. Zhang, N. Fisher, L. Y. Wang and G. Yin, "Probabilistic Per-Packet Real-Time Guarantees for Wireless Networked Sensing and Control," in *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2133-2145, May 2018.

3) H. Zhang *et al.*, "Scheduling With Predictable Link Reliability for Wireless Networked Control," in *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 6135-6150, Sept. 2017.

4) Y. (Ed.). (2020, August 28). Full LTE architecture and components. Retrieved November 10, 2020, from https://yatebts.com/documentation/concepts/lte-concepts/

5) "C-V2X: Cellular Vehicle-to-Everything." *Qualcomm*, 14 Apr. 2021, www.qualcomm.com/products/automotive/c-v2x.

6) Harri, Jerome & Villeforceix, Bernadette. "LTE V2X Communication – Scenario and OAI Roadmap". 2017. PDF File. https://www.openairinterface.org/docs/workshop/4_OAI_Workshop_20171107/Talks/HAERRI_V2X_OrangeLabs_OAI_Workshop_release.pdf

7) Figure 7: Basic Simulator Structure and Components
Kaltenberger, Florian. "OpenAirInterface 5G Overview, Installation, Usage" 209. PDF File
https://www.openairinterface.org/docs/workshop/8_Fall2019Workshop-Beijing/Training/2019-12-03-KALTENBERGER-1.pdf

8) Figure 8: L2 Simulator Structure and Components
Kaltenberger, Florian. "OpenAirInterface 5G Overview, Installation, Usage" 209. PDF File
https://www.openairinterface.org/docs/workshop/8_Fall2019Workshop-Beijing/Training/2019-12-03-KALTENBERGER-1.pdf

6.3 Appendix I - "Operation Manual"

6.3.1 OpenAirInterface

Open Air Interface (OAI) is an open source software developed by OpenAirInterface Software Alliance (OSA). OSA developed OAI to gather a community of developers from all over the world to build Radio Access Networks (RAN) and Core Network (CN) technologies for wireless cellular networks.

6.3.1.1 L2NFAPI_NOS1

These instructions follow the documentation for L2NFAPI_NOS1 found in the doc directory on the Develop branch. First we will discuss the software/hardware requirements for running the simulator and then instructions on how to build and run the simulator.

For this simulator, two machines are needed as pictured in figure x. One machine will be dedicated to hosting the eNB and the other machine will be dedicated to hosting the UE. It is important that these are hosted on separate machines. These two components will be passing traffic between each other and if they are on the same machine, the linux kernel will likely drop the packets. See this link for more details: https://github.com/torvalds/linux/blob/29dcea88779c856c7dc92040a0c012332631o1d4/net/ipv4/route.c#L1881-L1885

It is advised that these be real machines as functionality may not be guaranteed with virtual machines. If you do choose to use virtual machines ensure that enough CPUs processing is available and a low latency kernel is installed. Machines must have an intel processor with power management features in the BIOS and CPU frequency scaling removed

Once these machines have been acquired, Ubuntu 16.04 LTE must be installed on both and it is recommended to use a low-latency linux kernel.

After Ubuntu is installed on both machines, it is now time to clone the repositories from gitlab. Pick one machine to be the eNB host and run the following commands:

```
$ ssh sudousername@machineB
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git enb_folder
cd enb_folder
git checkout develop
```

On the other machine, run these commands:

```
$ ssh sudousername@machineC
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git ue_folder
cd ue_folder
git checkout develop
```

Now that you have the repository cloned on each machine, it's time to configure the simulators. In the doc folder, you will find a file called "L2NFAPI_NOS1.md" which will contain the full instructions on how to configure the files. You will need to replace the remote and local addresses in the configuration files to match the addresses of the machines you are using. Since this is the no S1 version of the simulator, we assumed that the configurations for the S1 interface can be left alone.

After configuring the files, you will need to build the softmodem executables before you can run the simulation. The instructions for this can be found in the "BUILD.md" file under the doc directory. Essentially, if it is your first time building the modems you will need to use the -I flag to install prerequisites, otherwise this is not required. Use the --eNB and --UE flags on the corresponding host machines to build the respective modems.

Finally, once the files have been configured and the build has successfully finished you can try running the simulator. Follow the instructions listed in the "L2NFAPI_NOS1.md" file for running the eNB and UE executables. You will first run the eNB program and then the UE program. Once both are running you can open a separate terminal to issue the ping tests listed in the instructions.

Observe the eNB and UE log files when debugging. Our team ran into issues with only 1 UE device showing up and were unable to successfully run the ping test listed at the end of the tutorial. Based on our eNB and UE log files it appears that we may have had some timing issues relating to our use of virtual machines.

6.3.1.2 Basic Simulator

Running the Basic Simulator is much more straightforward. For this simulator, we will be following the instructions found in the "BASIC_SIM.md" file under the doc directory on the Master branch.

The Basic Simulator consists of a single UE connected to a single eNB. The instructions appear to indicate that the UE and eNB are hosted on the same machine and do not specify any files to configure when running in noS1 mode.

To run the Basic Simulator we have been following the instructions to build the eNB and UE modems as follows:

```
$ source oaienv
$ cd cmake_targets
```

```
$ ./build_oai --eNB --UE
```

After building the modems and running the using the commands for starting the eNB and UE in the instructions we were still unable to run a successful ping test.

We reached out to members of the OAI community to see what could be going wrong and were told that the Basic Simulator needs to be run on different machines due to the Linux Kernel dropping packets from locally sourced addresses.

For future teams trying to run the Basic Simulator, or the L2 simulator for that matter, we would recommend trying to set up two different machines for the UE and eNB and trying to use an EPC. Our team seemed to face a lot of challenges when trying to simulate on a single machine and without an EPC.

6.3.1.3 T Tracer

The T tracer is a graphical user interface for the eNB that shows data on what UE devices are connected to it.

To run the T tracer, first you need to run make in "openairinterface5g/common/utils/T/tracer". This will build the an executable called "enb" in the tracer directory to launch the T tracer server.

Now that you have the executable you can run the T tracer server by issuing the following command:

```
./enb -d ../T_messages.txt in openairinterface5g/common/utils/T/tracer
```

Then, when you run the lte-softmodem, you will need to append the flag: `--T_stdout 0`.

6.3.1.4 Other Simulators

Our team also attempted to run the L1 simulator and the rf simulator.

When trying to run the L1 simulator we faced an issue with a missing configuration file (/openairinterface5g/ci-scripts/conf_files/rcc.band7.nos1.simulator.conf doesn't exist) and came to the conclusion that the simulator may not be supported anymore.

As for the rf simulator, we found that the instructions for this were kind of hidden in the repository and ambiguous. The instructions can be found in "../targets/ARCH/rfsimulator/README.md". We were not able to make much progress running this simulator as the configuration instructions were not clear.

6.3.2 PaparazziUAV

Paparazzi is the open-source drone simulation that we used. It can be used with hardware or software however, our project was completely software based. Paparazzi has a lot of tutorials and setup information on their wiki page at https://wiki.paparazziuav.org/wiki/Main_Page. We will go over the main components of Paparazzi here.

6.3.2.1 Paparazzi Center

Paparazzi Center is the main graphical user interface of the flight simulator. It contains three major parts that consist of configuration, control panel, and a page for GCS to be embedded. There is also a console for displaying information about what you are doing. To start the paparazzi center simply run ./paparazzi from the root of your installation.

The configuration section is a part of the center where you can choose different configuration files that you want to use in your simulation. A new aircraft or new flight plans can be created and selected here along with extra things such as general settings, radio, and telemetry. The control panel is where you can select your target to build and execute your session. You can also add different tools and agents to your build such as the GCS to use in the simulation.

6.3.2.2 Configuration

With paparazzi there are several XML configuration files you need for simulation. These include the airframe, flight plan, settings, radio, and telemetry files. For our project we mostly only looked at the airframe and flight plan files.

Airframe configuration files are a major part of the simulation especially if you are wanting to incorporate hardware to the project. With just software this is where you can define the type of aircraft you want to use along with the type of simulation. You can also define different commands and command laws. For example the default commands for a simple fixed-wing aircraft are Throttle, Roll, and Pitch. This file is also where you can customize anything extra you want in the GCS for your specific aircraft.

Flight plan configuration files are important for how you want your aircraft to travel. Flight plans consist of two major components: waypoints and blocks. Waypoints are geographic locations that are used to specify trajectories in flights/maneuvers. Waypoints are specified by the name and coordinates. Blocks describe each unit of the mission. If autopilot is used blocks are followed in order. The basic idea of blocks is to give commands to the aircraft such as hold, go, circle, follow, etc. Waypoints are involved in each of these blocks. A basic example would be “circle waypoint A”. You can customize these blocks for different flight patterns that you need.

6.3.2.3 Ground Control Station

Once the simulation has been started the GCS interface will be opened. This is where you can monitor and control your aircraft. This is also a good place to visually edit your flight plan and waypoints. From here you can visualize everything that is going on. You have a map supplied by Google Earth and can see how your aircraft is moving. You can also see all of your waypoints and how everything is laid out. On the left hand side you can see a variety of different information about the aircraft such as altitude, position, etc. On the bottom of the interface you can see your flight plan layout. To choose different actions (blocks) you can double click that action.

6.3.2.4 Integration

For integration with OAI we needed to be able to share the location points. To do this we can use several methods located in the state.c file in the paparazzi project. There are methods to get the position in enu, lla, utm, ecef, and ned coordinates. We started experimenting with these methods towards the end of the project.