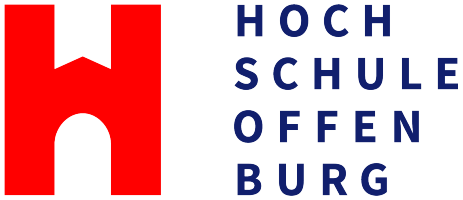

Integrating Raspberry Pi with UAV for GPS and LiteVNA data

Laahiri Adusumilli

INTERNSHIP REPORT

Course of Studies: Electronics and Communication Engineering
Indian Institute of Information Technology Kottayam

Institute for Unmanned Aerial Systems
Offenburg University

29.07.2024

Supervisor
Prof. Dr. -Ing. Marlene Harter, Hochschule Offenburg
Prof. Dr. Tobias Kreilos, Hochschule Offenburg

Adusumilli, Laahiri:
Integrating Raspberry Pi with UAV for GPS and LiteVNA data / Laahiri Adusumilli. –
INTERNSHIP REPORT, Offenburg: Offenburg University, 2024. 14 pages.

Adusumilli, Laahiri:
Integration von Raspberry Pi mit UAV für GPS und LiteVNA-Daten / Laahiri Adusumilli.
–
PRAKTIKUMSBERICHT, Offenburg: Hochschule Offenburg, 2024. 14 Seiten.

Vorwort

This report represents the work undertaken during an internship funded by the DAAD (Deutscher Akademischer Austauschdienst), through its WISE (Working Internships in Science and Engineering) program. I am deeply grateful for the opportunity provided by DAAD, which has significantly contributed to my professional and academic growth.

I would like to extend my heartfelt gratitude to Prof. Dr.-Ing. Marlene Harter for accepting me as an intern at Institute for Unmanned Aerial Systems (IUAS), Hochschule Offenburg and for her invaluable guidance throughout the internship.

I also wish to express my sincere thanks to Prof. Dr. Tobias Kreilos for his insightful guidance and support, which have been crucial to the completion of this project.

Additionally, I am thankful to Mr. Sadanand Milind Gavde for his work with LiteVNA, which has been an integral part of this project.

—...—

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich diesen Praktikumsbericht selbständig und ohne fremde Hilfe erstellt habe selbständig und ohne unerlaubte fremde Hilfe verfasst habe, insbesondere, dass ich alle Stellen, die wörtlich oder sinngemäß oder gedanklich aus Veröffentlichungen, unveröffentlichten Dokumenten und Gesprächen entnommen sind, an den entsprechenden Stellen der Arbeit durch Zitate als solche gekennzeichnet habe.

Ich bin mir bewusst dass eine falsche Angabe rechtliche Konsequenzen nach sich ziehen wird. Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d.h. dass die Arbeit elektronisch gespeichert, in andere Formate umgewandelt, auf den Servern der Hochschule Offenburg öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Offenburg, 29.07.2024

Laahiri Adusumilli

Zusammenfassung

Integration von Raspberry Pi mit UAV für GPS und LiteVNA-Daten

Ziel dieser Arbeit ist es, einen Raspberry Pi in die Drohne zu integrieren und Verbindungen zwischen Raspberry Pi und LiteVNA sowie zwischen Raspberry Pi und dem GPS-Modul der Drohne herzustellen, um mit Hilfe eines Python-Skripts gleichzeitig LiteVNA-Daten zu sammeln und GPS-Daten aufzuzeichnen, wann immer dies durch einen einzigen Befehl von der Bodenstation aus erforderlich ist, und die Daten für jeden Sweep in verschiedenen Dateien zu speichern, um sie nach dem Flug offline weiter zu verarbeiten. Dazu gehören auch Überlegungen zur Platzierung der Antenne auf der Drohne.

Das Projekt sieht vor, 2 Vivaldi-Antennen zusammen mit einem LiteVNA auf einem UAV zu platzieren und Messungen an verschiedenen Punkten während des Fluges durchzuführen. Dazu sollte das LiteVNA im USB-Modus betrieben und von einem System gesteuert werden, da sonst bei jedem Sweep die Daten überschrieben werden. Zu diesem Zweck wird ein Raspberry Pi zur Flugzeitdatenspeicherung eingesetzt. Außerdem sollten die LiteVNA-Sweepdaten sowohl mit einem Zeitstempel versehen sein als auch GPS-Positionsdaten enthalten. Um die GPS-Daten vom GPS-Modul der Drohne zu erhalten, wird eine USB-Verbindung zwischen Raspberry Pi und GPS-Modul verwendet.

Abstract

Integrating Raspberry Pi with UAV for GPS and LiteVNA data

The aim of this work is to integrate a Raspberry Pi into the UAV and establish connections between the Raspberry Pi and LiteVNA, as well as between the Raspberry Pi and the UAV's GPS module. This integration will enable the collection of both LiteVNA data and GPS logs simultaneously using a Python script, triggered by a single command from the ground station. The collected data will be saved for each sweep in separate files for offline processing after the flight. This also includes considerations for antenna placement on the UAV.

The project setup involves mounting two Vivaldi antennas along with a LiteVNA on a UAV and taking measurements at several points during the flight. To facilitate this, the LiteVNA should operate in USB mode and be controlled by a system; otherwise, the sweep data would be overwritten with each new sweep. A Raspberry Pi is used for storing flight time data. Furthermore, the LiteVNA sweep data should be both timestamped and contain GPS position data. To obtain the GPS data from the UAV's GPS module, a USB connection between the Raspberry Pi and the GPS module is used.

Contents

1. Raspberry Pi

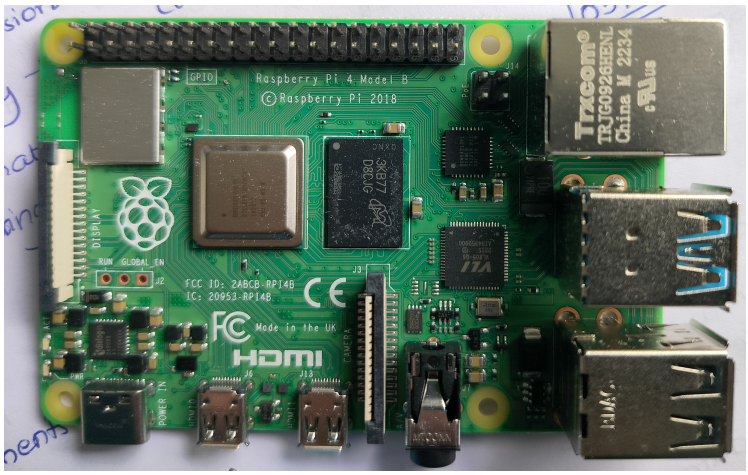

Figure 1.1: Raspberry Pi 4 B

Table 1.1: Raspberry Pi 4 Model B Specifications (Part 1)

Specification	**Details**
Processor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory	1GB, 2GB, 4GB or 8GB LPDDR4 (depending on model) with on-die ECC
Connectivity	• 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE • Gigabit Ethernet • 2 × USB 3.0 ports • 2 × USB 2.0 ports

Table 1.2: Raspberry Pi 4 Model B Specifications (Part 2)

Specification	Details
GPIO	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
Video & Sound	• 2 × micro HDMI ports (up to 4Kp60 supported) • 2-lane MIPI DSI display port • 2-lane MIPI CSI camera port • 4-pole stereo audio and composite video port
Multimedia	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
SD Card Support	Micro SD card slot for loading operating system and data storage
Input Power	• 5V DC via USB-C connector (minimum 3A) • 5V DC via GPIO header (minimum 3A) • Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment	Operating temperature 0–50ºC
Production Lifetime	Raspberry Pi 4 Model B will remain in production until at least January 2034.

1.1 Configuration of Raspberry Pi

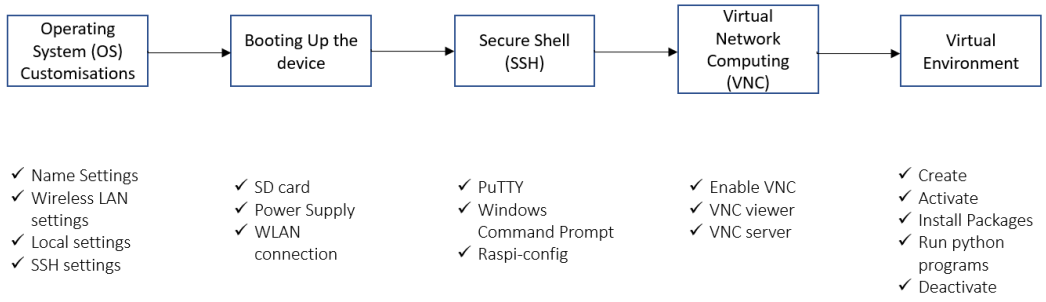

Figure 1.2: Raspberry Pi configuration

From 1.2 steps involved in configuration of the Raspberry Pi can be seen. The model used for this project is Raspberry Pi 4 B.

1.1.1 OS Customisations

For flashing OS onto the microSD card, using Raspberry Pi imager is the easiest way. The type of the Raspberry Pi (4 B in this case), the storage device (microSD card) and the type of OS should be chosen from the options on the start page of Raspberry Pi imager. From the imager, one of the supported OS or any custom OS in the format .img can be flashed onto a microSD card. The recommended OS is Raspberry Pi (64-bit) which is being used in this project. The name settings include hostname, username and password. These settings are important in future for accessing the terminal or graphical user interface of Raspberry Pi. The wireless LAN configuration includes SSID and password, where the wifi network name and password are to be entered. SSH should be enabled before flashing the OS.

1.1.2 Booting Up the device

After the OS is written on the SD card, it can be inserted into the slot on Pi and the device is booted up by connecting it to power supply. The red LED indicates that the Pi is powered and the yellow LED indicates the status of wifi connectivity of the Pi. The fast blinking of yellow LED means there is no connectivity and slow blinking means the device is connected to the network. The first time powering of the device will take longer time than subsequent usage.

1.1.3 SSH

SSH (Secure SHell) provides secure access to a terminal session on the Raspberry Pi. We can SSH into Raspberry Pi using PuTTy, a free and open-source terminal emulator software or Windows command prompt can be used. For using PuTTy either the hostname or the IP address of the Raspberry Pi should be known and should be entered in the space for the details. Then a terminal window opens up asking for username and password to login to the Raspberry Pi and we can use the terminal.

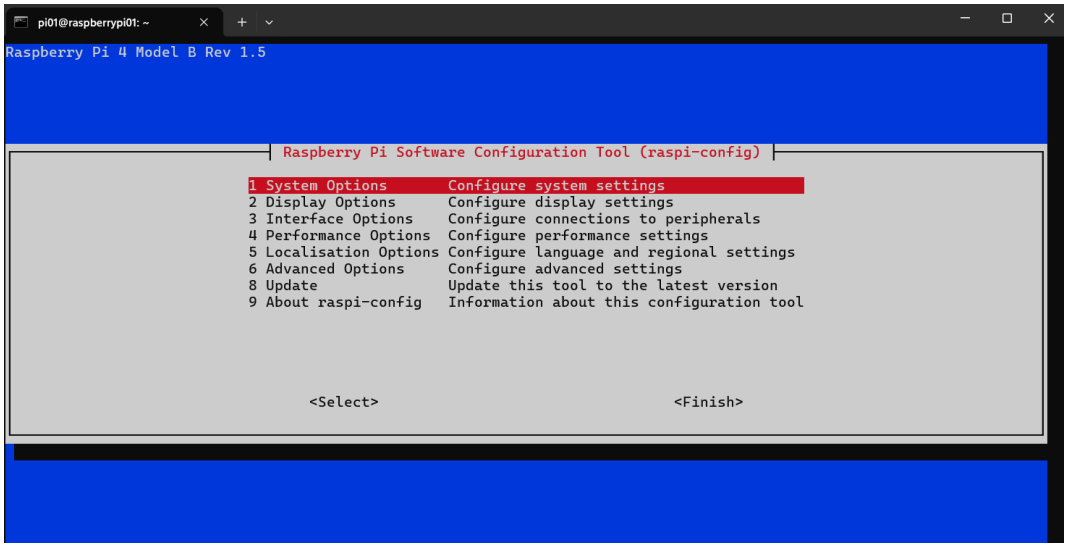

Figure 1.3: Raspberry Pi Software Configuration Tool

Using "sudo raspi-config" command in Pi terminal, Raspberry Pi Software Configuration Tool can accessed. In the interfacing options VNC can be enabled.

1.1.4 VNC

Virtual Network Computing (VNC) is a graphical desktop-sharing system that remotely controls another computer. It transmits the keyboard and mouse input from one computer to another, relaying the graphical-screen updates, over a network. [1]

- The VNC server is the program on the machine that shares some screen (and may not be related to a physical display – the server can be "headless"), and allows the client to share control of it.
- The VNC client (or viewer) is the program that represents the screen data originating from the server, receives updates from it, and presumably controls it by informing the server of collected local input.
- The VNC protocol (RFB protocol) is very simple, based on transmitting one graphic primitive from server to client ("Put a rectangle of pixel data at the specified X,Y position") and event messages from client to server.

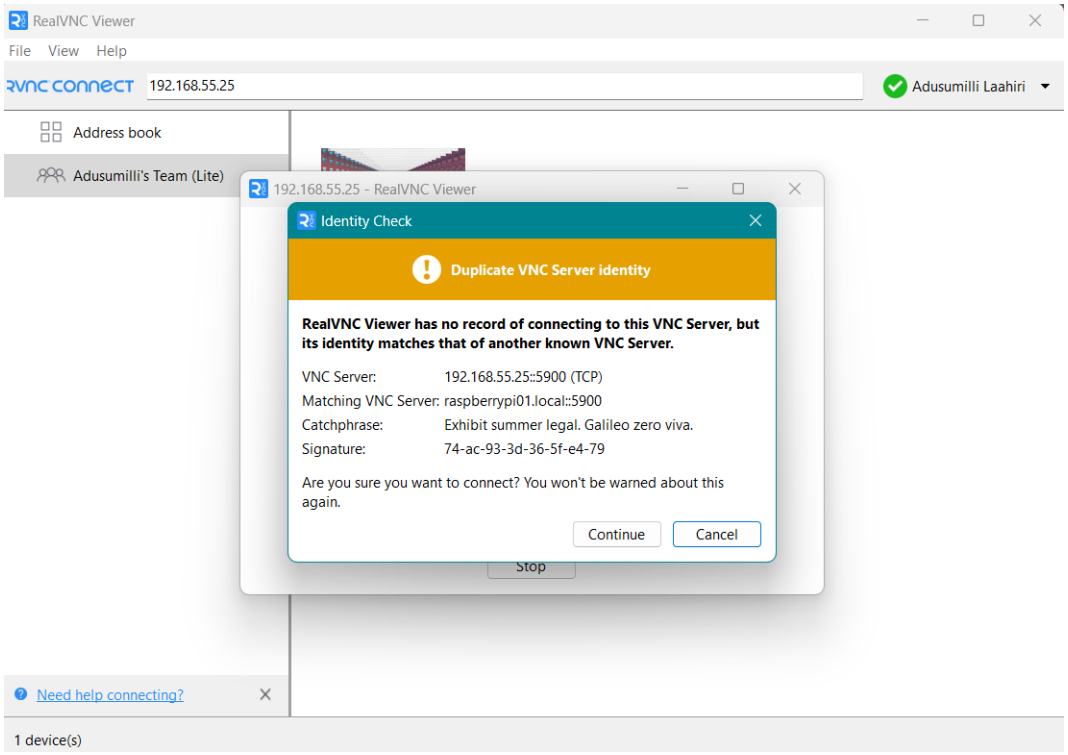

Figure 1.4: VNC viewer

1.1.5 Virtual Environment

Creating a virtual environment helps contain all the Python packages that are used in the project at the same place, making it easier to reproduce later and as Raspberry Pi is python-based OS, this helps avoid having conflicting packages breaking the OS.

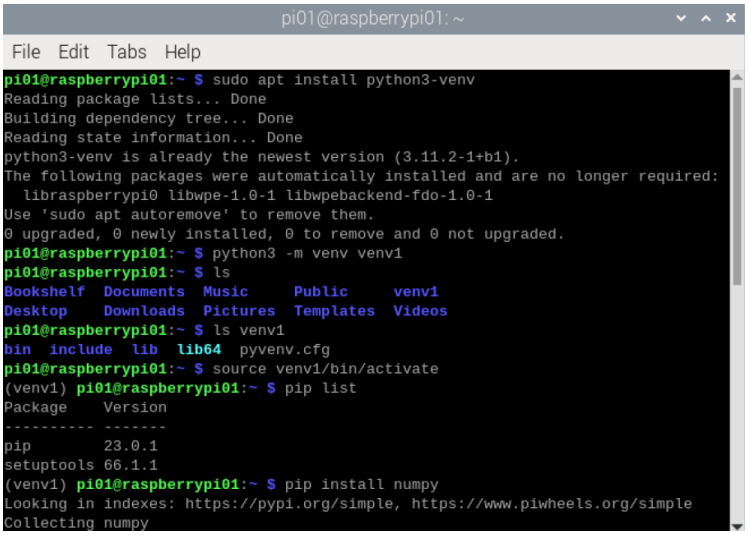

Figure 1.5: Creating and activating Virtual Environment

1.2 Raspberry Pi-LiteVNA

A Virtual Network Analyzer (VNA) is a test system that enables the RF performance of radio frequency and microwave devices to be characterised in terms of network scattering parameters, or S parameters. A VNA has a set of transmitters and receivers, through which it can send stimulus signal to the device-under-test and receive the transmissions and reflections from the DUT. A variety of different VNAs available on the market, each with a different number of ports and paths for which the stimulus signal flows. For a 2-port 1-path VNA, both the reflected and transmitted signal (S11 and S21) can be measured, however, the DUT must be physically reversed to measure the reverse parameters (S22 and S12). LiteVNA is a portable VNA with 2-port 1-path, with only port 1 having the capability of emitting a stimulus signal, hence it can measure only S11 and S21 .

Table 1.3: Parameter Specifications for LiteVNA 64

Parameter	**Specification**	**Conditions**
Frequency range	50kHz 6.3GHz	-
System dynamic range	>70dB	f < 3GHz, calibrated
	>90dB(v0.3.1 20avg)	f < 3GHz, calibrated
	>50dB	f >= 3GHz, calibrated
S11 noise floor	<-50dB	f <3GHz, calibrated
	<-40dB	f >= 3GHz, calibrated
Frequency stability	<0.5ppm	0°C - 50°C
Sweep rate	>550 points/s	Avg=1
Sweep points (on device)	21 1001 points, adjustable	-
Sweep points (USB)	1 65535 points, adjustable	-
Power supply	USB, 5V±0.5V 1A MAX	-
Battery	Li-polymer 3.7V 2000mAh	LiteVNA 64
Operation ambient temperature	-10°C 50°C	The battery performance decreases below 0 °C.
RF connectors	SMA female	-
Display	3.95”TFT LCD (480×320)	LiteVNA 64

Figure 1.6: LiteVNA in USB mode

LiteVNA is connected to Raspberry Pi using USB A to C. Using bash script, the name of the USB port on Raspberry Pi can be found. This port name is to be entered in the serial port category of the LiteVNA code.

1.3 Raspberry Pi-GPS module

MAVlink is a serial protocol used to communicate with drones. Pymavlink is a python package that implements the MAVlink protocol. The flight controller continuously emits few MAVlink messages such as HEARTBEAT, GPS_RAW_INT, etc. at a specific frequency. The GPS module on the UAV is connected to the Raspberry Pi using USB A to micro B. Mavproxy, a command line based ground station is used to establish a heartbeat in the above connection. To do that, "mavproxy.py –master=Portname" is used as the command. Portname can be found in similarly to LiteVNA. After the connection is established, the MAV console can be used to get a live stream of GPS data using "watch GPS_RAW_INT". The rate at which MAVlink messages are received can be modified by sending a command using pymavlink. For this particular setup, recv_match function of mavutil module in pymavlink is used to listen for required messages [2].

mavutil: MAVLink utility functions for setting up communication links, receiving and decoding messages, running periodic tasks, etc.

- mavutil.mavlink_connection(device, baud, ...) for setting up a link to (initially) listen for messages or send messages on a channel (e.g. udp, serial, etc.). This returns an object representing the connection.
- recv_match() for capturing messages with particular names or field values

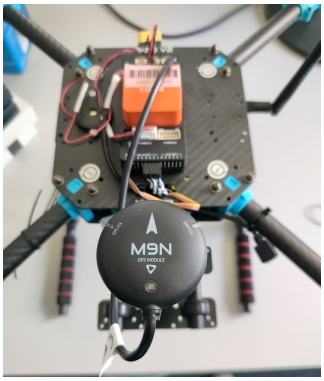

Figure 1.7: GPS module of the UAV

.

Table 1.4: GPS_RAW_INT message fields [3]

Field Name	Type	Units	Multiplier	Values	Description
time_usec	uint64_t	us			Timestamp (UNIX Epoch time or time since system boot). The receiving end can infer timestamp format (since 1.1.1970 or since system boot) by checking for the magnitude of the number.
fix_type	uint8_t			GPS_FIX_TYPE	GPS fix type.
lat	int32_t	degE7			Latitude (WGS84, EGM96 ellipsoid)
lon	int32_t	degE7			Longitude (WGS84, EGM96 ellipsoid)
alt	int32_t	mm			Altitude (MSL). Positive for up. Note that virtually all GPS modules provide the MSL altitude in addition to the WGS84 altitude.
eph	uint16_t		1E-2	invalid:UINT16_MAX	GPS HDOP horizontal dilution of position (unitless * 100). If unknown, set to: UINT16_MAX
epv	uint16_t		1E-2	invalid:UINT16_MAX	GPS VDOP vertical dilution of position (unitless * 100). If unknown, set to: UINT16_MAX

Field Name	Type	Units	Multiplier	Values	Description
vel	uint16_t	cm/s		invalid:UINT16_MAX	GPS ground speed. If unknown, set to: UINT16_MAX
cog	uint16_t	cdeg		invalid:UINT16_MAX	Course over ground (NOT heading, but direction of movement) in degrees * 100, 0.0..359.99 degrees. If unknown, set to: UINT16_MAX
satellites_visible	uint8_t			invalid:UINT8_MAX	Number of satellites visible. If unknown, set to UINT8_MAX
alt_ellipsoid ++	int32_t	mm			Altitude (above WGS84, EGM96 ellipsoid). Positive for up.
h_acc ++	uint32_t	mm			Position uncertainty.
v_acc ++	uint32_t	mm			Altitude uncertainty.
vel_acc ++	uint32_t	mm			Speed uncertainty.
hdg_acc ++	uint32_t	degE5			Heading / track uncertainty
yaw ++	uint16_t	cdeg		invalid:0	Yaw in earth frame from north. Use 0 if this GPS does not provide yaw. Use UINT16_MAX if this GPS is configured to provide yaw and is currently unable to provide it. Use 36000 for north.

2. Synchronisation between LiteVNA and GPS data

To achieve synchronisation between LiteVNA and GPS data, the following three methods were proposed as seen in Fig 2.1.

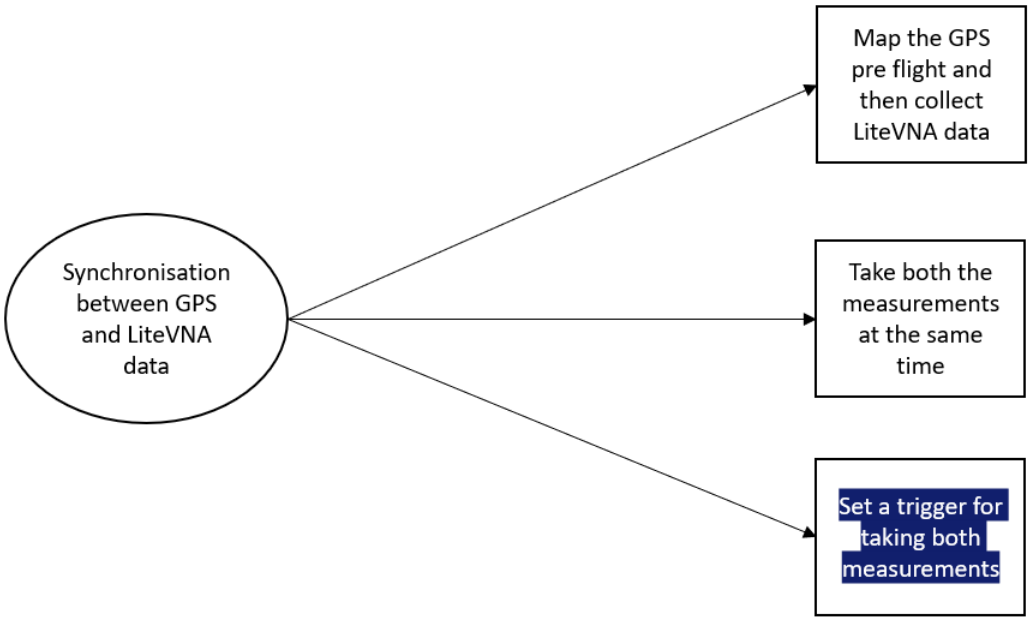

Figure 2.1: Proposed methods for Synchronisation

1. Map the GPS of the test area pre-flight and then take LiteVNA measurements
2. Run the code every time when data is to be taken
3. Set a trigger, such that whenever it is sent from the ground station, the data is logged

Method 3 is chosen, as the goal is to get both data together and running the code every time is not a possibility, since Raspberry Pi might be out of network range

during the flight. The code is started before the flight and whenever data is to be collected, a MAVlink (has a better range than wifi) message is sent to trigger the data logging.

2.1 MAVProxy

MAVProxy is a MAVlink protocol proxy and a ground station. It is a command line based ground station and can be simply installed as a python package. It can be used to establish a connection with the flight controller over USB, UART and internet (UDP or TCP). With MAVProxy, a live stream of MAVlink messages can be seen in the terminal using "watch MESSAGE". PC is connected to the flight controller with a USB A to micro B connector and using MAVProxy to use UDP port on Raspberry Pi to listen to the output stream using "python mavproxy.py –master=portname –out=IP Address of Pi : 14550". As seen from Fig 2.2 the executable path for python and location of mavproxy.py file should be entered in the above command.

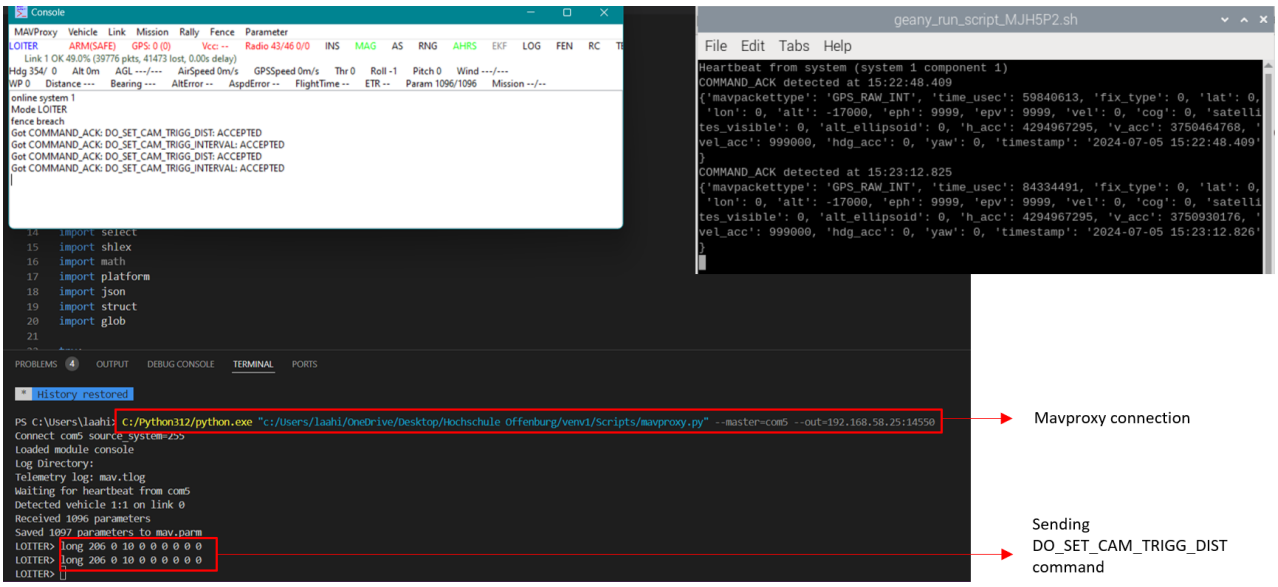

Figure 2.2: MAVProxy Setup for testing

2.2 QGroundControl

QGroundControl (QGC) provides full flight control and vehicle setup for PX4 or ArduPilot powered vehicles. It provides easy and straightforward usage for beginners, while still delivering high end feature support for experienced users. [4]

Key Features:

- Full setup/configuration of ArduPilot and PX4 Pro powered vehicles.
- Flight support for vehicles running PX4 and ArduPilot (or any other autopilot that communicates using the MAVLink protocol).
- Mission planning for autonomous flight.
- Flight map display showing vehicle position, flight track, waypoints and vehicle instruments.
- 3D viewer visualizing the 3D map of the environment (.osm file), the 3D model of the vehicle (only multi-rotors for the moment), and the mission 3D trajectory (including the waypoints).
- Video streaming with instrument display overlays.
- Support for managing multiple vehicles.
- QGC runs on Windows, OS X, Linux platforms, iOS and Android devices.

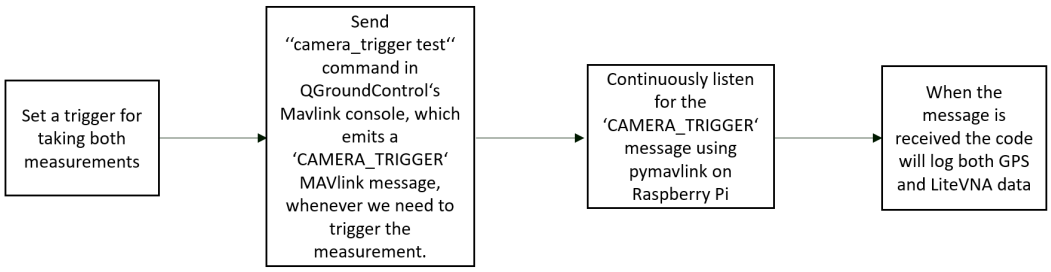

Figure 2.3: Selected method for Synchronisation in QGC

As seen in Fig 2.3, for logging both GPS and LiteVNA data at a particular instant, a command "camera_trigger test" is sent from Mavlink console, which is a part of Analysis Tools of QGC. The command emits a MAVlink message "CAMERA_TRIGGER". On Raspberry Pi the code continuously listens for this message and when it is heard, it logs the data.

2.3 Time consideration

The LiteVNA mesaurements are taken for the frequency range of 1GHz to 2.5GHz. In the following table, the number of sweep points and the corresponding time taken for sweep for this frequency range are seen. The measurement range of LiteVNA is 1-6GHz as seen in Table 1.3. The time taken might vary just at a scale of milliseconds, even if the complete frequency range is used, as long as the number of sweep points remain constant.

No of points in the sweep	Time taken for the sweep
101	0.34 seconds
1023	3.41 seconds

Table 2.1: Sweep Points and Corresponding Sweep Times

2.4 File Format

Each sweep data has to be recorded in separate .csv files and the begining of the file should contain GPS data. For this purpose, a sweep_counter variable is initialized to 0 and it gets incremented every time "CAMERA_TRIGGER" messages is heard. This variable is used in the filename to create separate files for separate sweeps. Fig 2.4 shows the file format.

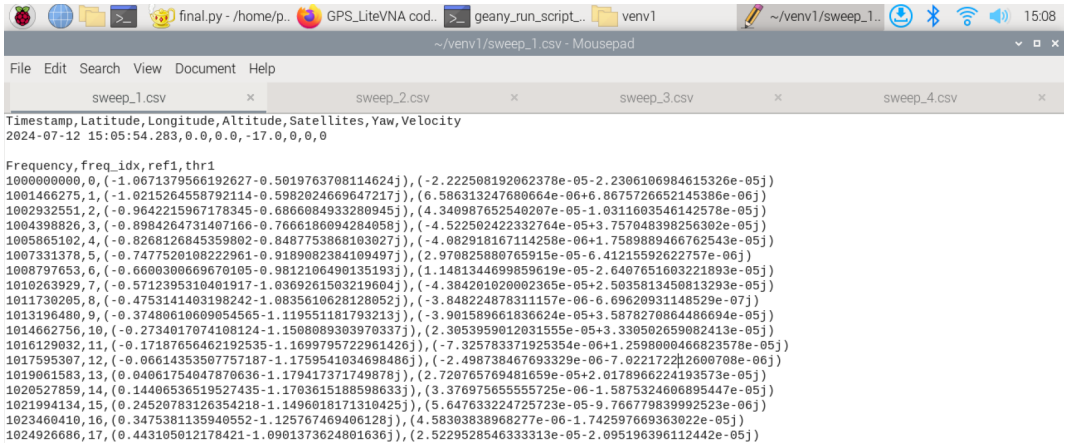

Figure 2.4: File Format

3. Considerations for Vivaldi Antenna Placement on the UAV

3.1 S11 measurements

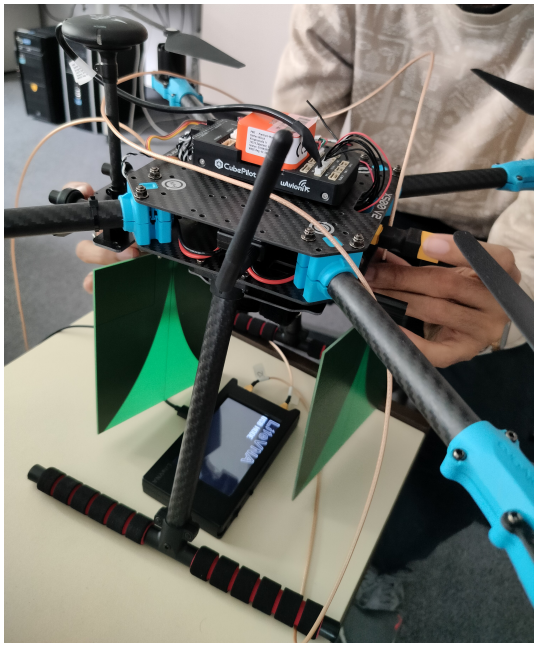

Figure 3.1: Antenna placement on UAV for S11 measurements

The antennas are placed on the UAV such that they have least obstructions in their path as can be seen in Fig 3.1 .The effect of legs on antenna can be seen in the Fig 3.2 at 23 cm. This is same as the distance between the start of antenna and the legs. Hence moving the zero of the measurements to 23cm i.e. removing the measurements before 23cm as an offset can remove the effect of legs of the UAV. As the setup has been shifted to a bigger drone which has the capability of lifting the legs away from the measurement field, the offset removal is not necessary.

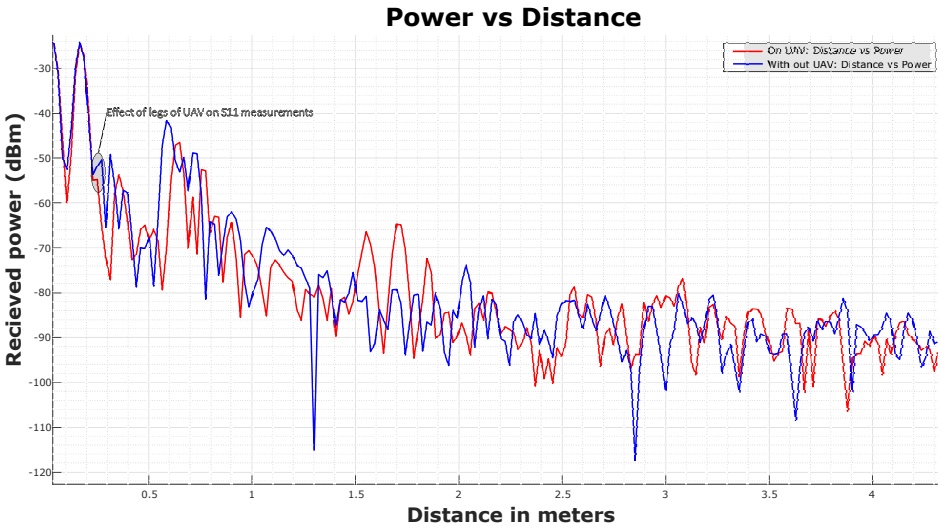

Figure 3.2: S11 measurements with and without antenna placement on UAV

3.2 S21 measurements

The separation between antennas should be such that they are not near enough to have any coupling between them and they are not too far away either. S21 measures the coupling between the antenna and hence is being used to determine the optimal distance between the antenna. The S21 measurements are taken by placing antennas at a distance of 7cm, 10cm, 12cm, 15cm. As can be seen from Fig 3.3 as the distance between antennas increases, the S21 decreases. 15cm is decided to be the distance between antennas.

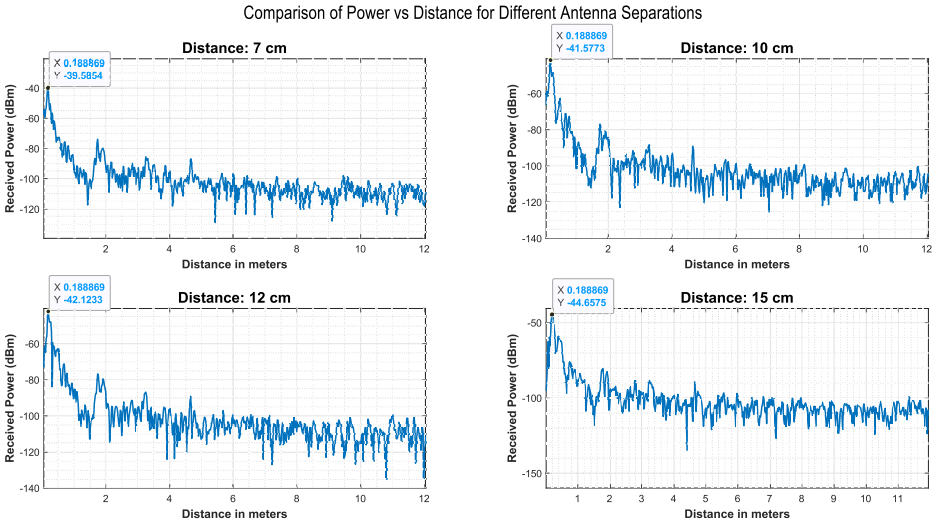

Figure 3.3: S21 measurements with different separations between the antenna

List of Tables

List of Figures

Bibliography

[1] Wikipedia contributors, *Virtual network computing — Wikipedia, the free encyclopedia*, [Online; accessed 24-July-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Virtual_Network_Computing&oldid=1216325354.

[2] MAVLink Development Team, *MAVLink Python Generator — mavlink documentation*, Accessed: 2024-07-24, 2024. [Online]. Available: https://mavlink.io/en/mavgen_python/.

[3] MAVLink Development Team, *GPS_RAW_INT — MAVLink Common Message Set*, Accessed: 2024-07-24, 2024. [Online]. Available: https://mavlink.io/en/messages/common.html#GPS_RAW_INT.

[4] MAVLink Development Team, *Qgroundcontrol*, Accessed: 2024-07-24, 2024. [Online]. Available: https://github.com/mavlink/qgroundcontrol.