Summer Internship Project Report

By: Dhruv Jain

May-July 2014

Directed By: **Dr. Nitin Gupta**

at NavStik Autonomous System,Pune

Contents

1 The Internship

1.1 Acknowledgements

I would like to express my highest appreciation to Dr. Nitin Gupta, my project advisor, for his invaluable guidance, excellent support and availability at all times throughout internship. This work could never be completed without his contributions, comments and reviews.

I would also like to thank Mr. Pradeep Gaidhani and Mr. Vallabh G, for their valuable help and support regarding technical aspects of my project.

My heartfelt thanks go to my beloved ones and my friends and other colleagues at Company for their continuous love and moral support.

1.2 NavStik

NavStik is a small group of engineers and researchers with background in electronics, computers, robotics, and aerospace-systems. We are excited about helping other researchers and DIY folks get started with implementing their automation algorithms for mobile robotics quickly and efficiently. We are continuously working on making recent developments in technology more accessible, thereby assisting in development of advanced systems for a variety of applications.

NavStik is a Micro Navigation and Control Board, designed for use in a variety of Autonomous Mobile Robots. It provides access to accelerometers, gyroscopes, magnetometers, pressure sensor, air-speed sensor and GPS. It can easily be interfaced to additional sensors, if required. NavStik is one of the smallest and lightest such modules available today.For more details visit navstik.org

1.3 My Curriculum

I am undergoing my B.Tech. + M.Tech. (dual degree) in Computer Science and Engineering at Indian Institute Of Technology Kharagpur.I have completed my 2nd Year in April 2014.During May-July 2014 I was doing my internship at NavStik.

2 Introduction

This project presents a system where Unmanned Autonomous Vehicles (UAVs)'s flying data called telemetry data which comprises of 259 parameters like roll,pitch,yaw ,FPE, GPS, acceleration and so on, and the video stream from UAV can be sent over IP.So telemetry link and VideoDownlink can be established over network assuming sufficient bandwidth is available.We can use 3G/4G or wifi connection to do so.
The project explains how the much demanded objective of establishing wireless link from UAV to ground can be established.

2.1 General Problem and Current situation

Currently when UAV flies,to get the telemetry data from UAV to the ground we use pair of Zigbees to establish wireless radio link.Obviously we have the disadvantages like:

1. Low range: upto 100 meters maximum.
2. Since we have pairs of Zigbees we have a standalone Ground Control Station where data can be transferred.
3. The bandwidth offered by Zigbee is around 250 kbit/s which is unsufficient for the transmission of video.

So to address these problems has been a topic of concern.So this project come up with an architecture where the problems we face now can be dealt with.

2.2 Objective and Scope of the Project

The area of interest for this project is establishing **telemetry link and Video Downlink over internet(3G/4G)** and make it available to several viewers on the ground.
We developed an architecture for onboard on UAV and for ground control station so that our problem can be addressed.It has several applications like long-distance viewing and monitoring.Since all the data are being transferred over internet we may expect that we have serveral UAVs flying at places several distances apart from ground and we at ground are able to have a link with the UAV.

2.3 Basic Approach and Final Picture

2.3.1 Telemetry Link from UAV

To establish telemetry link from UAV,on **Hardware side** we have our autopilot named navstik which carries all the data encoded in it, we have a single board computer Odroid-u3.Odroid is connected to Navstik via USB cable.Navstik has a program which enable it to send the data to some particular port which can be accessed via on odroid-u3.
On Software part,

Source or UAV runs a code on odroid which access the serial data from Navstik and send that via UDP to the server.

Server handles those UDP packets from several sources and keep their record.

Client uses QGroundControl Software where he can select the UAV data he wants to get and depending upon authentication QGCS receives UDP packets of corresponding UAV from the server.

We have multiple UAVs and multiple clients connected to some UAV.We have used socket programming on a core part at all ends i.e. source, server and client.Refer [1] and [2] for more details on socket programming.

2.3.2 Video Stream from UAV

To establish Video Downlink ,on **Hardware Side** we have a usb cam mounted on UAV connected to odroid via USB cable.So odroid can access the camera and get its feed.

Software Details:

On Source we have a program that sends the video packets by accessing camera to the server.

Server receives video stream from various sources.Server in turn streams the video and any client who have access to the url which comprises of server'ip

ip and specific name of corresponding video stream can view the video stream.

Client in this case has Web interface where we have a specific login for authentication and buttons corresponding to every active UAV clicking on which starts the video stream.

We have used VLC and VLM for managing video streams on source and server side. So as a final picture we have several sources sending video streams to the server.Server in turn streaming those streams.User on ground logs in and get the video stream.

3 Literature Review

The concept of getting telemetry link and video stream from UAV to ground is not new.It has been a topic of interest for various researchers.
A group named skydrone [5] has been involved in the development of UAVs to achieve the same objective.Their main focus is video transmission over cellular IP.Their system allows first person view only i.e. they doesn't allow user to control UAV, moreover it is one to one or many to one architecture not many to many.

There are several groups focussing on establishing telemetry link and video stream over IP especially using 3G/4G like that of [6].Again they talk more about one to one communication over IP.

Many methods have been implemented for video streaming like those using image processing at both source and client which are not reliable.Moreover there have been methods like mjpeg-streamer but even for direct communication from source to client, there is latency of about 1-2 sec and packet loss.

The basic methodology of this project also relies on known problems or previously done work but it addresses the problems not been addressed so far.By the end of this project, we are able to establish many to many communication and allow user to monitor flight at great distances.The user would be able to view low latency ,live video feed of every UAV and can get the telemetry data too.

4 Methodology,Design and architecture

We follow a 3-tier architecture for both establishing telemetry link and video stream from UAV.We have sources, server and clients.

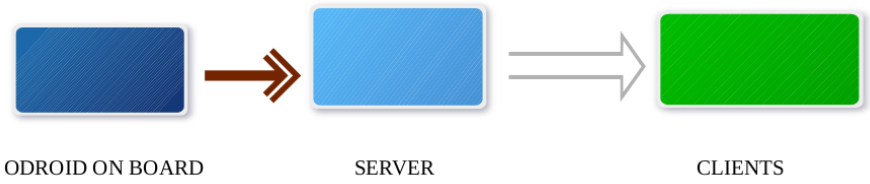

Figure 1: 3-tier Architecture

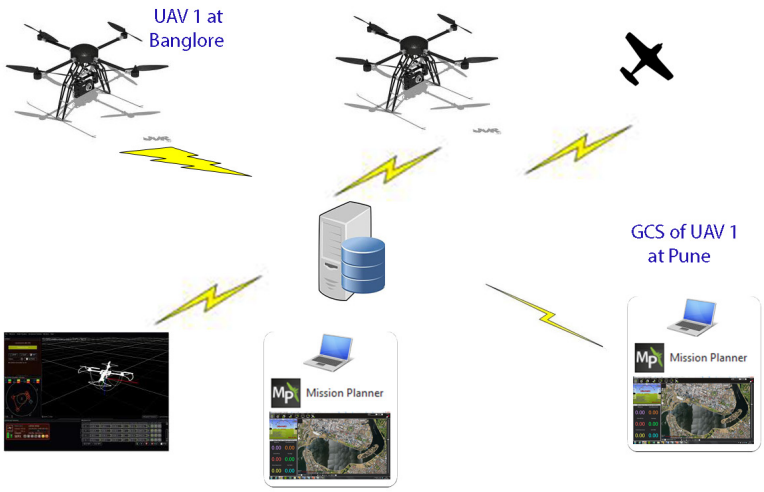

Figure 2: Final picture

4.1 Telemetry Link from UAV

Here we would discuss in detail the implementation on each side i.e autopilot,odroid-u3, server and client.

Implementation on autopilot: Navstik
A program runs on autopilot which when connected to odroid-U3 on board via USB cable sends mavlink packets to the connected serial port. So we get serial data at that port. The incoming serail data can be checked by executing the following command:

sudo minicom -D /dev/ttyACM0 -b 115200

where, /dev/ttyACM0 is the serial path, 115200 is the budrate for the telemetry data,
which results in some junk sort of serial data , which is actually mavlink packets to be seen in the console of odroid-u3 terminal display.
Since odroid-u3 has linux operating system(arm based), so for testing purposes we can also use any linux PC.Odroid-U3 also behaves in a similar way as any other linux PC, but may have different processing speed of programs as compared to normal linux PC.

Implementation on Odroid-U3

Here we run the main program that basically performs the task of receiving mavlink message through serial port and sending those messages via UDP to the server.
We have devised two ways to do so:

1. We have a C code (commtoserver.c) which does the following:

   • Opens the serial port and activates it to access the serial data over specified serial path like /dev/ttyACM0.

   • Sends some handshaking request to the server so that server receives the address of the source.

   • Runs two threads :

   • In the first thread it forwards serial data over UDP to the specified server's IP.

• In the second it receives UDP packets from server , convert it into mavlink format and forward it to serial port, like it can set waypoints, change some parameters etc.

2. **Using Mavproxy:** It is a command line based ground control station.More Information can be found at [3].It can be installed on a linux system by executing following commands:

   sudo apt-get update
   sudo apt-get install screen python-wxgtk2.8 python-matplotlib python-opencv python-pip python-numpy
   sudo pip install mavproxy

   It has a feature which allows us to connect our mav with many ground control stations over IP via UDP by sepecifying the port UAV is communicating upon and the output port where mavlink messages are to be sent. Through Mavproxy data can be sent to multiple GCS and can communicate with multiple UAVs.
   So we use mavproxy on Odroid-U3 in the following way:

   • Source sends server first handshake message.

   • It then receives the port at which server would listen to its data like 1234 for 1st UAV, 1235 for 2nd and so on.

   • It then runs the mavproxy through command as:

sudo mavproxy.py - -master=/dev/ttyACM0 - -baud=57600 - -out server's IP:1234

   In the above command –master is used for specifying the port UAV is communicating upon –out specifies the output port which in above case is the server's IP

Implementation at Server Server's main task is to be able to get the telemetry data from several UAVs and then sending it to any specific client.We again have come up with two approaches as follows:

1. Here server receives continuous telemetry data and stores them in a string array where each string stores the data of every UAV.On request by client it sends the data of corresponding UAV to the client. So we

have a code to do so.But it has some issues which would be discussed later.

2. **Server Uses Mavproxy**. Here server firstly sends the port at which UAV should send the continuous data stream, then server gets to know at which port it is getting data stream of which UAV.Then server on request by client runs mavproxy with input/source as input network stream and output as client's address.The basic flow which follows is :

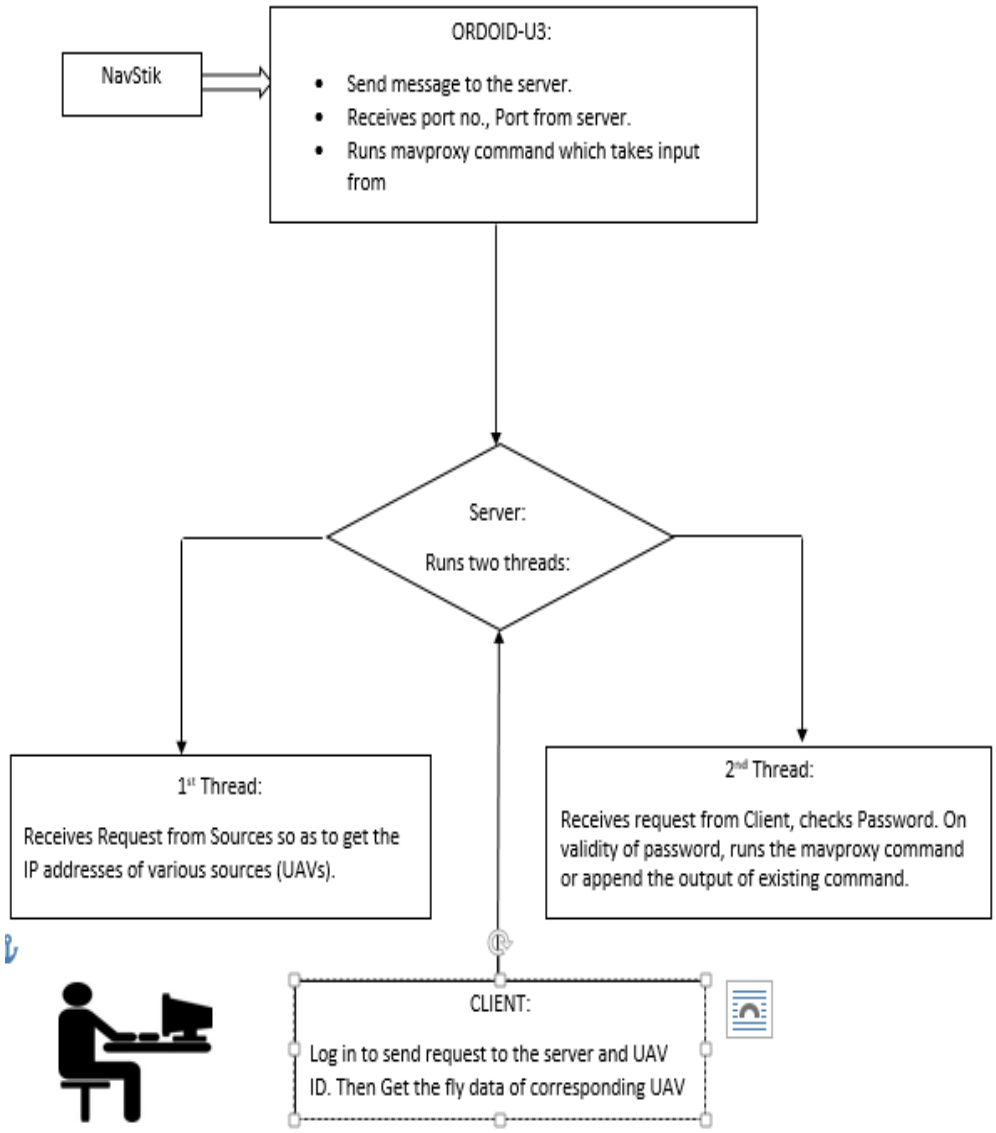

Figure 3: Flow for Telemetry link

It also has some issues to be discussed later.

Implementation at Client

Client here possess QGroundControl Software.It is an open source project.It has an option to receive/send UDP packet of mavlink data.So selecting this option would led QGCS to view telemetry data and also send messages via UDP to the server which in turn send it to the corresponding UAV.
Moreover we have provided a login for the users where users login and authenticated users gets the number of UAVs selecting which gives user access to data of corresponding UAV.The UI for the QgroundControl Software looks like:

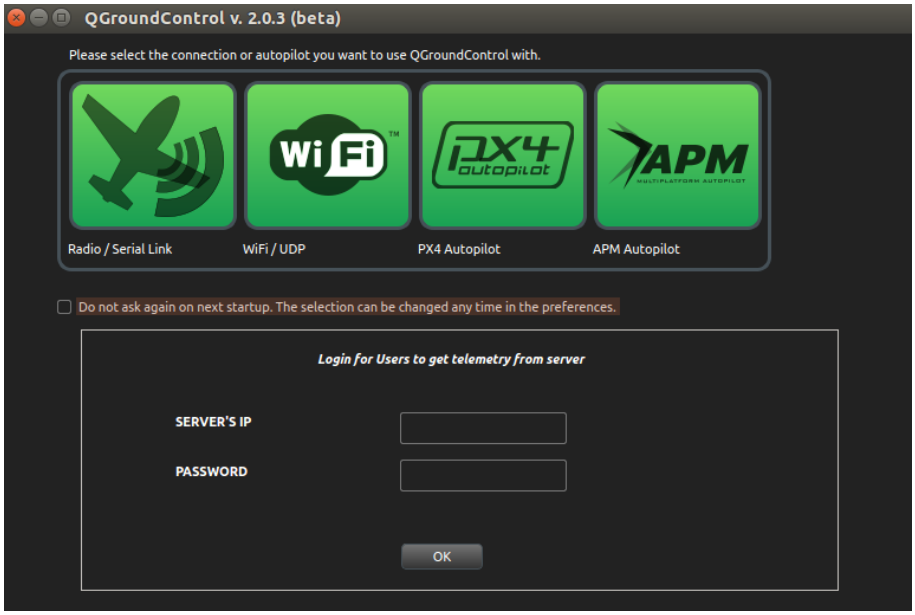

Figure 4: QgroundControl Initial Interface

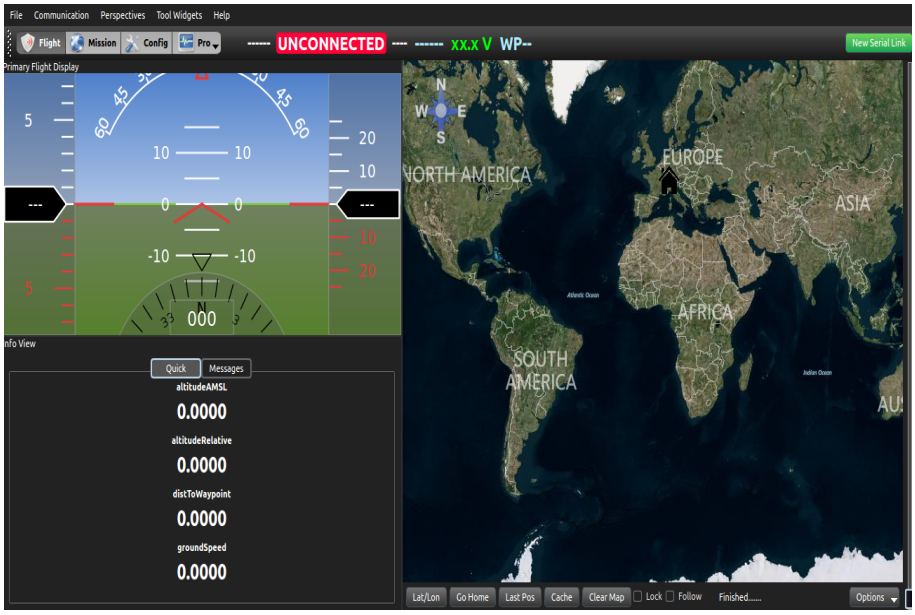

Figure 5: QgroundControl Telemetry View UI

4.2 Video Downlink from UAV

For video Odroid mounted on every UAV is attached to a USB camera.The setup looks like this:

Figure 6: Setup on UAV

4.2.1 Implementation On UAV

We run a program on odroid which does the following:

1. Send handshaking message to the server so that server knows the IP address of the UAV.

2. Source then starts VLC streaming.VLC provides a way so that video from any device attached to the system can be streamed.We can access the stream by knowing the IP address and port number of the stream.More details can be found at [4].So we first install VLC on odroid by writing the command:

   sudo apt-get install vlc

   The command for starting video stream is:

   cvlc v4l2:// :v4l2-vdev="/dev/video0" –sout '# transcodevcodec=mp4v, acodec=mpga,vb=800,ab=128,fps=29.97,width=640,height=480:standard access=http,mux=asf,dst=:8080' –no-sout-audio

   Here, Different kind of processing are applied to the stream during this process (transcoding, re-scaling, filters, re-muxing...).

   Transcode: This module is used to transcode a stream, i.e. to change its codecs or the encoding bitrates. Some additional processing can be done during this process, such as re-scaling, deinterlacing, resampling, etc.

   vcodec: This options allows to specify the codec the video tracks of the input stream should be transcoded to.We have several codecs available like MJPEG,MPEG,mp4v,AVI,WMV,H.264 and many more.Here we have used mp4v or mpeg-4 for its greater compatibility with http streaming.

   vb: This option allows to set the bitrate of the transcoded video stream, in kbit/s.It depends which type of connection we are using.For Wifi connection it is recommended that for a resolution of 640X480 use bandwidth of 600-850 kb/s.For cellular connection, we should use

a bandwidth of 200kb/s for resolution of 400X300.

fps: This options allows to set the framerate of the transcoded video, in frames per second; reducing the framerate of a video can help decrease its bitrate.

dst: This option allows to give informations about the location where the stream should actually be saved or sent.
Here is the meaning of the dst option depending on the parameter used for the access option.

Streaming Protocol: Several protocols are available like RTP/UDP,RTSP, HTTP,MMSH,File.
The RTSP (Real Time Streaming Protocol) protocol is a client-server multimedia presentation control protocol, designed to address the needs for efficient delivery of streamed multimedia over IP networks.
Here We have used HTTP. HTTP stands for HyperText Transport Procotol. It's a application network layer developed in early days of Internet for transmit html and text files. Today it's used for every kind of media, thanks to it's flexibility. VideoLAN can use this protocol for multimedia streaming. It can read a stream from an HTTP server like Apache, or act like a HTTP server dumping multimedia to the network.

Mux: This option allows you to set the encapsulation method used for the resulting stream. This option has to be set.Available mux for http streaming are ts,ogg,asf.

More details on compression, streaming protocols,congestion control,error control and other such parameters required for video streaming can be found at [7].

4.2.2 Implementation on the Server

Server for video also uses VLC (Video Lan Client) and VLM (Video Lan Manager).We have come up with the two approaches one using VLC streaming as done on the source side and other using multiple streaming using VLM configuration files.

The basic steps of execution of program (server1.cpp) which uses vlc are as follows:

1. The server first waits for source to connect to the server so that it gets the IP of source.

2. Then it runs a command in a separate thread for each UAV which streams the video of particular UAV at some port starting form 8080 for 1st UAV, 8081 for 2nd UAV and so on.

3. Then it creates asf files for each UAV which basically contains a script having the address of network stream to play.For example, it will have a tag:
   <REF HREF="https://server's IP:8080" / >
   Playing a asf file in a VLC with such a tag will play the corresponding network stream.
   So, asf files equal to number of UAVs are created on the server which is then played by client on request.

Now we will see the server program (server2.cpp) which uses VLM.More info on VLM can be found at [9].

1. Here also server firstly listens the source so that it gets the network url at which it gets the video stream of corresponding UAV.

2. Then it makes is a VLM configuration file (.conf file) which is a list of command lines : one line corresponds to one command line.To create a configuration file, just edit a text file and type a list of VLM commands. So server creates a vlm.conf file which is dynamically edited as no. of UAVs increase.For each UAV it contains the following command lines:

VLM Configuration File :

new channel1 broadcast enabled
setup channel1 input "http://source1's ip:8080"
setup channel1 output # httpmux=ts,dst=:8095/live1
setup channel1 option http-reconnect
setup channel1 option no-sout-rtp-sap option no-sout-standard -sap option ttl=1 option sout-keep

control channel1 play

The file is appended as more and more UAVs add on.Here user can access the stream of source1 by playing the network stream as http://server's ip:8095/live1.

3. Server in the same way makes .asf files for every UAV with network url as above.
   Here one thing to note is that we have used a single port i.e 8095 in this case, to obtain the stream of every UAV.Here address also extends the information within the port itself so that we have streams of every UAV on a single port.

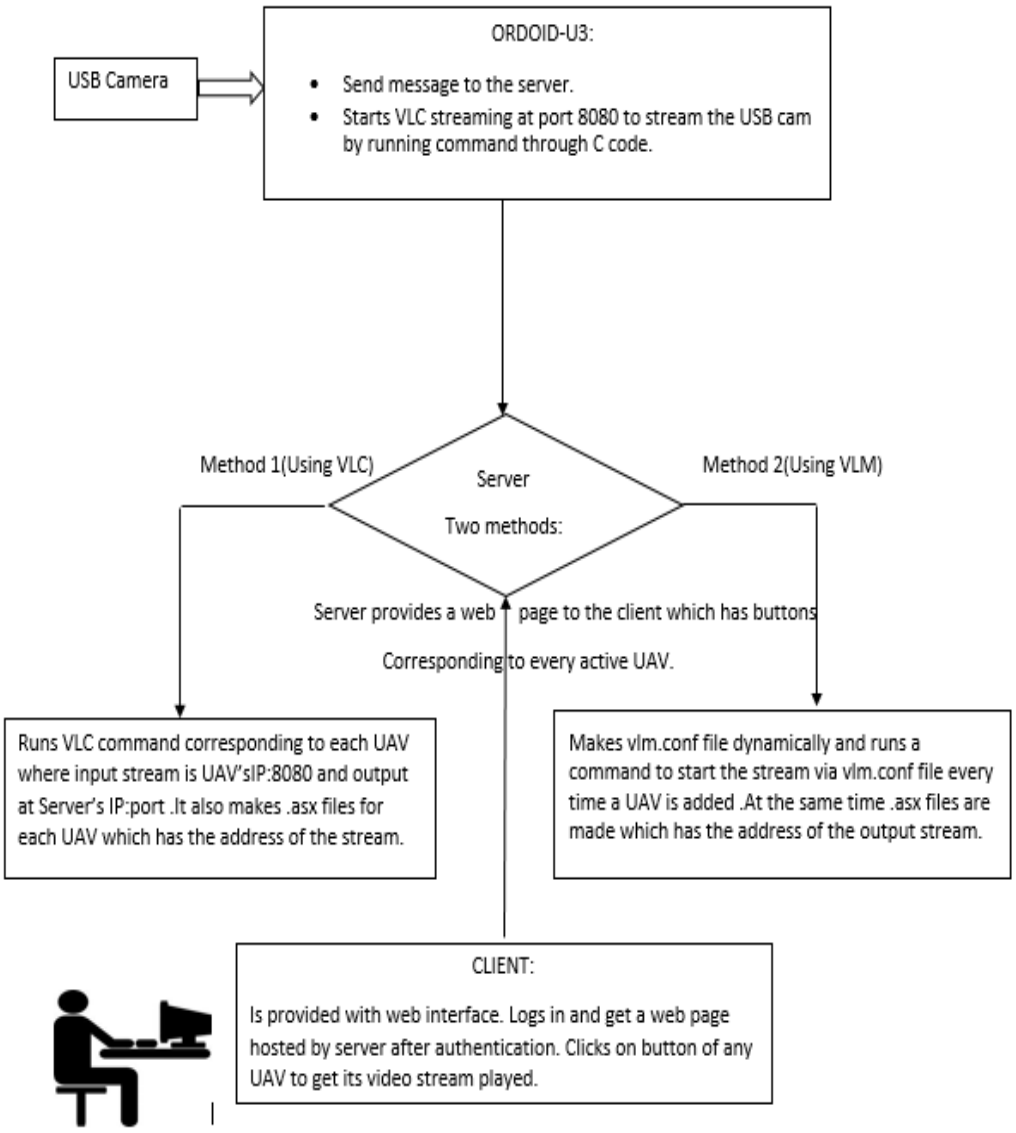

Figure 7: Process for Video Streaming

4.2.3 Implementation On the Client

Client in this case is provided with a web interface.Server provides a portal where user has to log in, and after authentication client is directed to a web

page where user can access the video feed of any active UAV by clicking on button.
User first have to install suitable plugin.Information of suitable plugin is as follows:

1. **For Windows User:** Reinstall VLC and while installing **don't untick** VLC browser plugin for chrome or mozilla.

2. **For linux Users:** User must use **Mozilla firefox** browser.Run the following command in the terminal:
   sudo apt-get install vlc browser-plugin-vlc
   sudo apt remove totem-mozilla
   Then User can view the feed in the browser itself.User can also see the stream in the VLC media player also.Just change the preferences in mozilla by edit→preferences →applications.Then change action for asf video to use vlc by browsing to the executable file of vlc.

Figure 8: Web-Interface for Users

5 Tests and Results

We successfully conducted tests for both for getting telemetry data and Video Downlink with various variations like single source vs multiple source, single client vs multiple clients and other such variations which we will discuss here.Here for the source we have used linux PCs.The results are well applicable for odroid-u3 also.

5.1 Telemetry link Experiments

We conducted tests for establishing telemetry link in wifi network with following variations and methods:

1. **Single source and Single Client:** Here, firstly we ran C code (commtoserver.c) which transfers UDP packets to the server.Server(oneonetelemetry.c) firstly is made capable of listening to a single client on request.Then on source we ran mavproxy which does the same thing of sending UDP packets to the server, and server does the same thing.

   Result: We got satisfactory result.Client was able to get the telemetry data in both cases(code and mavproxy).But in case of code communication is one way i.e from source to client only (not the other way).

2. **Single source and Multiple Clients :**Here we used mavproxy code on the source.On server firstly we simply received UDP packets and then send those to the no. of clients on their request.Then we used Mavproxy on the server, so as to run the command to send the data to the multiple clients network ports.

   Result: For the first case we got a warning in the QGCS : Another system is using the same system ID, but we got the telemetry data.The second case worked perfectly and each client is now capable to send and receive UDP packets via QGCS.

3. **Multiple Sources and Multiple Clients**Here on the source we used mavproxy .On the server end we again tried both methods.Firstly we

used a method of receiving raw UDP message, storing them in a string and then sending to the client.
Then we used Mavproxy on the server.So we run several mavproxy commands equal to the no. of UAVs and send the data of UAV to the corresponding client on request.

Result: Using mavproxy at the server's end works perfectly fine.Each client is able to access and monitor the telemetry data of UAV requested.
Using the method of storing the data in a string and then sending to the specified client did'nt work in this case.We get an warning: Mavlink baud rate mismatch.We will evaluate this issue in later section.

5.2 Video Downlink Experiments

We conducted tests for getting video stream from different sources using following variations.

1. **Single Source:** Here we run the code for starting video stream on a single linux PC.Server then streams that video.
   Result: Client is able to access the video stream by entering server's network url and clicking on UAV1.We get a delay of maximum 500 ms with almost no packet loss and jitter.

2. **Multiple Sources:** Here we run the same code on 3 linux PCs.Here we test using two programs on the server discussed above(VLC and VLM).On the client side we get 3 buttons for 3 different UAVs, clicking on which gives the video feed of corresponding UAV.
   Result: Video feed for all UAVs can be accessed.But as compared to previous case there is latency of about 1s for all UAVs.Packet loss or jitter is almost negligible.

6 Evaluation and Discussion

The report fully describes a fully functioning system to establish telemetry link and video downlink from UAV to ground.Although We successfully con-

ducted experiments to achieve the same and got satisfactory results,there were some issues in the current system which are discussed as follows:

1. **Telemetry link** We are able to send the data form source to the server either using commtoserver.c or mavproxy, but on the server end using method of receiving UDP packets , storing them in string and sending string to the client having Qgroundcontrol doesn't work perfectly.It gives error: **mavlink baud rate mismatch**.Since QGCS is designed to receive mavlink messages at a specific baud rate, the error arises.

2. **Telemetry link** As of now the method of receiving and sending UDP packets work one way i.e. server is capable of only sending UDP packets not other way.

3. **Telemetry link** Using Mavproxy on the server end works perfectly.Moreover it allows both way communication to all clients.So it creates problem of accessibility i.e. as of now every user is capable of receiving as well as sending commands to particular MAV.So multiple user can monitor the same UAV.Single User monitoring and multiple users just viewing would be more realistic.

4. **Telemetry link** As of now the server code allows a particular user to access the telemetry link of single UAV only till the server is on.In other words, any user getting the data of some UAV can't switch to get the data of some other UAV.Once the server sends the data to particular client, that client gets the data of that UAV only as long as server is on.If the user tries to switch he will get the data of both UAV on single GCS which is not reliable.

5. **Video Downlink** The system successfully allowed us getting video stream from UAV to the ground.Server provides a web-interface to the users where it has buttons for every active UAV.But as of now there is no provision of checking if the UAV once activated has been active at some moment or not i.e. once a UAV is activated server provides a button for that UAV, which remains as long as server is on.We rather expect that once the UAV goes off button for that UAV either should not be visible or there should be indication that the UAV has been deactivated.

We have conducted all the experiments over wifi. These are some of the issues discussed.We will try to solve the issues in the later version of our project.Some of the future work involving solving these issues are being discussed in next section.

7 Further Improvements

The major objective of the project was achieved but still in the coming future we would like to address the issues discussed above and some of the challenges are left open ended.

- The server for establishing telemetry link should be designed to effectively use the methiod of receiving and sending UDP packets at correct baud rates instead of using mavproxy as it can be achieved by source sending data on a single port and will be independent of third party module - mavproxy.

- **Mavproxy module should be modified to be capable for two way as well as one way communication from UAV to ground**, so that only one user monitors the flight set parameters, set waypoints etc. and multiple users get to only see and get the parameters.

- **Server should allow user to also switch the UAV data they are accessing** and should not limit them to just capable of getting telemetry data of just single UAV as long as server is on.This can be done by storing the address of client and UAV data it is getting.If the same user tries to connect a check is performed and both the previous as well as new commands are changed and then run so that the user only get the data of updated UAV he requested.

- Another module for mavproxy named **mavelous [8] is available which provides web interface for getting telemetry link also**.But since mavelous lacks certain features we didn't use it in this project.But in future we would like to use mavelous module for mavproxy, so that for telemetry data also we are able to have a light weight web interface instead of using Qgroundcontrol Software.

- **Server for video stream should provide the information of active or disconnected UAV**.Server should kill the command for disconnected UAV.Moreover, the active UAV if disconnected should again send the video stream to the server and server should recognize it.

8 Conclusion

An approach for establishing telemetry link and video link was discussed and implemented in this report.We presented a new and novel idea and addressed the problems in the work done previously.The whole system have been tested and the results have been quite satisfactory.Again, We are in the process of addressing the issues discussed above and making the overall system more efficient.

9 References

[1] http://www.cs.dartmouth.edu/~campbell/cs60/socketprogramming.html

[2] Beej's Guide to Network Programming Using Internet Sockets Brian Beej Jorgensen Hall beej@beej.us Version 3.0.15 July 3, 2012

[3] http://tridge.github.io/MAVProxy/

[4] http://www.videolan.org/doc/streaming-howto/en/ch05.html

[5] http://www.skydrone.aero/

[6] http://diydrones.com/group/telemetry-over-cellular-ip

[7] http://users.ece.cmu.edu/~peha/streaming_video.pdf

[8] https://github.com/wiseman/mavelous

[9] http://www.videolan.org/doc/streaming-howto/en/ch05.html