Queensland University of Technology

Autopilot Failure Mode Analysis Simulation in the Loop Software (SITL)

User Guide

Developed by:

Osama Yahya | 4th Year Bachelor of Avionics Engineering.

Supervised by:

Dr.Troy Bruggemann | Accelerate Research Fellow in Robotics.

CONTENTS

1. SITL Overview:

SITL is a simulation in the loop software. It was developed to understand the behaviour of Unmanned Air Vehicles (UAV) in different failure events, and obtain accurate probabilities of failure occurrences. Such objective can only be achieved by performing numerous flight simulation tests. SITL achieves this by automating simulated flights and obtain its' telemetry logs for probability calculations.

The figure below shows the process stages of SITL

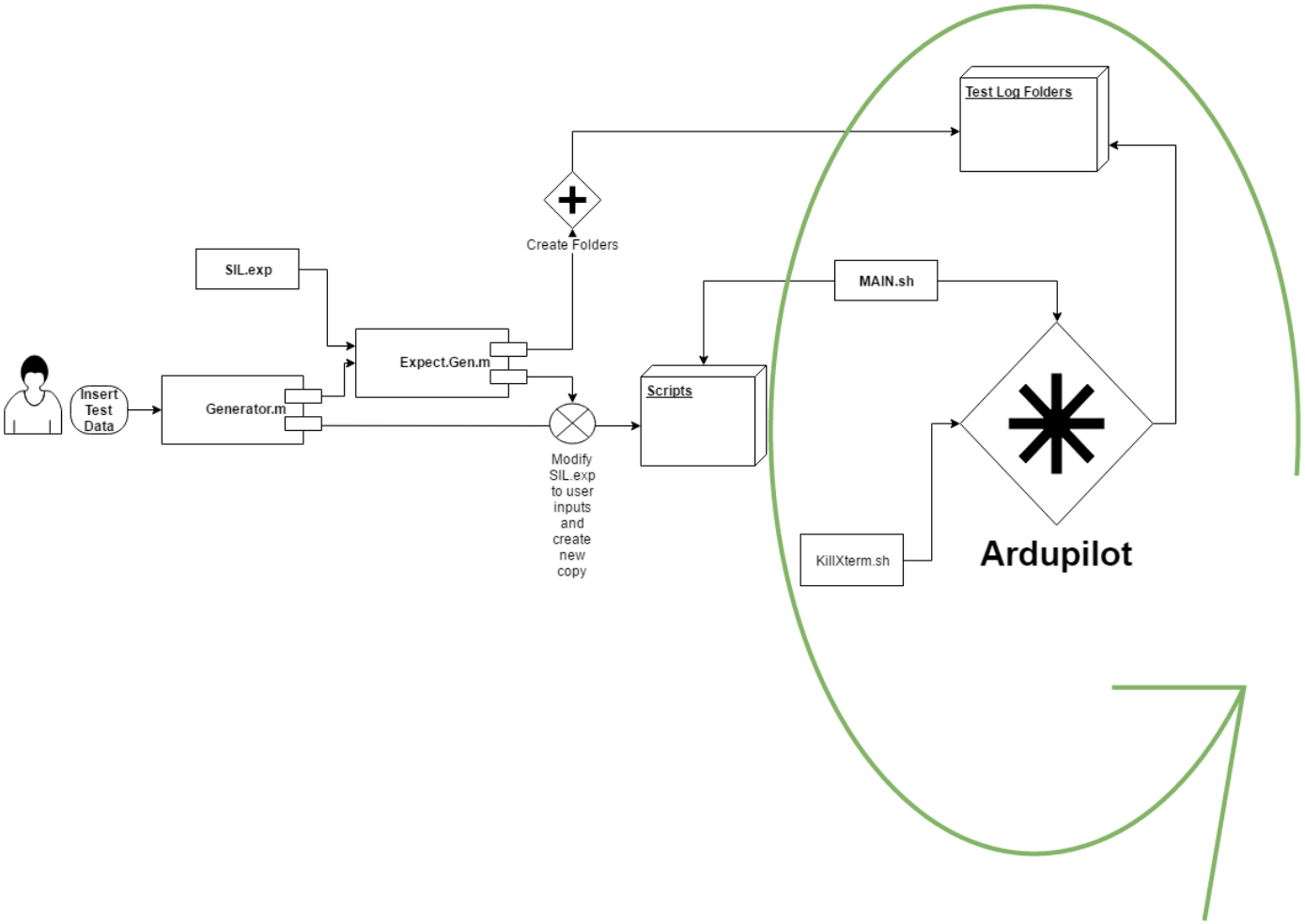

Figure 1 SITL Process

2. SITL Credential Details:

2.1 Dropbox Account:

Dropbox contains all the files you will need to operate SITL

Email: quasuav@gmail.com

Password:SITLQUASUAV

3. SITL Software:

SITL consists of the following software:

1. **MAVProxy.**
2. **ArduPilot Repository.**
3. **MATLAB.**
4. **Expect Package.**
5. **Dropbox.**
6. **MissionPlanner.**

4. SITL Environment:

This software was developed and tested in **Linux Ubuntu 16.04 LTS** operating system on a computer with the following specifications. Please consider them as minimum specifications:

Memory	2.9GiB
Processor	Intel Core 2 CPU T7200 @ 2.00GHz x 2
Graphics	Intel 945GM
OS-Type	64-bit

5. SITL Files:

SITL consists of the following documents. Please ensure they are available in the Dropbox folder as shown in Figure 2

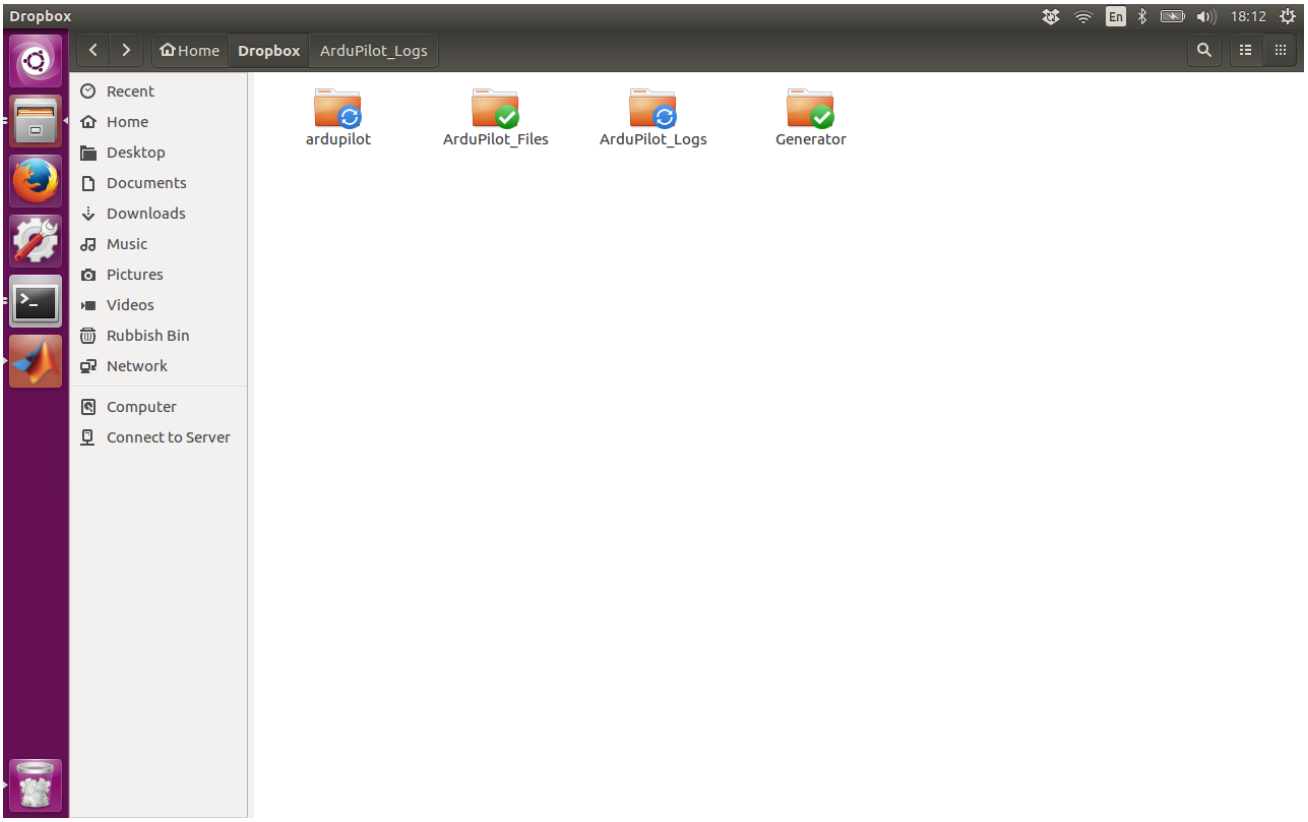

Figure 2 Dropbox Folder

5.1 ardupilot:

This file contains the current version of ardupilot simulator at the date of writing this document (12/2016). You can obtain an updated version from the following link

(http://ardupilot.org/dev/docs/sitl-native-on-windows.html)

5.2 Generator:

This file contains all the functions to handle and generate the flight scripts

1. **Expect_Gen.m**

This is the function that generates the scripts. The function is connected to Generator.m file. It will take the inputs in Generator.m file and modify SIL.exp based on these inputs.

2. **Generator.m**

This is the Expect_Gen.m function user interface. The user uses this function to enter the desired test parameters, and run it.

3. **SIL.exp**

This is the script template that Expect_Gen.m uses to modify the desired changes entered by the user in Generator.m file.

5.3 Ardupilot_Files:

This folder contains essential codes to run with SITL.

1. **KillXterme.sh**
   This will kill the Mavlink connection after each test. Otherwise, ardupilot will keep creating new Mavlink for each test without closing the previous one. Therefore, resulting in poor computer performance.

2. **FP.Waypoints**
   This is the flight plan. Refer to section 6 for detailed information.

3. **MAIN.sh**
   This file will give permission to all the generated exp files to be executed and then execute them all.

5.4 Ardupilot_Logs:

This folder will host all the log files generated by Expect_Gen.m function. Expect_Gen.m function will dynamically create folders for each test, titling each folder by the test ID as shown in Figure 10.

6. Flight plan:

Figure 3 Flight Plan

The flight plan stored in FP.Waypoints is shown above in Figure 3.

6.1 Waypoint H:

This is the home waypoint. At this point, the following commands are executed:

1. Reset all parameter to default.
2. Set simulation speed to the user input.
3. Switch to mode GUIDED.
4. Execute the initial parameter.
5. Arm Throttle.
6. Take-off to 10m
7. Once height 10 is reached, load FP.Waypoints
8. Switch to mode AUTO.

6.2 Waypoint 1:

1. Once waypoint 1 reached, execute the final parameter.

6.3 Waypoint 2:

1. Once waypoint 2 reached, switch to mode LAND.
2. Once landed, disarm.
3. Reset all parameters to default.
4. Switch simulation speed to the user input.
5. Start the next flight.

It is important to mention that the flight scripts were written so the UAV vertical and horizontal movements are separated. Look at Figure 4:

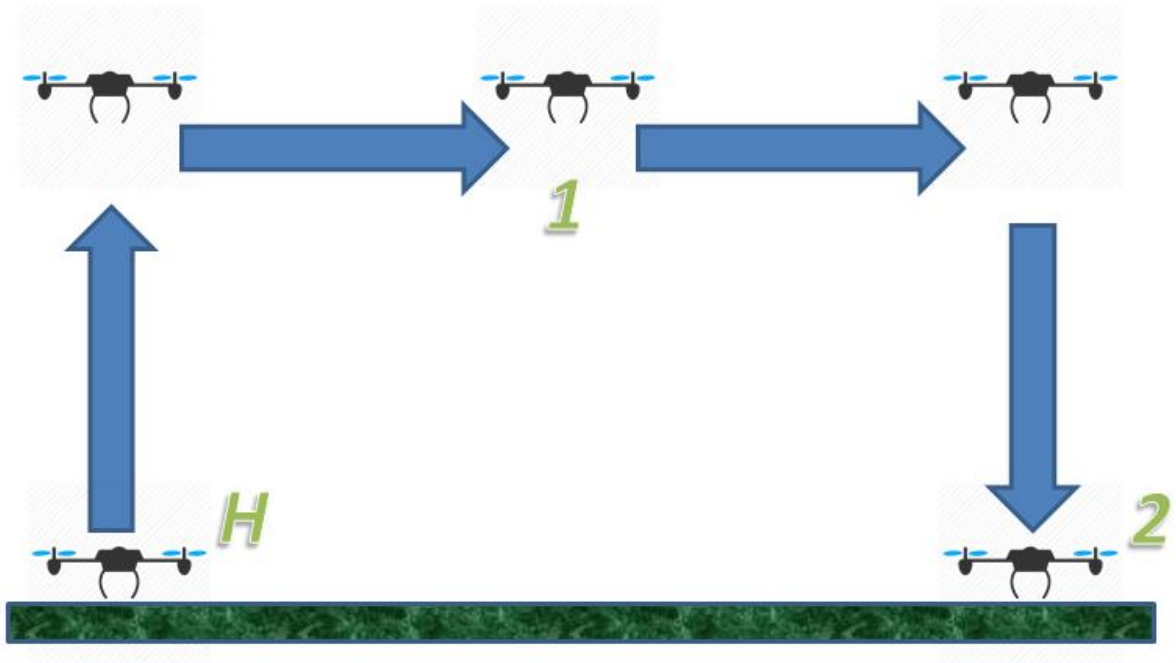

Figure 4 Flight Movements

7. Installation:

7.1 Ardupilot Repository and MAVProxy:

Please follow the instructions in the link below to install Ardupilot repository and MAVproxy on your Linux machine.

http://ardupilot.org/dev/docs/setting-up-sitl-on-linux.html

7.2 MissionPlanner:

MissionPlanner is only available in Windows operating systems. You will need it to convert binary telemetry logs produced by ardupilot to m files so they can be processed in MATLAB. Follow the link below to download MissionPlanner.

http://ardupilot.org/planner/docs/common-install-mission-planner.html

7.3 Dropbox:

Currently, SITL is programmed to execute Expect_Gen.m and store the logs in Dropbox folder. Therefore, you will need a Dropbox directory in your Linux machine. Follow the link below to download it.

https://www.dropbox.com/install-linux

7.4 MATLAB:

MATLAB is used to analyse and plot the flight logs produced by Ardupilot. Download MATLAB Linux or windows version depending on which machine you decide to use for log analysis.

7.5 Expect Package:

Expect package is automation tool that allows the computer to operate certain actions; in this case it's the UAV simulation. To install this package, open a bash terminal and enter the following command:

```
sudo yum install expect
```

OR

```
sudo apt-get install expect
```

8. Operating SITL:

8.1 Operating in QUAS Machine:

If you want to use SITL on the QUAS Linux machine (The one that SITL was developed on), you will only need to go to Generator folder in Dropbox and open Generator.m

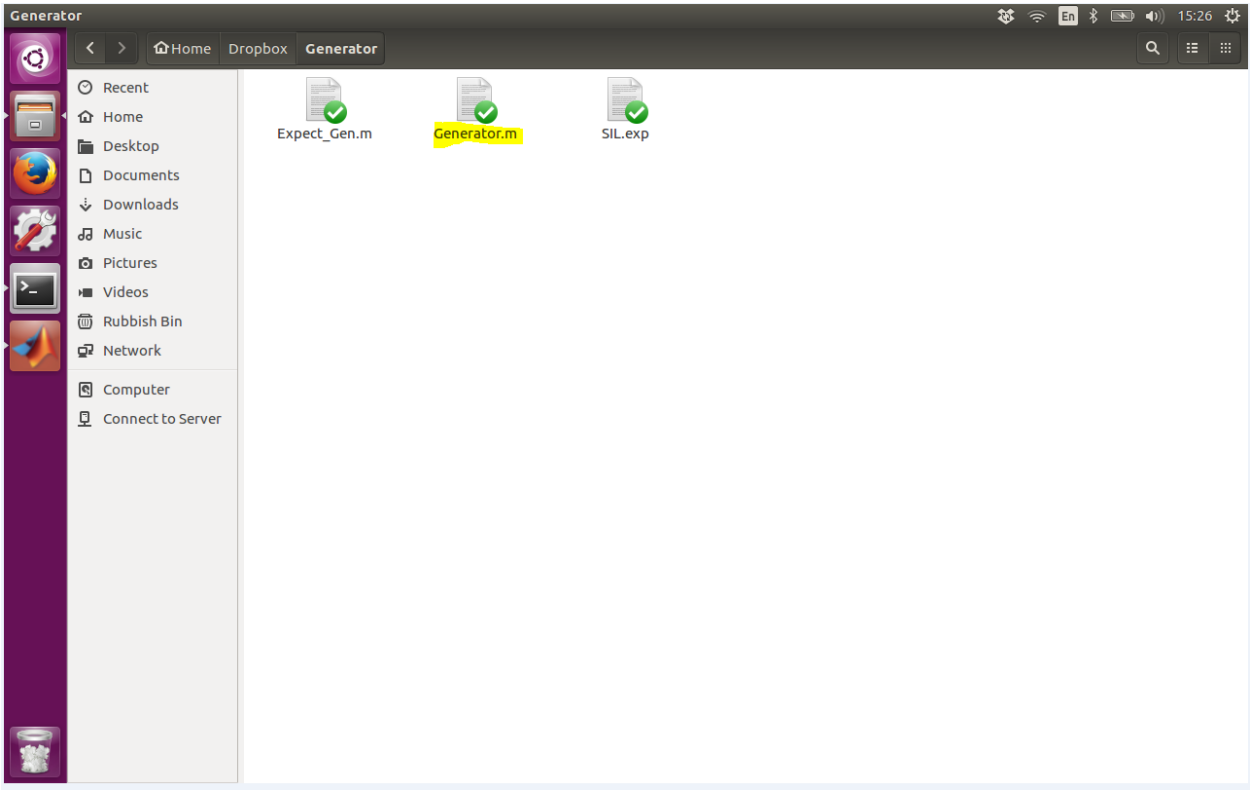

Figure 5 Generator Folder

Then fill up the scripts parameters as in Figure 6.

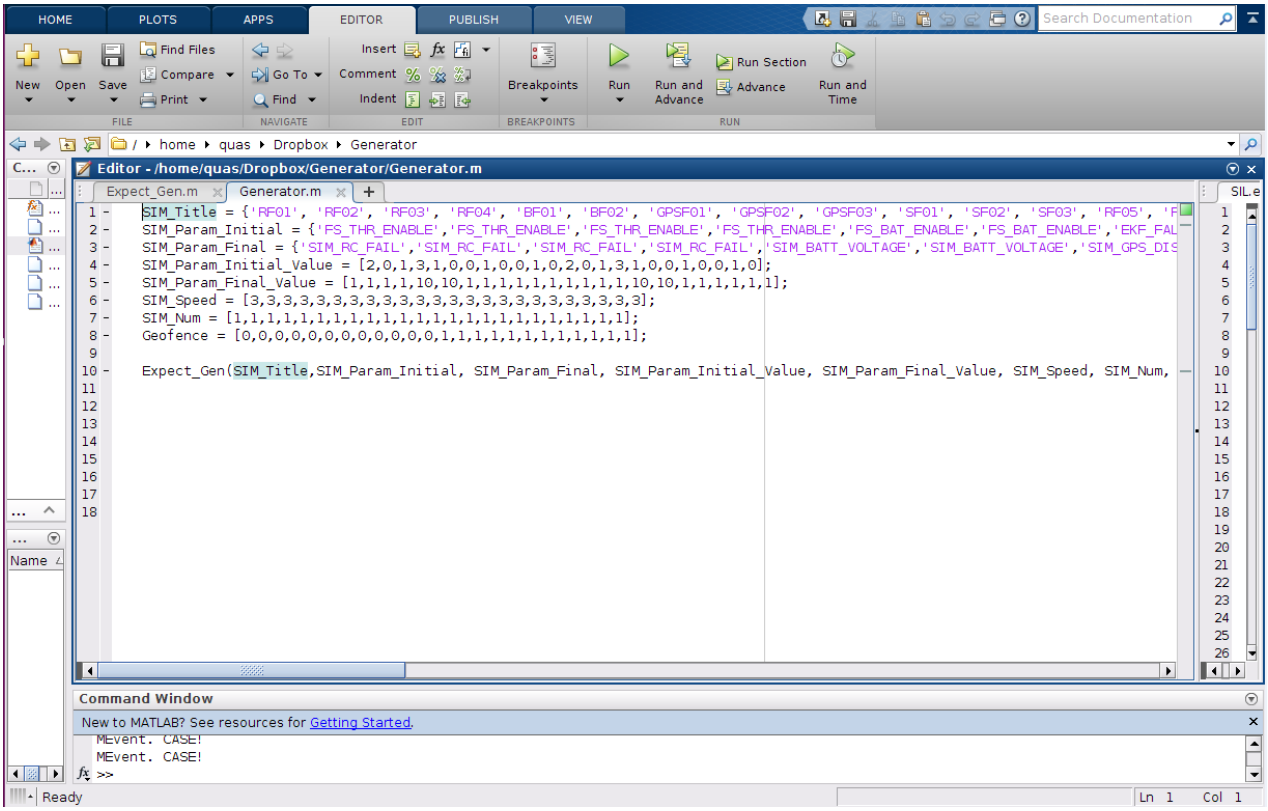

Figure 6 Generator Function User Interface

1. **SIM_Title**: The failure ID.
2. **SIM_Param_Initial**: The first parameter that will be changed before takeoff.
3. **SIM_Param_Final**: The final parameter that will be changed in mid-air.
4. **SIM_Param Initial_Value**: The value of the first parameter that will be changed before takeoff.
5. **SIM_Param_Final_Value**: The value of the final parameter that will be changed in mid-air.
6. **SIM_Speed**: The simulation speed.
7. **SIM_Num**: The number of tests that need to be performed.
8. **Geofence**: Geofence on or off.

In the image above, the first script that MATLAB will generate will be titled as RF01.exp, this script will automate a flight that have an initial parameter FS_THR_ENABLE = 2, and final parameter SIM_RC_FAIL = 1, the simulation speed will be 3, and the number of flights will be 1, with no Geofence.

After executing MATLAB, you will find all the scripts in the ArduCopter directory (**~/ardupilot/ArduCopter)** as shown in in Figure 7.

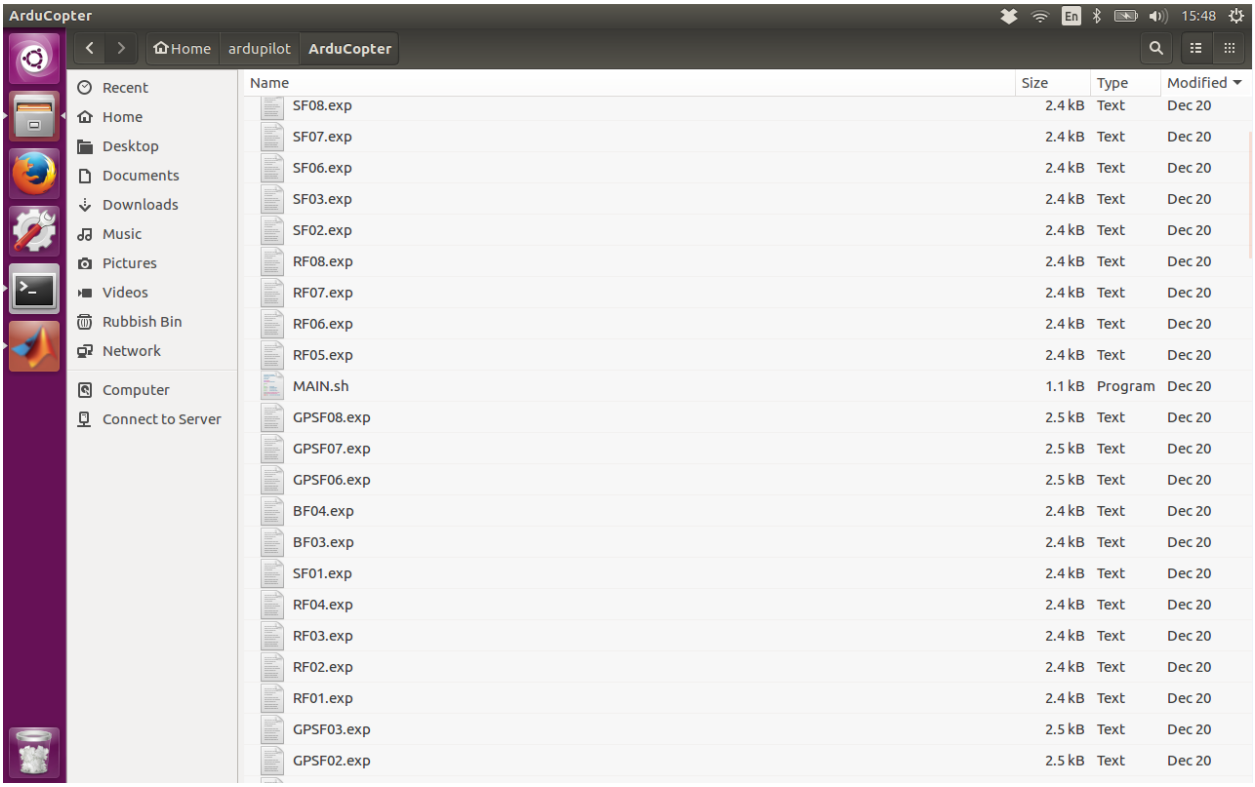

Figure 7 ArduCopter Directory

You will also find a MAIN.sh file in the same directory. This is the file that will run all the exp files above.

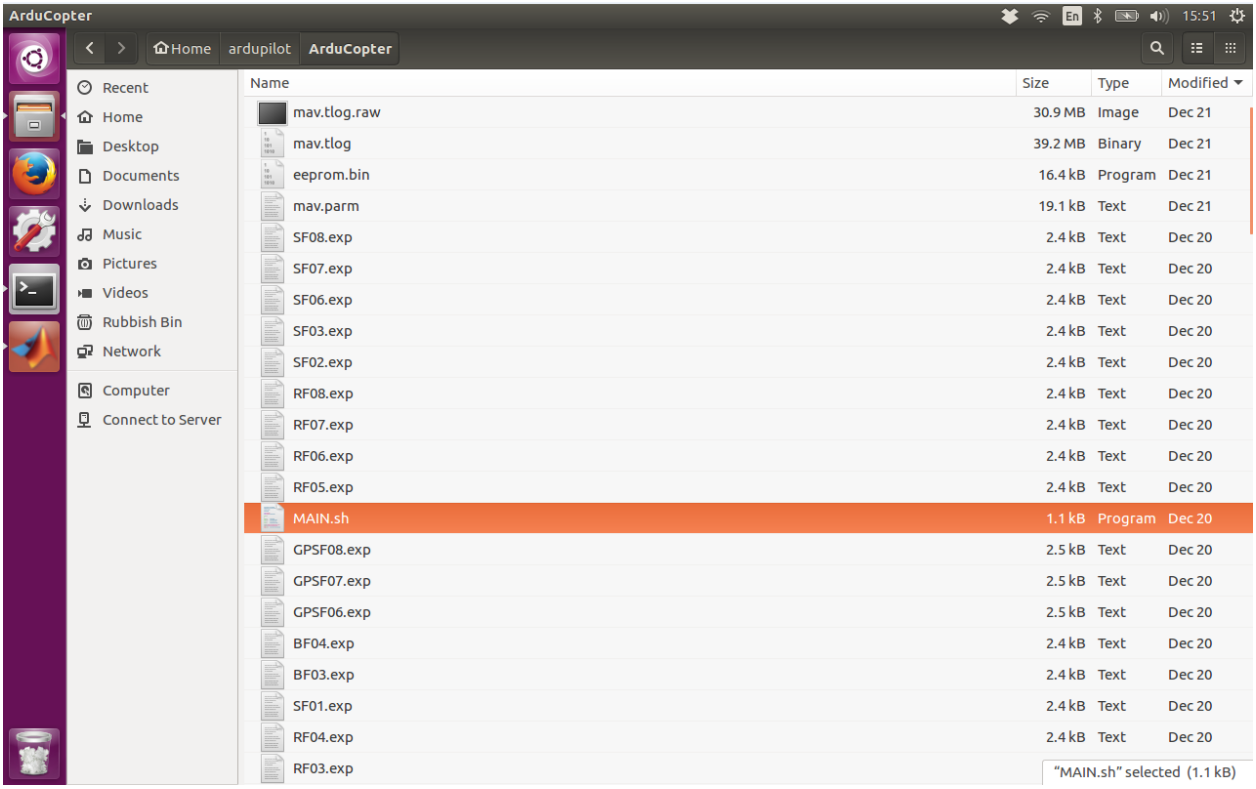

Figure 8 ArduCopter Directory

Open a bash terminal and direct it to the ArduCopter directory (**~/ardupilot/ArduCopter)** and type:

```
./MAIN.sh
```

The software will then run and execute all the scripts as shown in Figure 9.

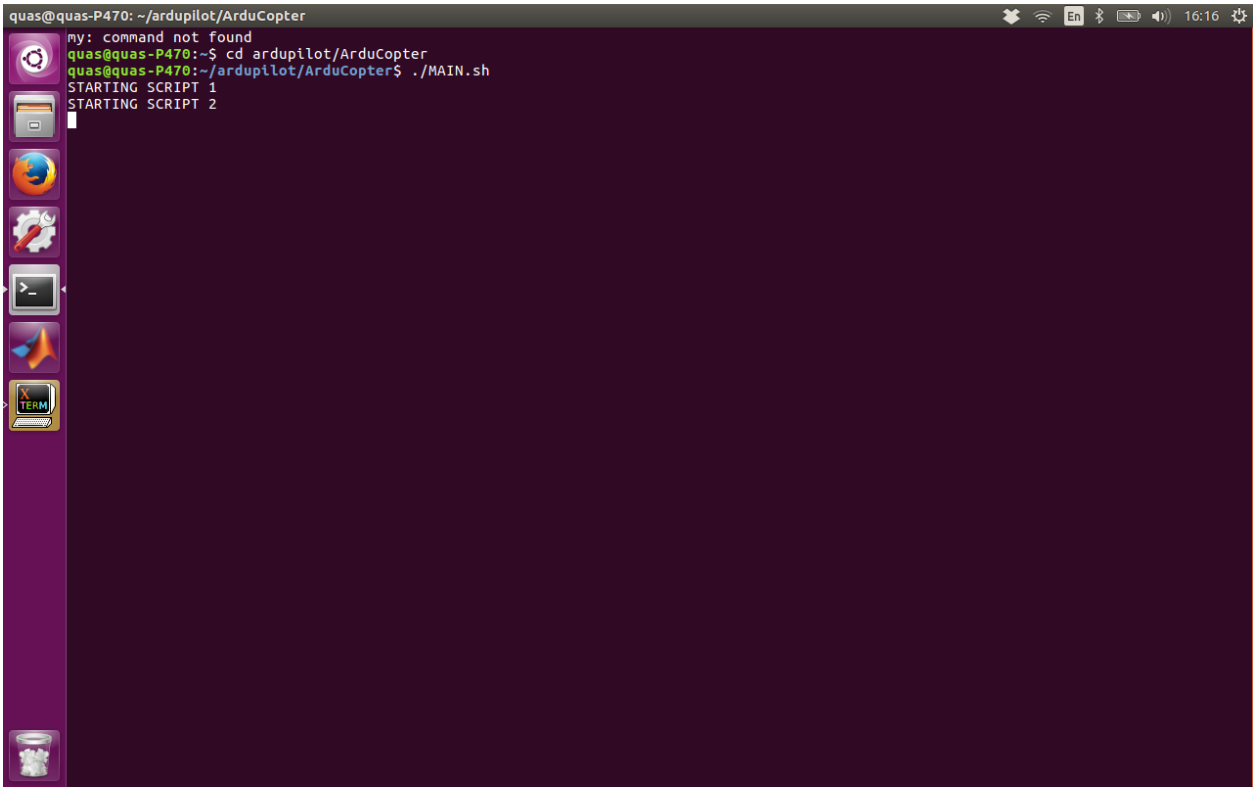

Figure 9 Software User Interface

The software will store all the logs in Ardupilot_Logs folder in Dropbox. (**~/Dropbox/ArduPilot_Logs**) as shown in Figure 10.

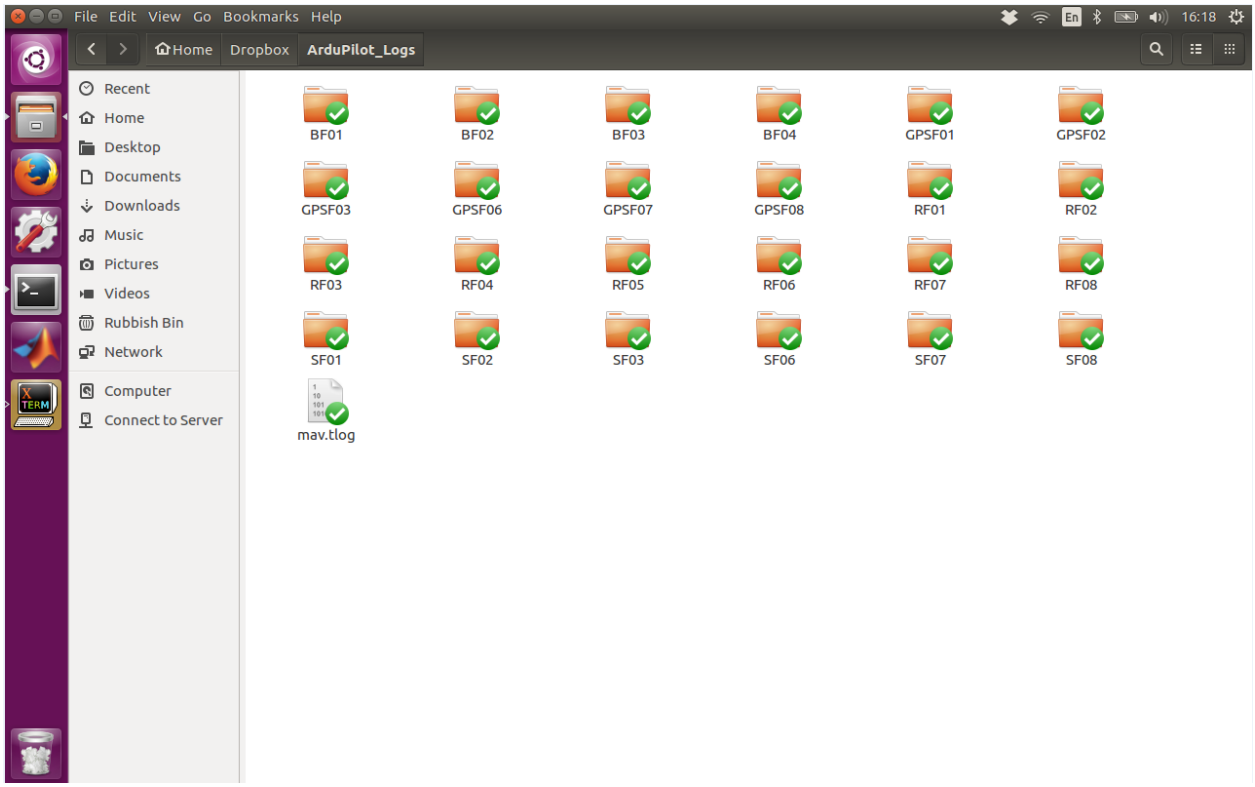

Figure 10 ArduPilot_Logs

8.2 Operating in Different Linux Machine:

If you would like to use the same ardupilot repository that was used during the development of SITL, copy the ardupilot folder in Dropbox **(~/Dropbox/ardupilot)** to your home **(~/ardupilot)**.

If you would like to use a new version, then refer to section 7.1. However, make sure ardupilot repository is in your home directory **(~/ardupilot)**

After that, copy the files in ArduPilot_Files folder to the ArduCopter folder **(~/ardupilot/ArduCopter).**

Finally, follow the procedure in section 8.1.

9. Important Notices:

9.1. General:

1. This software was developed to have time counter and battery detection functions. The counter function will forcefully end the test and start the next one after 15 minutes, also, it will forcefully end the test if the battery reached 0%. These two events are considered as crash. SITL will indicate a timer out event by titling the flight log with starting T (T_SIM_TITLE_xx.tlog) and battery ran out event with a starting B (B_SIM_TITLE_xx.tlog), and in the event of a flight finishing successfully SITL will title the flight log with no starting letters (SIM_TITLE_xx.tlog).

2. You might not find Ardupilot repository in Dropbox folder. Please install a new repository in this case.

3. The software was NOT developed to run in Windows operation systems, as Expect package was found to cause various compatibility errors.

4. Ardupilot is open source simulator, it is continuously developed and core changes may occur in the future. If you found computability issues and wish to use the same development repository, use the repository in Dropbox, or download the 12/2016 repository from Github.

5. With the minimum specifications in section 4, SITL was found to complete 337 flight tests, after that, the processor got exhausted and SITL froze. Therefore, it is advised to use better performing Linux machine than the one specified in section 4.

6. If you want to perform big number of tests > 100, it is highly advised that you restart your Linux machine to ensure maximum RAM and processor capability.

7. If you would like to see the commands being executed while SITL is running, run the generated script file directly instead of MAIN.sh. However, make sure to give the script file permission to be executed.

```
./RF01.exp
```

9.2 SIL.exp:

1. Always keep a backup copy of this file.
2. Ensure there are no empty lines in SIL.exp file. Expect_Gen.m function will only read filled lines.
3. The time counter value can be changed to allow for longer or shorter duration tests. The code to edit the time counter value can be found in line [**14**]

```
4  set timeout -1
5  spawn bash
6  set bash $spawn_id
7  spawn sim_vehicle.py
8  set sim $spawn_id
9  set ::original [clock seconds]
10 proc updatethetime {} {
11 set now [clock seconds]
12 set Timer [expr {$now-$::original}]
13 after 500 updatethetime
14 if {$Timer > 900} {
15 send -- "Timer Finished - Reseting\r"
16 }
17 }
18 updatethetime
```

Figure 11 Timer Function

4. Resetting the parameters to default before and after each test is vital. If parameters are not reset to default, the next test will be influenced by the pervious test parameters. The code for resetting the parameters can be found in line [**27, 41, 55, 102**].
5. A 5 seconds time delay was implemented before and after resetting the parameters to default. This is to allow enough time for SITL to keep responding to the commands efficiently. In case of no delay, the logging data will not be accurate. The delays can be found in lines [**26, 29, 40, 43, 54, 57, 101, 103**].
6. The delays in lines 101, and 103 were set to 1 second only; this is to ensure Expect doesn't miss reading any output expressions in the terminal while the computer is sleeping.
7. It is important to de-spawn bash and Ardupilot after every test. This is to prevent consuming all the pipes which will cause the computer to stop the test. This code can be found in lines [**35,36,37,49,50,51,63,64,65**]
8. Killing Mavlink connection after every test is important to prevent exhausting the computer, a bash file was introduced KillXterem.sh to kill Mavlink after each test. The code can be found in line [**34,48,62**]

9. Help Resources

If you find any issue or enquiry about this document or the software, I advise you to utilize these resources:

1. Exploring Expect Book by O'Reilly & Associates, INC. (Available at QUT Library)
2. ArduCopter Website
3. TCL tutorial
4. Contact me: osamaibrahimsuliman.yahya@connect.qut.edu.au **[NOT GUARANTEED]**