eYSIP-2018

Multiple Drone Control

Sunil Kumar
Shruti Joshi
Vikrant Fernandes, Simranjeet Bhangu
Duration of Internship: 22/05/2018 – 07/07/2018

2018, e-Yantra Publication

Multiple Drone Control

Abstract

This project involves controlling more than one drone connected in the same network. The drones can also be made to perform coordinated motion where each drone follows a specific path relative to other drones in the network. Several coordinated tasks have been presented and the results are verified by simulations on Gazebo, where certain amount of communication delay between the drones is also considered. Pluto drones are mounted with esp8266 Wi-Fi module for communication. Each drone in the multiple drone network acts as a client. Multiple nodes are developed to communicate, localize and control the drones. For each drone in the network a unique topic is defined. Drone commands are published on this topic while each drone subscribes to its unique topic and sets its parameters accordingly. For localization, whycon markers are mounted on each drone. The drones sensor values are published on a topic after reading from socket buffer to use further.
Robot Operating System:
For multiple nodes to communicate, localize and control the drone, ROS environment is used. For each drone in the network, a unique topic is defined. Drone commands are published on this topic while each drone subscribes to its unique topic and sets its parameters accordingly.

Completion status

- Implemented PID on AR-Drone model in Gazebo
- Established communication between the drones and processing device(Ground Control System).
- Accomplished individual thread feedback control for each drone.
- Position hold of multiple drones.

- Coordinated motion of all drones accomplished to form patterns.
- Studied latency in the network and it's dependency on the number of drones in the network.

1.1 Hardware parts

- **Pluto drone by Drona Aviation Pvt. Ltd.**
  - STM32F103C8 Micro-controller
    Datasheet — Reference Manual
  - MPU9250 Accelerometer and Gyroscope
    Datasheet
  - AK8963 Magnetometer
    Datasheet
  - MS5611 Barometer
    Datasheet
  - ESP8266 Wi-Fi Wireless Module
  - Coreless Motors and Propellers
  - 3.7V Li-Po Battery
- **Wide angle camera**
  - **ROS compatible**
  - **Resolution**: 640 x 480
  - 30 frames per second

1.2 Software used

- **Robot Operating System**
- Detail of software: Indigo, download link
- **Opencv**
- Details of software: Version 3, download link
- **Gazebo**
- Details of software: Gazebo version 2.2.3 download link

1.3 Assembly of hardware

Circuit Diagram

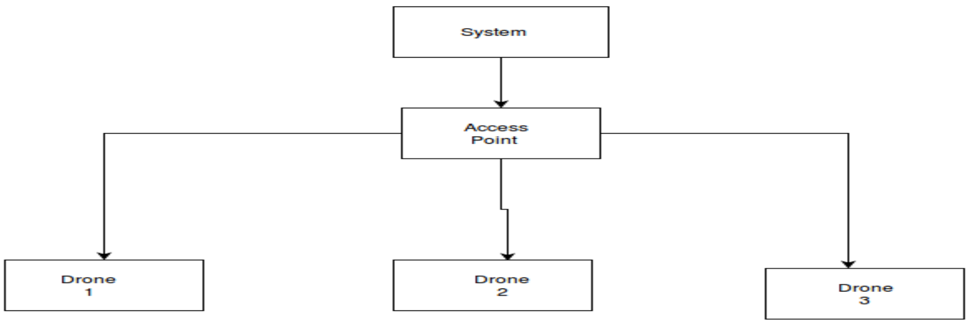


figure 1: Block diagram of the Network.

The figure 1 is a block diagram for connection of multiple drones to a common network. The Ground Control System and the drones are connected to the same access point. For each drone, a socket is created with drone IP address and port 23.

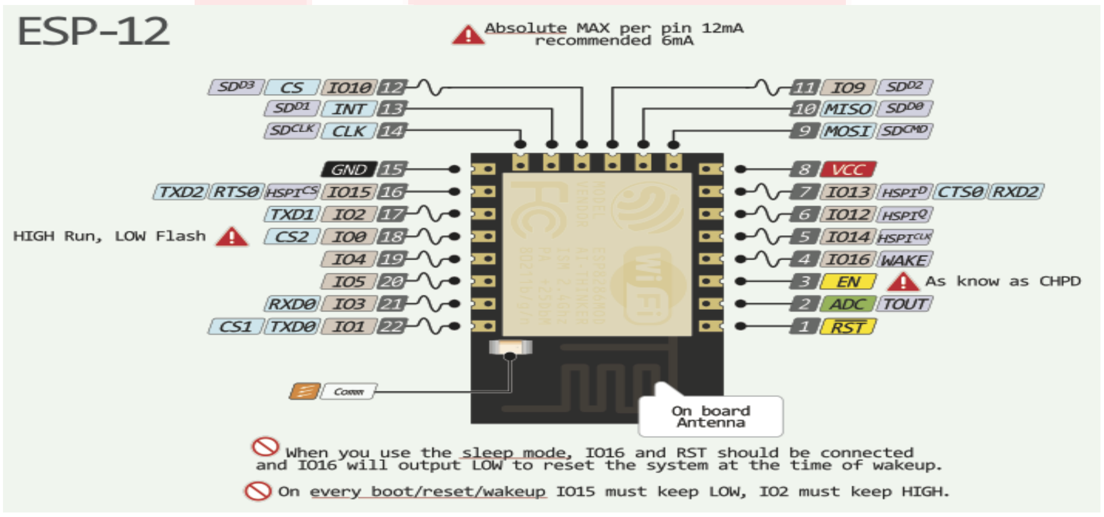


figure 2: ESP 8266 working with an ESP12 module.

Communication takes place over Wi-Fi, using ESP8266 Wi-Fi modules mounted on the drones.

1.4 Multiple drone Communication

Each drone is connected to an AP and Ground control station(Laptop) is also connected to same AP. IP address of each drone is identified and marked. Individual thread is implemented in C++ to communicate with each drone separately with their unique socket. Individual thread is created for every operation like read, write etc. for each drone.

Sensor data of each drone is published on individual topics. The topics also run on individual threads in service which receives the sensor data as request form C++ nodes.

1.5 Multiple drone Feedback Control

A feedback control system is run on individual thread for each drone. Each thread reads the position of a particular drone from global variables and calculate pitch, roll, yaw and throttle for each drone for a fixed target position. Each thread is initialized at start-up for its associated drone with its PID values and drone number as argument. Drone number is used to mark particular IP of a particular drone.

1.6 Whycon Marker Identification

Each drone has identical whycon marker. Hence to avoid mixing of marker, each drone is assigned an approximate fixed position at start-up according to IP address of the drone. Once the script starts, the previous position of the respective drone is used to track the drone individually irrespective to position of the drone. Each marker values are compared with previous marker position and right values are assigned to the global list depending on the error between previous and current positions.

1.7 Drawing drone trajectory

To show path traversed by drone every position of marker is stored in an array and all pixels at that positions are colored with a unique color. Hence this color shows the path traversed by the drone.

1.8 Path planning for Pattern

To traverse drone in a specific path, target co-ordinates of each drone must be changed by taking feedback from each drone's position in arena. The position calculation is done in the main thread of the program. There are different patterns implemented:-

- **Rotating a drone in circle around a drone hold at center**
  Calculating the circle arc co-ordinates at 10' difference. When drone reached at a desired point then the target of the drone is shifted to the next co-ordinates of the circle arc. Hence the drone completes a circular trajectory in a total of 36 drone coordinates.
  Co-ordinate calculation:-

  x=r$\times sin(x \times 10)$
  $y = r \times cos(x \times 10)$

where x range from 0 to 36 and r is radius of the circle.

- **Rotating two drones on arc of a square opposite to each other, holding a drone at center**
Calculating the vertices of a square of defined size. Two drones traverse to vertices of square opposite to each other. A drone is hold at center.

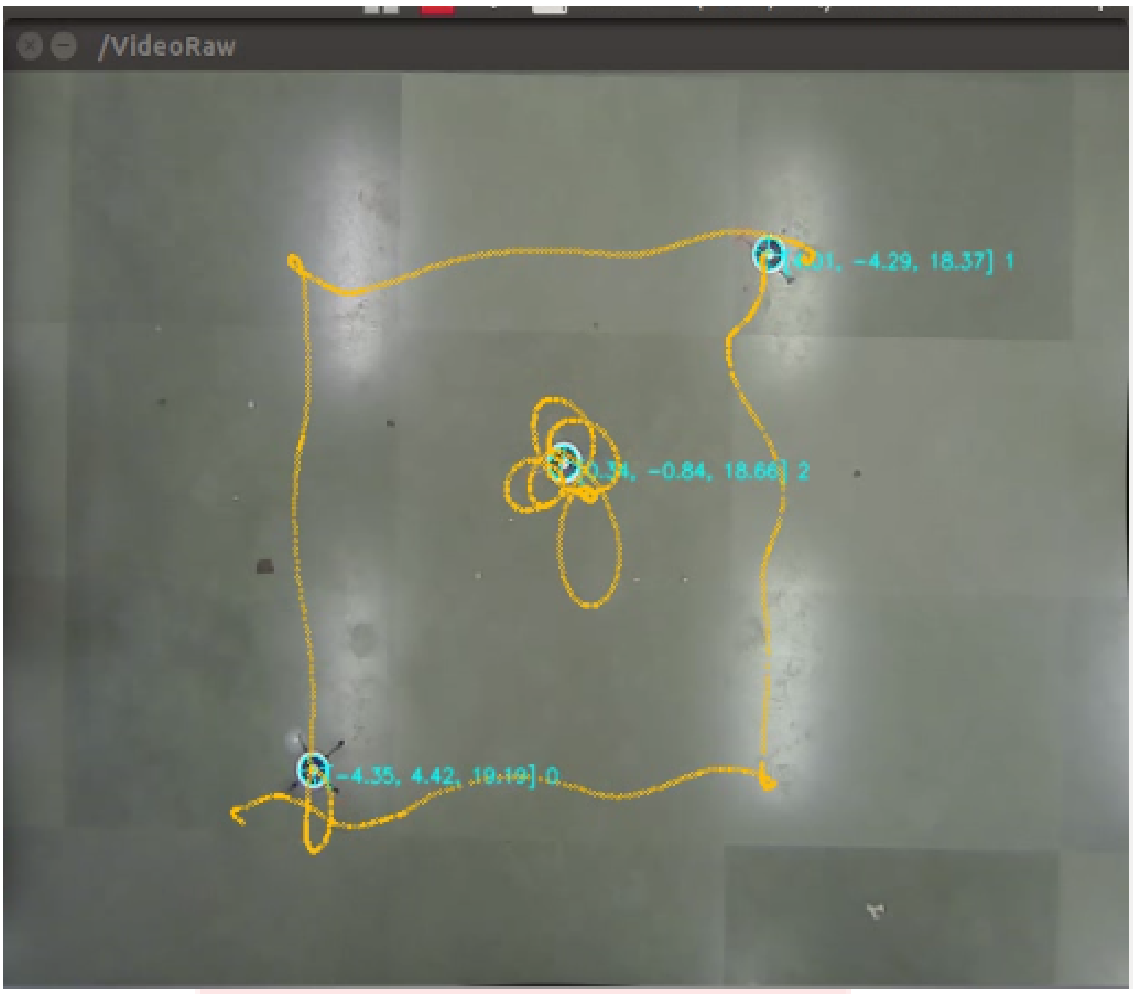


figure 3: Square pattern drawing with three drones.

- **Rotating three drones on arc of a circle at a 120' to each other**
  Defining initial position of the drone as the arc co-ordinates of a circle at 120' each. The new co-ordinates are calculated as the arc of the circle 120' apart.

  $x_0 = r \times sin(x \times 10)$
  $y_0 = r \times cos(x \times 10)$
  $x_1 = r \times sin((x \times 10 + 120)\%360)$
  $y_1 = r \times cos((x \times 10 + 120)\%360)$
  $x_2 = r \times sin((x \times 10 + 240)\%360)$
  $y_2 = r \times cos((x \times 10 + 240)\%360)$

  where x is integer having value from 0 to 36 and r is radius of the circle.

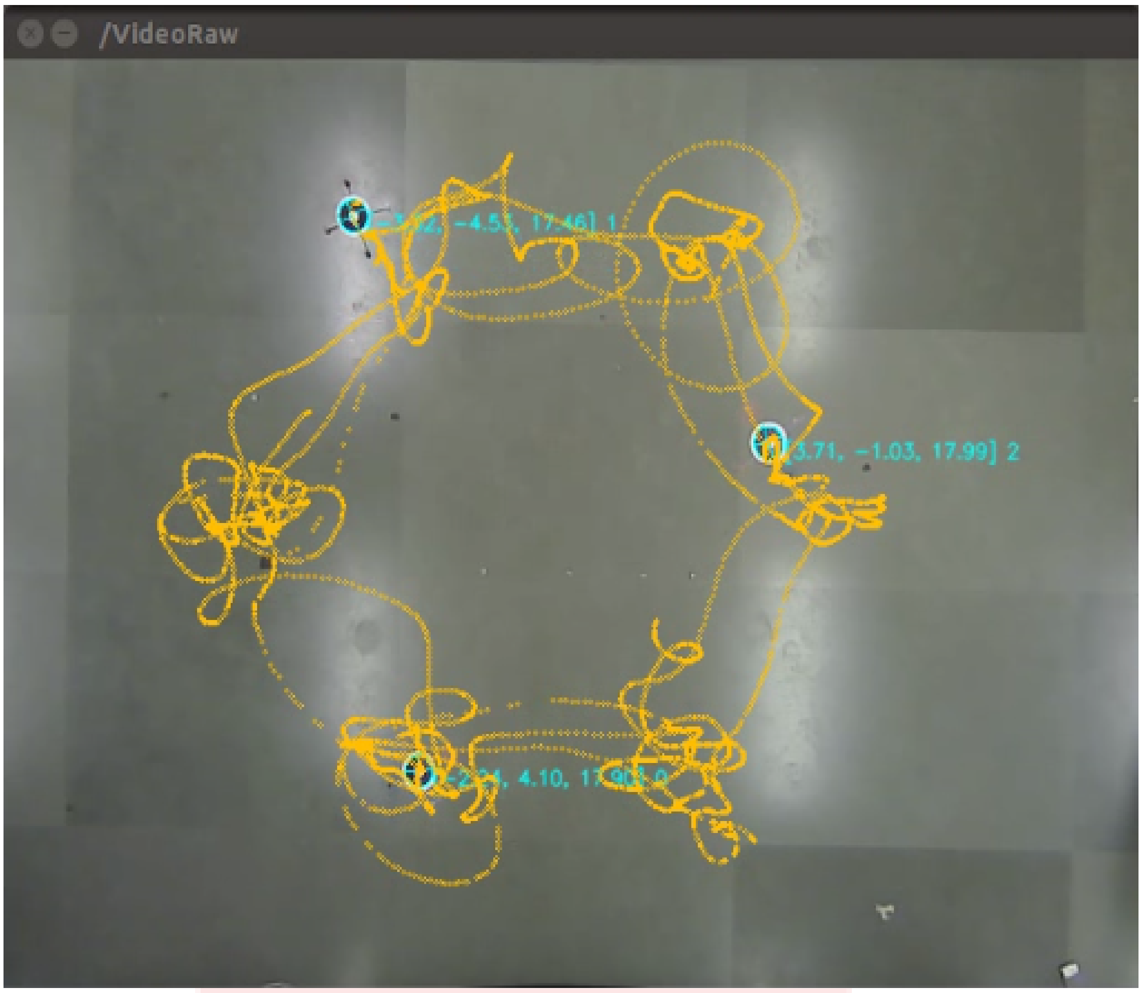


figure 4: Circle drawing using three drones.

1.9 Software and Code

The code can be found here.

- branch to working_branch.
- The directory plutoserver contains scripts to receive data from the drones (data_via_rosservice.py), and to control the drones (multi_drone_control.py)

1.10 Latency

The delay which a device take to give response after giving a certain command is known as latency. There are different factors which cause latency in the myltidrone control system i.e. :-

- **Latency due to image processing** Image is processed using whycon package to detect the markers in the image. Whycon uses openCV library to detect the marker using flood fill algorithm. Hence it takes an average of 12 microseconds to detect the marker and determine the relative position of the markers.
- **Latency due to marker splitting process** Every drone has same type of marker hence it is very difficult to keep eye on individual drone when the drones move from their position. Hence, the initial position of the drone is fixed in arena and a previous position of drone is saved in every iteration to calculate nearest previous position of drone to determine the current positions of the drones. It takes 27 microseconds(approx) to calculate positions of each drone.
- **Latency due to feedback control loop for each drone** Parallel PID is used to control the roll, pitch, yaw and throttle of drone. The feedback control loop for each drone takes 100 microseconds(approx).
- **Latency due to path planning** Path planning algorithm run in the main thread. This thread also take significant resources to compare the relative position of each drone and assign target position accordingly. Hence this thread effect on the scheduling latency of other threads.
- **Latency in processing drone command to make MSP packet** After calculating roll, pitch, yaw and throttle by feedback control loop, this data is converted to MSP packet to send it through the network. Firmware on the drone receive the MSP packet and decode it according to the MSP protocol. The conversion from raw data to MSP packet takes 190 microseconds(approx).
- **Latency in communication** The communication medium is wireless and the topology used is star network i.e. MSP packets first go to router and then routed to the drone. Hence the communication latency is 4 milliseconds(approx).
- **Latency in effect of command to drone** After sending command to drone it takes some time to take effect because micro-controller on

drone also decode the packet and change the speed of motors accordingly after a lot of calculation from the difference sensors. Hence it takes 100 milliseconds(approx) to a command to take effect on drone.

- **Latency due to thread scheduling** Every thread has some clock assigned by scheduling algorithm frequently. The frequency of the time slot assignment to thread depends on the no of threads running on the system and requirement and priority of each thread. Hence the latency due to each of the process can vary due to scheduling of thread. Variance in latency in feedback control thread due to scheduling :-

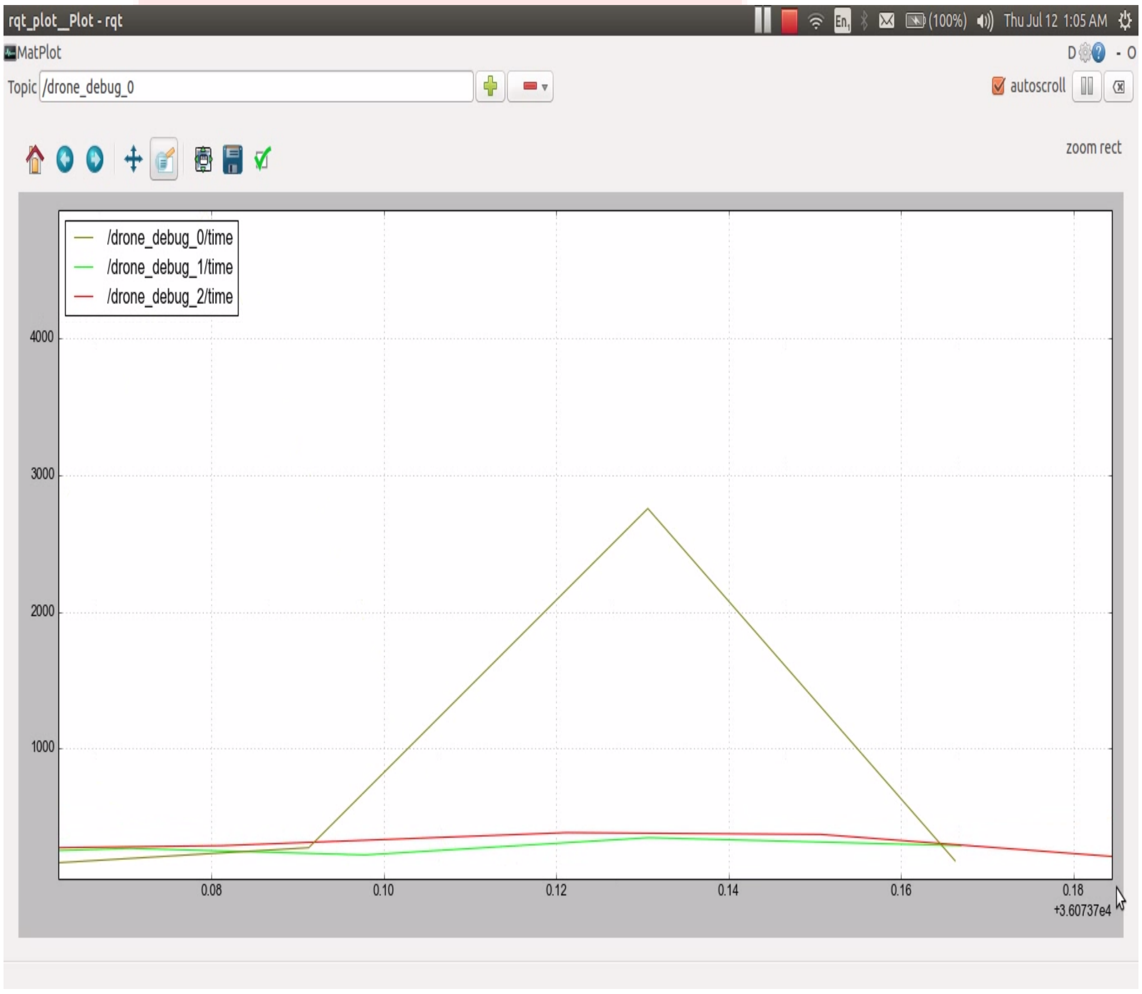


figure 5: Latency due to scheduling.

1.11 Use and Demo

Youtube Link of demonstration video part 1.
Youtube Link of demonstration video part 2.

1.12 Future Work

- **Study the factors affecting latency**
  One of the major factors that affect the system is latency in the network. The time taken for execution of multiple threads is not constant. As a future objective, we will study the factors that affect the time taken taken by different parts of the program.

- **Exploring Drone-Drone communication**
  An improvement in the system could be brought about by introducing inter-drone communication in the network. We have implemented a star network. We can work further to form drone mesh networks.

1.13 Challenges

There were many challenges we faced during the implementation. To differentiate the drones having same type of marker was also a challenge. We overcame this challenge by comparing the marker's current position with the previous position of markers. Tuning PID of every drone individually and running in an individual thread having different PID values was also a challenge. Most pain-full challenge was due to scheduling of threads. Because OS give time slot to different thread according to available resources but the control threads calculation must be done in real time. This problem can be solved by either using RTOS(Real Time Operating System) or by decreasing number of threads such that the threads get time slot very frequently which is rarely possible in a commercial OS. Fluctuation latency due to scheduling of threads are:-

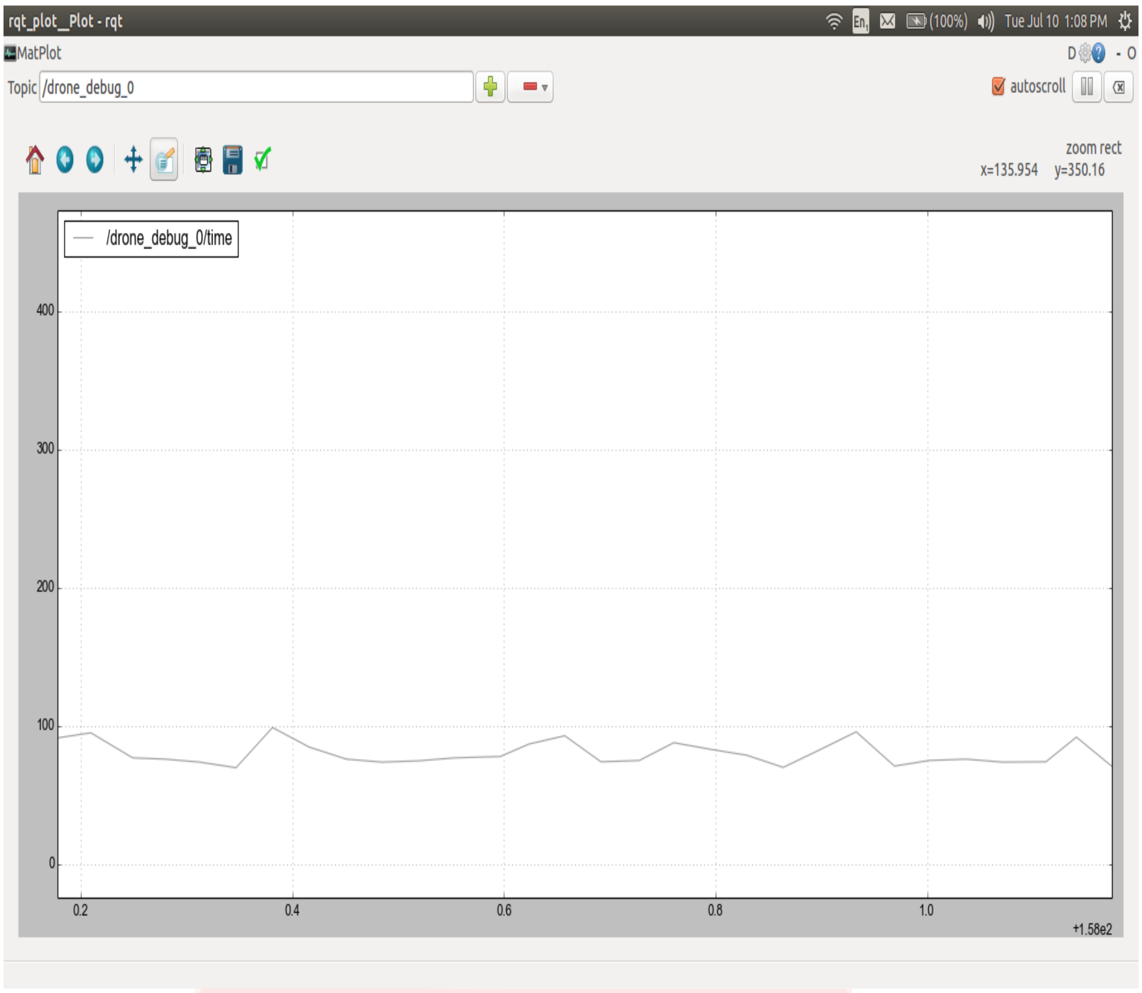


figure 6: Latency when one drone is controlled.

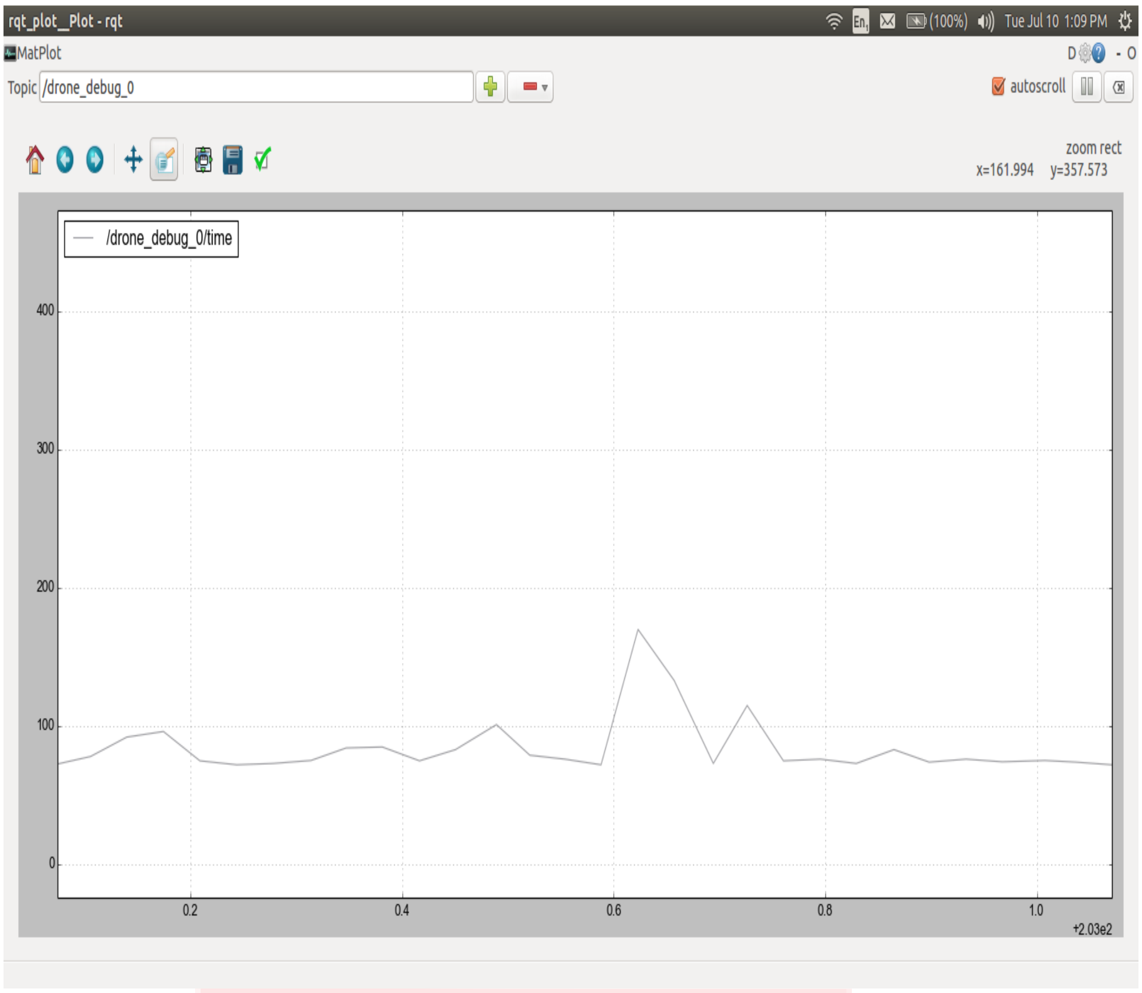


figure 7: Latency when one drone is controlled.

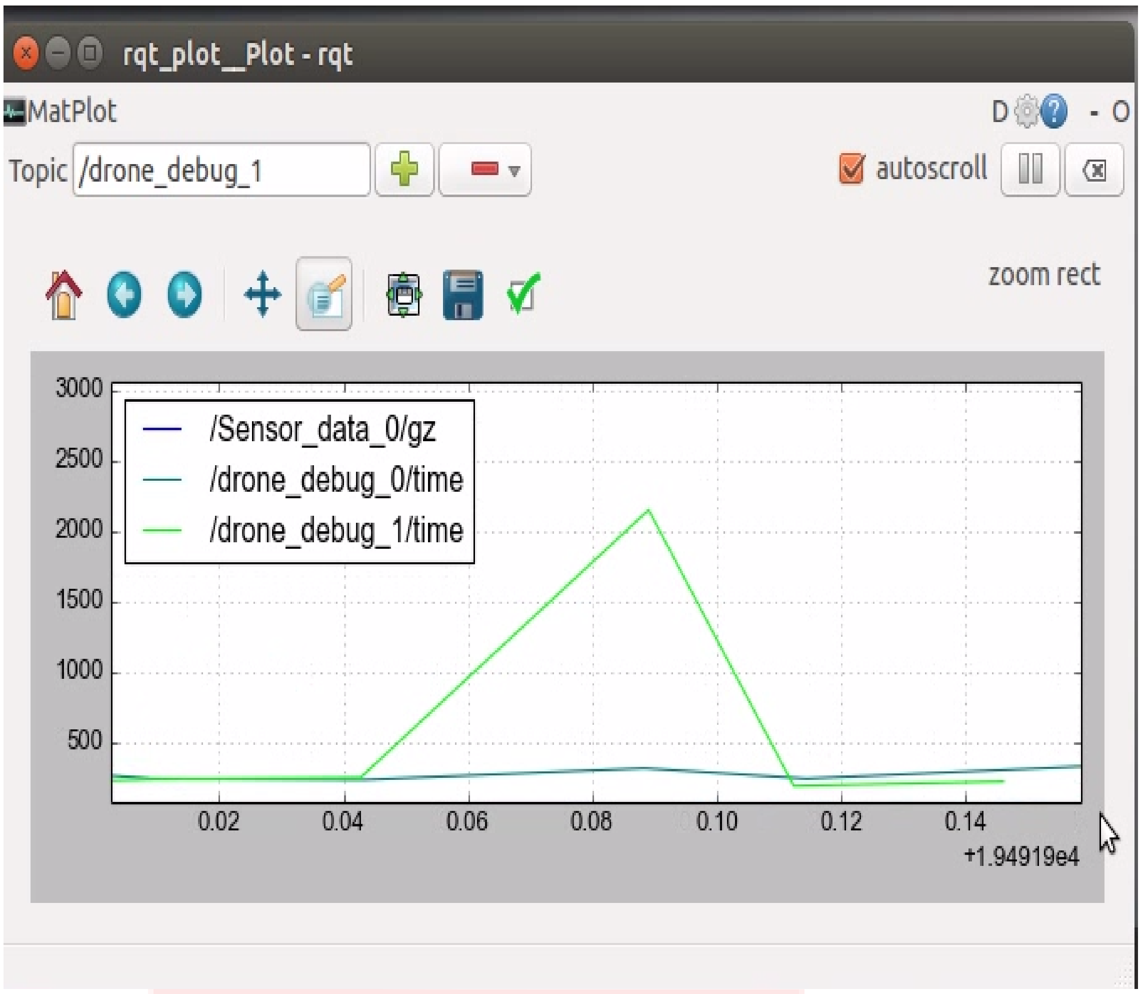


figure 8: Latency when two drones are controlled.

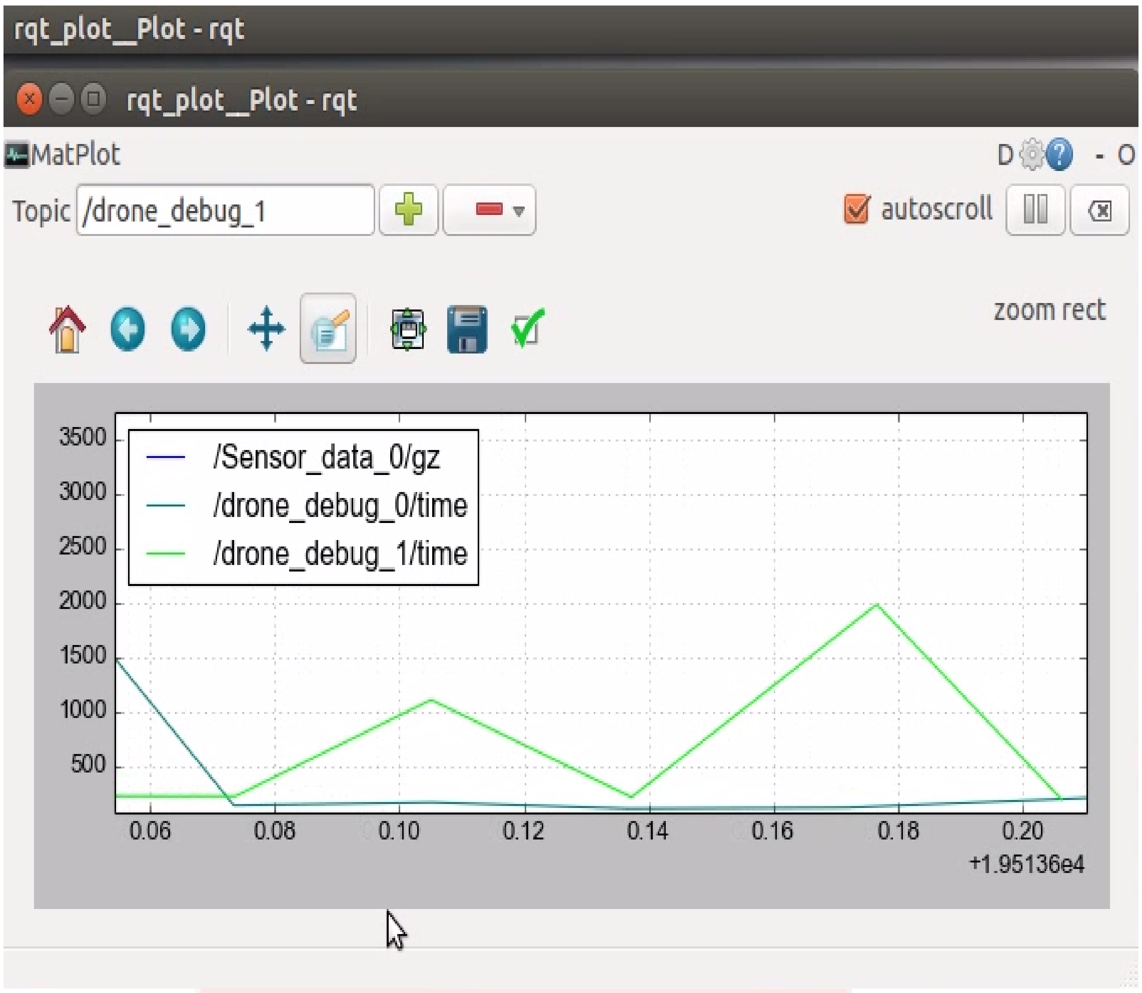


figure 9: Latency when two drones are controlled.

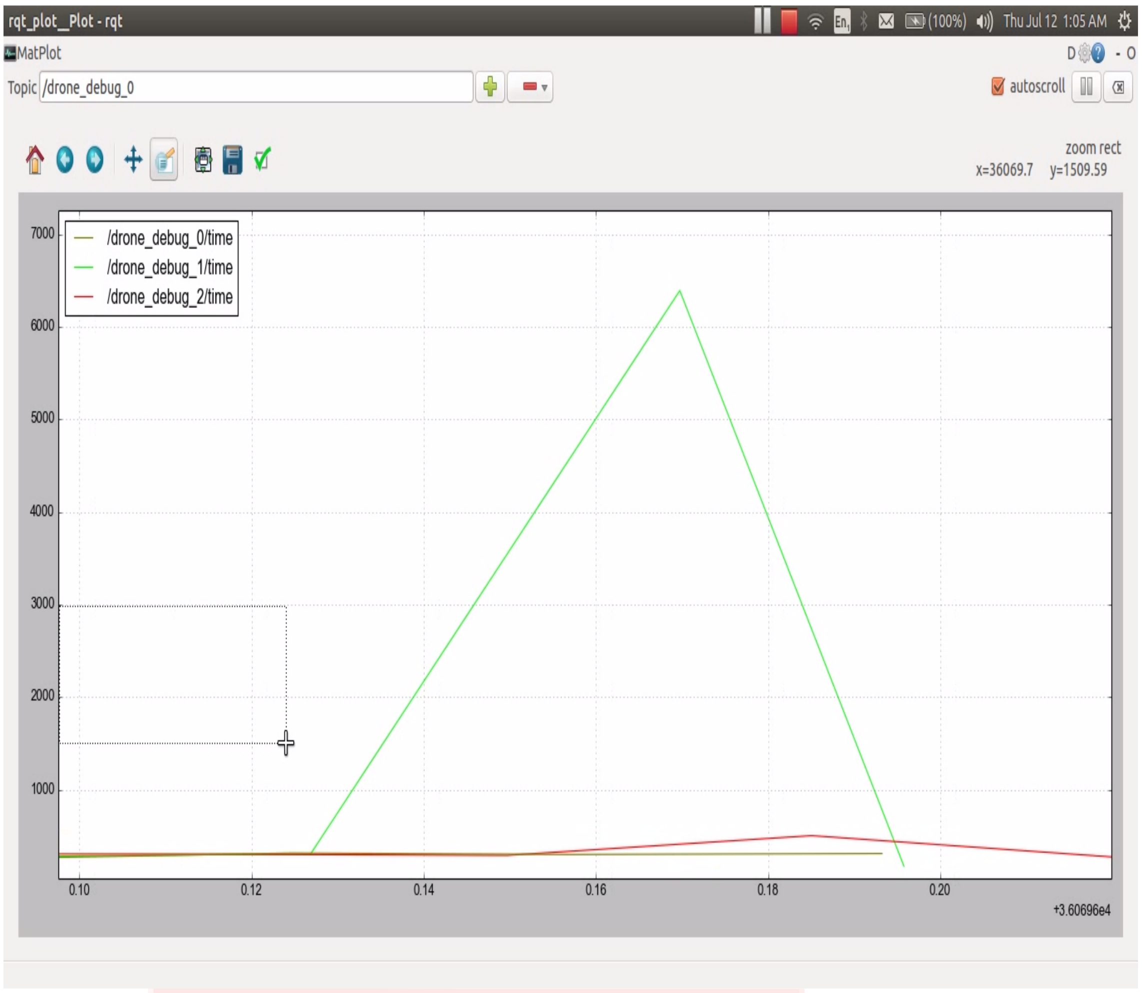


figure 10: Latency when three drones are controlled.

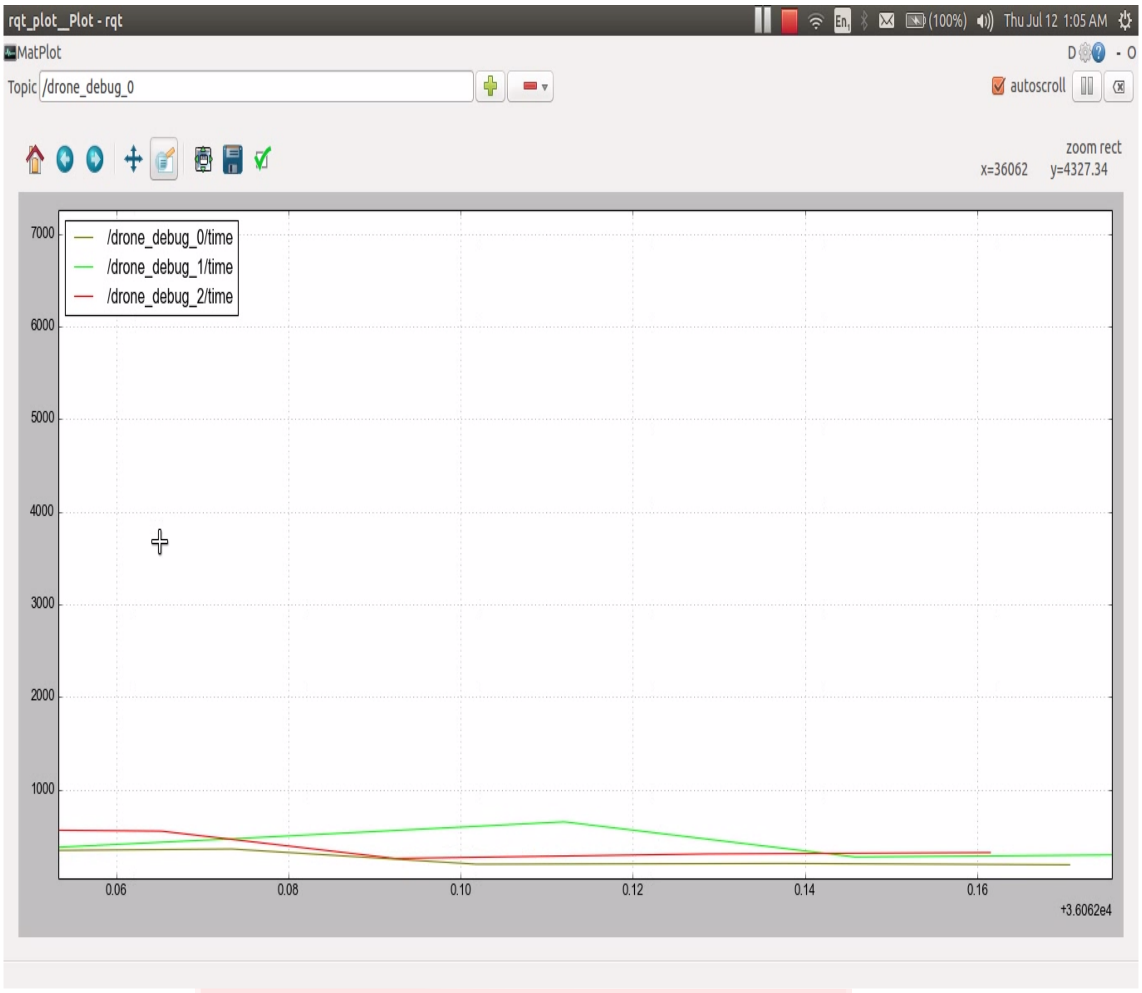
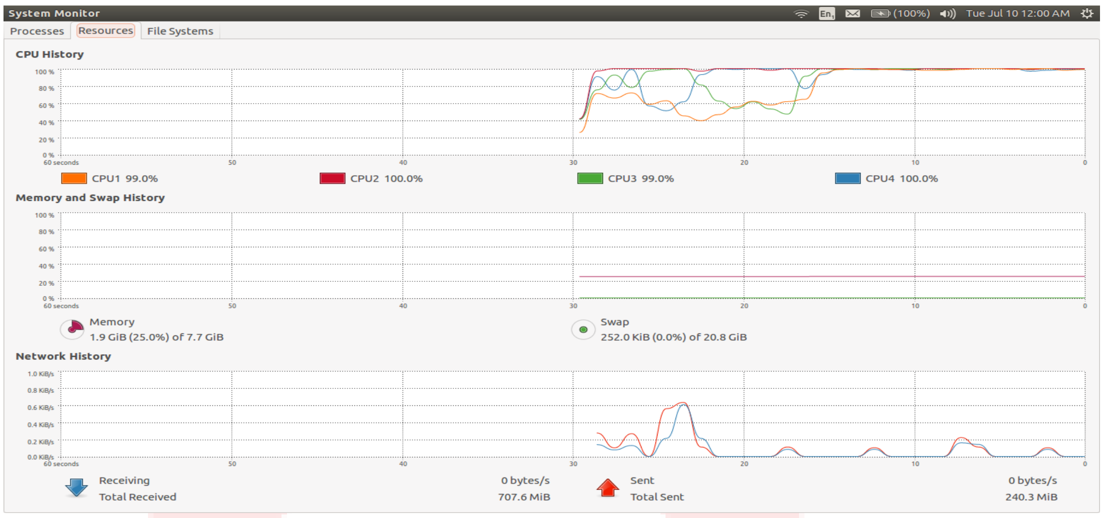


figure 11: Latency when three drones are controlled.



🟧 CPU1 99.0%
🟥 CPU2 100.0%
🟩 CPU3 99.0%
🟦 CPU4 100.0%

Memory
1.9 GiB (25.0%) of 7.7 GiB

Swap
252.0 KiB (0.0%) of 20.8 GiB

Receiving 0 bytes/s
Total Received 707.6 MiB

Sent 0 bytes/s
Total Sent 240.3 MiB

figure 12: CPU usage as the code runs.