Launching Multiple AR-Drones Using ROS Guide

Made by Or Hanoch for MSS at the University of the Witwatersrand

June 28, 2018

Contents

0 Introduction

This guide assumes you have configured your router to have a local LAN distributing its own IP address using DHCP. Instructions on this using a NETGEAR Nighthawk R7000 can be found in a separate guide titled "NETGEAR Nighthawk (R7000) Router Setup Guide For AR-Drone Use (Multiple Drones)".

This guide will show you how to configure the AR-Drones themselves to connect to our local LAN and how to make a launch file in order to launch them using ROS. This guide was tested using ROS kinetic on Ubuntu 16.04.

This guide was mostly taken from:
https://github.com/AutonomyLab/ardrone_autonomy/wiki/Multiple-AR-Drones

1 Starting State

In this guide we continue directly from the "NETGEAR Nighthawk (R7000) Router Setup Guide For AR-Drone Use (Multiple Drones)" guide. This starts us off with the following:

- We have a router configured with its own local LAN and distributing IP addresses using DHCP.
- The name of the routers network is "dronenet"
- Our PC is connected to the router (and its local LAN) via Ethernet and has no other network connection.
- For this guide we will also need a separate computer capable of connecting to the drones wifi network.

2 Drone Configuration

Note:
This section requires you to have a separate computer with wifi.
This is due to your router not being configured to bridge the connection.

2.1 Changing Drone Settings

1. Connect to your AR-Drone's ad-hoc wifi (directly to the drones wifi router, like you would connect to any wifi connection)
2. Open the terminal and telnet to the drone:
   (a) Run:
   ```
   telnet 192.168.1.1
   ```
3. create a file called wifi.sh in /data of the drone:
   ```
   vi /data/wifi.sh
   ```
4. Copy the following into the file:

```
killall udhcpd
ifconfig ath0 down
iwconfig ath0 mode managed essid dronenet
ifconfig ath0 192.168.1.10 netmask 255.255.255.0 up
```

Notice:

- In line 3 change "dronenet" to the SSID you chose when setting up the router.
- The IP in line 4 should be unique to each drone, so change it between different drones.

If you are not familiar with vi:

- To enter insert mode in vi press "i".
- To save and exit vi press "esc" to exit insert mode and then ":wq")

5. Make the newly created file executable:

   ```
   chmod +x /data/wifi.sh
   ```

6. Close the telnet connection:

   ```
   exit
   ```

2.2 Run wifi.sh from PC:

Notice:
While the previouse sections should only be done once, this step should be done whenever you want to make the AR-Drone connect to your wireless (for example every time you turn on the drone)

1. Connect to your AR-Drone's ad-hoc wifi (directly to the drones wifi router, like you would connect to any wifi connection)
2. Open a terminal
3. Execute the following command to remotly run wifi.sh on the drone:

   ```
   echo "./data/wifi.sh" | telnet 192.168.1.1
   ```

   (a) Your computer should now disconnect from the drones network. This is due to the drone's network configurations being changed.

4. On the computer that is connected to the previously configured routers wireless network check that you have connection to the drone using ping:

   ```
   ping 192.168.1.10
   ```

   (a) Where the IP address is the same as the one configured in wifi.sh of this drone.

3 Run the ROS Driver (can be skipped)

When connecting to the drone using ROS you will need to make sure to supply the correct drone IP to ardrone_autonomy. This can be done by:

1. If running ardrone_autonomy directly in the command line use:

   ```
   rosrun ardrone_autonomy ardrone_driver -ip 192.168.1.10
   ```

2. If using a launch file add/change in the node line the argument:

   (a) `args="-ip 192.168.1.10"`

   Thus you will end up with something like:

   ```
   <node name="ardrone_driver" pkg="ardrone_autonomy"
   type="ardrone_driver" args="-ip 192.168.1.10" />
   ```

   Notice:

   - The IP address in the above examples should be the same as the one configured in the wifi.sh file of this drone.
   - Make sure to give the different nodes for different drones different names, i.e:

     ```
     <node name="ardrone_driver1" pkg="ardrone_autonomy"
     type="ardrone_driver" args="-ip 192.168.1.10" />
     ```

     and

     ```
     <node name="ardrone_driver2" pkg="ardrone_autonomy"
     type="ardrone_driver" args="-ip 192.168.1.12" />
     ```

     If using two drones, one with the ip 192.168.1.10 called "ardrone_driver1" and the other with ip 192.168.1.12 called "ardrone_driver2"

   (b) You will notice that even though you are controlling multiple drones now, you can't send them separate commands. If you run

   ```
   rostopic list
   ```

   You will see that the different drones do not have separate topics. In order to get separate topics for each drone you need to remap the topics in the launch files, writing above each drones node:

   ```
   <remap from="/name/of/topic" to="/new/name/of/topic"/>
   ```

   i.e to change the /ardrone/takeoff topic to /ardrone1/takeoff:

   ```
   <remap from="/ardrone/takeoff" to="/ardrone1/takeoff"/>
   ```

   By remapping the topics to different names for each drone you will now have control of each drone via its own topics.

   (c) Run using:

   ```
   roslaunch <name_of_launch_file>
   ```

4 Script For Launching Multiple Drones Easily

If using multiple drones it can be easier to use a launch file that launches a single drone, and edit it to remap the topics to a given name. Thus you can run that file with a script multiple times in order to launch multiple drones easily, and you won't need to edit the launch file itself to a ridiculous extent. The script is a simple for loop that calls the launch file with different names and IPs as parameters, and the launch file takes those parameters as inputs and launches a single drone using them.

- Lanuch file - launch_multi_drone.launch:

```
<launch>
        <!-- declare variables to be taken in from parameters -->
        <arg name="droneip" default="192.168.1.1" />
        <arg name="drone_name" default="ardrone"/>

        <!-- remap ardrone topics to specific topics per drone -->
        <remap from="/ardrone/land" to="/$(arg drone_name)/land"/>
        .
        . About 30 remaps. run `rostopic list` to get a list of topics.
        .
        <remap from="/ardrone/takeoff" to="/$(arg drone_name)/takeoff"/>
        <remap from="cmd_vel" to="/$(arg drone_name)/cmd_vel"/>

        <!-- create node for drone -->
        <node name="$(arg drone_name)_driver1" pkg="ardrone_autonomy"
        type="ardrone_driver" output="screen" args="-ip $(arg droneip)">
                <param name="navdata_demo" value="False" />
                <param name="realtime_navdata" value="True" />
                <param name="realtime_video" value="True" />
                <param name="looprate" value="30" />
        </node>
</launch>
```

- Run BASH script - launch_multi_drone.sh:

```
#!/bin/bash

# This bash script will run launch_multi_drone.launch with
# different drone names for the amount of drones specified.
# This will initialize nodes for the amount of drones specified
# as well as separate topics for each drone.
# Each drones topic will be of the form /ardrone#/<topic_name>.
# i.e /ardrone/takeoff => /ardrone42/takeoff for drone # 42
# usage: launch_multiple_drones.sh <num_of_drones>

for (( i=1; i<=$1; i++))
do
        roslaunch launch_multi_drone.launch\
                drone_name:=ardrone$i droneip:=192.168.1.$((9+$i)) &
        sleep 15
done
```

- Save both files in the same directory
- To run the script:

  launch_multiple_drones.sh <num_of_drones>

  i.e for 2 drones:

  launch_multiple_drones.sh 2