International Micro Air Vehicle Conference and Flight Competition (IMAV2013)
17-20 September 2013, Toulouse, France

Paparazzi: how to make a swarm of Parrot AR Drones fly autonomously based on GPS.

Bart Remes, Dino Hensen, Freek van Tienen, Christophe De Wagter,
Erik van der Horst, and Guido de Croon

Micro Air Vehicle laboratory, Delft University of Technology, Kluyverweg 1, 2629 HS, Delft, the Netherlands.
Contact: microuav@gmail.com

Abstract

The Parrot AR drone 2 is a commercially available quad rotor with many sensors onboard, such as two cameras, accelerometers, gyrometers, barometer, and a magnetometer. Recently, a GPS-module has become available, which can be used to geo-tag images. We have adapted the particularly versatile and powerful Paparazzi open source autopilot for use with the Parrot AR drone. Hence, just by uploading software to the drone, it is able to perform GPS-based autonomous outdoor flight. In this article, we explain the inner workings of the adapted autopilot and what procedure a user should follow for performing experiments with a swarm of AR drones.

1 Introduction

In the past decade, the fast developments in micro-electronics and the related price reductions have led to Micro Air Vehicles (MAVs) able to perform GPS-based outdoor flight. The low cost of the necessary components and the creation of open source autopilot projects give a growing group of hobbyists and researchers the opportunity to construct and fly their own MAVs [8]. Example projects include ArduPilot [1], OpenPilot [2] and Paparazzi (cf. [3, 6, 13]).

In the same time, toy planes and helicopters have become more and more sophisticated. As an ultimate example, one can think of the Parrot AR drone [4, 7], which is equipped with two cameras, accelerometers, gyros, sonar, and sufficient onboard processing to autonomously cancel drift if there is (visual) texture on the ground surface. The availability of these sensors, the relatively low price, and an open source Standard Development Kit (SDK) have resulted in the robotics community making extensive use of the Parrot AR drone for their experiments (cf. [5, 10, 14, 9]). However, these experiments have been limited to flights without GPS-based navigation.

In order to facilitate MAV research with GPS-enabled flight, we have recently adapted the particularly versatile and powerful Paparazzi autopilot for use with the Parrot AR drone. The autopilot works completely with the Parrot hardware, i.e., the drone itself and the GPS-module. Hence, no hardware modification is necessary[1].

In this technical paper, we first explain the straightforward procedure for enabling GPS-based flight with the Parrot AR drone (Section 2). Subsequently, we provide insight into the basic inner-loop and outer-loop control of the now GPS-enabled drone (Section 3), and explain how to set up an experiment with a swarm of drones (Section 4). We conclude in Section 5.

[1]Please note that the explained software project is not supported nor endorsed by Parrot S.A. Using the software will most likely void the warranty and use of the software is entirely at the user's own risk.

2 How to make the Parrot AR drone fly with GPS

There are two ways to make the AR drone fly with GPS: (1) enhancing the Parrot firmware with Paparazzi firmware (the *Paparazzi AR drone SDK* version), (2) only using Paparazzi firmware (the *Paparazzi AR drone raw* version). The difference between these two versions is explained in more detail in Section 3. For now, it is important to know that the raw version is also called the expert version, requiring some more work but also providing more control of what is happening onboard the AR drone. Below, we explain for both versions how Paparazzi can be installed. Please remark that the Paparazzi wiki has a more elaborate, up-to-date explanation of these procedures [2,3].

2.1 Paparazzi AR drone SDK version

The only requirements besides an AR Drone 2, are a Parrot GPS receiver and a laptop with Ubuntu Linux OS installed (a virtual machine can be used to this end). To be fully compatible you need at least ARDrone 2 software v2.4.3 updated on your AR Drone 2 and it is best to have Linux v12.04 LTS on your laptop.

The steps to enable GPS-based flight are:

1. Install Paparazzi. Please remark that you need to check out the master version, and that you need to allow access for USB flashing with a few commands explained on the wiki[4].
2. Launch Paparazzi by clicking on the desktop item.
3. Insert the Parrot GPS module into your ARDrone2 and power it up by connecting the battery.
4. Establish a Wifi connection between the ARDrone 2 and your laptop. Select ardrone2 v2.4.3 in Wireless Network List.
5. At Paparazzi Center select ardrone2 sdk at the "A/C" drop-down box. Make sure "ap" (Autopilot) is set as Target.
6. Press the "Upload" button.
7. In the Paparazzi control center select "ARDrone 2 Flight" as session.
8. Press "Execute"..

After the above-mentioned steps, the Parrot AR drone 2 is ready for GPS-enabled autonomous flight. It is the responsability of the operator to ensure the safety of autonomous flight in accordance with local regulations. It is important to realize that GPS-measurements can have uncertainties of several meters, so it is best to test in areas with sufficient available space.

When going outside after performing the above-mentioned steps, the autopilot will attempt to lock on to sufficient GPS-satellites. After waiting for about a minute a GPS signal will be found. The ARDrone2 location then appears in the map of the Ground Control Station (GCS). When flying for the first time in a certain location, one needs to download the Google map tiles on to the GCS computer. This can be done in advance. Figure 1 shows a screenshot of the Paparazzi GCS for autonomous flight with the AR drone 2. The flight can start by pressing "Takeoff". The remainder of the flight will happen according to the Paparazzi flight plan that the user has constructed for the mission. More information on flight plans can be found on the Paparazzi wiki[5].

[2] `http://paparazzi.enac.fr/wiki/AR_Drone_2/getting_started`
[3] `http://paparazzi.enac.fr/wiki/AR_Drone_2/getting_started_advanced`
[4] `http://paparazzi.enac.fr/wiki/AR_drone_2/Software_installation`
[5] `http://paparazzi.enac.fr/wiki/Main_Page`

Figure 1: View of the Ground Control Station of Paparazzi for the Parrot AR drone 2. The overlays include current information of the drone's state such as its height, controls such as the home-button, and the waypoints in the flight plan.

2.2 Paparazzi AR drone raw version

The requirements for using the Paparazzi AR drone raw version has in principle the same requirements as the Paparazzi AR drone SDK version. However, best results can be obtained if a different GPS receiver is used, such as a u-Blox with high update rate GPS. In addition, it is recommendable to have a a joystick.

The steps to install the raw version are then, in brief:

1. Install Paparazzi. In this case, for now one has to download a specific, TU Delft "fork" of the code.
2. Install the joystick.
3. Connect the joystick with your computer.
4. Connect the GPS with the AR.Drone 2 USB port.
5. At paparazzi center select at "A/C" the ardrone2.
6. Select "AP" as Target and press Build.
7. Turn the AR.Drone 2 on by connecting the battery.
8. Connect your computer with the AR.Drone 2 trough WiFi.
9. Select "Flight Wifi" as session and press the Execute button.

10. At paparazzi center press the "Upload" button. This step will automatically 'kill' the Parrot firmware running onboard the drone and install Paparazzi flight software and drivers. Please note that upon restart, the Parrot AR drone firmware is restored automatically. So the Parrot firmware is not lost.

11. Calibrate the compass by having the drone perform a full turn. The details of this procedure are explained on the Paparazzi wiki.

3 Inner-loop and outer-loop control of GPS-enabled AR drones

In this section, we give some insight into the control loops involved in both Paparazzi versions for the AR drone. In the Paparazzi AR drone SDK version, only the outer loop control is performed by Paparazzi. This implies that Paparazzi ensures the execution of the flight plan and makes the drone move from waypoint to waypoint, but that all lower-level control remains in the hands of the Parrot firmware. For example, the sensor fusion is performed by the Parrot firmware: the barometer, sonar, accelerometers, gyrometers, and magnetometer are integrated with optic flow estimations from the bottom camera. Combined with the barometer and sonar, the optic flow estimations provide accurate estimates of the drone's speeds. Paparazzi regulates the vertical speed, attitude angles, and yaw rate of the drone via 'AT-commands' from the Parrot SDK. Currently, Paparazzi's rotorcraft control sets the velocity of the drone in the direction of the next waypoint, which implies that the drone will roll and pitch to fly in the right direction. Parrot's firmware receives the AT-commands and controls the velocities and attitude. The use of Parrot's firmware for inner loop control provides a user the ability of GPS-enabled flight, but limits the possibilities to change and analyze this level of control. Namely, the Parrot firmware is closed source.

In the Paparazzi AR drone raw version, both the outer loop and inner loop control is performed by Paparazzi, providing the user with more possiblities especially useful for research. The version currently available on the Paparazzi repository can read out and use all sensors via Parrot's sensor board, ranging from the accelerometers to the barometer, except for the camera's. Moreover, the current rotorcraft state estimation does not use the sonar by default. Figure 2 shows a global overview of the control loops involved. The safety pilot has control over the vehicle via the RC link, which in the case of the Parrot AR drone means the joystick connected to the GCS. The flight plan controls the thrust and attitude for achieving the right speed and position. All control loops are open source and can be changed by the user[6]. The standard Paparazzi version makes extensive use of well-known PID-loops, but git-branches are available that implement more complex forms of control.

4 How to search for a target with a swarm of AR drones

Many research groups perform research on *swarming* algorithms (cf. [12, 11, 15]). The developed Paparazzi versions for the AR drone now provide the opportunity to research groups to test these algorithms on real platforms. Here we explain how to connect multiple AR drones to one GCS.

The following steps then suffice to perform a swarming experiment:

1. Add this line to the AR Drone 2 Airframe file:
   <define name="LINK_HOST" value="\\\"192.168.2.0\\\"" />

[6] `http://paparazzi.enac.fr/wiki/Control_Loops`

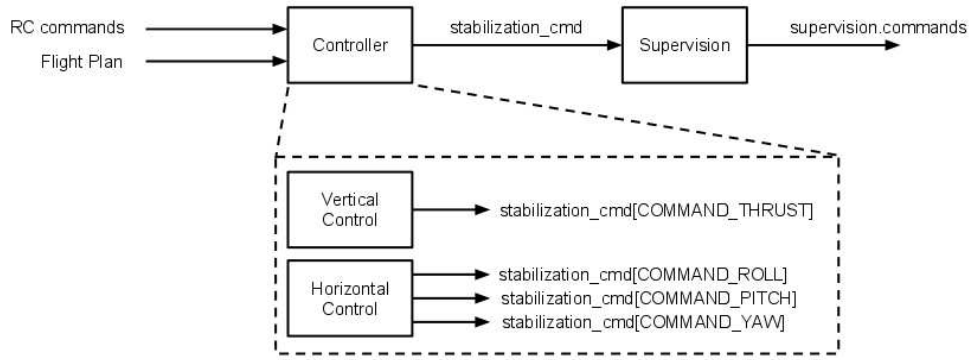

Figure 2: Global overview of the control loops involved in the Paparazzi AR drone raw version (see the text for a detailed explanation).

2. Switch on the first AR Drone
3. Connect to the Drone via Wifi SSID with name "ardrone2_XYZ" where XYZ is the current SSID of the drone.
4. Upload the Paparazzi autopilot into the AR drone as explained in previous sections.
5. Replace XYZ with the wished IP number of your ARDrone, e.g. 123 in the line below. Give the following command in the Linux terminal: *echo "killall udhcpd && ifconfig ath0 down && iwconfig ath0 mode managed essid hier ap any channel auto && ifconfig ath0 192.168.2.XYZ netmask 255.255.255.0 up && sleep 1" — telnet 192.168.1.1*
6. After this, connect the laptop Wifi to the router via SSID "thenameoftheaccespoint"
7. If all was done correctly, one of the AR drones is now connected to the swarm access point.
8. To test, type this in your terminal: *ping 192.168.2.XYZ*
9. To add more MAVs to the swarm, just repeat the steps above for every new AR drone 2 where one assigns a new IP for every new drone.

Figure 3 shows the GCS when there are five drones connected to the router. Now the flight plan can be adjusted to performing swarming experiments with the Parrot AR drone 2!

5 Conclusion

We have presented and explained two different Paparazzi versions that allow the Parrot AR drone 2 to fly autonomously outdoors on the basis of GPS. The Paparazzi AR drone SDK version performs the outer loop control for executing a flight plan, but makes use of Parrot's firmware for sensor fusion and inner-loop control. Although these functions are well-implemented in the firmware, the code is

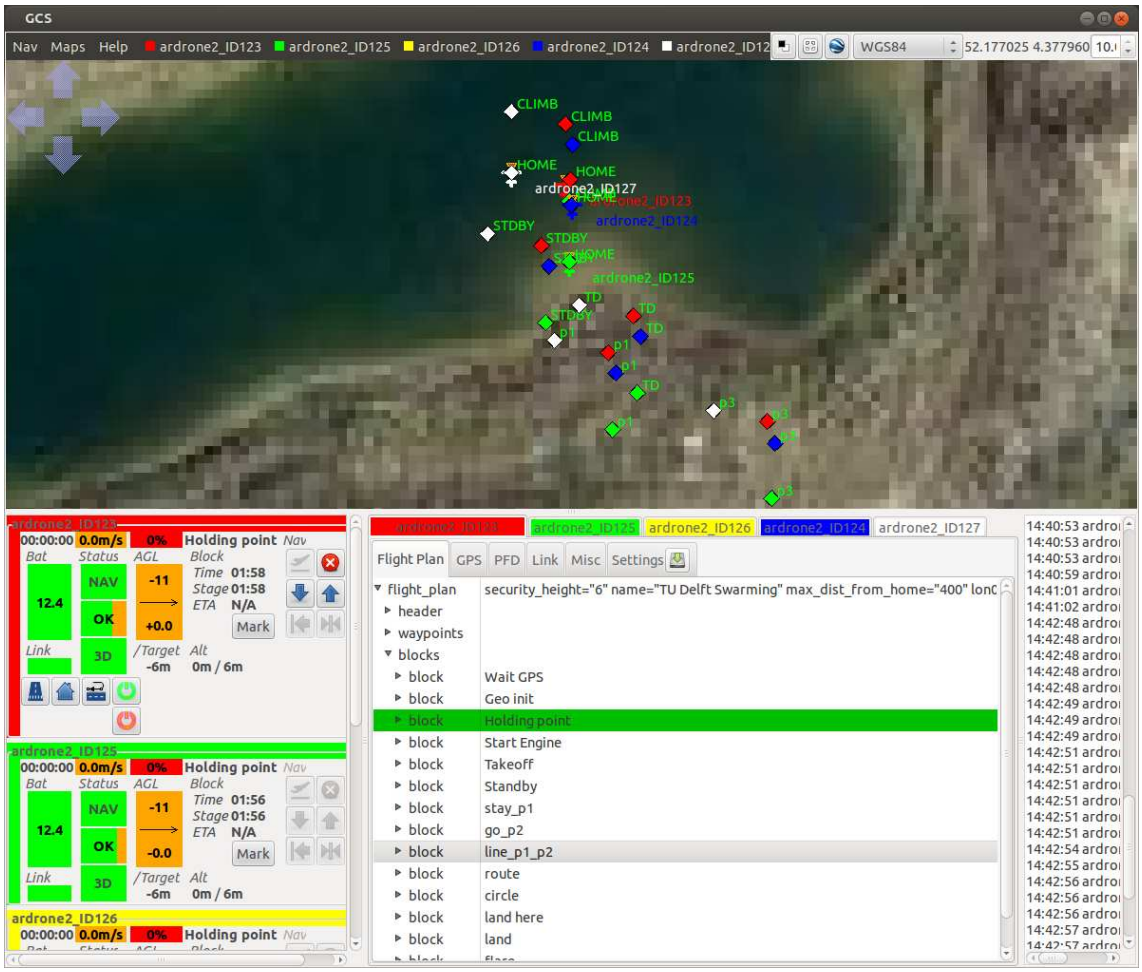

Figure 3: GCS with five AR drones.

closed source and cannot be adapted for research purposes. The Paparazzi AR drone raw version performs both the outer loop and the inner loop control. The currently available version has access to all sensors, except for the cameras. Both versions allow the connection of multiple drones to the GCS via a router. This means that research groups all over the world can now perform swarming research with a group of relatively affordable MAVs.

Finally, we want to remark that we have just gained access to the cameras onboard the AR drone. This implies that researchers can also perform onboard vision processing on the Parrot AR drone, by adding vision algorithm code to the drone's firmware. We are now working on an easy integration of this code into Paparazzi. Accessing the cameras and adding vision algorithms will be discussed in a follow-up paper.

References

[1] http://diydrones.com/.

[2] http://www.openpilot.org/.

[3] http://paparazzi.enac.fr/wiki/.

[4] http://ardrone2.parrot.com/.

[5] C. Bills, J. Chen, and A. Saxena. Autonomous MAV flight in indoor environments using single image perspective cues. In *ICRA*, pages 5776–5783, 2011.

[6] P. Brisset, A. Drouin, M. Gorraz, P.S. Huard, and J. Tyler. The paparazzi solution. In *MAV2006*, 2006.

[7] P.J. Bristeau, F. Callou, D. Vissière, and N. Petit. The navigation and control technology inside the ar. drone micro UAV. In *18th IFAC World Congress*, pages 1477–1484, 2011.

[8] H. Chao, Y. Cao, and Y. Chen. Autopilots for small unmanned aerial vehicles: a survey. *International Journal of Control, Automation and Systems*, 8(1):36–44, 2010.

[9] G.C.H.E. de Croon and S. Nolfi. Act-corner: Active corner finding for optic flow determination. In *ICRA*, 2013.

[10] N. Dijkshoorn and A. Visser. Integrating sensor and motion models to localize an autonomous ar drone. *International Journal of Micro Air Vehicles*, 3:183–200, 2011.

[11] A. Kushleyev, V. Kumar, and D. Mellinger. Towards a swarm of agile micro quadrotors. In *Robotics: Science and Systems*, 2012.

[12] L.M. Gambardella F. Mondada S. Nolfi T. Baaboura M. Birattari M. Bonani M. Brambilla A. Brutschy D. Burnier A. Campo A.L. Christensen A. Decugnire G.A. Di Caro F. Ducatelle E. Ferrante A. Frster J. Guzzi V. Longchamp S. Magnenat J. Martinez Gonzalez N. Mathews M.A. Montes de Oca R. O'Grady C. Pinciroli G. Pini P. Rtornaz J. Roberts V. Sperati T. Stirling A. Stranieri T. Stuetzle V. Trianni E. Tuci A.E. Turgut F. Vaussard M. Dorigo, D. Floreano. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics and Automation Magazine*, 2012.

[13] J. Reuder, P. Brisset, M. Jonassen, M. Muller, and S. Mayer. The small unmanned meteorological observer sumo: A new tool for atmospheric boundary layer research. *Meteorologische Zeitschrift*, 18(2):141–147, 2009.

[14] S. Ross, N. Melik-Barkhudarov, K.S. Shankar, A. Wendel, D. Dey, J.A. Bagnell, and M. Hebert. Learning monocular reactive uav control in cluttered natural environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[15] J. Zufferey D. Floreano T. Stirling, J. Roberts. Indoor navigation with a swarm of flying robots. In IEEE, editor, *Proceedings of the 2012 IEEE International Conference on Robotics and Automation*, 2012.