GoDrone Documentation

Release 0.1.0

Felix Geisendörfer

Sep 27, 2017

Contents

GoDrone is a free software[1] alternative firmware for the Parrot AR Drone 2.0. It is developed by Felix Geisendörfer and Contributors using the Go programming language. There is no affiliation with Parrot and running this firmware may void your warranty.

The current release is 0.1.0, and is aimed at adventurous software developers. If you're curious what to expect, check out this teaser video.

[1] GoDrone is licensed under the AGPLv3 license.

CHAPTER 1

User Manual

Install

Disclaimer

Please be careful. Ideally you should experiment with this firmware inside a large indoor space. If you go outside, please try to avoid proximity to streets and people. The worst case scenario is your drone causing a traffic accident, and you'll have nobody to blame but yourself.

Installing this software may also void your warranty and cause damage to your drone. But don't worry, you get back the original firmware by reconnecting the battery, and it's very easy to fix your drone if you crash it.

Download

The latest version is 0.1.0. You can download it here:

- **OSX:** Universal (32 and 64-bit)
- **Windows:** Universal (32 and 64-bit)
- **Linux:** 32-bit | 64-bit

Install

1. Download one of the archives from above.
2. Extract the archive you downloaded.
3. Place your drone on a level surface, ready for takeoff, and connect the battery.
4. Connect your computer to your drone's WiFi network.
5. Double-click the `deploy` binary inside the folder extracted from the archive.

That's it! The `deploy` binary will upload GoDrone via FTP and then telnet into your drone to start it.

Uninstall

If you want the original firmware back, simply reconnect the battery. The install is currently not permanent. This may change if the firmware becomes more mature in the future.

First flight

1. Make sure that the drone was on a level surface when you executed the `deploy` program, and that you have not rotated it since. If not, please run the `deploy` program again. Otherwise the drone will try to correct for the rotation on takeoff, which may lead to a crash right away.
2. Open http://192.168.1.1/ in your web browser, you should see the GoDrone web user interface. But don't expect too much, it's not pretty yet.
3. Connect a gamepad to your computer. Right now the only tested model is this PS3-style controller (please share your results of using other controllers).
4. Press the B0 button (X) to turn on the engines.
5. Move the right joystick up / down to control thrust.
6. Use the left joystick to control the roll and pitch angle of your drone, allowing you to fly in any desired direction.
7. Press the B0 button (X) again to turn off the engines.

Next steps

First of all, congratulations, you've earned the dubious badge of honor for having tried out version 0.1.0 of a drone software, made by amateur robotics enthusiasts, that you just found on the interwebs! You're a truly adventurous person.

Take all the time you need to recover from this experience, but if you feel hungry for more, please start to get involved with the community. Report your experience, share your problems or demand your money back.

After that, feel free to poke around the *source*, and maybe even send a patch for that thing that annoyed you the most!

Config

Various settings of the drone can be changed by editing the `godrone.conf` file that came with the download. You have to run the `deploy` command for your changes to take effect.

This page will contain more information about the various options in the future.

Frequently Asked Questions

I need help, where can I get support?

The best way to get help is by posting to the *godrone mailing list* or joining our *IRC Channel*.

Please keep in mind that this is a volunteer effort, so try to be nice.

I want to help, how can I get involved?

If you're a designer or a software developer who knows HTML, CSS, JavaScript or Go, the best way to help is by sending pull requests on *GitHub* . There are many *open issues* and *ideas*, but any proposed changes will be considered.

That being said, it's always a good idea to post to the *godrone mailing* to discuss things before getting started.

Additionally help with the documentation and answering questions on the mailing list would be really appreciated.

Where can I report bugs or propose features?

If you're not sure that what you found is a bug, and you'd like to discuss the problem first, please post to the *godrone mailing list*. Otherwise feel free to directly post on our *issue tracker*.

Features are usually best discussed on the mailing list and added to the tracker once there is consensus.

What happens if the drone goes out of WiFi range?

To prevent the drone from flying into outer space, it will shut off the motors if the network connection gets interrupted for longer than the configured `ControlTimeout`. A more graceful approach to landing will be implemented in the future.

Does GoDrone provide any features not provided by the official firmware?

Not a lot at this point, but being able to control the drone with a gamepad using your web browser is nice. It also seems that the official firmware puts some limits on speed, especially vertically, so you may find that GoDrone will help you with crashing your drone in more spectacular ways :).

What are the goals of the project?

The project was started as a personal challenge to write a firmware for the Parrot AR Drone in Go. Now that this is achieved, there are many possibilities, and it will really depend on whatever users and contributors are most interested in. Here are a few ideas:

- Improved stabilization and control algorithms (kalman filter, optical flow, etc.)
- Video drivers and support for streaming video to the HTML client
- Compatibility with the various mobile apps and NodeCopter.
- Improved HTML client, e.g. usability, design, graphs, WebGL, support for mobile devices
- More aggresive flight, faster vertical speed, and additional tricks.
- GPS support, including the ability to plan missions in the HTML client on a map
- 3G support, to allow controlling the drone over cellular networks
- Education, e.g. a specialized HTML client to demonstrate quadcopter physics
- Research, GoDrone could make it incredible easy and cheap to reproduce quadcopter research anywhere in the world.
- Support for uploading JavaScript scripts that run on the drone and allow users to create simple applications.
- Mounting the AR Drone electronics on custom frames, attaching custom sensors, and maybe even using different motors.

- Encourage vendors such as Parrot to embrace a full open source strategy.

Why is the firmware written in Go?

Originally to see if a firmware like this could be written in a high level garbage collected language.

Go seemed like a good choice because it has great support for cross-compiling, concurrency, systems programming and is just a very pleasant language to work with.

Isn't Go unsuitable for real-time applications like this?

This question is one of the reasons this project exists. Go uses a stop-the-world garbage collector that does not provide any real-time guarantees[1], so it's certainly not a perfect fit for a flying robot.

However, for all practical purposes the GC just needs to keep up with the stabilization loop which runs at 200 Hz. This means that GC pauses below 5ms have no impact on performance. Longer pauses will degrade stabilization performance, but the tolerance threshold may be up to a second depending on altitude and the situation.

Considering that stabilization cannot be guaranteed due to environmental factors to begin with, it will be interesting to see if drone vendors will make similar compromises for reducing the costs of software development , or if governments will provide detailed software architecture regulations for commercial drones.

Given that the AR Drone is a very light weight toy that has an extremely low chance of causing direct harm, the GoDrone project will continue to use the current approach for now. However, if problems are observed, or the project becomes more popular than expected, the plan is to rewrite the stabilization loop in C, run it on a separate thread with strong scheduling guarantees, and use some form of IPC to communicate with it.

Why aren't you building on existing projects?

This project is about exploring what it would be like to build a drone firmware using a high level language, and embracing web technologies for providing a portable user interface.

Other projects are certainly more mature at this point, and e.g. Paparazzi already has support for the AR Drone 2. However, they are significantly more difficult to run and work with, written in C/C++, and use rather archaic UI technology.

Where can I get the source code?

The source code is available on GitHub. https://github.com/felixge/godrone

What license is GoDrone released under?

GoDrone is licensed under the AGPLv3 license.

This basically means that any derived software products will have to be licensed under the same license, and that their source code needs to be made available.

The license was chosen to ensure that the GoDrone will always remain free software. Contributors are not asked to sign a CLA, so there will be no dual licensing model in the future.

[1] What kind of Garbage Collection does Go use?

Community & Support

This page lists the best ways for interacting with the GoDrone community.

godrone mailing list

If you'd like to ask a question, report a bug, or discuss ideas, please post to the godrone mailing list.

#godrone IRC channel

The fastest way to discuss GoDrone related topics is joining the #godrone IRC channel on freenode.net.

Bug / Feature Tracker

Bugs and features are tracked with GitHub issues: https://github.com/felixge/godrone/issues?state=open

Changelog

0.1.0

2013-12-25: This is the first release. It is aimed at adventurous software developers, and contains the following features:

- HTML client for controlling the drone via Gamepad. Uses WebSockets, jQuery, and React. Data is simply rendered into tables. There is no design yet.
- Basic navboard and motorboard drivers.
- Basic binary installer for OSX, Windows and Linux.
- Initial documentation.
- Complementary attitude filter. A kalman filter will likely provide better results in the future.
- Basic PID algorithm for control. This will require more tuning.
- Configuration via simple toml config file. Contributed by gwoo.
- Logging to stdout and log file on the drone.
- Manual altitude control. The ultra sound sensor is already working, but the initial results with it were not good enough yet. This will require some filtering.
- Automatic flat trim calibration on startup. These values will need to be saved in the future, and the yaw will have to be reset before takeoff to allow moving the drone around while on the ground.
- ControlTimeout to shut off the motors if the the network connection gets interrupted. Prevents the drone from flying into outer space.

CHAPTER 2

Contributor Manual

Install from source

OSX & Linux

Before getting started:

- Installed the latest version of Go (1.2 as of writing this)
- Configure a `$GOPATH` (e.g. via adding `export GOPATH="${HOME}/go"` to your `~/.profile`)
- Add `$GOPATH/bin` to your `$PATH` (e.g. via adding `export PATH="${GOPATH}/bin:${PATH}"` to your `~/.profile`)

Run the commands below to make sure you're in good shape:

```
$ go version
go version go1.2 darwin/amd64
$ go env GOPATH
/Users/felix/code/go
$ echo "${PATH}"  | grep -q "$(go env GOPATH)/bin" && echo "good" || echo "bad"
good
```

Next, you will need to setup Go for cross compiling to Linux/ARM. Luckily this only takes a minute. The fastest way is this:

```
$ cd $(go env GOROOT)/src
$ GOOS=linux GOARCH=arm ./make.bash --no-clean
```

If this doesn't work for you for some reason, you may try to follow Dave Cheney's guide for cross compiling Go instead.

With Go installed and ready, download GoDrone:

```
$ dst="$(go env GOPATH)/src/github.com/felixge"
$ mkdir -p "${dst}"
$ cd "${dst}"
$ git clone git@github.com:felixge/godrone.git
$ cd godrone
```

Now you should be ready to build godrone and upload it to your drone:

```
$ make
```

Windows

Building on windows is probably also doable, but has not been attempted yet. Please contribute to the docs if you get it working.

Reverse Engineering

To be written. Guide to the techniques used for reverse engineering the Parrot AR Drone 2.0.