What it is

This is an open source project that intended to help people to create autonomous drone missions that operate with a pixhawk controller. The project is written in C++ and Python in order to enable fast image processing and operating the drone in real time. The project also includes built in missions. Our goal was to fly to a specific GPS location, scan the area for a bullseye target and land on the center of the target. You can use the framework to create your own missions. The framework includes an API that helps to stream live video over wifi or/and record the video to file.

We've created this project in the Geomatric Image Processing lab at the Technion. Our goal was to create a simple framework to manage simple and complex missions represented by state machine

Requirements

1. g++ install g++ for mac or install for ubuntu : apt-get install g++
2. boost you can install it with apt-get in linux or with brew in mac.
3. boost-python - to link cpp with python check that github project for examples
4. Opencv cpp - for image processing [you don't need the opencv for Python, because the image processing is done in the C++ code] you can install opencv on mac with 'port' or with brew or install with apt-get in linux.
5. Armadillo you can install it with apt-get in linux or with brew in mac.
6. LAPACK — Linear Algebra PACKage you can install it with apt-get in linux or with brew in mac.
7. BLAS - Basic Linear Algebra Subprograms if you installed lapack you suppose to have blas check under /usr/lib
8. python install python form mac or install for ubuntu : apt-get install python
9. pip - python package manager
10. python packages : 'websocket-client', 'websocket-server', 'enum34', 'dronekit', 'dronekit-sitl', 'numpy' run `pip install -r requirements.txt` in to project_root to install all
11. nodeJs + http-server or any other way to simply create a localhost html server for displaying the UI

Build

You can build the project by running `sudo make` in the root directory
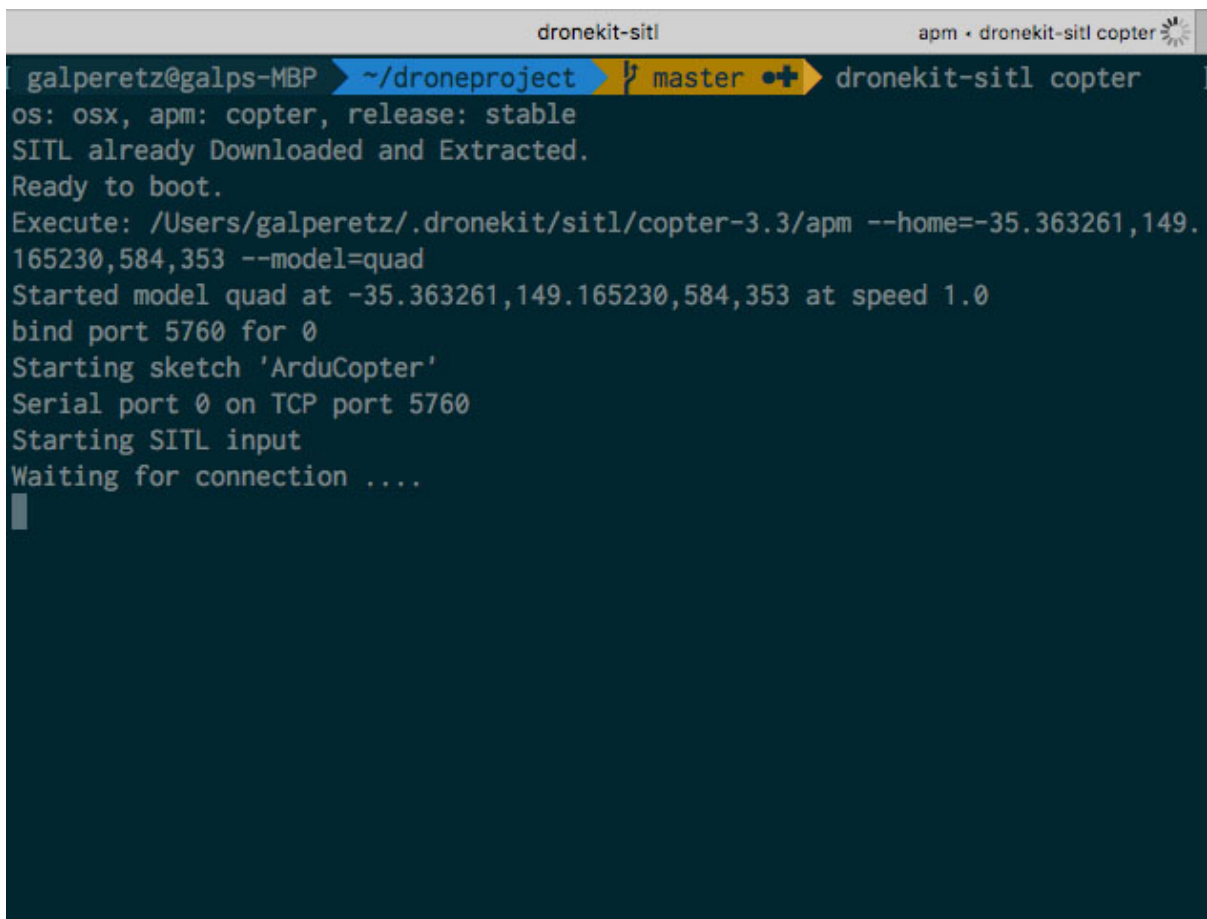
Components & Features

We divided the system to 3 main components:

1. vehicle - all the code that runs on the drone
2. ground - send commands to the drone and helps manage drone's missions.
3. common - common code that useful for both ground and vehicle

Vehicle

If the installation process went smoothly you should have dronekit-sitl already installed. dronekit-sitl is a program that simulates the pixhawk controller. you can test it by running `dronekit-sitl copter` in the terminal, and it should look like this:

```
                        dronekit-sitl              apm · dronekit-sitl copter

[ galperetz@galps-MBP   ~/droneproject   ⑂ master •✚  dronekit-sitl copter
os: osx, apm: copter, release: stable
SITL already Downloaded and Extracted.
Ready to boot.
Execute: /Users/galperetz/.dronekit/sitl/copter-3.3/apm --home=-35.363261,149.
165230,584,353 --model=quad
Started model quad at -35.363261,149.165230,584,353 at speed 1.0
bind port 5760 for 0
Starting sketch 'ArduCopter'
Serial port 0 on TCP port 5760
Starting SITL input
Waiting for connection ....
```

You can see that the simulator waiting for connection on port 5760 .
now you can run the vehicle program.
just run that in the project root directory:
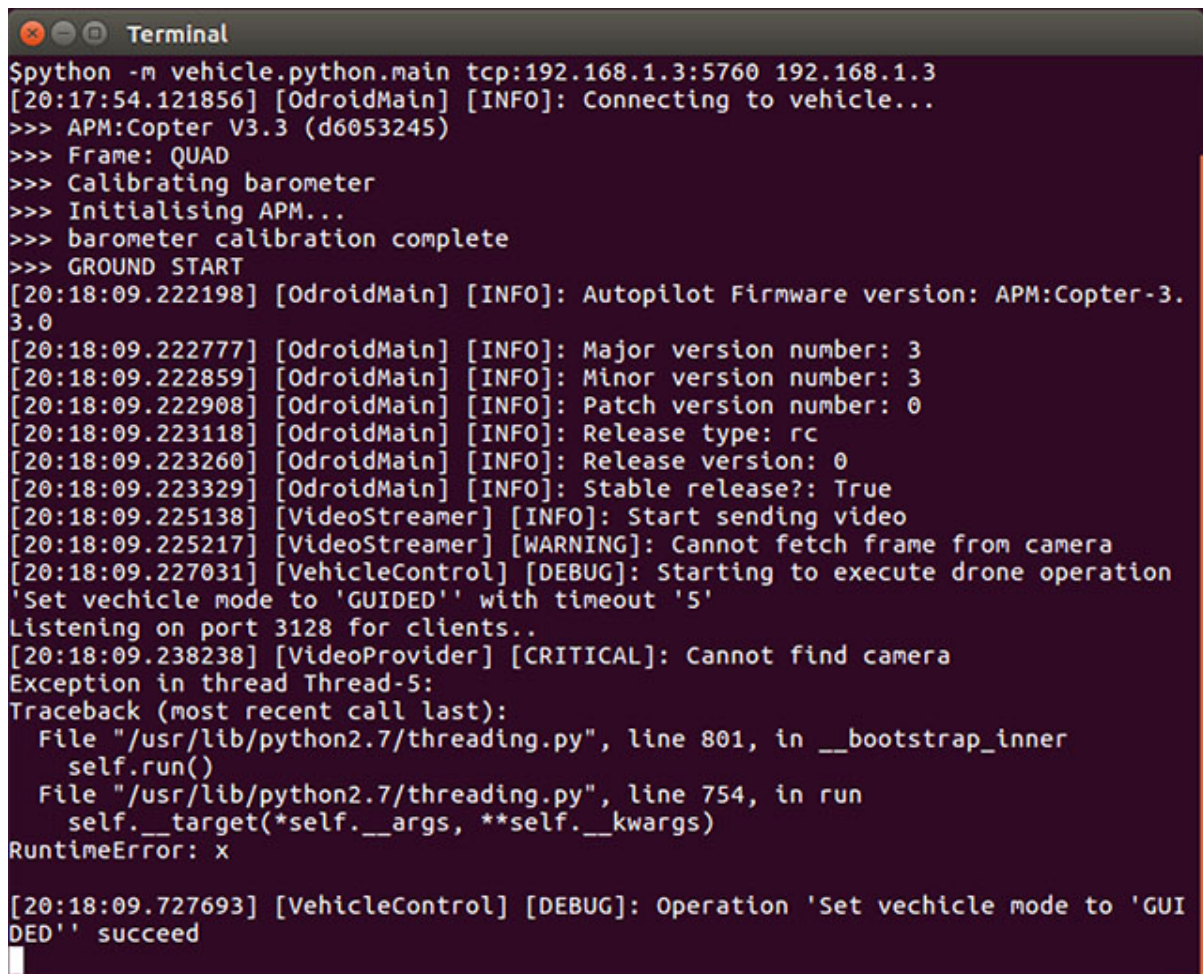
```
python -m vehicle.python.main tcp:<ip where you run dronekit-sitl>:5760 <ip where
you want to see the live video stream>
```

(we will explain later how to see the video stream)

For example if you run the code in the same pc that you run dronekit-sitl you can just run:

```
python -m vehicle.python.main tcp:127.0.0.1:5760 127.0.0.1
```

You should see something like that if you don't have camera connected:


```
$python -m vehicle.python.main tcp:192.168.1.3:5760 192.168.1.3
[20:17:54.121856] [OdroidMain] [INFO]: Connecting to vehicle...
>>> APM:Copter V3.3 (d6053245)
>>> Frame: QUAD
>>> Calibrating barometer
>>> Initialising APM...
>>> barometer calibration complete
>>> GROUND START
[20:18:09.222198] [OdroidMain] [INFO]: Autopilot Firmware version: APM:Copter-3.
3.0
[20:18:09.222777] [OdroidMain] [INFO]: Major version number: 3
[20:18:09.222859] [OdroidMain] [INFO]: Minor version number: 3
[20:18:09.222908] [OdroidMain] [INFO]: Patch version number: 0
[20:18:09.223118] [OdroidMain] [INFO]: Release type: rc
[20:18:09.223260] [OdroidMain] [INFO]: Release version: 0
[20:18:09.223329] [OdroidMain] [INFO]: Stable release?: True
[20:18:09.225138] [VideoStreamer] [INFO]: Start sending video
[20:18:09.225217] [VideoStreamer] [WARNING]: Cannot fetch frame from camera
[20:18:09.227031] [VehicleControl] [DEBUG]: Starting to execute drone operation
'Set vechicle mode to 'GUIDED'' with timeout '5'
Listening on port 3128 for clients..
[20:18:09.238238] [VideoProvider] [CRITICAL]: Cannot find camera
Exception in thread Thread-5:
Traceback (most recent call last):
  File "/usr/lib/python2.7/threading.py", line 801, in __bootstrap_inner
    self.run()
  File "/usr/lib/python2.7/threading.py", line 754, in run
    self.__target(*self.__args, **self.__kwargs)
RuntimeError: x

[20:18:09.727693] [VehicleControl] [DEBUG]: Operation 'Set vechicle mode to 'GUI
DED'' succeed
```

Or something like that if you do :

```
Terminal

$python -m vehicle.python.main tcp:192.168.1.3:5760 192.168.1.3
[20:21:37.423344] [OdroidMain] [INFO]: Connecting to vehicle...
>>> APM:Copter V3.3 (d6053245)
>>> Frame: QUAD
[20:21:48.390875] [OdroidMain] [INFO]: Autopilot Firmware version: APM:Copter-3.
3.0
[20:21:48.391005] [OdroidMain] [INFO]: Major version number: 3
[20:21:48.391036] [OdroidMain] [INFO]: Minor version number: 3
[20:21:48.391070] [OdroidMain] [INFO]: Patch version number: 0
[20:21:48.391112] [OdroidMain] [INFO]: Release type: rc
[20:21:48.391146] [OdroidMain] [INFO]: Release version: 0
[20:21:48.391184] [OdroidMain] [INFO]: Stable release?: True
[20:21:48.395329] [VideoStreamer] [INFO]: Start sending video
[20:21:48.395913] [VideoStreamer] [WARNING]: Cannot fetch frame from camera
Listening on port 3128 for clients..
 [20:21:48.396758] [VehicleControl] [DEBUG]: Starting to execute drone operation
 'Set vechicle mode to 'GUIDED'' with timeout '5'
[20:21:48.396855] [VehicleControl] [DEBUG]: Operation 'Set vechicle mode to 'GUI
DED'' succeed
[20:21:48.462962] [VideoProvider] [INFO]: Camera connected starting to read fram
es
[20:21:48.787829] [VideoStreamer] [WARNING]: Some packages lost in the way when
sending the frame
```

Working with a real pixhawk instead of a simulator:

If you working with Pixhawk and you have Odroid computer (or any other linux machine) that connects to it You can run that line to connect the computer to the pixhawk and start send it commands.

```
python -m vehicle.python.main /dev/ttyUSB0,115200 <ground_ip>
```
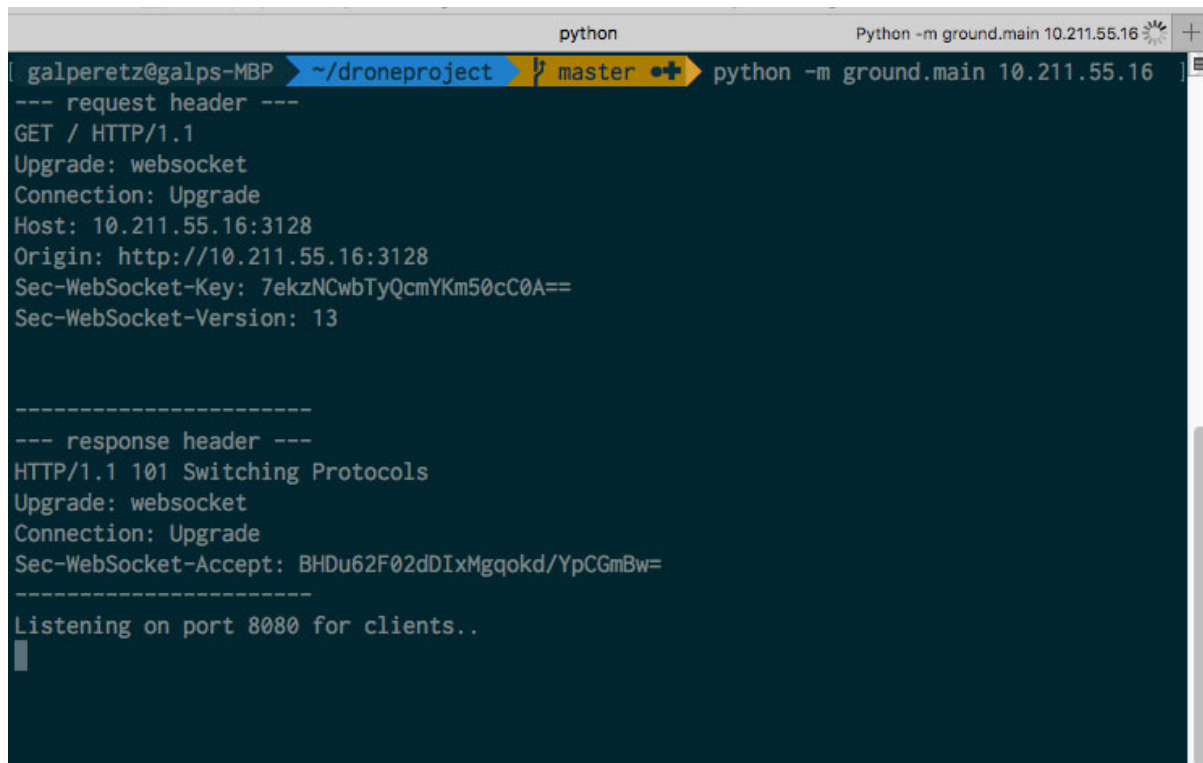
Ground

Communication

Now we need to create a communication between the vehicle(the drone) and the ground(the browser) Run this code to establish the communication channel

```
python -m ground.main  <drone_ip>
```

For example if you run all in the same pc just run

```
python -m ground.main  127.0.0.1
```

It should look something like that :

GUI - graphic user interface

We use html and javascript to display the state of the mission.

You can use http-server to display the GUI just follow those steps:

1. Install nodeJs
2. Installing nodeJs should install npmJs 'node package manager' use npm to install http-server by running `npm install -g http-server` in the terminal
3. navigate to droneproject/ground/ui and run http-server -p5555
4. naviagte to http://localhost:5555/ in chrome and open the console
5. load the mission by running loadMission("FindAndLandMission")

It should Look like this when you run http-server from the terminal:

If you've already successfully established communication with the drone you are ready to connect, else just jump back to the Communication section.

Connect the UI to the browser by following these steps:

1. Navigate to the browser open the console and load the mission by running loadMission("FindAndLandMission").
2. Connect to the drone by running client.connect().
3. Start the mission by running client.send('{"type" : "CMDTypes.FIND_AND_LAND_MISSION" , "body": {"alt":2, "distance": 100}}').

And after the mission start

And then when you run the mission from the browser :

Note : the 'Find and land' mission fits to our specific drone.
the camera sits on the drone at specific rotation and we analyzing the image based on that, and control the drone to move in order to minimize the distance to the center of the target.

How To Get Started

To better understand the architecture of that project start by reading the project's presentation that explains about all of the components.

Then you can take a look at the missions that we've already created under project_root/vehicle/cpp/src/mission. (for example understand how 'Up and down' mission works). Finally you can try to create your own simple mission.

Creating missions for the drone

The system provides an easy and convenient way to create missions for your drone.
Every mission can represented by a state machine every state in the state machine represent a state in the drone's mission.
you can add mission by adding it to the mission folder under vehicle/cpp/src/mission/. you can take a look on the UP_AND_DOWN_MISSION vehicle/cpp/src/mission/up_and_down_mission.hpp ,that is a simple mission with 2 states takeoff and land.
just follow that mission and try to create your own simple mission.

Project presentation

Under `project_root/presentation`
you can find our final project presentation, one version is a pdf and one is power point.

Video

Channels

We divid the video stream to channels.
There are 2 channels in the system for now but you can add more.

The DEFAULT channel:
when you want to get the original video from the camera you should read the frame from the DEFAULT channel .

The DEBUG channel:
you can apply collection of modifiers to that channel . before the video provider provides a frame from that channel (or from any custom channel) it will change the frame according to the that collection of modifiers that applied for that channel.

Create custom channel:
You can create any custom channel very easily and the apply any collection of modifiers that you want like the DEBUG channel . take a look in the vide_provider.hpp for more info.

Video stream

when you runs the drone code and get logs that every thing good with the camera and the video provider started to get frame for it you can start the video_server that sits under the ground folder run:

```
cd ground/video_server
./server 10000
```

And you should be able to see a live stream video from your drone The video streamer sends the video stream from DEBUG channel so you would be able to apply any masks filters and transformations on the video stream before you get it on the ground see the scan state in the find_and_land_mission for example.

Recording the video

The video recorder can record the video and save it under flights_video folder .
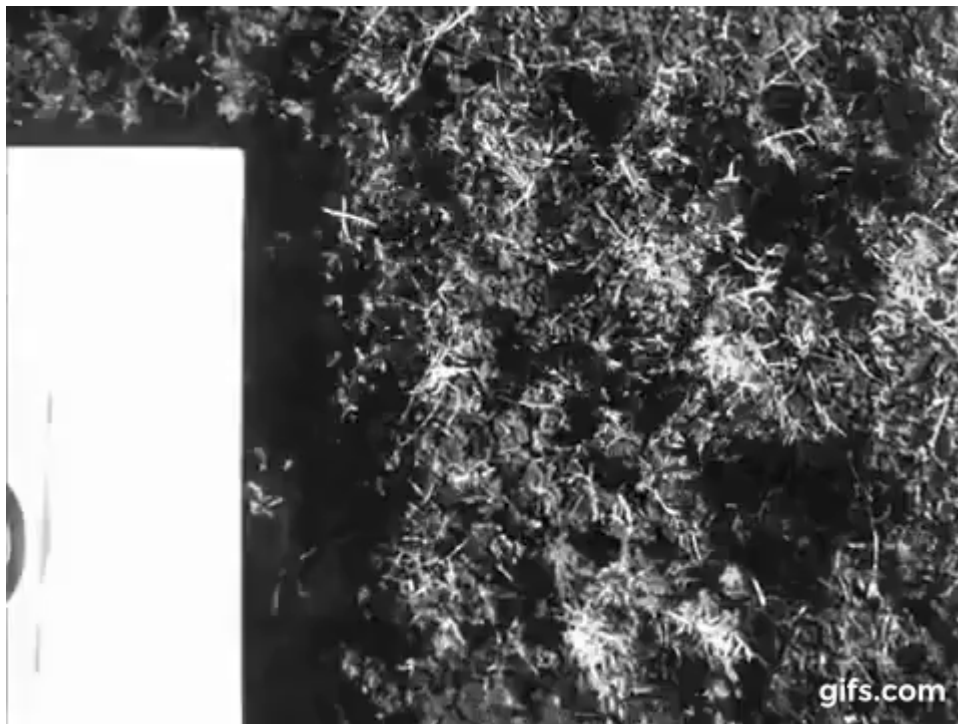In order to start recording you need to put the --record flag when you run the drone code .

```
python -m vehicle.python.main tcp:127.0.0.1:5760 127.0.0.1 --record
```

The video recorder reads the video from the DEAFULT channel so he gets the original video without any modifications. if you want to modify the video before saving it to disk create RECORD channel and set collection of modifiers to that channel and change the video recorder to read from that channel instead.

Videos from our mission

Find and land mission's videos from the drone camera (without the go to GPS part) After takeoff, the drone starts to scan the area and then decides where to go base on the image algorithm that tells him where is the center of the target.

Click on the GIF to see the full video:

Our Drone

Useful links

- Dronekit
- MavProxy
- Mavlous
- [APM planner](APM-planner)
- [Mission planner](Mission-planner)
- [Calibrate the camera](Calibrate-the-camera)
- [Calibrate the compose and the GPS](Calibrate-the-compose-and-the-GPS)