This project has been tested with the current versions:

Ubuntu: 18.04 Bionic

Ros: Melodic

Github link for all the documents:

https://github.com/hugoSaxion/Visual-servoing

1. Vision system setup

1. Create a workspace for catkin

```
$ mkdir -p ~/visual_ws/src
$ cd ~/visual_ws/
$ catkin_make
```

http://wiki.ros.org/catkin/Tutorials/create_a_workspace

2. Next, MAVROS needs to be installed. To have MAVROS installed, you will initially need to install wstool as well as catkin-tools which can be found with the following links:

(wstool) http://wiki.ros.org/wstool

(catkin tools) https://catkin-tools.readthedocs.io/en/latest/

Once this is done instead of building the workspace with "catkin_make", the workspace is now built with "catkin build".

3. ONE of the two following links can now be used as sources to get the proper version of MAVROS installed. Keep in mind that instead of the "`~/catkin_ws`", "`~/visual_ws/`" should be used in these manuals.

(from the header: "installation" onwards)

1. https://github.com/mavlink/mavros/tree/master/mavros#installation

(For this, mind that the ROS version used here should be compatible for other ROS versions)

2. https://docs.px4.io/master/en/ros/mavros_installation.html

4. Create a package

```
$ cd ~/visual_ws/src
$ catkin_create_pkg code_qrcode std_msgs rospy roscpp
```

http://wiki.ros.org/ROS/Tutorials/CreatingPackage

5. Put the three ROS nodes in a folder called *script* in *~/visual_ws/src/code_qrcode*

6. Add code to the end of *CMakeLists.txt* in ~/visuals/src/code_qrcode. The text:

```
catkin_install_python(PROGRAMS script/feature_scanning4.py
script/det_movement3.py script/distance_scanning2.py
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)
```

7. Make catkin workspace.
```
$ cd ..
$ catkin build
```

8. Adjust *feature_scanning4.py* to the current situation.
   - Line 37: adjust to frame rate of camera
   - Line 45: adjust to camera input. src=0: intern webcam; src=1 or 2: extern webcam
   - Line 56: Set field of view (FoV) to value of camera

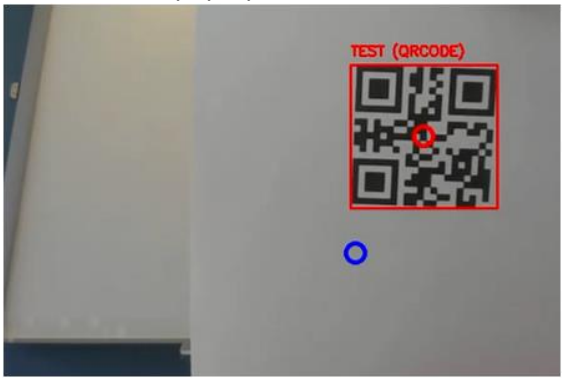
9. Adjust *distance scanning2.py* to the current situation.
   - Line 39: adjust value of the real size of the printed qr-code in cm

10. Install pyzbar for ROS in new terminal.
```
$ sudo apt-get install libzbar0
```

11. Close all terminals

12. Start ROS in a new terminal
```
$ roscore
```

13. Open a new terminal and start *feature_scanning4.py.*
```
$ cd visual_ws/
$ source ./devel/setup.bash
$ catkin_build
$ rosrun code_qrcode feature_scanning4.py
```
Now you should see a pop-up screen with a video stream and a blue circle



14. Open a new terminal and start *distance_scanning2.py*
```
$ cd visual_ws/
$ source ./devel/setup.bash
$ catkin_build
$ rosrun code_qrcode distance_scanning2.py
```

15. Open a new terminal and start *det_movement3.py*
```
$ cd visual_ws/
$ source ./devel/setup.bash
$ catkin_build
$ rosrun code_qrcode det_movememt3.py
```
Now all the ROS nodes that are needed are running.

16. Hold the qr-code in front of the camera.

17. Listen to the channel by opening a new terminal

```
$ rostopic echo /ch_mov
```

2. Vision controller node

If these steps have finished and work, the vision part of the program has been succesfully executed and the controller node can be added.

18. Start anew with a fresh terminal. At this point you should be able to make a new package for the vision controller through

```
$ cd ~/visual_ws/src
$ catkin_create_pkg offboard_control std_msgs rospy roscpp
```

And put the ROS node "vision_controller_node.cpp" in a new folder you create called *src* in "*~/visual_ws/src/*offboard_control"

19. From this point on you should add the following at the end of the *CMakeLists.txt* in "*~/visual_ws/src/*offboard_control".

```
add_executable(offboard_control src/offboard_control.cpp)
target_link_libraries(offboard_control ${catkin_LIBRARIES})
```

20. Finally you should go back to the root of your workspace and build it again with catkin

```
$ cd ~/visual_ws
$ catkin build
```

21. Once this is done you can check if it works by trying to connect to the drone with usb connection and run:

```
$ roslaunch mavros px4.launch fcu_url:="/dev/ttyACM0:57600"
```

Note: your drone might have a different fcu_url and it should be adjusted accordingly

If successful, one of the lines that should appear in the terminal contains:

CON: Got HEARTBEAT

If this is seen, a proper connection with the drone has been established.

The next step is to then open a second terminal window and run:

```
$ cd ~/visual_ws
$ rosrun offboard_control vision_controller_node
```

By itself this node won't show anything, but you can then open another terminal window and run:

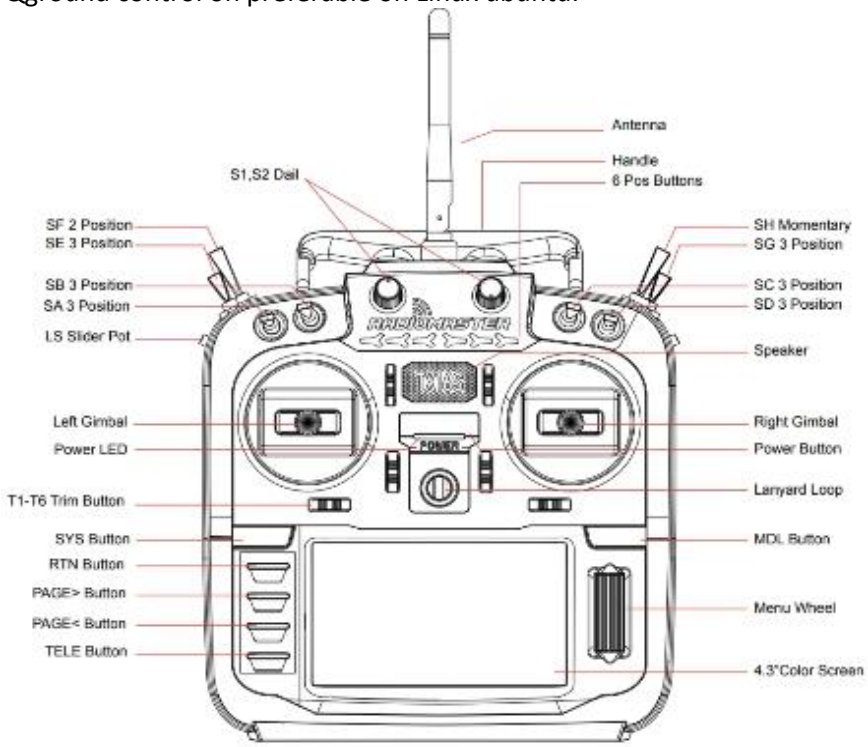
```
$ cd ~/visual_ws
$ rostopic echo /mavros/vision_pose/pose
```

Which is a channel the script should be publishing data towards. If you edit a starting value for this data in the script, it should be reflected in the response from the echo.

3. Controller settings and px4 parameters setup

Requirements

- install Qground control on preferable on Linux ubuntu.



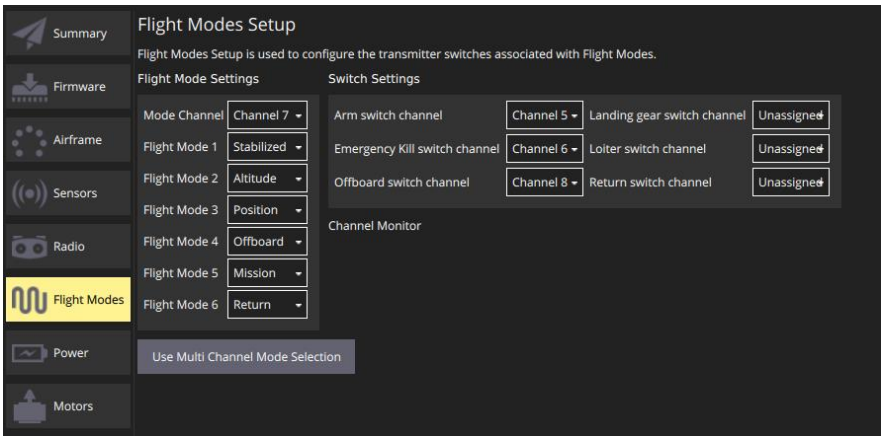
To connect the RC to the drone. See the Installation and setup of a multirotor with Pixhawk running PX4 firmware manual. If this RC is already connected and installed for this drone. the refed manual does not apply for u.

The functions of all the buttons of the controller for this project:

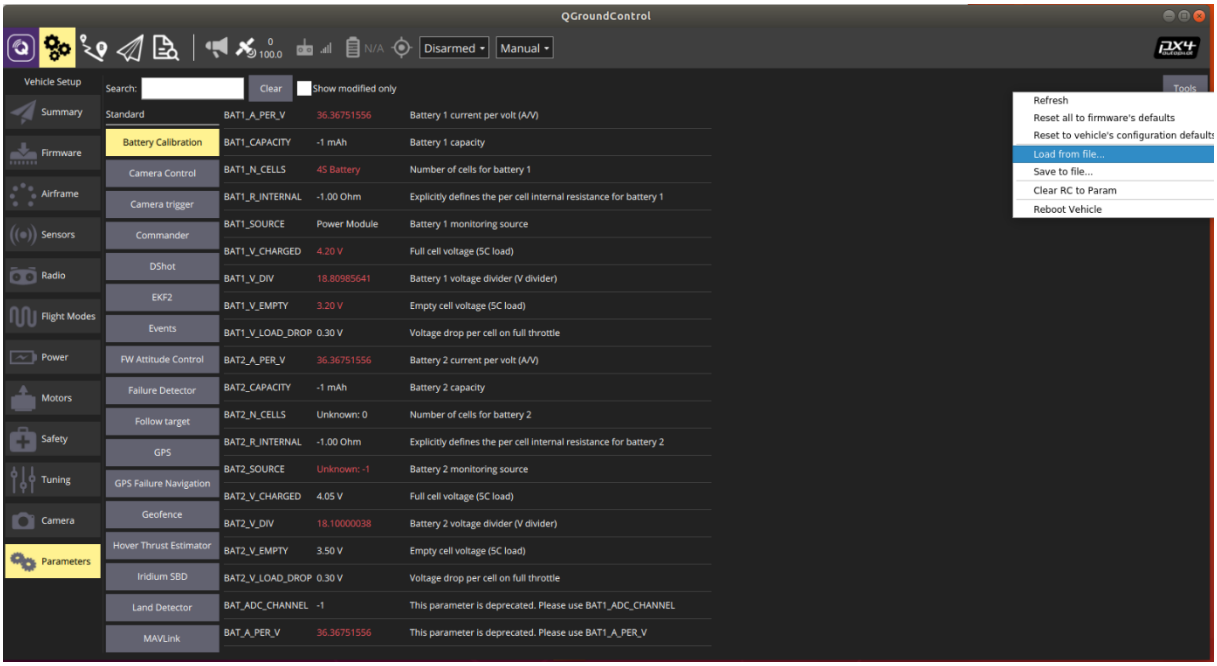
Button	Function
SF 2 position	Arm the drone
SH momentary	Kill switch (caution!! Before you let go of this button, first switch off the arm and offboard button. Before release the kill switch)
SG 3 position	Arm offboard switch
Button 4	Arm offboard switch

The standard procedure to switch a mode is to use the button 4. But because this is a test setup the SG 3 position button does the same thing and has a benefit that it is closer to the kill switch.

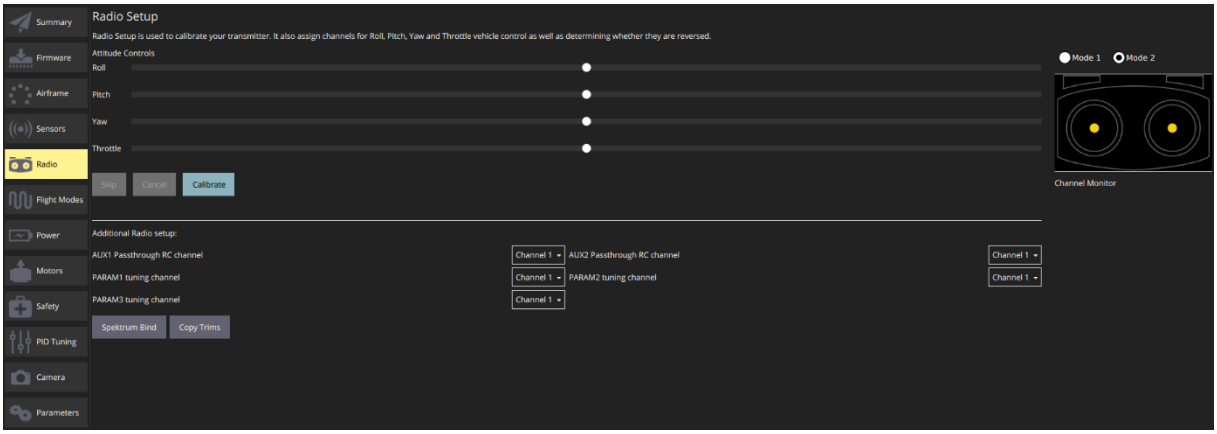
To chance the button functions for the RC. Go to QGroundcontrol and connect the drone with a normal USB cable. Not the long one! After loading go to settings and click **flight modes**. Chance all the settings until that it has the same as the picture below.



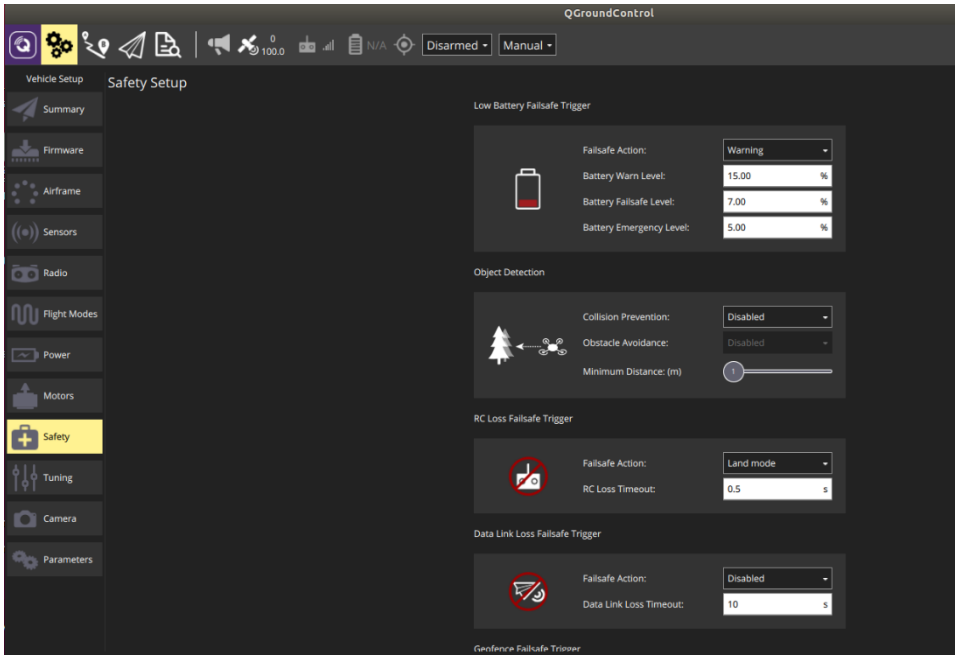
To chance the parameters, we used for or project. Go to **parameters** and click on **tools** right above the corner. Click on **load from file**. And select the **parameters Visual servoing UAV.params**. See picture below:

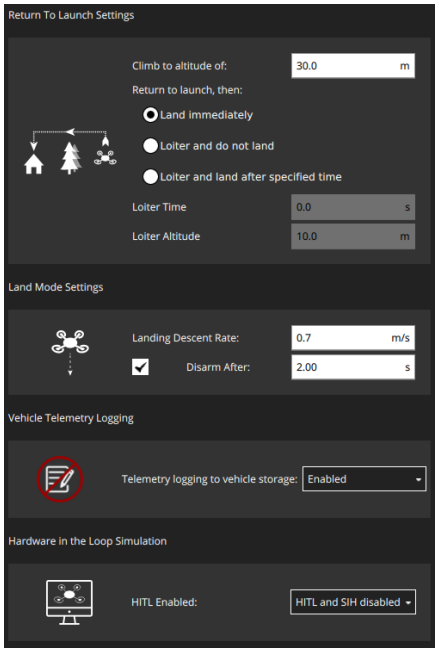


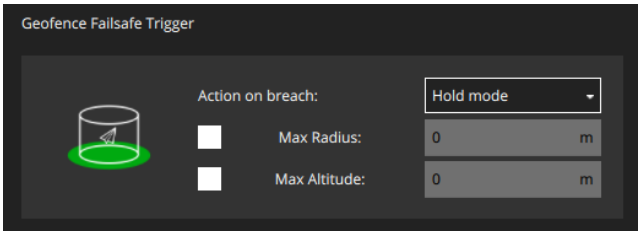
Now go to **radio** and check if the values are the same of the picture below:



Now go to **safety** and check if the values are the same of the picture below:







Once this manual has been successfully ran through. The set-up has been complete, and the startup manual can be follow in order to fly the drone.