Sam Church
RIIS, LLC
April 25, 2022

Yaw Offsets in Orbit Flight Mode - QGroundControl (Herelink)

A how-to guide with code samples on configuring and building your own custom QGroundControl application. We'll be modifying the existing yaw control to allow a pilot to input a yaw offset (in degrees) while in orbit flight mode.

Note if you are looking for a version on Windows, see the separate documentation labeled "Yaw Offsets in Orbit Flight Mode - QGroundControl (Windows)"

Environment Set-Up

If you already have the required software installed, double-check that you have the correct versions. Then you may continue to the section titled 'Configuring the Project.'

Required Software

- Microsoft Visual Studio 2019
- Qt/Qt Creator
- Android Studio
- PX4 Toolchain

Installing Visual Studio 2019

1. Follow this link to download the last version of Visual Studio 2019:
    a. https://my.visualstudio.com/Downloads?q=visual%20studio%202019&wt.mc_id=o~msft~vscom~older-downloads
2. Once downloaded, run the installer and check the box labeled, "Desktop development with C++"
3. Finish installing by checking the Install box

Installing Qt Creator

1. Follow this link, scroll down to the section titled "Looking for Qt binaries?"
    a. https://www.qt.io/download-open-source
2. Click the button labeled "Download the Qt Online Installer"
3. A page will show up with a green button labeled "Download". Click it and the QT Online Installer will download
4. After downloading, launch the installer
5. Log in / Create an account for Qt
6. Check the box claiming you are an individual

7. Hit next until you reach installation folder
8. Check the box labeled “Custom Installation” and hit next
9. Select the dropdown for Qt and check the box labeled “5.12.6”
10. Finish installation by hitting next until it is complete

Installing Android Studio

1. Navigate to this link:
    a. https://developer.android.com/studio
2. Click the button “Download Android Studio” to download Android Studio
3. Once installed, run the installer and follow the steps unit it is completed
4. Launch Android Studio
5. Continue until it shows you an option to create a new project
6. Select “More Actions”
7. Select the SDK Manager
8. Under SDK Platforms, check the boxes of the versions of Android you would like to test
9. Under SDK Tools, select these options:
    a. Android SDK Command-line Tools (Latest)
    b. CMake
10. Select “Apply” and install those packages

Installing Android NDK and SDK

1. Navigate to <Qt installation folder>/Tools/QtCreator/share/qtcreator/android
2. Open the sdk_definitions json file
3. Change lines 18-29 to this:

```
"specific_qt_versions": [
    {
        "versions": ["default"],
        "sdk_essential_packages": ["build-tools;33.0.0", "ndk;21.3.6528147"],
        "ndk_path": "ndk/21.3.6528147"
    }
]
```

4. Save the file and exit
5. Launch Qt Creator
6. Select “Tools” from the top toolbar
7. Select “Options…”
8. On the left sidebar, scroll down and click “Devices”
9. On the top tab, select “Android”
10. A prompt will appear stating there are missing necessary packages (This is the NDK, Google USB Drive, Android SDK Platform 20, and Android SDK Build-Tools 30.02). Select Yes to install the missing packages
    a. Scrolling down will show the installation progress

11. Qt Creator will prompt for you to review the licenses, select 'Yes”
12. Select “Yes” for all the licenses
    a. After selecting, the process of installing will take a while
13. Verify that the JDK Location under Java Settings match the Android Studio JRE folder
14. Verify that the NDK under Android Settings → Android NDK List being used is version 21.3.6528147
15. Close this window and open the QGC project, if you haven’t already
16.
17. On the left sidebar, select “Projects”
    a. Under “Build & Run”, an option for Android should appear. Select “Android Qt 5.15.2 Clang Multi-Abi”
18. This will add a build configuration
19. Select the build option under Android Qt 5.12.6 Clang Multi-Abi
20. Under “Build Steps”, select “Details” in the qmake step
21. In the ABI’s setting, enable arm64-v8a
22. The setup is complete for building on Android

Installing PX4 Toolchain

1. Follow this guide for the latest installation tutorial:
    a. https://docs.px4.io/v1.12/en/dev_setup/dev_env_windows_cygwin.html
    b. Make sure to check the box that clones the PX4 repository at the end of the installer

Configuring the Project

1. Clone the herelink-v4.0.8 branch of QGroundControl repository found here:
    a. https://github.com/CubePilot/qgroundcontrol-herelink
2. Open Qt Creator
3. Select “File” and then “Open File or Project...”
4. Navigate to the repository
5. Select the file “qgroundcontrol.pro”
6. Select the Android option
7. Select “Configure Project”
8. In the terminal window navigate to the [QGroundControl Repository]/custom
9. Update sub dependencies with this command:
    a. git submodule update --recursive
10. Restart Qt Creator
11. Click the button above the run (green arrow) to change where to build and make sure that the Android option is selected
12. The project will now be able to build without any modifications

Sam Church
RIIS, LLC
April 25, 2022

Build & Deployment

1. In Qt Creator, select the hammer button in the bottom right corner to build the project
2. Connect to Herelink device to the computer on whi h you are building QGroundControl
3. Where the green arrow is to run, change the build device to the Herelink device connected
4. Select the green button to run on Herelink
    a. It will create a folder outside of the project folder (default is [user]/Documents) containing the flight logs, missions, and other miscellaneous items
    b. It will also create a build folder next to the project folder containing a staging folder with an executable. This executable will run the whole application on a Windows device. The staging folder can be dragged and dropped anywhere and the executable will run

Development

All the source code used can be found at https://github.com/riis/herelink/

Creating Custom Input Sliders

There are 2 input sliders that we will be creating; A horizontal slider to control the yaw offset (in degrees), and the vertical slider which controls altitude.

1. Create 2 new files under src/QmlControls:
    a. QGCVerticalSlider.qml
    b. QGCHorizontalSlider.qml
    c. Copy the sliders from the riis/herelink repository to QGCVerticalSlider.qml and QGCHorizontalSlider.qml, respectively
2. In custom/qgroundcontrol.qrc, add these lines after the line where it defines QGCSlider.qml like so:

```
<file
alias="QGroundControl/Controls/QGCSlider.qml">../src/QmlControls/QGCSlider.qml</fil
e>
<file
alias="QGroundControl/Controls/QGCHorizontalSlider.qml">../src/QmlControls/QGCHoriz
ontalSlider.qml</file>
<file
alias="QGroundControl/Controls/QGCVerticalSlider.qml">../src/QmlControls/QGCVertica
lSlider.qml</file>
```

3. In qgroundcontrol.qrc follow the same as step 3.
4. In src/QmlControls/QGroundControl/Controls/qmldir, add these two lines below QGCSlider like so:

```
QGCSlider                       1.0 QGCSlider.qml
QGCHorizontalSlider             1.0 QGCHorizontalSlider.qml
QGCVerticalSlider               1.0 QGCVerticalSlider.qml
```

5. Copy the VirtualJoystick from the riis/herelink repository to src/FlightDisplay/VirtualJoystick.qml
6. Navigate to custom/updateqrc.py and execute the file in a command prompt

Modifications the User Interface

Making small changes to the existing user interface for quality of life improvements.

1. Copy the FlyViewMap from the riis/herelink repository to src/FlightDisplay/FlyViewMap.qml
2. Copy the QGCMapCircleVisuals from the riis/herelink repository to src/MissionManager/QGCMapCircleVisuals.qml
3. Copy the MissionItemIndexLabel from the riis/herelink repository to src/QmlControls/MissionItemIndexLabel.qml

Modifying Android Settings

1. Copy these files from the riis/herelink repository, respectively:
    a. AndroidManifest.xml
    b. QGCCommon.pri
    c. QGCExternalLibs.pri
    d. android/build.gradle

Modifying the Simulator

Note that currently this is only supported on MacOS and Linux

1. Create a new file using this command:

```
$ nano ~/.zshrc
```

2. Copy these lines into the file:

```
#start of file
---------------------------------------------------------------------
ulimit -S -n 2048
# Point pip3 to MacOS system python 3 pip
alias pip3=/usr/bin/pip3
export PATH="/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin"
```

```
export PATH="$PATH:"/Users/<yourusername>/Library/Python/3.8/bin""
export
PATH="$PATH:"/Users/<yourusername>/Library/Python/3.8/lib/python/site-packages""
export PATH="$PATH:"/usr/local/Cellar/bullet/3.21/lib""
export PATH="$PATH:"/usr/local/Cellar""
export PKG_CONFIG_PATH="/usr/local/lib/pkgconfig"
export PATH="$PATH:"/Users/<yourusername>/Library/Python/3.9/bin""
export PATH=$PATH:/Library/Frameworks/GStreamer.framework/Commands
#end of file
----------------------------------------------------------------------
```

3. Inside of where the PX4 tool chain was installed, navigate to and open /Documents/riis/code/PX4-autopilot/build/px4_sitl_default/etc/init.d-posix/px4-rc.mavlink
4. Comment out the API/Offboard link like so and add the second block to allow the PX4 simulator to be used over local network

```
# API/Offboard link
#mavlink start -x -u $udp_offboard_port_local -r 4000000 -f -m onboard -o
$udp_offboard_port_remote

#use this one to broadcast over the network - useful for
#debugging to desktop QGC or if serial connection not working
mavlink start -x -u $udp_offboard_port_local -r 4000000 -f -p
```

Using the Simulator

1. Ensure that both the device running the simulator and the herelink device is on the same network
2. Navigate to where the PX4-Autopilot was cloned to from the installer and run this command:

```
$ make px4_sitl jmavsim
```

3. Launch QGroundControl on Herelink and the devices will automatically connect with one-another