# Control allocation in PX4

Repository for the code specific to Firefly, a 15kg coaxial octorotor.

The vehicle is controlled using px4 software and Pixhawk 4. The purpose of this to: - Read live rotor data (throttle, RPM, current, voltage, etc) from ESC and other sensors - Send commands (e.g. delta_throttle) to the Control Allocation unit inside the px4 based on the collected rotor data - Write input data (from the rotor) and output data (commands sent to px4) to a log file

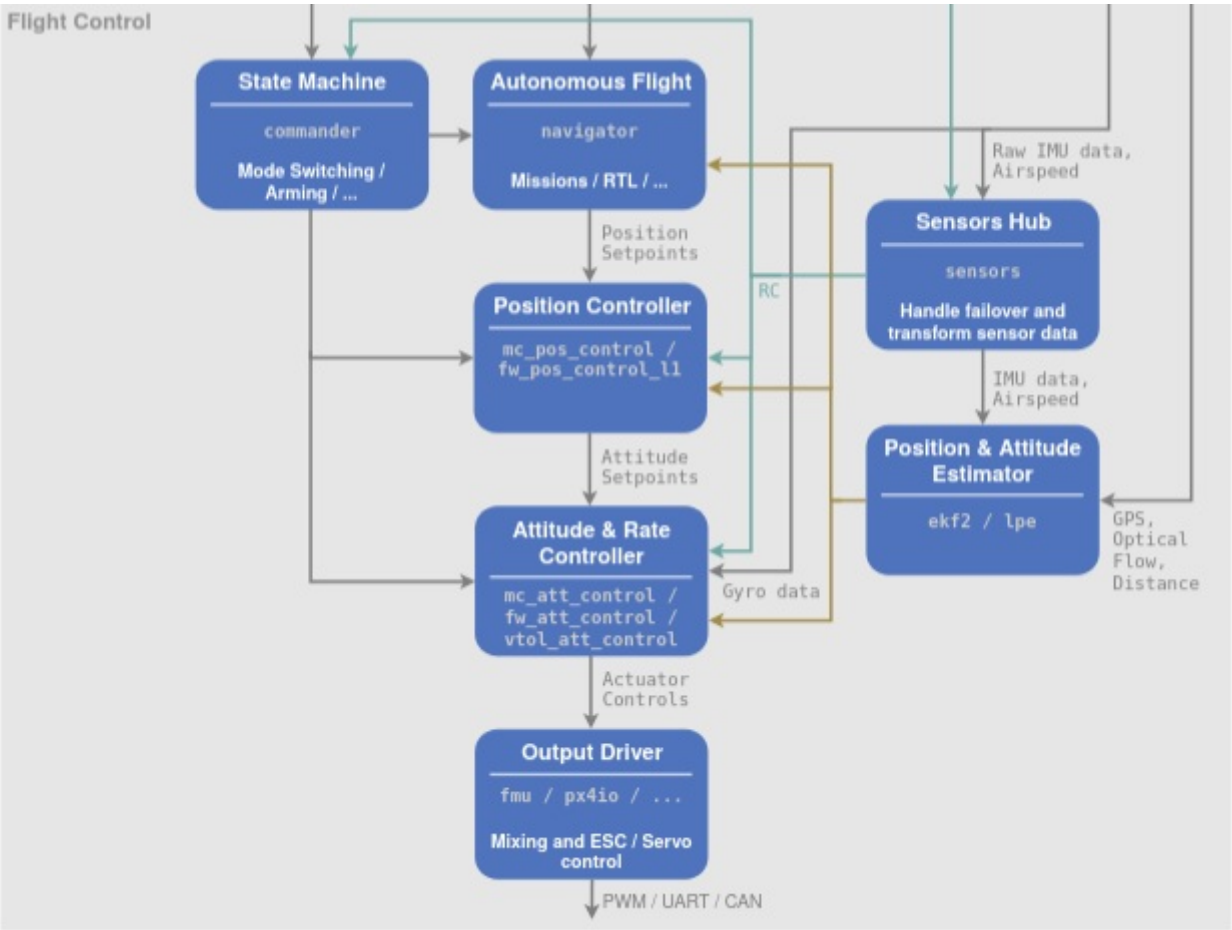
This effectively closes the px4 control loop with rotor data (throttle, RPM, current, voltage, etc). The idea is to incorporate secondary goals (e.g power, vibration, noise) into the control allocation unit of ***overactuated*** vehicles. Of course none of these is possible for systems that have only one actuator per degree of freedom.

## What do we need?

- Pixhaw 4 board
- Px4 software at version 1.12.0
- Knowledge of the general Px4 architecture

A good place to start are the links

- https://docs.px4.io/v1.12/en/concept/px4_systems_architecture.html
- https://docs.px4.io/v1.12/en/concept/architecture.html
- https://docs.px4.io/v1.12/en/flight_stack/controller_diagrams.html




## Changes to the source code

As of Nov 2021 the steps are

- Set up the developer toolchain following instructions in https://docs.px4.io/v1.12/en/dev_setup/dev_env_linux_ubuntu.html

- Download the code using

```
git clone https://github.com/PX4/PX4-Autopilot.git --recursive
```

- Do not build the code just yet
- Find v1.12.0 using

```
git tag -n
```

```
v1.11.1         Stable Release v1.11.1
v1.11.2         Stable Release v1.11.2
v1.11.3         Stable Release v1.11.3
v1.12.0         PX4 stable release v1.12.0
v1.12.0-1.10.0-ethzasl ramp in curvature dependent lower period bound
v1.12.0-21.10.0-ethzasl Use height rate setpoints
v1.12.0-beta1   v1.12.0 Beta 1
v1.12.0-beta2   v1.12.0 Beta 2
v1.12.0-beta3   v1.12.0 Beta 3
v1.12.0-beta4   v1.12.0 Beta 4
v1.12.0-beta5   v1.12.0 Beta 5
v1.12.0-beta6   PX4 Release Beta 6
:
```

- Checkout version v.1.12.0 using

```
git checkout tags/v1.12.0 -b v1.12.0-branch
```

- Now build the code using

```
make px4_fmu-v5_default
```

```
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tzo4/Dropbox/tomas/pennState/avia/
e
[327/1362] Performing build step for 'px4io_firmware'
[0/244] git submodule platforms/nuttx/NuttX/nuttx
[2/244] git submodule platforms/nuttx/NuttX/apps
[242/244] Linking CXX executable px4_io-v2_default.elf
Memory region         Used Size  Region Size  %age Used
           flash:       58508 B        60 KB     95.23%
            sram:        3856 B         8 KB     47.07%
[244/244] Creating /home/tzo4/Dropbox/to...ild/px4io_firmware/px4_io-v2_defau
[1360/1362] Linking CXX executable px4_fmu-v5_default.elf
Memory region         Used Size  Region Size  %age Used
      FLASH_ITCM:          0 GB      2016 KB      0.00%
      FLASH_AXIM:     1891617 B      2016 KB     91.63%
        ITCM_RAM:          0 GB        16 KB      0.00%
        DTCM_RAM:          0 GB       128 KB      0.00%
           SRAM1:       45748 B       368 KB     12.14%
           SRAM2:          0 GB        16 KB      0.00%
```

- Add a topic to capture mixer in/out information by following instructions in https://docs.px4.io/v1.12/en/middleware/uorb.html

- Create an application to test the newly created topic by following instructions in https://docs.px4.io/v1.12/en/modules/hello_sky.html
- You can check what files are using your newly created topics using

```
grep -R --exclude-dir={.git,build} "firefly_ctrlalloc" .
```

- In this articular example an app called `firefly` was created to read and write two new topics `firefly_ctrlalloc` and `firefly_delta`. The modified sourcode can be compiled and uploaded to a Pixhawk 4 (physical device) using a USB cable and running the command

```
make px4_fmu-v5_default upload
```

```
nsh> tzo4@E5-AERO-RHEA:~/Dropbox/tomas/pennState/avia/software/px4_sourcecode
[0/15] Performing build step for 'px4io_firmware'
ninja: no work to do.
[4/7] Linking CXX executable px4_fmu-v5_default.elf
Memory region         Used Size  Region Size  %age Used
      FLASH_ITCM:           0 GB      2016 KB      0.00%
      FLASH_AXIM:     1894929 B      2016 KB     91.79%
        ITCM_RAM:           0 GB        16 KB      0.00%
        DTCM_RAM:           0 GB       128 KB      0.00%
           SRAM1:       45748 B       368 KB     12.14%
           SRAM2:           0 GB        16 KB      0.00%
[6/7] uploading px4
Loaded firmware for board id: 50,0 size: 1894929 bytes (91.79%), waiting for

Attempting reboot on /dev/serial/by-id/usb-3D_Robotics_PX4_FMU_v5.x_0-if00 wi
If the board does not respond, unplug and re-plug the USB connector.

Found board id: 50,0 bootloader version: 5 on /dev/serial/by-id/usb-3D_Roboti
sn: 002800433138510e35393235
chip: 10016451
family: b'STM32F7[6|7]x'
revision: b'Z'
flash: 2064384 bytes
Windowed mode: False

Erase  : [====================] 100.0%
Program: [====================] 100.0%
Verify : [====================] 100.0%
Rebooting. Elapsed Time 28.702
```

- Then, the app `firefly` can be tested by connecting via mavlink

```
./Tools/mavlink_shell.py
```

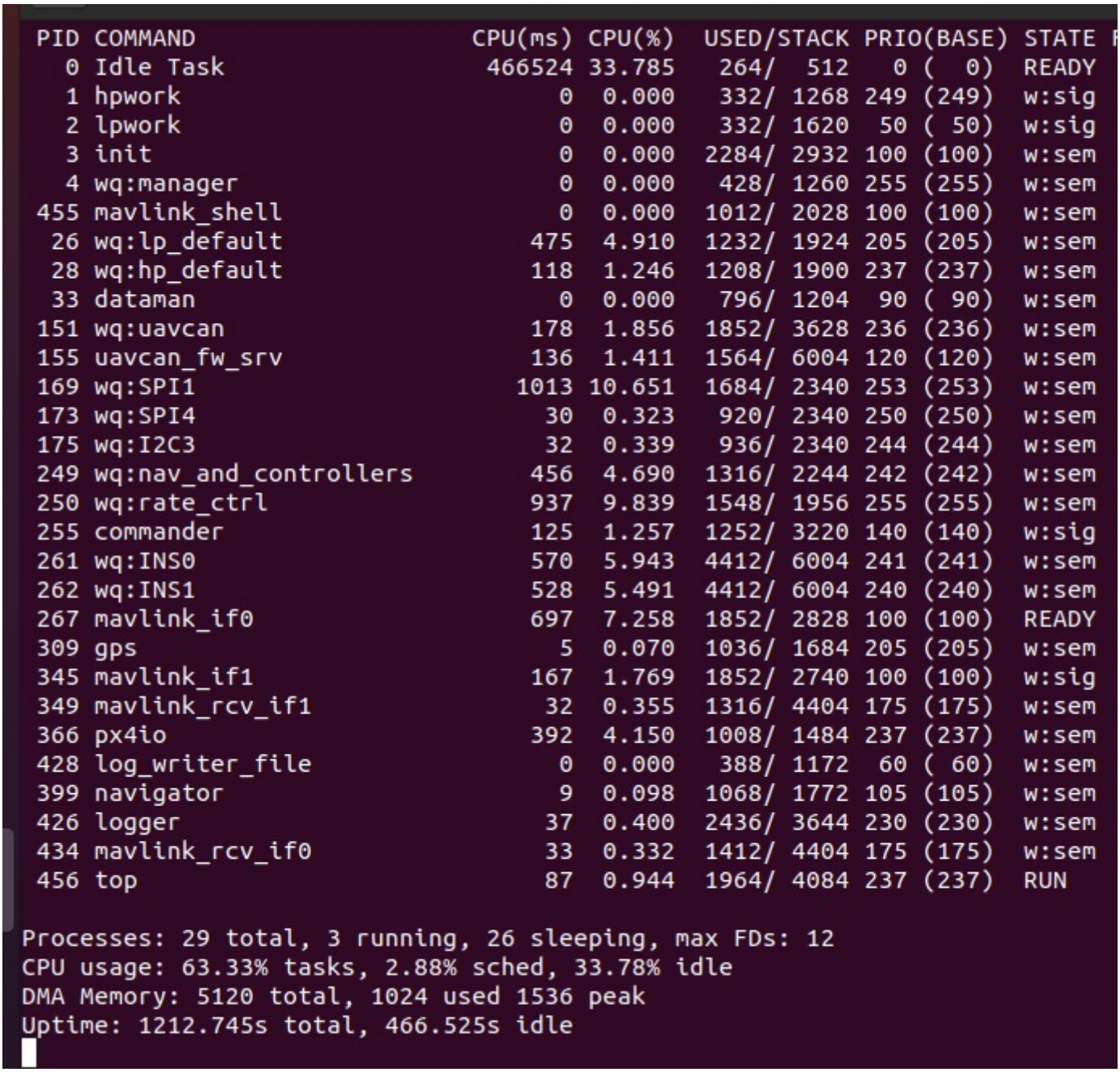
```
tzo4@E5-AERO-RHEA:~/Dropbox/tomas/pennState/avia/software/px4_sourcecode/PX4-
Using port /dev/serial/by-id/usb-3D_Robotics_PX4_FMU_v5.x_0-if00
Connecting to MAVLINK...

NuttShell (NSH) NuttX-10.1.0
nsh>
nsh> firefly write_delta 123 456
INFO  [firefly] firefly_main
INFO  [firefly] argc    4
INFO  [firefly] argv[0]=firefly
INFO  [firefly] argv[1]=write_delta
INFO  [firefly] argv[2]=123
INFO  [firefly] argv[3]=456
INFO  [firefly] write_firefly_delta
INFO  [firefly] Done
nsh> firefly read_delta
INFO  [firefly] firefly_main
INFO  [firefly] argc    2
INFO  [firefly] argv[0]=firefly
INFO  [firefly] argv[1]=read_delta
INFO  [firefly] read_firefly_delta
INFO  [firefly] firefly_delta.timestamp = 565616967.0000
INFO  [firefly] firefly_delta.delta = 123.0000, 123.0000, 456.0000, 456.0000,
INFO  [firefly] Done
nsh>
```

- Now we need to open QGroundControl and set some parameter values (https://docs.px4.io/v1.12/en/advanced_config/parameter_reference.html)

```
SYS_AUTOSTART   12001       Auto-start script index (https://dev.px4.io/master/en/airframes/airf
SYS_USE_IO      disabled    Set usage of IO board
```

- Make sure to reboot the vehicles after saving changes to te above parameters
- Now we need to look for the files that implement the mixer (control allocation unit). We can see the modules that are currently running using the command

```
top
```



```
 PID COMMAND                  CPU(ms) CPU(%)  USED/STACK PRIO(BASE) STATE
   0 Idle Task                466524 33.785   264/  512   0 (  0)  READY
   1 hpwork                        0  0.000   332/ 1268 249 (249)  w:sig
   2 lpwork                        0  0.000   332/ 1620  50 ( 50)  w:sig
   3 init                          0  0.000  2284/ 2932 100 (100)  w:sem
   4 wq:manager                    0  0.000   428/ 1260 255 (255)  w:sem
 455 mavlink_shell                 0  0.000  1012/ 2028 100 (100)  w:sem
  26 wq:lp_default               475  4.910  1232/ 1924 205 (205)  w:sem
  28 wq:hp_default               118  1.246  1208/ 1900 237 (237)  w:sem
  33 dataman                      0  0.000   796/ 1204  90 ( 90)  w:sem
 151 wq:uavcan                  178  1.856  1852/ 3628 236 (236)  w:sem
 155 uavcan_fw_srv              136  1.411  1564/ 6004 120 (120)  w:sem
 169 wq:SPI1                   1013 10.651  1684/ 2340 253 (253)  w:sem
 173 wq:SPI4                     30  0.323   920/ 2340 250 (250)  w:sem
 175 wq:I2C3                     32  0.339   936/ 2340 244 (244)  w:sem
 249 wq:nav_and_controllers     456  4.690  1316/ 2244 242 (242)  w:sem
 250 wq:rate_ctrl               937  9.839  1548/ 1956 255 (255)  w:sem
 255 commander                  125  1.257  1252/ 3220 140 (140)  w:sig
 261 wq:INS0                    570  5.943  4412/ 6004 241 (241)  w:sem
 262 wq:INS1                    528  5.491  4412/ 6004 240 (240)  w:sem
 267 mavlink_if0                697  7.258  1852/ 2828 100 (100)  READY
 309 gps                          5  0.070  1036/ 1684 205 (205)  w:sem
 345 mavlink_if1                167  1.769  1852/ 2740 100 (100)  w:sig
 349 mavlink_rcv_if1             32  0.355  1316/ 4404 175 (175)  w:sem
 366 px4io                      392  4.150  1008/ 1484 237 (237)  w:sem
 428 log_writer_file              0  0.000   388/ 1172  60 ( 60)  w:sem
 399 navigator                    9  0.098  1068/ 1772 105 (105)  w:sem
 426 logger                      37  0.400  2436/ 3644 230 (230)  w:sem
 434 mavlink_rcv_if0             33  0.332  1412/ 4404 175 (175)  w:sem
 456 top                         87  0.944  1964/ 4084 237 (237)  RUN

Processes: 29 total, 3 running, 26 sleeping, max FDs: 12
CPU usage: 63.33% tasks, 2.88% sched, 33.78% idle
DMA Memory: 5120 total, 1024 used 1536 peak
Uptime: 1212.745s total, 466.525s idle
```

- Another option is to directly check the status of modules we are interested in

```
mc_pos_control status
mc_att_control status
px4io status
```

```
nsh> mc_pos_control status
INFO  [mc_pos_control] running
nsh> mc_att_control status
INFO  [mc_att_control] running
nsh> px4io status
WARN  [px4io] loaded
protocol 4 hardware 2 bootloader 3 buffer 64B crc 0x255252255
8 controls 8 actuators 18 R/C inputs 2 analog inputs 0 relays
 px4io_status_s
        timestamp: 125095102  (0.121982 seconds ago)
        voltage_v: 0.0020
        rssi_v: 1.2380
        free_memory_bytes: 1608
        actuators: [-10000, -10000, -10000, -10000, -10000, -10000, -10000,
        servos: [900, 900, 900, 900, 900, 900, 900, 900]
```

- Another important componenet to check for are topics. Two important topics are `actuator_controls` y `actuator_outputs`. This can be checked using

```
listener actuator_controls_0
listener actuator_outputs
```

```
nsh> tzo4@E5-AERO-RHEA:~/Dropbox/tomas/pennState/avia/software/px4_sourcecod
Using port /dev/serial/by-id/usb-3D_Robotics_PX4_FMU_v5.x_0-if00
Connecting to MAVLINK...

NuttShell (NSH) NuttX-10.1.0
nsh>
nsh> listener actuator_controls_0

TOPIC: actuator_controls_0
 actuator_controls_s
        timestamp: 2477572309  (0.004078 seconds ago)
        timestamp_sample: 2477572073  (236 us before timestamp)
        control: [0.0005, -0.0529, 0.0001, 0.0000, 0.0000, 0.0000, 0.0000, -
nsh> listener actuator_outputs

TOPIC: actuator_outputs 2 instances

Instance 0:
 actuator_outputs_s
        timestamp: 2492710854  (0.005283 seconds ago)
        noutputs: 8
        output: [900.0000, 900.0000, 900.0000, 900.0000, 900.0000, 900.0000,

Instance 1:
 actuator_outputs_s
        timestamp: 2492725317  (0.000285 seconds ago)
        noutputs: 8
        output: [900.0000, 900.0000, 900.0000, 900.0000, 900.0000, 900.0000,
```

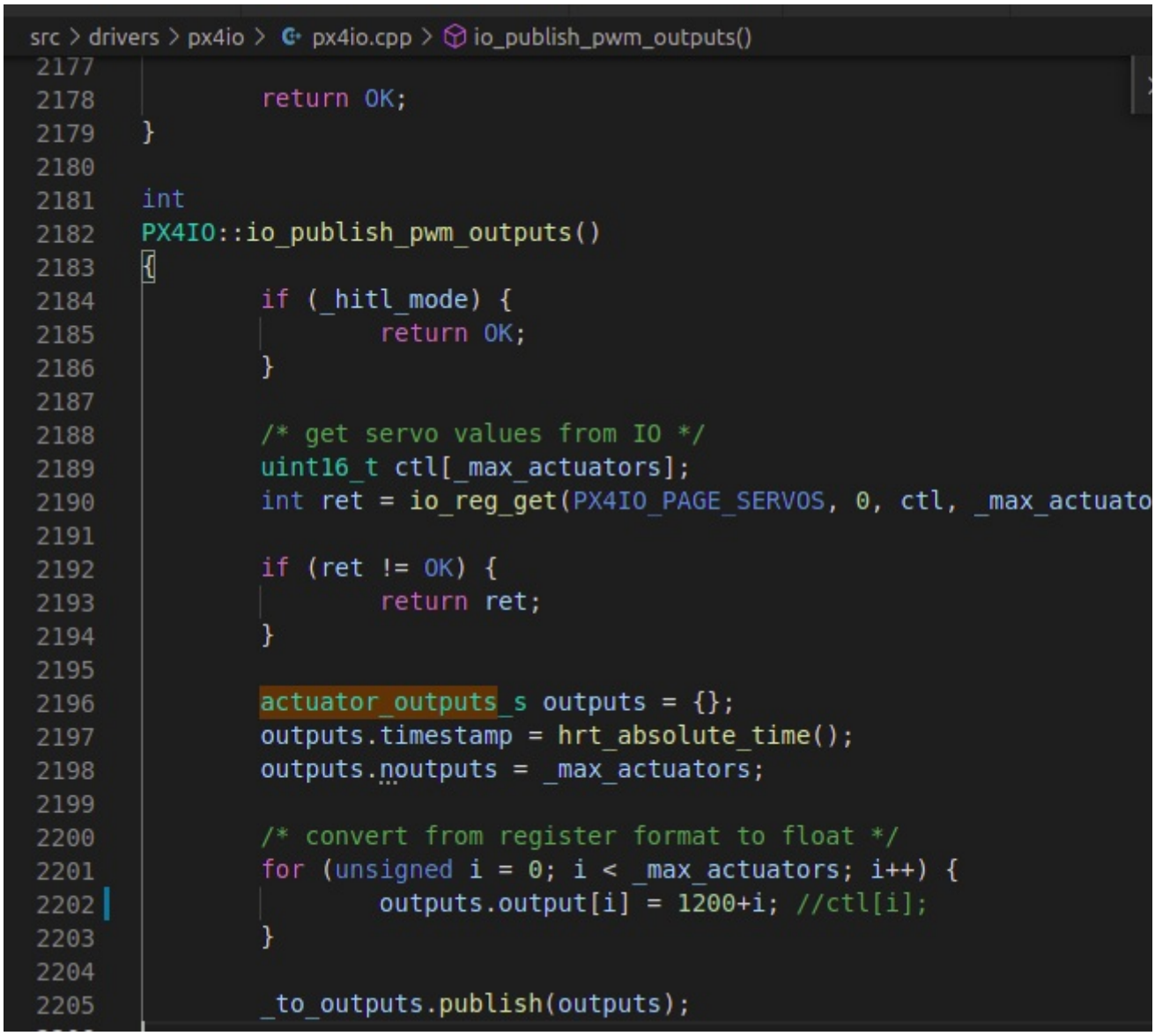
- The module we are looking for is `px4io` we can find it running

```
find . -name "px4io.c*"
```

```
tzo4@E5-AERO-RHEA:~/Dropbox/tomas/pennState/avia/software/px4_sourcecode/PX4-
./src/drivers/px4io/px4io.cpp
./src/modules/px4iofirmware/px4io.c
./build/px4_fmu-v5_default/external/Build/px4io_firmware/src/modules/px4iofir
./build/px4_fmu-v5_default/src/drivers/px4io/CMakeFiles/drivers__px4io.dir/px
tzo4@E5-AERO-RHEA:~/Dropbox/tomas/pennState/avia/software/px4_sourcecode/PX4-
```

- We can edit the file `src/drivers/px4io/px4io.cpp` to test its effect on the topic

`actuator_outputs`



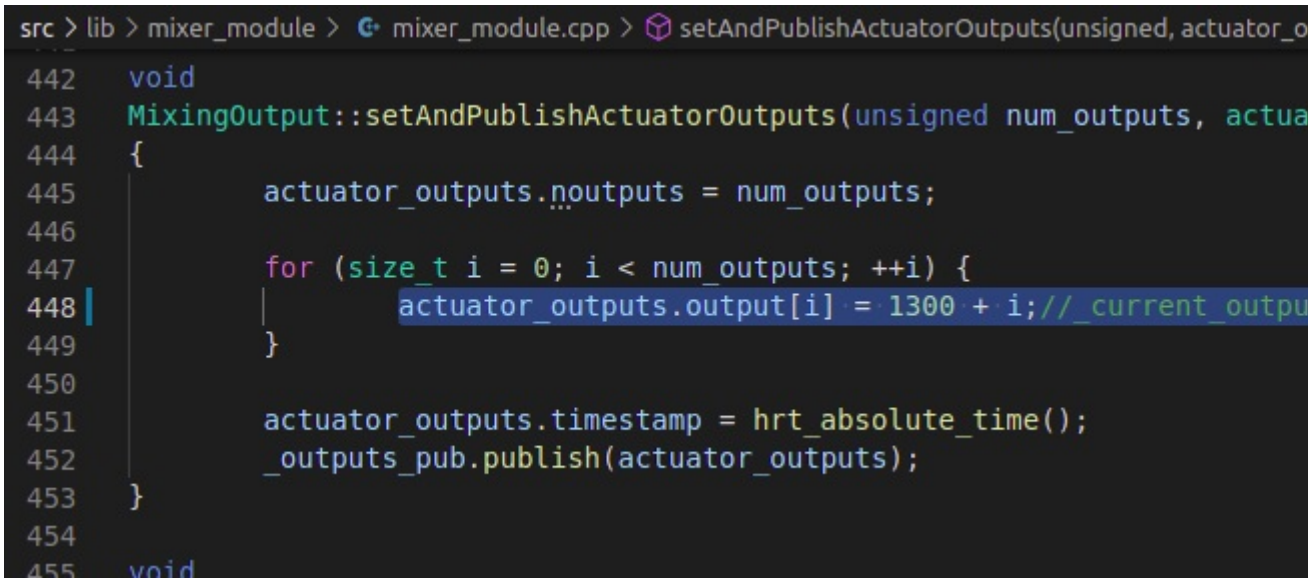

- After uploading the code to the board and connecting to the mavlink shell we can verify the change in `actuator_outputs -i 0`. This means that `src/drivers/px4io/px4io.cpp` manages the interaction with the IO board of the Pixhawk 4 but NOT the mixing (control allocation). The IO board receives `actuator_controls_0` data and sends back

`actuator_outputs` data.

```
nsh> listener actuator_outputs

TOPIC: actuator_outputs 2 instances

Instance 0:
 actuator_outputs_s
        timestamp: 13899977  (0.014641 seconds ago)
        noutputs: 8
        output: [1200.0000, 1201.0000, 1202.0000, 1203.0000, 1204.0000,

Instance 1:
 actuator_outputs_s
        timestamp: 13925475  (0.003363 seconds ago)
        noutputs: 8
        output: [900.0000, 900.0000, 900.0000, 900.0000, 900.0000, 900.0
```

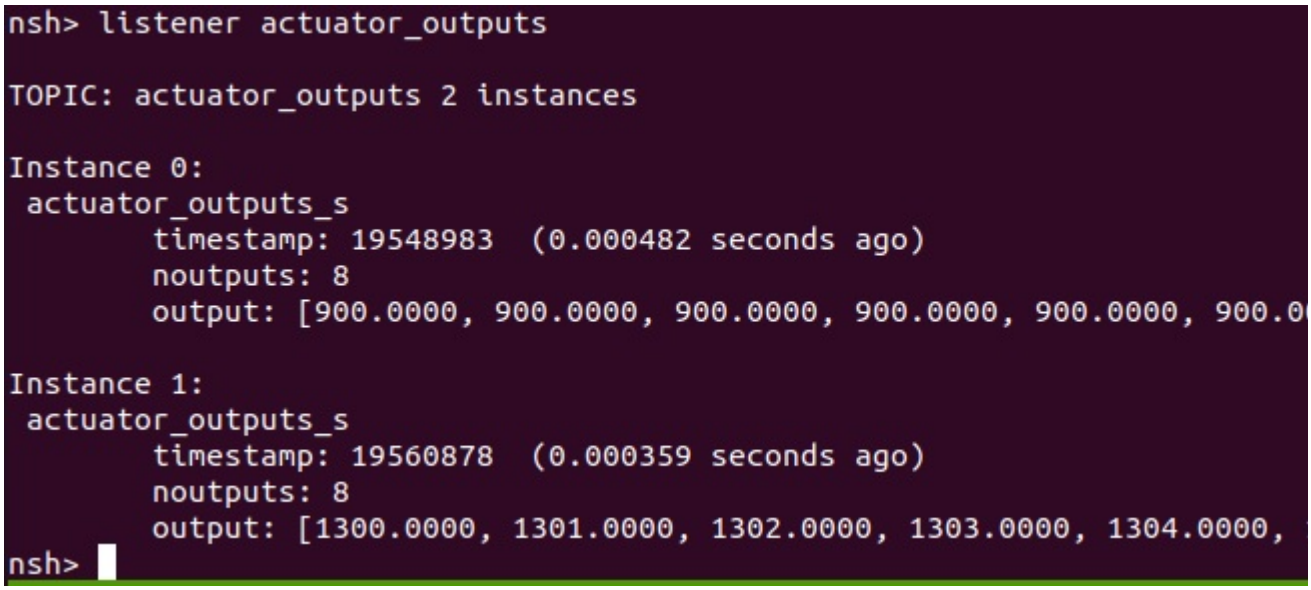
- In the same way we can edit the file `src/lib/mixer_module/mixer_module.cpp` to test its

effect on the topic `actuator_outputs`




- After uploading the code to the board and connecting to the mavlink shell we can verify the change in `actuator_outputs -i 1`. This means that `src/lib/mixer_module/mixer_module.cpp` manages the interaction with the main channel of Pixhawk 4 but NOT the

IO board.


nsh> listener actuator_outputs
TOPIC: actuator_outputs 2 instances
Instance 0:
actuator_outputs_s
timestamp: 19548983 (0.000482 seconds ago)
noutputs: 8
output: [900.0000, 900.0000, 900.0000, 900.0000, 900.0000, 900.0
Instance 1:
actuator_outputs_s
timestamp: 19560878 (0.000359 seconds ago)
noutputs: 8
output: [1300.0000, 1301.0000, 1302.0000, 1303.0000, 1304.0000,
nsh>