Fixed Wing UAV Documentation

Joe Lorenzetti

Contents

1 Skywalker Hardware

1. Airframe: ReadymadeRC 1720mm Skywalker EPO FPV Plane Kit[1]
2. Motor: Finwing M2815 (included in kit)
3. Servos: Finwing 17g/9g servo (included in kit)
4. Autopilot: Pixhawk 4[2]
5. Companion computer: Odroid XU4[3]
6. Radio Transmitter: Spektrum DX6e[4]
7. Radio Receiver: Spektrum Satellite Receiver[5]

1.1 Airframe

The ReadymadeRC Skywalker is a large aircraft, designed to carry an FPV payload. The airframe has a wingspan of

Figure 1: Skywalker 1720 airframe

1.72 meters (67.7 inches) and is almost entirely constructed out of EPO foam, with some parts (e.g. the tail boom and wing spar) made of carbon fiber. As it was designed to carry an FPV payload, the Skywalker is equipped with a large payload bay near the nose, which proved ideal to carry this project's electronics as well as a large, high-capacity battery. The airframe came with a motor, servos, and an electronic speed control (ESC). The motor is the Finwing 2815 1270KV, a 140g motor capable of drawing 45A, as listed on Finwing's website. The ESC is a Finwing 60A model with a Battery Eliminator Circuit (BEC) capable of providing 3A at 5.5V. The servos are 2 "17g" servos (actually 19g) on the ailerons, and 2 "9g" servos on the rudder and elevator.

[1] https://www.readymaderc.com/products/details/skywalker-1720mm-epo-fpv-kit
[2] https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk4.html
[3] https://www.hardkernel.com/shop/odroid-xu4-special-price/
[4] https://www.spektrumrc.com/Products/Default.aspx?ProdId=SPMR6650
[5] https://www.spektrumrc.com/Products/Default.aspx?ProdID=SPM9645

The Skywalker does not balance at the listed CG point, 75mm (3in) behind the leading edge of the wing unless a lot of weight is put in the front. Even after moving everything as far toward the nose as possible, our Skywalker was still balanced 20mm behind this point. To remedy this, two metal weights were placed just behind the nose of the aircraft.

1.2 Radio Transmitter

The Spektrum DX6e is a radio transmitter designed to supports 6 channels (which is sufficient to control our aircraft); in this case there are 4 channels required to control the aircraft and 2 customizable channels. In order to connect this transmitter to the Pixhawk 4, a satellite receiver compatible with the Pixhawk 4 is a must (the receiver provided when buying the Dx6e can only be used if the aircraft has no on-board computer mounted).

Figure 2: Dx6e radio transmitter

1.3 Autopilot

The Pixhawk 4 is the newest revision of the Pixhawk autopilot, which has become very popular in the autonomous vehicle space. The Pixhawk 4 has more processing power, more RAM, and more I/O than previous versions of the Pixhawk. The Pixhawk 4 has an inbuilt accelerometer, gyroscope, magnetometer, and barometer in order to determine its orientation and altitude- important information used to control an aircraft in flight. To connect to other peripherals, it uses the Pixhawk connector standard, as outlined at this website[6]. This autopilot was purchased as part of a combo with a GPS module, which contained a ublox Neo-M8N GPS receiver and an integrated magnetometer, as well as the safety switch. While its wide variety of I/O does allow a lot of peripherals to connect to it, the connector standard outlined above does mean that some special cables are required to connect to it. Soldering other cables to Pixhawk standard connectors is a workaround for this problem. It is also possible to find these cables online, though not without some difficulty.

[6] https://pixhawk.org/pixhawk-connector-standard/

Figure 3: Pixhawk 4

1.4 Companion computer

The Pixhawk communicates to companion computers over the Telem2 serial port, which uses a UART protocol for sending MAVLink messages[7]. On the Skywalker we are using the Odroid XU4 computer as it is relatively low power requirements but has sufficient processing power.

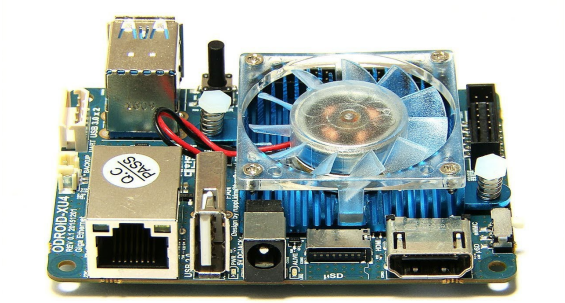

Figure 4: ODROID-XU4

1.5 Wiring

There are several components that need connected: Pixhawk 4, Odroid XU4, LiPO battery, GPS module, airspeed sensor, Spektrum Radio receiver, telemetry radio, propeller motor, and control surface servos.

1.5.1 Pixhawk 4

The Pixhawk 4 needs a connection to all of the components. A wiring diagram for the PX4 is shown in Figure 5.

1. The POWER1 port on the Pixahwk gets connected to the PWR1 port on the Power Management Board (PMB).
2. The GPS module (w/ compass, safety switch, and buzzer) gets connected to the GPS MODULE port.
3. The Spektrum satellite receiver (SPM9645) is connected to the DSM/SBUS RC port.
4. The telemetry radio gets connected to the TELEM1 port.
5. The TELEM2 port gets connected through the FTDI chip to the Odroid's USB port.
6. The I/O PWM OUT port gets connected to the PMB FMU-PWM-in port (which connects to the FMU-PWM-out pins).[8]

[7] `https://dev.px4.io/master/en/companion_computer/pixhawk_companion.html`

[8] From `https://docs.px4.io/master/en/airframes/airframe_reference.html` the standard plane configuration sends all outputs through the MAIN channels, which are output on the I/O PWM OUT pins of the Pixhawk 4.

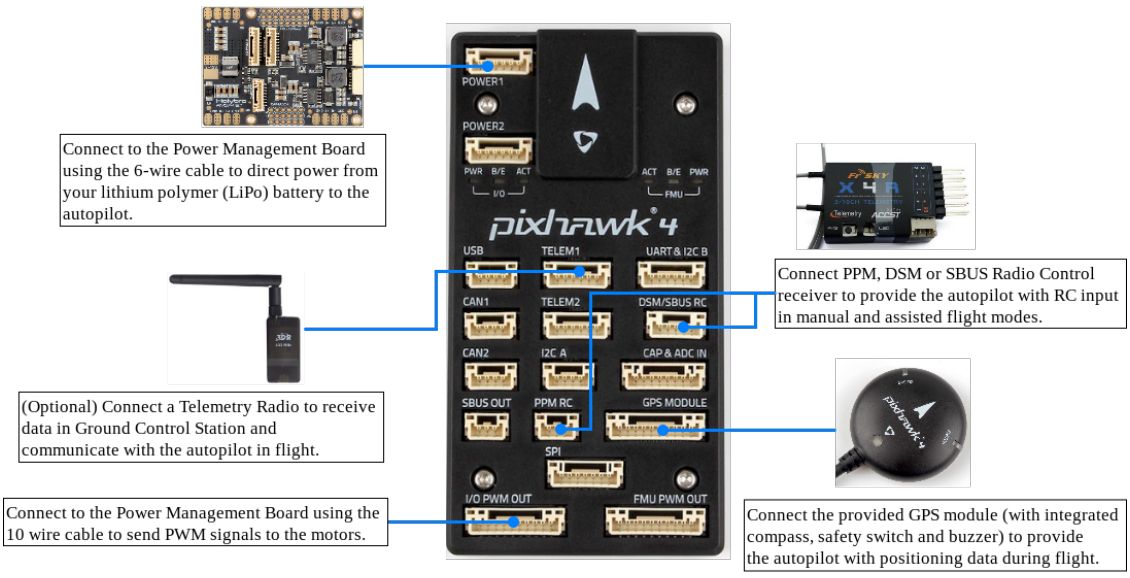

Figure 5: Pixhawk 4 Wiring Diagram

7. The airpseed sensor gets connected to the I2C A port.

1.5.2 Odroid

The Odroid gets connected to the Pixhawk through USB as previously mentioned. The only other wiring component is the power supply. The power is connected through the 5V port on the Odroid by using a modified cable to connect to the PWR2 port on the PMB.

1.5.3 Aircraft

The aircraft's core components that need to be connected are the propeller motor and the servos. These are all connected to the FMU-PWM-out rail on the PMB as follows[9]:

1. Aileron gets connected to FMU-PWM-out pin 1.
2. Elevator gets connected to FMU-PWM-out pin 2.
3. Throttle gets connected to FMU-PWM-out pin 3.
4. Rudder gets connected to FMU-PWM-out pin 4.

Note that the power +/- rails must be powered externally. However this is accomplished by simply plugging in the prop motor because it has a BEC that supplies power.

[9]The pin numbers are specific to the airframe and can be found at `https://docs.px4.io/master/en/airframes/airframe_reference.html`

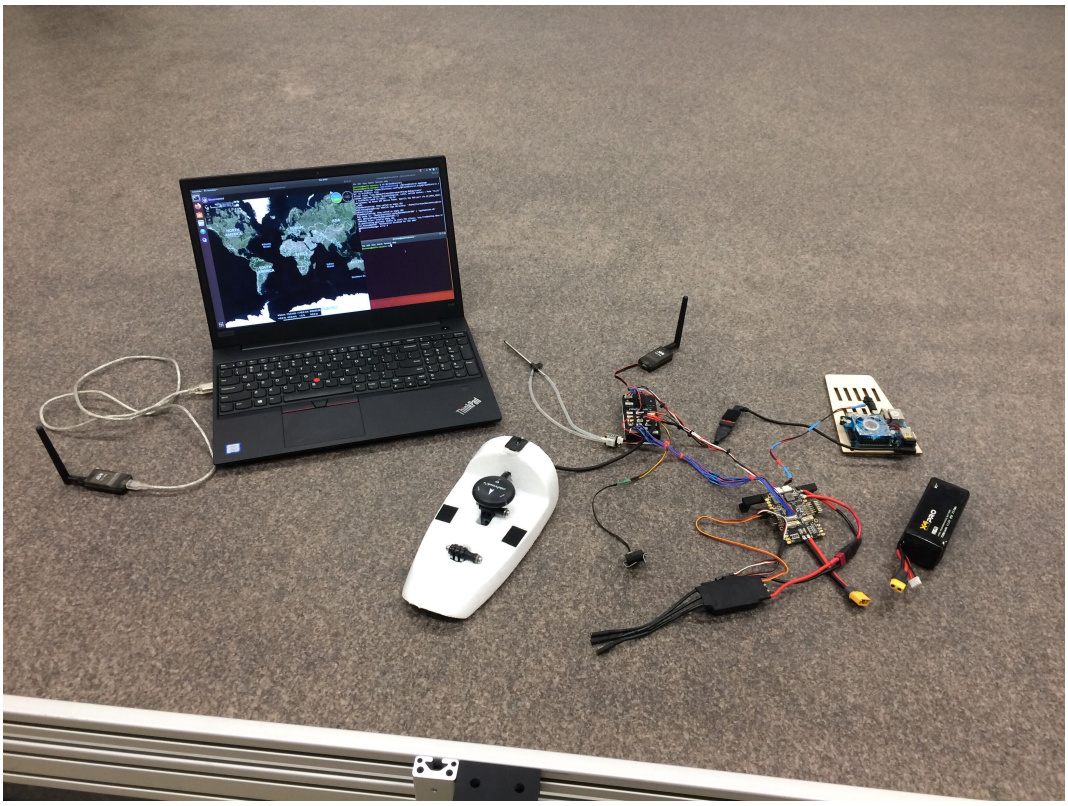

Figure 6: All electronic components of the Skywalker: ground station laptop running QGround control, Odroid companion computer, Pixhawk autopilot, PMB, LiPo battery, ESC, etc.

2 Software

2.1 QGroundControl

QGroundControl[10] is computer software used to control and configure the Pixhawk and radio transmitter. Install via website instructions.

2.2 PX4 Autopilot

PX4 is the autopilot software stack that runs on the Pixhawk, or can be run in SITL simulation with Gazebo. The PX4 firmware that will be used here has been forked from the main development repository. In particular, it is located in the `StanfordASL/Firmware` repository under the branch `asl_fixedwing-v1.10.1`, which can be locally cloned by:

```
cd ~
git clone https://github.com/StanfordASL/Firmware.git
cd Firmware
git checkout asl_fixedwing-v1.10.1
```

This repository includes a script that automatically downloads and installs any dependencies with the command

```
bash ~/Firmware/Tools/setup/ubuntu.sh --no-sim-tools
sudo reboot
```

where the computer should be rebooted after installation. The option `no-sim-tools` avoids installing additional simulation dependencies that we either do not need or will install ourselves separately.

[10] `http://qgroundcontrol.com/`

2.3 ROS + MAVROS

For simulations, we are using the full desktop install of ROS Melodic[11] on Ubuntu 18.04. ROS communication to PX4 is done over MAVlink communication protocol, and the MAVROS library is therefore installed to handle this. In particular, the source installation[12] of MAVROS is used. Before following the MAVROS source install instructions, a couple of other dependencies are required:

```
sudo apt-get install python-catkin-tools python-rosinstall-generator -y
```

It is also required that you create the ROS workspace at this point, following tutorial on ROS website[13]. It is also a good idea to add to add to your bashrc:

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

if it is not already there. Note that it would be a good idea to check the most up to date version of the ROS install script[14] to check for any extra steps. For example I was missing some dependencies[15].

2.4 Gazebo

Gazebo 9 should be installed automatically in the full desktop install of ROS Melodic. I also ended up having a build error (missing GSTREAMER_LIBRARIES) when trying to run the SITL with Gazebo simulation and fixed it by (re)installing the dependencies:

```
sudo apt-get install libprotobuf-dev libprotoc-dev protobuf-compiler libeigen3-dev
sudo apt-get install libxml2-utils python-rospkg python-jinja2
sudo apt-get install libgstreamer-plugins-base1.0-dev gstreamer1.0-plugins-bad
sudo apt-get install gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-ugly -y
```

To test that everything has been installed correctly, a Gazebo + PX4 + ROS simulation[16] can be run by:

```
cd ~/Firmware
DONT_RUN=1 make px4_sitl_default gazebo
source Tools/setup_gazebo.bash $(pwd) $(pwd)/build/px4_sitl_default
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:$(pwd)
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:$(pwd)/Tools/sitl_gazebo
roslaunch px4 mavros_posix_sitl.launch
```

Here the first line builds PX4 for gazebo target without launching the simulation, and then the packages are added to the ROS path. The launch file then launches Gazebo, MAVROS, and the PX4 stack. A nice "offboard control" tutorial for learning about the interactions between PX4 and ROS is located on the website[17].

2.4.1 Adding Fixedwing Model

A new model `aslfixedwing` was also added to PX4, which will leverage a new mixer file. This was done by creating the file `2107_aslfixedwing` in the following directories in Firmware:

```
~/Firmware/ROMFS/px4fmu_common/init.d/airframes
~/Firmware/ROMFS/px4fmu_common/init.d-posix
```

The model sdf file that will be used in Gazebo is located in the `asl_fixedwing` repository. Therefore before Gazebo can find this model the following should be added to a script or .bashrc:

```
export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:~/catkin_ws/src/asl_fixedwing/models
```

[11] http://wiki.ros.org/melodic/Installation/Ubuntu
[12] https://docs.px4.io/master/en/ros/mavros_installation.html
[13] http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment
[14] https://raw.githubusercontent.com/PX4/Devguide/master/build_scripts/ubuntu_sim_ros_melodic.sh
[15] https://raw.githubusercontent.com/PX4/Devguide/master/build_scripts/ubuntu_sim_common_deps.sh
[16] https://dev.px4.io/v1.10/en/simulation/ros_interface.html
[17] https://dev.px4.io/v1.10/en/ros/mavros_offboard.html

The custom mixer file is defined in:

`~/Firmware/ROMFS/px4fmu_common/mixers/aslfixedwing.main.mix`

Add the source and export commands from above into a bash script that can be sourced easily, or alternatively perhaps they could be directly placed into the .bashrc.

3 Setup/Install

3.1 Desktop for Gazebo Simulation

1. Install PX4 via instructions in Section 2.2.
2. Install ROS and MAVROS via instructions in Section 2.3.
3. Install Gazebo via instructions in Section 2.4.
4. Install the `asl_fixedwing` repository from `https://github.com/StanfordASL/asl_fixedwing`. Follow instructions for setup in the README.

3.2 Desktop for Flight Ground Station

1. Install QGroundControl via instructions in Section 2.1.
2. Install PX4 via instructions in Section 2.2.

3.3 Pixhawk

1. Build PX4: navigate to the Firmware directory and run

   ```
   make px4_fmu-v5_default
   ```

   which may take a while. Once the build is complete, the Pixhawk board can be attached to the computer via USB and flashed via

   ```
   make px4_fmu-v5_default upload
   ```

   The upload can be verified in QGroundControl on the summary tab under Airframe.
2. Configure the Telem2 port to be able to connect to the Odroid. In QGroundControl, set `MAV_1_CONFIG` to `TELEM2`, set the baudrate `SER_TEL2_BAUD` to 921600, and set `MAV_1_MODE` to `Onboard`.
3. Set up telemetry radios. First install APMPLanner2 by building from source following instructions here[18]. Then try to connect the radios using the instructions here[19]. Note a warning message told me to first ensure I had serial privilege by running:

   ```
   sudo usermod -a -G dialout $USER
   ```

   and logging out and back in. If this is not successful try updating the Firmware on both.
4. Calibrate all sensors for Pixhawk in QGroundControl.
5. Calibrate radio receiver. Also set the flight modes so that one is `Manual` and another is `Offboard`.
6. Verify that the servos all move in the proper direction based on commands given! There is an automated script `bodyrate_signs_verification` in the `asl_fixedwing` repo that can send commands for positive body rates to check that the actuators move correctly. Also check the manual mode directions too! If the aileron, elevator, or rudder servos need inverted you can modify the appropriate PX4 `PWM_MAIN_` parameter in QGroundControl.

[18] `https://github.com/ArduPilot/apm_planner`
[19] `https://ardupilot.org/copter/docs/common-sik-telemetry-radio.html`

3.4 Odroid

Setup is best done by hooking the Odroid up to a monitor + mouse + keyboard.

1. Make sure the Odroid is running Ubuntu 18.04 LTS.
2. `sudo apt install git screen curl`
3. Install ROS and MAVROS via instructions in Section 2.3. Except instead of doing a full desktop install of ROS only do the `ROS-Base` install. After installing MAVROS source, run the build by

   ```
   catkin build -j2
   ```

   to ensure the Odroid doesn't run out of memory while building. Note that the build could take a while. Run `source /.bashrc` after the build is complete.
4. Install the `asl_fixedwing` repository from `https://github.com/StanfordASL/asl_fixedwing`. Follow instructions for setup in the README. When building qpOASES, note that you will need to go into the file `qpOASES/make_linux.mk` and replace

   ```
   LIB_BLAS =   /usr/lib/x86_64-linux-gnu/libblas.so
   LIB_LAPACK = /usr/lib/x86_64-linux-gnu/liblapack.so
   ```

   to be

   ```
   LIB_BLAS =   /usr/lib/arm-linux-gnueabihf/libblas.so
   LIB_LAPACK = /usr/lib/arm-linux-gnueabihf/liblapack.so
   ```
5. Set up a network interface. This procedure is based on[20]. Note that first thing to check is that the chosen Wifi dongle supports the "AP" interface. This can be done by plugging in the dongle and running:

   ```
   iw list
   ```

   and checking if "AP" is listed under "Supported interface modes:". Now ensure everything is up to date and install some useful tools

   ```
   sudo apt-get update
   sudo apt-get upgrade
   sudo apt-get install vim hostapd isc-dhcp-server
   ```

   Then run `ifconfig` to see the name of the Wifi port, which is likely `wlan0`. Now make a backup of the network interface config file and then open it up:

   ```
   sudo cp /etc/network/interfaces /etc/network/interfaces.original
   sudo vim /etc/network/interfaces
   ```

   Add the following to the file:

   ```
   auto wlan0
   iface wlan0 inet static
       address 192.168.100.1
       netmask 255.255.255.0
   ```

   Now open the following file:

[20] `https://ardupilot.org/dev/docs/odroid-wifi-access-point-for-sharing-files-via-samba.html`

```
sudo vim /etc/hostapd/hostapd.conf
```

Add the following to the file:

```
interface=wlan0
driver=nl80211
ssid=odroid
hw_mode=g
channel=11
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=odroidwifi
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Note that the wpa passphrase is the password for the wifi connection! Now try running (after a reboot):

```
sudo reboot
sudo hostapd /etc/hostapd/hostapd.conf
```

If this is successful then go ahead and enable this to start on bootup:

```
sudo vim /etc/default/hostapd
```

and ensure that the following line is present:

```
DAEMON_CONF=/etc/hostapd/hostapd.conf
```

then run

```
sudo systemctl enable hostapd
```

which should set hostapd to run on boot.

Now make a copy of the DHCP configuration file and open it:

```
sudo cp /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf.original
sudo vim /etc/dhcp/dhcpd.conf
```

and ensure it has the following lines added and uncommented:

```
option domain-name "odroid.local";
option domain-name-servers dns.odroid.local;
default-lease-time 600;
max-lease-time 7200;
authoritative;
log-facility local7;
subnet 192.168.100.0 netmask 255.255.255.0 {
range 192.168.100.100 192.168.100.200;
}
```

Then run:

```
sudo -s
(for i in $(seq 100 200); do echo 192.168.100.$i client$i; done) >> /etc/hosts
```

Then reboot one more time: `sudo reboot`, and check to see if you can SSH into the Odroid by connecting to its network (password `odroidwifi`) and running:

```
ssh odroid@192.168.100.1
```

To return the Wifi dongle to connecting to other networks simply comment out the lines that were added to `/etc/network/interfaces` and run:

```
sudo service network-manager restart
```

6. Set up communication link to Pixhawk[21]. First plug in the Pixhawk into the desktop ground station computer via USB to provide power. Then on the Odriod run `lsusb`, then plug in the USB-serial adapter cable that connects to Pixhawk Telem2 port and run `lsusb` on the Odroid again. A new line should appear, for example

   ```
   Bus 003 Device 015: ID 0403:6001 Future Technology ...
   ```

   Then in the file `/etc/udev/rules.d/99-pixhawk.rules` add the line

   ```
   SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", SYMLINK+="ttyPixhawk"
   ```

   where idVendor and idProduct are set based on the lsusb output. Finally, run

   ```
   sudo usermod -a -G tty odroid
   sudo usermod -a -G dialout odroid
   ```

   After a reboot you should be able to run:

   ```
   roslaunch asl_fixedwing pixhawk_mavros.launch
   ```

   and it should work! If it doesn't make sure you have set the appropriate parameters on the Pixhawk as noted in Section 3.3.

[21] https://docs.px4.io/master/en/companion_computer/pixhawk_companion.html

4 Development Guide

4.1 Control Inputs/Commands

The standard fixed-wing attitude controller in PX4 is shown in Figure 7. Through MAVROS there are several points at

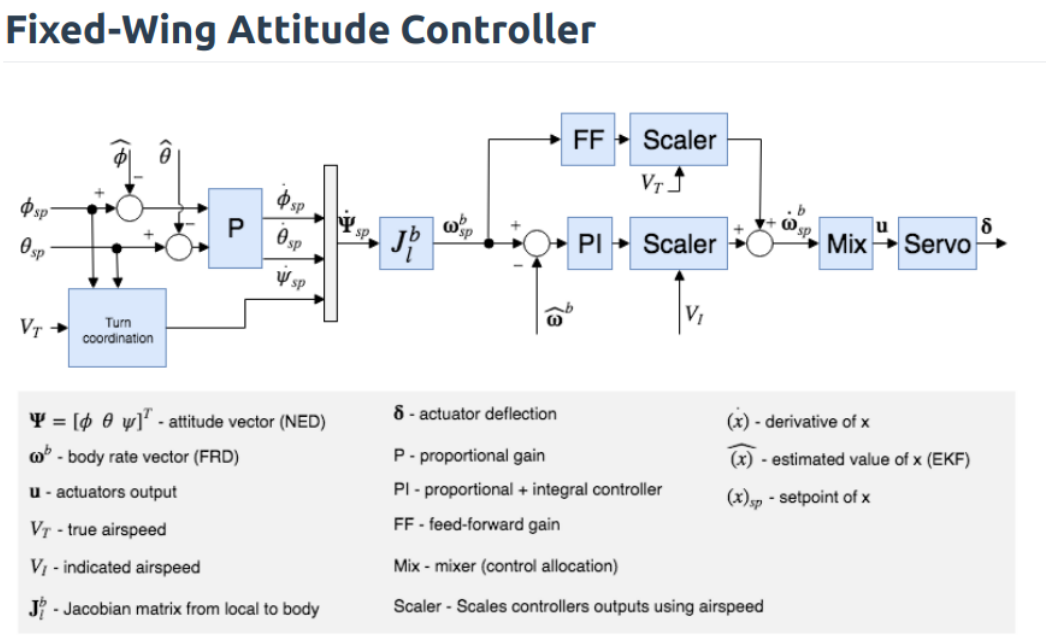

Figure 7: PX4 Fixed Wing Attitude Controller

which this control loop can be entered. Each corresponds to different MAVROS topics that should be used to publish commands:

1. `mavros/actuator_control` with `group_mix = 0` can be used to send roll, pitch, yaw, and thrust commands that are normalized to be in $(-1, 1)$ (and $(0, 1)$ for thrust). These commands enter the control loop right before the mixer. While they can be thought of as acceleration setpoints, these values actually just pass through the mixer and go directly to the servos. See Section 4.1.1 for more details.

2. `mavros/setpoint_raw/attitude` with `type_mask = AttitudeTarget::IGNORE_ATTITUDE` can be used to send roll, pitch, yaw body rates as well as thrust commands. This is essentially setting ω^b setpoints to the PX4 controller, which then tracks the setpoints with a lower level controller. Without the type mask set you can command attitude setpoints, which are an even higher-level input.

4.1.1 Actuator Control

Actuator controls can be sent over MAVROS using the `mavros/actuator_control` topic by:

```
mavros_msgs::ActuatorControl act_cmd;
act_cmd.group_mix = mavros_msgs::ActuatorControl::PX4_MIX_FLIGHT_CONTROL;
act_cmd.controls[0] = ...; // roll (-1,1)
act_cmd.controls[1] = ...; // pitch (-1,1)
act_cmd.controls[2] = ...; // yaw (-1,1)
act_cmd.controls[3] = ...; // thrust (0,1)
```

Based on the mixing files that exist in `Firmware/ROMFS/px4fmu_common/mixers(-sitl)/aslfixedwing(_sitl).main.mix` these commands get mapped by:

```
act_cmd.controls[0] -> left and right ailerons (anti-symetrically)
act_cmd.controls[1] -> elevator
act_cmd.controls[2] -> rudder
act_cmd.controls[3] -> propeller
```

Since the commands are sent as normalized units, it is important to know the mapping from normalized command to control surface deflection and thrust for each of these cases.

The mapping from command to control surface deflection will be assumed to be linear:

$$\delta = c_\delta u_\delta + \delta_0,$$

where δ is the control surface deflection angle in radians and u_δ is the normalized command. This leads to an inverse mapping of:

$$u_\delta = \min\{1, \max\{0, \frac{1}{c_\delta}(\delta - \delta_0)\}\}. \tag{1}$$

For the thrust model, we will assume a nominal thrust model[22] of the form:

$$T = C_T(V,\omega)\rho\Big(\frac{\omega}{2\pi}\Big)^2 D^4, \quad C_T(V,\omega) \approx C_{T,0}(1 - m\frac{2\pi V}{\omega D}),$$

where C_T is a thrust coefficient, V is the relative airspeed in the direction parallel to propeller axis in [m/s], ω is the rotation speed of the propeller in [rad/s], ρ is the air density in [kg/m^3], D is the propeller diameter in meters, and m is the slope of the (assumed to be linear) C_T vs. $J = \frac{2\pi V}{\omega D}$ table. An example of this plot for a similar propeller is shown in Figure 8.

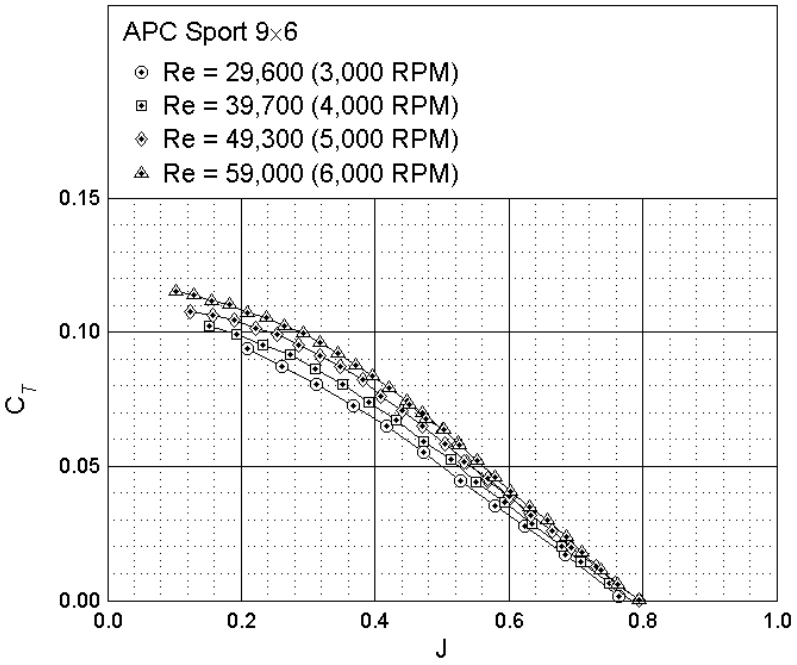

Figure 8: Thrust coefficient plot for APC Sport 9x6 propeller (from UIUC Propeller Data Site).

By combining the constants together and assuming a linear mapping from command to prop spin rate:

$$\omega = c_\omega u_T,$$

[22]Derived from `https://m-selig.ae.illinois.edu/props/propDB.html`

this expression becomes (note c_ω also gets merged into the other constants):

$$T = c_{T,0}(1 - c_{T,V\omega}\frac{V}{u_T})u_T^2,$$

which leads to an inverse mapping:

$$u_T = \min\{1, \max\{0, \frac{1}{2}\Big(c_{T,V\omega}V + \sqrt{(c_{T,V\omega}V)^2 + \frac{4T}{c_{T,0}}}\Big)\}\}. \tag{2}$$

Experimentally the coefficient $c_{T,0}$ could be determined on a static thrust stand where $V = 0$. The constant $c_{T,V\omega}$ is essentially given by a velocity V and prop command $u_T > 0$ that yield a thrust of zero. From the plot above this seems to occur when $J = 0.8$ and therefore perhaps a good approximation is $c_{T,V\omega} = 2\pi/(0.8Dc_\omega)$, but this might need to be tuned. For the ten inch prop we have this might be around $c_{T,V\omega} \approx 0.085$ (with appropriate units). Thus when the spin rate is around 3000 RPM (314 rad/s) a velocity of $V = 12$ m/s would produce almost no thrust according to this model.

4.1.2 Body Rate Control

Body rate commands can be sent over MAVROS using the `mavros/setpoint_raw/attitude` topic by:

```
mavros_msgs::AttitudeTarget att_sp;
att_sp.type_mask = mavros_msgs::AttitudeTarget::IGNORE_ATTITUDE;
att_sp.body_rate.x = ...; // body roll rate [rad/s]
att_sp.body_rate.y = ...; // body pitch rate [rad/s]
att_sp.body_rate.z = ...; // body yaw rate [rad/s]
att_sp.thrust = ...; // thrust (0,1)
```

It is important to note that MAVROS uses a Forward, Left, Up coordinate frame. Thus a positive pitch rate is actually a nose down maneuver. So make sure to convert from standard FRD to FLU before sending the commands.

4.2 Reference Frames and Nomenclature

PX4 defines the local inertial reference frame using NED convention (x points north, y is east, and z is down). The aircraft body frame uses a FRD convention (x points forward, y is right, and z is down). The FRD/NED are aligned with zero rotation and thus are often used interchangeably. MAVROS defines the local inertial reference frame using ENU convention (x points east, y is north, and z is up). The "ENU" aircraft body frame is the same as FLU convention (x points forward, y is left, and z is up).

MAVROS automatically performs the conversions, which means that the onboard controller needs to convert back and forth as well. In particular we consider a flat-Earth fixed inertial reference frame that is denoted by $\mathcal{I}$ and follows the standard North-East-Down (NED) convention, and consider a body-fixed frame denoted by $\mathcal{B}$ that is expressed with a Forward-Right-Down (FRD) convention. The position of the body frame with respect to the inertial frame origin is denoted by $P_{b/i}$ (i.e. the position of b relative to i). When expressed in inertial frame coordinates this is denoted by $P^{\mathcal{I}}_{b/i} = [x_i, y_i, z_i]^\top$. The inertial velocity of the aircraft is therefore denoted as $V_{b/\mathcal{I}}$ (i.e. the velocity of b as seen from an observer fixed in $\mathcal{I}$). When this vector is expressed in terms of body frame coordinates it is written as $V^{\mathcal{B}}_{b/\mathcal{I}} = [u, v, w]^T$. The angular velocity of the body frame with respect to the inertial frame is denoted as $\omega_{\mathcal{B}/\mathcal{I}}$ and when expressed in body-frame coordinates it is written as $\omega^{\mathcal{B}}_{\mathcal{B}/\mathcal{I}} = [p, q, r]^T$.

4.2.1 Position Transformation

The MAVROS topic `/mavros/local_position/pose` provides position information of the vehicle with respect to the local inertial frame in terms of ENU frame coordinates. To convert to NED coordinates:

$$P^{\mathcal{I},\text{NED}}_{b/i} = R_{\mathcal{I},\text{NED}/\mathcal{I},\text{ENU}} P^{\mathcal{I},\text{ENU}}_{b/i}, \quad R_{\mathcal{I},\text{NED}/\mathcal{I},\text{ENU}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

The equivalent quarternion that rotates from ENU to NED is:

$$q_{\mathcal{I},\text{NED}/\mathcal{I},\text{ENU}} = \begin{bmatrix} w & x & y & z \end{bmatrix} = \begin{bmatrix} 0 & \sin\pi/4 & \cos\pi/4 & 0 \end{bmatrix},$$

and the quarternion that rotates from NED back to ENU is:

$$q_{\mathcal{I},\text{ENU}/\mathcal{I},\text{NED}} = \begin{bmatrix} w & x & y & z \end{bmatrix} = \begin{bmatrix} 0 & -\sin\pi/4 & -\cos\pi/4 & 0 \end{bmatrix}.$$

4.2.2 Attitude Transformation

The MAVROS topic `/mavros/local_position/pose` provides attitude information in the form of a quaternion that defines the rotation from the local ENU frame to the body FLU frame. However we want the attitude rotation from NED to FRD. This rotation can be found through the composition of several rotations:

1. Rotate local inertial NED frame to local inertial ENU frame using rotation described above.
2. Rotate local ENU frame to body FLU frame using the quaternion rotation given by MAVROS.
3. Rotate the FLU frame to FRD frame, which is just a rotation of 180 degrees around the x axis given by the quaternion:

$$q_{\text{FRD}/\text{FLU}} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}.$$

4.2.3 Velocity Transformation

The MAVROS topic `/mavros/local_position/velocity_body` gives linear velocity of body with respect to body FLU frame. This can be converted into body frame (FRD) coordinates by taking the negative of y coordinate and z coordinate.

4.2.4 Angular Velocity Transformation

The MAVROS topic `/mavros/local_position/velocity_body` gives angular velocity of body with respect to body ENU (FLU) frame. This can be converted into body frame (FRD) coordinates by taking the negative of y coordinate and z coordinate.

4.3 Aircraft Nonlinear Rigid-Body Dynamics

This section defines one commonly used aircraft dynamics model, but of course there are others!

4.3.1 Reference Frames

This dynamics model is built using the reference frame convention mentioned in Section 4.2. The body-fixed frame $\mathcal{B}$ and inertial frame $\mathcal{I}$ relative orientation is related using Euler angles (ZYX) (Yaw-Pitch-Roll) (ψ, θ, ϕ). This process is done through the set of rotations: inertial coordinates $\rightarrow \psi \rightarrow$ Intermediate1 $\rightarrow \theta \rightarrow$ Intermediate2 $\rightarrow \phi \rightarrow$ body coordinates. In other words: $(\cdot)^{\mathcal{B}} = R_{\mathcal{B}/\mathcal{I}}(\cdot)^{\mathcal{I}}$, and $R_{\mathcal{B}/\mathcal{I}} = R_\phi R_\theta R_\psi$:

$$\begin{aligned} R_{\mathcal{B}/\mathcal{I}}(\phi,\theta,\psi) &= \begin{bmatrix} \cos\psi\cos\theta & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & \cos\theta\sin\phi \\ \sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix}, \\ R_{\mathcal{I}/\mathcal{B}} = R_{\mathcal{B}/\mathcal{I}}^\top &= \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta \\ \cos\theta\sin\psi & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \end{aligned} \tag{3}$$

The angular velocity and Euler angle rates can be related by:

$$\omega_{\mathcal{B}/\mathcal{I}}^{\mathcal{B}} = \begin{bmatrix} \dot\phi \\ 0 \\ 0 \end{bmatrix} + R_\phi \begin{bmatrix} 0 \\ \dot\theta \\ 0 \end{bmatrix} R_\phi R_\theta \begin{bmatrix} 0 \\ 0 \\ \dot\psi \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{bmatrix},$$

or by taking the inverse

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\theta\\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}. \tag{4}$$

4.3.2 Dynamics

From Newton's laws, assuming an inertial flat Earth, the differential equation representing the change in inertial velocity in body fixed coordinates is:

$${}^{\mathcal{B}}\dot{V}^{\mathcal{B}}_{b/\mathcal{I}} = \frac{F^{\mathcal{B}}}{m} + g\begin{bmatrix}-\sin\theta\\ \cos\theta\sin\phi\\ \cos\theta\cos\phi\end{bmatrix} - \omega^{\mathcal{B}}_{\mathcal{B}/\mathcal{I}} \times V^{\mathcal{B}}_{b/\mathcal{I}}, \tag{5}$$

Note that the derivative ${}^{\mathcal{B}}\dot{V}^{\mathcal{B}}_{b/\mathcal{I}}$ is the derivative with respect to the body fixed frame and the gravity g has been expressed in body frame coordinates as well. The forces $F^{\mathcal{B}}$ are aerodynamic and thrust forces acting on the aircraft, expressed in the body-fixed frame $\mathcal{B}$. The Euler equations of rotational motion can be written as:

$${}^{\mathcal{B}}\dot{\omega}^{\mathcal{B}}_{\mathcal{B}/\mathcal{I}} = I^{-1}(M^{\mathcal{B}} - \omega^{\mathcal{B}}_{\mathcal{B}/\mathcal{I}} \times J\omega^{\mathcal{B}}_{\mathcal{B}/\mathcal{I}}), \tag{6}$$

where I is the inertia matrix with respect to the body frame $\mathcal{B}$ and $M^{\mathcal{B}}$ is the external moment vector about the center of mass expressed in body-frame $\mathcal{B}$ arising from aerodynamics. When combined the nonlinear equations of motion become:

$$\begin{aligned}
\dot{u} &= \frac{T + F_x}{m} - g\sin\theta + rv - qw,\\
\dot{v} &= \frac{F_y}{m} + g\cos\theta\sin\phi - ru + pw,\\
\dot{w} &= \frac{F_z}{m} + g\cos\theta\cos\phi + qu - pv,\\
\dot{p} &= \frac{M_x + (I_y - I_z)qr}{I_x},\\
\dot{q} &= \frac{M_y + (I_z - I_x)pr}{I_y},\\
\dot{r} &= \frac{M_z + (I_x - I_y)pq}{I_z},\\
\dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta,\\
\dot{\theta} &= q\cos\phi - r\sin\theta,\\
\dot{\psi} &= q\sin\phi\sec\theta + r\cos\phi\sec\theta,\\
\begin{bmatrix}\dot{x}_i\\ \dot{y}_i\\ \dot{z}_i\end{bmatrix} &= R_{\mathcal{I}/\mathcal{B}}(\phi,\theta,\psi)\begin{bmatrix}u\\ v\\ w\end{bmatrix},
\end{aligned} \tag{7}$$

where we have assumed that the body-fixed frame $\mathcal{B}$ coincides with the principle axis such that the products of inertia are all zero.

4.3.3 Flight Equilibrium Conditions

Two equilibria for the aircraft include steady glideslope flight and steady turning flight.

Steady Glideslope Flight: For steady glideslope flight the aircraft descends at a constant velocity while maintaining wings level and a constant speed. The glideslope is defined not just by the constant speed S_0 but by the glideslope angle γ_0, which is defined to be negative when the velocity vector points below the local horizon. The equilibrium constraints are:

1. $\dot{u} = \dot{v} = \dot{w} = \dot{p} = \dot{q} = \dot{r} = \dot{\phi} = \dot{\theta} = \dot{\psi} = p_0 = q_0 = r_0 = 0$.
2. There is no side motion, $v_0 = 0$.
3. Roll angle $\phi_0 = 0$.
4. The total speed is constant $S_0 = \sqrt{u_0^2 + w_0^2}$.
5. Glideslope angle γ_0 constraint is $\dot{z}_i = -S_0 \sin\gamma_0$, which simplifies to $-S_0 \sin\gamma_0 = -u_0 \sin\theta_0 + w_0 \cos\theta_0$.

The final equilibrium state is:

$$x_0 = [u_0,\, 0,\, w_0,\, 0,\, 0,\, 0,\, 0,\, \theta_0,\, \psi_0,\, x_i(t),\, y_i(t),\, z_i(t)]^\top,$$

where the positions change in time according to:

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{z}_i \end{bmatrix} = R_{\mathcal{I}/\mathcal{B}}(0, \theta_0, \psi_0) \begin{bmatrix} u_0 \\ 0 \\ w_0 \end{bmatrix}$$

The yaw and position equations can now be replaced by *relative* yaw and position equations. To accomplish this switch a new inertial reference frame $\mathcal{R}$ is defined which will move in the inertial frame $\mathcal{I}$ as the aircraft would if it was perfectly following the equilibrium trajectory. The frame $\mathcal{R}$ moves with velocity $V^{\mathcal{R}}_{r/\mathcal{I}} = [u_0, 0, w_0]^\top$, $\omega^{\mathcal{R}}_{\mathcal{R}/\mathcal{I}} = [0, 0, 0]^\top$, and the roll, pitch, and yaw ϕ_0, θ_0, ψ_0 are all constant (with $\phi_0 = 0$). The relative yaw $\delta\psi = \psi - \psi_0$ now has the following dynamics:

$$\delta\dot{\psi} = q \sin\phi \sec\theta + r \cos\phi \sec\theta. \tag{8}$$

For the position error, let the position of the origin of frame $\mathcal{R}$ with respect to the inertial frame be denoted by $P_{r/i}$. The relative position vector is then defined as $P_{b/r} = P_{b/i} - P_{r/i}$ (i.e. position of body with respect to origin of frame $\mathcal{R}$). Taking the derivative with respect to the inertial frame gives:

$$\begin{aligned} {}^{\mathcal{I}}\dot{P}_{b/r} &= {}^{\mathcal{I}}\dot{P}_{b/i} - {}^{\mathcal{I}}\dot{P}_{r/i}, \\ V_{b/\mathcal{R}} + \omega_{\mathcal{R}/\mathcal{I}} \times P_{b/r} &= V_{b/\mathcal{I}} - V_{r/\mathcal{I}}. \end{aligned}$$

Since it is natural to express the relative position vector in terms of the $\mathcal{R}$ frame coordinates (let $P^{\mathcal{R}}_{b/r} = [x_r, y_r, z_r]^\top$) some rotations must be used to leverage the other state variables which are expressed in body coordinates:

$$\begin{aligned} V^{\mathcal{R}}_{b/\mathcal{R}} &= V^{\mathcal{R}}_{b/\mathcal{I}} - V^{\mathcal{R}}_{r/\mathcal{I}} - \omega^{\mathcal{R}}_{\mathcal{R}/\mathcal{I}} \times P^{\mathcal{R}}_{b/r}, \\ \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{z}_r \end{bmatrix} &= R_{\mathcal{R}/\mathcal{B}} \begin{bmatrix} u \\ v \\ w \end{bmatrix} - \begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix} - \begin{bmatrix} p_0 \\ q_0 \\ r_0 \end{bmatrix} \times \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}, \end{aligned}$$

where $R_{\mathcal{R}/\mathcal{B}} = R_{\mathcal{R}/\mathcal{I}} R_{\mathcal{I}/\mathcal{B}}$ which can be written as:

$$R_{\mathcal{R}/\mathcal{B}}(\phi, \theta, \delta\psi) = \begin{bmatrix} \cos\theta_0 & 0 & -\sin\theta_0 \\ 0 & 1 & 0 \\ \sin\theta_0 & 0 & \cos\theta_0 \end{bmatrix} R_{\mathcal{I}/\mathcal{B}}(\phi, \theta, \delta\psi).$$

Since $p_0 = q_0 = r_0 = v_0 = 0$ the relative position equations of motion are therefore:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{z}_r \end{bmatrix} = R_{\mathcal{R}/\mathcal{B}}(\phi, \theta, \delta\psi) \begin{bmatrix} u \\ v \\ w \end{bmatrix} - \begin{bmatrix} u_0 \\ 0 \\ w_0 \end{bmatrix}. \tag{9}$$

The full state of the new dynamics is then $x = [u,\, v,\, w,\, p,\, q,\, r,\, \phi,\, \theta,\, \delta\psi,\, x_r,\, y_r,\, z_r]^\top$ and the steady glideslope flight equilibrium condition is:

$$x_0 = [u_0,\, 0,\, w_0,\, 0,\, 0,\, 0,\, 0,\, \theta_0,\, 0,\, 0,\, 0,\, 0]^\top.$$

Steady Level Flight: The classic steady level flight is the same as the steady glideslope flight with $\gamma = 0$.

Steady Turning Flight: The steady turning flight constraints are:

1. $\dot{u} = \dot{v} = \dot{w} = \dot{p} = \dot{q} = \dot{r} = \dot{\phi} = \dot{\theta} = 0$.
2. Constant yaw rate $\dot{\psi}$
3. Total speed is fixed to a constant $S_0 = \dot{\psi} R = \sqrt{u_0^2 + v_0^2 + w_0^2}$ where R is the radius of the turn.
4. Constant altitude $\dot{z}_i = 0$, which can be expressed as the constraint:

$$0 = -u_0 \sin\theta_0 + v_0 \cos\theta_0 \sin\phi_0 + w_0 \cos\theta_0 \cos\phi_0.$$

In this case every value reaches a constant equilibrium except for the yaw angle ψ and the positions x_i and y_i. However by replacing the yaw angle and these positions with *relative* yaw and positions with respect to a nominal trajectory a *completely constant* equilibrium state is achieved.

First, the yaw error $\delta\psi = \psi - \psi_0$ has relatively trivial dynamics:

$$\delta\dot{\psi} = q \sin\phi \sec\theta + r \cos\phi \sec\theta - \dot{\psi}_0.$$

However the challenge here is that the rotation matrix $R_{\mathcal{R}/\mathcal{I}}$ is dependent on $\psi_0(t)$, which is changing and is not a state of the system. Fortunately this can be written in terms of the error $\delta\psi$ since $R_{\mathcal{R}/\mathcal{B}} = R_{\phi_0} R_{\theta_0} R_{\delta\psi}^\top R_\theta^\top R_\phi^\top$:

$$R_{\mathcal{R}/\mathcal{B}}(\phi, \theta, \delta\psi) = \begin{bmatrix} \cos\theta_0 & 0 & -\sin\theta_0 \\ \sin\theta_0 \sin\phi_0 & \cos\phi_0 & \cos\theta_0 \sin\phi_0 \\ \sin\theta_0 \cos\phi_0 & -\sin\phi_0 & \cos\theta_0 \cos\phi_0 \end{bmatrix} R_{\mathcal{I}/\mathcal{B}}(\phi, \theta, \delta\psi).$$

To summarize, the aircraft dynamics equations are given by (7) with the $\dot{\psi}$ and navigation equations replaced by:

$$\begin{aligned} \delta\dot{\psi} &= q \sin\phi \sec\theta + r \cos\phi \sec\theta - \dot{\psi}_0, \\ \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{z}_r \end{bmatrix} &= R_{\mathcal{R}/\mathcal{B}}(\phi, \theta, \delta\psi) \begin{bmatrix} u \\ v \\ w \end{bmatrix} - \begin{bmatrix} u_0 + q_0 z_r - r_0 y_r \\ v_0 + r_0 x_r - p_0 z_r \\ w_0 + p_0 y_r - q_0 x_r \end{bmatrix}. \end{aligned} \tag{10}$$

The full state is then $x = [u, v, w, p, q, r, \phi, \theta, \delta\psi, x_r, y_r, z_r]^\top$ and the steady turning flight equilibrium condition is:

$$x_0 = [u_0, v_0, w_0, p_0, q_0, r_0, \phi_0, \theta_0, 0, 0, 0, 0]^\top.$$

5 User Guide

5.1 Simulation

To run a simulation with PX4, ROS, and Gazebo, make sure that the PX4 SITL Gazebo target has been built. If it hasn't or needs to be rebuilt, you can do so with:

```
cd ~/Firmware
DONT_RUN=1 make px4_sitl_default gazebo
```

Then, to run a simulation (Gazebo will automatically launch) use ROS to launch the desired code. For example:

```
roslaunch asl_fixedwing mavros_posix_sitl.launch
```

To takeoff, command:

```
commander takeoff
```

in the terminal where the simulation was launched. The mode can be changed to "offboard" by issuing the command:

```
commander mode offboard
```

5.1.1 Examples

`position.cpp` Run this example by:

```
roslaunch asl_fixedwing mavros_posix_sitl.launch
rosrun asl_fixedwing position
```

5.2 Hardware

Once all of the hardware is set up properly, make sure the battery is sufficiently charged. Connect the battery to the PMB and all of the electronics should boot up automatically. The ground station laptop running QGroundControl (with telemetry radio attached via USB) should automatically connect to the Pixhawk.

As a first checkout, you should always press the safety switch until it is in pre-armed mode, and verify that the control surfaces respond appropriately to commands (prop motor will not run in pre-armed mode). At this time you should be able to log in to the Odroid computer from the ground station laptop by connecting to the `odroid` wifi network (password `odroidwifi`). Then SSH by:

```
ssh odroid@192.168.100.1
```

with password `odroid`. You should then be able to launch the MAVROS node by:

```
screen -S mavros
roslaunch asl_fixedwing pixhawk_mavros.launch
```

You can test to see that you are reading all of the MAVROS topics by running the `rostopic list` command. Any additional offboard ROS nodes should then be run:

```
screen -S node
roslaunch asl_fixedwing <node-name>
```

When running ANY ROS node on the odroid you should use `screen`!

Once in flight, you can use the back left switch on the transmitter to switch from `Manual` to `Offboard` mode.

6 Troubleshooting

1. Sometimes, the Pixhawk 4 doesn't connect to QGroundControl. This could be fixed by just changing the micro-USB used to power the Pixhawk 4, and upgrading the firmware via the firmware page in QGroundControl.

2. If the Spektrum receiver will not enter binding mode when "Spektrum Bind" is pressed in QGroundControl, check the parameter CBRK_IO_SAFETY. If it is set to anything other than 0, set it to 0, and try again. This issue will be indicated by a warning message in QGroundControl, "IO safety off, bind request rejected".

3. The telemetry radios sometimes fail to connect with each other; this could be due to the radios running different versions of the firmware. To fix this issue, first connect them to a computer, using a micro-USB cable, and open up Mission Planner: `http://ardupilot.org/planner/index.html`. In Mission Planner, go to Initial Setup > Sik Radio, and click Load Settings. This should display system information for the telemetry radios, including what firmware version they are running. If the radios are running different versions of the firmware, update both of them to the same version using the "Upload Firmware (Local)" button.

7 Gazebo FixedWing Model

The Gazebo model that is included in the repository is essentially the same as the plane model that comes standard with PX4, but with a different visualization just for fun. This SDF model defines a puller aircraft with a rudder, elevator, and one elevon on each wing. The aerodynamics are calculated using the LiftDrag plugin in Gazebo[23], and are defined by some simple aerodynamic coefficients.

7.1 Gazebo Aerodynamics Model

The body-frame quantities $F_{x,y,z}$ and $M_{x,y,z}$ in (7) consist of the aerodynamic forces acting on the aircraft. In Gazebo the aerodynamics are modeled using aerodynamic coefficients and are separated into the wings, rudder, and elevator. Thus:

$$\begin{aligned} F_a^{\mathcal{B}} &= F_{lw}^{\mathcal{B}} + F_{rw}^{\mathcal{B}} + F_e^{\mathcal{B}} + F_r^{\mathcal{B}}, \\ M_a^{\mathcal{B}} &= c_{p,\text{left}} \times F_{lw}^{\mathcal{B}} + c_{p,\text{right}} \times F_{rw}^{\mathcal{B}} + c_{p,\text{elev}} \times F_e^{\mathcal{B}} + c_{p,\text{rud}} \times F_r^{\mathcal{B}}, \end{aligned}$$

where the moments due to the aerodynamics are ignored (in fact they are not implemented in Gazebo) and therefore the only moments arise due to the forces acting on each surface.

A useful set of quantities when discussing the aerodynamic forces are the angle of attack α and sideslip angle β. The angle of attack is the angle between the body frame $\mathcal{B}$ x-axis and the projection of the *relative wind* onto the body frame x-z plane, and it is positive when the relative wind is on the underside of the aircraft. The sideslip angle is the angle between the relative wind and its projection onto the body frame x-z plane, and is positive when relative wind is on the right hand side of the plane. The lift and drag are defined in terms of a stability frame S that is rotated about the body y-axis by the angle of attack (i.e. aligns the frame with the relative wind projected into body x-z plane):

$$R_{S_i/\mathcal{B}} = \begin{bmatrix} \cos\alpha_i & 0 & \sin\alpha_i \\ 0 & 1 & 0 \\ -\sin\alpha_i & 0 & \cos\alpha_i \end{bmatrix}, \quad R_{\mathcal{B}/S_i} = R_{S_i/\mathcal{B}}^{\top}.$$

The subscript i denotes a particular location on the aircraft, because in general the inertial velocity $V_i^{\mathcal{B}}$ will be different at each of the different locations on the aircraft (can be written though as the sum of the center of mass velocity and rotation rates). Since the velocity vector $V_i^{\mathcal{B}}$ maps to the vector $V_i^S = (\|V_i^{\mathcal{B}}\|, 0, 0)$ this relationship can be used to compute:

$$\tan\alpha_i = \frac{w_i}{u_i}.$$

This isn't exactly what is done in Gazebo, but a slightly simpler approach is to compute the aerodynamics as:

$$F^{S_i} = \begin{bmatrix} -D_i \\ 0 \\ -L_i \end{bmatrix}, \quad D_i = |C_{D,i}| q_i S_i, \quad L_i = C_{L,i} q_i S_i,$$

where:

$$\begin{aligned} C_{D,i} &= C_{D_\alpha,i}(\alpha_i - \alpha_{0,i}), \\ C_{L,i} &= C_{L_\alpha,i}(\alpha_i - \alpha_{0,i}) + C_{L_\delta,i}\delta_i, \\ q_i &= \frac{1}{2}\rho(u_i^2 + w_i^2), \end{aligned}$$

where δ is the angle of the control surface deflection. This is simplified because we are essentially ignoring the spanwise component of the velocity.

Note that the rudder is of a different orientation than the elevator and wings. For the rudder the "upwards" direction is pointing to the left side of the aircraft. Thus in this case $\tan\alpha_r = v_r/u_r$ and the stability axis S is oriented differently than before:

$$R_{S_{\text{rudder}}/\mathcal{B}} = \begin{bmatrix} \cos\alpha_r & 0 & \sin\alpha_r \\ 0 & 1 & 0 \\ -\sin\alpha_r & 0 & \cos\alpha_r \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \cos\alpha_r & \sin\alpha_r & 0 \\ 0 & 0 & -1 \\ -\sin\alpha_r & \cos\alpha_r & 0 \end{bmatrix}.$$

[23] https://github.com/osrf/gazebo/blob/gazebo11/plugins/LiftDragPlugin.cc

Similarly $q_r = \frac{1}{2}\rho(u_r^2 + v_r^2)$.

Note that while the ailerons are modeled independently they are controlled through a single command δ_a, as this command is mapped anti-symmetrically to the two ailerons. A positive δ_a induces a right bank.

7.1.1 Computing Inertial Velocities at Different Locations

For now we will assume that $V_i^{\mathcal{B}} = V_{b/\mathcal{I}}^{\mathcal{B}}$, therefore ignoring the components due to rotations.

7.1.2 Orientation of Control Surfaces

δ_{la}: left aileron, a positive deflection angle induces a left bank.
δ_{ra}: right aileron, a positive deflection angle induces a right bank.
δ_e: elevator, a positive deflection angle induces a pitch up moment.
δ_r: rudder, a positive deflection angle induces a yaw to the right.

7.2 Gazebo Propeller Model

The propeller model that is used in the Gazebo simulator is in a simplified form:

$$T = \min\{1, \max\{0, 1 - \frac{v_0}{v_{\max}}\}\} k_{\text{motor}} (k_{\text{slowdown}}\omega)^2$$

where k_{slowdown} and k_{motor} are parameters defined in the model sdf file, v_0 is the relative wind speed in the direction parallel to the motor axis, and $v_{\max}$ is another parameter. This additional scalar out front is used to crudely model the effect of increased airspeed on decreasing the propeller's thrust. Another few notes: a negative thrust is not allowed. There are also other components to the motor model that are ignored here, such as computing the torques induced by the motor. Since this model does not exactly follow the structure of the propeller model discussed in Section 4.1.1 some guessing of the appropriate parameters will need to be done.

7.3 Gazebo Model Parameters

7.3.1 Mass Properties

The mass properties of the aircraft are:

$m = 1.5\,[\text{kg}]$
$I_x = 0.197563\,[\text{kg m}^2]$
$I_y = 0.1458929\,[\text{kg m}^2]$
$I_z = 0.1477\,[\text{kg m}^2]$

and it is assumed that gravity is $g = 9.8066\,[\text{m/s}^2]$ and the air density is $\rho = 1.2041\,[\text{kg/m}^3]$.

7.3.2 Wing Aerodynamics

Each wing is modeled using lift/drag vs. α curves, and the relevant parameters for both wings are:

$\alpha_0 = -0.05984281113\,[\text{rad}]$, the angle of attack at which there is no lift
$C_{L_\alpha} = 4.752798721$, the lift curve slope
$C_{D_\alpha} = 0.6417112299$, the drag curve slope
$C_{M_\alpha} = -1.8$, the moment curve slope
$C_{L_\delta} = -0.5$, change in lift coefficient per change in control surface deflection angle
$S = 0.12\,[\text{m}^2]$, the effective wing surface area
$c_{p,\text{left}} = (-0.05, -0.3, -0.05)\,[\text{m}]$, position of the center of pressure (where aerodynamic forces act) relative to the wing's center of mass. These coordinates seem to be in the coordinates of the aircraft body frame $\mathcal{B}$ (note the SDF file uses a Forward-Left-Up conventions while $\mathcal{B}$ is FRD.

$c_{p,\text{right}} = (-0.05, 0.3, -0.05)$ [m]
$|\delta_w| \leq 0.53$, max control surface deflection

7.3.3 Elevator Aerodynamics

$\alpha_0 = 0.2$ [rad]
$C_{L_\alpha} = 4.752798721$
$C_{D_\alpha} = 0.6417112299$
$C_{M_\alpha} = -1.8$
$C_{L_\delta} = -4.0$
$S = 0.01$ [m^2]
$c_p = (-0.5, 0, 0)$ [m]
$|\delta_e| \leq 0.53$

7.3.4 Rudder Aerodynamics

$\alpha_0 = 0$ [rad]
$C_{L_\alpha} = 4.752798721$
$C_{D_\alpha} = 0.6417112299$
$C_{M_\alpha} = -1.8$
$C_{L_\delta} = 0$
$S = 0.02$ [m^2]
$c_p = (-0.5, 0, -0.05)$ [m]
$|\delta_r| \leq 0.53$

7.3.5 Propeller

$k_{\text{motor}} = 8.54858\text{e-}06$
$k_{\text{slowdown}} = 10$
$v_{\text{max}} = 25$ [m/s]

8 Skywalker Mass Properties

8.1 Mass

The mass of the aircraft was measured by scale to be $m = 2470$ [g].

8.2 Center of Mass

The center of mass is assumed to lie on the plane of symmetry. The location within the plane of symmetry was approximated by hanging the plane in the cradle with nose up to get the vertical position (see Figure 11) and by placing on two scales to get the horizontal position. The center of mass position is shown in Figure 9.

8.2.1 Roll Axis

To determine the point along the body x direction the aircraft is weighed using two scales at different points length L apart to get the weights m_1 and m_2. Then the distance from point 1 to the center of gravity is $L_1 = \frac{m_2}{m_1+m_2}L$. Using this methodology the distance from the nose to the c.m. is 36.5 cm, from the carbon fiber tube/fuselage connection to the c.m. is 22.5 cm, and from the leading edge of the wing to the c.m. is 7.5 cm.

8.2.2 Yaw Axis

Determining the point along the body z direction we use a more complicated procedure. When holding the aircraft in the cradle with roll axis pointing upward the combined c.m. was a distance $L_{ac} = 16.5$ cm from the top of the cradle, and with just the cradle the distance from top of cradle to cradle c.m. is $L_c = 10$ cm. The difference is therefore $d = L_{ac} - L_c$.

Now considering that the center of mass of the assembly is in between the center of masses of the cradle and aircraft (and only focusing on the components perpendicular to gravity, the z direction in this case), we can compute the distance between the center of the mass of the cradle and the center of mass of the aircraft in the body z direction as $L = \frac{m_a+m_c}{m_a}d$, which in our case gives $L = 9.25$ cm. Therefore the distance from the top of the cradle to the aircraft c.m. is $L_a = L_c + L = 19.25$ cm. From the bottom of the cradle/the bottom of the aircraft the distances is approximately 10 cm.

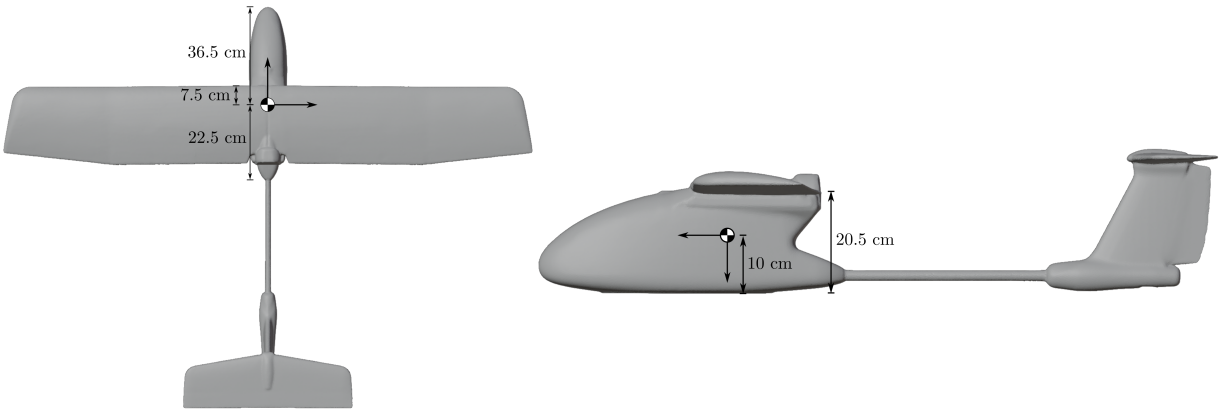

Figure 9: Skywalker center of mass position with all electronics attached.

8.3 Moment of Inertia

8.3.1 Pitch Axis

The pitch axis moment of inertia I_y is determined experimentally using a compound pendulum swing. The equations of motion for the compound pendulum are:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = Q, \quad L = \frac{1}{2} I_p \dot{\theta}^2 - m_s g l(1-\cos\theta), \quad Q = -K_D \dot{\theta}|\dot{\theta}| - C\dot{\theta},$$

where I_p [kg m^2] is the moment of inertia about the pivot axis of the swing, θ [rad] is the angular displacement about the pivot axis from the rest position, m_s [kg] is the mass of the entire (aircraft + cradle) system, l [m] is the distance from the pivot axis to the center of mass of the entire (aircraft + cradle) system. This results in the dynamics:

$$\ddot{\theta} + \frac{K_D}{I_p}\dot{\theta}|\dot{\theta}| - \frac{C}{I_p}\dot{\theta} + \frac{m_s g l}{I_p}\sin\theta = 0.$$

As a most simple estimate the dynamics can be linearized to give:

$$\ddot{\theta} + \frac{C}{I_p}\dot{\theta} + \frac{m_s g l}{I_p}\theta = 0.$$

This is a second order linear system, and with an initial value of $\dot{\theta}(0) = 0$ and $\theta(0) = \theta_0$ the Laplace transform gives:

$$\theta(s) = \frac{s+1}{s^2 + \frac{C}{I_p}s + \frac{m_s g l}{I_p}}\theta_0$$

which can be written as:

$$\theta(s) = \frac{(s+1)}{(s+\sigma)^2 + \omega_d^2}\theta_0, \quad \sigma = \zeta\omega_n, \quad \omega_d = \omega_n\sqrt{1-\zeta^2}, \quad \zeta = \frac{C}{2\sqrt{I_p m_s g l}}, \quad \omega_n = \sqrt{\frac{m_s g l}{I_p}}.$$

After the algebraic manipulation this transfer function yields a time domain solution of the form:

$$\theta(t) = M e^{-\sigma t}\cos(\omega_d t + \phi),$$

for some constant M and delay ϕ. The damped frequency ω_d can be computed from the experimental data. Notice also that σ can be determined as:

$$\sigma = \frac{-\ln\frac{\theta_2}{\theta_1}}{T_d}$$

where θ_2 and θ_1 are the values at two successive peaks and T_d is the period of oscillation (time between peaks). This can then be algebraically rearranged to compute:

$$\zeta = \frac{|\ln\frac{\theta_2}{\theta_1}|}{\sqrt{4\pi^2 + (\ln\frac{\theta_2}{\theta_1})^2}}.$$

The natural frequency ω_n can then be determined and thus the moment of inertia I_p.

A similar experiment is then repeated for just the cradle, to produce the moment of inertia of the cradle about the pivot point, $I_{p,\text{cradle}}$ [kg m^2]. The moment of inertia of the aircraft about the pivot is then:

$$I_{p,\text{aircraft}} = I_p - I_{p,\text{cradle}}$$

Once the moment of inertia $I_{p,\text{aircraft}}$ is determined, the moment of inertia I_y of the aircraft about the aircraft's center of mass can be computed by the parallel axis theorem:

$$I_y = I_{p,\text{aircraft}} - m l_{cm}^2,$$

where m is the mass of the aircraft and l_{cm} is the distance from the pivot axis to the center of mass of the aircraft. The expezrimental setup for this test is shown in Figure 10.

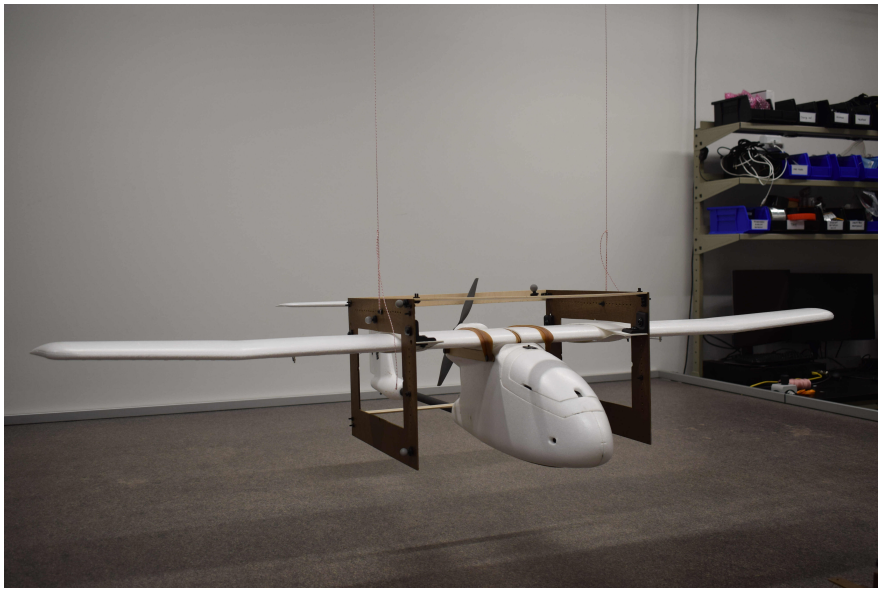

Figure 10: Skywalker in cradle configured for measuring the pitch moment of inertia.

8.3.2 Roll and Yaw Axes

The roll and yaw axes moment of inertia I_x and I_z are computed experimentally using a bifilar pendulum swing (Jardin, Mueller 2007). The orientation of the swinging is shown in Figure 11 for the roll axis, and the yaw axis test configuration is shown in Figure 10. However instead of performing a full nonlinear parameter estimation, a linearized approach is used for simplicity. The equations of motion of the bifilar pendulum are:

$$\ddot{\theta} + \frac{K_D}{I_p}\dot{\theta}|\dot{\theta}| + \frac{C}{I_p}\dot{\theta} + \frac{mgD^2}{4I_p h}\frac{\sin\theta}{\sqrt{1-\frac{1}{2}(\frac{D}{h})^2(1-\cos\theta)}} = 0,$$

where K_D and C are damping coefficients, I_p [kg m^2] is the moment of inertia about the pivot axis, m [kg] is the mass of the system, D [m] is the distance between the filars, h [m] is the length of the filars, and θ [rad] is the rotation angle. Linearizing these dynamics gives:

$$\ddot{\theta} + \frac{C}{I_p}\dot{\theta} + \frac{mgD^2}{4I_p h}\theta = 0,$$

which again yields a damped second order linear system and the moment of inertia I_p can be computed in a similar way to in the pitch case.

$$\ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta = 0, \quad \omega_n = \sqrt{\frac{mgD^2}{4I_p h}}. \tag{11}$$

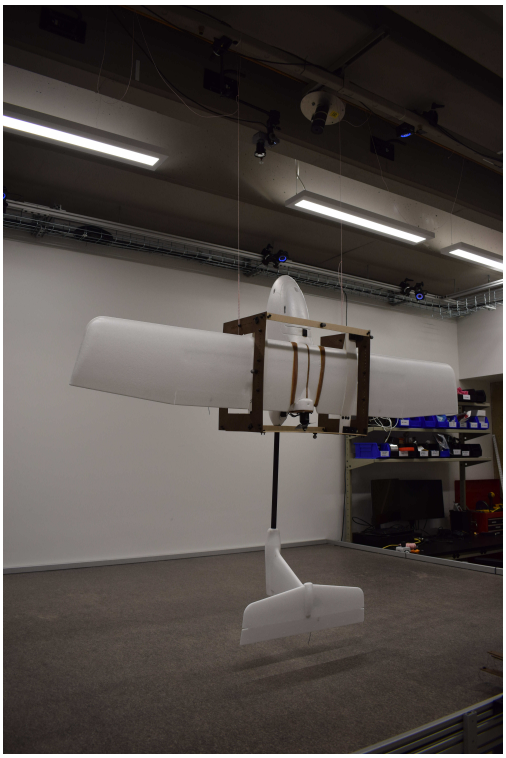

Figure 11: Skywalker in cradle configured for measuring the roll moment of inertia.

9 Skywalker Thrust Model

We are using a APC 10x6E propeller which has data available online[24]. As noted in Section 4.1 we are currently using a simple thrust model:

$$T = c_{T,0}(1 - c_{T,V\omega}\frac{V}{u_T})u_T^2,$$

where V is the airspeed in m/s in the direction of the propeller axis and u_T is the normalized commanded value. The coefficients $c_{T,0}$ and $c_{T,V\omega}$ are determined using a combination of static thrust stand testing and by leveraging the propeller data from the vendor. The static thrust properties are experimentally determined using the test rig shown in Figure 12.

The propeller data is correlated to the static thrust stand tests to determine the (assumed) linear mapping between command and motor rotation speed. A parameter estimation is then performed to determine both constants $c_{T,0}$ and $c_{T,V\omega}$ based on the propeller data. Figure 13 shows the propeller data from the vendor site (red) and the parameter fit model (blue). This thrust map then results in Figures 14 and 15 at $V = 0$ and $V = 15$ m/s. Finally, Figure 16 shows the J vs C_T plot for the APC10x6E propeller, based on the vendor propeller data.

[24] https://www.apcprop.com/technical-information/performance-data/

Figure 12: Static thrust stand testing for the Skywalker APC 10x6E Prop. A simple scale is used to measure the thrust.

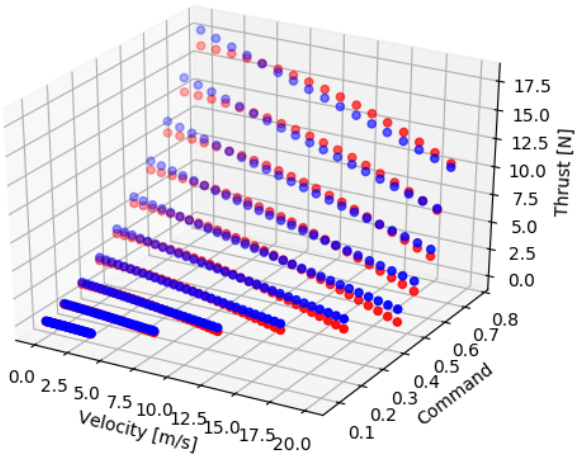

Figure 13: Thrust mapping based on velocity and normalized commanded value. Propeller data from vendor is shown in red and parameter fit model is shown in blue.

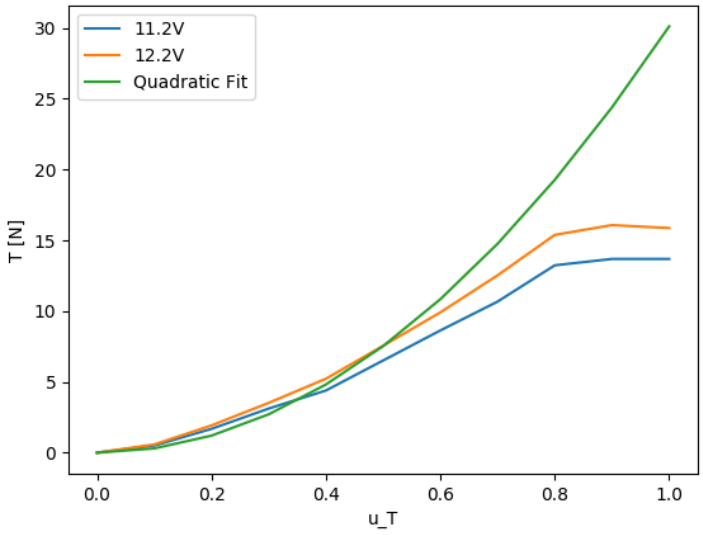

Figure 14: Total thrust curve for the Skywalker APC 10x6E Prop at an assumed airspeed of 0 m/s.

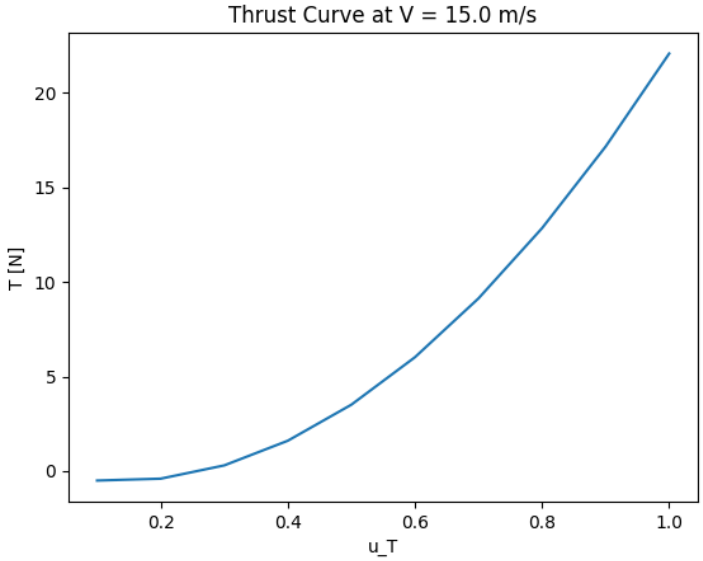

Figure 15: Total thrust curve for the Skywalker APC 10x6E Prop at an assumed airspeed of 15 m/s.

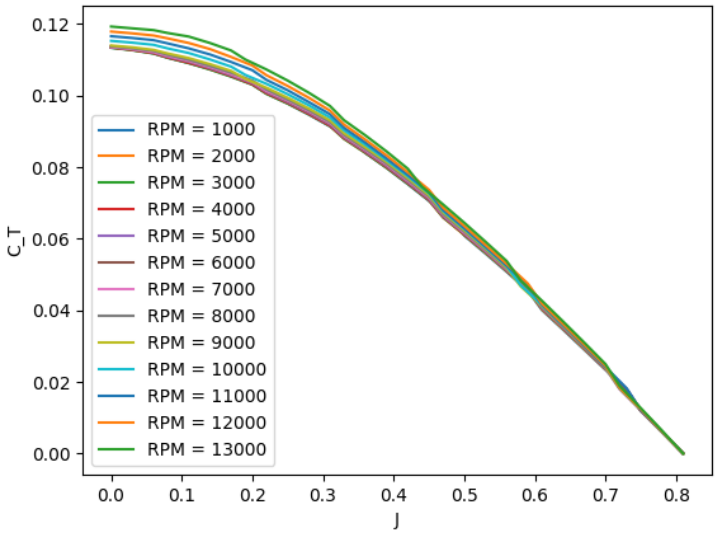

Figure 16: Plot of advance ratio J vs coefficient of thrust C_T for the Skywalker APC 10x6E Prop based on vendor data.