ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Semester Thesis

Sensing and characterization of the motor dynamics on an omnidirectional flying vehicle

Spring Term 2020

Supervised by:
Weixuan Zhang
Maximilian Brunner

Author:
Fausto Tapia

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Sensing and characterization of the motor dynamics on an omnidirectional flying vehicle

Authored by (in block letters):

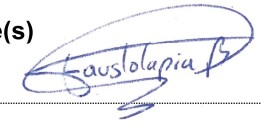
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Tapia Benavides	Fausto

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
27.07.2020	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

Abstract

Many control techniques used in multi-rotor flying vehicles only use feed-forward commands to spin the propellers. This includes an omnidirectional micro aerial vehicle (OMAV) with coaxial propellers introduced in this report. For highly complex systems, such as the vehicle introduced in this report, unaccounted perturbations can cause instabilities and significant errors. Therefore, electronic speed controller (ESC) behavior was analyzed as well as the capabilities to perform feedback control for rotor speed in the context of coaxial rotors. It is found that the throttle to rotor speed commanded is not linear and is inconsistent between items of the same model due to motor and electronic mismatches. A model of voltage and throttle dependent rotor speed is suggested for feed-forward commands. Response speed, efficiency and interactions in the coaxial propeller setup are also analyzed. Concluding, Bidirectional Dshot and UAVCAN protocols are suggested for better tracking of rotor speed for several-rotor vehicles, as well as meticulous calibration if only feed-forward rotor speed control methods are used.

Symbols

Symbols

t	Throttle command
V	Voltage
ω	Angular Speed

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
ASL	Autonomous Systems Lab
BLDC	Brush-less Direct Current
PMSM	Permanent Magnet Synchronous Motor
ESC	Electronic Speed Controller
FC	Flight Controller
MAV	Micro Aerial Vehicle
OMAV	Omni-directional Micro Aerial Vehicle
DOF	Degree of Freedom
PWM	Pulse Width Modulation
RCPWM	Radio Control Pulse Width Modulation
FOC	Field Oriented Control
EMF	Electro Motive Force
UART	Universal Asynchronous Receiver/Transmitter protocol
UAVCAN	Uncomplicated Application-level Vehicular Communication And Networking
CAN	Controller Area Network
RPM	Revolutions per Minute
ADC	Analog to Digital Converter

Chapter 1

Introduction

In the past decade, multi-copter type drones have been developed and spread all around the world for applications ranging from geographical mapping to entertainment. Furthermore, the latest developments on Miniature Aerial Vehicles (MAVs) have involved more physical interactions with the environment, power optimization and more Degrees of Freedom (DoFs) which leads to significantly more complex control pipelines. For instance, the Omnidirectional Miniature Aerial Vehicle (OMAV) at the Autonomous Systems Lab (ASL) in ETH Zurich and outlined in [1] consists of 6 pairs of co-axially aligned propellers with tilting axis. Mentioned platform is depicted in Figure 1.1.
Although this novel setup is highly versatile, the current rotor speed control implementation is open-loop, assuming relatively accurate and linear throttle to actual rotor speed mapping. Furthermore, the high number of propellers increases the chance of an individual failure on flight, requiring status information to be sent towards the flight controller (FC). Another important consideration is the aerodynamic interference between coaxial propellers, which surely causes disturbances leading to slower tracking. The extent of these disturbances is still unknown. Therefore, [2] recommends to perform a more thorough identification of this OMAV dynamics to decrease disturbances during optimal control design.

Objectives. In this context, this report will describe my work during my semester project with the following objectives:

- Familiarize with current firmware available for FC-ESC communication.

Figure 1.1: ASL OMAV

- Achieve rotor speed feedback from the ESC to the FC.
- Explore different ESC typologies and
- Investigate the characteristics of driving the Brush-less Direct Current (BLDC) motors using an ESC.
- Investigate features available in modern ESCs.

Chapter 2

Literature Review

2.1 Brush-less DC motor driving

2.1.1 Brush-less DC motor

Most UAV platforms spin their propellers using Brushless-DC (BLDC) motors due to their low weight, high top speeds and efficiency. Therefore, it is imperative to know the basic principles to analyze ESCs.
A BLDC motor can be 1, 2 or 3-phase. However, the most efficient version at high speeds and the ones used for UAVs is the 3-phase version. Therefore, the BLDC term will refer to 3-phase BLDC motors. Figure 2.1 shows a simplified diagram with only 4 magnetic poles whereas a typical motor used for UAVs would have 14. Here,the stator includes coils and the rotor has permanent magnets.

2.1.2 BLDC motor driving methods

Two configurations can be used to run BLDC motors as mentioned in [4], both based on Pulse Width Modulation (PWM). These are Trapezoidal and Sinusoidal. They differ in the ideal driving voltage wave-forms to use when driving the motor coils. In both cases, the best performing phase shift between phases is 120°. The effective instantaneous voltage in each coil is a fraction of the DC power supply. PWM approximates this fraction by providing a pulsing signal with a corresponding duty cycle (fraction of time the signal is high). To generate these PWM signals from a DC power source, power MOSFETs are used in bridge configurations. The

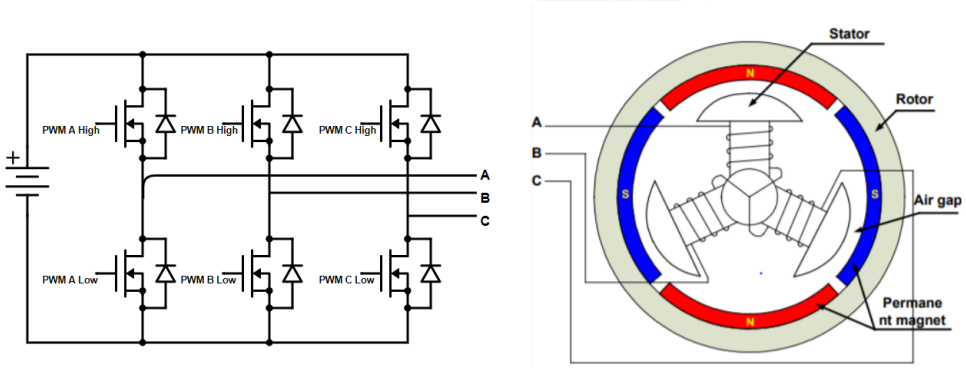


Figure 2.1: *Left* MOSFET bridge driver. *Right* BLDC motor simplified [3]

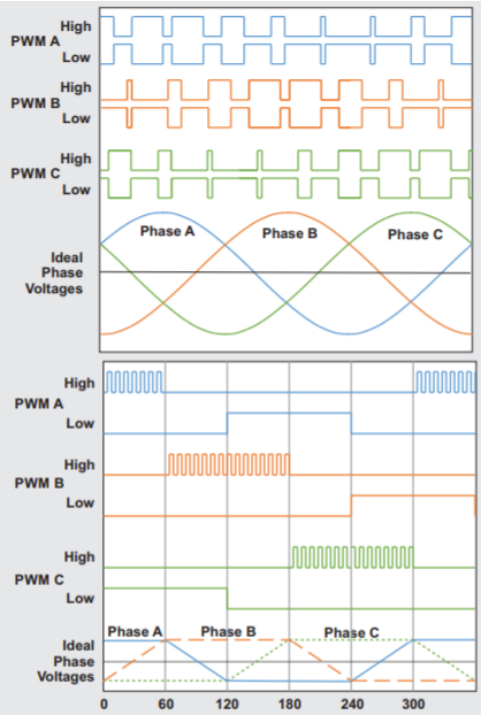


Figure 2.2: PWM Phase signals vs Rotor Position. *Top* Sinusoidal FOC, *Bottom* Trapezoidal. Adapted from [4]

specific configuration used for BLDC motors is shown in Figure 2.1

Trapezoidal: Here the ideal driving waveform is a trapezoidal wave. As shown in Figure 2.2 bottom. This method is easier to implement in hardware since a relatively low accuracy in rotor angular position estimation is needed. However, there are significant power loses caused by many spikes in the signal and also because the electrical magnetic field caused by the stator is only instantaneously at the optimal orientation relative to the rotor magnetic field (perpendicular). Here, the duty cycle put through the active phases is constant for the same rotor speed

Sinusoidal: In this case, the ideal driving waveform is a sinusoid as seen in figure 2.2 top. Note that for the same rotor speed, the duty cycle continuously changes over an electrical period to mimic the ideal sine-wave. This is because of the waveform continuously changing and because feedback control [1] is used to maintain the relative orientation rotor-stator magnetic field orientation optimal (90°). Since the duty cycle is changed more often and depending on the rotor position, this method requires a significantly higher accuracy in rotor angular position estimation. Higher efficiency is obtained in two fronts: less pulsing implies less spikes and therefore switching power losses; the magnetic fields relative orientation is controlled to be optimal.

[1] This is not rotor speed control

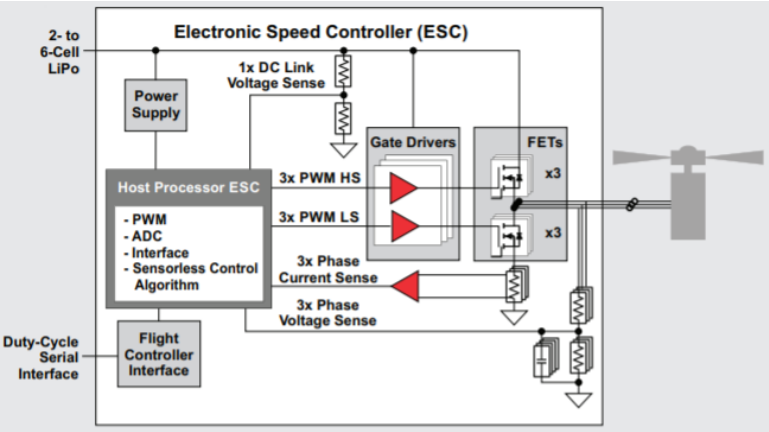


Figure 2.3: ESC general architecture [4]

2.2 Electronic Speed Controllers

2.2.1 General structure

The circuits that run the BLDC motors are commonly known as Electronic Speed Controllers (ESC). However, the majority of current available ones do not perform direct motor speed control. Their core components are a micro-controller, gate drivers, back Electro-motive force (EMF) measurement resistors and the MOSFETs (shown in Figure 2.1). In essence, what they do is receive a throttle signal from a Flight Controller (FC) representing the amplitude of the ideal driving signal (explained in previous subsection, Fig. 2.2). The diagram in Figure 2.3 shows a general electrical architecture of ESCs. Note that gate drivers are simply interface circuits that allow to turn on/off the MOSFETs.

2.2.2 Communications

ESCs need to communicate with the FC to receive throttle commands and in some cases they also send status messages. Hence, some protocols have been used and other have been developed in the context of UAVs.

Analog Protocols

All of these protocols are based on Pulse width modulation given a maximum and a minimum pulse width to be set as maximum and minimum throttle. Hence, these protocols are commonly called RCPWM (Radio Control PWM). Thee different protocols in this category merely differ in the pulse period. Figure 2.4 depicts these kind of signals on the top and Table 2.1 shows their transmission parameters. An inconvenience of analog protocols is that they need the so called "ESC calibration" which is a procedure to coordinate the the analog width range that will represent the 0-1 throttle range. This is not necessary in digital protocols as the range is setup virtually in software.

Digital Protocols

Even though most of these protocols are widely known as digital, the true purely digital protocol is UAVCAN. The rest define certain specific pulse time durations to high or low. Dshot will be explained thoroughly in particular given that it's the basis of all the other methods but UAVCAN.

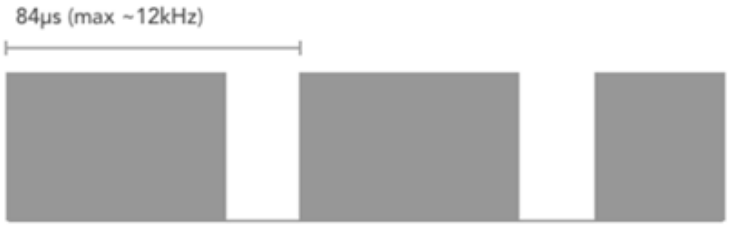


Figure 2.4: Oneshot protocol [5]

Table 2.1: ESC protocols' speeds

Protocol	Type	Pulse width range [μs]	Max Update Rate [kHz]
PWM	Analog	1000 - 2000	2000
Oneshot125	Analog	125 - 250	250
Oneshot42	Analog	42 - 84	84
Multishot	Analog	5 - 25	40
Dshot150	Digital	2.5, 5*	9
Dshot300	Digital	1.25, 2.5*	18
Dshot600	Digital	0.625, 1.25*	37
Dshot1200	Digital	0.312, 0.625*	75
Bidirectional Dshot	Digital	Same as Dshot	Same as Dshot
Proshot	Digital	Same as Dshot	Same as Dshot
UAVCAN	Digital	NA	NA

* Pulses can only take either of the two values (Digital)

Dshot: This method consists of a 16-bit packet streamed continuously. Each bit is encoded as a pulse with one of to widths, depending on the pulse width version. Figure 2.5 shows the typical DShot signal and the information organization in each packet. The packet is divided into three sections:

- Throttle command: Consists of 11 bits (Decimal values 0 -2047). Commands in the range [0,47] are status request commands whereas values in the range [48,2047] encode a 2000 levels throttle command.

- Telemetry request: When this bit is high, telemetry is high, the ESC is sends a telemetry information packet from the ESC to the FC. Including information such as: Motor Electrical RPM, ESC Temperature, Supply Voltage, Current (Available only in certain ESCs). This telemetry message is sent via another pin using Universal Asynchronous Receiver/Transmitter protocol (UART) at 115200 baud-rate. This rate is embedded in the closed-source BlHeli_32 firmware and it is not modifiable.

- Cyclic Redundancy Check (CRC): Used to check for data corruption during transmission.

Bidirectional Dshot: This protocol uses exactly the same signal from the ESC to the FC. The difference is the medium of transfer for the Telemetry signal. In this case, the telemetry packet is sent to the FC through the same throttle pin.

Proshot: The data side of this protocol works the same way as Dshot. The 16 bit digital signal is set as a 4 pulse train where each pulse can encode 4 pulse widths

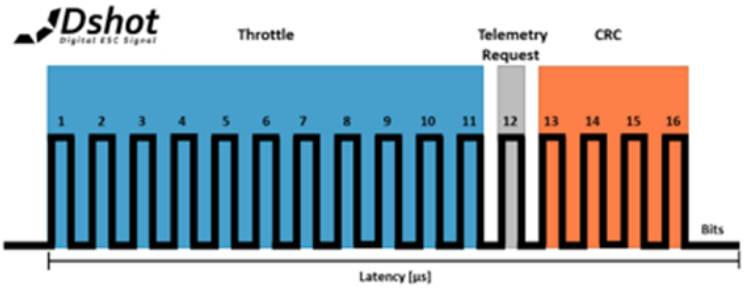


Figure 2.5: Dshot Signal Structure [6]

(as opposed to only 2 in Dshot).

UAVCAN: As its name suggests, this a full level software platform based on the Controller Area Network (CAN) bus. It supports two-way communication between multiple agents over a twisted pair wire. The signals sent are merely digital (Only high and low voltages at same duration). The protocol packet convention is illustrated in Figure 2.6, where the elements are explained in [7] as:

- SOF (1 bit): Dominant Start of Frame bit. Used for synchronization.
- Identifier (11-bits): Establishes message priority.
- RTR (1 bit): Remote Transmission Request. Asks next node for transmission.
- IDE (1 bit): Identifies if standard CAN is used or not.
- r0 (1 bit): Reserved.
- DLC (4 bits): Data length. Number of bytes transmitted.
- Data (1-64 bits): Actual data transmitted.
- CRC (16 bits): Cyclic redundancy check.
- ACK (1 bit): Indicates error free reception.
- EOF (7 bits): Indicates end of packet.
- IFS (7 bits): Time required to buffer data.

Hence, communication in this protocol is very reliable on the expense of having to send many extra bits for agent coordination and error checking. The maximum bitrate allowed in this protocol is $1Mbit/s$. However, its data transfer rate is decreased by the number of agents connected together and the amount of data each sends. If only one ESC was connected with the FC, the absolute maximum theoretical data rate would be 4.3kHz. Another problem that could arise when using this protocol is significant delays between each rotor command especially between the first motor and the last motor to receive commands. This is because each agent needs to queue when receiving and sending messages as the bus is shared.

2.3 Related Work on ESCs characterization

There have been some studies regarding modeling and characterization BLDC motors driving hardware, given the popularity of their use in UAVs in the last decade. There have been studies in both efficiency and estimation modeling for their dynamic behavior.

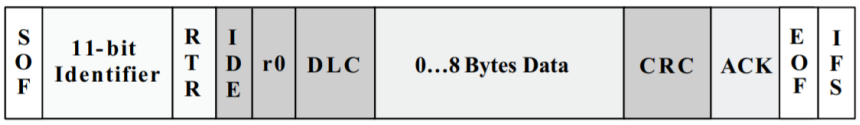


Figure 2.6: CAN bus protocol [7]

2.3.1 Efficiency

Regarding efficiency estimation, several power loss models have been developed based on non-ideal electronic components in the ESCs. Indeed, [8, 9] propose linear and non-linear models for ESC efficiency. There, it is shown that the efficiency changes depending on voltage powering the ESC and current provided to the motor. Furthermore, [10, 11] define complete models of power loss calculation for ESCs driving BLDC motors and [12] models MOSFET power losses. These factors were accounted for in a more generalized model described in [13] and summarized in Equation 2.1.

$$P_{loss} = R_{eq} I_o^2 + (\alpha I_o + \beta) V_{bus} + C f_{eq} V_{bus}^2 \,. \tag{2.1}$$

In this equation, R_{eq} represents the MOSFET on resistance usually given in datasheets, I_o the output current, V_bus the battery voltage, α and β are model values and Cf_{eq} is the equivalent capacitance frequency product for overall circuit.

2.3.2 Modeling for Control

Another stream of research is dedicated to determine the input-output characteristics of ESC-BLDC motor behavior.

Direct Throttle to Thrust mapping is subject of investigation by many sources. A linear first order differential relationship about operating points is proposed by [14], with only two determined parameters: gain and time constant. This model was further improved by [15] by introducing a dead-time parameter to account for communication delays and buffer timing which makes it non-linear. Supply Voltage is show as another parameter that affects ESC-Throttle mapping in [16] where they define a linear MIMO representation with voltage and throttle as inputs; and current and thrust as output, including non-linear elements.

Other studies have instead focused on the Throttle - rotor speed correspondence. In this category, [17] encountered non-linear steady state mapping. Furthermore, one of the first models that incorporated voltage is given by [18] where the steady state rotor speed is defined as proportional to the voltage and a first order polynomial of the throttle value, and the dynamics still as a first order differential function, these relationships are detailed in Equations 2.2.

$$\frac{\omega}{V} = at + b \quad and \quad \frac{\omega/V}{t} = \frac{k}{\tau s + 1} \,. \tag{2.2}$$

Where a and b are linear fit parameters based on linear fit of steady state data, k and τ are gain and time constant. ω represents rotor speed, V battery voltage and t throttle signal.

Another stream of investigation in this area focused on frequency methods to identify the responses to throttle commands. Chirp sinusoid input throttle signals are

used in [19] and the rotor speed output is analyzed. They define a first order differential relation for the ESC output voltage and utilize a sophisticated motor model to transfer as a transfer function between effective voltage on the motor and rotor speed. Their model even includes cogging modeling, specially important at low speeds. Their complete model is third order differential. Besides, [20] uses steady state response and frequency response using staircase and sinusoidal chirp input throttle signals, but analyses the thrust and torque generated. The model is semi-empirical and includes non-linearities such as torque friction, drag and rotor inertia. However, it assumes a PI loop in the ESC to purpose a model, which is not always the case.
The last stream of investigation in this domain was merely as data analysis. This includes [21] who analyzed thrust, rotor speed and power outputs, which where presented graphically. Additionally [22] also shows graphical representations of efficiency, power, current, voltage, thrust and torque for different thrust inputs.

Chapter 3

Experimental Setup

3.1 Equipment

3.1.1 ESC Assessment

The most important component for the analysis in this project is the ESC. Therefore, a thorough assessment of different ESCs and their firmware was made. The criteria to pick the particular board took into account that it is to be used with the Voliro OMAV at the ASL in ETH Zurich depicted in Figure 1.1. The parameters accounted for as well as the top rated ESCs are shown in Table 3.1. Further analyzed ESCs are show in Appendix A.

Therefore, we can see that there are two trends of high-performance ESCs. The first one relates to the most modern ESCs used in drone racing. They are very similar since they are mostly manufactured to use the BlHeli_32 firmware. This firmware, allows for multiple parameter modification via the software BlHeliSuite (see Appendix C for instructions). Given their modularity, they can be made into very small boards. For instance, they now come in 4 in 1 boards (i.e. one board includes 4 ESCs). The other group of high-performing ESCs use UAVCAN as communication protocol with the FC. This gives the advantage of less cabling. They use FOC which makes them more efficient than Sine Modulation versions specially at low rotor speed (below 1000 RPM). These also have possibility of on-board speed control, but it does not include any feed-forward prior. The Holybro Tekko32 F3 ESC was then selected for investigation due to the wide variety of protocols it supports, it's multiple rotor support, good heat-sinking and flexibility to change parameters.

Please refer to Section 2.2.2 for a more thorough description of Dshot, Bidirectional Dshot and UAVCAN protocols.

Table 3.1: ESC protocols' speeds

ESC Model	Speed feedback	Brake	Size [mm]	Motors	Protocol	Current Sens	Firmware	Efficiency
Flycolor X-Cross 4in1	Telemetry	Yes	42x45x7.5	4	Dshot	Telemetry	BlHeli_32	Sine Mod
Lumenier Elite 4in1	Telemetry	Yes	43x46x7.5	4	Dshot	Telemetry	BlHeli_32	Sine Mod
Holybro Tekko32 F3 Metal	Telemetry	Yes	42x42x4	4	Dshot	Analog	BlHeli_32	Sine Mod
Myxa by Zubax	UAVCAN	Yes	57x38x24	1	UAVCAN	UAVCAN	Sapog v2	FOC
Holybro Koleta20	UAVCAN	Yes	40x27x5	1	UAVCAN	UAVCAN	Sapog v2	FOC

NOTE: All ESCs that support Dshot in this table support all analog protocols and Bidirectional Dshot

3.1.2 Overall setup

The rig shown in Figure 3.1 was setup for testing purposes. The two propellers are positioned one in front of each other at 11.48cm separation, they are spun in opposite directions and positioned to exert thrust in the same direction. All of this, to emulate the behavior of rotor pairs in each arm of the OMAV presented in the introduction. The core components of this setup are as follows:

- 2 BLDC motors by KDE Direct model 2315XF-885Kv 14 poles.
- 1 clockwise Hobbyking Propeller 9in diameter 4.7in pitch.
- 1 counter-clockwise Hobbyking Propeller 9in diameter 4.7in pitch.
- 1 Tekko32 F3 Metal ESC.
- $10\mu F$ capacitor.
- 1 6S LiPo battery.
- Pixhawk 4 flight controller.
- Laptop (running Ubuntu 18.04 for firmware compilation).
- Support towers to attach motors.

Besides, the measurement equipment used was:

- 10A rated DC ammeter.
- 10A rated DC power source.
- Manual optical tachometer.

3.1.3 Electrical Wiring

Given the non-ambiguous nature of the Dshot protocol wiring, it is important to show how it should be done correctly. Hence, Figure 3.2 shows this setup in more detail.
Note that it is crucial to have the decoupling capacitor across the Analog to Digital Converter (ADC) of the Pixhawk. The reason to have it is to filter the noise induced by high frequency Dshot signals (on yellow). In fact, with no filtering, the ADC line is mostly composed of noise. A value of $10\mu F$ showed to attenuate this noise more than enough. It is also key to use ADC1_SPARE_1 (not ADC1_SPARE_2) as it has a lower reference voltage of 3.3V against 6.6V, which leads to higher resolution.
It is also important to mention that the ground wire can be connected to the ground pin in any port (usually the last pin). Further details on wiring can be seen in Appendix B

3.2 Software toolboxes

As developing tools for testing and firmware configuration, three pieces of software were used.

PX4 is the firmware running on the flight controller. It provides an on-board operating system called NuttX used in the Pixhawk 4. Additionally, it provides with a publisher-subscriber framework called *uORB* on top of which this investigation constructed. There are four structures of interest in the PX4 firmware for this purpose:

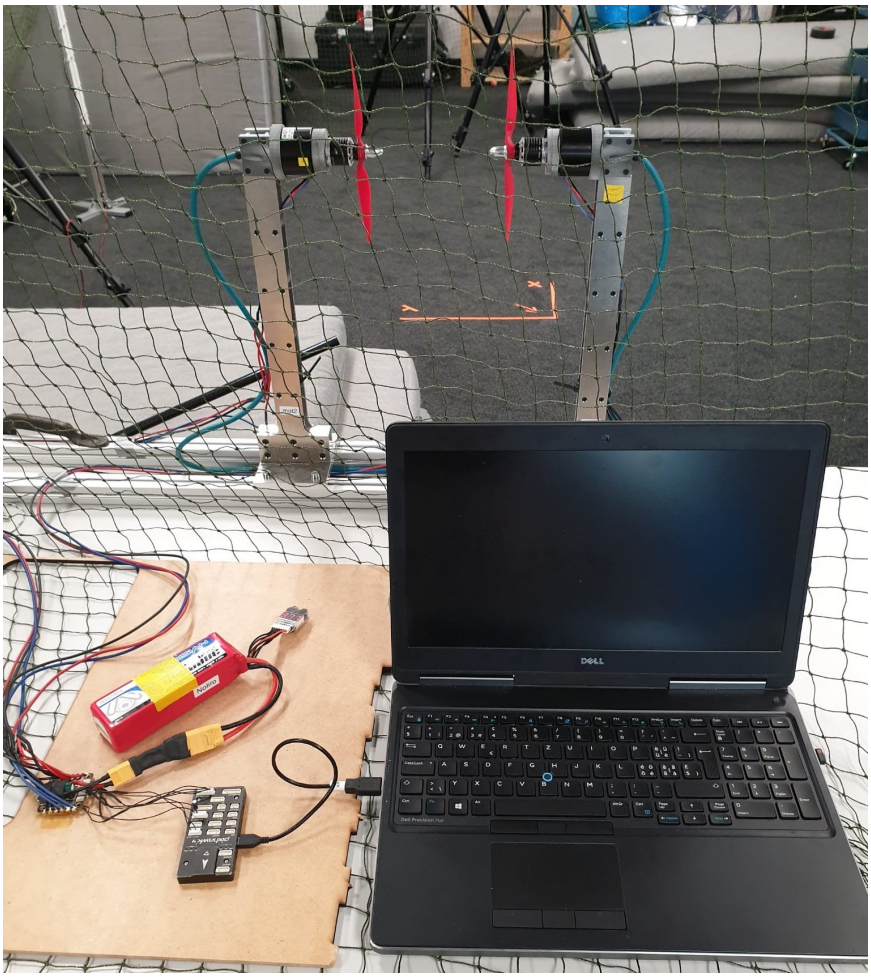


Figure 3.1: Experimental setup

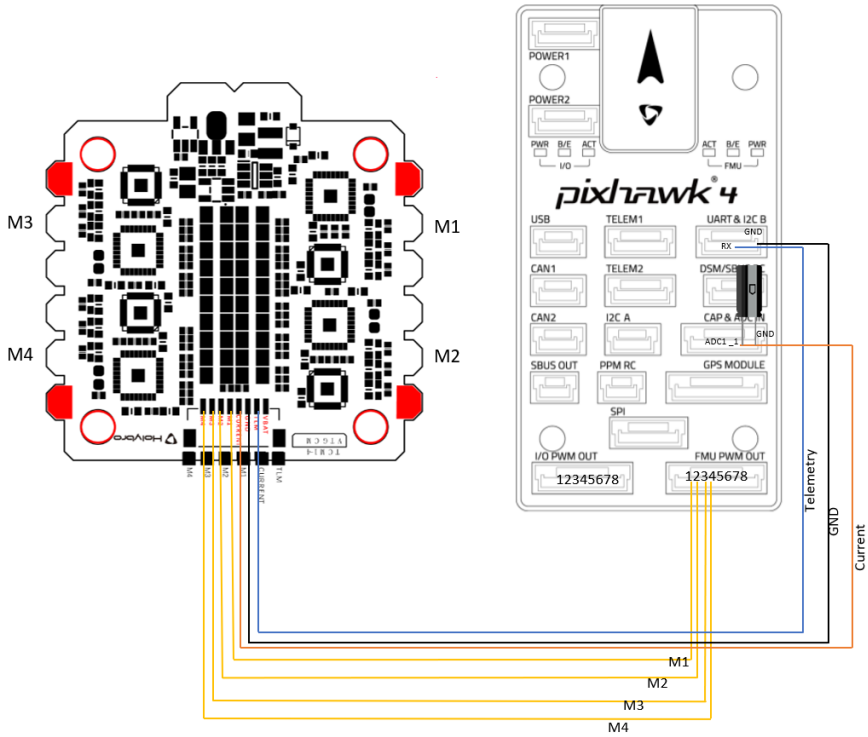


Figure 3.2: ESC-FC wiring. *Left: ESC*, *Right:* FC

drivers, modules, mixers and messages. A module is a piece of code with a specific functionality that usually runs in parallel to other modules. Drivers are special kinds of modules whose whole purpose is to interface the high-level of abstraction module structure with the low level embedded software used to generate physical signals to drive actuators. Mixers are used to normalize and transform input or output control signals. For example there are mixers that translate desired angular position into thrust values in each motor. Lastly, messages define the structure of the information sent via uORB.

QGroundControl is the Ground station software platform used with PX4. In general it allows to deploy full drones to different types of missions. It also

BlHeliSuite is the main software used to update the firmware of ESCs running with BlHeli_32. It not only allows to download newer versions, but also allows changing parameters of the ESC behavior. These include:

- Ramp-up power: Limits power on startup and low RPM.
- Motor timing: determines when angular positions with respect to magnetic poles when commutation happen. Beneficial to change with high inductance motors.
- PWM frequency: Modulation frequency to switch MOSFETs.
- Demag compensation: detects and compensates for high inductance motors.
- Sine Modulation: Approximation to FOC control.
- Maximum acceleration.

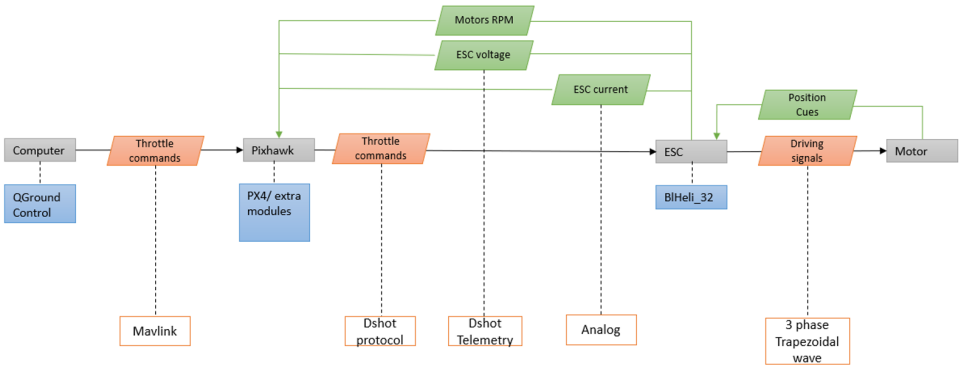


Figure 3.3: General Data Flow. Gray represents hardware, blue shows software packages, orange indicates forward control data, green represents backward measured data

- Motor Direction.
- Brake on stop. Applies regenerative brake.

Ardupilot is another flight controller firmware such as PX4. This was only used as an interface to configure ESCs using BlHeliSuite. Instructions on how to use it to configure BlHeli_32 are detailed in Appendix C

3.3 Software developed

The main aim of software development during this project was to drive motors using the Dshot protocol and obtain the telemetry packet to analyze responses and the feasibility of feedback control. The structure of data flow is color coded in Figure 3.3. It shows the protocols used for data transfer at the bottom. Furthermore, the source code can be found at:
`https://github.com/faustoTapia/px4_sem_project.git`.

3.3.1 Existent software

The software developed for this project was built on top of PX4 v1.10. The components of interest from PX4 are described below.

Dshot driver: This version of PX4 is the only one that provides support for Dshot. The driver is called "dshot.cpp". It runs the main control thread that receives commands, checks for updates and publishes received telemetry packets to uORB.

Telemetry: "telemetry.cpp". There is a telemetry handler that receives the serial stream and decodes it in the appropriate format. Since the data from all motors is shared through same telemetry line, they have to be multiplexed. This handler performs the multiplexing in a blocking manner for up to 25ms if no message is received.

3.3.2 Elements added

Mixer: "quad_x_v2.main". This mixer provides pass-through mixing, that is, it takes a control command and directly passes it as a percentage throttle to be

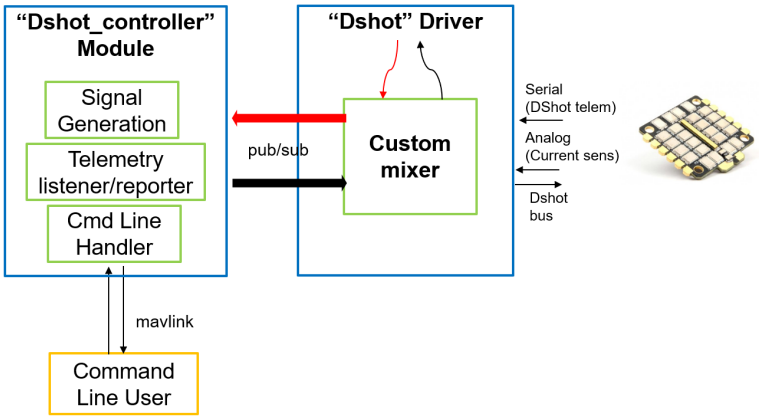


Figure 3.4: Software modules modules architecture

commanded to the ESC. The mixer is configured to receive inputs from the "actuator_controls_0" topic. Furthermore, in order to be loaded, it needs to be assigned to an airframe that itself is called on start-up within the PX4 framework. To do so, the "4001_quad_x" airframe was modified to pick the new mixer.

Dshot controller: "dshot_controller.cpp". This is the core of the functionalities developed during this project. It publishes to the "actuator_controls_0" topic at 400Hz, derived from the "vehicular_angular_velocity" topic. Furthermore, it provides with user interaction through the Nutshell. The allowed commands are:

- Listen ESC, reporting current telemetry data including RPM, voltage and temperature.
- Static throttle command to any of the 4 motors or all with non-blocking behavior.
- Sinusoidal wave of desired amplitude and period.Period can be changed from 10ms to 10000ms.
- Random waveform at a certain offset.
- Sinusoidal sweep (chirp) signal.

A thorough explanation of how to setup and utilize this tool is found in Appendix B.

The topic of interest is "esc_status" and it is published as soon as it receives data from all motors. On the ESC side, it was found that Dshot telemetry provided by BlHeli_32 only works at 115200 bits/s baud-rate and it is not a modifiable parameter. Given the size of the data packet, a telemetry rate of 300 Hz was achieved when using only one motor. This frequency would decrease inverse proportionally with the number of individual ESCs connected (number of motors). This is mainly because each the serial telemetry line is shared. Hence for 4 motors, the data-rate obtained was 75Hz.

A graphical representation of the functionalities added to PX4 is shown in Figure 3.4. Note that the standard PX4 driver procedure is that mixers are instantiated within a driver but with an argument being the driver itself. This in order to encapsulate actuator controls inside a driver while being able to use the same mixers for multiple kinds of drivers.

3.4 Setup limitations

Given the chosen hardware and firmware, specially the restriction to use a Pixhawk 4 flight controller gave a couple of limitations in when integrating the system. The most important ones are listed below:

- Digital Dshot based protocols imply relatively high CPU usage in the microcontroller due to their pulse length being very small (at most $6\mu s$ as opposed to $2000\mu s$ in RCPWM). Therefore, the software implementations use Direct Memory Access (DMA) to enable fast enough scheduling to write Dshot signals. In the case of the PX4 it only has 2 DMAs that can each be used to run 4 Dshot signal generation ports. Hence, the PX4 implementation allows for immediate 4 motor support but with some modifications up to 8 motors.
- The more sophisticated digital protocol Bidirectional Dshot would be the way of avoiding telemetry line sharing that happens in regular Dshot. Consequently, telemetry speed increases significantly. However, PX4 does not support this protocol and it seems like it does not want to move in that direction.

Chapter 4

ESC characterization

This section will introduce the different experiments realized to characterize the ESC input-output behavior. For all of them, the real-time data was recorded using the "logger" provided in PX4 that allows to store topics in the SD card connected to the circuit board. The topics logged were:

- "actuator_controls_0", to see the control signal send to the ESCs.
- "esc_status", to check the rotor speed RPM.
- "adc_report" to read the current sensor analog signal.

4.1 Sensing Evaluation

The purpose of this experiment is to determine the quality of the measurements that the ESC provides. The focus is on rotor speed measurements and current sensing. For both experiments in this section (rpm and current sensor validation), a staircase throttle signal was sent and the steady state values were used. Figure 4.1 shows typical experiment data for this section.

4.1.1 ESC RPM feedback

Experiment in the figure above was repeated 7 times to obtain more data. Here, the rotor speed was measured using the manual tachometer and comparing it to the

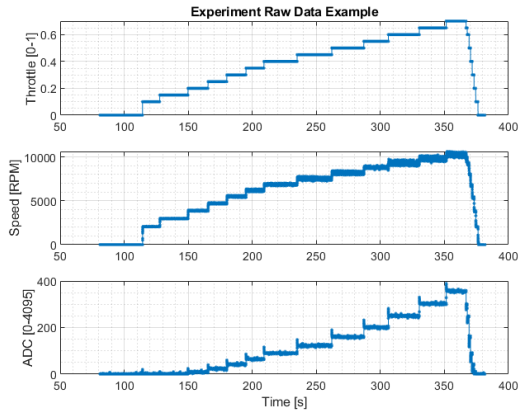


Figure 4.1: Sensing Validation Experiment Example

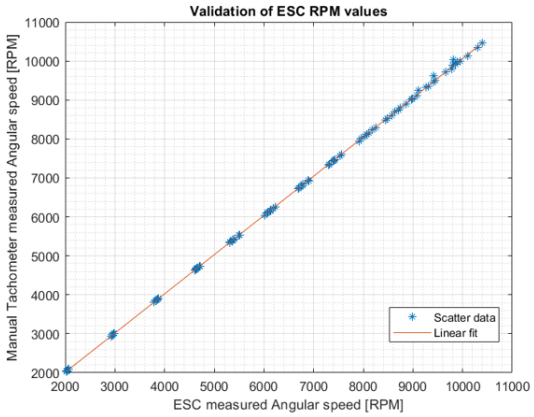


Figure 4.2: Validation of RPM measurements

RPM measurements coming from the ESC. Figure 4.2 shows this relationship. There is very high accuracy and a strong linear behavior. For instance, the fitted curve is characterized as in 4.1. Notice the high R^2 value and slope of 1. Accordingly, the average relative error is 0.6366%

$$RPM_{actual} = 1.0064 RPM_{ESC} - 1.523 \quad with \quad R^2 = 0.999\,. \tag{4.1}$$

Therefore, the RPM measurement is accurate enough to be used for feedback control purposes. The accuracy is expected to be high because the ESC needs to have angular estimates of the rotor positioning in order to run and rotor speed is internally calculated by counting revolutions in a fixed period.

4.1.2 ESC current sensing

Another important sensor, to calculate power consumption and in the future help estimate time of flight more accurately is the current sensor.

There are two trends to send this information: digitally via an element in the Dshot packet or analogically. In both cases, the the measurement is done via a shunt resistor and the voltage across it will be measured with an ADC. In the digital case, the ADC is located in the ESC and therefore the current estimation is calibrated with the nominal shunt-resistor value. In the analog case, the ADC needs to be in the flight controller. These shunt resistors are usually of 5% tolerance, hence the current measurements need to be calibrated accordingly. This is impossible in the digital case if the firmware is closed-source.

As the measurement is shunt resistor based, it is expected that the relationship is highly linear. This is confirmed in Figure 4.3 that was used to calibrate the current sensor measurement.

4.2 Motor-drive dynamics

4.2.1 RPM-Voltage-Throttle relation

As shown in previous literature, the rotor speed also varies with respect to battery voltage. Therefore, an experiment similar to the one described in Figure 4.1 was set up for different battery voltages that were varied using a power supply. After

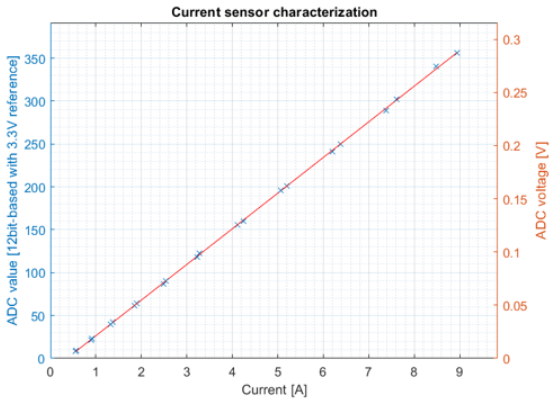


Figure 4.3: Validation of current measurements

analyzing the data, one can see that even though the voltage does not change the output as dramatically as throttle does, the variation is significant for instance it appears linear when looking at the cross-section. It is also coherent with theory as higher voltages generate linearly related higher currents and therefore higher induced magnetic field. In the end allowing for higher torques and steady state rotor speed. Furthermore, a significant non-linearity can be observed in the dependence from throttle. The reason for this is the way the ESC approximates driving voltages (switching transistors) because it introduces high frequency current components that can pass through parasitic capacitances in the circuit. Therefore, augmenting from the linear throttle dependency model explained in [18], the quadratic model in Equation 4.2 is introduced to account for non-linearities. The parameters were found using Non-linear least squares optimization method given the test data.

$$\begin{aligned} \omega &= (at^2 + bt + c)V\,. \\ a &= -7.038 \times 10^{-5} \\ b &= 0.4155 \\ c &= 3.168 \end{aligned} \tag{4.2}$$

Figure 4.4 shows the data points as black dots and the surface represents the model proposed above. The fit provided a good approximation with $R^2 = 0.996$ and on average 1.5% error in rotor speed obtained.

4.2.2 ESC to ESC discrepancies

Given the complexity of the OMAV and that the current control assumes equal throttle to RPM mapping for all ESCs, this investigation also assessed the variability in rotor speed depending on the hardware used. To do so, different ESC-motor combinations were subject to staircase-like signals as in the previous experiments. Two factors were considered into this analysis: motors and ESCs. Indeed, significant variations for different motors and ESCs combinations were found as it is summarized in Figure 4.5a and 4.5b.

From Figure 4.5, we can see that variations between same-model parts can cause discrepancies of 5% in rotor speed and 10% in thrust generated. Therefore, calibration is key. Additionally, Figure 4.6 shows the representation of these measurements as Gaussian random variables with the the mean and double standard deviation range (for 95% confidence interval). Note that the standard deviation increases

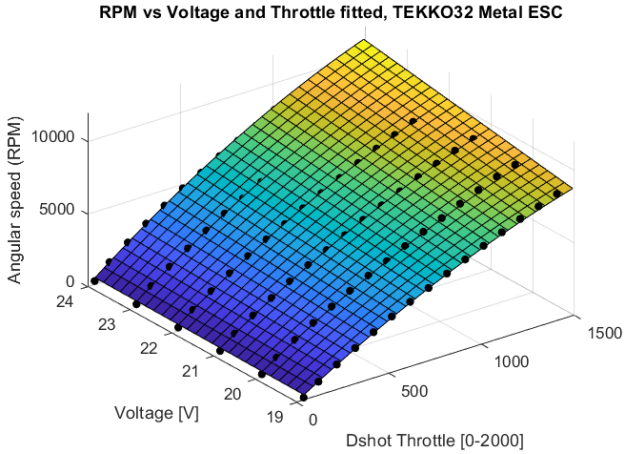
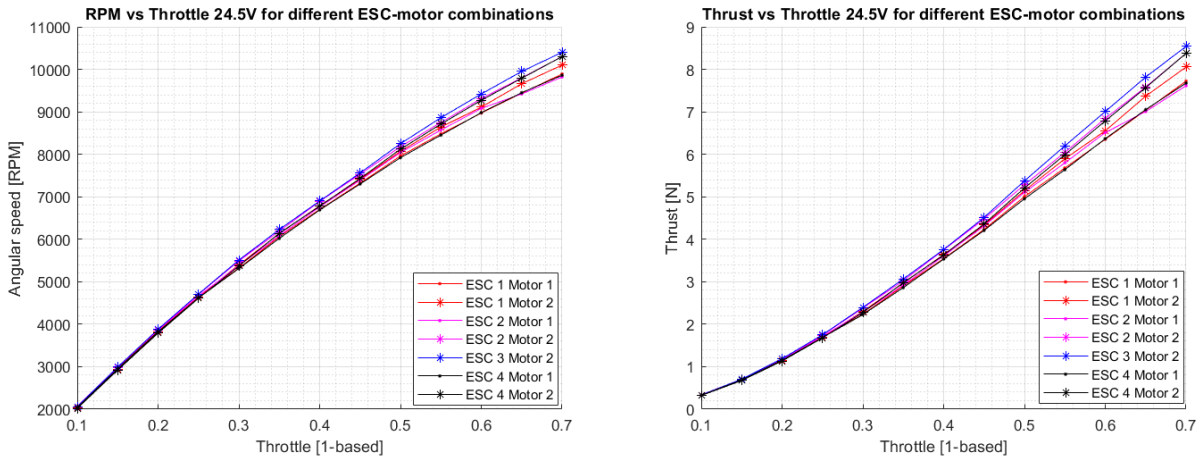


Figure 4.4: RPM dependence on throttle and voltage



(a) RPM (b) Thrust

Figure 4.5: Discrepancies for different ESC-motor responses

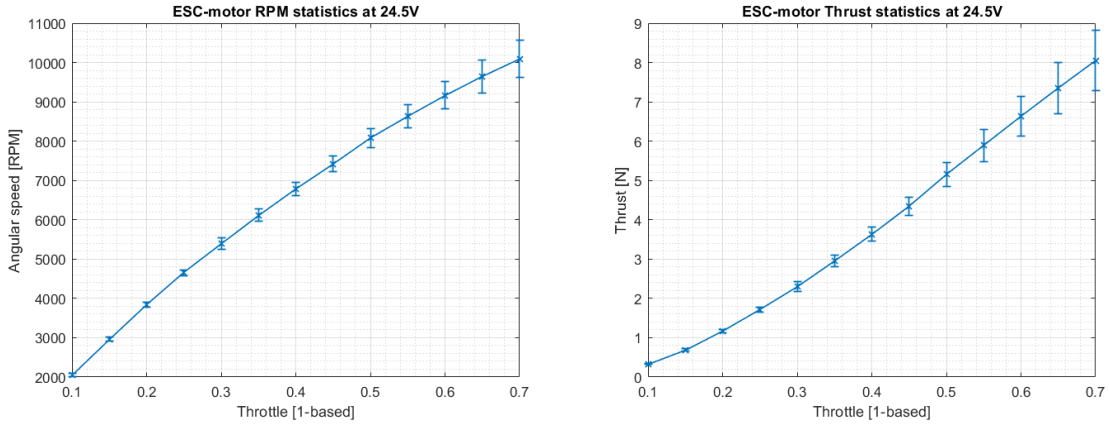


Figure 4.6: ESC-motor combination characterization showing mean and 2 standard deviation range

with throttle too. With this model, the variability range for rotor speed was was up to $1.5N$ and $960RPM$ at high throttle. Therefore, system identification is recommended to be in a motor-ESC pair basis. Two of the factors that could cause this variation were isolated and tested separately, to see individual contributions to uncertainty. That is, testing with different motors for a fixed ESC and vice versa.

Motor differences

For this analysis, the experiments with the same ESC but different motor were analyzed. A typical curve analyzed in this section is shown in Figure 4.7. Again, the difference between the response with two motors increases with throttle, reaching $440RPM$ in speed and $0.7N$ in thrust at 0.7 throttle.

Higher rotor speed values accentuate this difference more that lower ones. This suggests that a viscous friction element acting differently since the same propeller is used. The reason to suggest viscous friction are the ball bearings. In fact, [23] studied friction for robotic machines, where bearings are explained to have a representative viscous friction component.

Another element contributing to higher viscous friction would be slight miss-alignments when manufacturing the motors. That is, axis-bearing misalignments that would result bearing load increase , which in turn rises friction torque.

ESC differences

To analyze this, the 4 individual ESCs of a TEKKO board were compared using the same motor, taking the corresponding subset from the data in Figure 4.5. The data statistics is compiled in Figure 4.8 by including taking the mean and 2 standard deviation range for 95% confidence interval. Even though the individual ESCs belong to the same model, significant variation can be seen. For instance, it shows the 95% intervals can be as high as $503RPM$ in rotor speed and $0.81N$ in thrust at 0.7 throttle. In absolute terms, the largest discrepancy was $300RPM$ in speed and $0.49N$ in thrust at 0.7 throttle. It is also noticeable that these disparities increase with throttle.

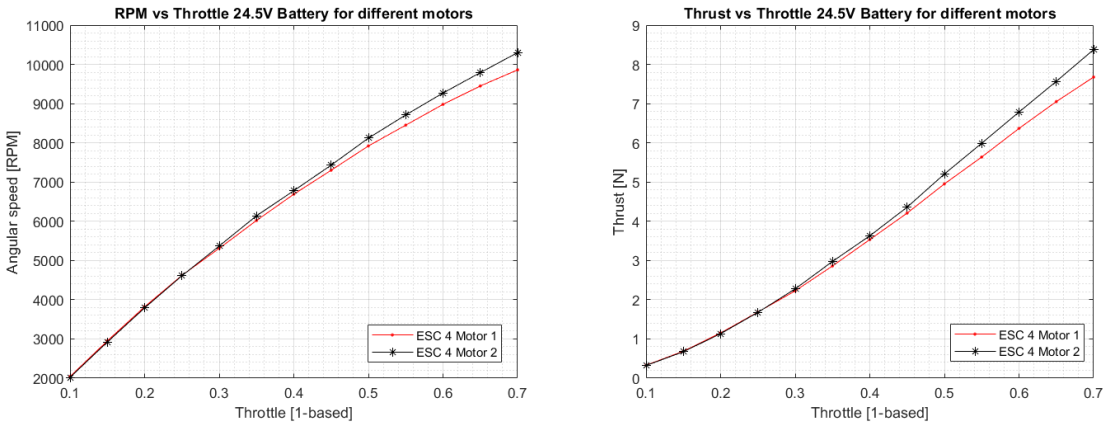


Figure 4.7: Comparison of different motors response

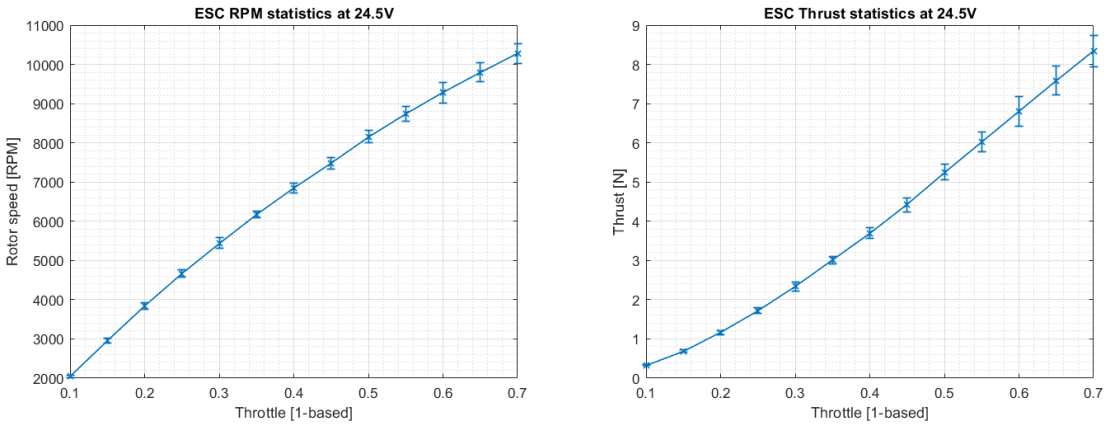


Figure 4.8: Different ESCs responses statistics

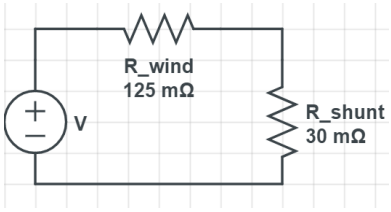


Figure 4.9: Shunt circuit simplification

The different behavior of same model ESCs can be explained mainly by the tolerances and imperfection of it's components. For instance, a big contributor is the shunt resistor used to measure current. These usually have 5% tolerance (which is also wise to calibrate the current measurement). To analyze the impact this has, the MOSFET to rotor windings connection is simplified to DC analysis as it is the main components that drives the motor. Therefore, the simplified circuit for each "on" part of the driving looks is in Figure 4.9, given the connection of shunt resistors shown in Figure 2.3.
With this simplified system, one can calculate the voltage across the motor active windings and then obtain the difference for maximum variation in shunt resistor, using Equation 4.3. The winding resistance was taken as $R_{wind} = 125m\Omega$ from the motor data-sheet [24] and the shunt resistor as $R_{shunt} = 30m\Omega$ from the ESC board. With this values, the maximum discrepancy between equivalent DC component voltage applied is 2% of the source voltage.

$$V_{wind-dif} = abs\left(\frac{R_{wind}}{R_{wind} + R_{shunt}} - \frac{R_{wind}}{R_{wind} + R_{shunt} \pm tol}\right) V \tag{4.3}$$

Another big contributor to this discrepancies due to ESCs non-uniformity is the switching loss. This corresponds to the power losses caused by the high frequency current peaks generated by the switching but also by the time during turning on-off transients, causing interferences between waveform sections and current flows in places where none is planned. Hence, these losses come from three different physical roots [11, 25]:

- Conduction loss. This refers to the power loss within each MOSFET, when they are turned on. and it is given by: $P_{cond} = V_{rms}^2/R_{DSon}$ where the drain to source resistor R_{DSon} is less than $1.2m\Omega$ (typically $0.9m\Omega$) obtained from the MOSFETs data-sheet [26]

- Switching loss. This corresponds to the time it takes to transition from low to high states in each MOSFET. It is calculated for rising and falling edges as $P_{switch} = 0.5V \times I \times t_R \times f_{sw}$. Where t_R is the switching time and f_{sw} the modulation frequency.

- Diode loss happens during the dead-time between commutation transitions (i.e. all MOSFETs are off in this time). When there is a deadtime, the instantatnous current must flow through some component. For instance, it finds a path through the body diode of the lower MOSFETs (See Fig. 2.1). The body diode forward voltage is given with a high uncertainty in the data-sheet as maximum of $1.2V$ [26] but without a minimum threshold. Therefore, this is expected to give significant variability between ESCs of the same model.

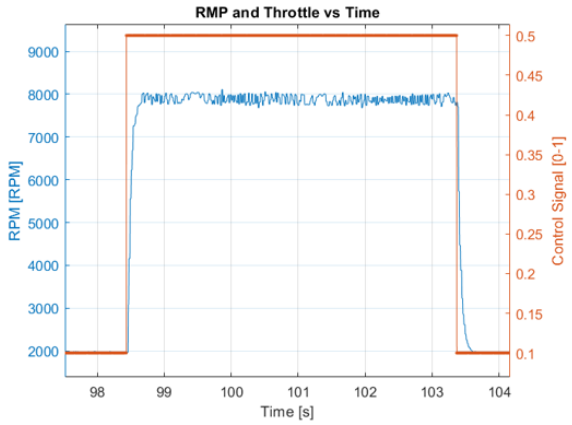


Figure 4.10: Signal used to characterize speed of response

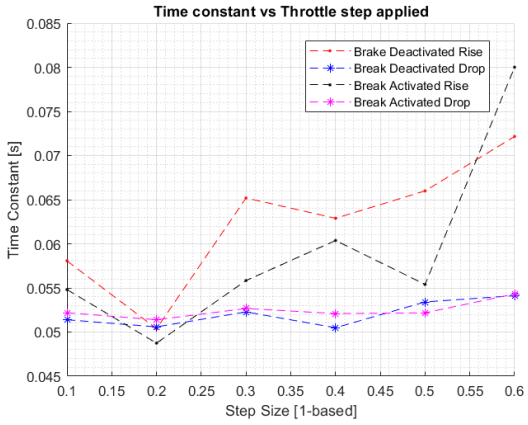


Figure 4.11: Speed response Rise time for different active break and edge direction settings

4.2.3 Transient response and active break

Another important characteristic for control, specially if cascade controllers are used (usually the case in UAVs), is the speed of the system when reacting to a command. This experiment aims to determine this characteristic by commanding rising and falling steps of different amplitudes. This was performed for the case with active and inactive brake [1]. A typical signal and it's response used to measure both rise and fall transitions is shown in Figure 4.10. Note that the response shows first order characteristics as no overshoot is seen, which was the case for all step sizes.

The time constant of these signals was computed for each step size by measuring the time take to reach 63% of the steady state point. Figure 4.11 summarizes these results.
The first fact to notice is that there is no distinctive trend that allows to discern active brake effects on speed of the response. However, we see a lower time constant for the falling edge case, averaging at $52,5ms$ for all step sizes which is not the case for the rising edges whose rise time changes from $49ms$ to $80ms$ with an overall

[1] This feature is activated in the ESC firmware, BLHeli_32. Instructions of how to change it are found in Appendix C

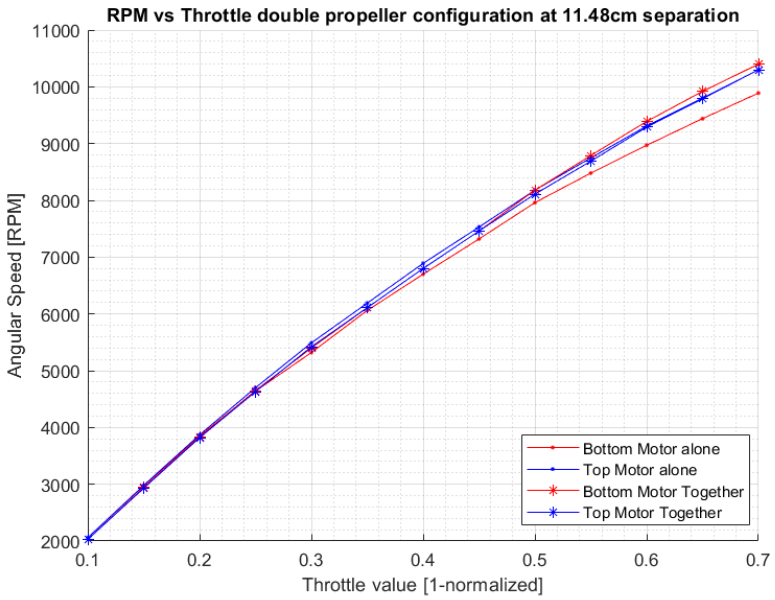


Figure 4.12: Interference between coaxially aligned motors

increasing trend with rising step size. Consequently, the only possible explanation for the falling edge time constant being lower than the rising edge is that the active brake is always active in this particular ESC. Therefore, this implies that the response is non-linear and therefore is very difficult to model in frequency domain.

From the same data collected, the maximum acceleration in rising and falling steps was calculated, encountering a maximum $9616\frac{rad}{s^2}$ rising acceleration and $8317\frac{rad}{s2}$ falling acceleration.

4.2.4 Coaxial rotor interference

A special characteristic of the OMAV in question, is the coaxial position of propellers. The interference was analyzed by performing staircase signals as in Fig. 4.1 for both rotors in the Fig. 3.1 setup simultaneously. The steady state values were used for analysis.

Figure 4.12 compares the response of bottom and top propellers when run individually opposed to when run together. It can be clearly seen that when both propellers are run together, the Bottom motor increases in speed. For instance, the error in comparison to the individual rises up to 5.2% or $510RPM$ at 0.7 throttle. This variation also increases with increasing throttle. Regarding the top propeller, it did not show significant change when run together, for instance only 1% error.

The speed increase in the lower propeller is explained by two facts: no feedback control is performed; and the top propeller induces a downward flow that pushes the bottom blades in the same direction of motion. Since the equilibrium rotor speed is dependent on the load it gets and the flow provided by the top motor would generate a favorable torque in the bottom propeller, the equilibrium rotor speed at the lower side will be higher.

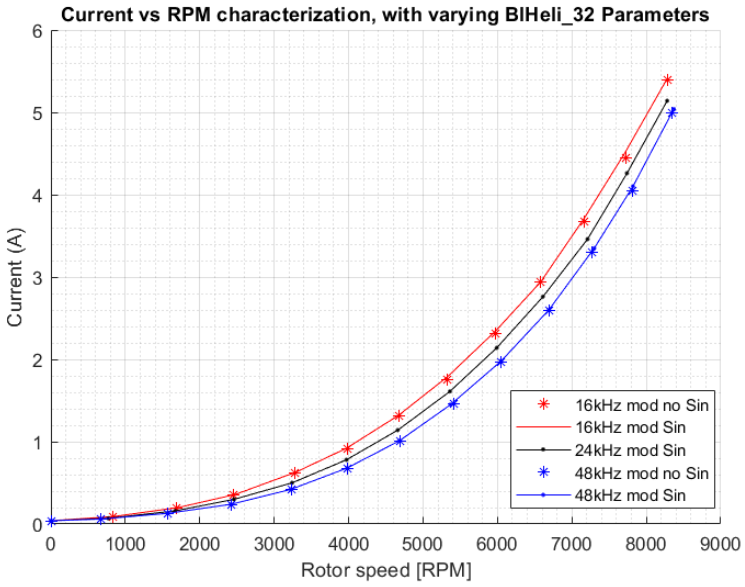


Figure 4.13: Current consumed for different modulation parameters

To summarize, this discrepancy is caused by the different air inlet speeds in the top and bottom propeller that cause different aerodynamic torques in each propeller.

4.3 Frequency Modulation and Efficiency

The last element analyzed in during the project was the effect of activating Sine Modulation mode and using different modulation frequencies for the motor driving signals [2]. Hence, a staircase signal as in the previous sections was used for each combination of modulations ($16kHz$, $24kHz$ and $48kHz$) with and without sine mode. in BlHeli_32 sine mode is an approximation to FOC. The speed curves did not show significant reference. However, the efficiency does change. As shown in Figure 4.13, higher modulation frequencies show less current consumption to achieve fixed rotor speeds. This implies that more thrust would be obtained for less electrical power, hence more efficient.

Higher efficiency comes from a better driving signal, explained more thoroughly in Section 2.1.2.

[2]Setup within BlHeli_32 as shown in Appendix C

Chapter 5

Conclusions and Recommendations

For high complexity copter vehicles like the OMAV at ASL, it is imperative to have a good understanding of how actuators react to commands, to maintain stability or to be able to design more aggressive controllers with less robustness to model inaccuracies. Therefore, this project aimed to investigate sensing and performance characteristics of modern ESCs relevant to control objectives. Especial focus is given to the factors that affect accurate throttle speed to rotor mapping.

During the project, a thorough investigation of the working principles governing BLDC motors and ESCs was performed. For instance, the different methods of driving BLDC motors are introduced concluding that at high speeds there is little difference between FOC and trapezoidal based ESCs. Different FC to ESC communication protocols were also discussed concluding that digital protocols are key to obtain ESC feedback. Current research has proven that throttle to rotor speed mapping is dependent on voltage. However, most of their models try to simplify the system to a linear model. A significant amount of research has been focused on efficiency of ESCs.

The "TEKKO32 F3 Metal 4 in 1" ESC with BlHeli_32 capabilities was used and examined. To do so, a software tool to utilize the Dshot protocol under the PX4 framework with a Pixhawk 4 was developed. It allowed to run motors via Nutshell and receive rotor status information including rotor speed, current consumed and battery voltage. This feedback signal was obtained at $300Hz$ for a single motor and $75Hz$ for 4 motors simultaneously. The hardware was setup in a rig mimicking coaxial rotor pairs in the OMAV setup.

Experimental data suggests that the current and rotor speed measurements given by the ESC are very accurate provided the current ADC is calibrated. Furthermore, a model quadratic on throttle and linear on voltage used to describe rotor speed showed to be representative with errors below 1.5%. Different ESCs of the same model response were analyzed for different motors, showing that there is significant variability of throttle to RPM mapping caused by ESC and motor inconsistencies. Besides, the active brake in the ESC is always active and allows for time constants of $52.5ms$ on falling steps and $49ms - 80ms$ on rising edges. Coaxial propellers cause significant interference that increases the lower propeller size up to 5.2%. Lastly, for BlHeli_32 based ESCs, higher PWM frequencies in the motor driving signals showed higher efficiency up to $48kHz$ which is the maximum modulation frequency allowed

in this hardware. However, for much higher frequencies, efficiency is expected to lower.

Recommendations.
The main recommendations to be presented are dependent on the use or not of a Pixhawk 4. This FC limits the number of of Dshot outputs to 4 (or 8 with significant change to PX4). PX4 does not support Bidirectional Dshot either.

Therefore, if not using a Pixhawk 4, and ESC with Bidirectional Dshot capabilities (such as the one used in this project) is recommended as it allows telemetry feedback through the same lines that throttle outputs are run allowing for real-time rotor speed feedback. Then, better rotor control can be performed in the FC.

If using a PX4, the best option would be to use a UAVCAN based ESC setup to the maximum baud-rate allowed of $1 Mbit/s$, allowing to obtain the same data as in Dshot protocol but at higher speeds. A good suggestion of this kind of ESCs is the Zubax Myxa motor, that support UAVCAN and rotor speed control internally [27].

Lastly, if only feed-forward control is desired, proper calibration for each motor-esc pair need to be performed in order to achieve desired rotor speeds when commanding throttle values. This calibration also needs to account for variations in voltage. I.e. the feed-forward model fitted needs to be a 3 dimensional, ideally non-linear on throttle and linear on voltage.

Bibliography

[1] K. Bodie, Z. Taylor, M. Kamel, and R. Siegwart, "Towards Efficient Full Pose Omnidirectionality with Overactuated MAVs," in *Proceedings of the 2018 International Symposium on Experimental Robotics*, J. Xiao, T. Kröger, and O. Khatib, Eds. Cham: Springer International Publishing, 2020, pp. 85–95.

[2] M. Allenspach, K. Bodie, M. Brunner, L. Rinsoz, Z. Taylor, M. Kamel, R. Siegwart, and J. Nieto, "Design and optimal control of a tiltrotor micro aerial vehicle for efficient omnidirectional flight," 2020.

[3] J. Zhao and Y. Yu, "Brushless DC Motor Fundamentals," Monolithic Power Systems, Tech. Rep., 2011.

[4] K. Mogensen, "Motor-control considerations for electronic speed control in drones," *Analog Applications Journal*, pp. 1–7, 2016.

[5] Backyard Robotics, "On ESC Protocols Part II," 2018.

[6] Speedgoat, "Dshot usage notes," 2020.

[7] S. Corrigan, "Introduction to the Controller Area Network (CAN)," Texas Instruments, Tech. Rep., 2016.

[8] A. Gong and D. Verstraete, "Experimental testing of electronic speed controllers for UAVs," in *53rd AIAA/SAE/ASEE Joint Propulsion Conference, 2017*, 2017.

[9] A. Gong, R. Macneill, and D. Verstraete, "Performance Testing and Modeling of a Brushless DC Motor, Electronic Speed Controller and Propeller for a Small UAV Application," 2018.

[10] P. Millett, "Calculating Motor Driver Power Dissipation," Texas Instruments, Dallas, Texas, Tech. Rep., 2012.

[11] A. Keskar, N. and Batello, M. and Guerra, A. and Gorgerino, "Application Note AN-1048: Power Loss Estimation in BLDC Motor Drives Using iCalc," International Rectifier, Tech. Rep.

[12] M. Graovac, Dušan and Pürschel, "MOSFET Power losses Calculation Using the Data-sheet Parameters," Infineon, Neubiberg, Germany, Tech. Rep., 2006.

[13] Tritium, "WaveSculptor 22 Motor Drive User's Manual," Tritium, Tech. Rep., 2013.

[14] M. Yoon, "Experimental Identification of Thrust Dynamics for a Multi-Rotor Helicopter," *International Journal of Engineering Research & Technology (IJERT)*, vol. 4, no. 11, pp. 206–209, 2015.

[15] R. T. Torres and F. Sergii, "Brushless Direct Current Propulsion System Identification," in *Integrated Computer Technologies in Mechanical Engineering*, M. Nechyporuk, V. Pavlikov, and D. Kritskiy, Eds. Cham: Springer International Publishing, 2020, pp. 105–113.

[16] G. Szafranski, R. Czyba, and M. BŁachuta, "Modeling and identification of electric propulsion system for multirotor unmanned aerial vehicle design," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, pp. 470–476.

[17] J. A. Prakosa, D. V. Samokhvalov, G. R. V. Ponce, and F. Sh. Al-Mahturi, "Speed Control of Brushless DC Motor for Quad Copter Drone Ground Test," in *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2019, pp. 644–648.

[18] A. Moutinho, E. Mateos, and F. Cunha, "The Tilt-Quadrotor: Concept, Modeling and Identification," in *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, 2015, pp. 156–161.

[19] C. Xiang, X. Wang, Y. Ma, and B. Xu, "Practical Modeling and Comprehensive System Identification of a BLDC Motor," *Hidawi Publishing Coorporation*, vol. 2015, pp. 1024–123X, 2015.

[20] L. Wu, Y. Ke, and B. Chen, "Systematic Modeling of Rotor-Driving Dynamics for Small Unmanned Aerial Vehicles," *Unmanned Systems*, vol. 6, no. 2, pp. 81–93, 2018.

[21] D. Kotarski, M. Krznar, P. Piljek, and N. Simunic, "Experimental Identification and Characterization of Multirotor {UAV} Propulsion," *Journal of Physics: Conference Series*, vol. 870, p. 12003, jul 2017.

[22] M. Sulewski, L. Ambroziak, M. Kondratiuk, and A. Stulgis, "Identification of electric propulsion system for unmanned aerial vehicles," *AIP Conference Proceedings*, vol. 2029, no. 1, p. 20072, 2018.

[23] B. Armstrong-H, P. Dupont, and C. Canudas, "A Survey of Models, Analysis Tools and compensation Methods for the Control of Machines with Friction," *Automatica*, vol. 30, no. 7, pp. 1083–1138, 1994.

[24] KDE-Direct, "KDE2315XF-885 Brushless Motor," 2020.

[25] E. Kahrimanovic, "Power Loss and Optimised MOSFET Selection in BLDC Motor Inverter Designs," Infineon, Tech. Rep., 2016.

[26] Infineon, "StrongIRFET IRF7480MTRPbF," Tech. Rep., 2016.

[27] Zubax-Robotics, "Myxa. High-end PMSM/BLDC motor controller," 2015.

Appendix A

ESCs analyzed

Table A.1: ESCs analyzed

Model	Source
Flycolor X-Cross 4in1	https://www.getfpv.com/flycolor-x-cross-60a-3-6s-blheli32-4-in-1-esc.html
Lumenier Elite 4in1	https://www.getfpv.com/lumenier-elite-60a-2-6s-blheli-32-4-in-1-esc.html
Diatone Mamba 4in1	https://www.getfpv.com/diatone-mamba-506-50a-6s-dshot1200-4-in-1-esc.html
Hobbywing Xrotor micro 4in1	https://www.getfpv.com/hobbywing-xrotor-micro-60a-6s-4-in-1-blheli32-esc.html
T-Motor F55A Pro II	http://store-en.tmotor.com/goods.php?id=766
Aikon AK32	https://fpvracing.ch/de/esc/2762-aikon-ak32-blheli32-55a-4in1-esc-3-6s.html
Holybro Tekko32 F3	https://fpvracing.ch/de/esc/2598-holybro-tekko32-f3-metal-4in1-esc-4-6s.html
Myxa by Zubax	https://zubax.com/products/myxa?gclid=Cj0KCQjwgJv4BRCrARIsAB17JI6NbuRoO7aqXnWGHVKWzni7bw07vGcZHrsc7aJW299Lf9puqg4C_V0aAiHGEALw_wcB
Holybro Koleta20	http://www.holybro.com/product/kotleta20/
KISS ESC32A	https://kiss.flyduino.net/the-kiss-experience/kiss-esc32a-32bit-electronic-speed-controller/
DJI TAKYON Z650/Z660	https://www.dji.com/ch/takyon-z650?site=brandsitefrom=landing_page
DJI TAKYON Z425-M/Z415-M	https://www.dji.com/ch/takyon-z425-m-and-z415-m?site=brandsitefrom=landing_page
Foxtech 8120 FOC	https://www.foxtechfpv.com/foxtech-8120-100kv-foc-esc-hv-combo-2.html#yt_tab_products1
X2-series TMM 6026-3	https://www.mgm-controllers.com/helicopters/speed-controllers-escs-3/tmm-6026-3-for-helicopters-x2-series.html
T-Motor ALPHA 40A	https://store-en.tmotor.com/goods.php?id=580
T-Motor FLAME 60A HV	https://store-en.tmotor.com/goods.php?id=370
T-Motor AT 30A	http://store-en.tmotor.com/goods.php?id=901

Appendix B

Custom PX4 toolbox instructions

PX4-DShot Configuration Documentation

1 Introduction

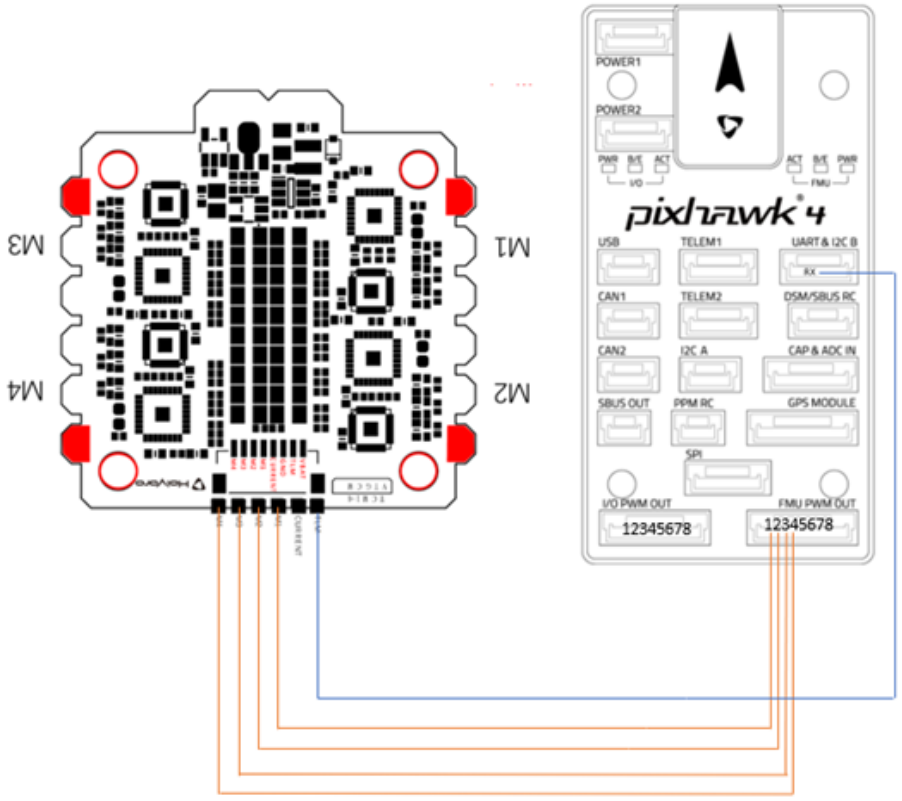
This document instructs how to setup a pixhawk 4 to run motors using the Holybro Tekko 32F3 Metal 4in1 ESC. The esc official site can be found at: http://www.holybro.com/product/tekko32f3-metal-4in1-esc65a/

Further explanation of using the DShot support module for testing developed by Fausto Tapia on top of the PX4 version 1.10. The source code can be cloned from:
https://github.com/faustoTapia/px4_sem_project.git

2 Hardware

The hardware needed for setup is the following:

- pixhawk 4
- Holybro Tekko 32F3 Metal 4in1 ESC
- USB type A to USB type B micro cable (to connect computer to pixhawk)
- 10 pin connector and 6 pin connectors for Pixhawk 4
- ESC connector (Included in ESC box)



The following Connections need to be made from the ESC to the Pixhawk 4:

ESC	Pixhawk
1. VBAT (Battery Voltage measurement)	Voltage measurement pin (not crucial for esc protocol, can be left disconnected). This value can also be retrieved from the telemetry package instead.
2. TLM	Requires configuration in software, but can be either of: • UART & I2CB (RX, pin 3) • TELEM1 (RX, pin 3) • TELEM2 (RX, pin 3)
3. GND	Can be connected in any ground pin in PX4. For convenience set to the same port as TLM. So Could be either of:

	• UART & I2CB (GND, pin 6) • TELEM1 (GND, pin 6) • TELEM2 (GND, pin 6) Another convenient placement could also be along with FMU PWM OUT: • FMU PWM OUT (GND, pin 10)
4. Current	PX4 analog current measurement pin
5. M1. Motor 1 Throttle	From port FMU PWM OUT, either of: • FMU_CH1, pin 2 • FMU_CH2, pin 3 • FMU_CH3, pin 4 • FMU_CH4, pin 5
6. M2. Motor 2 Throttle	
7. M3. Motor 3 Throttle	
8. M4. Motor 4 Throttle	

A thorough schematic of pins and ports numbering for the Pixhawk 4 can be found here:
http://www.holybro.com/manual/Pixhawk4-Pinouts.pdf

3 Software

NOTE: Although the development toolbox can be installed in Window, Linux or Mac OS X, it is RECOMMENDED TO use an Ubuntu LTD (I used it with Ubuntu 18.04 and it is also the recommended OS)

3.1 Install QGroundControl

Download and install QGroundControl. Detailed instructions are found here:
https://docs.qgroundcontrol.com/en/getting_started/download_and_install.html

3.2 Setup development environment

The instructions in this section will walk through the **Linux** setup.

3.2.1 Installing "Gazebo, JMAVSim and NuttX (Pixhawk) Targets"

Install the "Gazebo, JMAVSim and NuttX (Pixhawk) Targets" analogous to what is described in (https://dev.px4.io/master/en/setup/dev_env_linux_ubuntu.html) for the original px4 software, but with different source code. Follow the steps below:

1. Download PX4 source code:
    a. Open a terminal window
    b. Browse to the directory where you would like to setup the environment
    c. Perform the command:

        git clone https://github.com/faustoTapia/px4_sem_project.git --recursive

        Note: It's a private repository so contact Fausto Tapia to request access.

2. Run the ubuntu.sh with no arguments (in a bash shell) to install.
    a. In the in the px4_sem_project folder run:

bash ./Tools/setup/ubuntu.sh

3. Restart the computer

If you would prefer to check other OSs, please check: https://dev.px4.io/master/en/setup/dev_env.html and select your OS. Again, Linux has more extensive support, so it is suggested.

3.3 Install and setup Visual Studio Code

IMPORTANT: when Following the PX4 standard setup instructions, instead of opening the folder "Firmware", open the folder "px4_sem_proj". The instructions you want to follow are:

1. Installation and Setup
2. Building

Please find the instructions in the site below:

https://dev.px4.io/master/en/setup/vscode.html

3.4 Building and uploading the code

1. On a terminal window, Navigate to the folder "px4_sem_project"
2. Physically plug the Pixhawk 4 via USB to the computer
3. Run the command:

make px4_fmu-v4_default upload

A successful upload will end with:

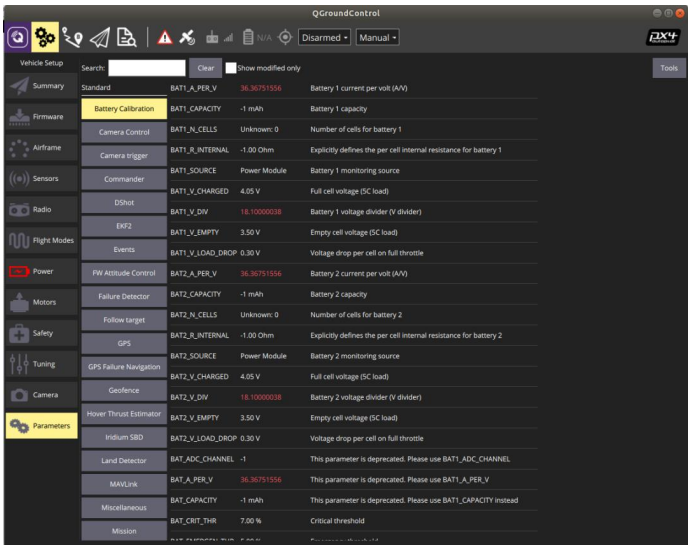
```
Erase  : [====================] 100.0%
Program: [====================] 100.0%
Verify : [====================] 100.0%
Rebooting.

[100%] Built target upload
```

3.5 Setting PX4 parameters (Necessary)

This section describes how to setup PX4 parameters to enable DShot esc communication protocol (Pixhawk 4 to ESC) and enable arming without safety checks (So that motors can be run in an experimental setup without a full UAV setup).

1. Open QGroundControl. And Navigate to the parameters setup window as shown below:

   Here you can search the parameters that will need to be changed using the "Search Window"



2. Search for the following Parameters and set them to the values as per the following tables:

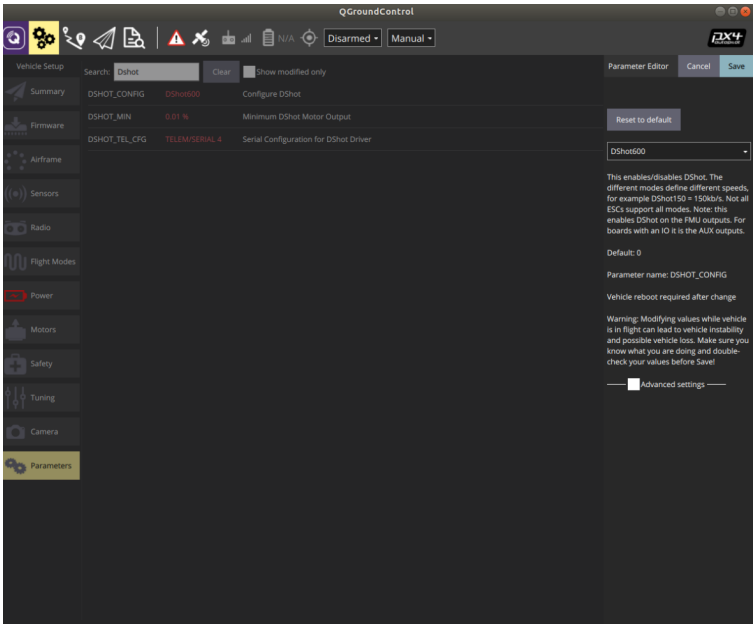
   To Enable Dshot:

Parameter	Value	Description
DSHOT_CONFIG	Dshot600 (Or other DShot options)	Enables DShot protocol. Either of 150, 300, 600 or 1200 can be used. Staying with intermediate speed as all of them can handle more than 800Hz command sending.
DSHOT_MIN	0.01	Minimum arm throttle as fraction.
DSHOT_TEL_CFG	TELEM/SERIAL 4	Directs the “UART & I2C B” pixhawk RX port to be used to receive ESC telemetry readings
MOT_POLE_COUNT	14	Used for RPM calculation. The current motors have 14 poles.
SYS_USE_IO	0	Directs output to FMU PWM OUT (As that port is directly connected to MC)

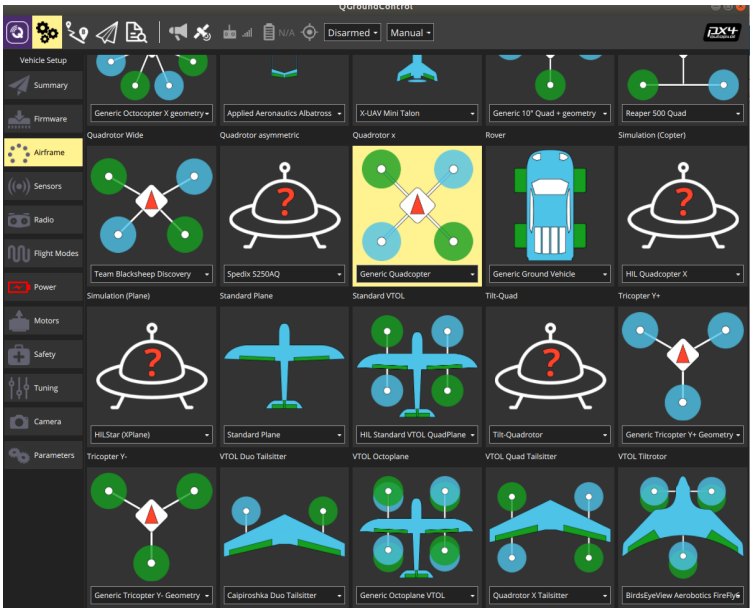
   To Disable safety checks when arming:

Parameter	Value	Description
CBRK_IO_SAFETY	22027	Disables IO safety
CBRK_USB_CHK	197848	Disables USB connected check
COM_ARM_CHK_ESCS	0	Disable ESC check for arming
COM_DISARM_PRFLT	0	Enables running motors pre-flight
COM_RC_IN_MODE	Joystick/No RC checks	Chooses to take inputs from joystick , hence enabling arming without RC

When selecting any parameter a sub-window will appear on the right where you can select the value desired. Then, need click Save.



3. Some of the mentioned parameters require System reboot. There after the setting all the parameters, reboot the pixhawk 4 by clicking: Parameters/Tools/Reboot Vehicle/OK.
4. Then check that the parameter changes have taken effect.
5. Select the Airframe to be a “Generic Quadcopter” and don’t forget to click “Apply and Restart” on the top right of the Airframe window



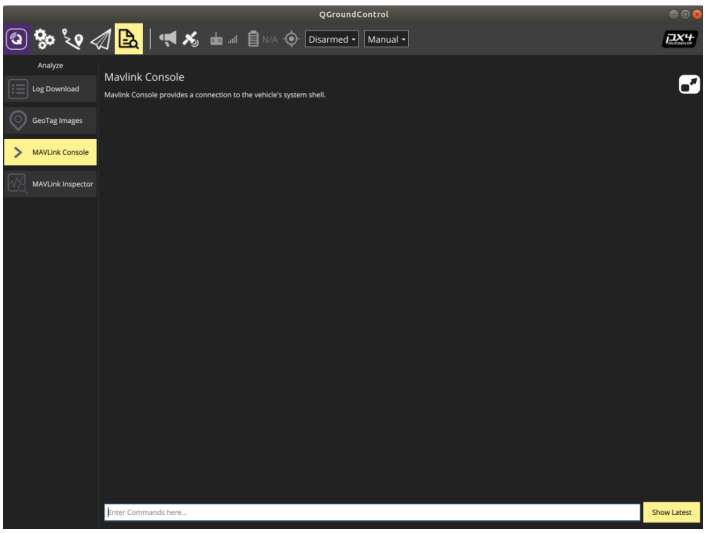
6. Wait for the Pixhawk 4 to restart and the setup is finished

3.6 DShot controller usage:

PX4 is setup in a modular structure. The module of interest is called: “dshot_controller”

First, you need to have access to the Nutshell, to interact with the pixhawk. Make sure that the pixhawk is connected to your computer via USB.

On QGroundControl, you can get access to the interactive Nutshell.



3.6.1 Startup procedure (NECESARY)

This needs to be performed to disable signal publishing conflicts with other modules. It must be done every time the pixhawk is restarted.

Perform the following commands on the Nutshell

1. mc_rate_control stop
2. dshot_controller start

3.6.2 Arming and disarming command:

This command will START SPINNING the motors at the minimum speed, so be CAREFUL:

commander arm -f

When you need to stop the motors, you can use the disarming command:

commander disarm

If everything run worked during setup, the nutshell should have given the following outputs

```
NuttShell (NSH)
nsh> mc_rate_control stop
nsh>
dshot_controller start
INFO  [dshot_controller] Instance Initialized
nsh>
commander arm -f
nsh>
```

Additionally, you can check the arming status with the command:

commander status

3.6.3 Using dshot_controller commands

If you type the module name (dshot_controller) in the nutshell it will show the available commands with a respective explanation. As below

```
NuttShell (NSH)
nsh> dshot_controller
Usage: dshot_controller <command> [arguments...]
 Commands:

   start

   stop

   status        print status info

   arm           Sends arming command to actuator_armed topic

   disarm        Sends disarming command to actuator_armed topic

   throttle      Prints current throttle being sent

   command       Sends throttle command to specified motor (others 0)
     [-m <val>]  Motor selected (1-8)
                 default: 1
     -p <val>    Throttle value (0-1)
     [-a]        Sets all motors to given power values

   listen_esc    Shows a number of esc datasets retrieved
     [-n <val>]  Number of datasets to receive (1-100)
                 default: 1

   run_sine      Outputs a Sinusoid to a selected motor
     [-m <val>]  Motor selected (1-8)
                 default: 1
     [-p <val>]  Period in ms (10-10000)
                 default: 2000
     [-d <val>]  Duration in s (2-20)
                 default: 10
     [-o <val>]  Wave Offset (0.01-0.9)
                 default: 0.5
     [-a <val>]  Amplitude (0-0.5)
                 default: 0.1
nsh>
```

The commands “arm” and “disarm” in the “dshot_controller” module will only be called instantaneously. They are only used for debugging (without commander module). Hence, you do not need to and should NOT use them.

Hence the way to call them in the nutshell is as follows:

dshot_controller <command> <option 1> <val1> <option 2> <val2> <option 3>…

Note that options with square brackets in the description are optional (I.e. if not given they will take a default value or behavior)

Appendix C

BlHeli Suite Instructions