Development & Construction of an Autonomous Path-Following Drone

Eduard Scherer

January 1, 2025

Contents

1 Introduction

In the world of today where technology is advancing faster than ever before this Matura Paper tries to shine a light into the corner of self-built and autonomous drones.

However, it first needs to be defined what a drone even is. According to the Cambridge Dictionary, a drone is defined as: *an aircraft that does not have a pilot but is controlled by someone on the ground* [17]. In this work, the term 'drone' will refer to a quadcopter which is a drone with four motors that lifts off and flies through the power provided by the motors and does not glide like a plane. There is also the term unmanned aerial vehicle (UAV), however this is mostly used to describe military grade drones.

In current conflicts the meanings of UAV and quadcopters have become more entangled than ever. The typical hobbyist drone is now widely and effectively used to target enemy forces. In the Ukraine war drones are one of the most effective weapons to use against the Russian aggressor. For example they use drones with thermite canisters attached to them to target the enemies under tree covers [22]. Since the war began the drones have become increasingly autonomous. Drone pilots with more advanced setups, are able to simply lock onto the target and let the drone follow and detonate near it [20]. This increased use in war has also had an impact on the hobby side of the project. The most used electronic speed controller (ESC) firmware BLHeli_32 ceased its operation due to laws being passed in Norway that penalised firms that are not able to verify that their products are not used in wars [19]. Which is really difficult to do regarding that both sides in the Ukraine war are buying the ESCs no matter what software is on them.

The goal of this Matura Paper is to develop an autonomous path-following drone; by path-following a global positioning system (GPS) path is meant not one on a map. Additionally it should be written so that somebody with no prior knowledge about drones can read, easily understand, and also might reproduce the first part until Section 2.5 (Companion Computer). After that first part, having some understanding of python will be useful, but it is not a requirement. However, without such knowledge, the explanations might be a bit confusing.

I was first interested in drones when was in fifth grade and got one as a Christmas present. After that, I also wrote a rather interesting piece about different types of drones in the sixth grade and then I lost interest. Later on, when I was in the "Gymnasium", I was randomly recommended a drone video on YouTube and began gaining interest again. I initially planned to build a first person view (FPV) drone. However, I never really found the time to build one. When the Matura Work came around I found it to be the perfect opportunity to work with a drone, although not just building a FPV drone, but doing something more complex.

1.1 General Software Considerations

1.1.1 Open Source Software

An important part of this Matura Work is open source software (OSS). OSS is a software which everybody can download and develop in their own time. This process is known as 'forking', which refers to when a person other than the original developer takes the code and modifies it independently. The main benefit over closed software is that it can be freely downloaded by everyone and everybody can modify it as they see fit. However the downside is that if the development team loses interest, the software will become outdated and will not work smoothly with other systems anymore. Both aspects will later be found both useful and time-consuming.

1.1.2 Flight Softwares

Based on the literature reviewed there are three main softwares to consider when it comes to drones Betaflight, INAV and Ardupilot. All of them are open source. Multiwii is the origin of Betaflight and INAV. Multwii was Arduino based and then upgraded to Baseflight to be able to utilize a better chipset. Then it was forked to Cleanflight, which was later forked again into Betaflight and INAV [6].

The following overview of the three softwares, is mostly based on the article written by Liang [11] and a video made by Schofield [12].

Betaflight appears to be the go-to option for FPV, commonly used for filming or racing. It is the most beginner friendly out of the three, because it has a large community, which results in a wide range of tutorials. When a new flight controller (Fc) comes out it is normally made to be used with Betaflight. However, Betaflight lacks support for different types of vehicles and generally the automated features are less developed compared to the other two.

INAV offers basic autonomous flight using waypoints and automated landing. It does not only support quadcopters, but also boats, rovers, planes and wings. It has a similar interface to Betaflight, so switching from one to the other is easier than switching from INAV to Ardupilot.

Ardupilot basically offers everything the other two have to offer and more, for example Submarines and vertical takeoff and landings (VTOLs). Although it is not commonly used with FPV, it is still possible. Ardupilot involves more steps to set up, which also leads to the main benefit of it, namely the nearly limitless customization options.

It is also the only option to use together with a companion computer like the RaspberryPi. A few years ago it was really expensive to start, because it only supported the Pixhawk family of flight controllers which cost several hundred Franks per piece. However, in recent years it has began to support more and cheaper flight controllers.
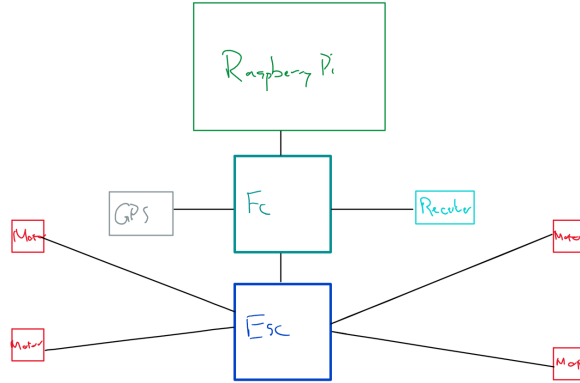
Figure 1: connection between the parts

2 Methodology

2.1 Drone Overview

If you want to fly a drone you have two options available: either you buy one or you build one. From a technical viewpoint, the second option is, not only the better option but also the cheaper one.

The main parts needed for a drone are the Fc, the ESC, the receiver, the battery, servos and of course the mainframe. Additionally a GPS and LEDs can be added as needed. The Fc and the ESC are sometimes more generally referred to as printed circuit boards (PCBs)[1] or simply 'boards'.

The Fc runs the chosen software and controls every other part in one way or another. It is directly connected to the ESC, which controls the motors and is connected to the battery. The Fc also communicates with the receiver and, if present, the GPS. In this case there will also be a microcomputer, I chose the RaspberryPi, which connects to the Fc. The connections between the parts are shown in Figure 1.

2.2 Data Transfer Protocols

There are two types of data transfer protocols: serial and parallel. The way they transfer data is by pulsing 5V for a 1 and 0V for a 0. The difference is that the parallel communication sends the data for each bit parallel to the other component through eight connections, as seen in Figure 2, and the serial sends the data over one connection. This also requires a clock to synchronize the

---

[1] A PCB is a flat board that houses electronic components and features conductive pathways etched onto its surface to connect those components.
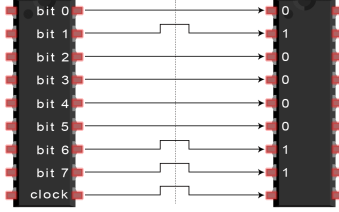
output of data.

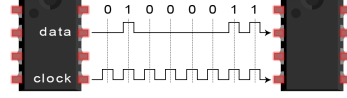Figure 2: parallel communication of the letter "C" [33]

Figure 3: serial communication of the letter "C" [33]

2.2.1 Uart

The universal asynchronous receiver/Transmitter (UART) is not a communication protocol, but a physical circuit. In the field of self-built drones it is used to communicate to and from the Fc. It works by transmitting data asynchronously, which means without a clock to synchronize the data transfer. Instead it has a so-called baud rate, which is a measure of bits per second (bps). When connecting two UART microcontrollers there are two lines needed from the Tx of one to the Rx of the other and vice versa, as shown in Figure 4. With a UART a data packet is transferred instead of individual bits. The packet consists of a start bit, followed by 5 to 9 data bits and at the end there are 1 to 2 stop bits, as shown in Figure 5. There is also the option to include a parity bit instead of one of the data bits. A parity bit is used to verify that none of the bits were changed during the transfer. It is a 0 if the number of 1 in the sequence is even and 1 if it is odd. [34]
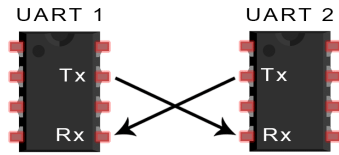
Figure 4: UART connection [34]
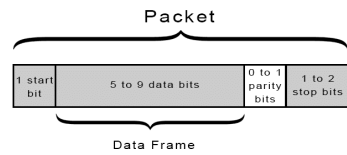
Figure 5: UART package [34]

2.2.2 I2C

The inter-integrated circuit (I2C) communication is also serial like UART, but it has the advantage of being able to communicate to more than one slave[2]

---

[2]The relationship between two parts where one controls the other is called a master-slave relationship. Obviously the master is the one that controls the slave. Although the terminology is rather controversial, but still widely used [4].

part. In this work, it is used for the compass built into the GPS. There are two lines between the master and the slave; the serial data (SDA) and the serial clock (SCL). The SCL line is used for synchronization and the SDA line for the transfer of a packet. The data packet looks quite different to the one from the UART. There is a start condition, afterwards the address frame is used to figure out to which slave the packet goes, followed by a bit to determine if the master sends or requests data, afterwards the data is sent followed by a ACK/NACK bit, which confirms that the data frame was received and at the end there is a stop condition [35].
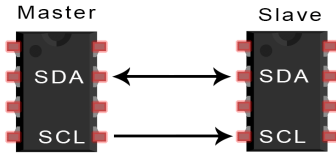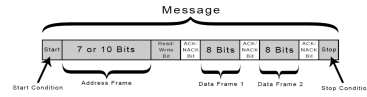

Figure 6: I2C connection [35]

Figure 7: I2C package [35]

2.2.3 MAVLink

The MAVLink protocol is more complex compared to UART or I2C. It can transfer up to 263 bytes or 280 bytes, depending on the version used. It is used for the connection between the Fc and a ground control station (GCS) or in this case the RaspberryPi companion computer.

2.3 Parts

2.3.1 Flight Controller

There are two different types of flight controllers: the All-in-One (AIO) and a standalone Fc. The AIO consists not only a Fc but also the ESC on the same board. This has the advantage of only needing one board instead of two or five, in some alternative setups. However, if only parts of the ESC or the Fc are damaged you need to replace the whole board, which is more expensive than replacing only the Fc or ESC.

Betaflight and INAV both support a wide variety of Fcs compared to Ardupilot which is only supporting a very specific sample of boards [36]. They have the option of open and closed hardware. Because the open hardware Fcs are quite expensive, so I decided to go with a closed one. The chip used in Fcs is usually a STM32. There are multiple generations of it the mainly used ones are F4, F7 and H7, the newest version is the H7. I then decided to go with the Kakute H7 v1.3 (MPU6000) from Holybro, because it was affordable and available as a stack [39]. A stack is a Fc and ESC mounted on top of each other. It normally comes with stack screws. It comes shipped with Betaflight so it is required to flash Ardupilot.

7

2.3.2   Electronic Speed Controler

There are two different kind of ESCs: 4in1 and single ESCs. If you use single
ESCs, then one is needed for each motor instead of a single board for all of them.
The advantage of 4in1 ESCs is that it does not require a power distribution
board, because it is already incorporated in the ESC, and that it can come in
a stack. The disadvantage is that if a part of the ESC is damaged you need to
replace the whole board. What needs to be considered before buying a ESC is
that the peak current of the motor is not higher than the burst current of the
ESC, because a too high current could damage the ESC.

My decision was to go with a 4in1 on a stack, because it is slightly cheaper and
normally easier to wire compared to four single boards. The only option with
the Kakute H7 was the stack with the Tekko32 4in1 with a continuous current
of either 50A, 60A or 65A. I chose the 50A one, because you do not need a high
continuous current rating when flying rather slowly and the motors I chose have
a peak current of around 42A [45].

At the time I bought the Tekko32, it was still shipped with BLHeli32, which
as mentioned in the beginning has since seized their operations. It is however
possible to flash AM32[3] onto it [15].

2.3.3   Motor

There are two types of motor: brushed and brushless ones. The difference
between the two types is that the brushed motors are mechanically driven,
while the brushless motors are electrically driven. As a result, brushless motors
also need an ESC to function compared to brushed ones. Brushed motors are
used in very small drones with 1S[4] batteries. However, even in smaller drones
brushless motors are the more popular choice [2]. A brushed motor, as shown on
the left in Figure 8, works by having two brushes that are delivering the power
to a coil in the middle of a permanent magnet. This creates a magnetic field
around the coiled, which is then attracted to the magnet surrounding it. Due to
the commutator the direction in which the provided electricity flows is changed
every half turn. This causes the magnetic field around the coil to change the
direction and in turn gets attracted to the other side of the magnet. A brushless
motor, as shown in Figure 8 on the right, works by having a field magnet in

---

[3]A different ESC software, which has gained quite some popularity after BLHeli32 is off
the market.

[4]What 1S means will be explained in the next section

the middle and coils surrounding it. The coils surrounding the magnet are part of the so-called stator. These coils are provided with a positive and negative voltage, which will cause the magnetic field around them to alternate. When the alternation is synchronized, which is done by an ESC, it will cause the field magnet to spin.
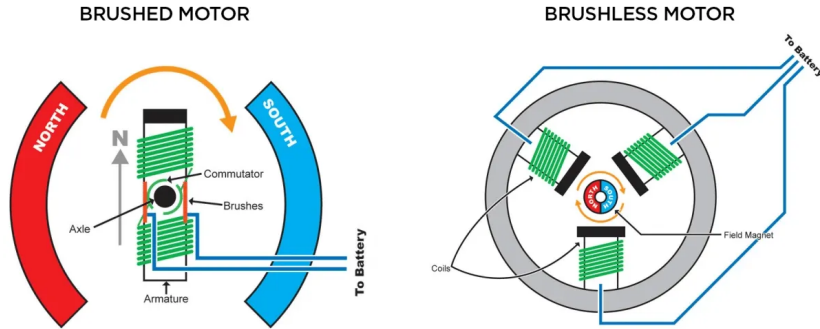

Figure 8: brushed and brushless motors [8]

The numbers that are seen on the motors, such as '2207', describe the stator of the motor itself. In the case of a 2207 that would be a 22mm diameter and a height of 7mm. The usual stator sizes of 5 inch drones are either 2207 or 2306. There is also the KV value, which has to be considered. The KV value is the number of revolutions per minute (rpm) a motor turns when one volt is applied. The lower the KV value, the more efficient the motor is and the higher the KV value the more responsive it is. The KV value for a 5-inch drone with a 4S battery ranges from 2300 to 2800. I chose the Iflight Xing-E Pro 2207[48] with a KV value of 2450, because the IFlight-Xing motors are known to be of good quality, *'they (Xing motors) also prove to be very reliable and most importantly durable'* Landa [5].

2.3.4 Battery

There are two main types of batteries: lithium polymere (LiPo) and lithium-ion (Li-ion) batteries. LiPo batteries have a tendency to go up in flames. They have a much higher discharge rate compared to Li-ion batteries, also denoted as the C value, so they are well suited for racing drones. However, Li-ion batteries have a higher energy density, which means that they can store more mAh for the same weight compared to LiPo batteries, so they are more common in long-range flying. The batteries normally have multiple cells, for example a 4S LiPo battery contains 4 cells. This is important, because the more cells you have, the higher is the voltage and so you will need a motor with a lower KV value.

I first wanted to buy two Li-ion battery packs, however they are hard to get

and much more expensive. The other option would be to solder them together, but it requires some soldering skill, which I personally did not have then. Hence I decided to go with 4S LiPo batteries from the Tattu R-line with 120C and 2000mAh[44], because I wanted something that can fly longer than just three or four minutes. I chose the brand Tattu, because it was the only known brand that was on AliExpress, from where I also sourced all the other components, so it seemed to be easier to also chose a battery from there.

2.3.5   GNSS(Global Navigation Satellite System)/Compass

There are two different kinds of global navigation satellite system (GNSS)[5] modules: the normal GPS and compass boards and real time kinematic (RTK) GPS. Usually the chips used in both the RTK GPS and the normal GPS are manufactured by the company Ublox. The RTK GPS can achieve an accuracy of 1cm by incorporating information correction data from a radio technical commission for maritime (RTCM)[6]. However, the correction data is either subscription-based, not guaranteed to cover all of the area or one needs to build one by themselves, which is quite complicated [42]. For a small drone it may not be worth it to have a RTK GPS that costs several hundred Swiss Franks instead of a GPS with a 2m circular error probable (CEP) which costs less than 30 Swiss Franks. Some of the GPS units also have a compass built-in, so you will not need to buy an extra compass when using ArduPilot.

> **Explanation 2.2: circular error probable (CEP)**
>
> The CEP refers to how close to the real value the GPS normally is. So if a GPS has a CEP of 2m it is normally in the range of 2 meters of the correct value [14].

I chose the Holybro Micro M10 GPS, because it is produced by the same brand as the Fc and therefore seemed to be easier to connect [40]. In addition, the M10 chip is the newest version of Ublox chips and it would be pointless to buy an older version at the same price. It also is at least as good as other better-known GPS as the Matek M10Q [21] and comes with a built-in compass that is needed for ArduPilot.

2.3.6   Radio/Transmitter

The following information about transmitter protocols is based on two YouTube videos from Bardwell [9] and Schofield [23].

The two things to consider when it comes to transmitter protocols are latency and range. Latency is the delay between the input on the radio controller and when the reaction of the Fc. The lower the latency, the faster the drone

---

[5]The GNSS is usually referred to as GPS even though GPS is only the American GNSS system

[6]It was first used for the positions of boats and other vessels

will react. It is related to the frequency bands of the receiver, of which there are normally two: 900 MHz and 2.4 GHz. When both are optimized the 900 MHz has better penetration and range, due to the longer wavelength, while the 2.4 GHz has a higher latency, because the frequency allows for faster data transmission.

ExpressLRS

ExpressLRS is the best protocol in long range and for the combination of long range and low latency. It has been shown by Varty and Neal [13][7] that it really can fly up to 100km away from the starting point. There are two version: a 900 MHz and a 2.4 GHz. The main difference is that the 2.4 GHz can reach over 30km, but it is unlikely to go to 100km. However, it has the lowest latency. It is together with mLRS, is the only protocol that is open-source.

mLRS

The main difference between ExpressLRS and mLRS is that mLRS has a higher latency, which allows it to send larger data packages to your telemetry device. Which is something that is needed if you want to adjust something or get more data from your drone over the MAVLink protocol during the flight.

Transmitter/Radio

I decided to go with the Radiomaster RP4TD ExpressLRS 2.4GHz True Diversity Receiver based on a recommendation of a friend [25]. It is also compatible with mLRS if I want to switch later on.

I chose with the Radiomaster Boxer[26] radio, because it is somewhat in the middle range from radios and seems to be quite reliable and has many switches to assign flight modes or other functions to.

2.3.7   Smoke Stopper

A smoke stopper is a device that prevents the ESC from short-circuiting due to incorrectly soldered parts and can save quite a lot of money. There are two groups of smoke stoppers one that you buy and get destroyed when the ESC short circuits instead of the ESC. There is another category that does not destroy itself and there are also some that you can solder together on your own[1].

2.3.8   Problems with the Parts

There were mainly two problems that arose during the process of assembling the drone. One less severe than the other. One problem was that my ordered

---

[7]Due to him being fined by the Australian government he took down his own video, so only a copy from an other YouTube channel exists.

Kakute-Tekko stack did not include stack screws, which they normally do. Stack screws are just screws that you can use to mount the stack onto the frame. So I needed to purchase them separately, which was rather tedious and time-consuming.

The more severe problem I ran into was that the batteries from AliExpress were first withheld by the Swiss border control and then I received two insect traps instead of batteries. Luckily I ordered one of the same batteries from another online shop (Conrad), because the other two took too long to deliver.

Two months later, when my father decided to open the insect traps. To our shock, the LiPo batteries were inside the insect traps at the bottom. And in hindsight it was also obvious that they were in there, because it had the numbering 3 4s 2000 on top which stands for version 3 of the 4s batteries with 2000 mAh. However, we dismissed the numbering on top as just some random numbers put there.

2.3.9 Propellers and Battery Charger

For the propellers and battery chargers, I went off the recommendations from AOS-RC and FPVknowitall. For the propeller, I chose the Foxeer Donut 5145 [27] and the HQ 5x4.3x3 V1S [28]. For the battery charger, I went with the cheapest option the 608 AC Lipo Battery Charger [29].

2.4 ArduPilot

This chapter summarizes everything done for the first time flying and what could go wrong based on my experience. It is based on the ArduPilot copter documentation [31].

2.4.1 Ground Station

To configure ArduPilot, there are multiple softwares, so called GCS required. They are normally ground-based and can transmit data via wireless telemetry device or USB cable. With the telemetry device, they can also control the drone from the ground and alter the route as the drone is autonomously flying.

The most widely used GCS is Mission Planner (MP) it is widely used, but runs only on Windows and Mac OS. It also has a wiki, which was used for the first-time configuration of my drone [41].

Another GCS is MavProxy it is based on Python and is only for the Linux operating system (OS). Which is the OS that will be used on the RaspberryPi companion computer.

> **Explanation 2.3: operating system(OS)**
>
> A OS is a software, which controls all the hardware of a computer.

2.4.2 Firmware Installation

The following section until 4.5 Companion Computer is rather technical and contains the necessary information for the steps to reproduce my setup.

For the first time installation the Kakute H7 is required to be flashed with ArduPilot, because it is shipped with Betaflight.

> **Explanation 2.4: to flash**
>
> Flashing is the process of taking new firmware and loading it onto the device the firmware is needed on.

To install the ArduPilot firmware for the Kakute H7 it needs to be downloaded onto a computer [32]. Afterwards the STM32CubeProgrammer is used to flash the firmware onto the Fc [43]. The Fc in device firmware updgrad (DFU) mode is directly connected with the computer using a USB cable. Then the USB port, with which the Fc is connected, select and firmware is flashed onto the Fc . A reboot is required to leave the DFU mode, before connecting the Fc to MP. The progress of flashing the firmware was straightforward, unlike for the rest of the configuration.

> **Explanation 2.5: device firmware upgrade (DFU)**
>
> The DFU mode is the mode, which allows the user to upload new firmware to the Fc. It is normally accessible through a button which needs to be pressed, while connecting the battery.

2.4.3 GPS Connection

In the beginning no immediate GPS connection appeared. Even changing the parameter `GPS_Type = 2` for the Ublox GPS, the 'No GPS' error, as seen in the bottom right corner of Figure 9, was still there. Even though it was clear that the GPS worked and was connected to the Fc, because the compass appeared in the compass calibration tab on MP.

The issue could be that the GPS is too close to metal surfaces, the computer, which is connected via USB cable, is too close, or that soldering was done poorly and the cables from the GPS to the Fc are falsely connected. After testing for each of the possible issues and mitigating the proximity in which metal was near the drone, the GPS was still not working. The GPS itself was also not damaged, because the blue led was blinking constantly which means it is connecting to a GNSS.

The problem was that the `Serial3_Protocol` (which is for the Uart3, which will be used for the connection to the RaspberryPi) was set to 5 which stands for GPS. However, this blocked the Uart4 from being received as a GPS, to which the GPS was really connected. After disabling the Uart3 it finally worked.
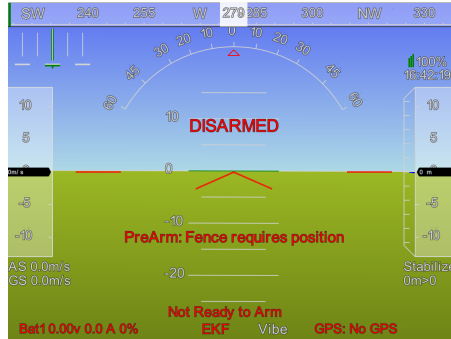
Figure 9: No GPS connection
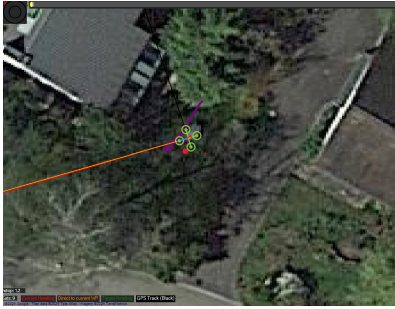
The GPS is quite precise outside Figure 10[8].

Figure 10: GPS outside

2.4.4 Receiver/Transmitter

To get the connection between the receiver and radio the ExpressLRS page needs to be followed [30]. It required changing the `Serial6_Protocol` to 23 and the `RSSI_Type` to 3 for the receiver protocol. I also changed the `RC_Options` to the correct bitmask (Figure 11). The connection between the radio and receiver seemed to be working, because the receiver had a constant blue light and the receiver notified that the telemetry recovered after it was lost for a moment. However, it did not yet have a connection to the Fc. There was the possibility to change the Uart6 ot the Uart1, which is usually the Uart used for receivers with the Kakute H7, but it would require a JST (a special connector) and more soldering. The solution was found in the Kakute H7 tab in the ArduPilot documentation, where it was listed that for a CRSF[9] interface the parameter `Brd_Alt_Config` needs to be set to 1. `Brd_Alt_Config` is a

---

[8]It also works sometimes inside and is precisely on my room, but it can also show that it is in Poland the middle of the Atlantic Ocean or Iceland.

[9]Even though it is a ELRS receiver it has the same interface as CRSF receivers
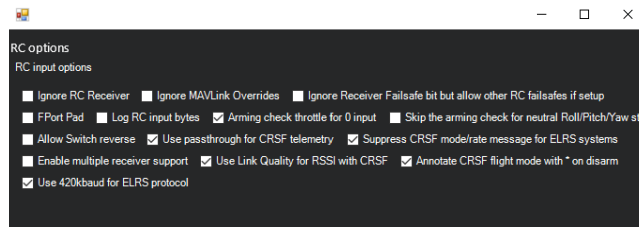
Figure 11: correct bitmask for ExpressLRS

Fc specific parameter, this is also the reason why it did not come up in the ExpressLRS page or the ArduPilot documentation.

2.4.5 Battery

The battery was rather straightforward. The smoke stopper did not light up when plugging it in. In the beginning the battery was not recognized by the Fc, but it was solved by changing the parameter `Batt_Monitor` to 4. After that the voltage and amperage shows up in MP.

2.4.6 Motor Test

The motor test works correct after assigning the right position to the motors, because I'm using the M5-M8 ports instead of the M1-M4. The parameter `Mot_PWM_Type` needs to be set to dshot 600. The motor testing works after assigning the correct motor to the correct output in the servo output tab in MP. Dshot is the digital protocol for communication between the Fc and ESC. The motors began to beep after a certain time, due to the beacon delay. This will be turned of later. This was also the first time the ESC had power and through that a new error appeared called 'battery failsafe'. This is caused by an unstable connection between the ESC and the Fc, which causes the Fc takes the computer as power source and has too little power to spin the motors.

2.4.7 Compass Calibration

The compass calibration needs a good GPS lock. However, even with a good lock, relaxed fitness[10], and `Compass_Orient` set to 6, as recommended by the Holybro docs, it still failed [38]. Large Metal parts could again influence the calibration, but there were none in the vicinity of the compass. To temporarily do the calibration the large vehicle MagCal can be done. However, this mostly takes away the prearm message. It is strongly discouraged by the ArduPilot documentation, because it can look as if it is correctly configured, but the orientation is incorrect. The best way to calibrate the compass is to put it directly on top of the Fc and then do the calibration.

---

[10]The fitness can be in four different states very strict, strict, default and relaxed. The stricter the fitness is the longer the calibration takes.

2.4.8   Road to First Flight

The first step to flying is to arm the drone.

> **Explanation 2.6: to arm**
>
> To arm a drone means that the motors begin to spin without producing enough thrust to lift the drone from the ground. It is used when you want to fly before the actual flight.

For that a switch on the radio needs to be assigned to arming and disarming. In my configuration the switch five is used. In order for it to work the parameter `RC5_Option` is set to 153, which means to arm the drone when the switch number 5 is flicked. To test the drone on the bench inside there will be many prearm errors to ignore the parameter `Arming_Check` is disabled and more importantly the geofence, which still blocks the arming when `Arming_Check` is disabled. After this when flicking the switch 5 the motors will arm and spin[11]. From my experience it is really important to tie down every cable or antenna before arming the drone with propellers, else they might just be cut through or teared off. This will cause a new prearm error to appear called 'crashdump bin detected'. This file can be used to analyze the crashes your drone has had. As long as it is tested on the bench, it should not be a problem and can be deleted by flashing new firmware onto the drone [18]. This time however not via STM32CubeProgrammer, but directly over MP. Additionally the some of the motors need to be reversed for the X-configuration of the drone, seen in Figure 12. To reverse them the reverse button in MP does not work. The BL-HeliSuite32 software is needed and the parameter `Servo_BLh_Auto` is set to 1 which enables a pass through from the ESC through the Fc. In the software one can change the motor spinning direction. It is also advisable to turn off beacon delay, at least for now, or else the motors will beep after ten minutes of being idle.

> **Explanation 2.7: beacon delay**
>
> The beeping of the beacon delay is used to located a crashed drone. If the drone has lost the contact to the radio it will automatically activate after a certain amount of time.

In my first test flight the drone was shaking violently, which is also known as wobbling. This caused me to crash it into the ground. Reasons for this behavior can be loosely fastened parts, imbalanced propellers, or also wrong proportional, integral, derivative (PID) values [7]. In my case it was the PID values, which were for a 9-inch drone and not a 5-inch. The PID values can be adjusted via the initial tuning parameter. After this the drone flew fine, except that the sticks of the radio controller needed to be reversed. This can be done using the

---
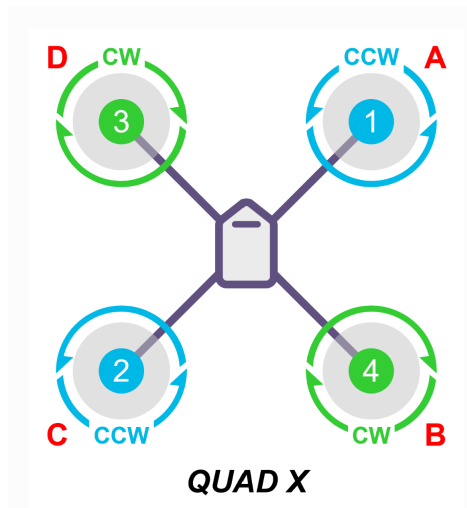
[11]A battery needs to be connected.

Figure 12: Motor turning directions

`RC2_Reversed` parameter.

> **Explanation 2.8: proportional, integral, derivative (PID)**
>
> The PID refers to three variables who control the error adjusting system in a drone. With error is meant the difference between how much the motors rotate and how much they should rotate.

2.5   Companion Computer

The following section is rather technical and can be used as a reference to reproduce the combination of the Kakute H7 and a RaspberryPi working together using Dronekit python.

2.5.1   RaspberryPi Setup

To let the drone fly on its own a companion computer is needed, because the Fc does not have enough power to process more complex operations such as autonomous flight.

Choosing the RaspberryPi instead of another companion computer is advisable, due to it being the most popular in drone projects.

The RaspberryPi is not shipped with an microSD card and through that also not with the OS. To download the OS, the RaspberryPi Imager is used. Through the imager it is possible to create an account for the RaspberryPi, configure the WiFi and the possibility for a secure shell (SSH) connection.

17

To see a desktop instead of only a command line based interface with an SSH a virtual network computing (VNC) is used. In this case the recommended option was RealVNC viewer a widely used VNC. For this RealVNC server needs to be installed over the terminal via `sudo apt-get realvnc-vnc-server` [10][12].

2.5.2 MAVProxy Installation

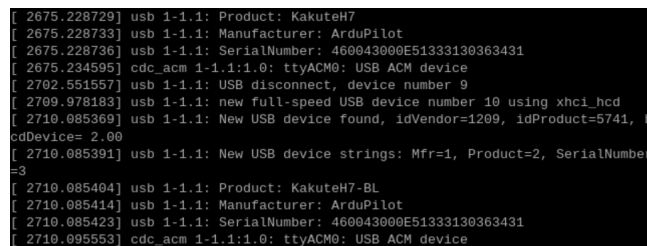The following was based on the MAVProxy Documentation from the ArduPilot website [46].

First all the needed packages need to be installed through the `sudo apt-get install` command.

```
1 sudo apt-get install python3-dev python3-opencv python3-wxgtk4
    .0 python3-pip python3-matplotlib python3-lxml python3-
    pygame
2 pip3 install PyYAML
```

What is not said in the MAVProxy Documentation, is that you need a virtual environment (venv) and cannot download it without one or the error; `error: externally-managed-environment` pops up. To get around this issue you need to create and activate a venv.

```
1 python3 -m venv mavproxy-env
2 source mavproxy-env/bin/activate
3 pip install MAVProxy
```

To connect the drone over MAVProxy it is needed to know how the Fc is connected to the RaspberryPi. The command `dmesg | tail` will provide the information, as seen in Figure 13.

Figure 13: output from `dmesg | tail`

The command `mavproxy.py --master=/dev/ttyACM0 --baudrate 115200 -- aircraft MyCopter`, is used to connect over MavProxy, an error will appear if \ttyUSB0 instead of \ttyACM0 is utilized. If the RaspberryPi is connected over the serial ports, \serial0 is used instead of \ttyACM0. The baudrate can also

---

[12]`realvnc-vnc-viewer` is only required if you want to see a VNC from the RaspberryPi

vary, in this case it is 921600. The RaspberryPi will not have another power source except over the Fc, the 5+ V pin on the RaspberryPi needs to be connected to a 5 volt pin on the Fc and additionally a ground pin. There will be a low voltage warning, but the RaspberryPi works fine. In addition the `Serial3_Protocol` needs to be changed to 2 for the MAVLink protocol.

When connected to the Fc you can use simple simple commands to change the parameters, as `param set arming_check 0`, or to arm the copter with `arm throttle`.

2.5.3 Dronekit

It first needs to be mentioned that the Dronekit python software is not maintained very well, as it is stated in the Github repository [37]. The following is still mostly based on the Dronekit Documentation [47].

First the Dronekit library needs to be installed in the venv with `pip install dronekit`. To create a file in the venv over the terminal `nano dronekittest.py` is used. It is noteworthy to not use the same name as the library itself, because python will confuse it. In the created document one then can connect the RaspberryPi to the Fc as can be seen in Listing 1.

```
1  from dronekit import connect
2
3  vehicle = connect('/dev/serial0', baud=912600, wait_ready=True)
```
Listing 1: Python DroneKit Example

However this will cause an attribute error, as one can see in Listing 2

```
1  dronekit/__init__.py" , line 2689, in <module>
2  class Parameters(collections.MutableMapping, HasObservers):
3  ^^^^^^^^^^^^^^^^^^^^^^^^^^^
4  AttributeError: module 'collections' has no attribute '
     MutableMapping'
```
Listing 2: AttributeError after connecting

The source of the error is that in python 3.10 the abstract base class, MutapleMapping, was moved from collections to collections.abc. This needs to be changed (Listing 4) in the dronekit source code. Which can be accessed with the the command Listing 3[13].

```
1  nano /home/EduPi/mavproxy-env/lib/python3.11/site-packages/
     dronekit/__init__.py
```
Listing 3: accessing source code

---

[13]If it would not be open source, the change(Listing 4) would not be possible

```
1 class Parameters(collections.abc.MutableMapping, HasObservers):
```
Listing 4: changed line in source code (change marked in green)

After this the program worked flawlessly and it was able to give information from the Fc over the terminal. As shown with the example Listing 5, which

```
1 print( "Autopilot version: %s" %vehicle.version)
```
Listing 5: information retrieval

```
1 Autopilot version: APM:Copter-4.5.7
```
Listing 6: output from Listing 5

The main problem of the outdated Dronkit python library is, that the function `vehicle.channels`, which should read the channel values from the radio controller, is returning none instead of values between 900 and 2100. This is a long-known issue and to circumvent it a decorator is used [3].

> **Explanation 2.9: Decorator**
>
> A decorator is a function that modifies the behavior of a function or a class [16].

```
1  @vehicle.on_message("RC_CHANNELS")
2  def rc_channel_listener(vehicle, name, message):
3    global latest_rc_channels
4    latest_rc_channels = message
5
6  def get_rc_channel_value(channel_number):
7    global latest_rc_channels
8    if latest_rc_channels is None:
9      return None
10   channel_value = getattr(latest_rc_channels, f"chan{
       channel_number}_raw", None)
11   return channel_value
```
Listing 7: decorator for channel values

The decorator in Listing 7 is already predefined in the Dronekit library. When calling it with `@vehicle.on_message("RC_CHANNELS)` (line 1 Listing 7) and saving it to the global variable `latest_rc_channels` (line 4 Listing 7) as a dictionary (Listing 8).

```
1 chan{i}_raw : x
```
Listing 8: output from decorator in Listing 7[14]

---

[14]With i being the channel number and x the value the channel receives form the radiocontroler

20

The location is straight-forward. There is a home location that is set every time the drone is armed. In addition there is also local frame generated with it, which takes over the coordinates for north and south, but sets the height to 0 at the starting point. The home location could also be newly set via a MAVlink command. To access any given location local or global can be done over `vehicle.location.global_frame`[15]. This will have a list output `[longitude, latitude, altitude]`, to access the longitude a `.lon` is added after the `_frame` from before[16].

I personally added the `log()` function to be able to troubleshoot what went wrong in the testflight, can be seen in Listing 9. On line 2 the correct file is defined, which is then opened in append mode in line 3. Line 4 writes the given content into the file and line 5 closes the file again.

```python
def log(content):
    p = pathlib.Path(__file__).with_name('log.txt')
    o = p.open(mode="a")
    o.write("\n"+ str(datetime.now())+" " +str(content))
    o.close()
```

<div align="center">Listing 9: log function</div>

The desired coordinates for the drone to fly to are taken from a .tex file in the format `X;Y;Z` in meters and need to be converted into degrees. This is done by the code from Listing 10.

```python
earth_radius = 6378137.0

changeX = home_location.lat + math.degrees(float(X[i]) /
    earth_radius)
changeY = home_location.lon + math.degrees(float(Y[i]) / (
    earth_radius * math.cos(math.radians(home_location.lat))))
changeZ = home_location.alt + float(Z[i])
```

<div align="center">Listing 10: coordinate conversion</div>

To fly to a certain coordinate the `vehicle.simple_takeoff` and the `vehicle.simple_goto` functions are used. However, with the standard parameters from ArduPilot this will not work, because the parameters bitmask of `Brd_Safetyoption` is set to be active even when `Brd_safety_deflt` is deactivated. This will not let anything except defined in ??????? go through. Hence both of them need to be deactivated by the code, as demonstrated in Listing 11

```python
vehicle.parameters["BRD_SAFETYOPTION"] = 0
```

<div align="center">Listing 11: deactivation of parameters[17]</div>

converting the coordinates from the coordinates.txt to

---

[15]"vehicle" is the connection to the Fc from Listing 1

[16] `.lat` for the latitude and `.alt` for the altitude

[17]This can be done with any parameter, by putting it into the place of `Brd_safetyoption`.

trying to let the drone fly with the `vehicle.simple_takeoff(Zcoordinate)` command does not work

it worked miraculously after diabling the parameter `BRD_Safety_Deflt` and setting the bitmask to zero of the parameter `BRD_Safetyoption` this is used to enable or disable the outputs to motors and servos. there is also the parameter `BRD_Safety_Mask` which controls which switches can still get through when the safety switch is enabled... So the problem was that the `Brd_Safetyoption` was set to be activated when `Brd_safety_deflt` was disabled and enabled... so there was nothing except radiocontrol going through to the motors not my script.

hdop and vdop problems...

3 Results

4 Discussion and Outlook

5 Conclusion

Appendices

Bibliography

[1] Joshua Bardwell. *Why you need a smoke stopper — HOW TO MAKE A SMOKE STOPPER*. May 23, 2019. URL: https://www.youtube.com/watch?v=I5a0TAmEwLE (visited on 10/20/2024).

[2] Oscar Liang. *Brushed Motors vs Brushless Motors for Quadcopter*. Apr. 2, 2019. URL: https://oscarliang.com/brushed-vs-brushless-motor/ (visited on 10/19/2024).

[3] phil-toppe, Andre van Calster, and mrthomasbarnard. *Getting 'None' for all channel values #969*. Aug. 26, 2019. URL: https://github.com/dronekit/dronekit-python/issues/969 (visited on 12/18/2024).

[4] Tyler Charboneau. Oct. 6, 2020. URL: https://www.allaboutcircuits.com/news/how-master-slave-terminology-reexamined-in-electrical-engineering/ (visited on 12/27/2024).

[5] Chris Landa. May 9, 2020. URL: https://brushlesswhoop.com/iflight-xing-0802-motor-review/ (visited on 12/25/2024).

[6] Paweł Spychalski. *A brief history of a flight controller - From MultiWii to Betaflight and beyond*. Mar. 18, 2020. URL: https://quadmeup.com/a-brief-history-of-a-flight-controller-from-multiwii-to-betaflight-and-beyond/ (visited on 10/14/2024).

[7] Andrew Stapleton. Nov. 25, 2020. URL: https://www.youtube.com/watch?v=5c4BP7BlEAw (visited on 10/20/2024).

[8] Ben Winstanley. July 3, 2020. URL: https://www.haredataelectronics.co.uk/brushed-dc-motors-vs-brushless-dc-motors (visited on 12/27/2024).

[9] Joshua Bardwell. Apr. 7, 2021. URL: https://www.youtube.com/watch?v=a8cy5BK5SbU (visited on 10/20/2024).

[10] IONOS editorial team. *How to set up a VNC on your Raspberry Pi*. Nov. 22, 2022. URL: https://www.ionos.com/digitalguide/server/configuration/setting-up-virtual-network-computing-on-raspberry-pi/ (visited on 10/31/2024).

[11] Oscar Liang. *Flight Controller Firmware for FPV Drone: Choosing Between Betaflight, iNav, Ardupilot*. Feb. 23, 2023. URL: https://oscarliang.com/fc-firmware/ (visited on 10/18/2024).

[12] Lee Schofield. *Choosing between Betaflight. INAV and Ardupilot: A guide for new builders*. Mar. 11, 2023. URL: https://www.youtube.com/watch?v=Y12tAXnGptY&list=PLf77vRmH7VrGk0IP6TiCyleA9ws_ooGxY&index=26 (visited on 10/18/2024).

[13] Wezley Varty and Dax Neal. Aug. 29, 2023. URL: https://www.youtube.com/watch?app=desktop&v=CYJ2UOrlXgM (visited on 10/20/2024).

[14] Jack Wu. *The most detailed explanation of P-10 Pro accuracy*. Jan. 29, 2023. URL: https://gpswebshop.com/blogs/tech-support-by-vendors-columbus/what-does-the-p-10-pro-0-5m-cep50-and-1-5m-cep95-horizontal-accuracy-mean (visited on 11/22/2024).

[15] Sept. 7, 2024. URL: https://github.com/AlkaMotors/AM32-MultiRotor-ESC-firmware/wiki/List-of-Supported-Hardware (visited on 12/25/2024).

[16] June 19, 2024. URL: https://www.geeksforgeeks.org/decorators-in-python/ (visited on 12/23/2024).

[17] *drone*. Dec. 19, 2024. URL: https://dictionary.cambridge.org/dictionary/english/drone.

[18] Chris Edelen and Andras Schaffer. *Can't delete Crashdump data*. June 8, 2024. URL: https://discuss.ardupilot.org/t/cant-delete-crashdump-data/119045 (visited on 10/20/2024).

[19] Oscar Liang. June 3, 2024. URL: https://oscarliang.com/end-of-blheli_32/ (visited on 12/20/2024).

[20] Paul Mozur and Adam Satariano. July 2, 2024. URL: https://www.nytimes.com/2024/07/02/technology/ukraine-war-ai-weapons.html (visited on 12/19/2024).

[21] Andy Piper. Aug. 20, 2024. URL: https://discuss.ardupilot.org/t/the-great-gps-showdown-speed-precision-and-a-few-surprises/122661 (visited on 10/20/2024).

[22] Marc Santora. *Rise of the Dragons: Fire-Breathing Drones Duel in Ukraine*. Oct. 12, 2024. URL: https://www.nytimes.com/2024/10/12/world/europe/ukraine-russia-dragon-drones.html (visited on 12/19/2024).

[23] Lee Schofield. Oct. 13, 2024. URL: https://www.youtube.com/watch?v=E9JYwwWr0LU (visited on 10/20/2024).

[24] URL: https://opensource.com/resources/what-open-hardware (visited on 12/27/2024).

[25] URL: https://www.radiomasterrc.com/products/rp4td-expresslrs-2-4ghz-diversity-receiver?variant=48177685070055 (visited on 10/20/2024).

[26] URL: https://www.radiomasterrc.com/products/boxer-radio-controller-m2?variant=45666886910183 (visited on 10/20/2024).

[27] URL: https://www.foxeer.com/foxeer-donut-5145-props-g-520 (visited on 10/20/2024).

[28] URL: https://www.hqprop.com/hq-durable-prop-5x43x3v1s-2cw2ccw-poly-carbonate-p0048.html (visited on 10/20/2024).

[29] URL: https://isdtshop.com/en-ch/products/isdt-608ac (visited on 10/20/2024).

[30] URL: https://www.expresslrs.org/quick-start/ardupilot-setup/ (visited on 10/20/2024).

[31] *ArduPilot Copter*. URL: https://ardupilot.org/copter/ (visited on 10/19/2024).

[32] *ArduPilot Firmware builds*. URL: https://www.st.com/en/development-tools/stm32cubeprog.html (visited on 09/29/2024).

[33] Scott Campbell. URL: https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/ (visited on 12/27/2024).

[34] Scott Campbell. URL: https://www.circuitbasics.com/basics-uart-communication/ (visited on 12/27/2024).

[35] Scott Campbell. URL: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/ (visited on 12/27/2024).

[36] *Choosing an Autopilot*. URL: https://ardupilot.org/copter/docs/common-autopilots.html (visited on 10/18/2024).

[37] *DroneKit Python*. URL: https://github.com/dronekit/dronekit-python (visited on 11/03/2024).

[38] *Holybro Official Documentation*. URL: https://docs.holybro.com/ (visited on 10/06/2024).

[39]   *Kakute H7 v1.3 (MPU6000)*. URL: https://holybro.com/products/kakute-h7 (visited on 10/18/2024).

[40]   *Micro M10 GPS*. URL: https://holybro.com/products/micro-m10-m9n-gps?variant=42981482954941 (visited on 10/20/2024).

[41]   *Mission Planner Home*. URL: https://ardupilot.org/planner/index.html#home (visited on 09/29/2024).

[42]   Nathan Seidle. *What is GPS RTK?* URL: https://learn.sparkfun.com/tutorials/what-is-gps-rtk (visited on 10/20/2024).

[43]   *STM32CubeProgrammer software for all STM32*. URL: https://www.st.com/en/development-tools/stm32cubeprog.html (visited on 09/29/2024).

[44]   *Tattu 2000mAh 4S 120C 14.8V R-Line Version 3.0 Lipo Battery Pack with XT60 Plug*. URL: https://genstattu.com/ta-rl3-120c-2000-4s1p.html (visited on 10/19/2024).

[45]   *Tekko32 F4 4in1 50A ESC(AM32)*. URL: https://holybro.com/products/tekko32-f4-4in1-50a-esc (visited on 10/18/2024).

[46]   Andrew Tridgell, Peter Barker, and Stephen Dade. *MAVProxy. A UAV ground station software package for MAVLink based systems*. URL: https://ardupilot.org/mavproxy/ (visited on 10/31/2024).

[47]   *Welcome to DroneKit-Python's documentation!* URL: https://dronekit.netlify.app/ (visited on 11/03/2024).

[48]   *XING-E Pro 2207 2-6S FPV Motor*. URL: https://shop.iflight.com/xing-e-pro-2207-2-6s-fpv-nextgen-motor-pro874 (visited on 10/19/2024).

List of Tables

List of Figures

Acronyms

AIO All-in-One. 7

bps bits per second. 6

CEP circular error probable. 10

DFU device firmware updgrad. 13

ESC electronic speed controller. 3, 5, 7–9, 15, 16

Fc flight controller. 4–7, 10, 13, 15–20
FPV first person view. 3, 4

GCS ground control station. 7, 12
GNSS global navigation satellite system. 10, 13
GPS global positioning system. 3, 5, 7, 10, 13–15, 25

I2C inter-integrated circuit. 6, 7, 25

Li-ion lithium-ion. 9
LiPo lithium polymere. 9, 10

MP Mission Planner. 12, 13, 15, 16

OS operating system. 12, 17
OSS open source software. 4

PCB printed circuit board. 5
PID proportional, integral, derivative. 16, 17

RTCM radio technical commission for maritime. 10
RTK real time kinematic. 10

SCL serial clock. 7
SDA serial data. 7

SSH secure shell. 17, 18

UART universal asynchronous receiver/Transmitter. 6, 7, 25

UAV unmanned aerial vehicle. 3

venv virtual environment. 18, 19

VNC virtual network computing. 18