Kantonsschule Im Lee

Matura Paper Fall Semester 2024/2025

Development & Construction of an Autonomous Path-Following Drone

Eduard Felix Scherer, 4a

Supervised by Cyril Wendel

Winterthur, January 6, 2025

Contents

1 Introduction

In the today's world where technology is advancing faster than ever before this Matura paper sheds light on 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* [19]. 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, in this conflict drones are used with thermite canisters attached to them to target the enemies under tree covers [27]. 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 [24]. This increased use in war has also had an impact on the market for components. 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 [23]. However, this is really difficult to do regarding that the ESCs are dual-use goods[1].

The goal of this Matura paper is to develop an autonomous path-following drone, where path-following refers to navigating a global positioning system (GPS) path rather than following a ground path using image recognition. 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 3.6 (Companion Computer). After that first part, having some understanding of python will be useful, but it is not a requirement. However, without prior knowledge, the explanations might be a bit confusing.

I was first interested in drones when I was in fifth grade and received one as a Christmas present. After that, I also wrote a rather interesting piece about different types of drones in sixth grade, but I lost the interest shortly after. 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. Sadly, I never really found the time to build one. When the Matura paper came around I found it to be the perfect opportunity to work with a drone. However, I wanted to do something new, like an autonomous drone, and not just building a FPV drone, which can even be bought as a prebuilt commodity.

2 Literature Review

After extensive internet research it becomes clear that only one kind of a comparable published autonomous drone project exists, one with a flight controller (Fc) from the Pixhawk family in connection to a Raspberry Pi [11, 21, 59]. However, it is obvious that there are many drone laboratories in the industry and for military purposes (Ukraine war), which do not publish their advances in drone technologies. For example, there are many large firms, like Amazon, that are trying to deliver their packages by drones [26].

[1]Dual-use goods are products that are designed for civilian purposes, but can become deadly weapons in the wrong hands [9].

3 Methodology

3.1 General Software Considerations

3.1.1 Open Source Software

The software integration in this Matura Paper is based on open source software (OSS). OSS is a software of which the source code is public. There is an important process known as 'forking', which refers to when a person other than the original developer takes the code and modifies it independently. Through this process, a new branch of the software is created, often referred to as a forked version. The main benefit over closed software is that it is often free of charge, and the source code can be downloaded by everyone and modified as one sees fit. However the downside is that if the development team loses interest, the software will become outdated and may no longer function properly with other systems anymore. Both of these aspects will later be found useful and time-consuming.

3.1.2 Flight Softwares

Based on the literature reviewed there are three main software stacks to consider when it comes to drones: Betaflight, INAV (intelligent navigation system for aerial vehicles), and Ardupilot [6]. 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

The following overview of the three softwares, is mostly based on an article written by Liang [13] and a video made by Schofield [14].

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 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 capabilities, including waypoints and automated landing. It does not only support quadcopters, but also boats, rovers, planes and wings. It has a similar interface as 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 Raspberry Pi. A few years ago it was really expensive to start, because it only supported the Pixhawk family of flight controllers which cost a few hundred Francs per piece. However, in recent years it has began to support more and cheaper flight controllers.

3.2 Drone Overview

If someone wants to fly a drone, they have two options available: either they buy one or build one. In the case of autonomous drones, however, the first option is not available. This leaves only the option of building one yourself, compared to FPV drones, which can be bought as a prebuilt commodity.

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)[2] or simply 'boards'.

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

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 Raspberry Pi, as companion computer, which connects to the Fc. The connections between the parts are shown in Figure 1 and in Figure 2 one can see the completely assembled drone.

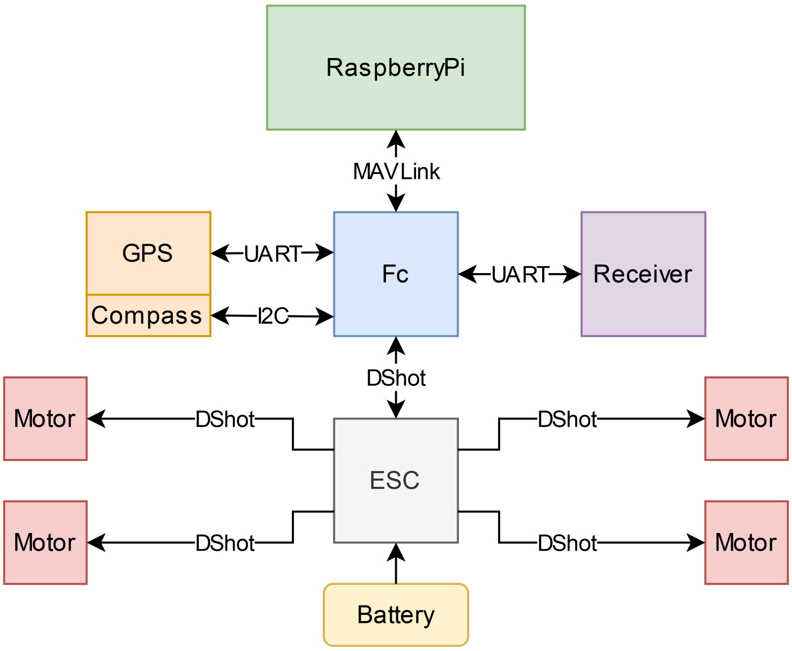


Figure 1: connection between the parts inclusive the data transfer protocols

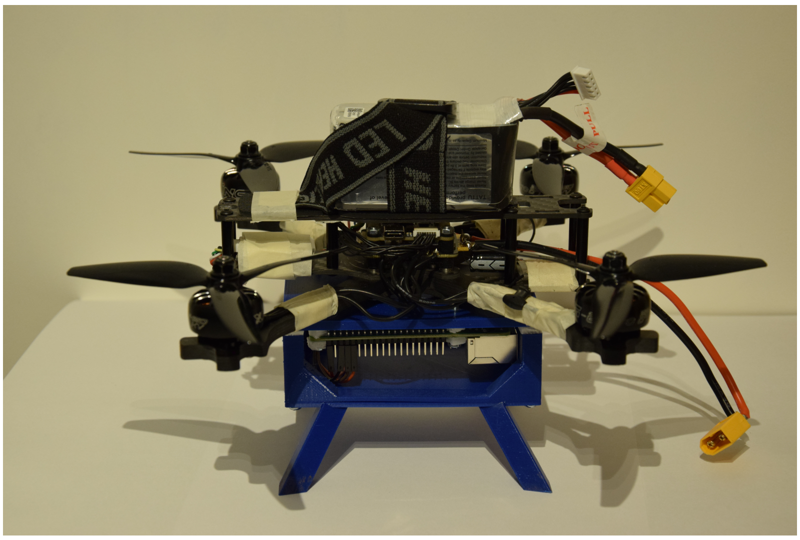


Figure 2: assembled drone

3.3 Data Transfer Protocols

There are two types of data transfer protocols: serial and parallel. 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 7 bits through eight connections, as seen in Figure 3, and the serial sends the data over one connection. This kind of data transfer requires a clock to synchronize the output of data.

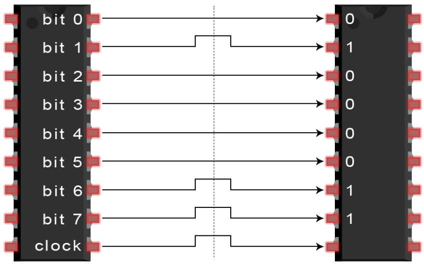


Figure 3: parallel communication of the letter "C" [43]

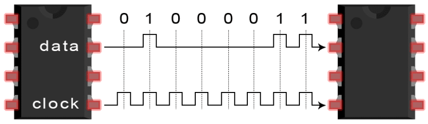


Figure 4: serial communication of the letter "C" [43]

3.3.1 UART

The universal asynchronous receiver/Transmitter (UART) is a physical circuit [44]. In the field of self-built drones it is used to communicate to and from the Fc to the GPS and receiver, as shown in Figure 1. It works by transmitting data asynchronously. 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 5. 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 6. 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 a 1 if it is odd.

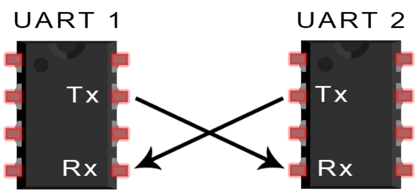


Figure 5: UART connection [44]

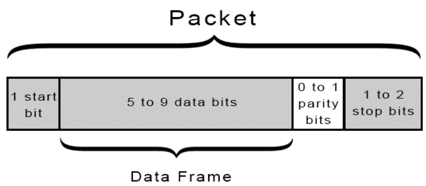


Figure 6: UART packet [44]

3.3.2 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[3]. In this work, it is used for the connection between the Fc and the compass, which is built into the GPS, as shown in Figure 1. 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 message. The message 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 message is addressed to, followed by a bit to determine if the master sends or requests data, the data is followed by a ACK/NACK[4] bit, which confirms either that the data frame was received or not, and at the end there is a stop condition.

[3] The relationship between two parts where one controls the other is called a master-slave relationship [45]. Obviously the master is the one that controls the slave. Although the terminology is rather controversial, it is still widely used [4].

[4] ACK stands for acknowledgment and NACK for negative acknowledgment

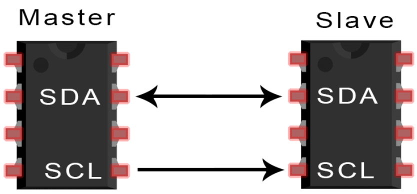


Figure 7: I2C connection [45]

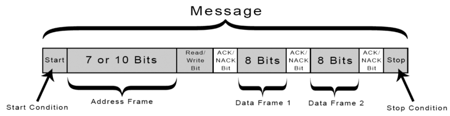


Figure 8: I2C message [45]

3.3.3 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 Raspberry Pi companion computer, as shown in Figure 1.

3.4 Parts

3.4.1 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 a part of the ESC or the Fc is 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 [46]. They have the option of open and closed hardware. Because the open hardware Fcs are quite expensive the decision was made to go with a closed hardware Fc. 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. The Kakute H7 v1.3 (MPU6000) from Holybro was then finally choosen, because it was affordable and available as a stack [49]. A stack is a Fc and ESC mounted on top of each other and normally comes with stack screws. It is shipped with Betaflight so it is required to flash Ardupilot.

> **Explanation 3.1: open and closed hardware**
>
> Like open source software, the "source code" -or in this case blueprints- open hardware can be freely accessed or bought by everyone and adjusted to meet one's individual needs [29].

3.4.2 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 they do not require a power distribution board, because it is already incorporated in the ESC, and that they 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.

The decision was made to go with a 4in1 ESC 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 [55]. The 50A version is already enough, because on one hand one does not need a high continuous current rating when flying slowly and on the other the chosen motors have a peak current of around 42A.

At the time the Tekko32 was bought, it was still shipped with BLHeli32, which as mentioned in the beginning has since seized their operations. It is however possible to flash AM32[5] onto it [17].

3.4.3 Motor

There are two types of motors: brushed and brushless ones. Brushed motors are used in very small drones with 1S[6] batteries. However, even in smaller drones brushless motors are the more popular choice [2]. A brushed motor, as shown on the left in Figure 9, 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 9 on the right, works by having a field magnet in 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. The advantage of brushless motors is that their is no wear and tear on a commutator and no contact issues from brushes that could cause sparks.

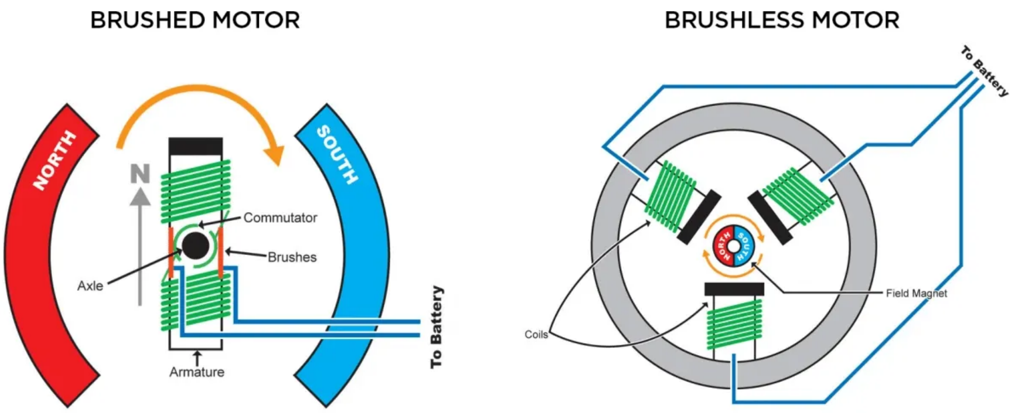


Figure 9: brushed and brushless motors [8]

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. The decision was made to use the Iflight Xing-E Pro 2207 with a KV value of 2450, because the IFlight-Xing motors are known to be of good quality [58], *'they (Xing motors) also prove to be very reliable and most importantly durable'* Landa [5].

3.4.4 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[7]. 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.

[5] A different ESC software, which has gained quite some popularity after BLHeli32 is off the market.
[6] What 1S means will be explained in the next section
[7] This happens due to unnoticed physical damage, manufacturing defects, improper storage or incorrect charging [22].

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.

The initial idea was 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 in parallel, but it requires some soldering skill, which I personally did not have at the time. Hence it was decided to go with 4S LiPo batteries from the Tattu R-line with 120C and 2000mAh, because it needs to, in addition to the Fc, power the Raspberry Pi aswell and testing the drone is rather tedious, when one needs to go back to charge every few minutes [54]. The brand Tattu was chosen, because it was the only known brand that was on AliExpress, from where all the other components were sourced, so it seemed to be easier to chose a battery from there.

3.4.5 GNSS(Global Navigation Satellite System)/Compass

There are two different kinds of global navigation satellite system (GNSS)[8] 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 a company called Ublox. The RTK GPS can achieve an accuracy of 1cm by incorporating information correction data from a radio technical commission for maritime (RTCM)[9]. 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 [52]. For a small drone it may not be worth it to have a RTK GPS that costs several hundred Swiss Francs instead of a GPS with a 2m circular error probable (CEP) which costs less than 30 Swiss Francs. Some of the GPS units also have a compass built-in, so one does not need to buy an extra compass when using ArduPilot.

Explanation 3.2: 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 [16].

The decision was made to use the Holybro Micro M10 GPS, because it is produced by the same brand as the Fc and therefore seemed to be easier to connect [50]. 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 and comes with a built-in compass that is needed for ArduPilot [25].

3.4.6 Radio/Transmitter

The following information about transmitter protocols is based on two YouTube videos from Bardwell [10] and Schofield [28].

The two things to consider when it comes to transmitter protocols are latency and range. Latency is the delay between the input of the radio controller and when the Fc reacts. The lower the latency, the faster the drone will react, to the inputs from the radiocontroler. 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.

[8] The GNSS is usually referred to as GPS even though GPS is only the American GNSS system
[9] It was first used to determine the position of maritime vessels

ExpressLRS[10]

ExpressLRS is the best protocol in long range and in the combination of long range and low latency. It has been proven by Varty and Neal [15][11] that it 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, even when compared to other protocols. Together with mLRS it is the only protocol that is open-source.

mLRS[10]

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

Transmitter/Radio

The decision was made to go with the Radiomaster RP4TD ExpressLRS 2.4GHz True Diversity Receiver based on a recommendation from friend [30]. It is also compatible with mLRS if it is needed later on.

The decision was made to go with the Radiomaster Boxer 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 too [31].

3.4.7 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 [1]. There are two groups of smoke stoppers one that you buy and is 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.

3.4.8 Propellers and Battery Charger

For the propellers and battery chargers, the recommendations from AOS-RC and FPVknowitall were used [32, 42]. For the propellers the Foxeer Donut 5145 and the HQ 5x4.3x3 V1S [33, 34] were chosen. And as the battery charger, the cheapest option, the 608 AC Lipo Battery Charger, was decided to be used [35].

3.4.9 Problems with the Parts Procurement

There were mainly two problems that. The first problem was that my ordered 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 surprise, 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.

[10] The LRS stands for long range system.

[11] 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.

3.5 ArduPilot

This chapter summarizes everything done for the first time flying and what could go wrong is based on my experience and the ArduPilot copter documentation [40].

3.5.1 Ground Station

To configure ArduPilot, there are multiple softwares, so called GCS required. Usually they are ground-based and can transmit data via wireless telemetry device or USB cable. With the telemetry device, they are able to 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), but runs only on Windows and Mac OS [51]. It has a wiki, which was used for the first-time configuration of my drone.

Another GCS is MavProxy. It is written in python and intended to be used on Unix-based systems. MavProxy will be used to test the connection between the Fc and the Raspberry Pi, which is using the Debian operating system (OS), a Unix-based system.

> Explanation 3.3: operating system(OS)
>
> A OS is a software, which controls all the hardware of a computer.

3.5.2 Firmware Installation

The following section until Section 3.6 (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 3.4: 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 on the Kakute H7 it needs to be downloaded onto a computer [41]. Afterwards the STM32CubeProgrammer is used to flash the firmware onto the Fc [53]. The Fc in device firmware updgrade (DFU) mode is directly connected with the computer, using a USB cable. Then the USB port, with which the Fc is connected, is 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 the rest of the configuration.

> Explanation 3.5: 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, during startup.

3.5.3 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 10, still occurred. Even though it was clear the GPS was working and connected to the Fc, because the compass, which is part of the GPS, appeared in MP and the GPS blinked blue, indicating a connection to a GNSS.

The issue could be that the GPS is too close to metal surfaces, the computer, which is connected via USB cable, is too close, that the soldering was done poorly, or the cables from the GPS to the

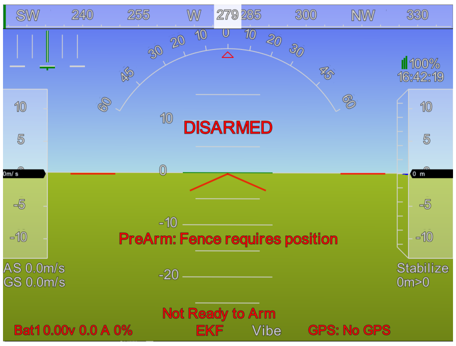


Figure 10: No GPS connection

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 problem was that the `Serial3_Protocol` (which is for the Uart3 that will be used for the connection to the Raspberry Pi) was set to 5 which stands for GPS. However, this blocked the Uart4, to which the GPS is connected, from being received as a GPS. After disabling Uart3 it finally worked.

The GPS is quite precise outside Figure 11[12].

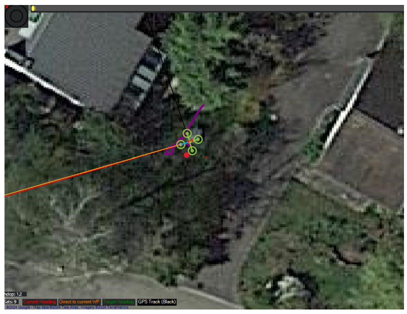


Figure 11: GPS outside, the red point shows the real position

3.5.4 Receiver/Transmitter

To establish the connection between the receiver and radio the ExpressLRS page needs to be followed [36]. It required changing the `Serial6_Protocol` to 23 and the `RSSI_Type` to 3 so it can be the receiving end. The `RC_Options` needed to be changed as well to the correct bitmask (Figure 12).

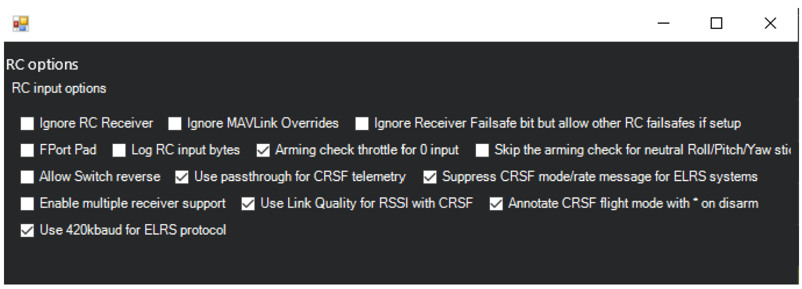


Figure 12: correct bitmask for ExpressLRS

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

The connection between the radio and receiver seemed to be working, because the receiver had a constant blue light. In addition the radio confirmed the connection, by notifying that the telemetry has been 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 to 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, in which was written, that for a CRSF[13] interface the parameter `Brd_Alt_Config` needs to be set to 1. `Brd_Alt_Config` is a Fc specific parameter, this is also the reason why it did not come up in the ExpressLRS page or the ArduPilot documentation.

3.5.5 Battery

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

3.5.6 Motor Test

The motor test worked correct after assigning the right position to the motors. Which needed to be done, in this setup the ports M5-M8 are used, instead of the M1-M4. Additionally the parameter `Mot_PWM_Type` needs to be set to DShot 600. Digital shot (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, because of this a new error appeared called 'battery failsafe'. This is caused by an unstable connection between the ESC and the Fc, which causes the Fc to take the USB cable from the computer as power source and has through that too little power to spin the motors, as they need a higher voltage than provided by the computer.

3.5.7 Compass Calibration

The compass calibration needs a good GPS lock. However, even with a good lock, relaxed fitness[14], and `Compass_Orient` set to 6, as recommended by the Holybro docs, it failed [48]. 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[15] 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.

3.5.8 Road to First Flight

The first step to flying is to arm the drone.

> **Explanation 3.6: 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 a safety measure, to prevent the motors from accidentally spinning, when working on the drone and touching the radio.

For that a switch on the radio needs to be assigned to arming and disarming. In my configuration 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 switch five is flicked. To test the drone on the bench inside there will be many prearm errors, to ignore them the parameter `Arming_Check` is disabled and more importantly the geofence, which still blocks the arming, even when `Arming_Check` is disabled. After this if switch

[13] Even though it is a ELRS receiver it has the same interface as CRSF receivers

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

[15] The compass calibration is done by rotating the copter in the air. However, this is not really an option for larger vehicles, to calibrate them the large vehicle MagCal is used, by just putting in the direction the compass is facing.

five is flicked the motors will arm[16]. It is really important to tie down every cable or antenna before arming the drone with propellers, else they might just be cut or teared off. This will cause a new prearm error to appear called 'crashdump bin detected'. Which is a file that is created when the drone crashes and 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 [20]. This time however not via STM32CubeProgrammer, but directly over MP. Additionally some of the motors need to be reversed for the X-configuration of the drone, seen in Figure 13. To reverse them the reverse button in MP does not work. The BLHeliSuite32 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 ESC will beep after ten minutes of being idle.

Explanation 3.7: 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.

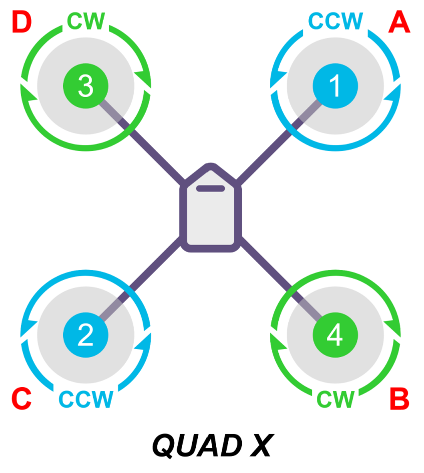


Figure 13: Motor turning directions

In the first test flight the drone was shaking violently, which is also known as wobbling. This caused the drone to crash into the ground. Reasons for this behavior can be loosely fastened parts, imbalanced propellers, or wrong parameters used for the proportional, integral, derivative (PID) controller [7]. In this case those parameters were for a 9-inch drone and not a 5-inch. The parameters can be adjusted via the initial tuning parameter tab in MP. After this the drone flew fine, except that the sticks of the radio controller needed to be reversed, because when one pulled the stick left the drone flew to the right. This can be done using the `RC2_Reversed` parameter.

Explanation 3.8: proportional, integral, derivative (PID)

The three parameters are used as weights for the three parts (PID) of the error function.

3.6 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 Raspberry Pi working together using Dronekit python.

[16] A battery needs to be connected.

3.6.1 Raspberry Pi Setup

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

Choosing the Raspberry Pi instead of another companion computer is advisable, due to it being the most popular in autonomous drone projects. However, such projects are rarely published as mentioned in Section 2 (Literature Review).

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

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 [12]. To be able to use RealVNC, RealVNC server needs to be installed over the terminal via `sudo apt-get realvnc-vnc-server`[17].

3.6.2 MAVProxy Installation

The following was based on the MAVProxy Documentation from the ArduPilot website [56].

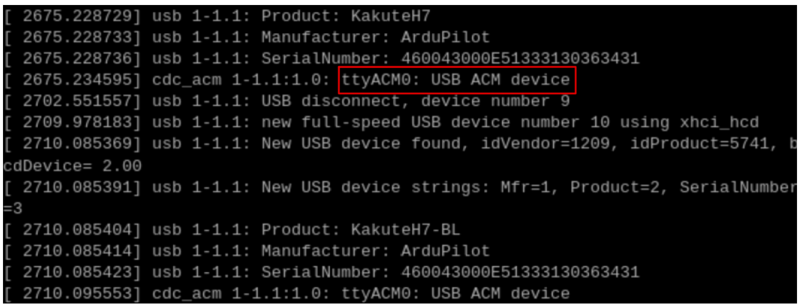
Firstly all the necessary packages need to be installed through the `sudo apt-get install` command.

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

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

To connect the drone over MAVProxy it needs to know how the Fc is connected to the Raspberry Pi. The command `dmesg | tail` will provide the information, as seen in Figure 14.



```
[ 2675.228729] usb 1-1.1: Product: KakuteH7
[ 2675.228733] usb 1-1.1: Manufacturer: ArduPilot
[ 2675.228736] usb 1-1.1: SerialNumber: 460043000E51333130363431
[ 2675.234595] cdc_acm 1-1.1:1.0: ttyACM0: USB ACM device
[ 2702.551557] usb 1-1.1: USB disconnect, device number 9
[ 2709.978183] usb 1-1.1: new full-speed USB device number 10 using xhci_hcd
[ 2710.085369] usb 1-1.1: New USB device found, idVendor=1209, idProduct=5741, b
cdDevice= 2.00
[ 2710.085391] usb 1-1.1: New USB device strings: Mfr=1, Product=2, SerialNumber
=3
[ 2710.085404] usb 1-1.1: Product: KakuteH7-BL
[ 2710.085414] usb 1-1.1: Manufacturer: ArduPilot
[ 2710.085423] usb 1-1.1: SerialNumber: 460043000E51333130363431
[ 2710.095553] cdc_acm 1-1.1:1.0: ttyACM0: USB ACM device
```

Figure 14: 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 Raspberry Pi is connected over the serial ports, `\serial0` is used instead of `\ttyACM0`. The baudrate can also vary, in this case it is 921600. The Raspberry Pi will not have another power

[17] `realvnc-vnc-viewer` is only required if you want to see a VNC from the Raspberry Pi

source except over the Fc. The 5+ V pin on the Raspberry Pi is connected to a 5 volt pin on the Fc and additionally a ground pin. There can be a low voltage warning, even though the Raspberry Pi is not receiving enough power, it is sufficient to run Dronekit python code. In addition the `Serial3_Protocol` needs to be changed to 2 for the MAVLink protocol.

When connected to the Fc you can use simple commands to change the parameters, as `param set arming_check 0`, or to arm the copter with `arm throttle`. This shows that the connection between the Fc and Raspberry Pi is working and one can advance further to Dronekit.

3.6.3 Dronekit

It first needs to be mentioned that the Dronekit python software is not maintained very well, as stated in the Github repository [47]. Even though outdated, the following is still based on the Dronekit Documentation [57].

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 not to use the same name as the library itself, because python will confuse it. In the created document one then can connect the Raspberry Pi to the Fc as can be seen in Listing 1.

```
from dronekit import connect

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

```
dronekit/__init__.py" , line 2689, in <module>
class Parameters(collections.MutableMapping, HasObservers):
^^^^^^^^^^^^^^^^^^^^^^^^^^
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[18].

```
nano /home/EduPi/mavproxy-env/lib/python3.11/site-packages/dronekit/__init__.
    py
```

Listing 3: accessing source code

```
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

```
print( "Autopilot version: %s" %vehicle.version)
```

Listing 5: information retrieval

```
Autopilot version: APM:Copter-4.5.7
```

Listing 6: output from Listing 5

The main problem of the outdated Dronekit 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].

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

Explanation 3.9: Decorator

A decorator is a function that modifies the behavior of a function or a class [18].

```
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).

```
chan{i}_raw : x
```

Listing 8: output from decorator in Listing 7[19]

The location is straightforward. 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 the longitude and the latitude from a normal GPS system, 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`[20]. This will have a list output `[longitude, latitude, altitude]`, to access the longitude a `.lon` is added after the `_frame` from before[21].

To be able to troubleshoot what went wrong in a test flight a `log()` function is used. This function can be seen in Listing 9. On line 2 the correct file to open 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. This function is used to write out different values during testing, for example the horizontal dilution of precision (HDOP) and vertical dilution of precision (VDOP) values, the coordinates and the flight mode.

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

Listing 9: log function

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.

```
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))))
```

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

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

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

```
changeZ = home_location.alt + float(Z[i])
```

Listing 10: coordinate conversion

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 the channels defined in `Brd_Safety_Mask`, go through to the ESC. Hence both, `Brd_Safetyoption` and `Brd_Safety_Deflt`, need to be deactivated by the code, as demonstrated in Listing 11

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

Listing 11: deactivation of parameters[22]

Something that needs to be looked out for when flying low over the ground is if there is a low VDOP, because it is not one of the prearm messages that would not hinder the drone from taking off.

Explanation 3.10: horizontal and vertical dillution of precision (HDOP/VDOP)

The HDOP or VDOP refers to the dillution of the precision of the GPS. The values are not in meters and range from under 1 (ideal) up to over 20 (poor).

After the most important functions have been defined. They can be put together in an overall flow, as showed in Figure 15. It is two nested loops, with the inner loop checking, if the target location is achieved and the outer loop looking if the achieved target location was the last in the list taken form the coordinates.txt file.

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

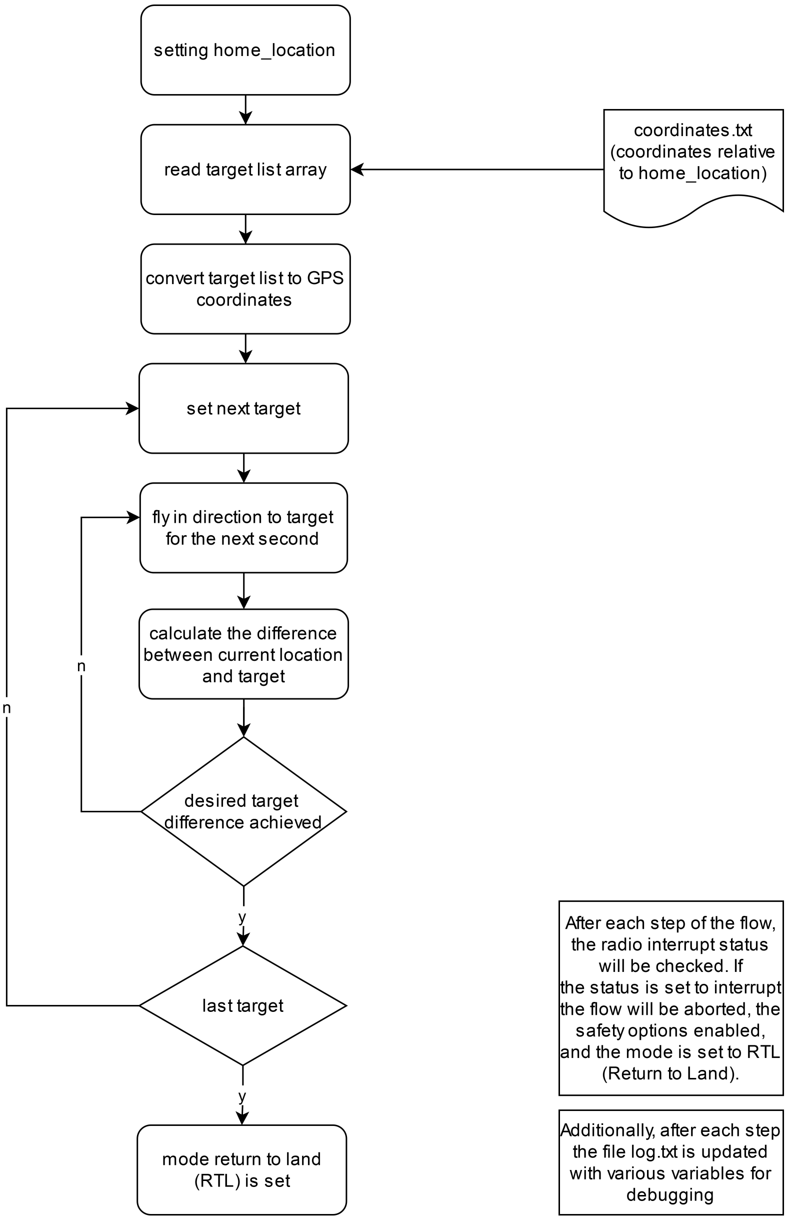


Figure 15: overall flow

4 Results

The various test flights showed promising results. The drone followed the path from the coordinates.txt file, which was the main goal of this Matura paper. The test flights have also shown the very sensitive reaction of the drone to the parameter settings.

The integration of the different software components was also a challenge, due to the fact that the open source software are not transparently documented and some are also outdated. The software components often conflicted during drone takeoff, with one allowing flight while another prevented it[23]. This also shows the difficulty of software integration, which is a tedious and time expensive task.

This Matura paper also shows how important a log file is for an autonomous system, because once

[23]The prevention always had the upper hand.

the system takes control, it is impossible to know what is happening and without the data provided by a log file it is difficult to improve the system.

5 Discussion and Outlook

The drone built in this work is only a proof of concept and requires further testing and tuning to reach full maturity. Additionally, a camera (or infrared camera) can be added to the Raspberry Pi. Unlike an Arduino, the Raspberry Pi is a fully functioning computer with the potential to add many other functionalities, such as sensor data and image processing. The pictures taken by the camera from the Raspberry Pi could be processed directly onboard for tasks such as image recognition and anomaly detection. This would be a straightforward development on the basis of the Raspberry Pi proven technology.

6 Conclusion

A general problem during the work on the drone was that the development location at home lacked both a GPS connection and space to test the drone. The trips to another location for testing were time-consuming because every time the drone needed to be tested the entire setup –including the laptop– was packed into a bag, moved to the location, and reassembled. Additionally, the weather interfered my testing plans sometimes.

On the other side the infrastructure at home was very efficient, due to the distinction in three blocks, the soldering and electronic workstation, a separate desk for software development, and an always available 3D printer. The 3D printer was used for printing the mount for the Raspberry Pi attached to the drone frame.

The entire project was interesting, and I learned a lot –from sourcing parts to soldering to complex integration of components– but it was also exhausting and sometimes even nerve-racking. Success was not guaranteed, due to the many challenges and technical hurdles faced. So it was quite relieving when the drone finally flew.

Appendices

A Thanks

First, I want to thank Cyril Wendel my supervisor for his support and valuable coaching, which I appreciated very much. I also want to thank my Father, my Mother, Claude Meier and Janik Bolli for reviewing the paper.

B List of Parts

Physical Parts

- Fc: Kakute H7 v1.3 (MPU6000) [49]
- ESC: Tekko32 F4 Metal 4in1 65A ESC (AM32) [55]
- GPS: Micro M10 GPS [37]
- Motors: 4x XING-E Pro 2207 2450KV [58]
- Propellers: Foxeer Donut 5145 Props [33] or HQProp 5X4.3X3V1S [34]
- Battery: Tattu 2000mAh 4S 120C 14.8V R-Line Version 3.0 [54]
- Receiver: RP4TD ExpressLRS 2.4GHz True Diversity Receiver [30]
- Radio: Boxer Radio Controller (M2) [31]
- Raspberry Pi: Raspberry Pi 4 Model B - 2GB [38]
- Frame: TBS Source One V5.1 5inch [39]
- Additional Screws: 4x M3x20, 4x M3x30 (for the Fc/ESC stack), 4x M3x40
- Raspberry Pi mount designed specially for this project. STL file can be found here: `https://github.com/pythonnoob9903/MA-project/tree/main/RaspberryPi%20Mount`

Non-Physical Parts

- Code and Coordinates[24]: `https://github.com/pythonnoob9903/MA-project/tree/main`

[24] Found as dronekittest.py, coordinates.py, and coordinates.txt

C References

[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] Mar. 24, 2021. URL: https://www.europarl.europa.eu/topics/en/article/20210319STO00424/dual-use-goods-what-are-they-and-why-are-new-rules-needed (visited on 01/03/2025).

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

[11] Aman Garg. Sept. 23, 2022. URL: https://all3dp.com/2/raspbery-pi-drone-simply-explained/ (visited on 01/03/2025).

[12] 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).

[13] 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).

[14] 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).

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

[16] 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).

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

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

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

[20] 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).

[21] Sally French. Jan. 31, 2024. URL: https://www.thedronegirl.com/2021/04/05/how-to-build-a-raspberry-pi-drone/ (visited on 01/03/2025).

[22] John. Feb. 2, 2024. URL: https://www.ufinebattery.com/blog/how-to-prevent-lipo-battery-explosion/ (visited on 01/03/2025).

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

[24] 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).

[25] 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).

[26] Kevin Roose. Dec. 20, 2024. URL: https://www.nytimes.com/2024/12/20/technology/amazon-prime-air-drone-delivery.html (visited on 01/03/2025).

[27] 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).

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

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

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

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

[32] URL: https://www.aos-rc.com/recommended-parts/5in-freestyle (visited on 01/04/2025).

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

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

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

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

[37] URL: https://holybro.com/products/micro-m10-gps?variant=42981482954941 (visited on 01/03/2025).

[38] URL: https://www.pi-shop.ch/raspberry-pi-4-model-b-2gb (visited on 01/03/2025).

[39] URL: https://www.team-blacksheep.com/products/prod:sourceone_v5 (visited on 01/03/2025).

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

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

[42] Joshua Bardwell. URL: https://www.fpvknowitall.com/fpv-shopping-list-charging/ (visited on 01/04/2025).

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

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

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

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

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

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

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

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

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

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

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

[54] *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).

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

[56] 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).

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

[58] *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).

[59] Jawwad Zahed. URL: https://www.instructables.com/Autonomous-Drone-Using-RPi/ (visited on 01/03/2025).

D List of Figures

E Acronyms

AIO All-in-One. 6

bps bits per second. 5

CEP circular error probable. 8

DFU device firmware updgrade. 10

DShot digital shot. 12

ESC electronic speed controller. 2–4, 6, 7, 12, 13, 17, 20

Fc flight controller. 2–6, 8, 10–16, 20

FPV first person view. 2, 3

GCS ground control station. 6, 10

GNSS global navigation satellite system. 8, 10

GPS global positioning system. 2–5, 8, 10–12, 16, 17, 19, 20, 24

HDOP horizontal dilution of precision. 16, 17

I2C inter-integrated circuit. 5, 6, 24

INAV intelligent navigation system for aerial vehicles. 3, 6

Li-ion lithium-ion. 7, 8

LiPo lithium polymere. 7–9

MP Mission Planner. 10, 12, 13

OS operating system. 10, 14

OSS open source software. 3

PCB printed circuit board. 3

PID proportional, integral, derivative. 13

RTCM radio technical commission for maritime. 8

RTK real time kinematic. 8

SCL serial clock. 5

SDA serial data. 5

SSH secure shell. 14

UART universal asynchronous receiver/Transmitter. 5, 6, 24

UAV unmanned aerial vehicle. 2

VDOP vertical dilution of precision. 16, 17

venv virtual environment. 14, 15

VNC virtual network computing. 14

VTOL vertical takeoff and landing. 3