UiT The Arctic University of Norway

Department of Electrical Engineering

Designing an Autonomous Racing Drone

Alexander Trældal

Master's thesis in Aerospace Control Engineering STE-3900 May 2021

Abstract

In recent years drone racing using human pilots have increased in popularity. This combined with the increased usage of autonomous driving vehicles have sparked a new field of interest which is Autonomous Racing Drones competitions. This thesis explores the field of drone design and avionics with the conclusion of presenting the first Racing Drone prototype that can be used in future work made by students at UiT. The body and avionics have all been designed with the idea of making future iterations and improvements possible. The design considerations and decisions are also explored in-depth to give a reasoning behind the final result.

Preface

I would like to start by thanking my supervisors T.S Andersen, Tu Dac Ho and Jose Corona for giving me insight into several aspect of UAV control theory and how the avionics operate. As being unfamiliar with how to approach the avionics and the implications the massive jigsaw puzzle of selecting the correct components, it really helped me move forward and arrive with a result that I can be proud off. A thanks also goes to Ketil Hansen for his swift help in procuring all the avionics as several orders had to be made throughout the project.

Contents

List of Figures

1 Introduction

In recent years drone racing using human pilots have increased in popularity. This combined with the increased usage of autonomous driving vehicles have sparked a new field of interest which is Autonomous Racing Drones competitions. Some of the more notable ones being IROS and Alpha Pilot Autonomous Racing Drone competitions. With this subject matter it is also in Aerospace Control Engineering at UiT's greatest interest to also be able to compete in such events. The goal is to create a prototype racing drone that can in collaboration with other students focusing on visual odometry which includes guidance and navigation, create an autonomous racing drone that can one day compete with other universities in the racing events. This thesis will act as the starting point for the larger project of making an Autonomous Racing Drone, this being the basis for the avionics and overall structural design.

1.1 Abbreviation

FCU - Flight Control Unit
ESC - Electronic Speed Controller
UAV - Unmanned Aerial Vehicle
PDB - Power Distribution Board
IMU - Inertial Measurement Unit
MCU - Microcontroller Unit
BLDC-motor - Brushless DC-motor
TI - Texas Instrument
STM - STMicroelectronics
IDE - Integrated Development Environment

1.2 Unmanned Aerial Vehicle

1.2.1 Subsystems

Drone subsystems can usually be divided into FCU, ESC, COM and SENSORS. The FCU is the main processing unit which inputs all sensory data and outputs an individual PWM signal to each of the four ESC's. The ESC can be either 4in1 or one circuit board dedicated to each motor, the ESC main function drive the brushless DC motors and also to stop any current spikes generated by the motors to impact the other subsystems. Depending on the drone design, the IMU and ESC can be located on the same board as the FCU, or in any combination of seperate modules. The COM subsystem is dependant on the drones use-case, this can include Bluetooth for linking to mobile devices or simply an RF-antenna based system to remotely control the drone. The SENSOR subsystem contains a variety of

measurement modules referred to as the IMU, this primarily includes a gyroscope and accelerometer combined into one package commonly referred to as a 6DOF IMU whereas those that include a magnetometer can be called a 9DOF IMU. The magnetometers needs to be carefully calibrated as large metal structures and the UAV motors will affect the magnetic field around the sensor[1]. A distance sensor can be included in drones where it is used for assisting in landing and collision detection.

1.2.2 Design

To get a better sense of what the drone should include we can follow the procedures used by S.A. Brandt[2]. Assuming that the Darwinian process applies to quadrotors as well, it should be perfectly viable to look at trends in the industry. From a layout perspective and our use-case, it can be made an approximation shown in Figure 1 over how the general layout should look like. The FCU receives sensory data which it sends to the Jetson (Xavier), the Jetson runs its algorithms and returns position data to the the FCU. The FCU then sends a PWM signal to the motors that changes its trajectory depending on the desired position received from the Jetson.

1.2.3 Flight Control Unit

The FCU is in charge of communication with all the other modules on the drone, this can include calculating the control algorithm, receiving commands from a ground station, sending PWM-signals to the ESC and in our case sending and receiving data to the Jetson module. Since the FCU has to be able to cycle through the code at a sufficiently high frequency depending on the drone use-case, it might even be viable to have more than one MCU if the data processing is too demanding.

1.2.4 Motor and ESC

The motor on the quadrotor should be a BLDC-motor and not a brushed DC-motor, which is more suited for very small quadrotors (MAV) [3]. The ESC consists of several MOSFET-pairs which switch power to the motor poles on and off. This can be achieved in different ways like FOC or trapezoidal control algorithms [4]. Regardless of what configuration used for the ESC, the angular velocity needs to be fed back to the FCU. This can be achieved by motors with a built in tachometer or install current, hall-effect, optical, or angle encoder-sensors. The topic of motor control is vast with many elements that needs to be considered when designing a drone [5]. As there probably wont be one correct answer for what the final prototype should have, testing of different sensors should be conducted to evaluate the best type for the given task.

1.2.5 Distance Sensors

A barometer can be used to more accurately land drones using the relative pressure at takeoff as ground level. This method is not considered precise enough for a fully automated drone and as such, is only useful for aiding a drone operator in landing. A ultrasonic or IR sensor can be used to more accurately measure the distance between the quadrotor and another object like the ground. These usually have limited range or field of view but very accurate compared to a barometer. There are limitations like reflective and translucent surfaces for the IR sensor and extreme textures for ultrasonic sensors. It is also possible to use the camera data used in the guidance and navigation part to calculate distance, which is not part of this literature review.

1.2.6 Inertial Measurement Unit

An IMU includes the motion sensors to measure the linear acceleration in X,Y and Z-direction. The gyroscope measures the angular velocity in each direction known as pitch, roll and yaw and the magnetometer measures the earth's magnetic field for each axis. The precision of a IMU is primarily determined by its price, which is made very cheap by the use in every smartphone and quadrotor on the market. The main concern with a IMU is that the accelerometer is very susceptible to vibrations usually made by the motors. This can be compensated for by using rubber feet to isolate the IMU from the rigid body of the UAV.

1.3 Extra modules

For a automated drone, a RF receiver and a SD card writer is not mandatory, but there are values to installing such modules. An RF receiver will make it possible to add the ability to remotely control the drone if problems occur during flight. Likewise, an SD card would make it easy to collect flight data which would act as a black-box for debugging potential problems with the drone.

1.3.1 Additional Considerations

Vibrations is something that is guaranteed to affect the IMU but its affect on other sensors, more importantly the Jetson Xavier module is unknown. As a precaution, testing with vibration dampening material should be conducted to see its effect on different modules. Thrust to weight ratio is an important value that determines the speed and response of the drone, in enquiring about system requirements this value should be higher than 1.5, prefferably 2.0 if possible.

1.3.2 State of the Art

The racing drone is intended to make use of the best available components while also keeping the price within a standard budget of drones available on the market. Looking at the current market on what the latest modules offer it is possible to make a list of possible components that should be explored moving forward with the main project.

1.3.3 MCU

A lot of FCU's on the market use a STM32 microcontroller which is a 32bit ARM Cortex processor which gets faster with every generation. The latest model offer speeds to 540MHz[6] and some even have dual cores enabling parallel computing. The more noteworthy part of using a STM32 is the amount of tools and resources available from ST and online, making prototyping faster and easier.

1.3.4 Prototyping

The selection of each individual part wont be possible without further testing and validation of each components performance, price and availability. It is however desired that the end product consists of at least 2-3 circuit boards (FCU,IMU and ESC) excluding other modules attached to specific parts and directions on the drone. While buying complete modules and attaching them to a frame can technically be considered completing the main task, it is desired to create as much as possible from scratch. This involves designing a FCU and possibly IMU and ESC circuit boards.

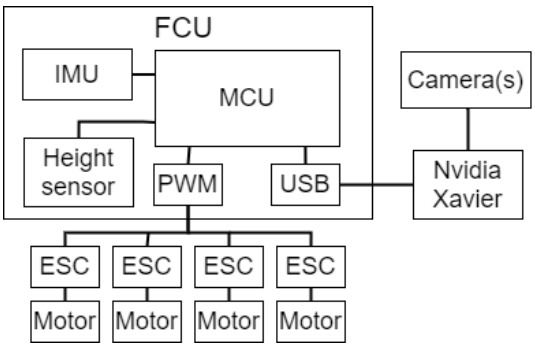

Figure 1: Drone block diagram

1.4 Contributions and scope of the thesis

The purpose of this thesis is to design the hardware of a autonomous racing drone utilising an Nvidia Jetson for visual guidance and navigation. The

main task is to select and implement all the necessary avionics in a custom designed body. While the plan was to implement the provided filter [7] and Nvidia Jetson communication with the FCU, in agreement with the supervisors it was decided that this thesis should focus mainly on completing the basic functionality of the standalone drone. The work completed in this thesis will become the building blocks for future drone projects, specifically the drone created will be the base platform for future design iterations.

1.5 Report outline

This paper is structured with each chapter containing the following. Chapter 2 includes the final result prototype drone with all relevant technical specifications and features. Chapter 3 gives an explanation and reasoning as to why certain components were selected for the drone. Chapter 4 discusses the design process, all the considerations and implications that had to be accounted moving forward with the design. Chapter 5 is the conclusion and Chapter 6 mentions possible ways to improve the current design and what needs to be experimented on further.

2 Autonomous Racing drone

Introducing the first Autonomous Racing Drone prototype as seen in Figure 2, which has been 3D-printed in PET-G and has a total weight of 1100g (including components, without battery).

Figure 2: Picture of the Autonomous Racing Drone.

2.1 Components

The drone is equipped with the components listed in Table 1 and the dimensions of the drone are designed for fitting these exact components. It also features support for implementation of angular velocity sensors (hall effect) for each motor and a PX4FLOW optical flow sensor.

Table 1: Drone Components

Type	Name
FCU	NUCLEO-H723ZG
ESC	B-G431B-ESC1
IMU	SF-ICM20948
Motor	DJI920
PDB	FCHUB-6S
Camera	e-CAM24_CUNX
Image processor	Nvidia Jetson Xavier NX

2.2 Body

The drone body is designed using Fusion360 and all the body parts are split up to make it printable on a 24x24cm 3D printer. For future reference, the drone is printed using "PET-G Prusament" at 0.2mm layerheight and 15% infill. The files necessary to replicate or modify the design can be found in Appendix A. Some key measurements of the drone can be seen in Figure 3

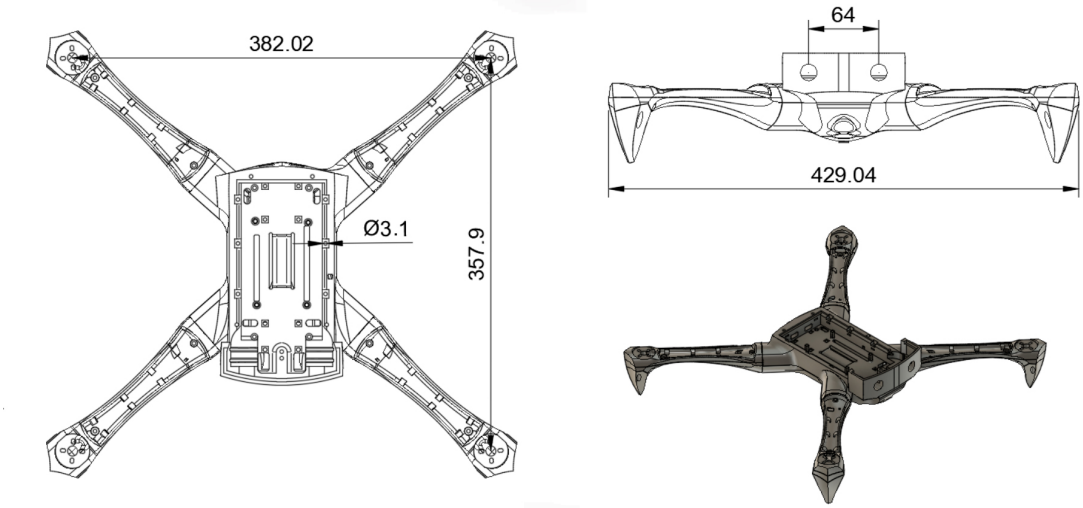

Figure 3: Drone dimensions specified in millimetre, all boltholes are for M3 bolts

2.3 Connections

The connection between components apart from the Jetson is shown in Figure 4. The connections are soldered and ESC/Motor connections are equal on all 4 sets. For programming and future reference, the FCU pinout connections are shown in Table 2 below.

Table 2: Drone Components

Type	Pinout	ID
PWM	PF6	TIM23 (Ch1)
PWM	PA0	TIM5 (Ch1)
PWM	PF11	TIM24 (Ch1)
PWM	PA5	TIM2 (Ch1)
SCL	PF1	I2C2
SDA	PF0	I2C2

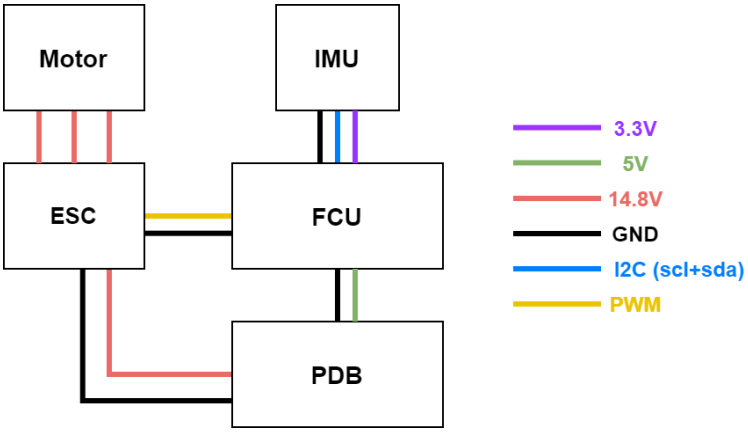

Figure 4: Block diagram of subsystem connections

2.4 Workflow

With the given selection of part the programmable ones being the FCU and ESC both fall into the same programming eco-system. The FCU and ESC have separate programs, STM32CubeMX and STMotorControlWorkbench respectively, both with a custom GUI that makes it easy to quickly auto-generate code that enables all the basic features of the two components. This saves considerable time that would be spent delving into the datasheets just to set up timers, motor tuning or protocols like I^2C or SPI. The auto-generated code is imported manually or automatically into STM32CubeIDE for both components and any code written inside specified areas are not deleted. This means it is possible to rapidly prototype and experiment with all the advanced features they have to offer without having to rewrite codes.

3 Part selection

A lot of time in this project was spent looking at diffrent parts and trying to determine what which to use for the racing drone. With the vast majority of parts being essentially marketed as plug-and-play or no technical knowledge needed, it was easy to narrow the list for most components as what is essentially a black box was not desirable from a prototyping standpoint.

Since the drone project was based on making a ”state-of-the-art” Autonomous Racing drone, the methodology for selecting parts or at least narrow down the list of possible parts. Old and redundant parts were not selected even though older IMU’s would probably do be ”good enough”. It was desirable to have as many possible new features as possible since even though it wont be possible to implement them all, the drone project itself will be continued in future works which giving more possibilities for improvements. It is also worth noting that due to budgets and time constraints it

wasn't feasible to buy a whole set of components just for comparison, especially the FCU as the programming eco-systems is also something that can take time learning. With that in mind, once the MCU was selected it wouldn't be possible to change this later in the project.

3.1 FCU

Looking at the mainstream options, the obvious choice for ease-of-use with ready available libraries for programming would be an ARDUINO or a pre-built FCU with plug-and-play functionality. The problem was that none were equipped with the desired clockspeed as most only went up to 72MHz which was known to not be good enough for this project. The other option was to go for a TI or STM32 MCU which are equipped with better processors but also has their own complete eco-system of debugging and programming tools that has a steeper learning curve than ARDUINO's.

Regardless of what MCU was selected, additional considerations had to be made since the final goal was to make a custom PCB. A development board is usually packed with all the features a MCU has to offer, with only a tiny amount being desirable for the racing drone. The upside with making a custom PCB is it would make it easy to have all the custom connections and power management on one single PCB, reducing weight and overall footprint of the drone. The plan was to use a vendor like LCSC (component vendor) which is partnered with JLCPCB (PCB vendor) to order a set of custom made PCB's that have all the parts assembled. Though they might not have the specific MCU in stock, all other passive components are usually available, saving a lot of time manual soldering. In case manual soldering of the MCU was needed, it is extremely desirable that the MCU is in a LQFP (Low Quad Flat Package) and not a BLGA (Ball Grid Array). The first is easily solder-able by hand if the pitch is not too small as all the pins stick out on the side, the latter as the name suggests has a grid of solder balls attached underneath it making it extremely difficult to solder by hand.

With the given criteria, STM offers ARM Cortex M7 processors running up to 550MHz for a reasonable price with the bonus of their CubeIDE software makes it easy to configure pinouts and coding without having to spend weeks researching the datasheet. The chosen processor family STM32H7xx have 144 pins but also offers a smaller LQFP 100pin variant as that was the smallest available pin number for their top processors available. The specific development board NUCLEO STM32H723ZG was chosen as it was in stock, it is worth noting that the other STM32H7 are near identical in terms of pinouts, making it possible to swap to a something like a STM32H743 without issues.

3.2 ESC

Following the same approach as with the FCU, except this was done after the FCU was bought and done preliminary testing on. The idea to make a custom PCB for the ESC was dropped after considering the complexity and compact format these are created on. The design considerations creating an ESC is a lot more strict as the PCB needs to be rated for high current and possibly high noise from the switching 3-phase generated. As a side note, the ESC selected is a 6-Layer board with an extremely dense component placing, this in itself would be quite time-consuming to design and create as well as a different and more expensive vendor would need to be selected for the PCB prototyping.

With the FCU eco-system already selected and the fact that the ESC would need to be programmable if FOC was to be implemented. The choice ended on another STM32 MCU as it was equipped with all the tools the FCU uses in addition to special motor tuning utilities.

3.3 IMU

During the learning process of using STM32CubeIDE, several old IMU's were used to test functionality and experiment with different programming approaches. These include HMC5883L,ADXL345/ITG3205 and LSM9DS0 which could potentially be used in the final build, but considering they are discontinued products they were discarded. The choice ended up on a ICM20948 as seen in Figure 5 which is one of the newer IMU's on the market, the mounting holes were also slotted with vibration dampeners to reduce noise.

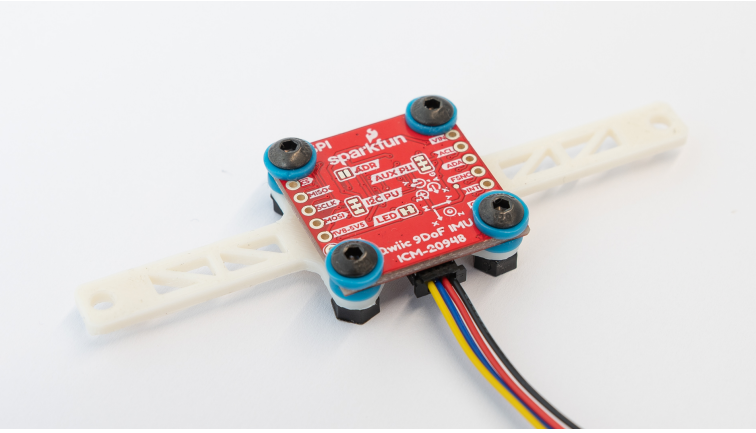

Figure 5: SparkFun 9DoF ICM20948 secured with bolts surrounded with vibration dampeners

3.4 PDB

In order to keep the cabling and power systems tidy, a PDB (Power Distribution Board) was added later in the design process. This is strictly not an essential part of a drone and more of a way to keep the cabling tidy. An added bonus of the selected board is that it has two integrated step-down converters to 5v and 10v.

3.5 Jetson & Cameras

The original plan was to use a Jetson Xavier AGX 6 but due to Cameras connections adding significant cost and weight it was decided to go with a Xavier NX. This meant a reduction in price and weight as well as an added 2 camera connections. The Xavier AGX would require an extra PCB with a special addon card to be slotted in Nvidia's proprietary connector to then split into multiple camera connections. This not only added extra weight but the addon boards themselves could cost as much as the Xavier NX. As an added bonus of going with the Xavier NX, a lot of camera vendors which only supported the Jetson Nano could be used, as the Xavier NX is technically a new and improved version of the Jetson Nano. The criteria for the cameras was primarily that they can record with a speed of at least 90fps and being colour cameras. Temporal resolution is considered a lot more important when working with "Racing" drones since at fast speeds it could become a problem if the drone moved half a meter per image. Since the drone was suppose to include the best possible equipment for the budget, two cameras running 720p@120fps were chosen for their good specifications and driver support for the Xavier NX. As an added bonus they also feature global shutter which removes possible distortions caused by fast moving objects.

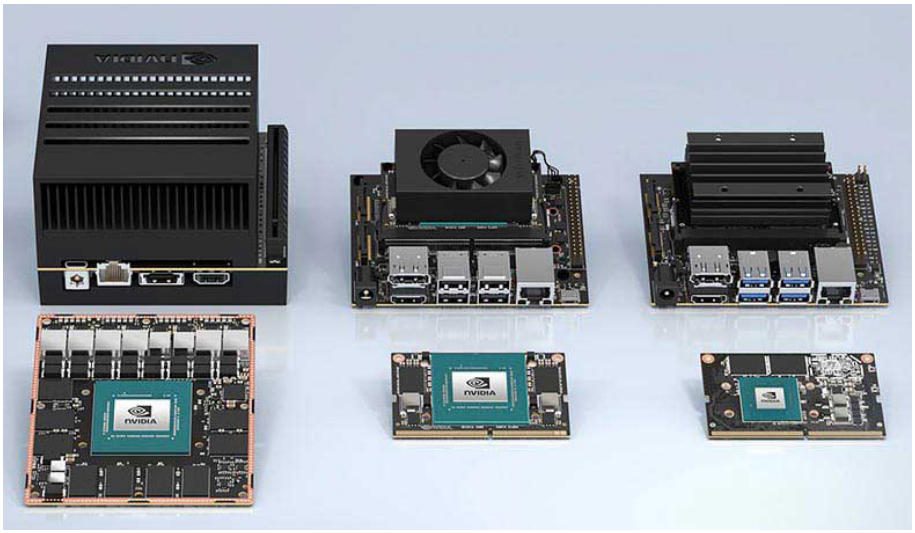

Figure 6: From left to right, Nvidia Jetson- Xavier AGX, Xavier NX and Nano. Source: Nvidia

4 Discussion

While working on the drone project, it was apparent that the goals I've set myself might have been way beyond a 5 month thesis, not to mention a certain pandemic that didn't contribute to making this easier. Regardless of the final result, there was a lot to learn from creating a whole drone from scratch. A lot of careful planning went into designing the drone body since once the drone was printed and assembled, it would take considerable work to create a new prototype. Even so, the final result still ended up having a lot of problems that was missed while trying to account for so many other variables. I do feel it is important to mention these to not only for self reflection but also give insight to future students continuing this project.

The first step was to 3D sketch the included electronics as the dimensions and bolt hole positions of these would need to be known. Most components had accurate 3D models that could be found online at the manufacturer or third party, this helped visualise how they could be fitted in the drone. While sounding easy to just place them, selecting a spot for all the parts prior to having a basic model was a challenge in itself. The obvious was to first account for the centre of mass which is to try and stack the heavier objects like the Nucleo and Jetson on top of each other, this did require some angled standoffs to be able to secure the Jetson in place. Another consideration was that the ESC and cameras do not have proper mounting options and to make matters worse, they have tiny components near the edge of their PCB's. To account for that, a special slot and clamp had to

be designed as to make sure they didn't rattle around inside the drone. The final configuration of the components can be seen in Figure 7 where the top and bottom main body was removed.

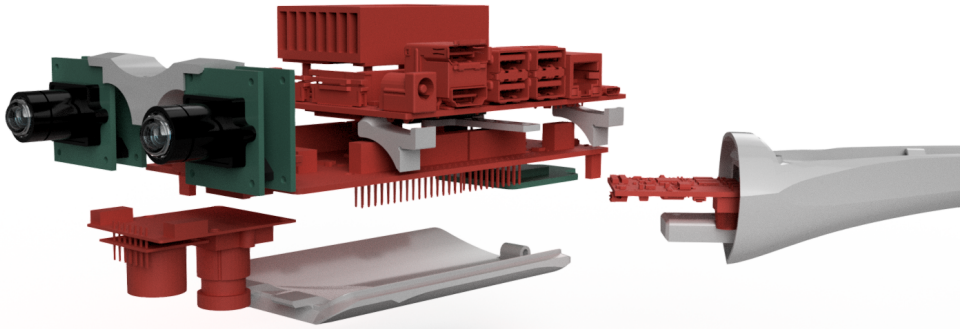

Figure 7: 3D rendering of component fitting, red models were obtained from each manufacturer and the rest was created

While it would have been easier to make the drone in as few parts as possible, there were also some hard limits. The plan was to print this on my personal 3D printer, which has a max build volume of 25×21×21 cm (Prusa MK3), this meant that the arms had to be made as separate parts, adding a fairly obvious point of weakness. This was put some extra consideration into, too much in fact so that some obvious points were missed in the process. This ended up adding more time in redesigning the arm slots, the final version can be seen in Figure 8.

A 3D printer work on the basis of melting a spool of filament to just the correct temperature and layer-by-layer constructing the part upwards, this creates an internal structure similar to wood which also means it has similar strength properties as wood. While the arms could have been created vertical to reduce overhangs, it would make them extremely weak to bending on their "long" side, this applies to all 3D printed parts and was an added considerations. Another aspect of 3D printing is that parts that are midair as seen from the origin (build plate) or in general have too steep angles will need support material which are temporary structures added to hold up the parts. These support structures will make any visible surface rough or crude

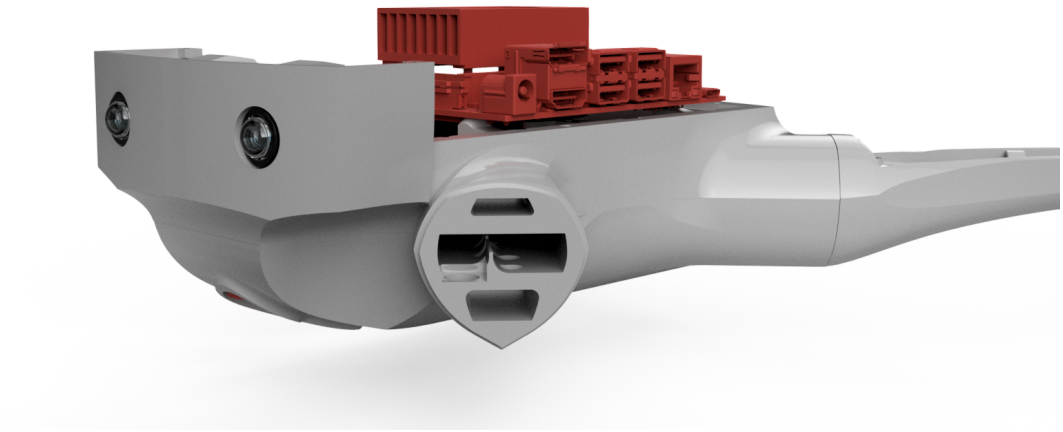

Figure 8: 3D rendering of drone arm slot

compared to those that didn't need support as well as adding to the printing time considerably if there is a lot of overhang. The properties of filament is also important, the one used for the drone was PET-G and one of the downsides is the extremely good adhesion to itself, which is not good for removing the support material, hours were spent removing it in the final build. It would have been easier to print the drone in a more user-friendly filament like PLA, which weakness compared to PET-G is mainly strength. Another key component is how to assemble and secure the parts to each other, using bolts there are several ways to add bolt threads to printed parts. Considering early prototypes it would probably be more efficient to let the boltholes be smaller than the bolt so it would self-thread into the printed parts rather than creating extra holes to slot in square nuts like I did. For future reference I would probably use a hole tap to make the hole threads and make sure that the wall thickness is a bit larger around those holes, this would save a lot of time used making space for all the rectangular nut holes as it should only be needed on places were a significant load is put on the threads.

When using 3D printers the error margin you have can vary from every printer depending on the printer or filament imperfections. The drone was printed with a layer-height of 0.2mm which means that any holes printed vertically would have resolution in the z-direction of 0.2mm. This is also different for the x and y direction as the nozzle diameter was 0.4mm, focusing on squishing each layer to 0.2mm, it can be hard to guess the real error margin from just the software alone though it does try to resolve outer edges as precise as possible. From manually measuring I found that for my

printer I needed to make 3mm holes 3.15mm in order for the M3 bolts to enter without resistance. It is worth noting that the type of filament used can also affect the error, as some filaments shrink slightly when cooled down to room temperature (from +250°C). The total print time for the finished drone was around 45 hours with parts designed and positioned such that it would generate the least amount of support while still trying to maintain structural integrity, this is shown in Figure 9.

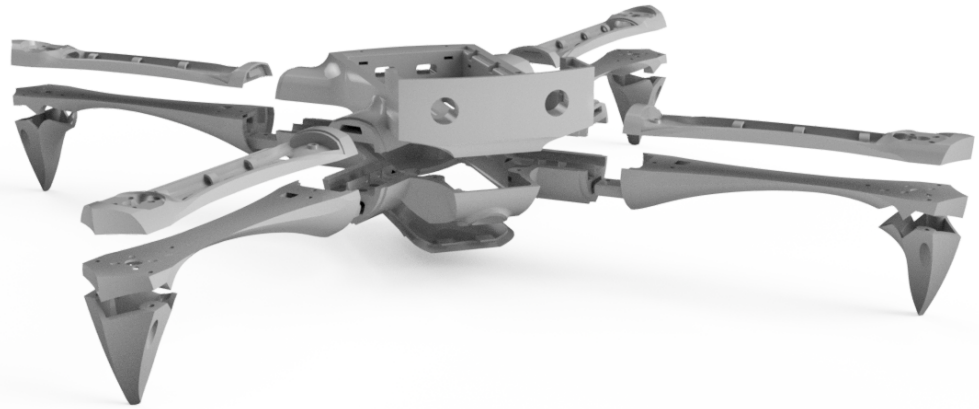

Figure 9: Exploded view of each individual 3D-print ready parts

Cable management was something I tried to account for early in the design process, several notches and holes were created to fit cables into. This looked fine on the 3D design and all the way up to actually soldering the wires. I did want to use the same type of elastic cables with thin insulation that came attached with the motors to save space, but since I was approaching the deadline I did not have the luxury of finding and ordering these special type of cables. As such, the only ones available were stiff and bulky, making assembly quite challenging. In hindsight I should have 3D modelled the cables to get a better indication of how it would look like so I could make more room for them. A comparison of the 3D model and the real model can be seen in Figure 10.

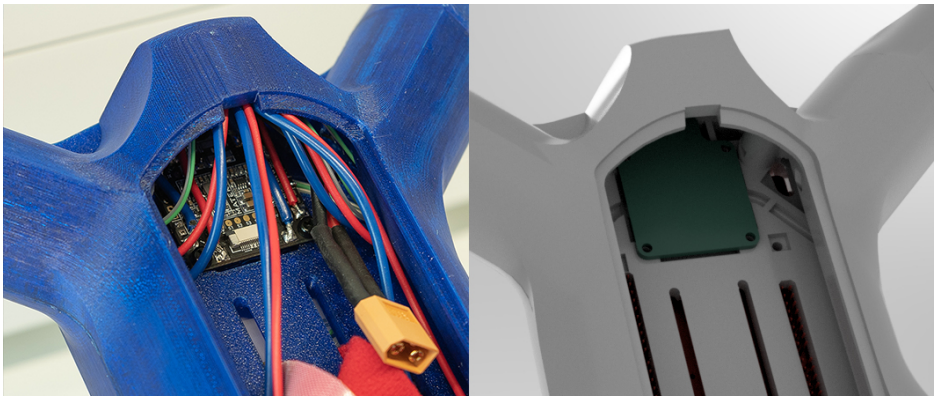

Figure 10: Drone cable mess

5 Conclusion

The process of designing a drone can be thought of like trying to solve a 3D jigsaw puzzle with an infinite number of solutions. Trying to visualise and questioning "what is the best placement?" and "Will this even be possible to assemble?" was a thought that kept bugging me throughout the design process. I did realise that I am an engineer, not an artist and that the process of designing a finished product is usually done over several iterations of the design, not just 5 months. Looking back at the design process, I should have created a simple flat model that all the electronics could be placed on, preferably making it in such a way that all the electronics could be removed without cutting any wire like would be required to disassemble the current prototype.

6 Future Work

In the process of creating the drone, a lot of features and planned implementations had to be set aside in order to get the basic functionalities working. There were also discoveries along the way which should be explored when moving forward with the autonomous racing drone project. Some important technical notes regarding continuation of this drone project can be found in Appendix B

6.1 FCU

In order to read sensor data from encoders, hall-effect or other PWM-generating sensors, it is possible to use some of the MCU's built in timers in reverse (input capture mode). This opens up a non-interrupt based way to read RPM from the motors and this method should be explored further. Another large programming task is to test out RTOS which STMCube has

built-in support for [8]. Both the FCU and ESC's has a built in debugger, but this requires USB connection to each of the 5 components, it is instead possible to STM's ST-Link which is a external debugging tool. This will require some extra cables running to each component but in return it would be possible to have them all placed neatly in one place for faster debugging. This would also eliminate some of the restrictions that comes to the design as it had to account for the USB cable being able to fit in each of the components.

6.2 ESC

In hindsight the choice of ESC should have been a prebuilt bundled with motors. The main issue with the selected ESC is that it requires tuning with whatever motor is selected and given that the FCU is in early prototyping it is unnecessary difficulty added to have the task of tuning several components at the same time. The STM32 ESC should have been added as a next step in the drone project once the FCU is fully functional with basic flying functionality.

6.3 IMU

The included IMU had several features that required the user to register on the manufacturers website and download driver codes written in C. These are not beginner friendly codes and will take some time to fully comprehend. The manufacturer of the IMU does briefly mention some special features like outputting orientation quaternions using their DMP^{TM}(Digital Motion Processor). The basic code currently written to test the IMU functionality should also be expanded to see if using the built in buffer to reduce the amount of calls made to the IMU would affect overall system performance. The built in digital low pass filters are also something that should be explored further by doing several controlled comparison experiments. The white bracket in Figure 5 should be redesigned slightly as it does not fit perfectly due to some pins on the FCU. It is also desirable that the IMU is as close to the centre of mass as possible, this needs to be measured first.

6.4 Power spikes

As the motors draw several Amps of current when they reach their max load usually referred to as A_{peak}, further research and testing should be made with in regards to how this could cause damage to other components. There is also the introduction of noise generated from the switching frequency of the ESC's and any ultrasonic distance sensor if added, the possible implications this may cause on other sensitive components is currently unknown, further testing is required.

References

[1] H. Liu, M. Liu, Y. Ge, and F. Shuang, "Magnetometer calibration scheme for quadrotors with on-board magnetic field of multiple dc motors," in *Proceedings of 2013 Chinese Intelligent Automation Conference*, pp. 409–419, Springer, 2013.

[2] S. A. Brandt, "Small uav design development and sizing," *Handbook of Unmanned Aerial Vehicles. Springer*, pp. 165–180, 2015.

[3] J. Winslow, M. Benedict, V. Hrishikeshavan, and I. Chopra, "Design, development, and flight testing of a high endurance micro quadrotor helicopter," *International Journal of Micro Air Vehicles*, vol. 8, no. 3, pp. 155–169, 2016.

[4] K. N. Mogensen, "Motor-control considerations for electronic speed control in drones," *Analog Applications Journal [online]. Texas Instruments*, 2016.

[5] D. Wilson, "Motor control compendium," *URL: http://focus.ti.com/download/trng/docs/c2000/TI_MotorControlCompendium_2010.pdf*, 2011.

[6] "STM high performance mcus." URL: https://www.st.com/en/microcontrollers-microprocessors/stm32-high-performance-mcus.html#overview. Accessed: 01-25-2021.

[7] T. Andersen, "Filtering and control for high-performance autonomous drone racing," *Technical Report*, 2021.

[8] "STM developing applications on stm32cube with rtos." URL:https://www.st.com/resource/en/user_manual/dm00105262-developing-applications-on-stm32cube-with-rtos-stmicroelectronics.pdf. Accessed: 04-25-2021.

A Digital Attachment

The digital attachments include all the 3D design project files which are made to work with "Autodesk Fusion 360" as well as the source code for the FCU and IMU written in C-code using STM32CubeIDE.
Drone_sourcecode.zip
Drone_3D_files.zip

B Remaining Work

In order to get the drone functioning the following steps needs to be done. These are according to the current pin-setup that is located in H723_test_04 project that can be altered by opening H723_test_04.ioc while inside STM32CubeIDE.

IMPORTANT:
The way the FCU is currently configured, it needs to first be powered by a 4cell battery (14.8v) or a power supply through the XT60 connection. Afterwards it can be connected with USB on the FCU, this order does not seem to be neccesary with the ESC. There seem to be some issues in the enumerator if the code currently on the FCU is bugged it might not want to debug so just redo the procedure, it might even be win10 so not sure what caused this rare behaviour.

IMU:
The I2C2 pinout is shown in one of the tables in the final report. The coloured cable attached to the IMU (Sparkfun Qwiic cable) needs to be connected to the FCU with the following connections:
black = gnd
red = 3,3v
blue = SDA
yellow = SCL

ESC:
STM has seperate programs to make code generation and debugging easier, this is called "Motor Profiler" and "Motor Control Workbench" First run the Motor Profiler to configure a connected motor, then create a new project in the Motor Control Workbench with the correct ESC board selected with the profile created in Motor Profiler.

FCU:
H723_test_04.ioc needs to be configured with the timers mentioned in the main report, currently only TIM3 is activated and was used to test the PWM generation while connected to an oscilloscope. TIM3 should be set at the correct frequency required by the ESC's so just copy the timer settings over to the four timers mentioned.