Semester Project

Title :	Drone Avionics Board Design
Author :	Quentin Delfosse (EPFL Xplore - Drone Avionics Engineer)
Technical Supervisor :	Arion Zimmermann (EPFL Xplore - Drone Systems Engineer)
Professor :	Pr. Alexandre Schmid (EPFL - BNMS Laboratory)

January 2023

Revision History

Revision	Author	Description	Date
R01	Quentin Delfosse	Document baseline	14.12.2022
R02	Quentin Delfosse	Document finalized	06.01.2023

0 - Abstract

This report introduces the design of an avionics board for UAV applications. As this is the first project to be conducted on the topic within EPFL Xplore, a special emphasis was placed on the research and selection of the navigation sensors allowing for an efficient and precise localization of the drone. Its computing units were also researched and a general hardware architecture was developed to further define the interfaces required within the drone. For debug purposes, additional interfaces were added to the design by means of UART, USB and SPI ports.
A first version of the avionics firmware was then developed to validate the proper functioning of the system. The sensors could therefore be fetched on their I2C buses, and the data transmitted to the user's terminal.
While this board has served its purpose by proving the feasibility of the study and the proper functioning of the design, there is still room for improvements, especially regarding its safety and interface modules. Additional design considerations for a second iteration are therefore mentioned at the end of this study.

Table of contents

1 – Introduction

For the past 50 years, the development of our robotics technologies for space applications has allowed us to reach further into space and to learn more about our universe than ever before. Taking the form of planetary robots (a.k.a. rovers), the space exploration programs have evolved and are now planning for a lasting installation of mankind on the Moon.

In the scope of this new chapter of space exploration, the EPFL Xplore project was launched. As a student-led space robotics association, it aims at developing autonomous and modular rovers to conduct lunar mission simulations on Earth. These missions include:

- Maintenance and operation of a control panel
- Fast mapping of a challenging terrain
- Soil sample retrieval and onsite analysis
- Soil probing and retrieval of objects on the field

To support the navigational needs of our rovers, numerous iterations of avionics module have been developed. Nevertheless, as they only rely on ground-based sensor perception, our current navigation strategy does not allow for post-obstacle vision, leading to a non-optimal control. Adding an aerial point of view to the control of our rovers is expected to improve their behavior in such situations. A drone project was therefore launched to answer this need. Its avionics module is presented in this report.

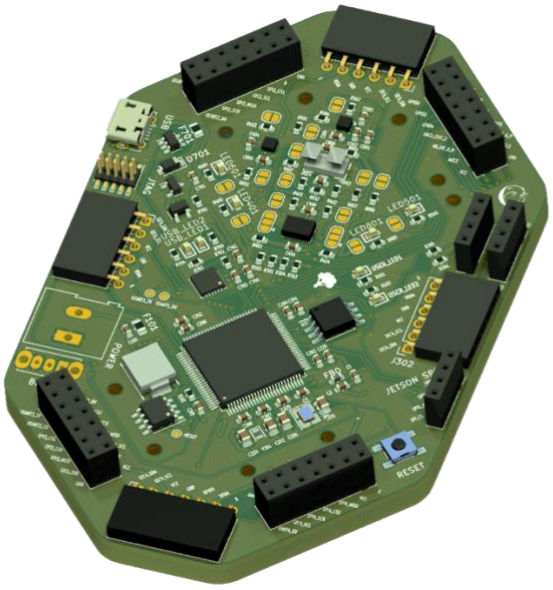
2 – System requirements

The avionics board of the drone system has two main requirements. Namely, it shall:

- Provide the necessary sensor data for course planning and control of the drone.
- Provide external interfaces to communicate with the other systems of the drone (power supply, main computer, telemetry) as well as with the ground station.

These requirements will serve as a guiding principle for the design of the different stages of the board. The more in-depth requirements will be mentioned in the design sections.

3 – System specifications



To comply with the main requirements stated hereinabove, the avionics board relies on several stages.

First, the computer stage orchestrates the communication between the sensors and the external systems. It is based on a STM32H750 microcontroller equipped with an Arm Cortex M7 core. External oscillators and flash memories were also added to the design to improve the board's performance.

Table 3.1: Clock characteristics

Parameter	Comment	Min.	Typ.	Max.	Unit
Internal oscillator	ST32H750 oscillator	0.032	-	480	MHz
External oscillator 1	External oscillator	-	48	-	MHz
External oscillator 2	External oscillator	-	32.768	-	kHz

Table 3.2: Memory characteristics

Parameter	Comment	Typ.	Unit
Internal RAM	ST32H750 RAM	1	MB
Internal flash	ST32H750 flash memory	128	KB
External flash	External flash memory	128	MB

Regarding its power stage, the board is designed to work at a nominal voltage of 3.3V. A safety module was added to increase its robustness against small input voltage variations (see Table 3.3). It is important to note that this version of the board does not provide reverse voltage protection nor ESD protection on the main power input. Please refer to the safety section (chapter 15) before powering the board.

Table 3.3: Power characteristics

Parameter	Comment	Min.	Typ.	Max.	Unit
Input voltage	Provided by the battery pack	1.1	3.3	10	V
Maximum peak input current		-	-	3	A
Current consumption		0	56	200	mA

Then comes the sensors stage. To provide the operator and the main computer with the position, orientation and information on the surroundings of the drone, the board is equipped with a series of sensors.

Table 3.4: Localization sensors characteristics

Sensor	Comments	Min. Range	Max. Range	Unit	Sensitivity	Unit
IMU	3 axis accelerometer	-24	+24	g	92	µg
	3 axis gyroscope	-2000	+2000	°/s	0.004	°/s
Magnetometer	3 axis magnetometer	-49.152	+49.152	Gauss	1.5 (0.09)	mGauss / LSB (°)
	Temperature	-40	+85	°C	3.9	m°C
Barometer	Pressure	300	1250	hPa	0.03 (0.25)	hPa (m)
	Temperature	-40	+85	°C	0.5	°C
GPS	Up to 4 concurrent GNSS (Beidou, Galileo, GLONASS, GPS)	-	-	-	-	-

Table 3.5: Navigation sensors characteristics

Sensor	Comments	Min. Range	Max. Range	Unit	Sensitivity	Unit
Time of flight	8*8 multizone detection sensor	0.005	4	m	15	mm

Finally, the board provides several communication and mechanical interfaces allowing for an easy integration of the board as part of the overall drone system. The interfaces are listed in the tables below.

Table 3.6: External communication interfaces

Parameter	Comment	Max.	Unit
3* I2C	For external sensors and modules	64	MHz
4* SPI	3 SPIs for external sensors and modules, 1 SPI to communicate with the main computer	250	MHz
3* UART	For external sensors and modules	100	MHz
1* Micro USB	To communicate with a computer (relies on USB/UART converter)	250	MHz
3* ADC	For external sensors and modules	100	MHz
6* GPIOs	For debugging purposes and resets	100	MHz

Table 3.7: Mechanical characteristics – Main Board

Parameter	Comment	Min.	Typ.	Max.	Unit
8* M2 mounting holes	For integration on the drone structure	-	2.2	-	mm
Length		-	107.1	-	mm
Width		-	68.0	-	mm
Weight	Without GPS board plugged	-	29.0	-	grams
PCB Material	Specified by manufacturer (Aisler)	-	FR-4	-	-
Number of PCB layers	Chosen between 2 and 4 according to the manufacturing capabilities of the manufacturer (Aisler)	-	4	-	-

Table 3.8: Mechanical characteristics – GPS Board

Parameter	Comment	Min.	Typ.	Max.	Unit
Length		-	39.70	-	mm
Width		-	19.75	-	mm
Weight	Depending on whether the GPS module is plugged		4.0		grams
PCB Material	Specified by the manufacturer (Aisler)	-	FR-4	-	-
Number of PCB layers	Chosen between 2 and 4 according to the manufacturing capabilities of the manufacturer (Aisler)	-	2	-	-

Table 3.9: Thermal characteristics

Parameter	Comment	Min.	Typ.	Max.	Unit
PCB temperature rise	Tested at full load (all sensors on) at 20°C for 30 mins.	0	0.1	0.1	°C

4 – System decomposition

When diving into a new project, the system decomposition phase is always critical to its success. By system decomposition, we refer to its separation into multiple independent subsystems which, thanks to their carefully defined interfaces, can be integrated together to form the final fully functioning system. This decomposition can happen at different levels (system / subsystem / sub-subsystem, etc.). Here, we present two levels of decomposition: the drone level and the avionics level.

4.1 – Drone system decomposition

First, we need to understand how our avionics board will be integrated as part of the overall drone system.

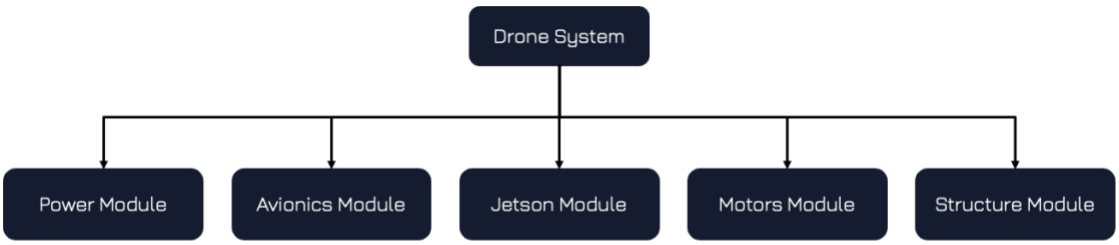


Figure 4.1: Drone system decomposition

The drone is composed of several modules exchanging power and information.

- The power module provides the different supply voltages to all the other modules and measures their power consumption.
- The avionics module gathers navigation data and spatial awareness information thanks to its sensors.
- The Jetson module uses the navigation data from the avionics sensors and the cameras to control the motors of the drone. It also includes telemetry functions to communicate with the ground station (see Figure 4.2).
- The motors module is composed of the Electronic Speed Controllers (ESCs) and the drone motors and propellers.
- Finally, all these systems are mechanically integrated thanks to the structure module.

4.2 – Drone hardware architecture

To better understand the interfaces required for our avionics module, an overall system architecture schematic is required.

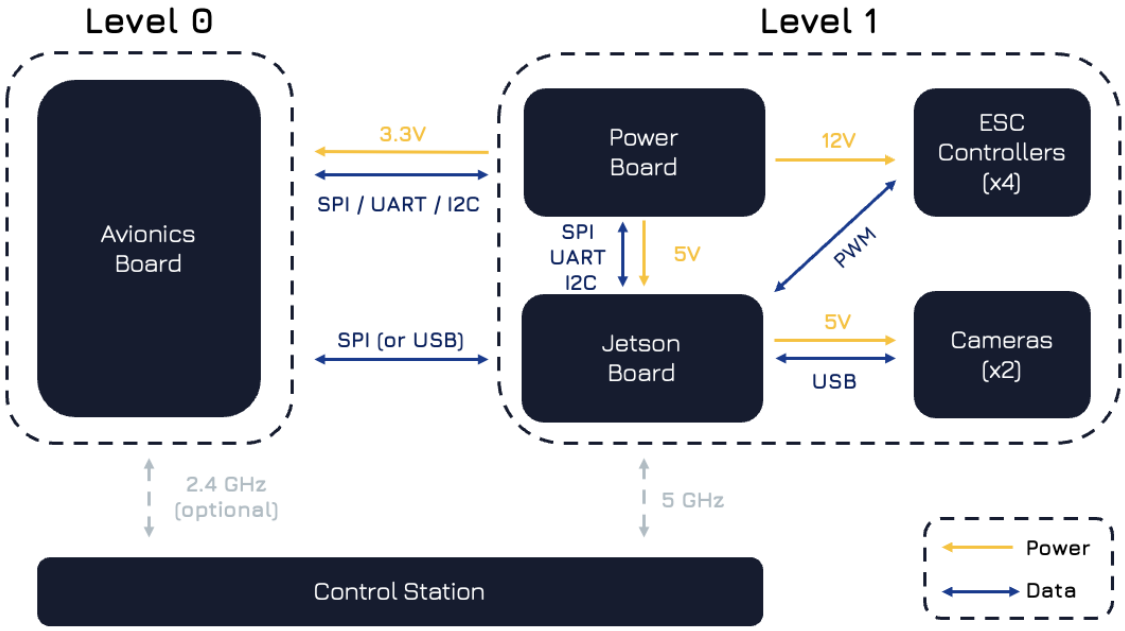


Figure 4.2: Drone hardware architecture

In addition to presenting the power and communication interfaces between the different systems of the drone, this schematic presents their stack-up position. The avionics module is located at the lowest level of the drone (level 0) to allow for ground detection while the other systems are located on the upper level (level 1). An additional level also can be added between the avionics and the Jetson board in case additional sensors are required.

4.3 – Avionics hardware architecture

Thanks to the overall drone architecture, we now have a better understanding of the interfaces required on the avionics board and how this module will be powered and mounted.

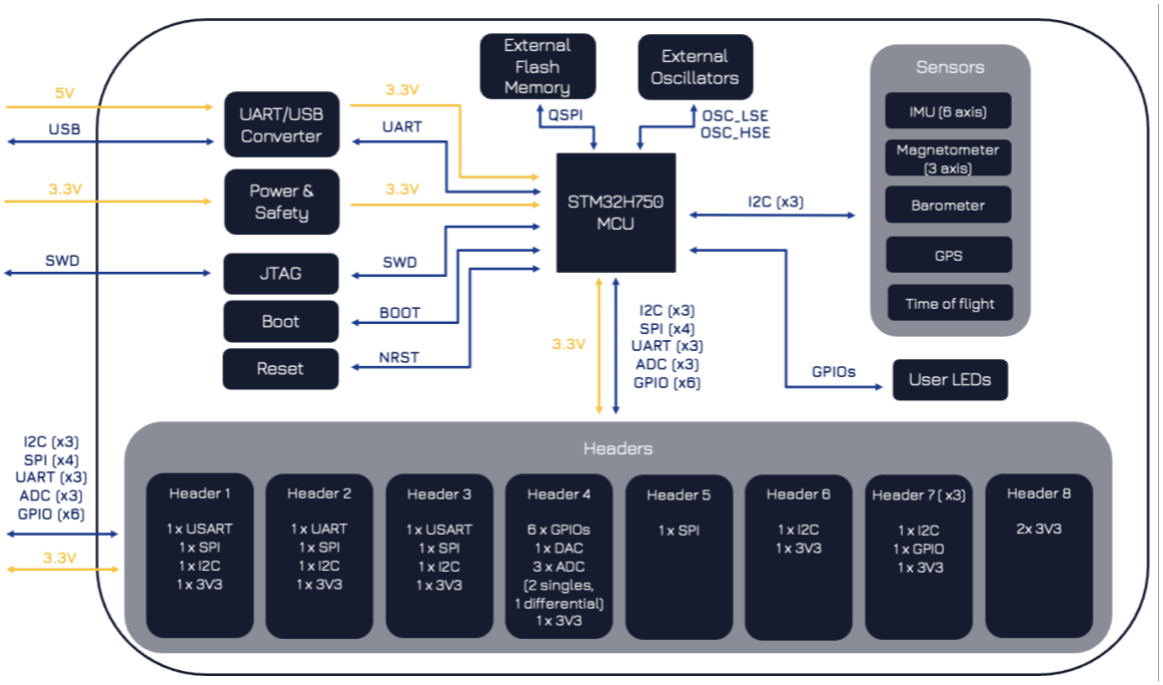


Figure 4.3: Avionics hardware architecture

Those interfaces include the following:

- The avionics board is powered at a nominal 3.3 V input voltage from the battery stage. For debug purposes, a micro-USB connection is also provided, powering the board with a 5 V input before being converted to its 3.3 V nominal voltage.

- The JTAG interface allows the user to flash, debug and reset the microcontroller via a ST-Link or J-Link interface. In case the user wishes to manually reset the board, a reset switch is provided. A boot switch was also added to select the start-up configuration of the microcontroller.

- For external communication purposes, the board includes 8 headers which functions vary between external sensors communication, Jetson communication and external power supply.

4.4 – Avionics schematics overview

The PCB schematics were designed using the KiCAD PCB software. Here, we present the general overview of these schematics.

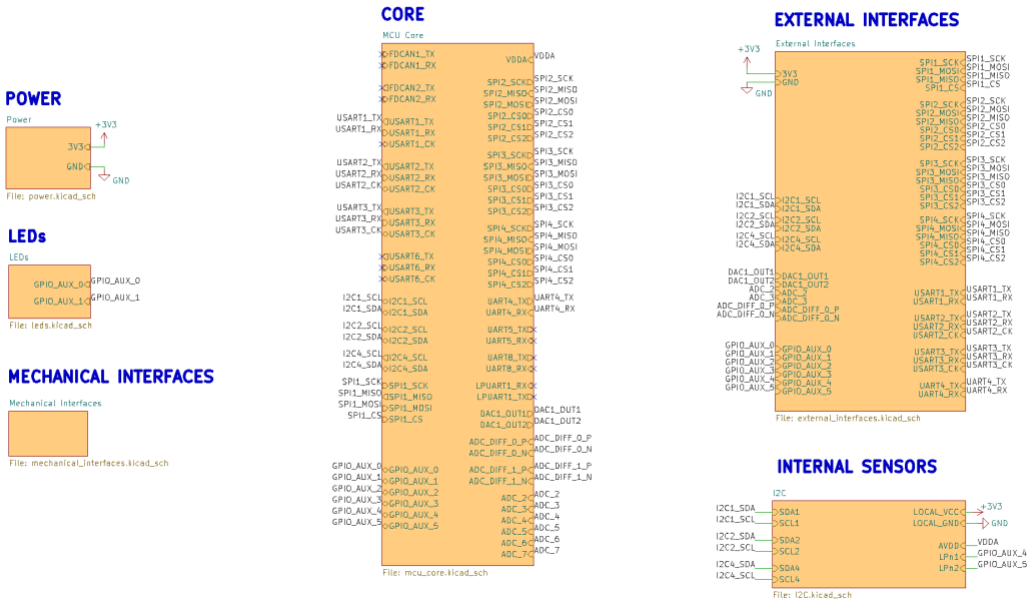


Figure 4.4: Avionics design schematics overview

5 – Computer Stage Design

One of the main objectives of this project is to select the main computer of the drone and to interface it with the necessary navigation sensors. To guide the selection process, some requirements were defined.

5.1 - Requirements

The drone computer shall:

- Be able to run the 2nd version of the Robot Operating System (ROS2)
- Be able to process camera feeds up to 4K resolution
- Be able to run at least 2 cameras in parallel
- Present at least 2 buses for each of the following communication protocols: I2C, SPI and UART
- Present a USB interface for debugging
- Consume less than 10W of power
- Weight less than 40 grams
- Fit within a 10*50*6 mm^3 box
- Be mountable on the drone under 2 minutes
- Cost less than CHF 150

5.2 – Main computer selection

Upon researching a computer fitting such criteria, a few candidates emerged.

Table 5.1: Main computer selection

	Nvidia Jetson Nano Module [1]	**Nvidia Jetson TX2 NX Module** [2]	**Raspberry Compute Module 4** [3]
Dimensions	69.6 x 45.0 x 5.6 mm^3	69.6 x 45.0 x 5 mm^3	55.0 x 40.0 x 4.7 mm^3
Weight	18g	17 g	17 g
Price	CHF 99	CHF 250	CHF 95 (wireless) CHF 90 (standard version)

Nominal power consumption	5 ~ 10 W	7.5 W	7 W
Input power	5V DC	5V DC	5V DC
ROS2 compatible	Yes	Yes	Yes
GPU	128-core NVIDIA Maxwell GPU	256-core NVIDIA® CUDA®	Broadcom VideoCore VI
CPU	Quad-core A57 ARM	ARMv8 (64-bit) heterogeneous multi-processing (HMP)	Quad-core Cortex-A72 ARM v8 (64-bit SoC @ 1.5GHz)
LPDDR4	4 GB 64-bit LPDDR4	4 GB 128-bit LPDDR4 DRAM	Options for 1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM with ECC
eMMC	16 GB eMMC 5.1	16GB eMMC 5.1 Flash Storage	Options for 8GB, 16GB, or 32GB eMMC flash memory
SDRAM	-	-	Options for 1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM with ECC
Computing power	472 GFLOPS	1.33 TFLOPS	13.5 GFLOPS
Camera resolution	Up to 4K	Up to 4K	Up to 1080p
Market availability (by October 2022)	Limited	Available	Limited

On the one hand, the Raspberry Compute Module 4 is an extremely compact and user-friendly computer which comes in multiple versions depending on the project's needs. It is a lightweight, cheap and low-power solution that can be used as an integrated solution to a wide variety of environments.

On the other hand, the Nvidia brand is well known for the machine learning capabilities of its computers. Having up to 1.33 TFLOPS of calculating power would come in handy for the development of our computer vision algorithms. Plus, the possibility to process 4K camera feeds as required is an advantage that the Raspberry module doesn't have.

In the end, given the strict requirement for 4K resolution processing, the Nvidia brand was chosen. While the difference between the two Nvidia versions only resides in a higher computing power for the Jetson TX2 module, our project would not be impacted by smaller specifications at such high performances already. The Nvidia Jetson Nano would therefore have been the optimal choice for our project. Nevertheless, due to limited market availabilities for the Nano version, the TX2 module remained as the only fitting solution.
As the budget requirement was not satisfied, a meeting took place to reallocate some financial resources and allow for the acquisition of the computer.

5.3 – Secondary computer selection

As mentioned in the architecture section, the drone operates on two computers: a main one (the Nvidia Jetson TX2 Module) and a secondary one (the STM32H750).

The choice for a separated processing architecture is justified by:

- A fast implementation of the STM solution for sensors data acquisition (no external board required to flash the computer)
- Prior knowledge of the team with STM applications (Xplore's rovers all have avionics modules running on the STM32H750 microcontroller)
- Preliminary tests to enable a Jetson I2C and UART buses for sensors data acquisition failed
- Possibility to communicate via USB between the STM32 and the Jetson in case SPI buses can't be accessed.

Due to the short duration of the project (6 months to design, 6 months to integrate and test), the 2-computers architecture was selected as the fastest result-producing solution.

Note that in this report, we will only focus on the STM32 computer board as it is the one in charge of the avionics tasks.

5.4 – Design

In the section, we introduce the designs of the core of the avionics board, namely the secondary computer or STM32H750 microcontroller.

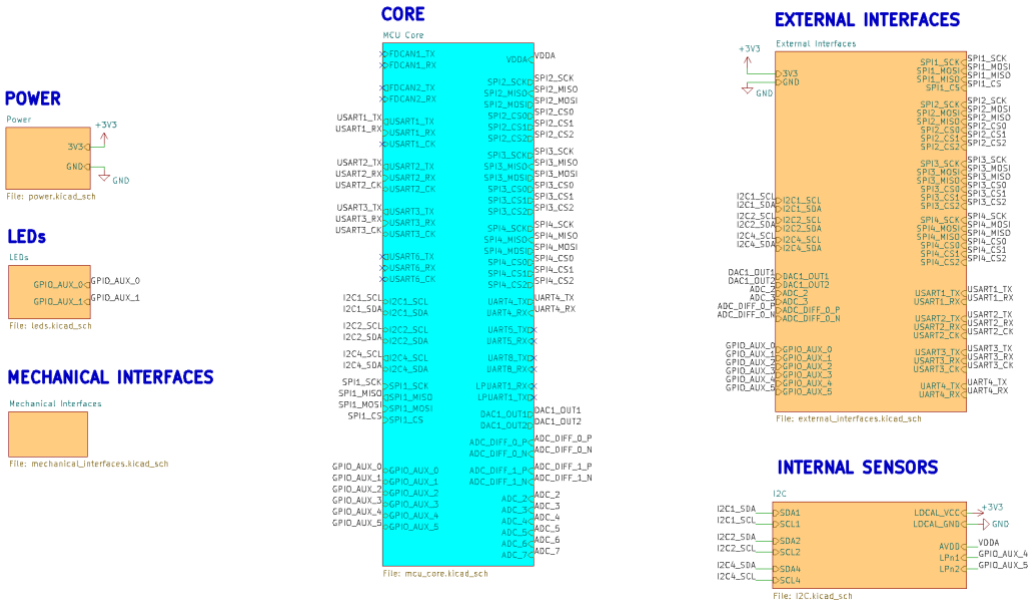


Figure 5.1: Drone system decomposition - Core

5.4.1 Decoupling capacitors

According to the STM32H750 datasheet (p.95) [4], the power line must be decoupled with as many 100nF ceramic capacitors as there are power inputs on the microcontroller in addition to a 4.7µF bulk capacitor. The microcontroller shows 6 VDD or VBAT inputs, hence the following decoupling configuration.

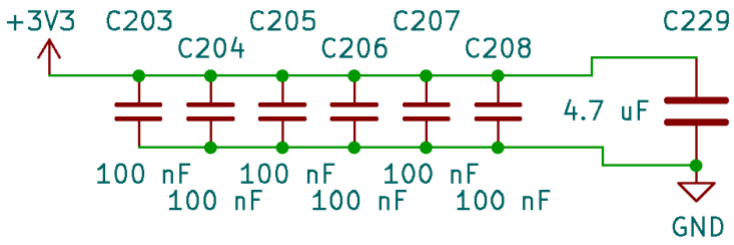


Figure 5.2: Decoupling capacitors

5.4.2 Voltage regulation

When using the internal voltage regulator of the STM32 chip, the datasheet asks to connect two 2.2µF capacitors to the VCAP inputs to stabilize the power line (p.206) [4].

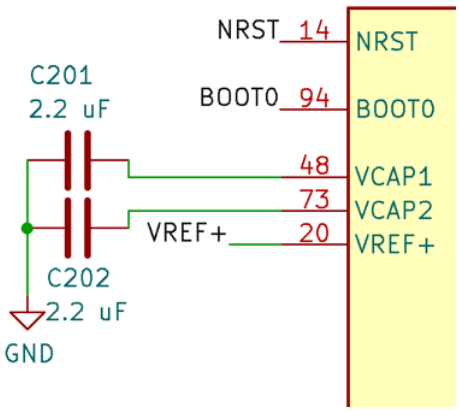


Figure 5.3: Voltage regulation

5.4.3 Ferrite inductance

To filter high-frequency noise on the analog power line (VDDA), a ferrite inductance coupled with decoupling capacitors can be used as a LC circuit (see application note AN2834 [5], p.24).
According to the application note AN2834, the power side of the ferrite must be decoupled with a 1µF to 10µF capacitor while the microcontroller side must be decoupled with two capacitors in parallel, one of 1µF and the other of 10nF. To further improve the high frequency noise reduction and reduce the size of the bill of materials, the latter was changed to 100nF.

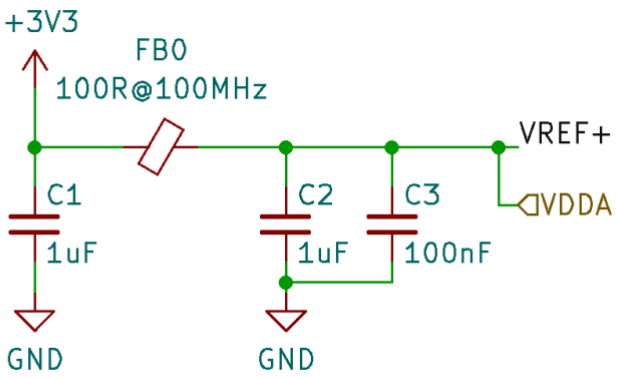


Figure 5.4: Ferrite inductance

5.4.4 SWD/JTAG

Whenever the user wants to load a new software on the microcontroller, it is necessary to go through its SWD debug lines. A ST-Link or J-Link adapter is also used to interface the microcontroller with our computer. Here, we used a ST-Link adapter to load our application onto the MCU. To comply with such device, a 2x5 male header connector was used.

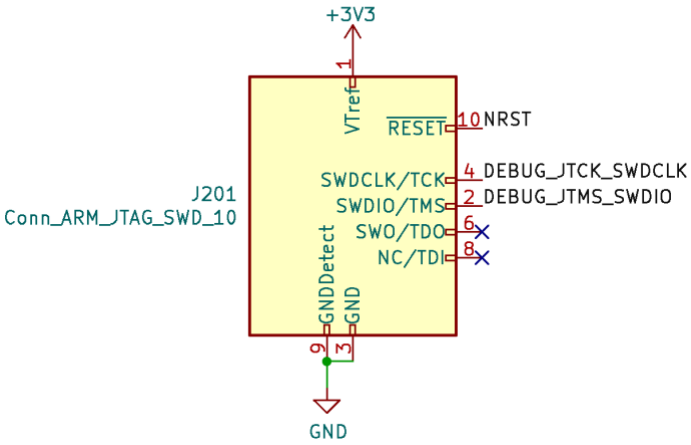


Figure 5.5: SWD Connector

Note that the SWO pin was not connected in this first version, but it could be useful to implement it in the second iteration to easily send debug messages via the ST-link debugger.

5.4.5 Reset

The reset pin of the STM32H750 is an active low pin. Connecting it to the ground will result in a reset of the microcontroller.

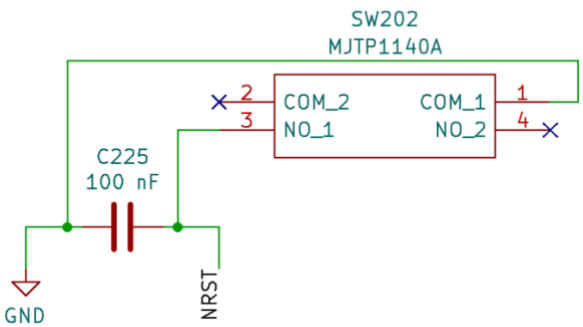


Figure 5.6: Reset button

A 100 nF capacitor is used to protect the line against high frequency switching (see the STM32H750 datasheet [4], p.243).

5.4.6 Booting

After each reset of the board, the MCU will look at the flash memory. If it is blank, the internal bootloader will kick in and start the MCU under default settings. If not, the user code present on the flash will run.

Pulling the BOOT0 pin high will bypass this verification and the bootloader will assume that the flash is blank.

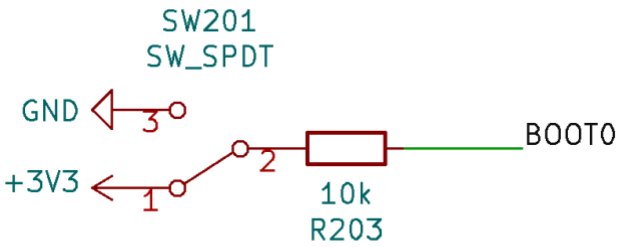


Figure 5.7: Booting switch

The 10kΩ resistor was selected based on the application note AN5307 [6] (p.27).

5.4.7 External oscillators

To improve the board's performances, high precision crystal oscillators can be used in addition to the internal clock of the MCU. The suggested frequencies of these oscillators are 32.768 kHz (Low-Speed External oscillator or LSE) and 40 MHz (High-Speed External oscillator or HSE) according to the MCU datasheet (p.28) [4].

To implement such oscillators, the value of the decoupling capacitors is computed according to the application AN2867 (p.12) [7]:

$$CL - CS = (CL1 * CL2)/(CL1 + CL2)$$

Setting CL1 and CL2 as equal then leads to:

$$CL1 = CL2 = 2 * (CL - CS)$$

With:

- CL the load capacitance of the oscillator (defined by the manufacturer)
- CS the stray capacitance of the oscillator (defined by the manufacturer)
- CL1 and CL2 the decoupling capacitances (to be computed)

LSE decoupling capacitances

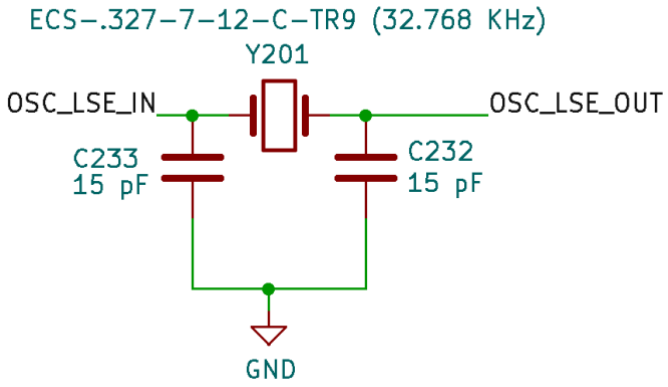


Figure 5.8: Low Speed External Oscillator

According to the crystal's datasheet [8] and the ST community forum [9], we have:

- CL = 7 pF
- (CL – CS) = 5.8 pF

Therefore, we can extract:

$$CL1 = CL2 = 11.6\,pF$$

The closest market available value for these capacitances is:

$$CL1 = CL2 = 15\,pF$$

HSE decoupling capacitances

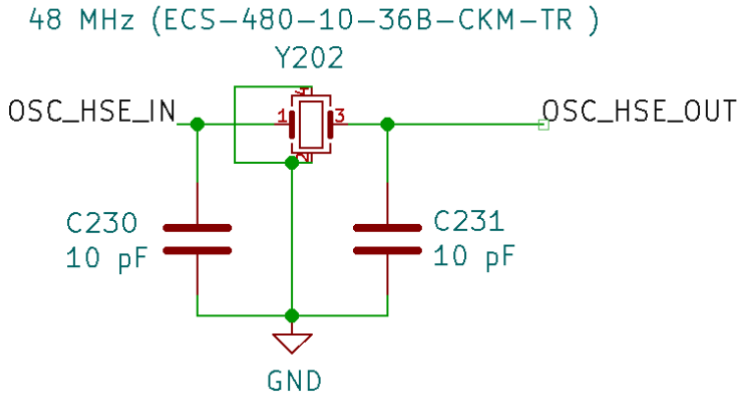


Figure 5.9: High Speed External Oscillator

This time, the crystal's datasheet [10] gives:

$$CL = 8\,pF$$
$$CS = 3\,pF$$

This leads to:

$$CL1 = CL2 = 10\,pF$$

5.4.8 Flash Memory

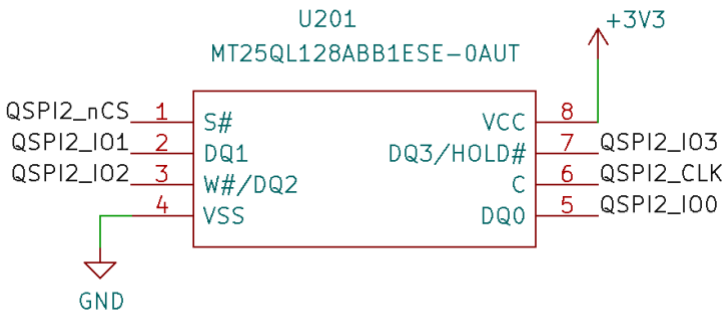


Figure 5.10: Flash memory

The external flash memory interface of the STM32 makes use of Quad-SPI protocol allowing for high-speed communication up to 133 MHz [11] with the microcontroller.

Given the high amount of sensors data to be logged, the internal flash memory would not be sufficient. Supposing 100 bytes of data per acquisition and an I2C bus speed of 2 MHz, the sampling frequency would result in:

$$Facq = 2'000'000 / (100 * 8) = 2'500\ samples/s$$

A flash memory of 128 Kbytes would then be able to log the data during:

$$T = 128'000 / (100 * 2'500) = 0{,}512\ seconds$$

Using an external flash memory of 128 Mbytes would instead allow for:

$$T = 128'000'000 / (100 * 2'500) = 512\ seconds\ (or\ 8\ mins\ and\ 32\ seconds)$$

This time is more than the estimated flight time of the drone and would be sufficient to log all the sensors data during the flight time. A smaller logging frequency can also be used to add a safety factor and avoid overloading the flash memory.

6 – Power Stage Design

In this part, we will look into the main power input design and the safety that it provides to the board.

6.1 – Requirements

Regarding the power stage, it shall:

- Output a nominal voltage of 3.3V (±0.1V) to the board
- Prevent any other voltage to pass the safety circuit
- Work under input voltage variations between at least 0V and 5 V
- Protect the board from current peaks above 3A lasting more than 1 µs

6.2 – Design

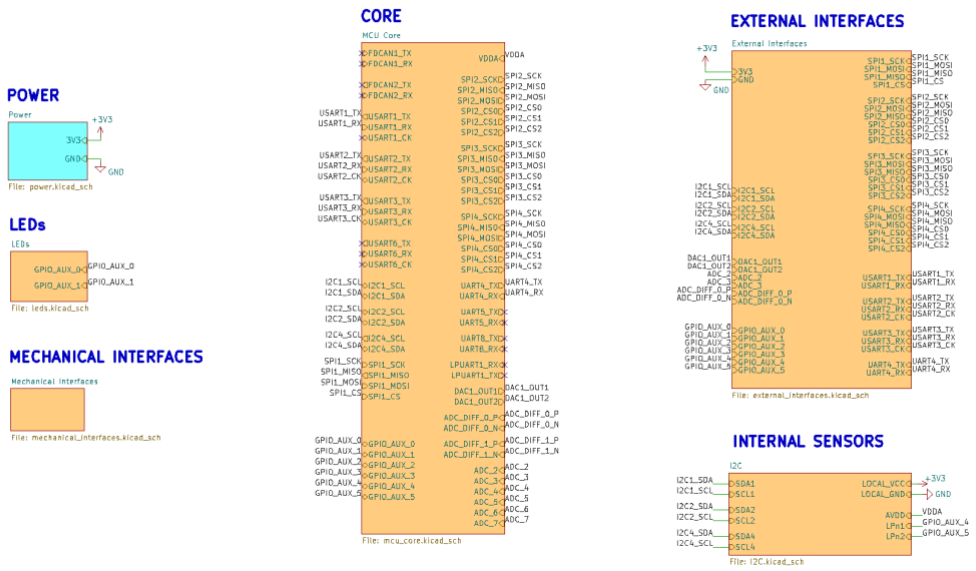


Figure 6.1: Drone system decomposition - Power

To meet the criteria stated in the previous section, the power stage relies on several components which are presented below.

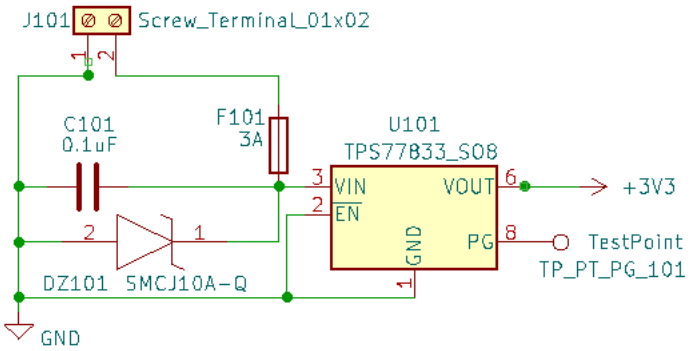


Figure 6.2: Input power circuit

6.2.1 Low dropout voltage regulator

First, a low dropout linear voltage regulator (TPS77833) [12] ensures that the voltage used to power the board is fixed to 3.3 V. More precisely, the output voltage of the chip under loads ranging from 10 µA to 750 mA and a nominal input voltage supply varies between:

$$3.234\,V \;<\; Vout\;(=Vcc) \;<\; 3.366\,V$$

This shift from 3.3 V is still within the requirements range of ±0.1 V and validates the selection of this model. Further, the chip can output the nominal voltage of 3.3 V under the following conditions:

$$1.1\,V \;<\; Vin \;<\; 13.5\,V$$

Any input below such conditions will result in the voltage dropping to 0 V. Likewise, any input voltage above 13.5 V will result in the destruction of the chip and might impact the board. This justifies the need for a protection ahead of the chip making sure to block any voltage above this limit.

6.2.2 Zener diode

To protect the voltage regulator, a Bourns SMCJ10A-Q Zener diode was used. Indeed, this component shows a reverse voltage of 10 V which ensures that no voltage above 10 V would supply the chip.

Nevertheless, in case of input voltages higher than 10 V, the diode would have to dissipate the energy coming from the voltage drop by heating up. To prevent it from overheating and being damaged in case of high voltage gaps, a fuse was added to the design.

6.2.3 Fuse

The selected fuse is a Vishay MFU0805FF03000P100 allowing for rated currents up to 3A [13].

To make sure that the fuse will blow before the Zener diode, one can look at the maximum dissipated energy by both component for a defined period (or pre-arc time). According to its manufacturer, the diode is able to sustain the following maximum peak pulse current:

$$Ipp_{\ zener} = 88.3\ A\ (for\ 1\mu s)$$

Looking at the fuse specifications, the selected model shows a pre-arcing of:

$$PA_{\ fuse} = 0.0227\ A^2s$$

Plotting it on its pre-arcing vs pre-arc time graph (Figure 6.3), we find that for a pre-arc time of 1 µs, the diode specifications (plotted in red) exceed those of the fuse (plotted in blue) by a factor 4.

$$PA_{\ fuse} = 0.0227\ A^2s < PA_{\ zener} = 0.1\ A^2s$$

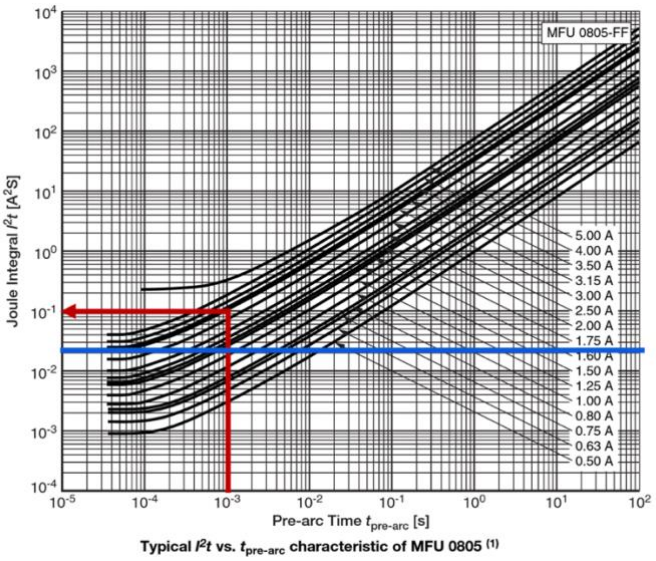


Figure 6.3: Fuse power dissipation

The fuse will therefore blow before the Zener diode and ensure that the system is safe for input voltages above 10 V.

6.2.4 Decoupling capacitors

As mentioned in the design section of the main computer, for any power inputs, a decoupling capacitor of 100nF needs to be used to filter high frequency noise on the line. The input of the TPS77833 voltage regulator was therefore decoupled with such a capacitor.

7 – LED Stage Design

7.1 – Requirements

The avionics board shall:

- Provide visual indication of its overall power status
- Provide visual indication of its sensors power status

7.2 – Design

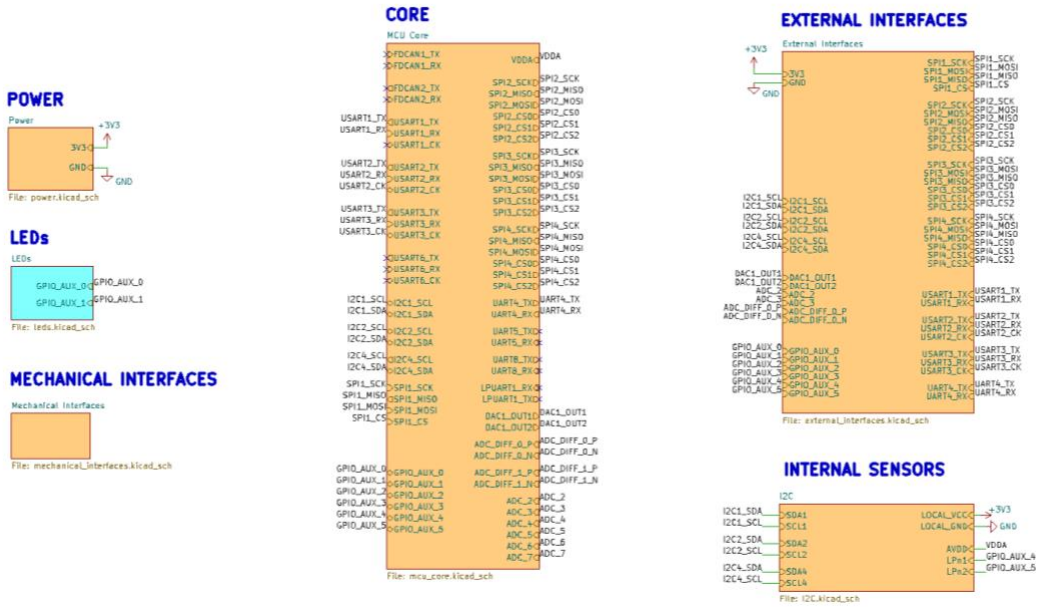


Figure 7.1: Drone system decomposition - LED

All the LEDs used are powered either from the 3.3 V supply of the avionics or by user controlled GPIO pins. The 2 following LED models can be found on the board.

Table 7.1: LEDs

Brand	ams OSRAM	ams OSRAM
Model	LB Q39G-L2OO-35-1	LT Q39G-Q1OO-25-1
Color	Blue	Green
Forward voltage	2.85 V	2.85
Forward current	5 mA	5mA
Wavelength	470 nm	530
Luminous intensity	28 mcd	71 mcd

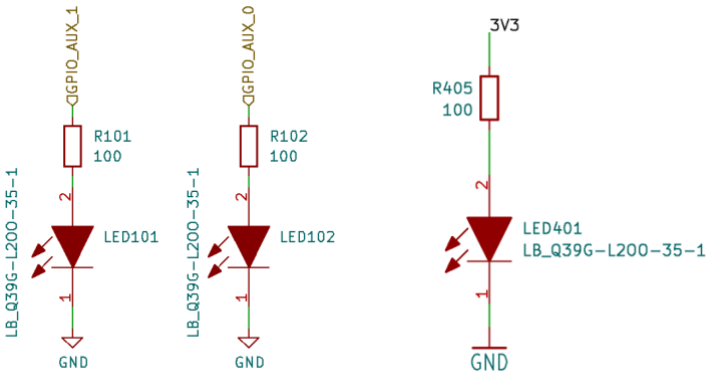


Figure 7.2: LED power configurations

First, we need to make sure that the power supply and GPIO pins can provide enough current to turn the LEDs on. The limiting factor would be the GPIO pin current.

$$I_{max}(GPIO) \ = \ 25\ mA$$

Given that the maximum current needed by the LED does not go above 5 mA, the pins are more than capable to power the LEDs. The series resistance needed on the line can then be computed as follows:

$$R = \frac{V(GPIO) - V_f(LED)}{I_f(LED)}$$

$$R = \frac{3.3 - 2.85}{0.005} = 90\ Ohms$$

The closest available resistor value is $R = 100\ Ohms$.

8 – Sensor Stage Design

8.1 – Requirements

The sensor stage of the drone shall:

- Allow for a localization of the drone within 5 cm in all directions and 1° in all orientations
- Allow for ground detection up to 1 m below the drone

8.2 – Sensors Selection

8.2.1 Localization sensors

When dealing with localization tasks, the research conducted has shown that most mobile platforms rely on the sensors stated below:

- Accelerometers
- Magnetometers
- Gyroscopes
- Barometers
- GPS (if applications allow)

Nevertheless, the main difference that can be observed between these platforms resides in how they integrate these sensors together. In our case, we could be looking for a IMU including an accelerometer and a gyroscope or a more integrated version of it with an additional magnetometer.

Due to supply shortages on the latter, a 3-axis magnetometer was used in parallel with a 6-axis IMU (including a 3-axis accelerometer and 3-axis gyroscope). The final sensors choice is presented in the table below.

Table 8.1: Localization sensor specifications

Sensor	Supplier	Model	Comments	Reference
IMU	Bosch	BMI088	Including 3-axis accelerometer and 3-axis gyroscope	[14]
Magnetometer	ST	IIS2MDC	-	[15]
Barometer	Bosch	BMP390	-	[16]
GPS	uBlox	MAX-M10S	-	[17]

8.2.2 Navigation sensors

In the scope of the competition, the teams need to be able to control their drones from a closed tent without any line-of-sight feedback. Therefore, to ensure that the operator and the main computer get all the necessary information to safely fly the drone in manual or autonomous control, some proximity sensors are required. From that statement, some technologies can be identified.

Table 8.2: Navigation sensor selection

Technology	Optical	Time of flight	Ultrasonic
Advantages	Very large field of view (up to 360°) Works on all surfaces Surface texture detection	High resolution (cm) Low cost (<10 CHF) Fast response time (light speed) Long range measurement (~5m) Large field of view (up to 45°) Lightweight (~ 1 gram)	Works on most materials and surfaces Medium field of view (up to 30°)
Drawbacks	Complex processing High power consumption Heavy (>50 grams)	Weak against highly reflective surface (bright surfaces) Unstable in smoky or foggy environments	Slow response time (sound speed) Medium range measurement (1 m) High cost (>30 CHF) Heavy (>10 grams)

Given the complexity that comes with detecting obstacles from a camera feed, the optical solution was dropped in the design of the first version.

The weight being the most important factor for our design, we computed the ratio of the weight over the field of view of both sensors.

$$Fultrasonic = \frac{10\ grams}{30º} \approx 0.33\ grams\ /\ º$$

$$Ftof = \frac{1\ gram}{45º} \approx 0.02\ grams\ /\ º$$

This therefore justifies the need for time-of-flight sensors over ultrasonic ones. The table below presents the final choice for the navigation sensors

Table 8.3: Navigation sensors specifications

Sensor	Supplier	Model	Comments	Reference
Vertical ToF	ST	VL53L5CX	This version allows for multizone ranging, making the ground relief detection possible.	[14]
Horizontal ToF	ST	VL53L1X	This version only allows for single ray emissions	[15]

Based on this choice, a first schematics of the drone field of view can be drawn. Note that the cameras presented here are not part of the avionics module and are expected to be connected to the main computer.

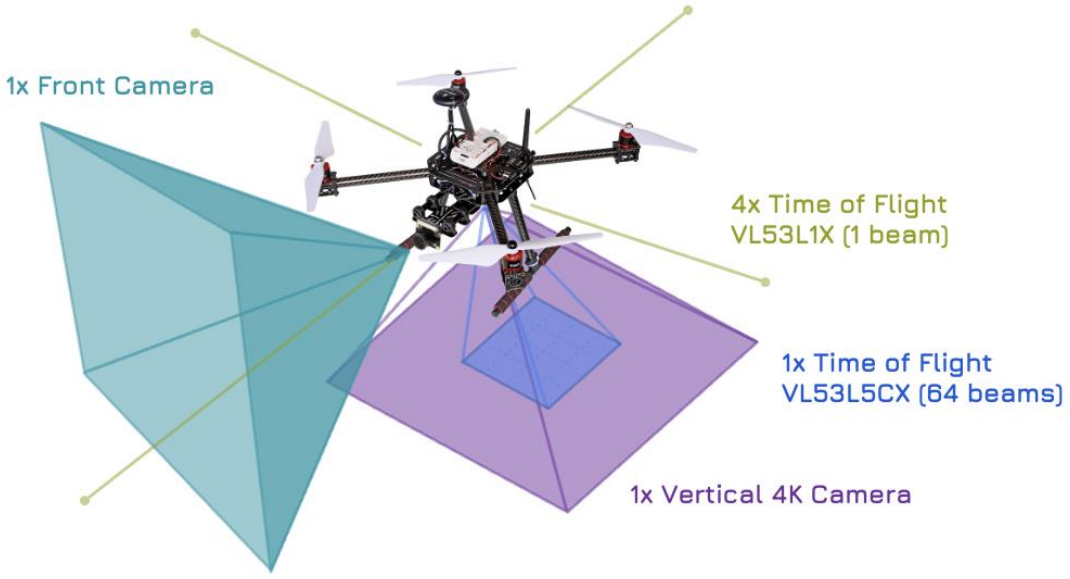


Figure 8.1: Drone visual field

Of the two time-of-flight sensors mentioned, the designs presented below only develop the case of the VL53L5CX sensor as it is directly integrated on the board. Interfaces for the VL53L1X sensor will nonetheless be presented in chapter 9.2.

8.2.3 Summary

Table 8.4: Sensors summary

Sensors	Protocol	Supply Voltage			Max. current consumption
		Min.	Typ.	Max.	
IMU	I2C (up to 400 kHz), SPI	2.4	3.0	3.6	5 mA
Magnetometer	I2C (up to 3.4 MHz), SPI	1.62	3.0	3.6	4.9 mA
Barometer	I2C (up to 3.4 MHz), SPI	1.65	3.0	3.6	730 µA
GPS	I2C (up to 400 kHz), UART	3.0	3.0	3.0	25 mA
ToF (vertical)	I2C (up to 1 MHz)	2.6	2.8	3.5	80 mA
ToF (horizontal)	I2C (up to 1 MHz)	2.8	2.8	3.3	18 mA

8.3 – Design

In this section, we will focus on the implementation of the sensors chosen.

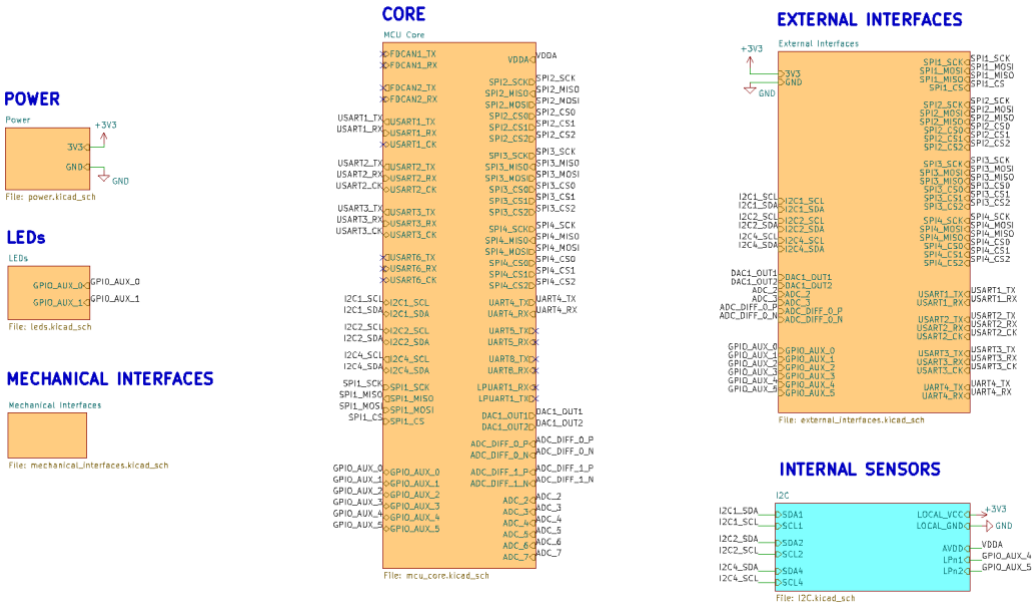


Figure 8.2: Drone system decomposition – Internal sensors

8.3.1 I2C

Since the avionics sensors all come as I2C compatible, it was decided to use only use this protocol for design simplicity reasons. The sensors are therefore split between three I2C buses as follows.

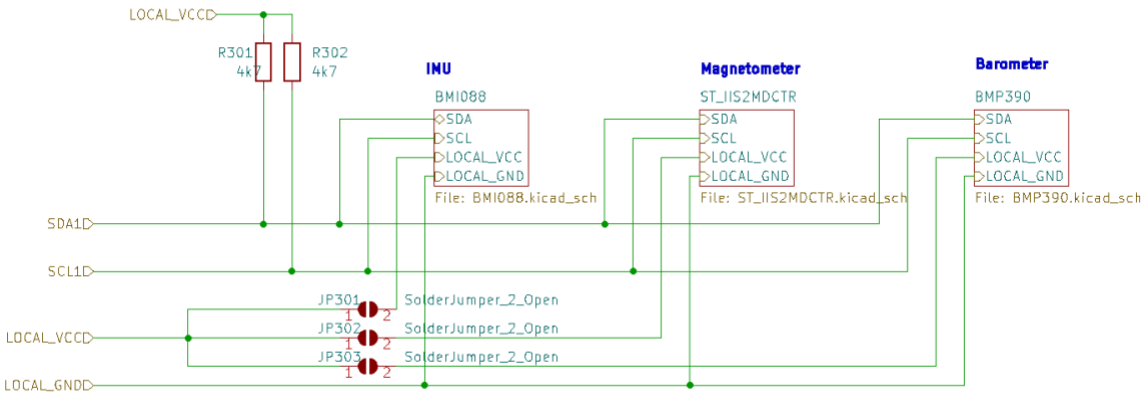


Figure 8.3: I2C1 Sensors

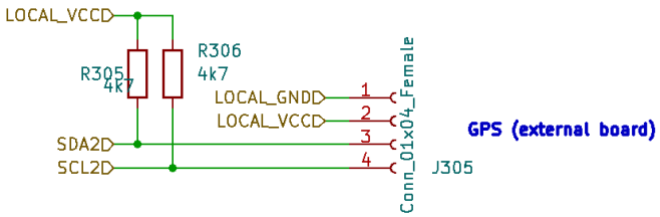


Figure 8.4: I2C2 Sensors

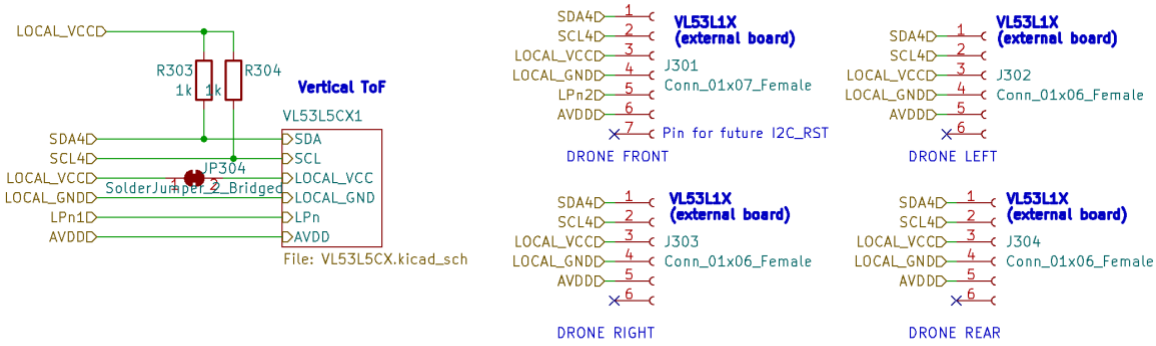


Figure 8.5: I2C4 Sensors

The I2C protocol relies on two signals: a serial data line (SDA) and a serial clock line (SCK). It is a multi-master multi-slave protocol which relies on two pull-up resistors to change the communication speed of its lines. A higher pull-up resistance leads to a lower current and therefore a lower communication speed on the lines.

As a result, a particular attention needs to be paid to the resistor values that are used in the design. To select the latter, we based our calculations on the Texas Instrument SLVA689 application report [14].

$$Rp(min) = \frac{V_{CC} - V_{OL}(max)}{I_{OL}}$$

$$Rp(max) = \frac{t_r}{0.8473 * C_b}$$

With:

- V_{CC} the pull-up supply voltage
- V_{OL} the low-level output voltage
- t_r the rise time of the bus lines
- C_b the load capacitance of the bus line
- I_{OL} the current sink of the bus lines

Given that the I2C lines are quite short (< 10 cm), in standard mode we have:

$$V_{OL} = 0.4\ V$$
$$t_r = 1000\ ns$$
$$C_b = 200\ pF$$
$$I_{OL} = 3\ mA$$

This leads to:

$$Rp(min) = \frac{3.3 - 0.4}{0.003} = 966.7\ Ohms$$

$$Rp(max) = \frac{1 * 10^{-6}}{0.8473 * 200 * 10^{-9}} = 5.9\ kOhms$$

Looking at the I2C modes tolerated by the sensors (Table 8.4), the I2C1 and I2C2 buses are limited to 400 kHz (fast mode) and therefore a standard 4.7 kΩ pull-up resistance is sufficient. The I2C4 sensors allowing for a higher I2C speed (up to 1 MHz), a lower pull-up resistance of 1 kΩ was chosen.

Note that each sensor has its own pull-up resistance that can be connected thanks to a solder bridge to further increase the communication speed of the lines.

8.3.2 IMU

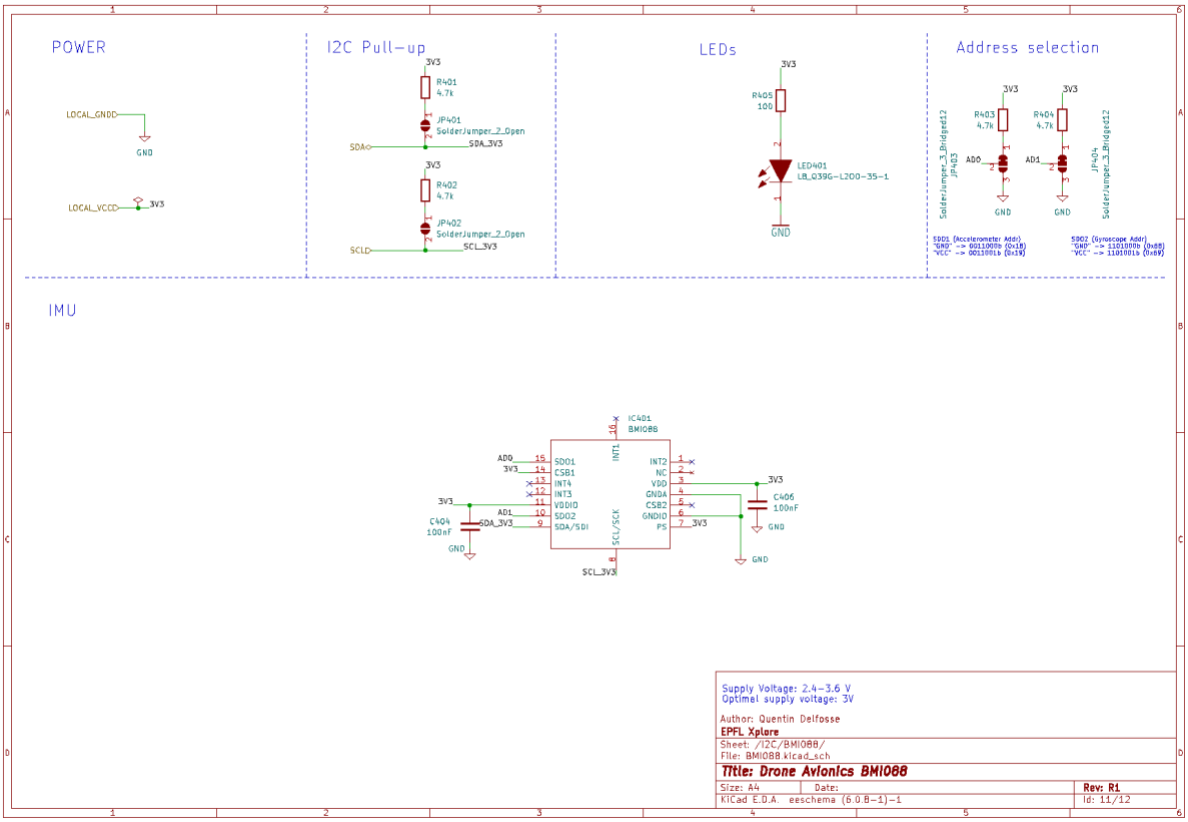


Figure 8.6: IMU design

The BMI088 IMU [15] includes both a 3-axis accelerometer and a 3-axis magnetometer and can be interfaced via I2C and SPI protocol. The sensor is configured in I2C on start-up, and will keep this configuration until a rising edge is detected on the CBS1 pin.

To select which sensor to communicate with, two I2C addresses are provided for each of them and are defined by pulling the SDO1 and SDO2 lines up or down.

Table 8.5: IMU address selection

Configuration	Gyroscope I2C address (SDO1 select pin)	Magnetometer I2C address (SDO2 select pin)
Pulled-up (VCC)	0x18	0x68
Pulled-down (GND)	0x19	0x69

To ensure that I2C is selected, we need to pull the CSB1 pin up. As a rule of thumb, we usually want the pull-up resistor on to be at least 10 times smaller than the internal pull-un resistor of the pin. In this case, $R_{up} = 100\,kOhm$, so choosing a resistor of 4.7 kΩ would be appropriate.

As the chip shows two power inputs, two decoupling capacitors of 100nF were used next to it.

8.3.3 Magnetometer

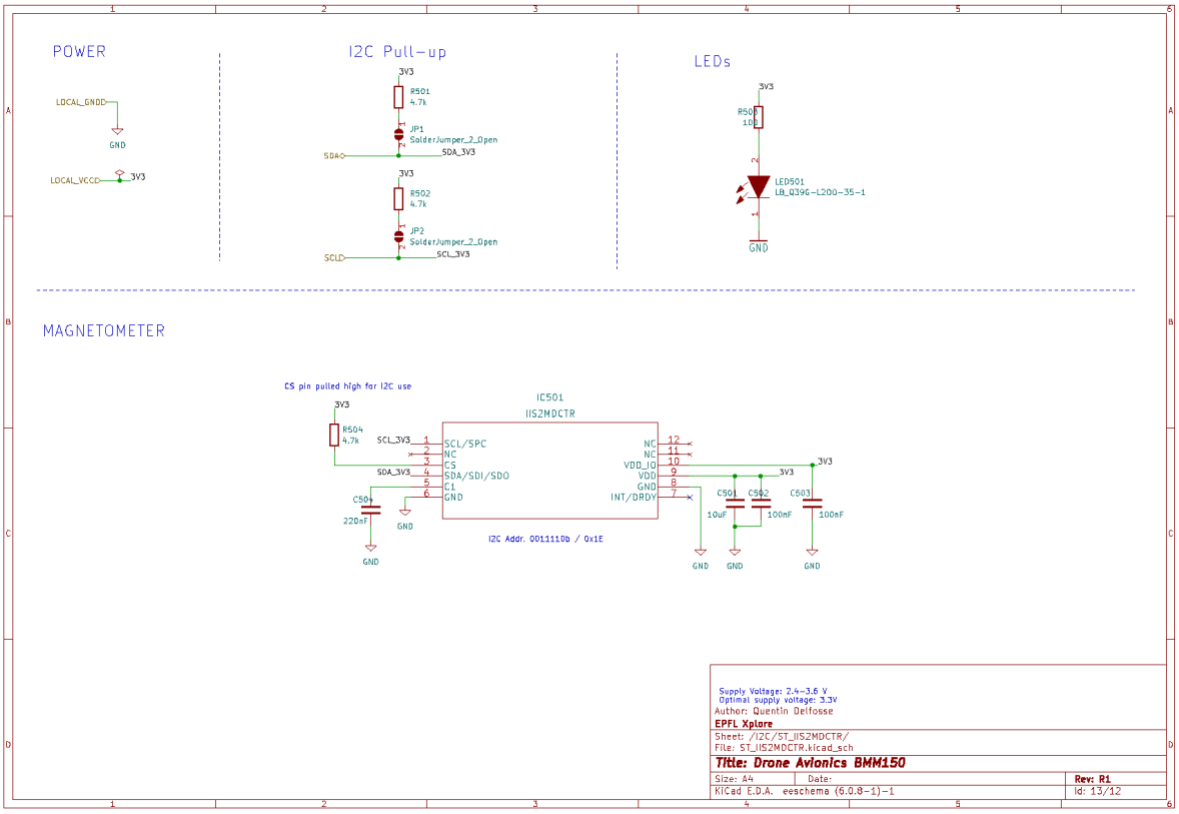


Figure 8.7: Magnetometer design

The IIS2MDC magnetometer [16] allows for SPI and I2C protocols. As our application relies on the use of the I2C protocol, the latter is enabled by pulling the chip select pin (CS) high.

To access the device, a single address is provided by the manufacturer: 0x1E.

The decoupling capacitors presented in the schematics above were selected based on the datasheet guidelines.

8.3.4 Barometer

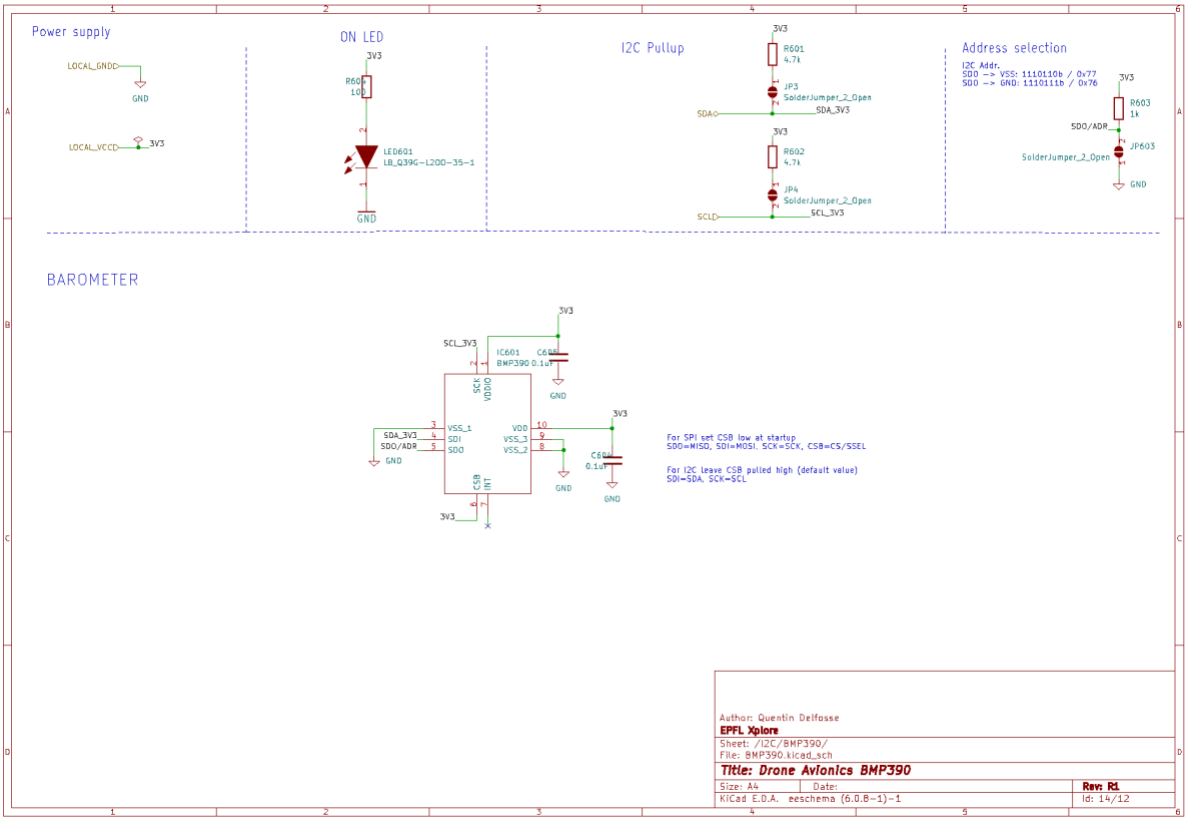


Figure 8.8: Barometer design

The BMP390 barometer [17] also allows for SPI and I2C protocols. Pulling the CSB pin high ensures that is stays in I2C configuration.

Two I2C addresses are provided to communicate with this sensor. The address selection depends on the state of its SDO pin.

Table 8.6: Magnetometer address selection

Configuration	Barometer I2C address (SDO select pin)
Pulled-up (VCC)	0x77
Pulled-down (GND)	0x76

8.3.5 GPS

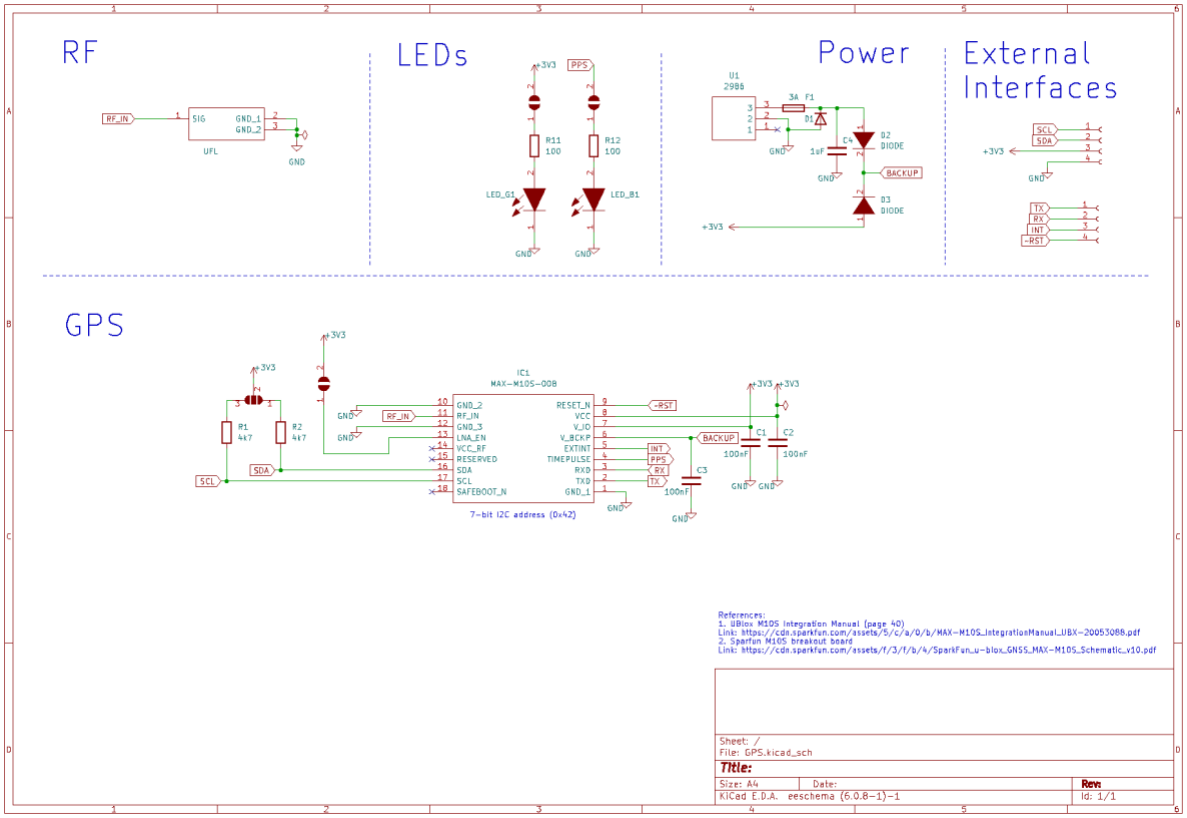


Figure 8.9: GPS design

As it is still unsure whether the use of GNSS signals will be allowed at the competition, the GPS board was designed as a separate module.

The GPS chip used is a MAX M10S [18] which supports concurrent reception of four GNSS (GPS, GLONASS, Galileo, and BeiDou). It can be interfaced via I2C or UART protocols through two separate buses, allowing for both protocols to be used simultaneously.

Power

Depending on the GPS startup sequence, an alternate power source needs to keep the board powered on when the drone is shut down. A backup pin is provided on the chip to allow for such a connection.

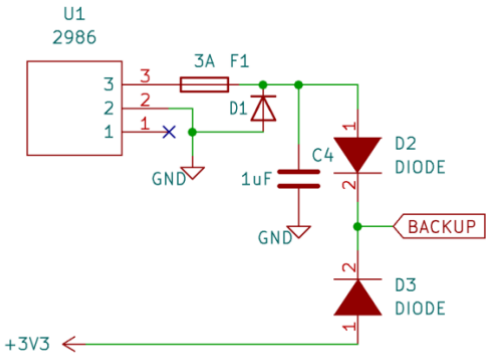


Figure 8.10: GPD design - Power

To ensure a supply voltage of 3.3 V on the backup pin, a Seiko MS518SE 3.0 V coin cell (U1) [19] is used. A safety circuit composed of a Zener diode (D1) with a reverse voltage of 3.3 V and a 3A fuse (F1) was also added to ensure that no voltage above 3.3 V could reach the pin.

The backup pin could therefore be powered either from the battery cell or the avionics 3.3 V power line. To preserve the coin cell battery when the avionics is powered on, a bridge made of 2 Schottky diodes facing each other (D2, D3) was designed to link the different power sources to the backup pin.

As the backup pin takes between 1.65 V ($V_{bckp}(min)$) and 3.60 V ($V_{bckp}(\max)$) as an input, using 2 diodes with a direct voltage $V_f = 0.7\,V$ allows for a nominal operation of the chip.

$$V_{bckp}(cell) = V_{cell} - V_f = 3.0 - 0.7 = 2.3\,V > V_{bckp}(min)$$
$$V_{bckp}(avionics) = V_{cc} - V_f = 3.3 - 0.7 = 2.6\,V > V_{bckp}(min)$$

Further, to ensure that the current coming from the coin cell is blocked when the board is powered by the avionics, the following equation needs to be satisfied.

$$V_{cell} - (V_{cc} - V_f) < V_f$$

The use of a 3.0 V battery cell allows to satisfy the previous equation:

$$V_{cell} - (V_{cc} - V_f) = 3.0 - (3.3 - 0.7) = 0{,}4$$
$$V_f = 0.7$$

Time pulse

To determine the time pulse signal acquired by the GPS module, the PPS (Pulse Per Second) pin is connected to a LED and will make it blink at the signal pulse frequency.

Startup Modes

- **Cold start** (up to 60 seconds sequence): the receiver has no information about its status (i.e. time, frequency, position, velocity, etc.). It will therefore look for satellite signals on different frequencies to decode the ephemeris. Retrieving the latter will allow it to calculate its position and velocity.

- **Warm start** (up to 36 seconds): the receiver has outdated ephemeris data (every 4 hours) and needs to retrieve it again.

- **Hot start** (a few seconds): the receiver still has valid ephemeris data (it was powered down for less than 4 hours). The position and velocity can be determined directly on startup.

Note that the warm and hot startup sequences require an alternate power source to keep the satellites data (frequencies and time) in memory (for more details on the GPS source, please refer to the power section of Chapter 8.3.5).

Antenna

For weight and size concerns, a patch antenna was used [20]. It is connected to the PCB via an IPEX connector and linked to the RF_IN pin of the GPS module.

The specifications of the antenna are presented hereunder.

Table 8.7: Antenna specifications

Brand	Pulse Electronics
Model	GPSGB1330
GNSS Signals	GPS, GLONASS, Galileo, Beidou
Connector	IPEX MHF
Min. frequency	1.56 GHz
Max. frequency	1.602 GHz
Impedance	50 Ω
Length	13 mm
Width	13 mm
Height	2.2 mm
Weight	12.838 g

As for all RF lines, some considerations need to be made regarding the design of the PCB traces to match the impedance of the lines and that of the antenna. This will be discussed in the PCB design section (see Chapter 11.3.2).

8.3.6 Vertical time of flight

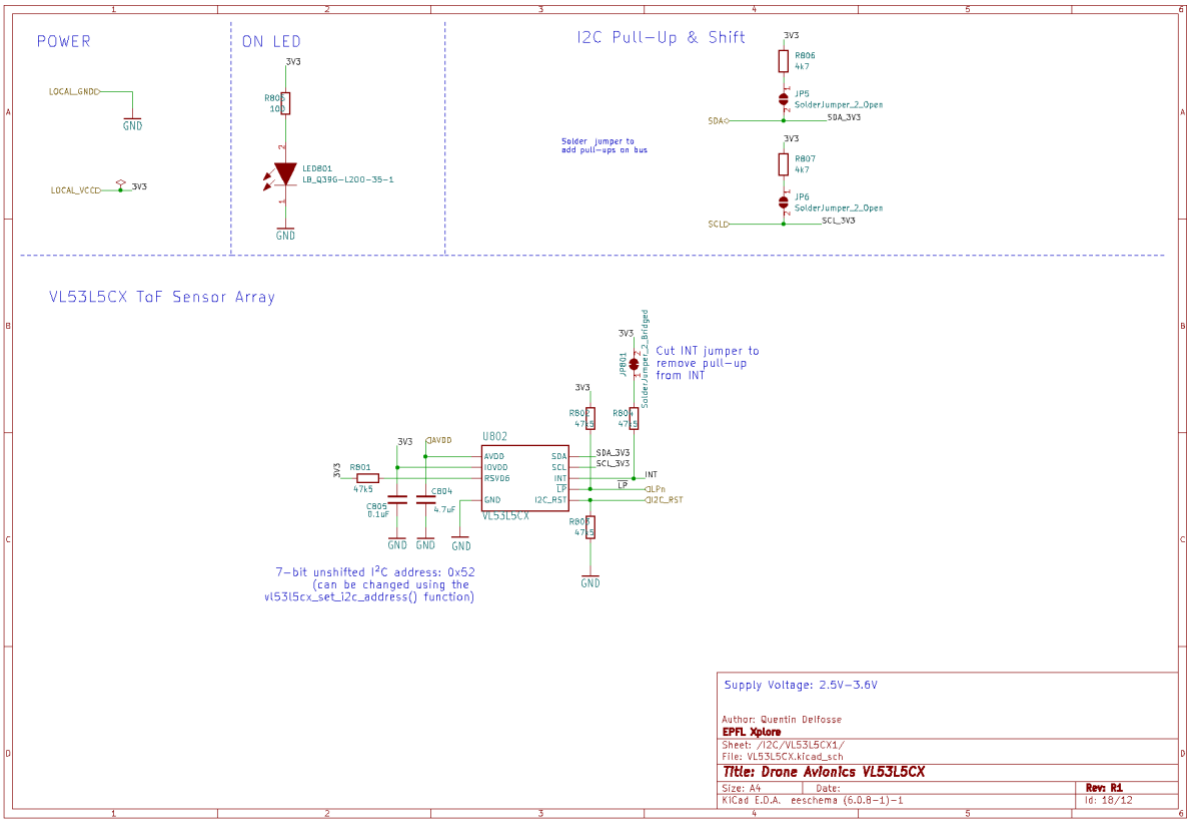


Figure 8.11: Vertical time of flight design

The final sensor of the avionics board is a STMicroelectronics VL53L5CX time-of-flight sensor [21]. As its I2C address is programmable, no chip select pin is shown.

In our case, as the number of I2C lines is limited, we are forced to put several instances of this module on the same bus. Therefore, we need to be able to modify their addresses from the default one: 0x52. This is done via to the LPn pin controlled by the GPIO5 pin of the microcontroller. For the detailed sequence, please refer to ST's application note AN4846 [22].

Finally, the selection of the resistors and capacitors was made according to the product's datasheet.

9 – Interfacing Stage Design

To integrate the avionics board together with the drone, and for system debugging purposes, the board shows a series of physical connectors.

9.1 - Requirements

The avionics interfacing stage shall:

- Allow for an additional sensor layer to be interfaced on top of it
- Have a dedicated I2C, SPI and UART protocols connector to communicate with the power supply
- Have a dedicated SPI protocol connector to communicate with the main computer
- Have a dedicated I2C protocol connector to communicate with the GPS
- Have 4 dedicated I2C protocol connectors to communicate with the horizontal time-of-flight sensor boards
- Ensure that no reverse voltage can be applied to the input of the board

9.2 – Design

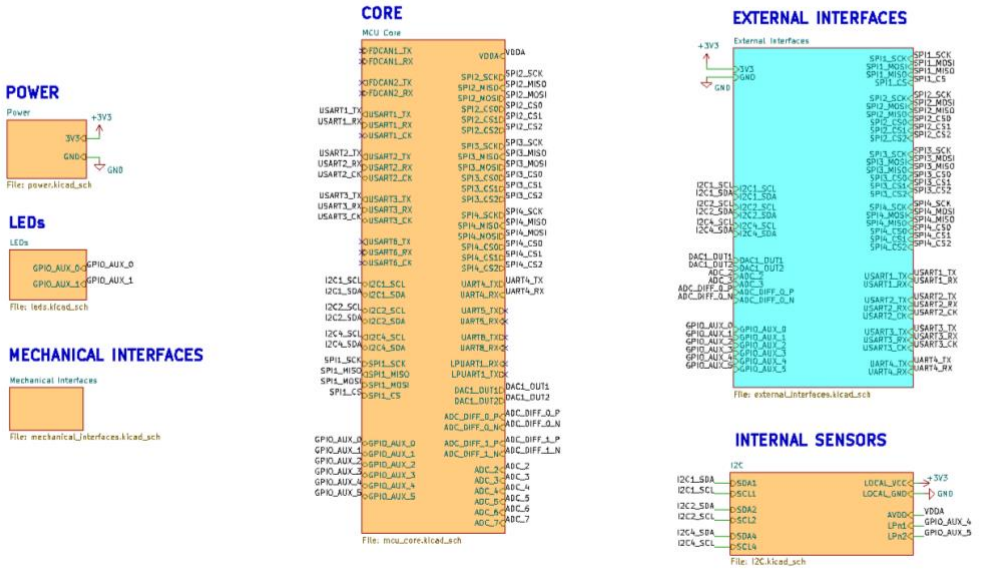


Figure 9.1: Drone system decomposition – External interfaces

To comply with the board requirements on interfaces, the following design is proposed.

9.2.1 Input power interface

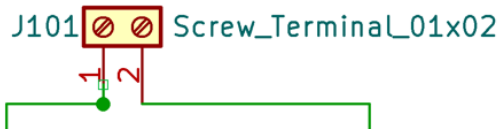


Figure 9.2: Input power connector

To prevent reverse voltages from being applied as an input to the board, a single-orientation connector needs to be used. Among the most common power connectors is the XT series one. The XT30PW model [23] came as the perfect solution given its horizontal configuration, limited weight, compactness and power ratings (up to 15A). As its footprint could not be retrieved, a screw terminal footprint fitting its dimensions was used in the schematics.

9.2.2 Output power interface

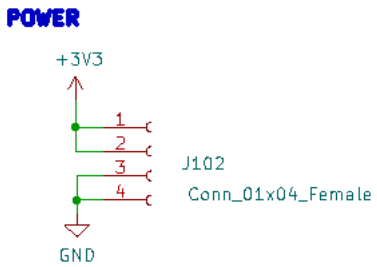


Figure 9.3: Output power connector

In case additional power lines are required, and for debugging purposes, a dedicated connector with two 3.3 V and two GND pins is provided on the board.

9.2.3 General purpose interfaces

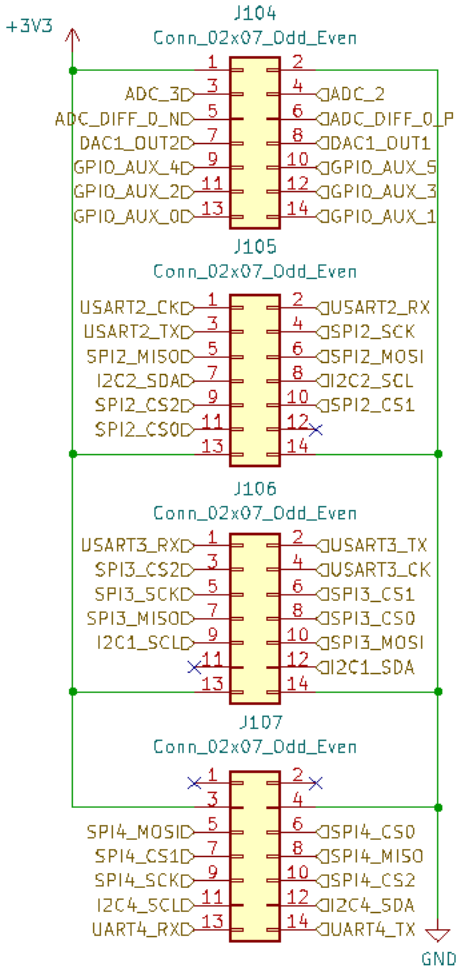


Figure 9.4: General purpose connectors

Given that the design is expected to evolve in the coming years, interfaces to mount additional sensors on top of the board are expected. The first connector is mainly expected to be interfaced with analog sensors (3 ADCs, 2 DACs) and provides 6 instances of GPIOs. The next 3 connectors are expected to be connected to digital sensors. For modularity reasons, they each show a different instance of I2C, SPI and UART protocols.
Finally, all connectors come with a 3.3 V and GND pin.

9.2.4 Dedicated sensors interface

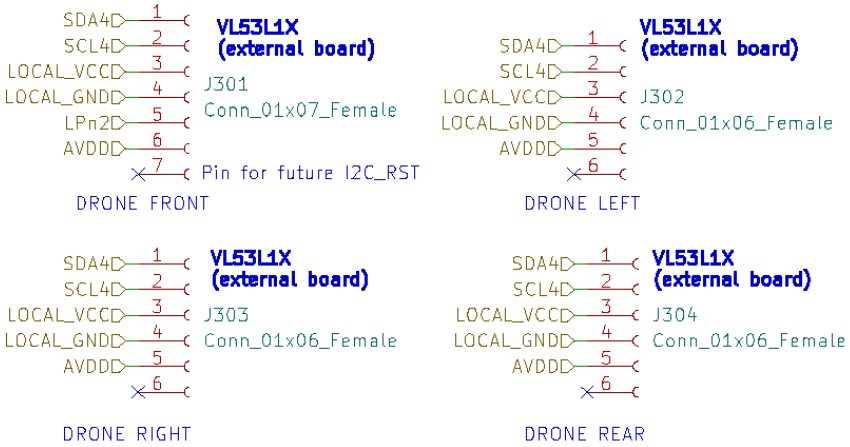


Figure 9.5: Time of flight connectors

Figure 9.6: GPS connector

As mentioned in the previous section, some sensors could not be placed directly on the board but were still planned for in the design. These include the GPS module on the I2C2 bus and the horizontal time of flight sensors on the I2C4 bus.

9.2.5 SPI interface

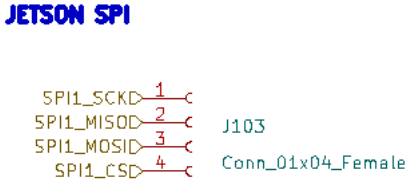


Figure 9.7: Jetson SPI connector

According to the drone architecture (see Figure 4.2), the avionics board will communicate with the main computer via SPI. The version of the protocol used here is 4-wire as it relies on

4 signals: serial clock (SCK), master to slave (MOSI), slave to master (MISO) and chip select (CS).

9.2.6 Additional USB interface

Finally, the question of how to easily communicate with the board needs to be addressed.

Based on the designs presented above, the SWD lines can serve this purpose thanks to the SWO pin. Nevertheless, this possibility was not known at the time of the design and so other solutions were explored.

Another option that was therefore thought of was to use an external UART to USB converter linked to one of the general-purpose interfaces. While it allows the user to monitor the board data in real time from the terminal, having to disassemble the drone to reach these pins would not be practical.

As a result, we decided to go beyond the scope of this project and directly add the UART to USB bridge on the board. As a similar design had already been tested on the rover's avionics boards, we decided to implement it as is.

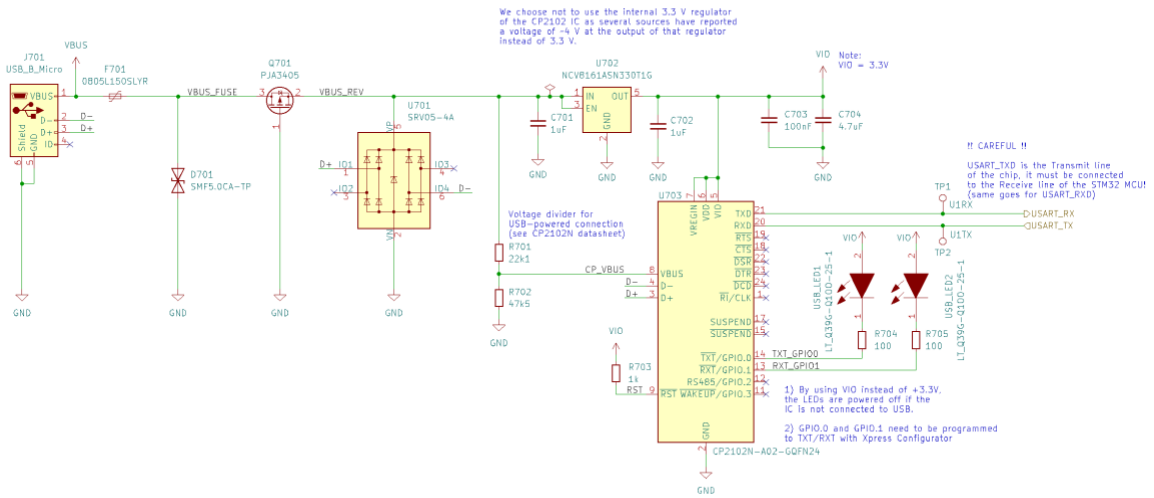


Figure 9.8: UART to USB bridge design

The chip responsible for UART to USB conversion is a Silabs CP2102N [24]. It can either be powered on the USB end through a voltage divider (R701, R702) or on the board end by its 3.3V supply. LEDs can also be found on the GPIO.0 and GPIO.1 pins of the chip to indicate a functioning communication.

Another advantage of the USB interface over the SWD link is that it can power and communicate with the board at the same time. To do so, we simply need to reduce its 5V functioning voltage to the 3.3 nominal input voltage of the board through a LDO regulator. Here, we used the NCV8161 chip [25] for its precision within the working range of 1.9 V to 5.5 V.

Once again, safety measures were taken on the power lines. This is materialized by the use of a bidirectional Zener diode maintaining a 5 V reverse voltage [26], an ESD protection chip [27], a P-channel mosfet [28] and a fuse [29]. More details on the safety calculations can be found in the *Electronics Subsystem Main Board Design Report* from January 2023 by Mr. Vincent Nguyen.

10 – Mechanical Stage Design

10.1 – Requirements

The mechanical stage shall:

- Allow the board to be mounted on the avionics structure under 2 minutes
- Prevent the board from moving in any direction when mounted on the drone

10.2 – Design

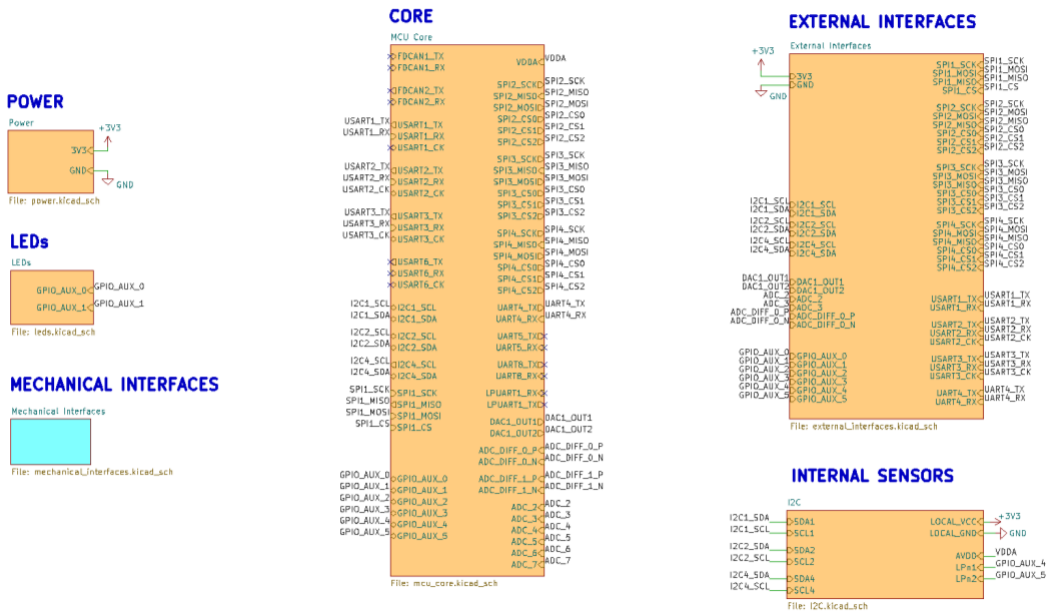


Figure 10.1: Drone system decomposition – Mechanical interfaces

The mechanical interface stage relies on height M2 fixing points located on the corners of the board.

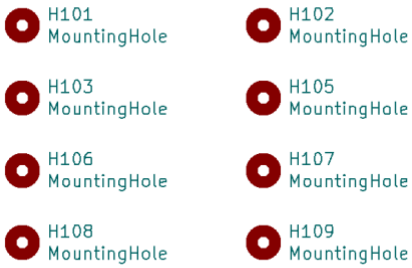


Figure 10.2: Mechanical interfaces

Supposing that we use stainless steel M2 screws, we can compute the maximum load that the screws are able to bear [30]:

$$d_0 = \frac{d_2 + d_3}{2} = \frac{1.74 + 1.509}{2} = 1.6245\ mm$$

$$A_t = \frac{\pi}{4} * d_0^2 = 2.07\ mm^2$$

With:

- d_2 the pitch diameter
- d_3 the root diameter
- A_t the tensil stress area

The load at yield $F_{0.2}$ can be computed from the yield strength $R_{p0.2}$:

$$R_{p0.2} = 300\ MPa$$

$$F_{0.2} = A_t R_{p0.2} = 0.62\ kN$$

There are 8 screws sharing the load, therefore the overall structure can bear:

$$F'_{0.2} = 8 * F_{0.2} = 4.96\ kN$$

This corresponds to a weight of about 500 kg, which is more than enough for our application.

11 – PCB Layout

11.1 - PCB Overview

The stages presented in the previous sections are displayed hereunder.

11.1.1 Avionics board

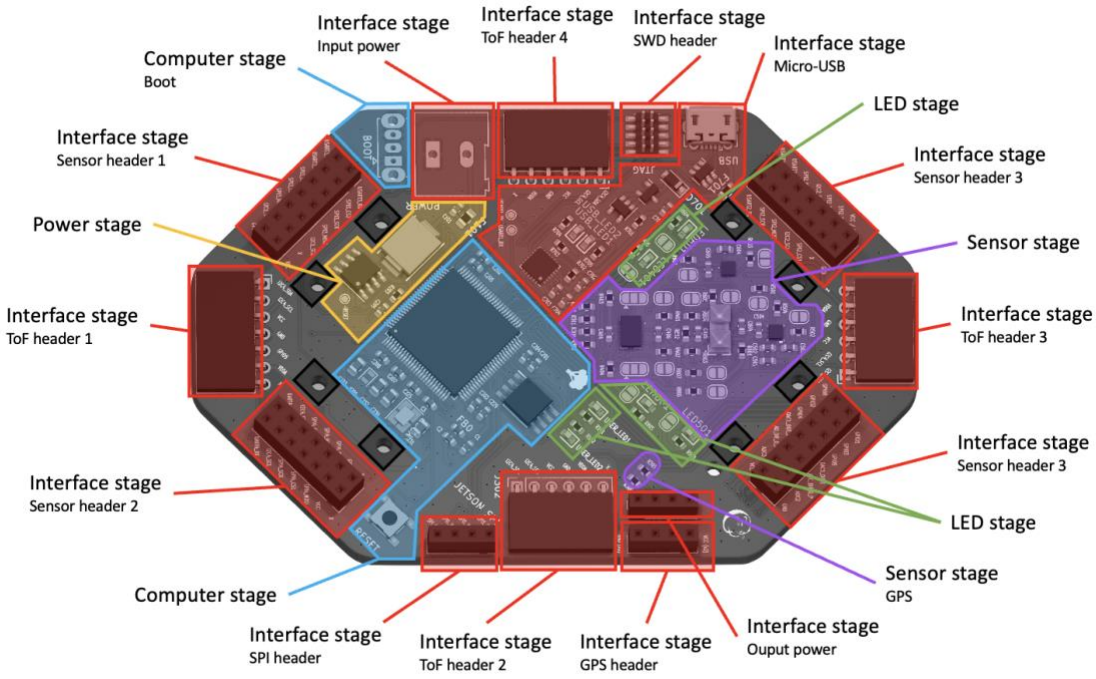


Figure 11.1: Avionics board overview

11.1.2 GPS board

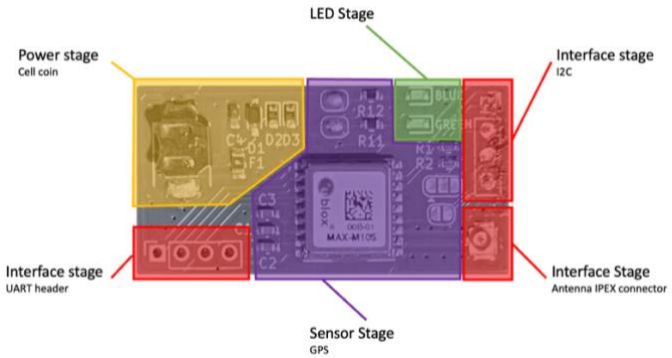


Figure 11.2: GPS board overview

11.2 - Avionics PCB Layout

The final board is a 4-layer FR4 PCB with the following stack-up configuration.

Table 11.1: Avionics PCB layers

Layer	Signal
Top layer	Data /Power / GND
Internal layer 1	Power
Internal layer 2	GND
Bottom layer	Data

To make sure that the PCB could be manufactured, the first step that was taken when launching the PCB editor tool for the first time was to define some constraints on the traces. The following parameters were therefore defined based on Aisler's manufacturing guidelines for a 4-layer PCB [31].

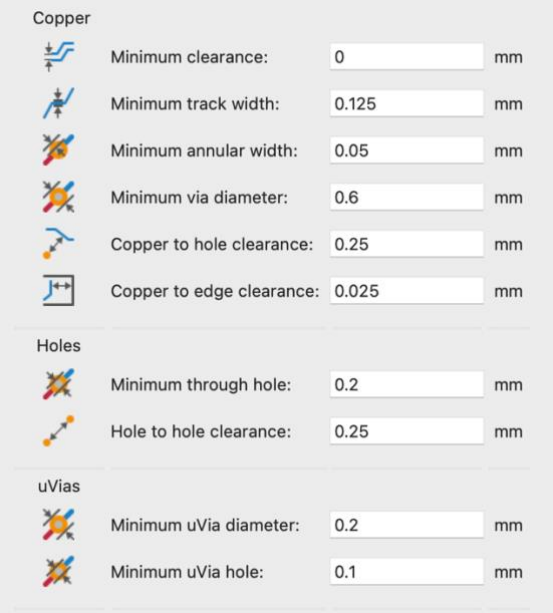


Figure 11.3: Avionics PCB design considerations

11.2.1 Top Layer

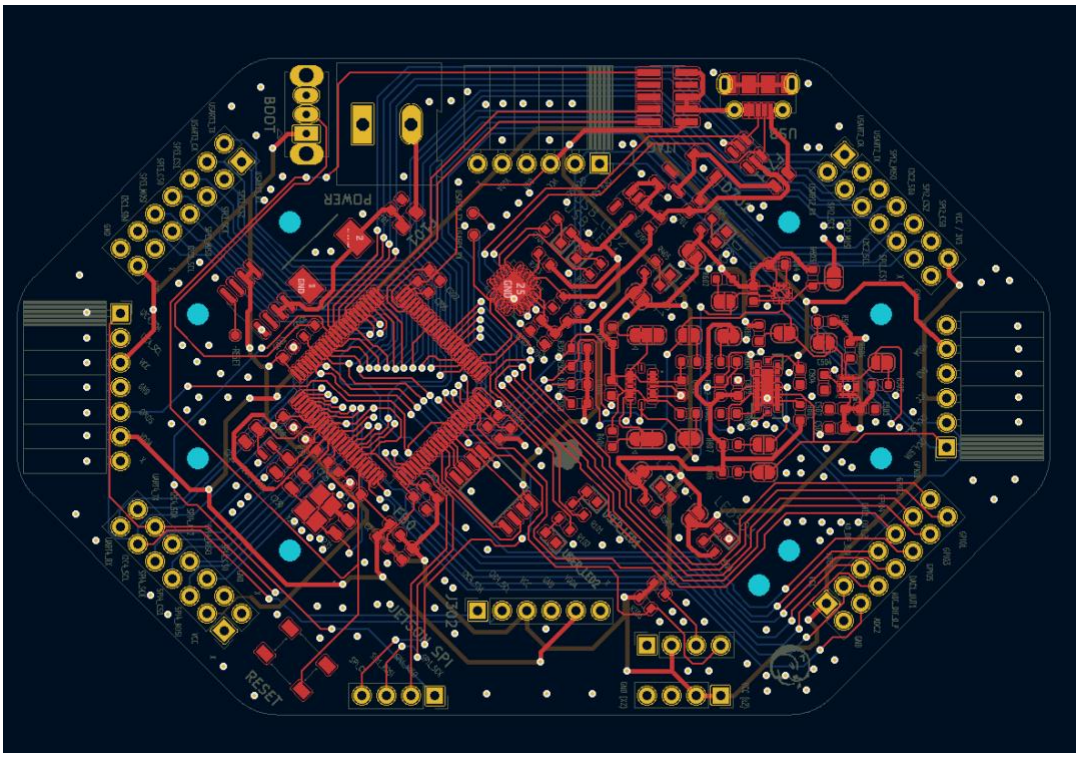


Figure 11.4: Avionics PCB - Top layer layout

As this version of the board is one-sided only, the upper layer includes all components of the board. Since numerous power and data lines are running next to other, some additional design considerations were made. Please note that these constraints apply to all layers.

Table 11.2: Avionics PCB design considerations

Parameter	Value	Comments
Data traces width	0.2 mm	Always verified
Power traces width (away from chips)	0.5 mm	If the trace does not directly connect to a chip
Power traces width (close to chips)	0.2 mm	For the latest trace segment between a passive component and a chip
Inter traces minimum clearance	> 0.4 mm	Equivalent to 2 data traces in between 2 traces (when possible)

Reducing the trace width of the power lines next to active components ensures that the resulting noise on their data pins is limited. Further, as our application is low power (<0.5 A), a trace width of 0.5 mm is deemed sufficient.

After having defined these constraints, we can look at the implementation of our components.

Microcontroller

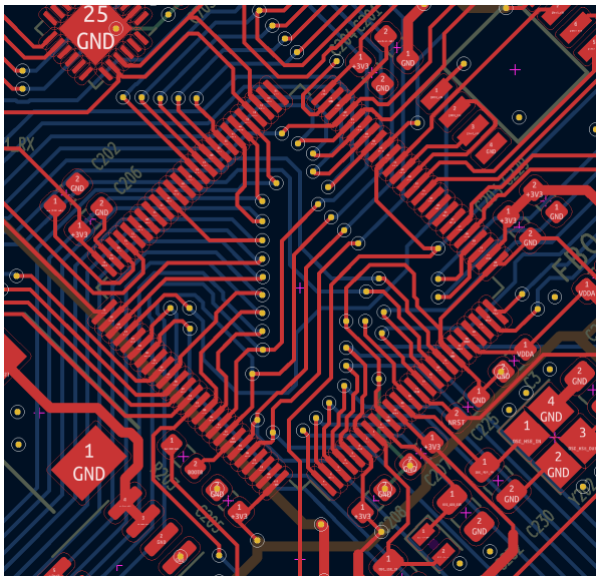


Figure 11.5: Avionics PCB - Microcontroller layout

As mentioned in most active components datasheets, it is important to keep the decoupling capacitors next to their dedicated power pin. This ensures a small loop area between the VCC lines and GND planes which prevents any amplification effect and therefore noise on the power lines next to the components. Here, the decoupling capacitors were placed as close as possible to their power pin.

Sensors

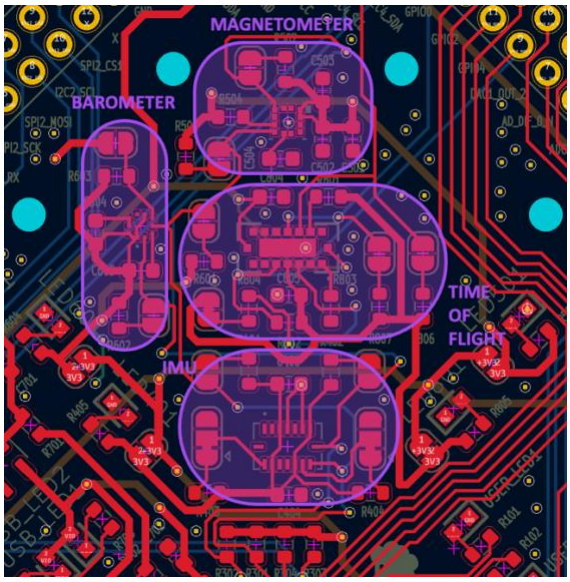


Figure 11.6: Avionics PCB - Sensors layout

The same rule applies to the sensors implementation. Whenever a power pin is present, the decoupling capacitor is placed next to it at the edge of the component. The pull-up resistors

for I2C bus speed regulation and address selection can be placed further away without impacting the performance of the sensor.

To facilitate switching between the different I2C addresses of a sensor, solder jumpers can be used to pull the address selection pins up or down.

Connectors

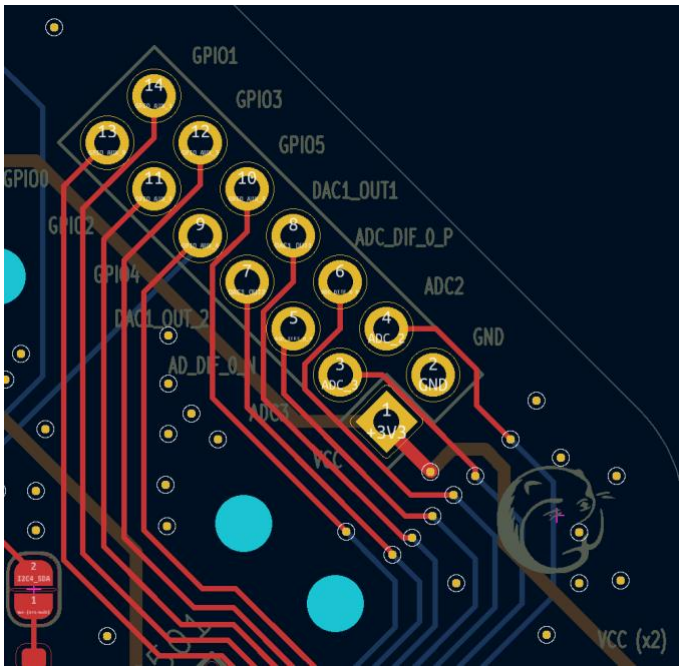


Figure 11.7: Avionics PCB – External sensors connector layout

To make the routing procedure to the external connectors as easy as possible, the connectors pinouts were reorganized. This allowed to reduce the use of further layers while keeping the design clear. The trace clearance constraint set to 0.4 mm can be observed here.

Power distribution

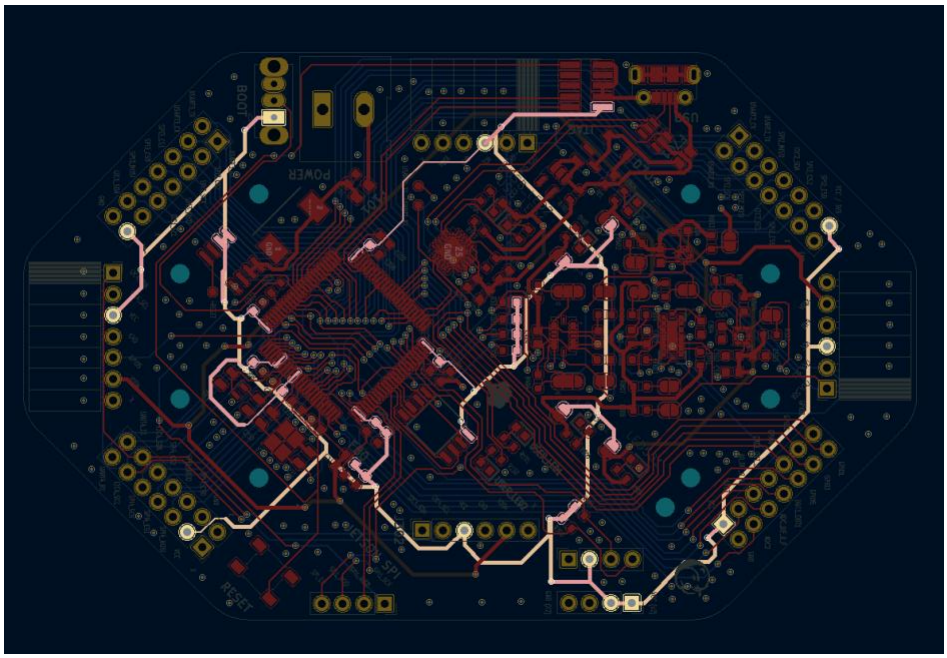


Figure 11.8: Avionics PCB – Top layer power layout

Finally, one important design consideration when dealing with power lines is to make sure that they do not form any closed loop. This could indeed lead the PCB to act as an antenna and be sensitive the magnetic fields, thereby generating noise on the power lines. Here, the power lines act as a tree structure and do not form any loops.

11.2.2 Internal Layer 1

The first internal layer is used as a fully grounded plane and allows to further reduce the noise of the upper layer.

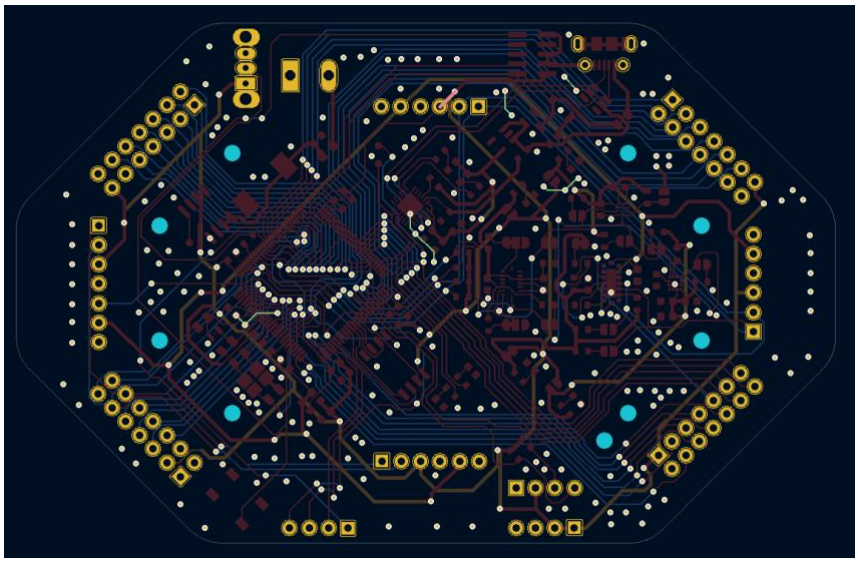


Figure 11.9: Avionics PCB – Internal layer 1 layout

11.2.3 Internal Layer 2

The second internal layer acts as a bridge layer for power lines to jump over the data lines on the other layers.

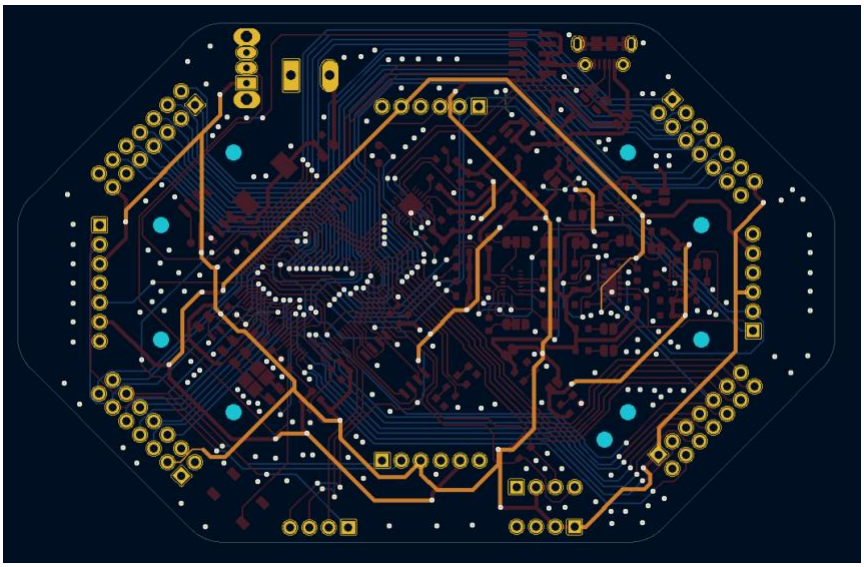


Figure 11.10: Avionics PCB – Internal layer 2 layout

11.2.4 Bottom Layer

Finally, the bottom layer perfectly illustrates the routing process of the data lines to the interface headers. As these traces can be very long and are mostly straight, some care was taken to avoid having them interact with other traces following the same path on other layers. Whenever possible, the traces were routed to perpendicularly cross the path of the traces from the other layers.

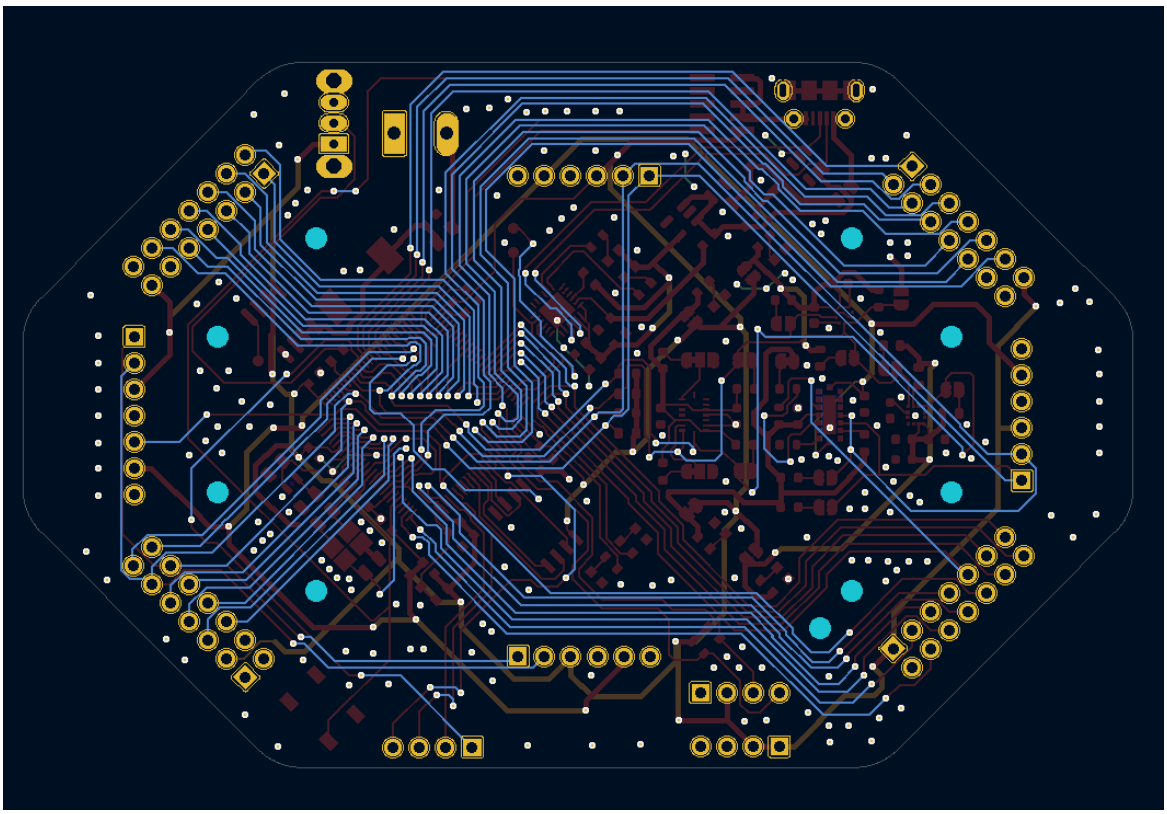


Figure 11.11: Avionics PCB – Bottom layer layout

11.3 - GPS PCB Layout

The GPS board is a 2-layer FR4 PCB with the following stack-up configuration.

Table 11.3: GPS PCB layers

Layer	Signal
Upper layer	Data /Power / GND
Lower layer	GND

Note that the same design considerations as for the avionics board apply here.

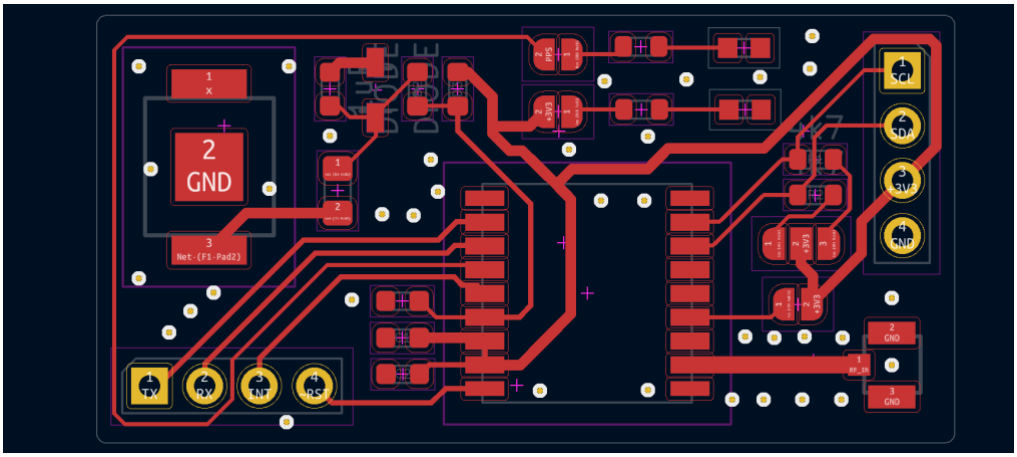


Figure 11.12: GPS PCB – Top layer layout

As this module is quite simple (few traces are required), only the first layer is used for the routing. All components are therefore present on the upper side of the PCB.

11.3.1 GNSS

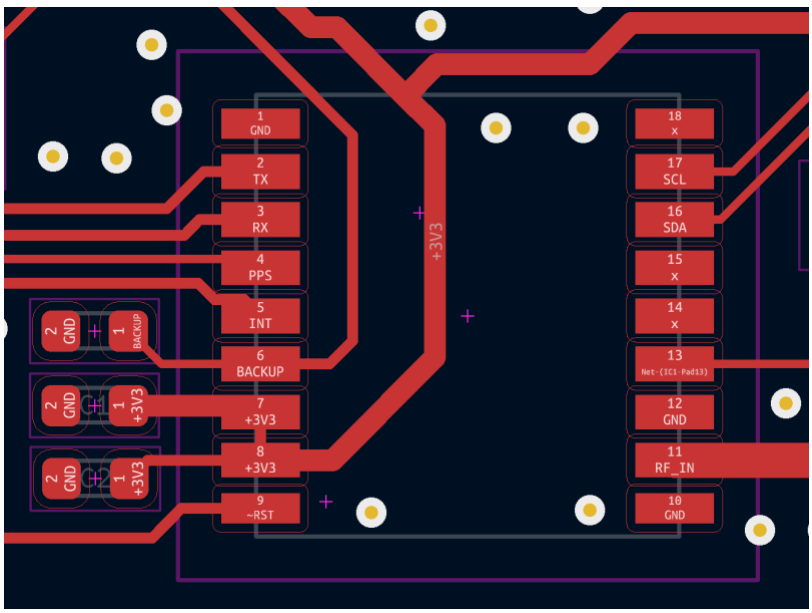


Figure 11.13: GPS PCB – GPS layout

Once more, the decoupling capacitors of the backup, VCC and VIO pins and are located next to the chip.

11.3.2 Antenna connector

When dealing with RF lines, one must make sure that their impedance is matched with the one of the antenna. In our case, the antenna has an impedance $Z_{antenna} = 50\ \Omega$.

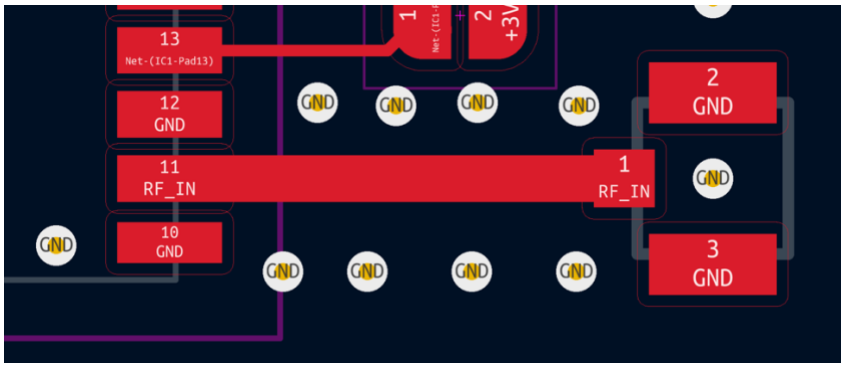


Figure 11.14: GPS PCB – Antenna connector

Varying the trace width and the inter trace clearance will directly impact its impedance. Using the physical parameters of the board, the KiCAD PCB calculator is able retrieve the necessary trace width.

Given:

$$\varepsilon_r = 4.6$$
$$\rho = 1.72 * 10^{-8}$$
$$H = 0.2\ mm$$
$$T = 0.035\ mm$$

The calculator gives W (the width) and S (the inter trace clearance):

$$W = 0.8\ mm$$
$$S = 0.2\ mm$$

Moreover, making sure that both sides of the RF line are well grounded by means of ground vias and adjacent ground planes helps stabilize the line. Finally, no power lines were routed near the RF trace.

12 – Manufacturing & Assembly

Once the board designed, the Kicad files were sent to Aisler for manufacturing. A couple of weeks later, the board was received and the assembly process could start. Thanks to the advice of the Spot (Discovery Learning Lab Building) coaching team, the following procedure was applied to assemble the boards and their components.

- Fix the PCB on a flat surface thanks to holding pads (in green Figure 12.1).
- Place the PCB stencil on top of it

Note: The PCB pads should be nicely aligned with the stencil holes.

- Fix the upper part of the stencil on the holding pad above the PCB using tape
- Retrieve the flux paste from the refrigerator
- Heat the flux paste up by spreading it on a flat surface thanks to a scraper
- Place a large amount of flux paste on the holding pad above the PCB
- Spread the flux paste on the stencil from the top down using the scraper

Note: Make sure to apply a good amount a pressure on the scraper, otherwise the flux paste will not stick to the PCB.

- Gently remove the stencil and the holding pads and check if all pads display flux paste
- If not, clean up the PCB with an isopropyl spray and restart the procedure

The PCB is now ready for components placement.

- Place the PCB on the rail of the pick and place machine
- One after the other, select the component to place and use the handle to precisely position it on top of its pad

Note: the component should stick to the PCB thanks to the flux paste

- Once all components are mounted, start the pre-heating sequence of the reflow oven
- When the sequence is finished, place the PCB inside of the oven and start the reflow sequence

- A few minutes later, the PCB can be retrieved
- Clean up the PCB using an isopropyl spray

The PCB is now ready to be tested.

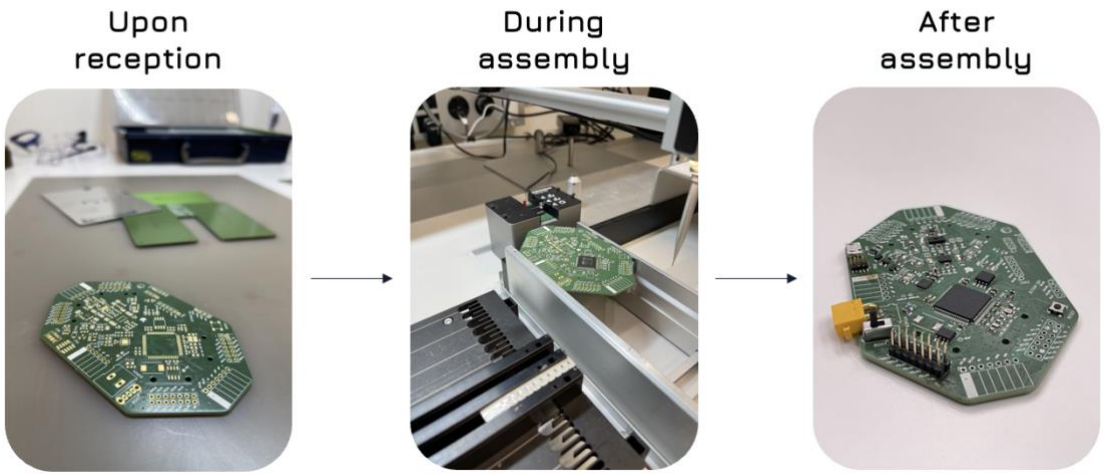


Figure 12.1: PCB assembly steps

13 – Software

In parallel to the design of the avionics board, this project aims at developing its software architecture and some drivers to test its proper functioning.

13.1 - Requirements

In the frame of this project, the following software requirements apply:

- Setup a basic software architecture allowing for sensor data acquisition and relay
- Develop a bi-directional communication driver to communicate with the avionics board from a computer
- Develop and test at least two sensor drivers

13.2 - Design

To allow for sensors data acquisition and to communicate with external computers, the following software architecture was developed.

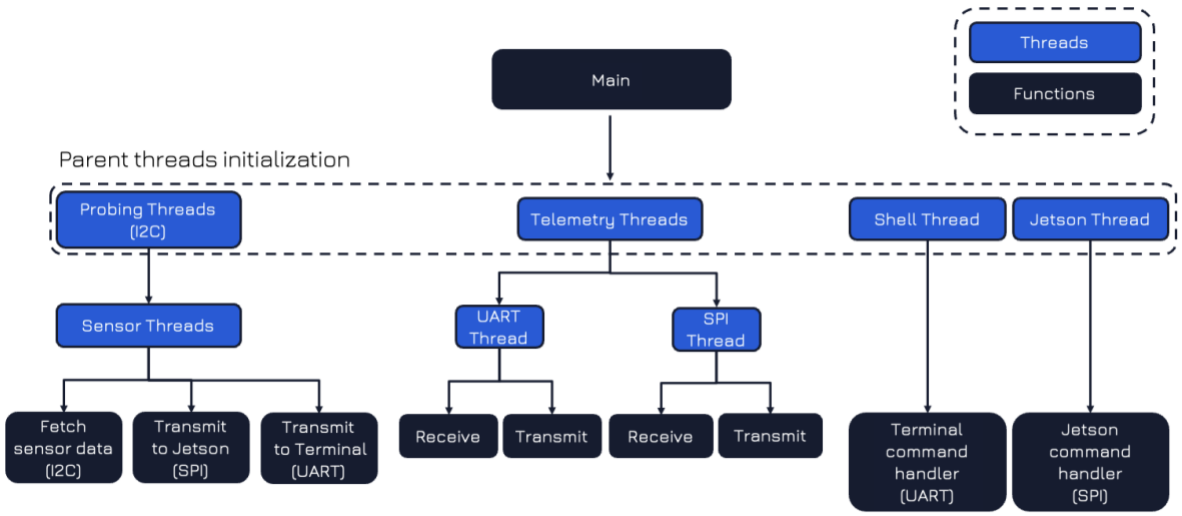


Figure 13.1: Avionics software architecture

It relies on a series of parent threads responsible for initializing low level threads upon completion of specific tasks. These parent threads are launched on system start up thanks to a FreeRTOS instance called by the main function.

13.1.1 Telemetry thread

The first thread to be instantiated is the telemetry one. It oversees the setup of the UART and SPI instances that will be used to communicate with external boards or computers. In our case, the UART1 and SPI1 instances are used for such purposes.

13.1.2 Probing threads

The probing threads is in charge of initializing the sensors on their dedicated I2C bus. Each of them is responsible for probing an I2C bus with the addresses of the avionics sensors. Receiving an acknowledge on the sensor part after transmitting its expected address on the bus means that the device was found at this address and can be initialized. The specific sensor thread is then instantiated by the prober before moving on to the next sensor address.

```
void ProberThread::loop() {

    if(probeI2C(BMP3_ADDR_I2C_SEC) && !baro_initialized) {      // Looking for Barometer (BMP390)
        this->instance = new BarometerThread(this);
        this->baro_initialized = 1;
    }
    if(probeI2C(0x1E) && !magneto_initialized) {                // Looking for Magnetometer (ILS2MDCTR)
        this->instance = new MagnetometerThread(this);
        this->magneto_initialized = 1;
    }

    // [.....]

    if(baro_initialized && magneto_initialized) {
        xSemaphoreTake(semaphore, portMAX_DELAY);
    }

    vTaskDelay(1000 / portTICK_PERIOD_MS);

    HAL_I2C_DeInit(hi2c);
    HAL_I2C_Init(hi2c);
}
```

Figure 13.2: Probing thread

13.1.3 Sensor threads

Once the sensor thread has been initialized, it will fetch and transmit the sensor's data until the line is disconnected or a reset occurs.

```
void BarometerThread::loop() {

    if(HAL_I2C_GetError(parent->getI2C()) == HAL_I2C_ERROR_NONE)
    {
        vTaskSuspendAll();
        rslt = bmp3_get_sensor_data(BMP3_PRESS_TEMP, &data.baro, &bmp390dev);
        xTaskResumeAll();

        if(monitor.enter(BARO_MONITOR)) {
            println("%s", data.toString(cbuf));
            monitor.exit(BARO_MONITOR);
        }


        data.toArray((uint8_t*) &packet);
        network.send(&packet);
        portYIELD();

        //HAL_UART_Transmit(&huart3, (uint8_t *) &data.baro.temperature, 4, HAL_MAX_DELAY);
        //HAL_UART_Transmit(&huart3, (uint8_t *) &data.baro.pressure, 4, HAL_MAX_DELAY);

    } else {
        println("BMP390 disconnected");
        terminate();
        parent->resetProber();
    }
}
```

Figure 13.3: Barometer sensor thread

Currently, all sensors data are only transmitted via UART, either through the UART1 (linked to the internal UART to USB bridge) or UART3 (used for debugging with an external UART to USB bridge) buses.

13.1.4 Shell thread

Now that we have sensor data acquisition, we need to transmit this data to a computer and display it. This is done via the `println` method which allows for a string buffer to be sent on a defined UART line.
Using a UART to USB bridge, the user can access the boar's data on its computer's terminal by screening the dedicated serial port. A monitor can then be launched to organize the information received on the terminal and allow the user to observe the sensors data change over time.

Furthermore, one might need to send commands from the terminal to the board. Resetting the board or selecting a specific sensor to retrieve data from could prove useful. Therefore, a shell thread is used to look for any buffer reception on the board side. Once a message is received, a handler checks it against a pre-defined message databank and a specific action occurs.

```
void Shell::loop() {
    HAL_StatusTypeDef status = HAL_UART_Receive(uart, dma_buffer, 1, 0);

    if(status == HAL_OK) {
        receiveByte(dma_buffer[0]);
    }
}
```

Figure 13.4: Shell thread

13.1.5 Jetson thread

Following up on the previous section, the same principle needs to be applied to the SPI protocol to receive commands from the Jetson. As this module is currently still under development, any bi-directional communication with the Jetson needs to go through the UART to USB converter.

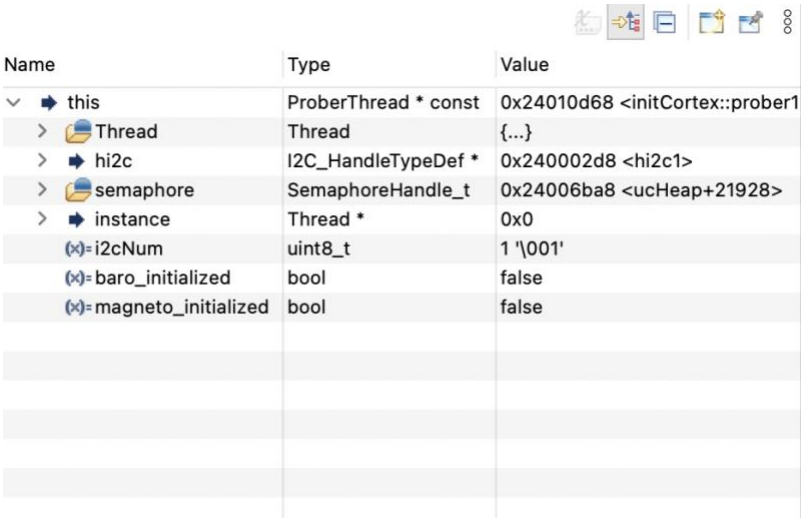
14 – Results

Two methods are used to validate the proper functioning of the board. First, the board is configured in debugging mode to monitor the initialization of the sensor threads and the acquisition of data on the I2C lines. Second, the board is connected via USB to a computer to validate the proper functioning of the UART communication drivers.

14.1 - Debugging

For this test, we use the ST-Link V3 SWD debugger to launch the avionics application in debug mode from the STM32 IDE. Thanks to a series of breakpoints, we can easily follow the start-up procedure of the board until reaching the sensors' probing threads. Here, the barometer and magnetometer threads are inspected. Both sensors are located on the I2C1 bus.

Before probing the magnetometer and barometer, we show the initial state of the I2C1 thread instance.



Name	Type	Value
this	ProberThread * const	0x24010d68 <initCortex::prober1
Thread	Thread	{...}
hi2c	I2C_HandleTypeDef *	0x240002d8 <hi2c1>
semaphore	SemaphoreHandle_t	0x24006ba8 <ucHeap+21928>
instance	Thread *	0x0
i2cNum	uint8_t	1 '\001'
baro_initialized	bool	false
magneto_initialized	bool	false

Figure 13.4: I2C1 initial state

In the pictures below, the debugger will enter the sensors' initialization section (in green) upon reception of an address acknowledge from the sensors on their I2C lines.

14.1.1 Barometer thread initialization

```
37  void ProberThread::loop() {
38
39      if(probeI2C(BMP3_ADDR_I2C_SEC) && !baro_initialized) {      // Looking for Barometer (BMP390)
40          this->instance = new BarometerThread(this);
41          this->baro_initialized = 1;
42  //      xSemaphoreTake(semaphore, portMAX_DELAY);
43      }
44      if(probeI2C(0x1E) && !magneto_initialized) {                // Looking for Magnetometer (ILS2MDCTR)
45          this->instance = new MagnetometerThread(this);
46          this->magneto_initialized = 1;
47      }
48      if(probeI2C(BMI08X_ACCEL_I2C_ADDR_PRIMARY)) {   // Looking for IMU Acc (BMI088)
49          this->instance = new IMUThread(this);
50          xSemaphoreTake(semaphore, portMAX_DELAY);
51  //      vTaskDelay(100 / portTICK_PERIOD_MS);
52  //      vTaskDelay(100 / portTICK_PERIOD_MS);
53      }
```

Figure 13.5: Barometer probing

Name	Type	Value
this	ProberThread * const	0x24010d68 <initCortex::prober1
Thread	Thread	{...}
hi2c	I2C_HandleTypeDef *	0x240002d8 <hi2c1>
semaphore	SemaphoreHandle_t	0x24006ba8 <ucHeap+21928>
instance	BarometerThread *	0x24006cb0 <ucHeap+22192>
i2cNum	uint8_t	1 '\001'
baro_initialized	bool	true
magneto_initialized	bool	false

Figure 13.6: I2C1 state after barometer probing

The breakpoint in Figure 13.5 shows that the probing thread entered in the initialization section of the barometer, meaning that the device was found at its address on the I2C1 line. Its dedicated thread state has now changed to initialized (baro_initialized is now true in Figure 13.6).

14.1.2 Magnetometer thread initialization

```
37  void ProberThread::loop() {
38
39      if(probeI2C(BMP3_ADDR_I2C_SEC) && !baro_initialized) {      // Looking for Barometer (BMP390)
40          this->instance = new BarometerThread(this);
41          this->baro_initialized = 1;
42  //      xSemaphoreTake(semaphore, portMAX_DELAY);
43      }
44      if(probeI2C(0x1E) && !magneto_initialized) {                // Looking for Magnetometer (ILS2MDCTR)
45          this->instance = new MagnetometerThread(this);
46          this->magneto_initialized = 1;
47      }
48      if(probeI2C(BMI08X_ACCEL_I2C_ADDR_PRIMARY)) {   // Looking for IMU Acc (BMI088)
49          this->instance = new IMUThread(this);
50          xSemaphoreTake(semaphore, portMAX_DELAY);
51  //      vTaskDelay(100 / portTICK_PERIOD_MS);
52  //      vTaskDelay(100 / portTICK_PERIOD_MS);
53      }
```

Figure 13.7: Magnetometer probing

Name	Type	Value
this	ProberThread * const	0x24010d68 <initCortex::prober1
Thread	Thread	{...}
hi2c	I2C_HandleTypeDef *	0x240002d8 <hi2c1>
semaphore	SemaphoreHandle_t	0x24006ba8 <ucHeap+21928>
instance	MagnetometerThread *	0x24007dc8 <ucHeap+26568>
i2cNum	uint8_t	1 '\001'
baro_initialized	bool	true
magneto_initialized	bool	true

Figure 13.8: I2C1 state after magnetometer and barometer probing

Likewise, the pictures above show that the magnetometer instance has also been initialized.

As both sensors are now properly instantiated, their data can be monitored on buffers from their dedicated threads.

14.1.3 Barometer data buffers

```
66 void BarometerThread::loop() {
67
68     if(HAL_I2C_GetError(parent->getI2C()) == HAL_I2C_ERROR_NONE)
69     {
70         vTaskSuspendAll();
71         rslt = bmp3_get_sensor_data(BMP3_PRESS_TEMP, &data.baro, &bmp390dev);
72         xTaskResumeAll();
73
74         if(monitor.enter(BARO_MONITOR)) {
75             println("%s", data.toString(cbuf));
76             monitor.exit(BARO_MONITOR);
77         }
78
79
80         data.toArray((uint8_t*) &packet);
81         network.send(&packet);
82         portYIELD();
```

Expression	Type	Value
data.baro	bmp3_data	{...}
temperature	double	25.095284526294563
pressure	double	102487.63899543732

```
Name : data.baro
    Details:{temperature = 25.095284526294563, pressure = 102487.63899543732}
    Default:{...}
    Decimal:{...}
    Hex:{...}
    Binary:{...}
    Octal:{...}
```

Figure 13.9: Barometer data register

Here, the data buffer of the barometer shows a room temperature of 25 degrees and a pressure of 102'487 Pa corresponding to atmospheric conditions at sea level.

14.1.4 Magnetometer data buffers

```
61 void MagnetometerThread::loop() {
62
63     if(HAL_I2C_GetError(parent->getI2C()) == HAL_I2C_ERROR_NONE)
64     {
65         /* Read output only if new value is available */
66         vTaskSuspendAll();
67         iis2mdc_mag_data_ready_get(&iis2dev, &drdy);
68         xTaskResumeAll();
69
70         if (drdy) {
71 //          memset(data_raw_magnetic, 0x00, 3 * sizeof(int16_t));
72             /* Read magnetic field data */
73
74             vTaskSuspendAll();
75             iis2mdc_magnetic_raw_get(&iis2dev, data_raw_magnetic);
76             xTaskResumeAll();
77
78             data.raw_magnetic.x = data_raw_magnetic[0];
79             data.raw_magnetic.y = data_raw_magnetic[1];
80             data.raw_magnetic.z = data_raw_magnetic[2];
81
82             data.magnetic_mG.x = iis2mdc_from_lsb_to_mgauss(data.raw_magnetic.x);
83             data.magnetic_mG.y = iis2mdc_from_lsb_to_mgauss(data.raw_magnetic.y);
84             data.magnetic_mG.z = iis2mdc_from_lsb_to_mgauss(data.raw_magnetic.z);
85
86             /* Read temperature data */
87 //          memset( &data.raw_temperature, 0x00, sizeof(int16_t) );
88             iis2mdc_temperature_raw_get( &iis2dev, &data.raw_temperature );
89             data.temperature_degC = iis2mdc_from_lsb_to_celsius(data.raw_temperature);
90
91             if(monitor.enter(MAGNETO_MONITOR)) {
92                 println("%s", data.toString(cbuf));
93                 monitor.exit(MAGNETO_MONITOR);
94             }
95
96             data.toArray((uint8_t*) &packet);
97             network.send(&packet);
98             portYIELD();
```

Expression	Type	Value
data	Drone_magneto_data	{...}
raw_temperature	int16_t	-21
raw_magnetic	Vector2	{...}
x	int16_t	209
y	int16_t	-41
z	int16_t	-258
temperature_degC	float	22.375
magnetic_mG	Vector	{...}
x	float	313.5
y	float	-61.5
z	float	-387

```
Name : data
    Details:{raw_temperature = -21, raw_magnetic = {x = 209, y = -41, z = -258}, temperature_degC = 22.375
    Default:{...}
    Decimal:{...}
    Hex:{...}
    Binary:{...}
    Octal:{...}
```

Figure 13.10: Magnetometer data register

On the magnetometer side, the data buffer reads magnetic values ranging from 60 mGauss to almost 400 mGauss, which is in the range of the Earth's magnetic field. Further, a temperature of 22.375 degrees can be read. The gap between the measured temperature of the two sensors might result from an initial offset and will require to be troubleshooted.

14.2 - Monitoring

As the current SWD debug mode does not support real time monitoring, another method needs to be used to continuously show a variation of sensor data when moving the board around. This is done via the UART1 to USB bridge implemented on the board which allows to monitor the sensors data in real time on a computer terminal.

To enable the USB communication on the user's computer, one needs to look for the serial port to which the board is plugged in. Here, the device appears as ***SLAB_USBtoUART***.
The terminal monitor can then be launched with the UART1 baud rate (9600 Bd):

screen /dev/cu.SLAB_USBtoUART 9600

A monitor then appears on the terminal with the status of the sensors.

Figure 13.11: Terminal monitor initialization

To display the sensors' data, simply type the following command:

monitor enable

A new interface is then called, showing the sensors' data in real time. The current terminal refresh rate was set to 5 Hz to prevent it from crashing (the display setting can be configured in the *Terminal.cpp* file).

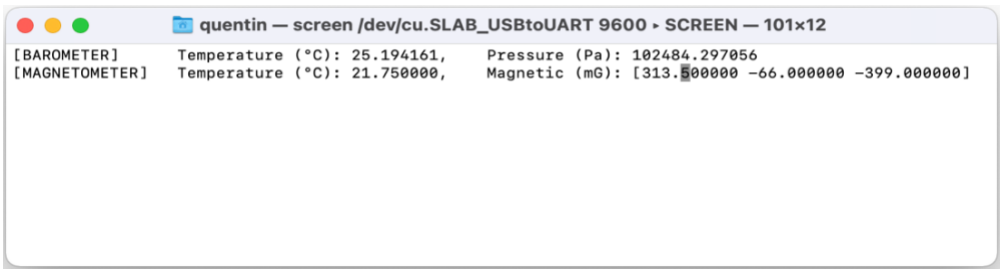


Figure 13.12: Terminal monitor enabled

15 – Safety

Please make sure to carefully read this section before any manipulation is to be intended on the board.

To ensure a safe operation, the following measures needs to be taken every time the board must be powered. Any failure to pass the checkpoints below should result in an immediate interruption of the power up sequence.

15.1 – Powered on battery

- Place the board on a flat non-conductive surface
- Make sure that the board isn't already powered
- Check for short circuits between the VCC and GND lines using a voltmeter
- Check for short circuits between the two pins of the board's XT30PW female connector
- Place the drone battery (or power supply) next to the board
- Check if the battery output is positive and strictly below 10 V
- Connect the battery's XT30PW male connector to the board's XT30PW female connector.

15.2 – Powered by USB

- Place the board on a flat non-conductive surface
- Make sure that the board isn't already powered
- Check for short circuits between the VCC and GND lines using a voltmeter
- Check for short circuits between the D+ and D- lines
- Check for short circuits between the VBUS and GND lines
- Check for short circuits between the VBUS_REV and GND lines
- Check that the other end of the USB cable is connected to a dedicated COST module or computer
- Connect the micro-USB end of the cable to the board

16 – Suggested improvements

Upon completion of the project, the following improvements were suggested:

- Implement reverse voltage protection (at least up to -10V)
- Add a buzzer for state debugging and power up notification
- Add the sensor orientations to the PCB silkscreen
- Place the vertical time of flight sensor on the bottom layer of the board
- Add mechanical fixations to the GPS board
- Look for headers with reduced height to interface with the upper layer of the drone
- Place the 4*1 power header on a side of the board to easily access it
- Change the SWD connector to a single orientation one
- Connect the SWO pin to the microcontroller to facilitate SWD debugging

17 – Conclusion

The first iteration of the avionics module presented in this report was thought of as a modular platform to experience on and learn from. As a project of this kind has never been attempted by a student team in any rover competition, the rules framing the development of this project are still unclear and will likely evolve in the years to come.

Coming back to the original requirements for this project, we believe that the boards that were developed this semester have answered the team's expectations.
Through the integration of 4 navigation sensors on the avionics board, the test of 2 of them and the design and test of a separate GPS module, we have demonstrated that the proposed avionics design is appropriate and properly functioning.
Further, the numerous interfaces that are available on the avionics board make it easily interfaced with any kind of sensor, external board, or computer.

On the Jetson side, while the SPI communication remains an issue to be solved, the USB protocol can for now be used as an alternative solution and allow the project to continue with the design of the main computer board and power supply.

18 – Acknowledgments

As this project arrives at its end, I would like to particularly thank Pr. Alexandre Schmid for allowing it to happen in the first place. His supervision from an academical level ensured that the project was well under way and that it could meet the requirements defined in accordance with the Xplore team.

Further, I wish to thank Mr. Arion Zimmermann as technical supervisor of the project. His experience in the domain has been more than enlightening and rewarding. I could not have hoped to learn as much as I did without his input.

I also wish to extend my thanks to Xplore's avionics team, and particularly Mr. Vincent Nguyen (hardware team leader) and Mr. Yassine Bakkali (software team leader), for their technical advice and for sharing their extensive troubleshooting experience.

Finally, I would like to thank the Spot coaching team for allowing me to access the DLL (Discovery Learning Labs) electronics infrastructures and for forming me on the machines necessary for the assembly of PCBs.

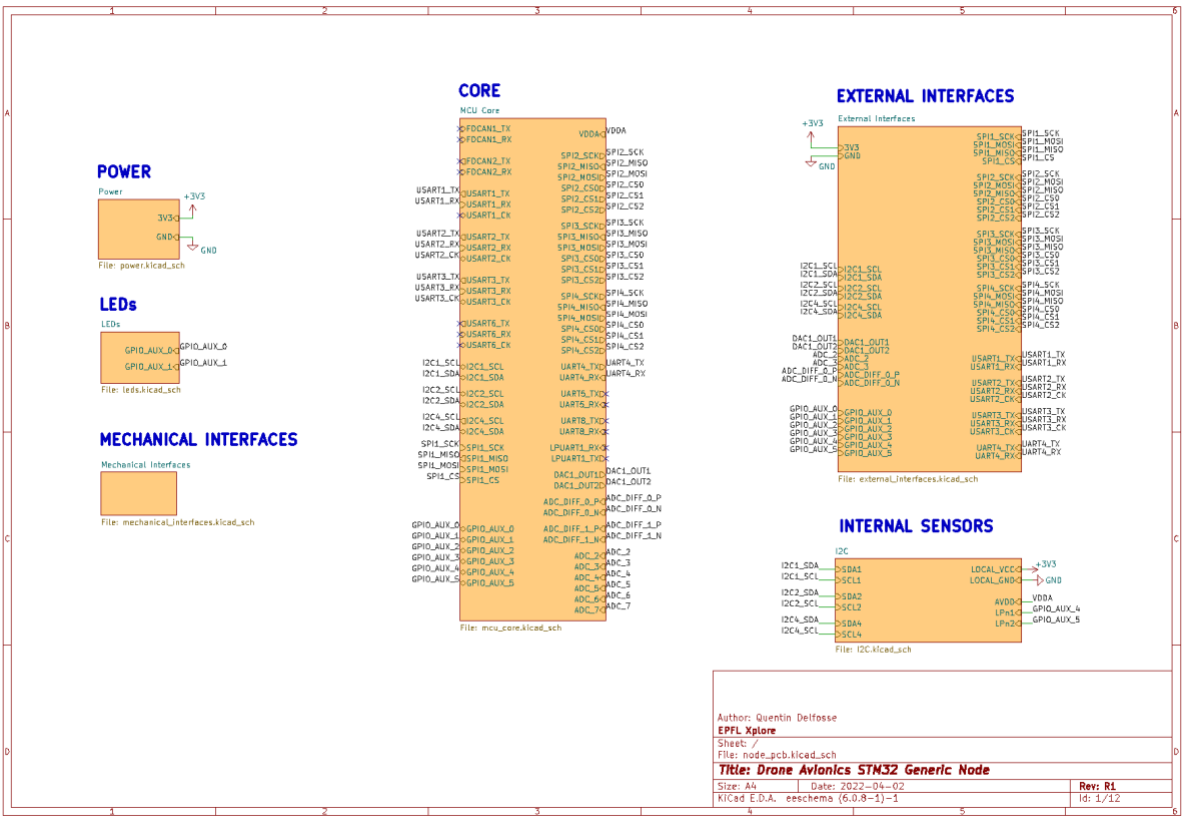
19 – References

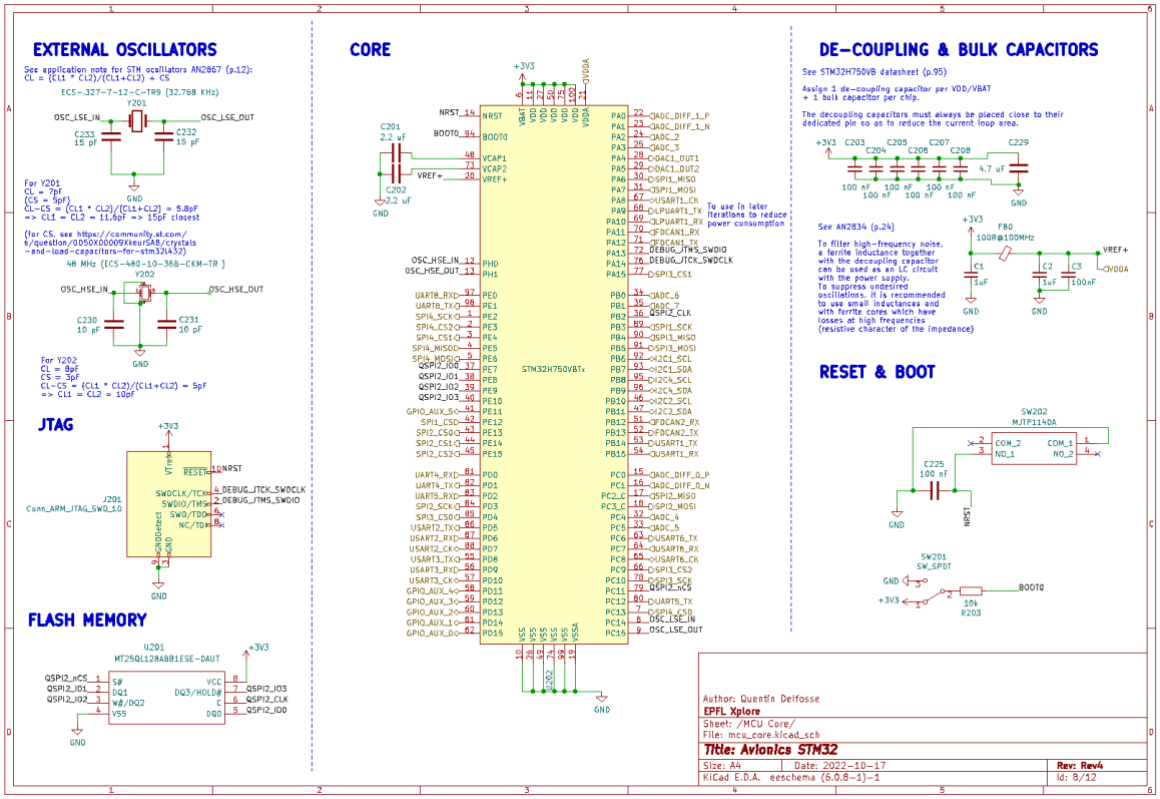
ID	Title	Link
[1]	Nvidia Jetson Nano Website Page	https://developer.nvidia.com/embedded/jetson-nano
[2]	Nvidia Jetson TX2 Module Website Page	https://developer.nvidia.com/embedded/jetson-tx2-nx
[3]	Raspberry Compute Module 4 Website Page	https://www.raspberrypi.com/products/compute-module-4/?variant=raspberry-pi-cm4001000
[4]	STM32H750 Datasheet	https://www.st.com/resource/en/datasheet/stm32h750xb.pdf
[5]	AN2834	https://www.st.com/resource/en/application_note/cd00211314-how-to-get-the-best-adc-accuracy-in-stm32-microcontrollers-stmicroelectronics.pdf
[6]	AN5307	https://www.st.com/resource/en/application_note/an5307-getting-started-with-stm32h7a37b3-line-and-stm32h7b0-value-line-microcontroller-hardware-development-stmicroelectronics.pdf
[7]	AN2867	https://www.st.com/resource/en/application_note/cd00221665-oscillator-design-guide-for-stm8afals-stm32-mcus-and-mpus-stmicroelectronics.pdf
[8]	ECX_12 (LSE Crystal) Datasheet	https://ecsxtal.com/store/pdf/ECX_12.pdf
[9]	LSE Oscillator Decoupling - ST Forum	https://community.st.com/s/question/0D50X00009XkeurSAB/crystals-and-load-capacitors-for-stm32l432
[10]	CX_2236B-1649452 (HSE Crystal) Datasheet	https://www.mouser.ch/datasheet/2/122/ECX_2236B-1649452.pdf
[11]	AN4760	https://www.st.com/resource/en/application_note/an4760-quadspi-interface-on-stm32-microcontrollers-and-microprocessors--stmicroelectronics.pdf
[12]	TPS77833 Datasheet	https://www.ti.com/lit/ds/symlink/tps777.pdf
[13]	MFU0805FF03000P100 Datasheet	https://www.mouser.fr/datasheet/2/427/mfuserie-1762582.pdf
[14]	SLVA689	www.ti.com/lit/an/slva689/slva689.pdf
[15]	BMI088 Datasheet	https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmi088-ds001.pdf
[16]	IIS2MDC Datasheet	https://www.st.com/resource/en/application_note/an5080-iis2mdc-highaccuracy-ultralowpower-3axis-digital-output-magnetometer-stmicroelectronics.pdf
[17]	BMP390 Datasheet	https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp390-ds002.pdf
[18]	MAX M10S Datasheet	https://content.u-blox.com/sites/default/files/MAX-M10S_IntegrationManual_UBX-20053088.pdf

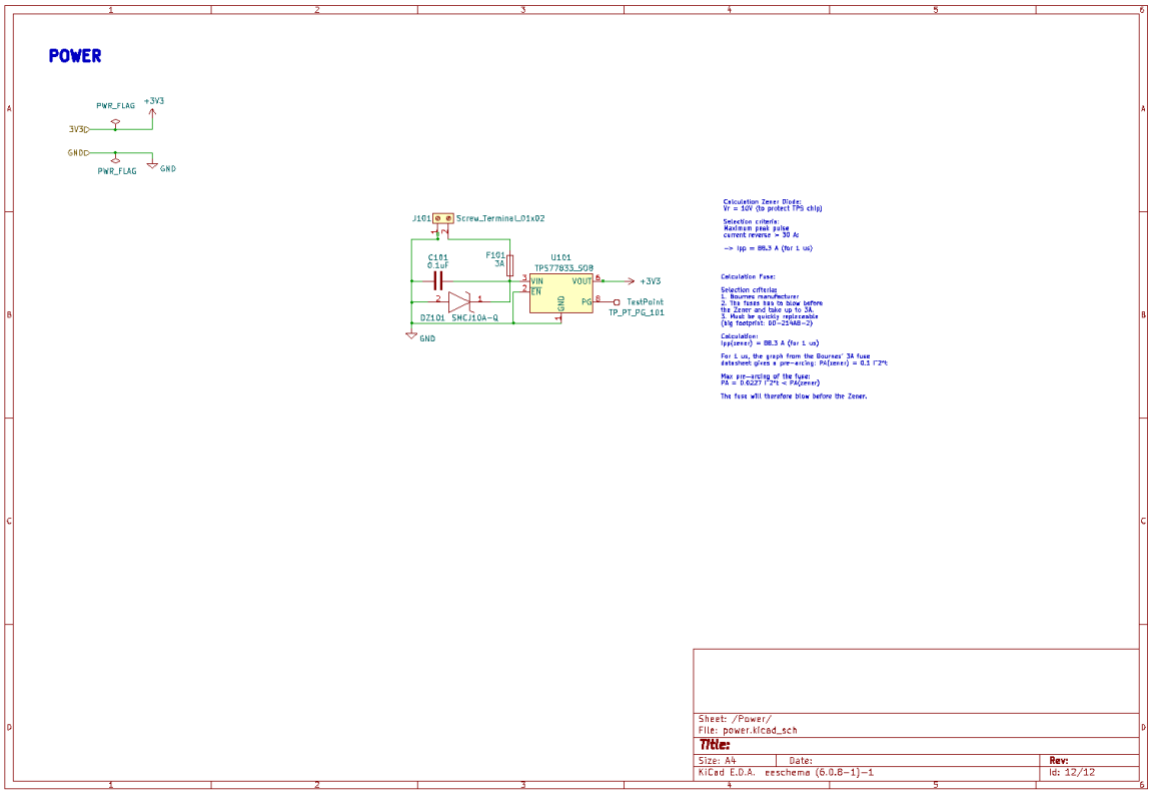
[19]	GPS Coin Cell Datasheet	https://www.sii.co.jp/en/me/datasheets/ms-rechargeable/ms518se/
[20]	Antenna Datasheet	https://www.mouser.ch/datasheet/2/447/GPSGBXXXX-2903608.pdf
[21]	VL53L5CX Datasheet	https://www.mouser.ch/datasheet/2/389/dm00797923-3048842.pdf
[22]	VL53L0X Datasheet	www.st.com/resource/en/application_note/an4846-using-multiple-vl53l0x-in-a-single-design-stmicroelectronics.pdf
[23]	XT30PW-M Datasheet	https://docs.rs-online.com/ed7b/A700000008956679.pdf
[24]	CP2102N Datasheet	https://www.silabs.com/documents/public/data-sheets/cp2102n-datasheet.pdf
[25]	NCV8161 Datasheet	https://www.onsemi.com/pdf/datasheet/ncv8161-d.pdf
[26]	SRV05-4 Datasheet	http://www.onsemi.com/pub/Collateral/SRV05-4-D.PDF
[27]	SMF5.0CA-TP Datasheet	https://www.mouser.ch/datasheet/2/240/Littelfuse_TVS_Diode_SMF_Datasheet.pdf-1698977.pdf
[28]	PJA3405 Datasheet	https://www.panjit.com.tw/upload/datasheet/PJA3405.pdf
[29]	0805L150SLYR Datasheet	https://www.littelfuse.com/~/media/electronics/datasheets/resettable_ptcs/littelfuse_ptc_low_rho_datasheet.pdf.pdf
[30]	Screw Loading Calculator	https://www.tribology-abc.com/calculators/e3_6d.htm
[31]	Aisler Design Rules	https://community.aisler.net/t/pcb-design-rules/41

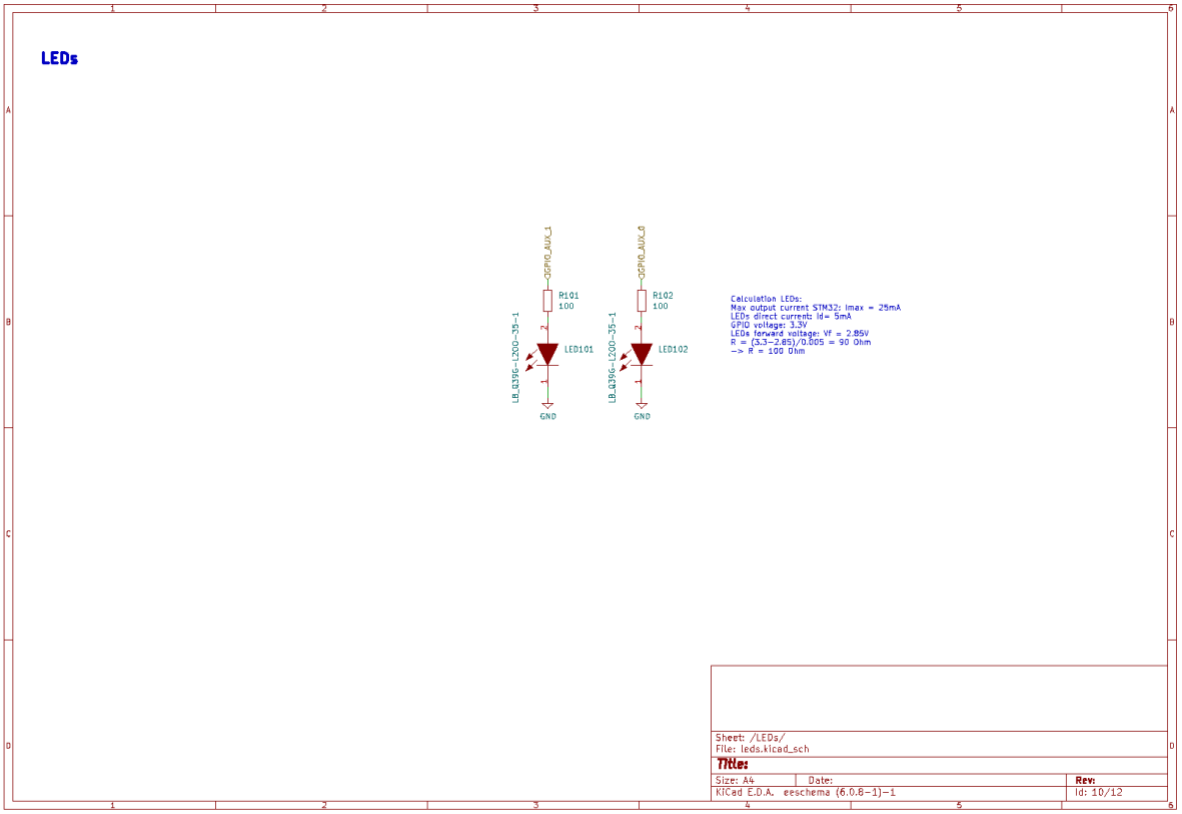
20 – APPENDICES

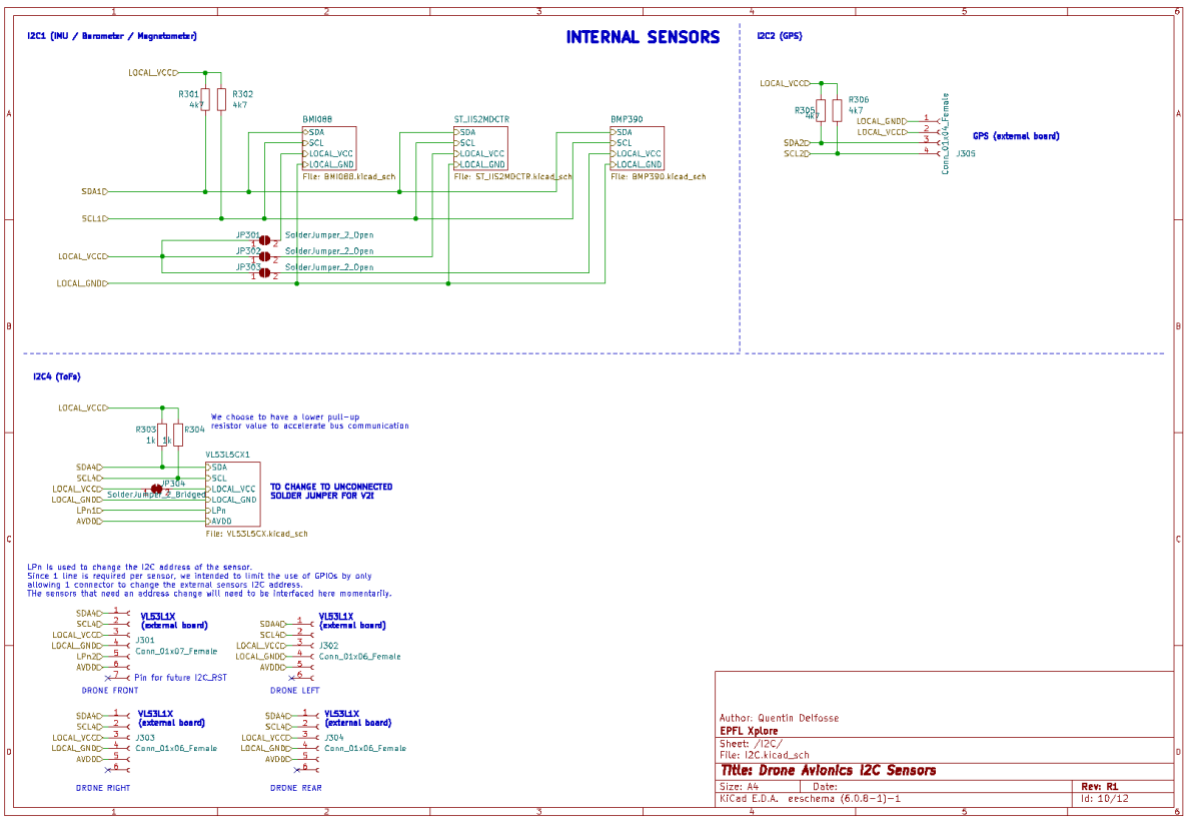
APPENDIX I - SCHEMATICS

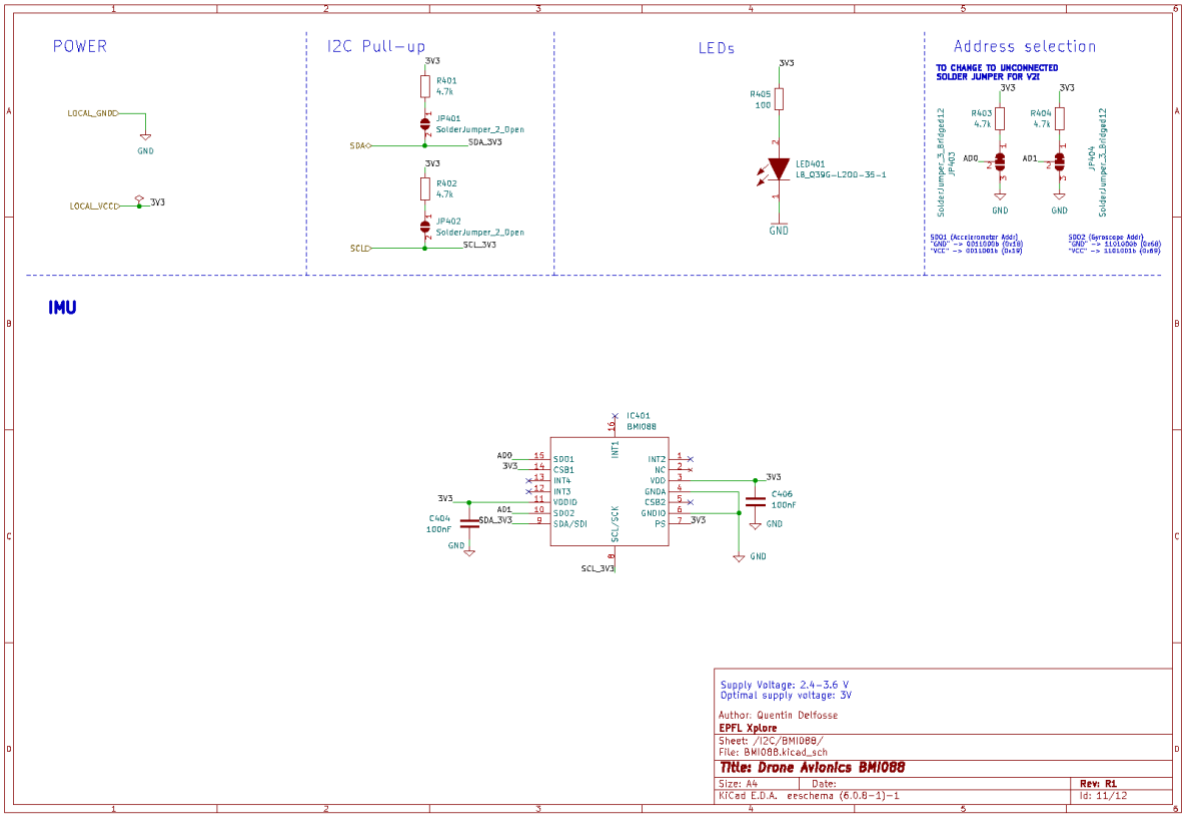


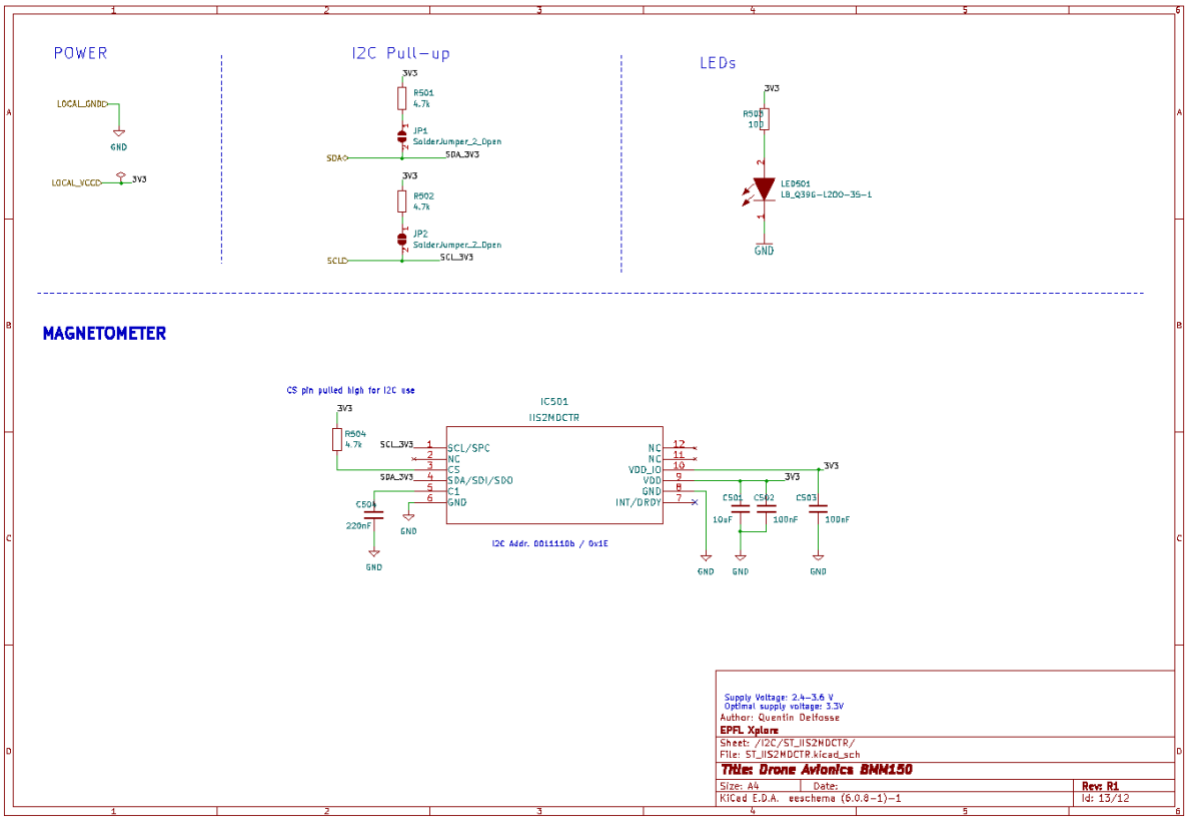


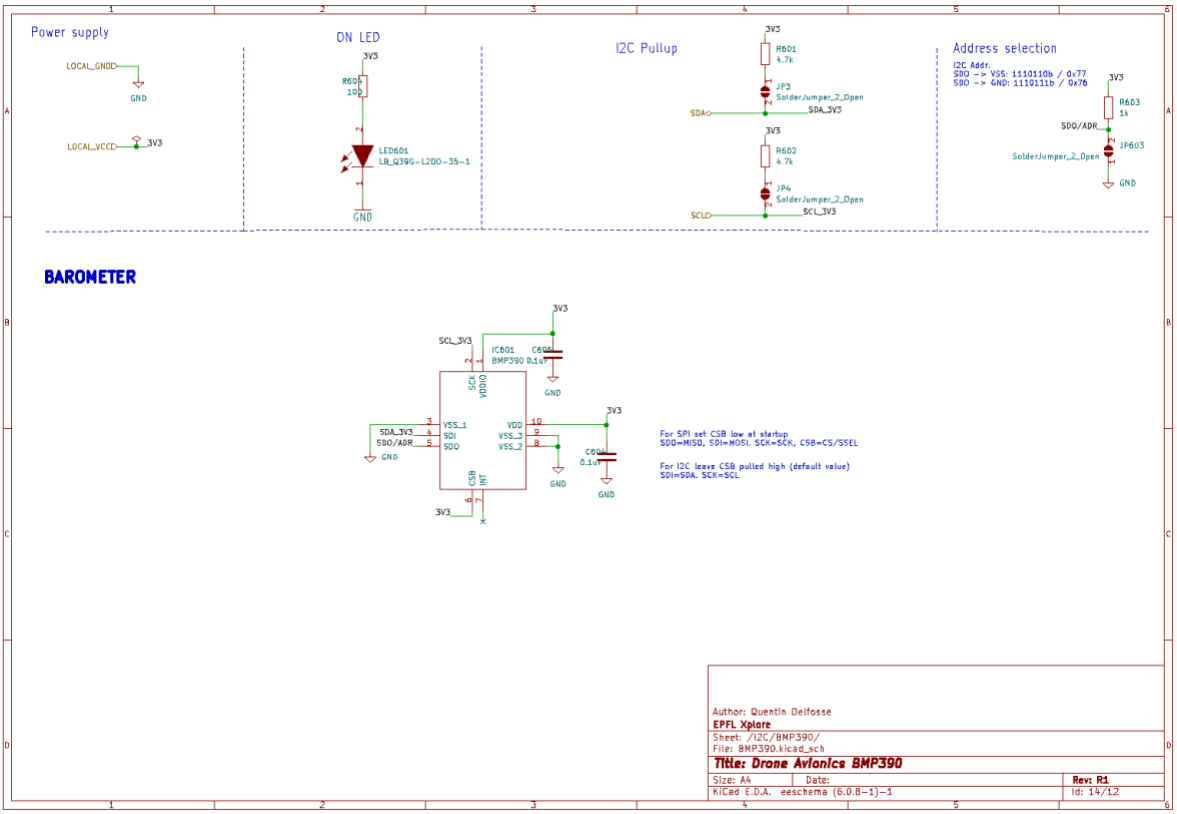


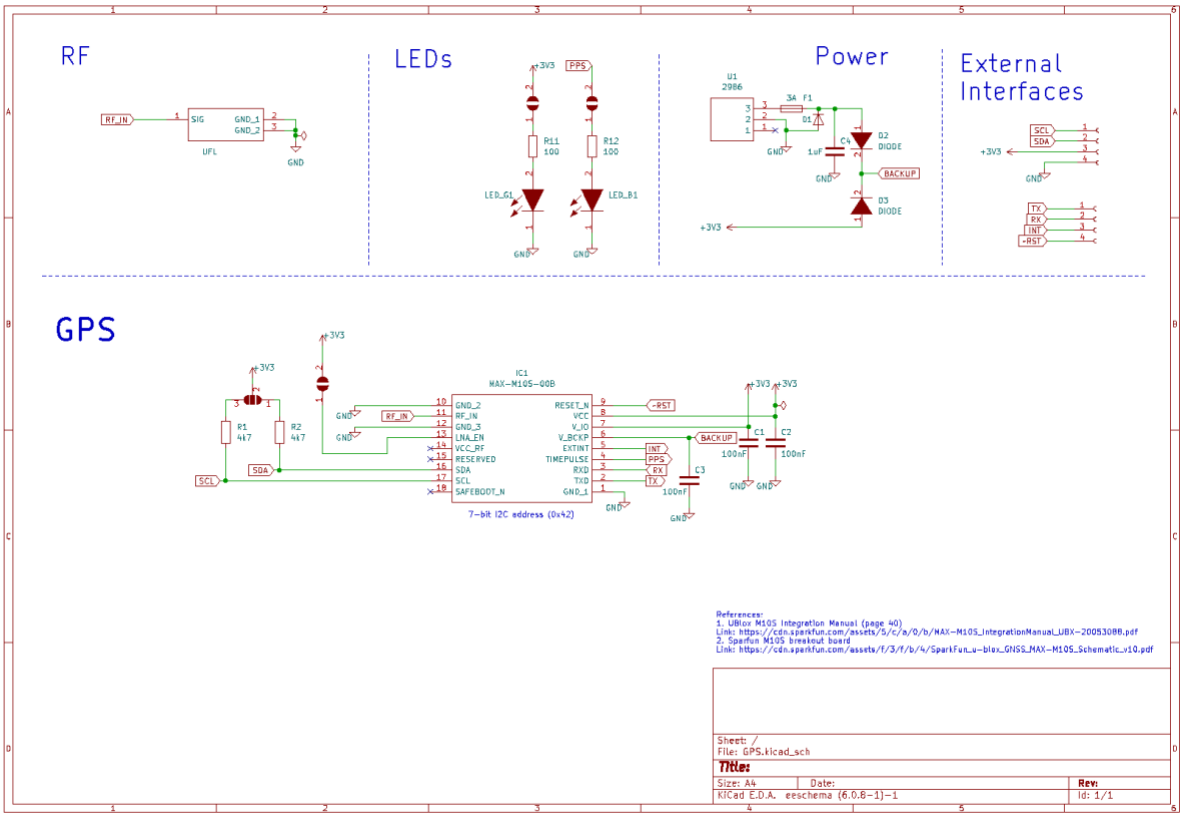


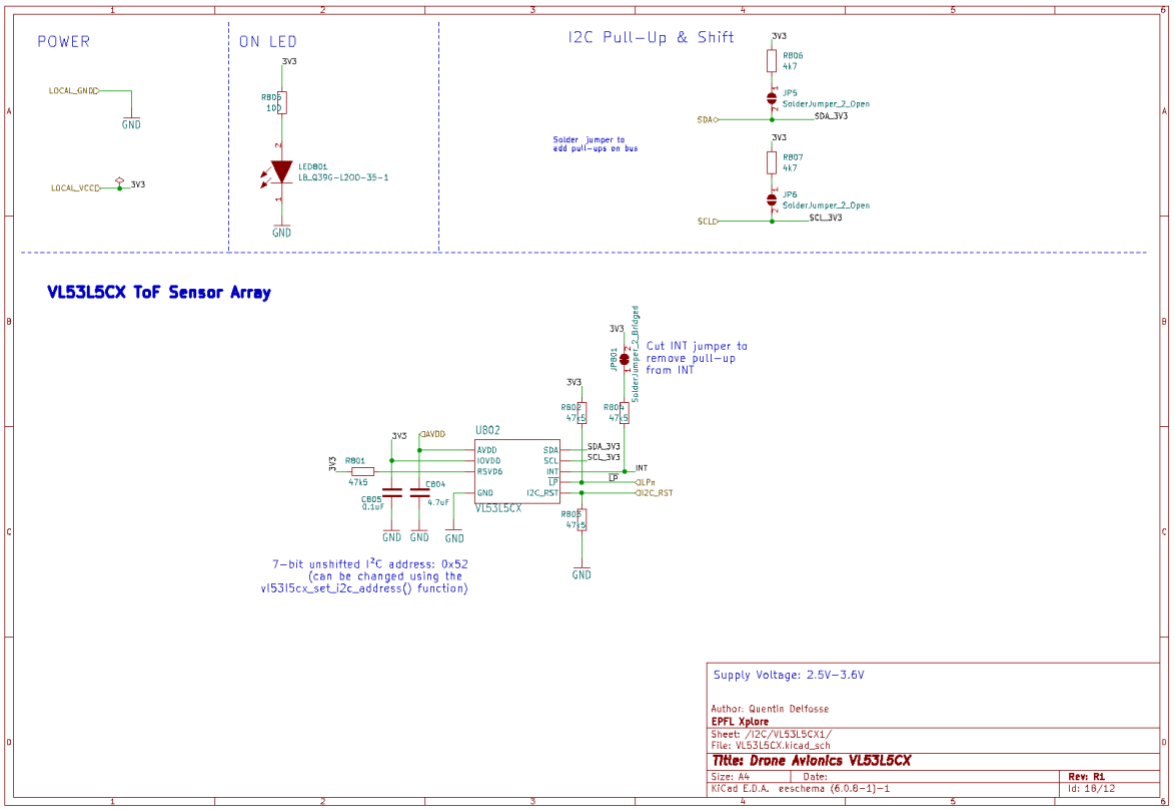


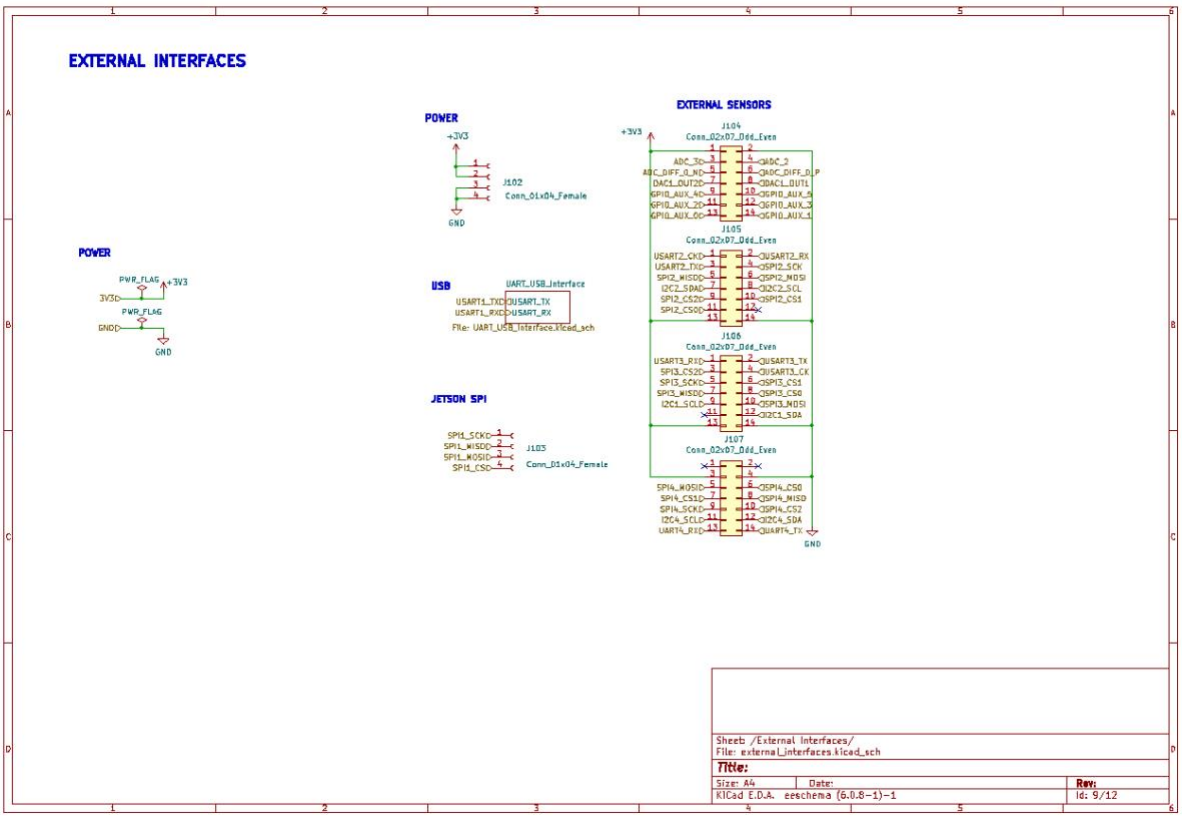


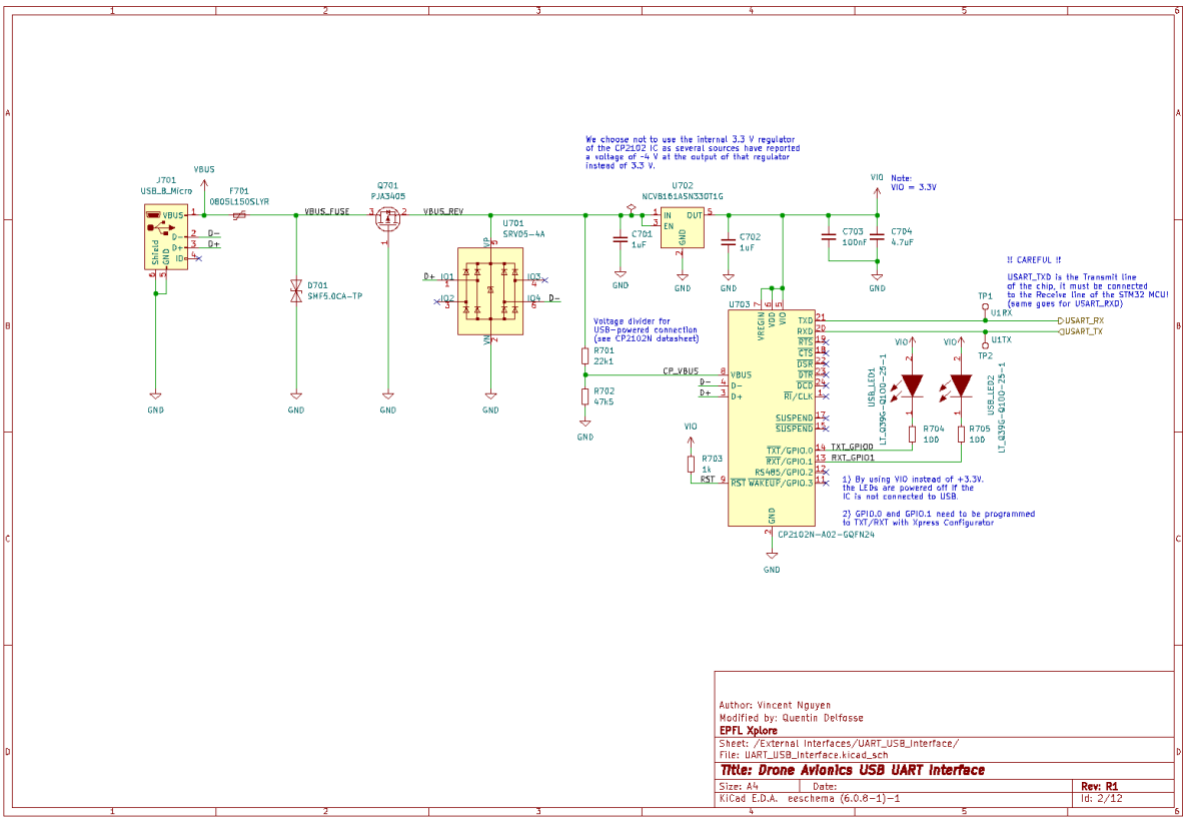


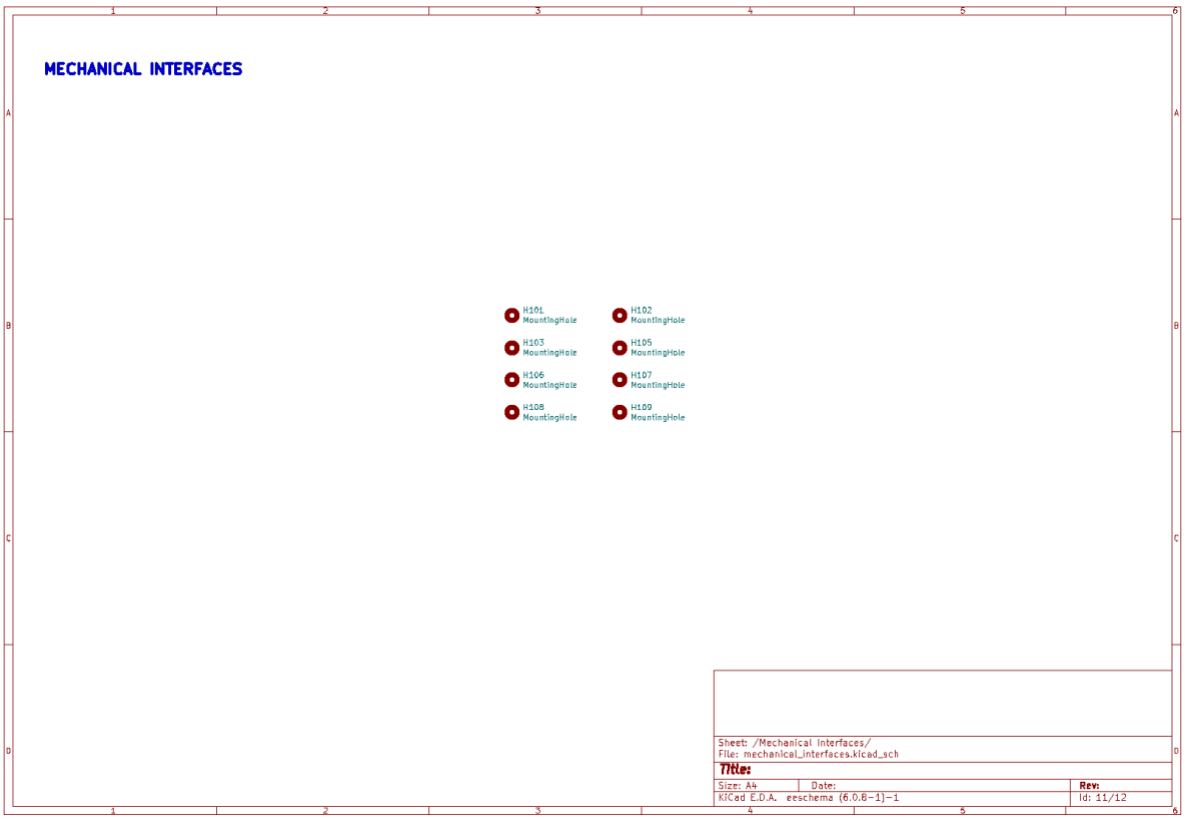






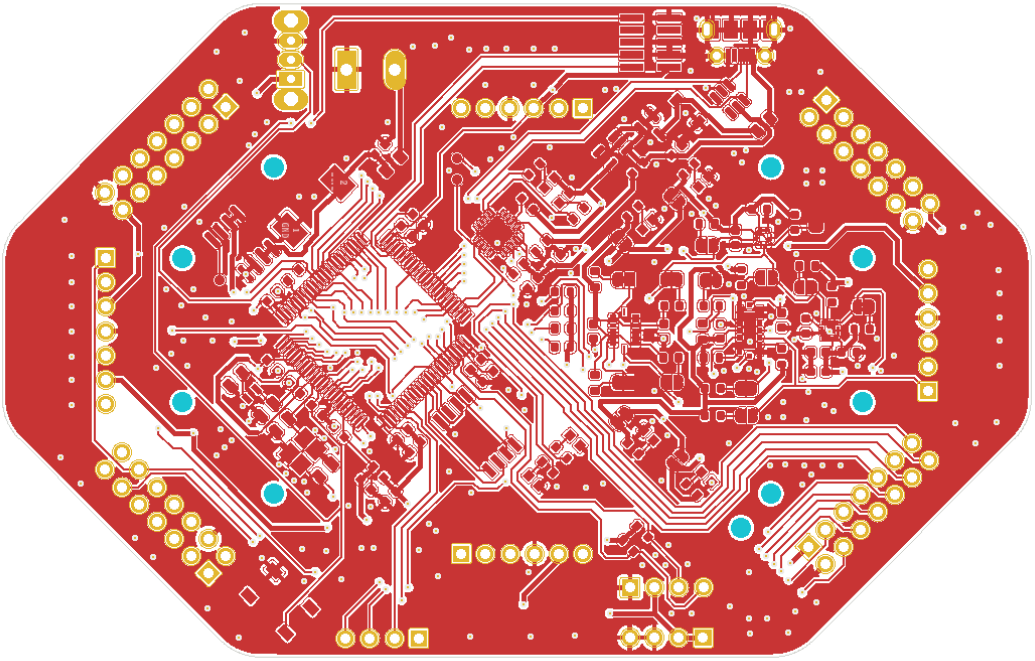




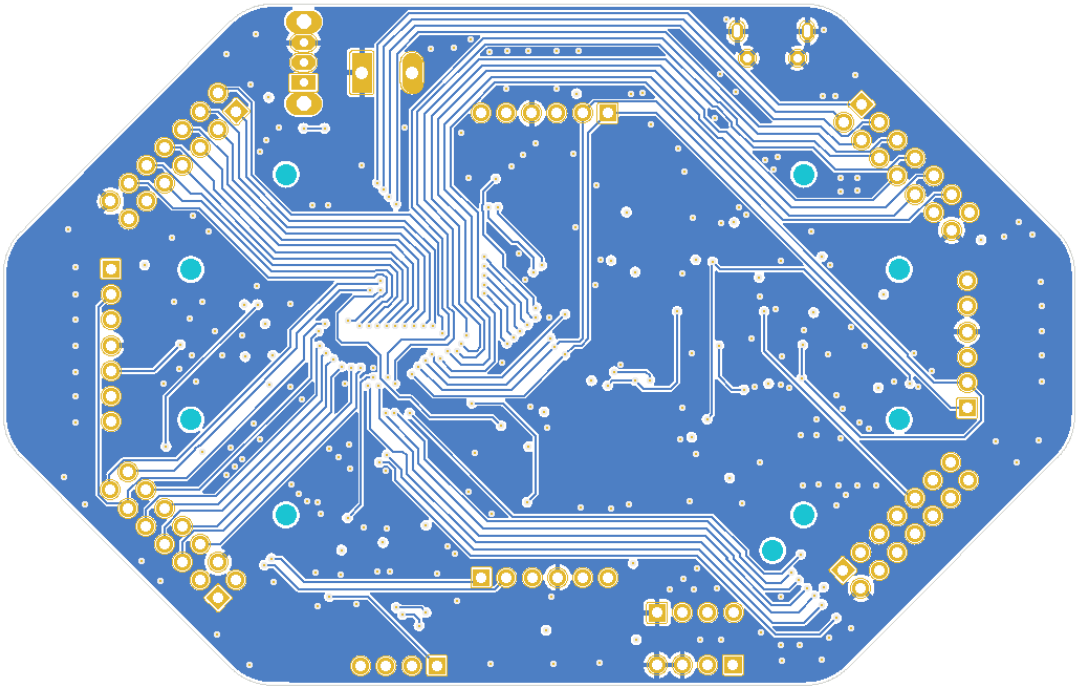


APPENDIX II – LAYOUTS

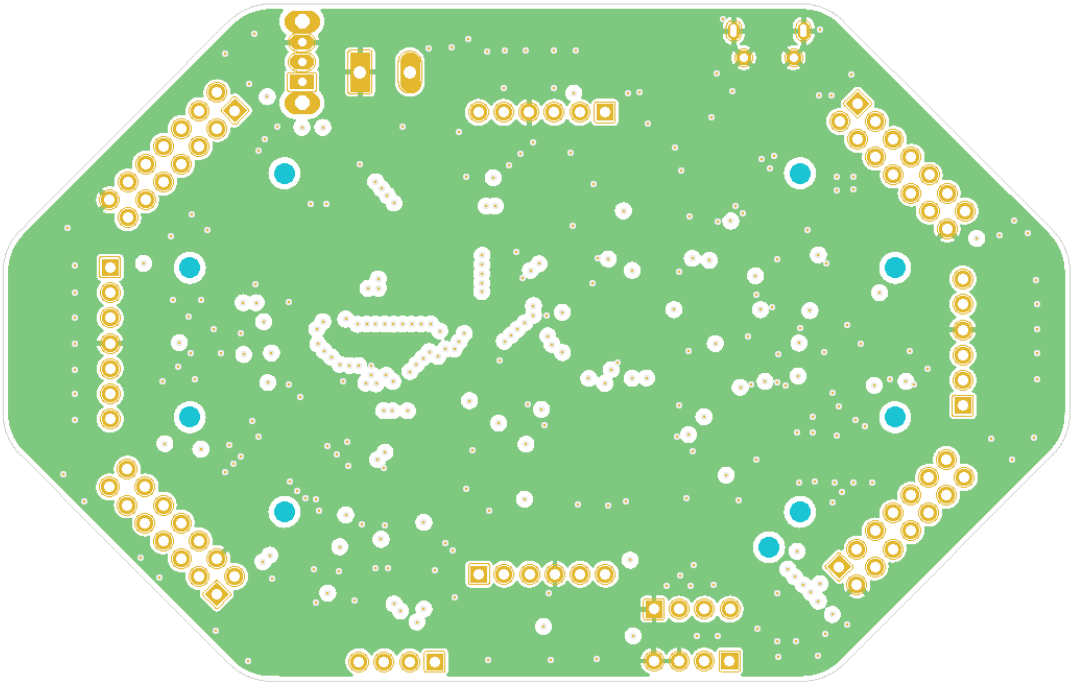
AVIONICS PCB - TOP LAYER LAYOUT (PWR / SIGNAL / GND)



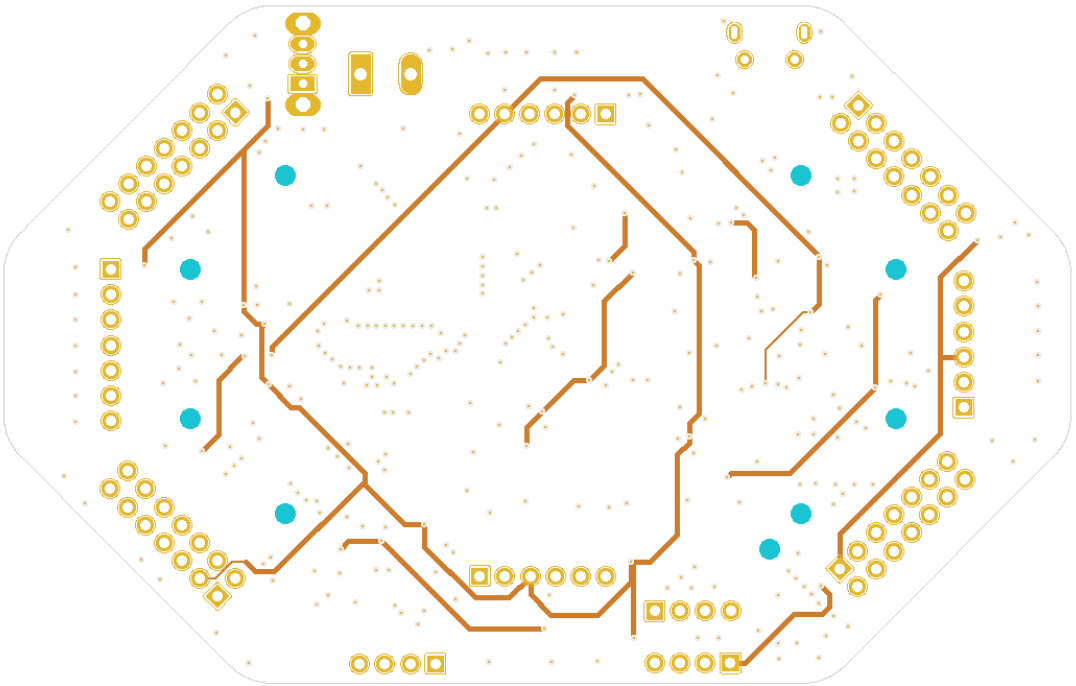
AVIONICS PCB -BOTTOM LAYER LAYOUT (SIGNAL)



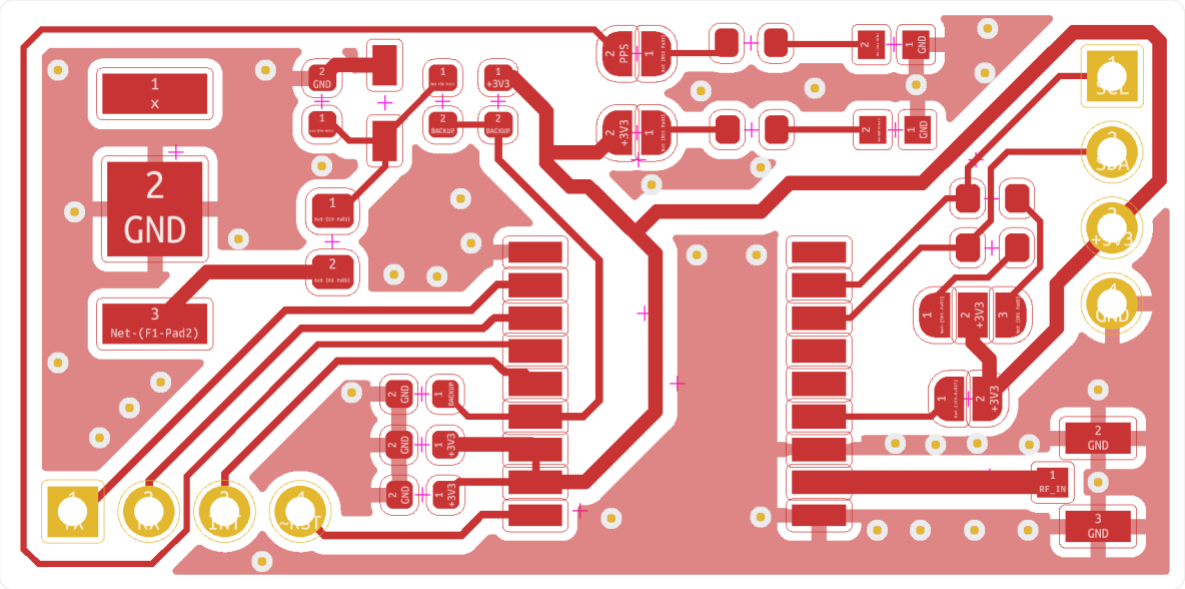
AVIONICS PCB - INTERNAL LAYER 1 LAYOUT (GND)



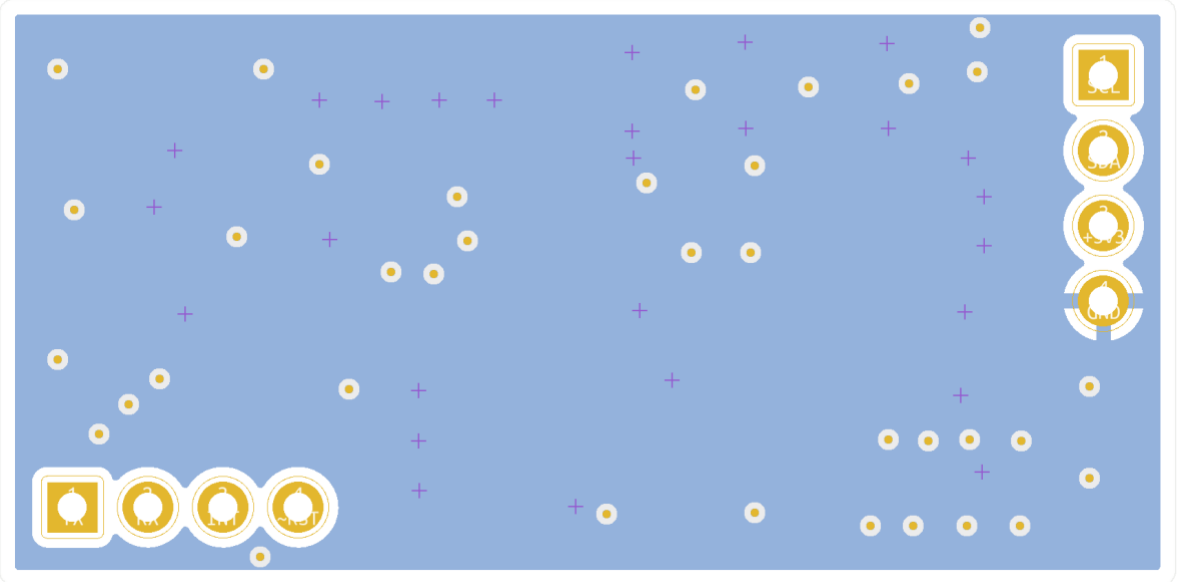
AVIONICS PCB - INTERNAL LAYER 2 LAYOUT (PWR)



GPS PCB - TOP LAYER LAYOUT (PWR / SIGNAL / GND)



GPS PCB – BOTTOM LAYER LAYOUT (GND)



APPENDIX III – BILL OF MATERIALS

AIVIONICS PCB BOM

Reference	Qty	Value	Manufacturer reference
C1, C2, C701, C702	4	1 uF	C0603C105K8RACAUTO
C201, C202	2	2.2 uF	C0603C225K8RAC7411
C203, C204, C205, C206, C207, C208, C225, C404, C406, C3, C703, C101, C604, C605, C805, C502, C503	19	0.1 uF	C0603C104M4RAC
C230, C231	2	10 pF	C0603C100K4RACAUTO
C232, C233	2	15 pF	C0603C150K5RAC7867
C504	1	220 nF	C0603X224K8RACAUTO
C501	1	10 uF	C0603C106M9PACTU
C704, C804, C229	3	4.7 uF	GRM188Z71A475ME15D
D701	1	-	SMF5.0CA-TP
DZ101	1	-	SMCJ10CA-Q
F101	1	3 A	MFU0805FF03000P100
F701	1	1.5 A	0805L150SLYR
FB0	1	100R@100MHz	BLM18KG101TH1D
IC401	1	-	BMI088
IC501	1	-	IIS2MDCTR
IC601	1	-	BMP390
J101	1	-	XT30PW-M
J102, J103, J305	3	Conn_01x04_Female	90147-1204
J104, J105, J106, J107	4	Conn_02x07_Odd_Even	215309-7
J201	1	Conn_ARM_JTAG_SWD	20021121-00010C4LF
J301	1	Conn_01x07_Female	216602-7
J302, J303, J304	3	Conn_01x06_Female	M20-7910642R
J701	1	USB_B_Micro	105017-0001
LED101, LED102, LED401, LED501, LED601, LED801	8	Blue	LB Q39G-L2OO-35-1
LED701, LED702	4	Green	LT_Q39G-Q1OO-25-1
Q701	1		PJA3405_R1_00001
R101, R102, R405, R503, R604, R805, R704, R705	8	100 Ω	ERA-3VEB1000V

R203	1	10 kΩ	ERA-3ARB103V
R401, R402, R403, R404, R501, R502, R504, R601, R602, R301, R302, R305, R306, R806, R807	15	4.7 kΩ	ERA-3ARW472V
R303, R304, R603, R703	4	1 kΩ	ERA-3AED102V
R701	1	22.1 kΩ	ERA-3AED2212V
R702, R801, R802, R803, R804	5	47.5 kΩ	ERA-3ARB4752V
SW201	1	-	OS102011MS2QN1
SW202	1	-	MJTP1140A
U101	1	-	TPS77833D
U201	1	-	MT25QL128ABB1ESE-0AUT
U202	1	-	STM32H750VBTx
U701	1	-	SRV05-4A
U702	1	-	NCV8161ASN330T1G
U703	1	-	CP2102N-A02-GQFN24
U802	1	-	VL53L5CX
Y201	1	32.768 KHz	ECS-.327-7-12-C-TR9
Y202	1	48 MHz	ECS-480-10-36B-CKM-TR

GPS PCB BOM

Reference	Qty	Value	Manufacturer reference
C1, C2, C3	3	100 nF	C0603C104M4RAC
C4	1	1 uF	TMCJ1C105MTRF
D1	1	-	UCLAMP3301D.TCT
D2, D3	2	-	SD0603S040S0R2
F1	1	3 A	MFU0805FF03000P100
IC1	1	-	MAX-M10S-00B
J1	1	-	CONUFL001-SMD-T
J2, J3	2	Conn_01x04_Male	90120-0764
LEB_B1	1	Blue	LB Q39G-L2OO-35-1
LED_G1	1	Green	LT Q39G-Q1OO-25-1
R1, R2	2	4.7 kΩ	ERA-3ARW472V
R11, R12	2	100 Ω	ERA-3VEB1000V
U1	1	-	2986
A1	1	-	GPSGB1330

APPENDIX IV – BOARD DIMENSIONS