Report

Arduino UNO SMD using ATmega328p (CSUduino V1, Customized for CSU Côte d'Azur)

Ali Elkerm

R&D Embedded Spacecraft Electronics Engineer

Observatoire de la Côte d'Azur

aly.elkarm@oca.eu

January 5, 2026

Version	Date	Author	Description / Changes
1.0	2026-01-05	A. ELKERM	Initial draft with Grove, QWIIC, & USB-C

Revision History

Contents

1 Introduction **1**

2 Methodology **3**

2.1 Schematic Design 3

2.2 Footprints Assignment 5

2.3 Layout Design 6

3 Results and Discussion **9**

3.1 3D Design 9

4 Conclusion **10**

1. Introduction

This report describes in details the steps followed to design the full PCB layout of an Arduino Uno based on ATmega328p customized for academic use. It briefly introduces the concept of the circuit, discusses the methodology followed by the circuit schematic explanation, the footprints assignmenet, the PCB layout design, and finally the preparation for the PCB production.

An Arduino Uno is a microcontroller developped by Arduino and is widely used by students and amateur electronics eggineers and hobbyists as a powerful and relatively easy to use and program microcontroller board.

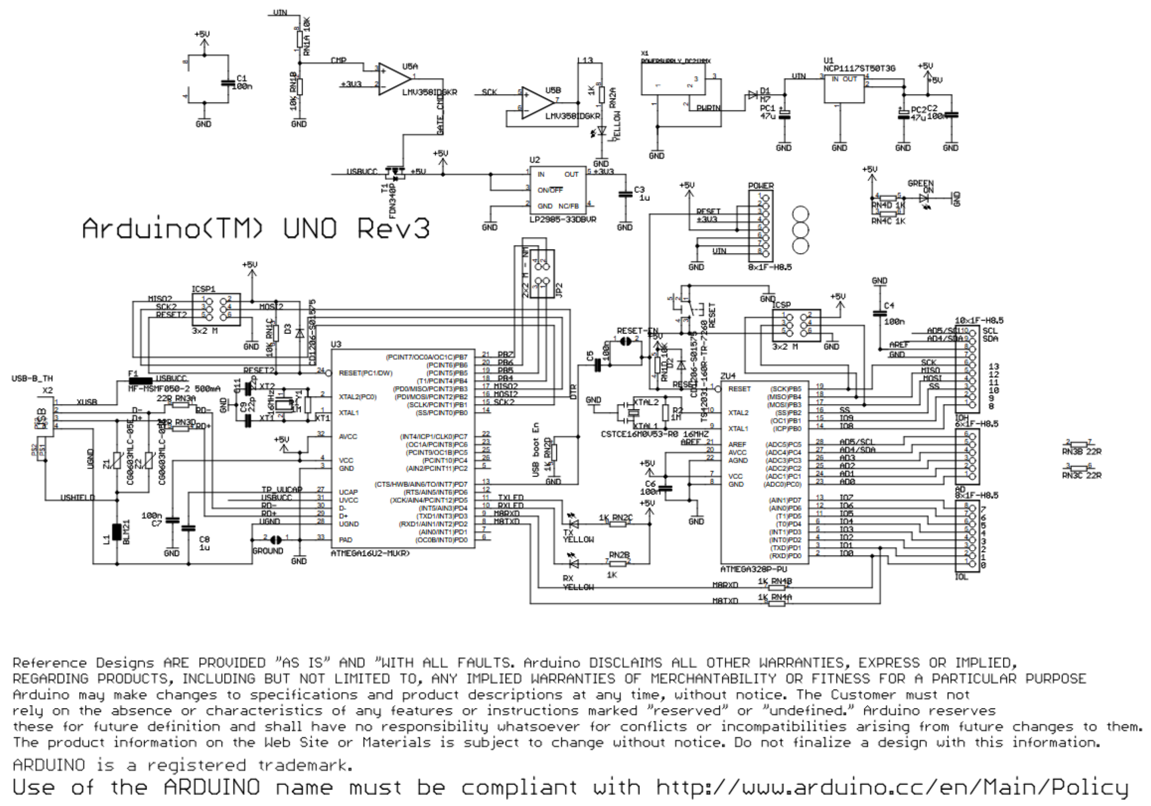
The controller of the circuit is of the type ATmega328p which is a single microcontroller unit manufactured by Microship Technology. It beloongs to the megaAVR family and it has a modified 8-bit RiSC processor core. The circuit schematic is shown in Figure 1.1

The design discussed in this document discusses a customized design of the stndard one shown in the previous figure. The main procedure has been adopted with some changes to adapt the current technologies and for the purpose of training bachlor as well as masters students to use simple microcontrollers with the addition of some important interfaces to connect to other commercial modules.

The main modifications done on this design are the following

1. Replacing the USB-B input port by a USB-C one
2. Adding a Grove Connector for serial communication (UART)
3. Adding a QWIIC Connector for I2C communication

I will discuss in depth the modifications done with a complete guide of the design considerations, trade-offs and risks.



Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.
ARDUINO is a registered trademark.
Use of the ARDUINO name must be compliant with http://www.arduino.cc/en/Main/Policy

Figure 1.1: Arduino UNO Rev3 Schematic

2. Methodology

A PCB design is completed through the completion of these 4 steps:

1. The Circuit schematic design
2. Assigning footprints
3. Layout design (This project is a 4-layers PCB)
4. Signals Routing
5. Generation of Gerber files, BoM, and pick and place files to send to the manufacturing house

2.1 Schematic Design

During this step, following the Aduino Uno Rev3 standard schematic was crucial to insure a reliable final schematic design, it is the first yet the fundamental block of the design because it specifies the electrical components and sets the nets that will be used later on in routing the signals over KiCAD.

The next step was the one where the changes were introduced. Since most devices and electrical sensors, modules, etc.. at the moment have a USB-C port as the main external interface, replacing the USB-B by a USB-C port was the first decision. FOllowing this step adds a small detail which is connecting two pull resistors to regulate the current flow.

Best practice is to create different sheets for each part of the drawing and connecting these Hierarchical sheets together at the end using Hierarchical pins.

The schematic design was split into the following Hierarchical sheets:

1. ATmega328p circuit
2. USB to serial converter ATmega16U2
3. Power circuit
4. Power select circuit (between the USB port and the barrel)

5. Connectors (pin sockets, pin headers, QWIIC, and Grove)

6. Mechanical assembly

Grouping each components in a separate sheets based on their specific functions help with the component placement later on, it is cleaner and easier to debug and correct errors.

As mentioned in the introduction, this board is a customized board having both a QWIIC and a Grove connector. Grove connector is provided by Seeed while QWIIC is provided by SparkFun Electronics. Grove can be driven by 2 voltages, either 3.3V or 5V. On the other hand, QWIIC connector can only operate using 3.3V.

Nevertheless, a very important design consideration was to fix the reported issue regarding the 3.3V modules that connects to the Arduino Uno connectors. Arduino Uno users repeatedly reported that connecting external modules or devices operating at 3.3V damages devices.

The solution adopted in this design was to add a low drop out voltage regulator to drop the Arduino Uno 5V to the 3.3V for Grove and QWIIC. The component chosen was SPX2920M3-3.3 SOT223. It is a 400 mA LDO to respect having an output current less than the 1A output current of the first stage (barrel to 5V).

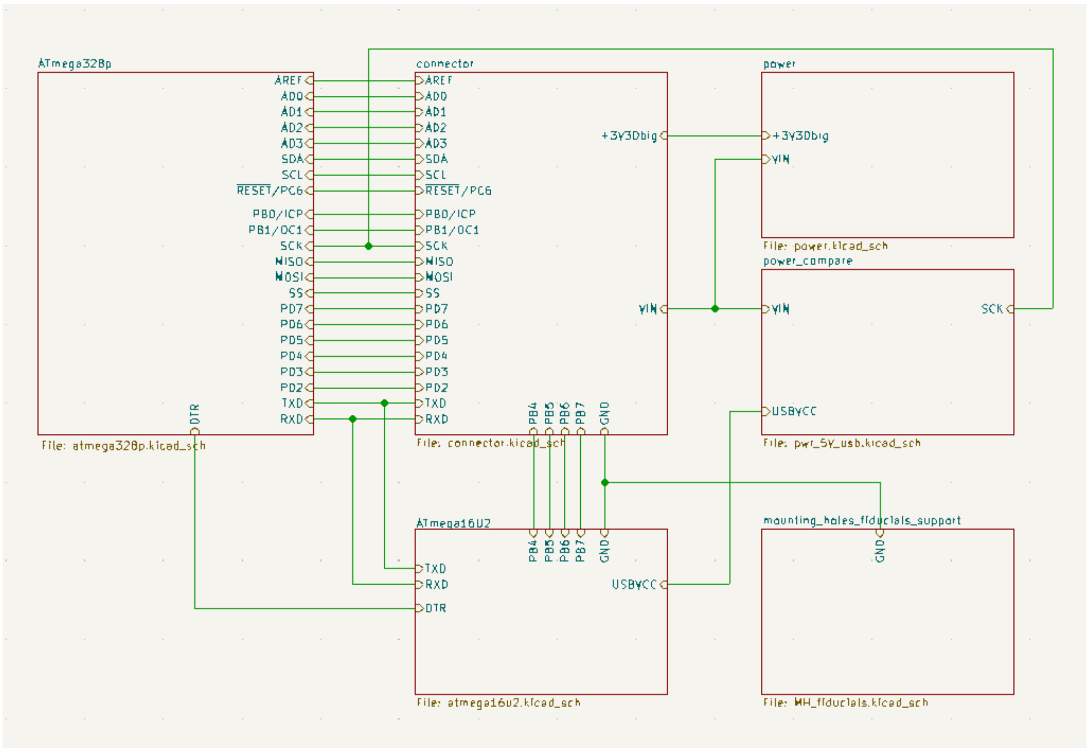
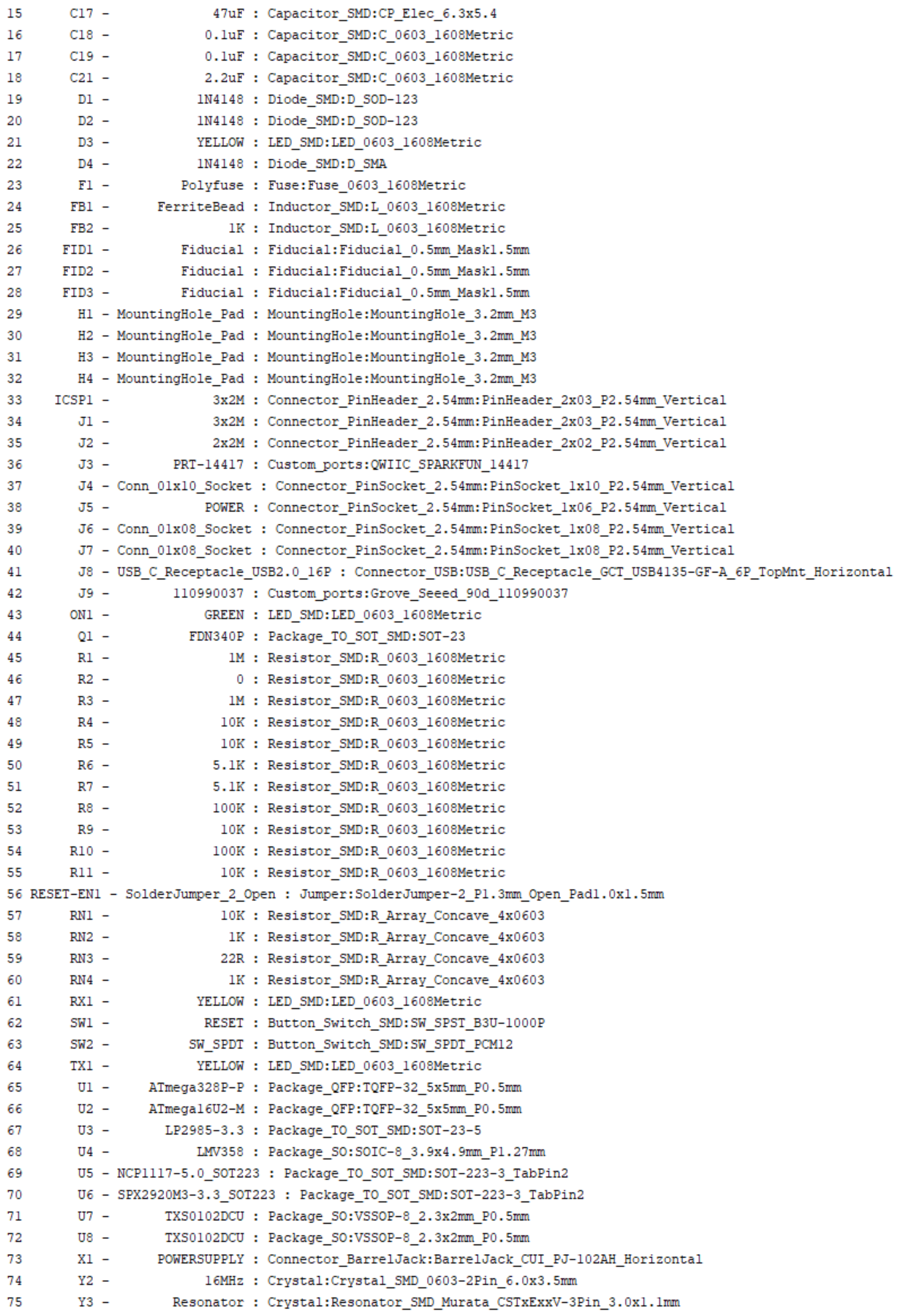


Figure 2.1: Arduino Uno Hierarchical Sheets

The full hierarchical schematic is shown in Figure 2.1.

After finalizing the schematic design, it is crucial to run the Electrical Rules Checker (ERC) to ensure there are no errors or violations in the schematic design.

2.2 Footprints Assignment



```
15      C17 -            47uF : Capacitor_SMD:CP_Elec_6.3x5.4
16      C18 -           0.1uF : Capacitor_SMD:C_0603_1608Metric
17      C19 -           0.1uF : Capacitor_SMD:C_0603_1608Metric
18      C21 -           2.2uF : Capacitor_SMD:C_0603_1608Metric
19       D1 -          1N4148 : Diode_SMD:D_SOD-123
20       D2 -          1N4148 : Diode_SMD:D_SOD-123
21       D3 -          YELLOW : LED_SMD:LED_0603_1608Metric
22       D4 -          1N4148 : Diode_SMD:D_SMA
23       F1 -        Polyfuse : Fuse:Fuse_0603_1608Metric
24      FB1 -     FerriteBead : Inductor_SMD:L_0603_1608Metric
25      FB2 -              1K : Inductor_SMD:L_0603_1608Metric
26     FID1 -        Fiducial : Fiducial:Fiducial_0.5mm_Mask1.5mm
27     FID2 -        Fiducial : Fiducial:Fiducial_0.5mm_Mask1.5mm
28     FID3 -        Fiducial : Fiducial:Fiducial_0.5mm_Mask1.5mm
29       H1 - MountingHole_Pad : MountingHole:MountingHole_3.2mm_M3
30       H2 - MountingHole_Pad : MountingHole:MountingHole_3.2mm_M3
31       H3 - MountingHole_Pad : MountingHole:MountingHole_3.2mm_M3
32       H4 - MountingHole_Pad : MountingHole:MountingHole_3.2mm_M3
33    ICSP1 -            3x2M : Connector_PinHeader_2.54mm:PinHeader_2x03_P2.54mm_Vertical
34       J1 -            3x2M : Connector_PinHeader_2.54mm:PinHeader_2x03_P2.54mm_Vertical
35       J2 -            2x2M : Connector_PinHeader_2.54mm:PinHeader_2x02_P2.54mm_Vertical
36       J3 -       PRT-14417 : Custom_ports:QWIIC_SPARKFUN_14417
37       J4 - Conn_01x10_Socket : Connector_PinSocket_2.54mm:PinSocket_1x10_P2.54mm_Vertical
38       J5 -           POWER : Connector_PinSocket_2.54mm:PinSocket_1x06_P2.54mm_Vertical
39       J6 - Conn_01x08_Socket : Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Vertical
40       J7 - Conn_01x08_Socket : Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Vertical
41       J8 - USB_C_Receptacle_USB2.0_16P : Connector_USB:USB_C_Receptacle_GCT_USB4135-GF-A_6P_TopMnt_Horizontal
42       J9 -       110990037 : Custom_ports:Grove_Seeed_90d_110990037
43      ON1 -           GREEN : LED_SMD:LED_0603_1608Metric
44       Q1 -         FDN340P : Package_TO_SOT_SMD:SOT-23
45       R1 -              1M : Resistor_SMD:R_0603_1608Metric
46       R2 -               0 : Resistor_SMD:R_0603_1608Metric
47       R3 -              1M : Resistor_SMD:R_0603_1608Metric
48       R4 -             10K : Resistor_SMD:R_0603_1608Metric
49       R5 -             10K : Resistor_SMD:R_0603_1608Metric
50       R6 -            5.1K : Resistor_SMD:R_0603_1608Metric
51       R7 -            5.1K : Resistor_SMD:R_0603_1608Metric
52       R8 -            100K : Resistor_SMD:R_0603_1608Metric
53       R9 -             10K : Resistor_SMD:R_0603_1608Metric
54      R10 -            100K : Resistor_SMD:R_0603_1608Metric
55      R11 -             10K : Resistor_SMD:R_0603_1608Metric
56 RESET-EN1 - SolderJumper_2_Open : Jumper:SolderJumper-2_P1.3mm_Open_Pad1.0x1.5mm
57      RN1 -             10K : Resistor_SMD:R_Array_Concave_4x0603
58      RN2 -              1K : Resistor_SMD:R_Array_Concave_4x0603
59      RN3 -             22R : Resistor_SMD:R_Array_Concave_4x0603
60      RN4 -              1K : Resistor_SMD:R_Array_Concave_4x0603
61      RX1 -          YELLOW : LED_SMD:LED_0603_1608Metric
62      SW1 -           RESET : Button_Switch_SMD:SW_SPST_B3U-1000P
63      SW2 -         SW_SPDT : Button_Switch_SMD:SW_SPDT_PCM12
64      TX1 -          YELLOW : LED_SMD:LED_0603_1608Metric
65       U1 -     ATmega328P-P : Package_QFP:TQFP-32_5x5mm_P0.5mm
66       U2 -     ATmega16U2-M : Package_QFP:TQFP-32_5x5mm_P0.5mm
67       U3 -      LP2985-3.3 : Package_TO_SOT_SMD:SOT-23-5
68       U4 -          LMV358 : Package_SO:SOIC-8_3.9x4.9mm_P1.27mm
69       U5 - NCP1117-5.0_SOT223 : Package_TO_SOT_SMD:SOT-223-3_TabPin2
70       U6 - SPX2920M3-3.3_SOT223 : Package_TO_SOT_SMD:SOT-223-3_TabPin2
71       U7 -      TXS0102DCU : Package_SO:VSSOP-8_2.3x2mm_P0.5mm
72       U8 -      TXS0102DCU : Package_SO:VSSOP-8_2.3x2mm_P0.5mm
73       X1 -     POWERSUPPLY : Connector_BarrelJack:BarrelJack_CUI_PJ-102AH_Horizontal
74       Y2 -           16MHz : Crystal:Crystal_SMD_0603-2Pin_6.0x3.5mm
75       Y3 -       Resonator : Crystal:Resonator_SMD_Murata_CSTxExxV-3Pin_3.0x1.1mm
```

Figure 2.2: Arduino Uno Custom Assigned Footprints

After running the ERC and getting error-free results, the next step is assigning footprints. Footprints are simply how the PCB chip components will physically appear on the PCB after production, this steps is crucial for mechanical assembly.

The footprint choice for this project largely followed the standard Arduino Uno SMD board with some adjustements. Some of the adjustements done are to reduce the occupation surface of some components. Hence, some THT components have been replaced by SMD ones. All resistors, capacitors, inductors and LEDs were assigned to the SMD_0603_1608Metric package. Pin headers and pin sockets are all THT for mechanical endurance with 2.54mm standard dimension. Other components are chosen according to commercial norms.

Figure 2.2 shows the footprints choice based on the industry standards. Almost all footprints exist already in the KiCAD library. Some footprints were added from external libraries such as the Grove footprint (by SEEED) and the QWIIC footprint (by SparkFun Electronics).

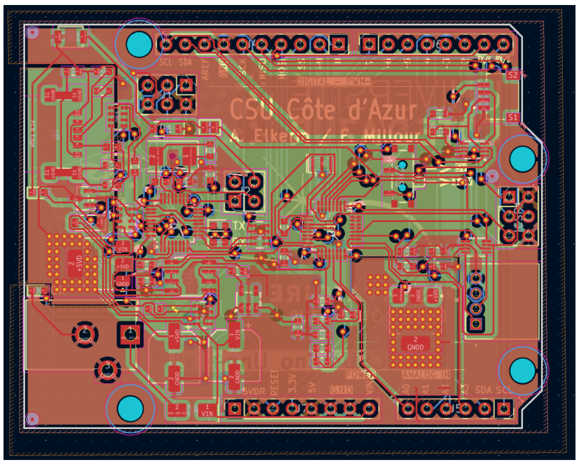
2.3 Layout Design

After finishing the schematic design and the footprint assignment phase, the next step is to proceed with the layout implementation of the design. In the PCB editor, import the Netlist exported from the schematic editor. Once the layout is generated on the PCB editor, the final step is to route signals using general and standard routing techniques. The routing lines in KiCAD are by default tilted by 45 degrees to avoid the 90 degrees tilt that induce impedance mismatch and therefore losses due to reflection. Also, the tracing routes are all set to be 0.25mm in width to reduce the spacing used on the chip. Finally, some references state that the power lines traces are better to be set to 0.5 mm in width. However, this was not feasible due to the small board area and the narrow distances between the pins on the footprint.

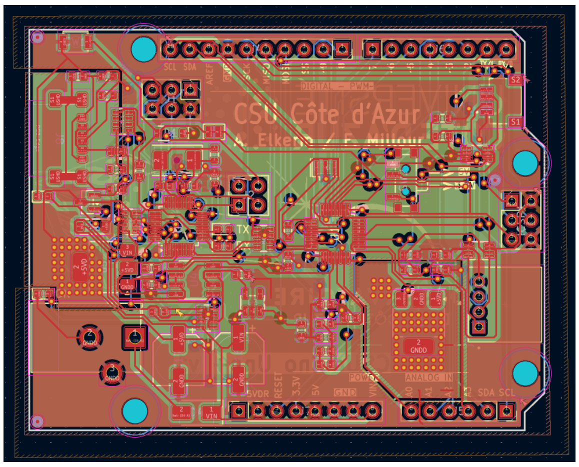
During this phase, the space optimization vs the length of the copper links is an important trade-off. In the PCB industry where miniaturization is of high importance, designing compact circuits is indispensable. On the other hand, routing has to be done with high care to ensure reducing parasitic effects for passive circuit elements (Capacitors and inductors).

Once the routing is done, running the DRC is crucial to make sure there are no errors and warnings.

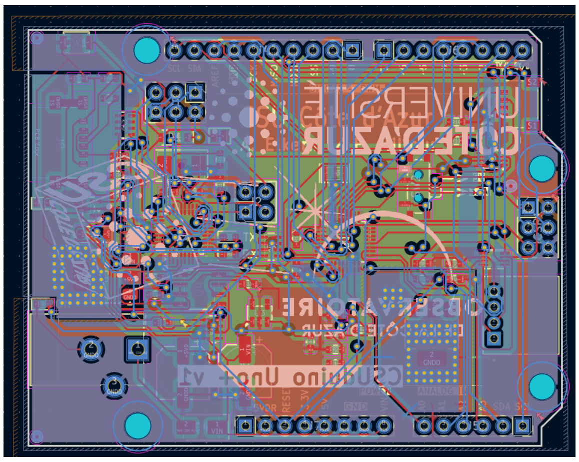
Figure 2.4 shows the different layers in the PCB layout editor. The layout is done using the imported netlist.



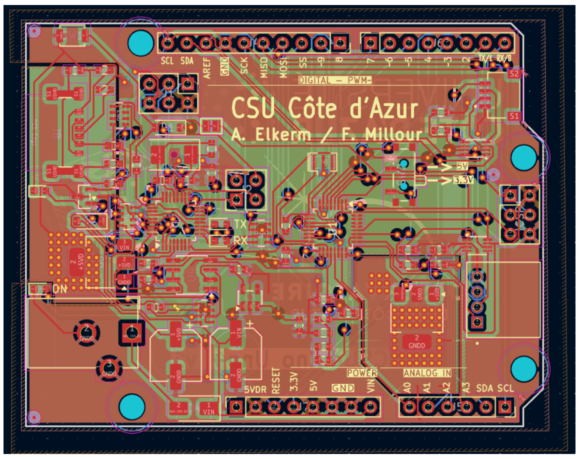
(a) Edge Cuts



(b) F_Cu Layer



(c) B_Cu Layer



(d) Front Silkscreen

Figure 2.4: PCB layout layers

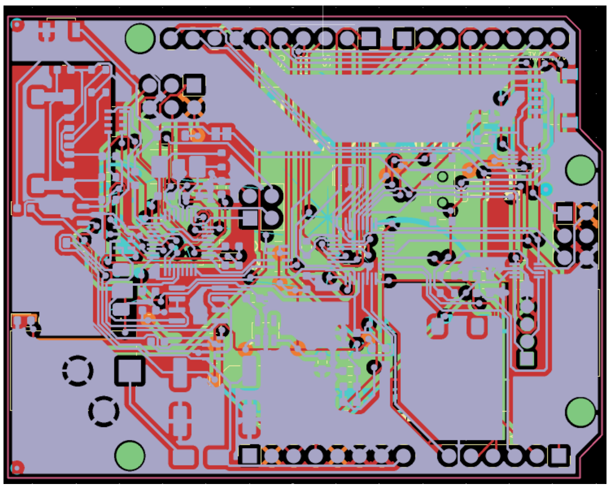


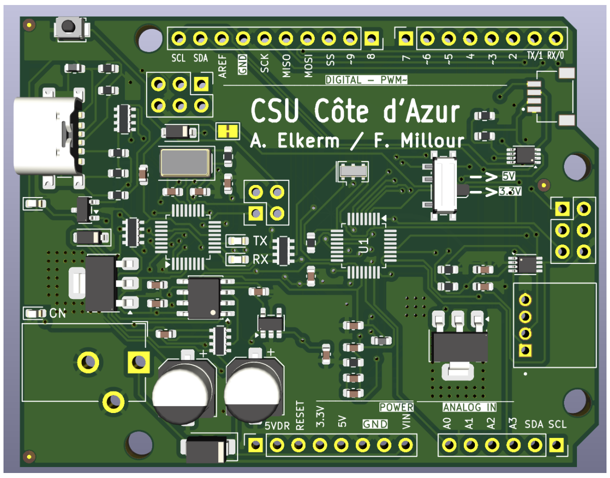
Figure 2.3: Example of gerber file visualization: F_Cu

After finalizing the design and getting no errors or warnings. The design is plotted as gerber files and is visualised in the Gerber viewer app in KiCAD. The gerber files are the `.gbr` files sent to the manufacturing house for production. Figure 2.3 shows the generated gerber file for the PCB top view *F_Cu* from the Gerber Viewer.

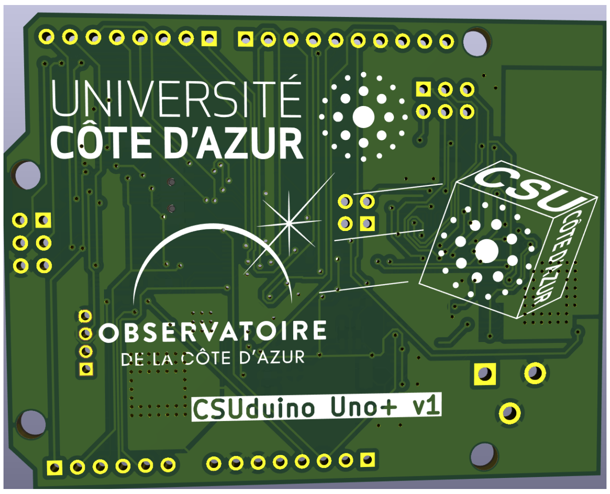
3. Results and Discussion

Figure 3.1 shows the top view and the bottom view of the 3D design of the CSUduino board.

3.1 3D Design



(a) PCB 3D Top View



(b) PCB 3D Bottom View

Figure 3.1: CSUduino board 3D View

4. Conclusion

In this project, I learnt to go through the full cycle of PCB design by doing the PCB design of the custom Arduino Uno board based on ATmega329p (CSUduino V1). This board will be used for academic purposes in the academic space centre of observatoire de la côte d'Azur. I started by doing the schematic of the board and choosing the different analog electronic devices in this circuit to implement the given circuit. Second, I conducted the ERC check to check the validity of the design from the electrical connections and signals. Next, I assigned the footprints which are the type, size and shape of the physical components in the PCB design. I also exported the Netlist from the schematic editor and imported it to the PCB editor to finalize my steps by routing and testing the DRC in the editor. The next steps I learnt to desgn round-edged PCBs while adding the mounting holes fo the mechanical support and the fiducials for the visuals. I also used FreeRouting open source spftware to help with the routing process.