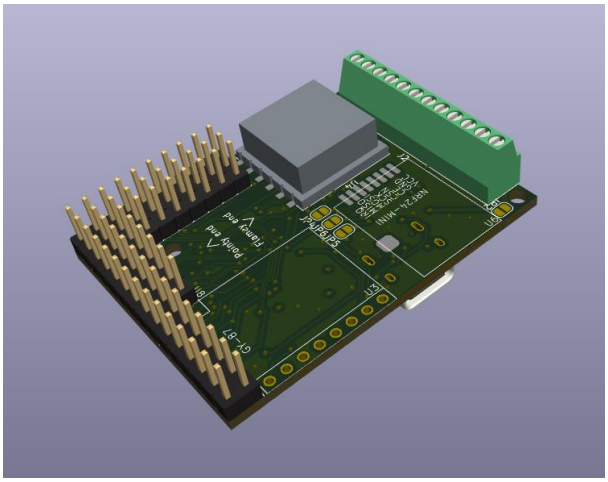
CANSAT FLIGHT COMPUTER USER'S AND MANUFACTURER'S MANUAL



Picture 1: 3D renders of the board

Introduction:

The CANSAT flight computer is an all-in-one, multi-purpose, model rocket flight computer, it can be used to deploy recovery mechanisms, perform measurements, and conduct experiments.

Capabilities:

Processor: STM32F405RGT6

Sensors: MPU6050, BMP180, HMC5883l, L80 GPS

8x PWM ports

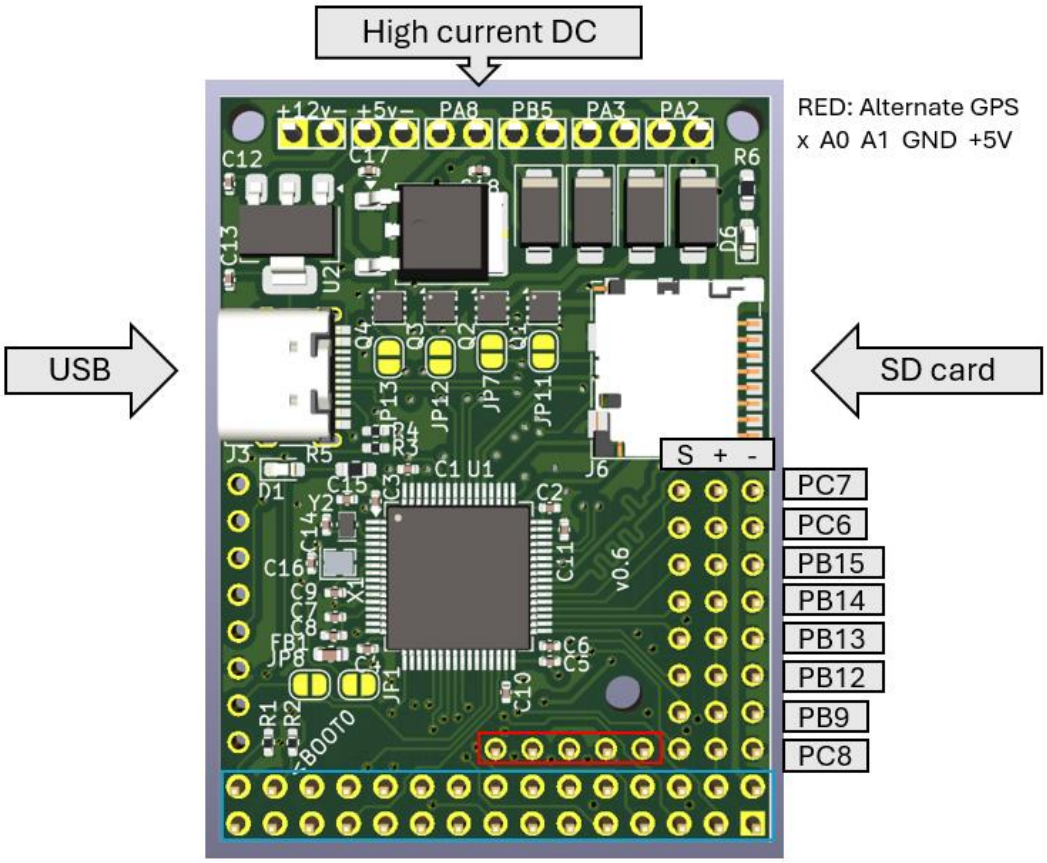
4x High current DC ports rated at 10A momentary

Spare GPIO ports

SD card interface

Size: 51.4096mm x 38.1mm

Pinouts and features:



Picture 2: Pinouts

Note: x denotes pins that are not connected to anything

Important safety information:

1. Only connect power to the device you are using it, NEVER leave it unattended with the power connected
2. Nichrome wire can heat up to dangerous temperatures, DO NOT touch the wire in any circumstances when power is connected
3. The length of the nichrome wire should be similar to the width of the board
4. The firmware must drive all High current channels to low at startup, as the MOSFETs are defaulted to ON
5. Since the MOSFETs are defaulted to the ON state, the microcontroller needs to be powered on whenever the high current channels are powered. See table below for acceptable power situations

Microcontroller	High current channels	Is this situation safe
Off	On	Unsafe
On	Off	Safe
On	On	Safe
Off	Off	Safe

(Rule 4, and 5, only applies to hardware versions prior to v1.0, devices with hardware version v1.0 and later have the MOSFETs defaulted to OFF)

Firmware development:

This board is compatible with Arduino® ecosystem, the prefer tool for firmware development is stm32duino [1], follow the stm32duino documentation to install it in Arduino® IDE.

To turn on a nichrome wire connected to a specific high current DC channel, use digitalWrite(<PIN_NAME>, HIGH). E.g. to turn on the channel closest to the battery terminals (see picture 2), use digitalWrite(PA8, HIGH). Use digitalWrite(<PIN_NAME>, LOW) to turn it off.

Recommended 3rd party libraries, these libraries have extensive documentation and easy to understand examples:

https://docs.arduino.cc/libraries/mpu6050_light/

https://docs.arduino.cc/libraries/bmp180mi/

https://docs.arduino.cc/libraries/tinygpsplus/

https://docs.arduino.cc/libraries/rf24/

If the BMP180MI library is used, a temperature reading must be taken prior to taking a pressure reading, even if the temperature reading is discarded.

If the RF24 library is used, there is an in-house developed helper library(rf24_transeiver.h) that can be used alongside if desired.

1. Copy the rf24_transeiver.h into your project directory
2. Setup the radio transceiver using radioSetup(), returns 0 on success
3. To send a piece of text, use sendStringRadio(<text>, <length_of_text>), <text> must be in char, max length is 32 characters each call, <length_of_text> must match the actual length of the text, and must be int
4. Calling pollRadio() will return all the text the radio has received so far, in order to retrieve all the data continuously, and send it over the serial terminal, use

```
while(radio.available()) {

  Serial.print(pollRadio);

}
```

Power supply requirements:

It is highly recommended to use 2 separate batteries to power the product, with one being used to power the computer system (system battery), and the other being used to power the high current nichrome wire (wire battery). The system battery need to be around 5v +- 1v, and the wire battery need to be able to deliver around 3A of current continuously. Typically, a 1s lipo is sufficient for the system battery, and a 1s – 2s should be sufficient for the wire battery. Alternatively, a 9v battery can be used to provide power to the nichrome wire.

The nichrome wire needs to be at minimum 50mm long, and it is not recommended to turn on the wire for more than a few seconds. The wire can be turned on by applying a high signal to the microcontroller pin whose name is next to the port with in which the nichrome wire is connected.

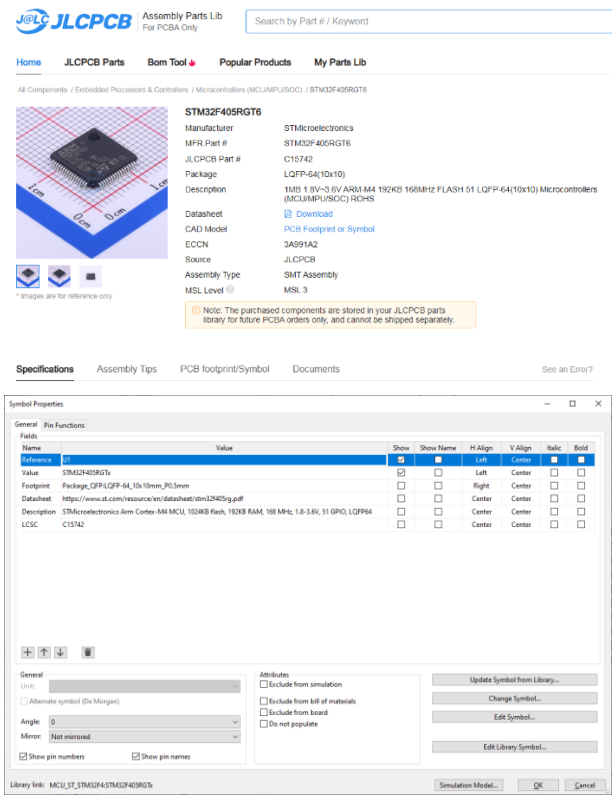
The system battery is labelled with +5v- under the screw terminals, and the +12v- is the wire battery. The positive wire should be connected to the terminal closest to the + sign, the negative wire should be connected to the terminal closest to the – sign.

For manufacturers, hardware developers / contributors, and advanced users

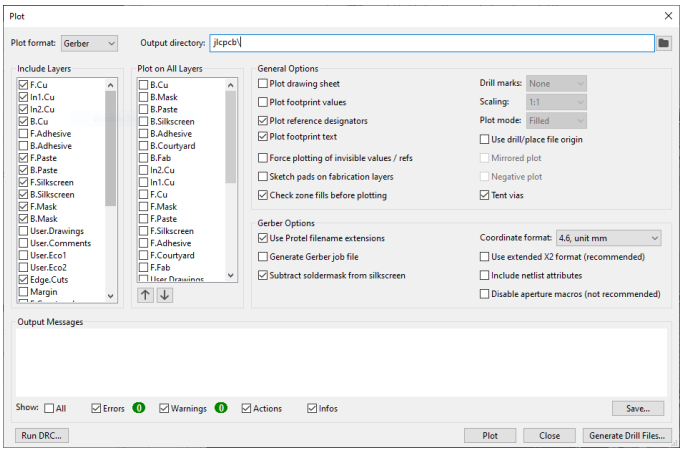
The product is exclusively designed in KICAD and already have JLCPCB part numbers pre-populated. JLCPCB seems to be a good manufacturer, they offers a high quality manufacturing service, while keeping the prices low in comparison to other manufactures. However, they are not the only manufacturer capable of producing these boards.

1. Adding of parts: To add additional parts, it is preferred if it can be assembled by JLCPCB SMT service. To inform JLCPCB on what part to place it is necessary to

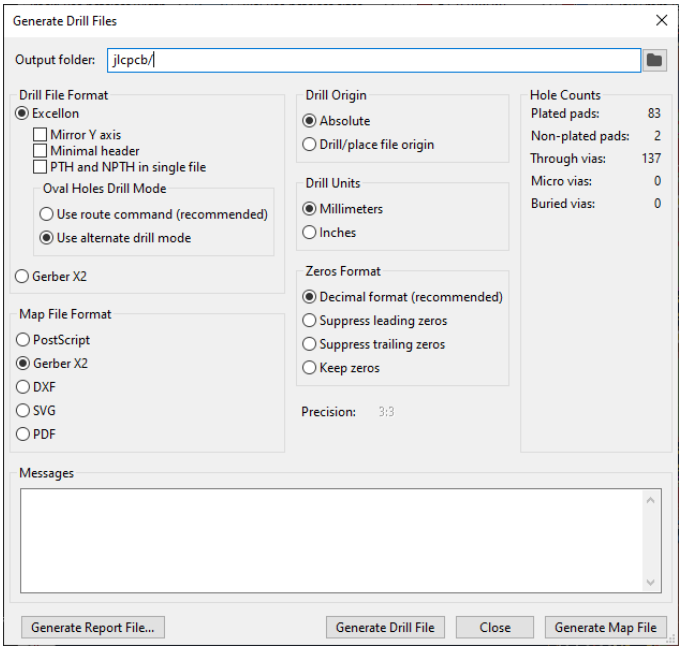
add another field called “LCSC” and input the JLCPCB part number, see images below for examples



2. Generating fabrication files: To have the boards manufactured by JLCPCB gerber files, BOM(bill of material), and placement file needed to be provided, please use the following instructions for generating these files. These instructions are based on JLCPCBs recommendation for KICAD 7 [2] [3], but has been updated for KICAD 8
    a. Gerber: PCB editor > File > Fabrication output > Gerber

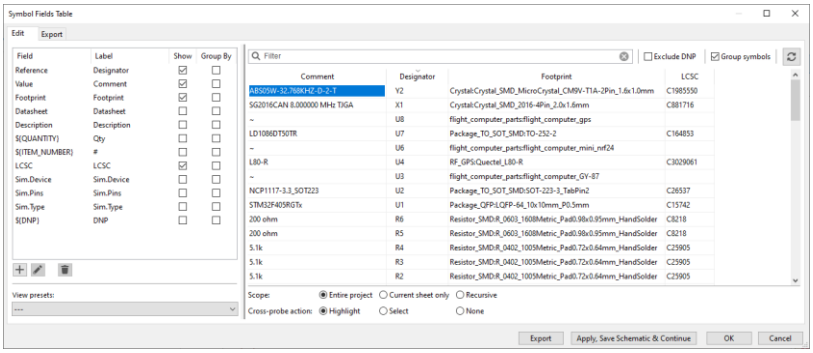


    b. Drill: PCB editor > File > Fabrication output > Drill

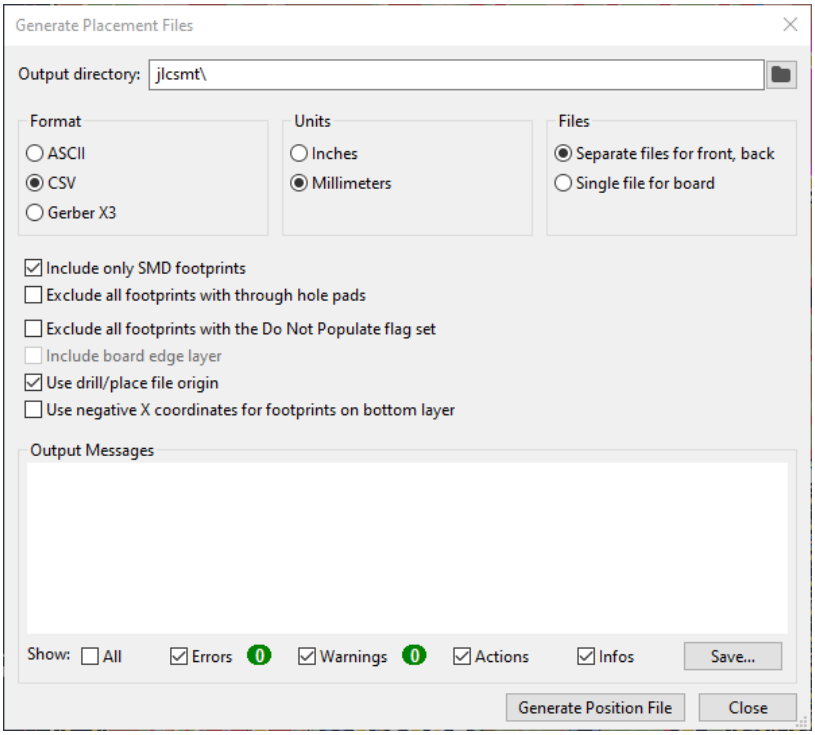


c. Zip all the outputs from the above steps into a .zip archive, store it for use later

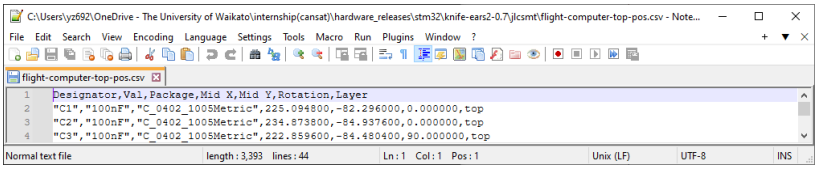
d. Generating BOM: Schematics Editor > Tools > Generate bill of materials



e. Generate component placement: PCB editor > File > Fabrication output > Component placement:



f. Open the component placement file, and change the first line to match the example below



g. By now you should have 1x zip file containing gerber and drill files, 1x BOM, 2x position .csv files, collect all these files and submit to JLCPCB for manufacturing

Alternatively use https://github.com/Bouni/kicad-jlcpcb-tools after you have finished the design of the board, with this extension, it allows you to check the parts are available at JLCPCB, and allows for one-click compilation without disruption to your workflow.

Sometimes, JLCPCB will fail to process the board placement and BOM files, in this case, open the file in LibreOffice or Microsoft Excel, and delete all empty rows.

References

[1] STMicroelectronics, “GitHub - stm32duino/Arduino_Core_STM32: STM32 core support for Arduino,” 23 1 2025. [Online]. Available: https://github.com/stm32duino/Arduino_Core_STM32. [Accessed 23 1 2025].

[2] JLCPCB, “How to generate Gerber and Drill files in KiCad 7,” JLCPCB, 2025. [Online]. Available: https://jlcpcb.com/help/article/how-to-generate-gerber-and-drill-files-in-kicad-7. [Accessed 13 2 2025].

[3] JLCPCB, “How to generate the BOM and Centroid file from KiCAD,” JLCPCB, 2025. [Online]. [Accessed 13 2 2025].