CanSat_2022 – Kit report

Made by: Balazs Farkas
Position: CanSat trainer (voluntary)
Contact: balazs_farkas@yahoo.com

This report is about the testing and evaluation of the McHobby Cansat kit provided for the Luxembourg Cansat Challenge for year 2022. The detailed instruction manual for the kit can be found here: https://wiki.mchobby.be/index.php?title=ENG-CANSAT-BELGIUM

Despite the easy-to-follow manual mentioned above, certain bugs and difficulties have been recognised during testing of the kit that are worth knowing about before assembly, programming and testing starts.

The following report should help all mentors navigate better with the kit and avoid some of the potential pitfalls. (**Note:** This technical report is merely to extend on what the manual offers and not to replace it.)

Fig 1: The kit provided for the 2022 CanSat Challenge

Acronyms

UNO	Arduino UNO Microcontroller
Feather	Adafruit Feather M0 Express Microcontroller
OBC	On-Board Computer
GS	Ground Station
TX	Transmitter of a (radio) signal
RX	Receiver of a (radio) signal
C	Programming language “C”
IDE	Arduino IDE software tool
I2C/SPI/UART	Various communication protocols
TTC	Telemetry, Tracking and Command
COM	Communication

General notes and tips

Use only 3.3 V

The Feather provides only 3.3V from its voltage output. Even though all sensors and the antennas can handle higher voltage (up to 5V), it is recommended to remain on 3.3V throughout the entire build. **Accidentally connecting the Feather to 5V could damage it!**

Solder well

There is a lot of soldering involved with the kit. Since it takes good skill to solder the pins properly to the board, it is highly recommended to practice before and become familiar with supporting tools and materials, such as soldering paste and the suction tool. Faulty soldering can lead to loss of function within the satellite or a completely chaotic behaviour in case of shorting.

Fig 2: Necessary extra tools for soldering

KISS (Keep It Simple, Stupid!)

It is necessary to have an Arduino UNO (or similar) to run the ground station segment of the experiment. At the same time, an UNO is a significantly simpler tool to learn compared to the Feather due to having less functions, less extras…and no need to solder!
Also, keep in mind that it is generally better to ensure each subsystem works properly separately before everything is integrated into one, unified design.

Programming tool

The Feather comes with the feature where it could use both CircuitPython as well as C to run a code. Considering the differences and the many-many supporting materials available on the internet, it is recommended to remain within the C programming language, relying on the Arduino IDE software tool to do the programming. (**Note:** In case the coding is done in CircuitPython, the proposed testing and evaluation of the CanSat may not be possible.)

Libraries

Libraries are freely available code segments that can be imported within the main code from your local Arduino code repository. All provided breakout boards (the BMP280 sensor and the RFM69 antenna)

come with a library that – when included in the local repository – can be called upon to make the functioning of the board that much easier, practically removing the need to understand the sensors below surface-level. These libraries also come with example codes that could be used to test the functions of the boards. Especially basic examples are highly recommended for testing purposes. Use the Arduino IDE's Tools->Library Manager to find libraries. (**Note 1:** Basic Arduino functions also often come in libraries. A good example is the library „SPI.h" which needs to be included in the main code to use SPI communications. **Note 2:** Copy-pasting an entire example code into the final integrated code usually carries over a lot of unnecessary elements. Use examples as guidance only!)

Dealing with direct inputs

The way how a microcontroller deals with any direct voltage put on a pin is by transforming the input into a number ranging from 0 to 1023, representing 0 to 5 V (or in case of the Feather 0 to 3.3V). This means that receiving a temperature of, say, 50 Celsius on the TMP36 temperature sensor would be first transformed into roughly 1 V of voltage on its output (as it is within its formula/working mechanism, see below), which then will be interpreted by the microcontroller as an „incoming" number of 204 (in case of the Feather, a 310). This extra calculation must only be applied on direct incoming voltages, with sensor breakout boards, the transformation is done by the sensors' libraries.

Variable types

Programming in C is very sensitive to consistency regarding variable types. If you have any issues with them, ask yourself if you are being consistent with the data type you are using. It may occur that a certain variable type must be used for a library, thus any data that is generated during the primary mission must first be transformed into that particular variable type before it could be processed by the library. (**Note:** SPI communication with the antennas uses character arrays to send data, while the data generated during the measurements will usually be in strings or floats.)

Communication protocols (UART, SPI, I2C)

The main SPI communication protocol pins are pre-set on a microcontroller, meaning that you cannot connect to them a device wherever you wish. Designated SPI pins can be found in the documentation of a microcontroller (the Arduino UNO's SPI pins are 11, 12 and 13, the Feather's are on 22, 23 and 24). If there are additional pins to connect (CS and RST), these pins are freely defined within the code of the master device. (**Note:** The Feather actually has two SPI connections, but the second one is reserved for the SPI Flash and thus is not freely available.)

Also, please make not that the "Wire.h" library must be included into the code to make I2C communication protocol possible.

Hardware notes

Power subsystems (POW)

LiPO battery

The kit comes with a Lithium battery. Charging of the battery happens through the Feather. How long the battery lasts will depend on the amount of current drawn from it through an extended period of time. Once the design is finalized, it is highly recommended to make an estimation over how long the satellite could function, by measuring the current (in mA) drawn from the battery and then dividing it with the battery capacity (800 mAh, 1400 mAh or 2500 mAh, depending on the kit).

Communications subsystem (TTC)

RFM69

Two identical, RFM69 breakout board transceivers (two-way radios) are supplied with the CanSat_2022 kit. They can be used as either transmitter (TX) or receivers (RX) without any penalty, making them ideal as telemetry, tracking and command (TTC) communication systems. In other words, the RFM69-s can be used either to send or receive data, thus a CanSat could either send it's state (telemetry) to the ground station, or receive some instructions (command). Tracking is not used here.

Regarding the RFM69, they are rather easy to make to work just by following the detailed instructions on the kit's official site on antennas (https://wiki.mchobby.be/index.php?title=ENG-CANSAT-RFM69HCW). For frequency plan, see

To point out some matters of interest:

- For the primary mission, you don't need to solder the pins G1-G5. The radios will be able to run without them.

- The antennas do not work without an antenna attached to the antenna port (the pin position without any designated name assigned to it). An antenna could be as simple as a wire (or for a very limited range (!) testing, a pin soldered into the hole). Mind that antenna should only be attached to the hole and nothing else, otherwise shorting could made the board function poorly or not at all. Also, be aware that antennas are very sensitive to connection quality.

- The RFM69 boards use SPI communication protocol to share their information with their „host" device, the so-called master. They do not function on their own, making a second microcontroller (Arduino UNO or similar) necessary to set up TTC.

- The libraries recommended by the kit provider function perfectly. As mentioned within the description though, it is necessary to set the communication frequency to 433 MHz (or whichever the RFM69 ends up using, see above) in the „#define RFM69_FREQ 433.0" line on both the TX and the RX codes. Without it, communication may seemingly occur between the hardware, albeit the software would not be able to interpret it. It is also necessary to update the TX side code so it may recognise the microcontroller it is running upon. (**Note:** For the

frequency plan, see https://wiki.mchobby.be/index.php?title=ENG-CANSAT-FREQUENCY-PLAN)

- Sending information goes step-by-step between the RFM69-s, thus after the message to be sent is defined, it is also necessary to define, how many letters one wishes to send from that message by providing the packet length. Failing to do so usually leads to empty messages being exchanged between the antenna boards.

- The RFM69 has four freely defined connections to a microcontroller, the CS „chip select" (used to send the data to the antenna), the RST „reset" (used to shut down/wake up the antenna), the INT (used to send the antenna some extra data towards the G1-G5 pins – only used for more complex interactions) and the LED (used for controlling the microcontroller's debugging LED). For pure function, one merely needs to use the CS pins, plus manually put the RST pin to ground (LOW).

Payload subsystem (P/L)

The payload of the primary mission consists of a temperature sensor (TMP36) and a breakout board (BMP280). This latter is capable to measure the temperature and the pressure, then use those inputs to calculate the absolute altitude of the sensor above sea level.

TMP36

The TMP36 temperature sensor is easy to install and made to work, relying on only simple programming and electronics skills. Like most sensors of its type, it measures temperature by changing the voltage on its output pin, making it necessary – through coding – to transform that data into actual temperature values. The manual provided for the kit gives detailed explanation over how that needs to be done (https://wiki.mchobby.be/index.php?title=ENG-CANSAT-TMP36). The formula of the TMP36 is

```
Temp in °C = ( output_voltage_in_mV - 500) / 10
```

It must be mentioned that sensors directly plugged into the breadboard may show significant uncertainty due to poor connection between the sensor's pins and the breadboard.

BMP280

The BMP280 sensor can detect temperature and pressure, and then calculate the altitude from those two (using the barometric formula). It is a breakout board, just like the RFM69. The board can be run both on SPI and I2C communication protocols. Considering that within the integrated model already two SPI protocols will be in use (one for the RFM69 antenna, one for the SPI Flash internal memory), it is recommended to use I2C to run the BMP280. The manual give a good summary on how to make the BMP280 work using either protocols (https://wiki.mchobby.be/index.php?title=ENG-CANSAT-

BMP280). (**Note:** Unlike the antenna, the BMP280 can be connected and tested without soldering, using an extra StemmaQT cable instead.)

SPI Flash

The Feather comes with a small Flash drive of 2 MB, that behaves like a small SD card. It is connected to the SPI pins 2, 3 and 4, with CS on pin 38, making these pins not freely available during the build. The manual (https://wiki.mchobby.be/index.php?title=ENG-CANSAT-FEATHER-M0-SPI) gives a good walkthrough over how to make datalogging work, albeit it does not touch upon very well over how to access the file system on the board, that is, how to put it into "CircuitPython mode".

- For "CircuitPyhon mode", start the Feather bootloader by pressing the reset button twice. This will open the bootloader as an external drive (FEATHERBOOT), into which you need to copy and paste the appropriate ".uf2" file, downloaded from the official CircuitPython website (here: https://circuitpython.org/board/feather_m0_express). (**Note:** Be very specific that the right uf2 file was downloaded, otherwise it won't work!) Once the file has been copied, the bootloader will disappear and a new external drive will pop up (likely called CIRCUITPY), this time with a file system available there. The Feather is now in "CircuitPython mode" and your measurement data files should be visible on the new external drive.

- Unplugging and re-plugging the Feather after it was put into "CircuitPython mode" will not reset it back to its original "Arduino mode". Pushing the reset button any number of times will not do it either. In order to remove it from "CircuitPython mode", one has to force-upload a code to whichever port the Arduino IDE has managed to recognize the Feather on.

-If nothing works, download the bootloader update (https://learn.adafruit.com/adafruit-feather-m0-express-designed-for-circuit-python-circuitpython/uf2-bootloader-details) and drop it on the "FEATHERBOOT" drive to completely reset the Feather.

- The Feather will forget any previous Arduino code when put into "CircuitPython mode".

- It is always recommended to format the SPI drive before use to ensure that it will work fine. **Warning: This will permanently wipe the SPI Flash, so remove your valuable data beforehand!**

- To make the library examples recommended within the Feather manual work properly, the SdFat.h library also needs to be downloaded. Without it, coding will encounter compilation issues. (**Note:** The SDFat library is there to deal with the file system on the SPI Flash.)

On-Board Computer (OBC) and Ground station (GS) subsystems

As mentioned within the antenna section above, two – otherwise interchangeable - microcontrollers are necessary to run the full experiment: one for the OBC (M0 Feather provided in the CanSat kit) and one for the GS (provided on request). The difference between the OBC and GS comes from the coding of the devices: while the GS will only need to run the RX's code for the antenna for basic functionality, the OBC will run an integrated code (TX, BMP, TMP36 and SPI flash, all in one). If you consider to switch

the OBC and the GS devices, please keep in mind that the OBC will need to fit into the frame of a CanSat, which is easy with the Feather but might be complicated with anything else.

Arduino UNO (or similar) for GS

The recommended microcontroller to run the GS is an Arduno UNO. It is the simplest microcontroller available on the market, yet more than capable to run the GS or even the OBC. It is plug-and-play, easy to connect to a breadboard and can even deal with diodes pushed into its pins slots directly.

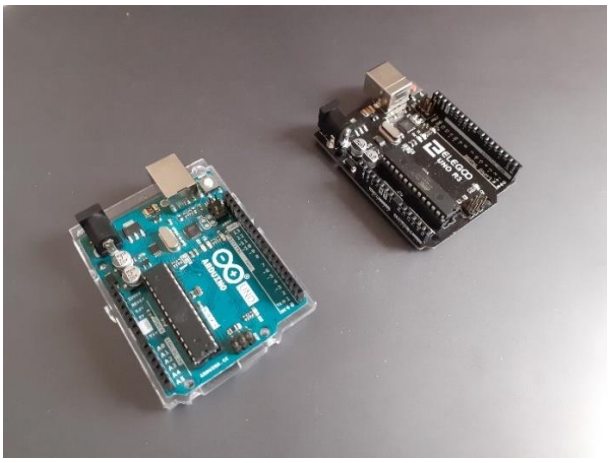

Fig 3: Arduino UNO (or similar) for GS

There are multiple training and know-how videos available on the internet as well as books. Recommended book is the official Arduino Projects Book by Scott Fitzgerald (pdf available on request). Generally, the UNO microcontroller – and the related experiments – are a very good way to introduce programming and electronics. Programming is done using only the C language.

Feather M0 Express for OBC

The specialized microcontroller from Adafruit provided for the CanSat kit. Unlike the UNO, it can run C codes as well as CircuitPython, the microcontroller-optimized version of the Python language. Apart from being noticeable smaller, it has multiple extra functions compared to its UNO counterpart, albeit they aren't that relevant for the primary mission of the CanSat.

Nevertheless, there are multiple points to address with this controller:

- All pins must be soldered into position, otherwise it could not be connected to a breadboard (unlike the UNO). Without soldering, only the BMP280 can be tested from the kit, albeit even that one would need an extra purchase for a StemmaQT cable first since the cable not provided in the kit.

- The reset button works differently than on other Arduino-type microcontrollers. While with other controllers, pushing the reset button once restarts the board completely, on the Feather, it does not restart the bootloader (the pre-code that is run before the setup), only the rest of the functions. In order to actually restart the bootloader, the reset button must be pushed

twice quickly. The ensuing outcome is different too: while other boards do not remain stuck in the bootloader after a reset, the Feather does, and must be „removed" from this state by pushing the reset button a third time.

- Installing all the necessary drivers and libraries could take up 2 Gb of space on the disk where the Arduino IDE is installed.

- The PC might recognize the internal SPI Flash memory as a separate device with a separate COM port. It can also recognise the Feather stuck in the bootloader (!) as a separate COM port (!!!) as well. When communicating with the Feather itself, be sure that the right COM port (COM Adafruit Feather M0 Express) is chosen and the Feather is not stuck in the bootloader (the small LED is not blinking red).

- The installation of the board is rather complex and may result in an outcome where the board refuses to be recognised on a COM port. In case this happens, it is recommended to remove the entire installation and redo everything from scratch. (**Note:** For me, installing the drivers for the board first, and then the SAMD board elements was what made the Feather work properly.)

- In case the board is on, the COM port is recognized and all the board-specific updates are done...but there still doesn't seem to be any activity from the Feather, just upload a code to whichever COM port it is currently recognized on. If that doesn't work, push the reset button twice and then upload. In general, a combination of reset button-pushing and uploading will make the Feather sort itself out, eventually.

- You might need to change the COM port in the Arduino IDE when between restarts the Feather forgets which COM ports to be on.

- After connecting the board to the PC, it sometimes shows no activity at all. It must be unplugged and reconnected.

- In general, once the first upload is done, the board starts to work as intended.

- A known bug with the Feather is that it sometimes pops up as „ FEATHERBOOT" connected disk drive using Windows. This message can be ignored if not aiming for the "CircuitPython mode".

(Note 1: do not forget to adjust the Arduino IDE's sketch compiler within board manager to the board you are using. From this point, the Feather is not compatible with the UNO. **Note 2:** for a more detailed manual on the Feather, see https://learn.adafruit.com/adafruit-feather-m0-express-designed-for-circuit-python-circuitpython)

Fully integrated breadboard model

The following photo shows the integrated CanSat system, ready to be soldered onto a prototyping board. The model runs the primary mission only.

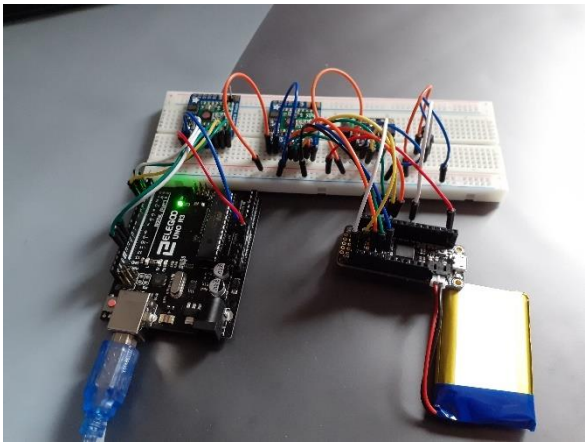

Fig 4: This is what you should have in the end

As mentioned above, the Feather is the OBC, while the GS is an Elegoo UNO R3 (an Arduino UNO similar). The antennas are just two pins, soldered on the RFM69-s. The OBC runs on the battery and is fully autonomous. The GS is connected to a PC. This very breadboard model is available for inspection during in-person presentations held jointly by Esero and the University of Luxembourg.

Have fun!