betaLogger - Project Presentation

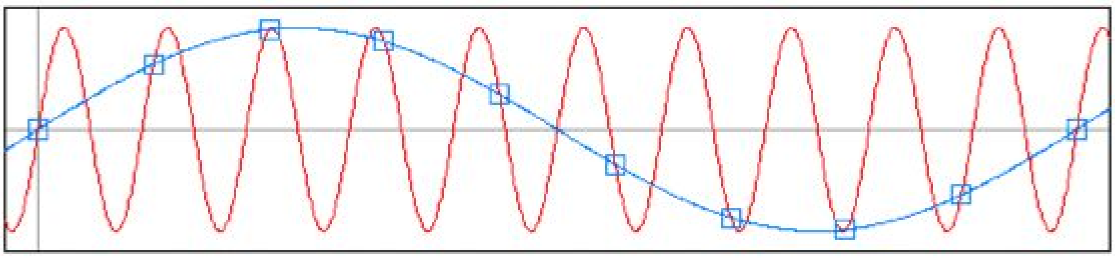
Introduction

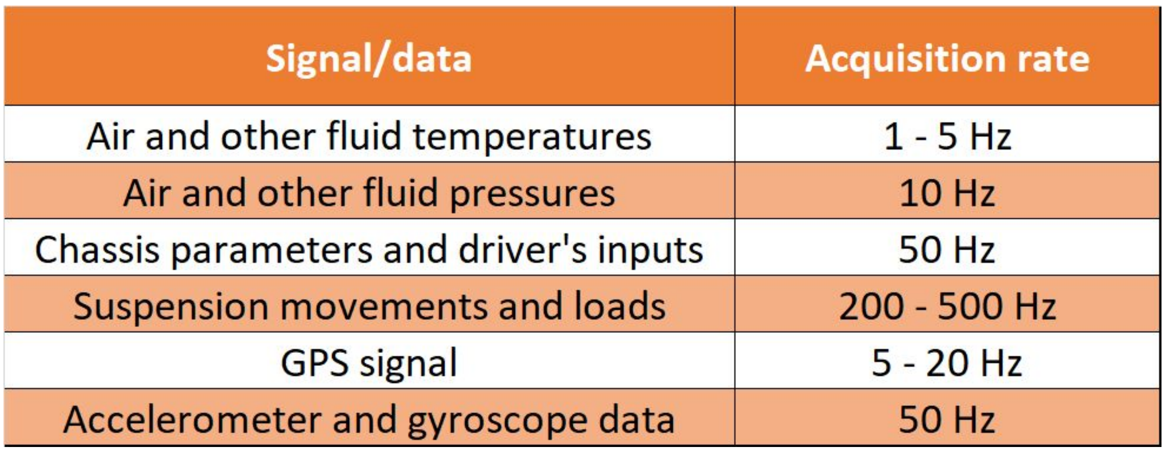
This project, nicknamed ***betaLogger***, was developed by ***Patrick Metzner Morais*** in 2019 in order to log automotive data. The goal of the project was to get relevant data from the car developed by Fórmula UFSC, during test sessions and competitions.

This project was developed using an ***STM32 microcontroller***. It is capable of logging the following data:

- 7 analog channels [ADC] (200 Hz);
- 3 axis Accelerometer 3 axis Gyroscope [I2C] (200 Hz);
- GPS – Latitude, longitude and speed [USART] (10 Hz);
- Data logged into an SD Card [SPI] (10 Hz);
- CAN bus will be implemented in the future.

These logging rates were chosen according to ***Jorge Segers*** book, ***Analysis Techniques for Racecar Data Acquisition***. According to ***Segers***, in order to ***avoid aliasing*** (figure below), an effect that causes different signals to become indistinguishable (or aliases of one another) when sampled, the data acquisition must be made at rates higher than the double of the highest frequencies present in each signal. ***Segers*** suggests that the minimum data acquisition rates must follow the table below, adapted from his book.





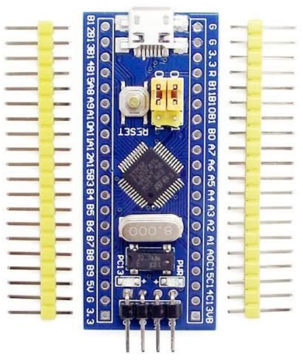
Signal/data	Acquisition rate
Air and other fluid temperatures	1 - 5 Hz
Air and other fluid pressures	10 Hz
Chassis parameters and driver's inputs	50 Hz
Suspension movements and loads	200 - 500 Hz
GPS signal	5 - 20 Hz
Accelerometer and gyroscope data	50 Hz

About The Project

Hardware

The ***microcontroller*** chosen for this project was the ***STM32F103C8Tx*** (image below). Some of its features can be found in the list below:

- 72 Mhz clock frequency;
- 128k bytes of Flash memory;
- Peripherals supported:
    - timers (16-bit);
    - ADC (12-bit);
    - SPI (18 Mbit/s);
    - I2C (400kHz clock);
    - USART (4.5 Mbit/s).

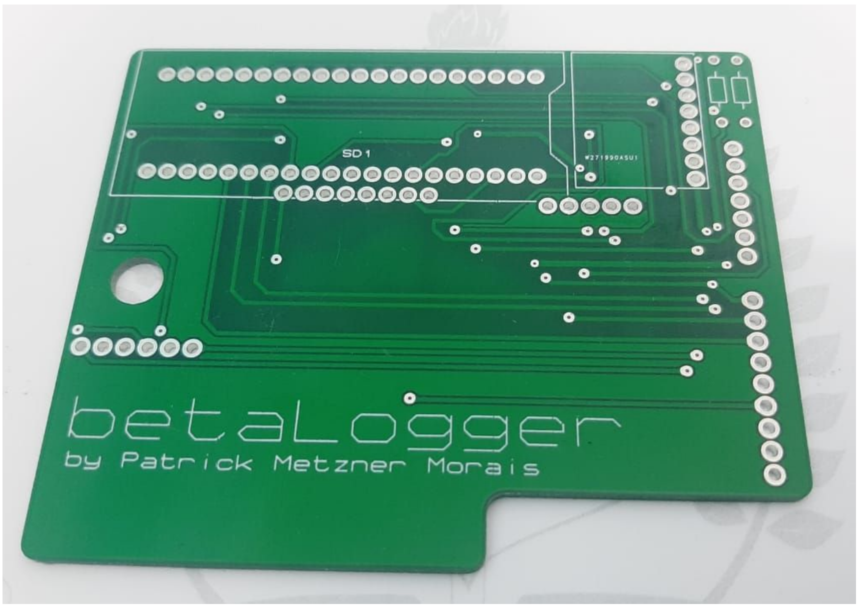


The Motion Processing Unit chosen was the ***MPU6050*** (image above). This module can register 16-bit ***accelerometer*** and ***gyroscope*** data. The data can be sent to other devices by I2C or SPI and can be acquired in four different ranges:

- Accelerometer - ±2g, ±4g, ±8g, and ±16g;
- Gyroscope - ±250, ±500, ±1000, and ±2000°/sec (dps).

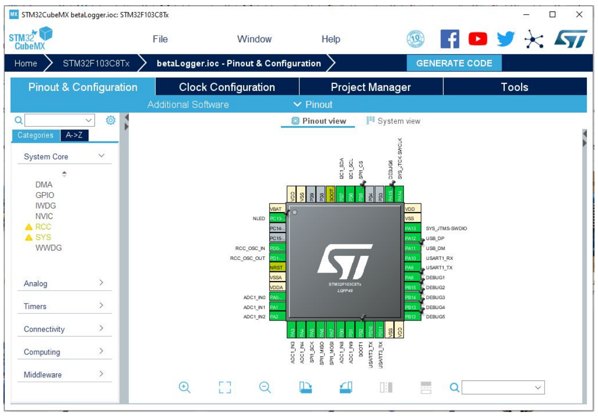
The ***GPS*** chosen for the project was the **U-Blox NEO-6M** (image below). It can transmit data via USART or SPI at 1 ou 10Hz.

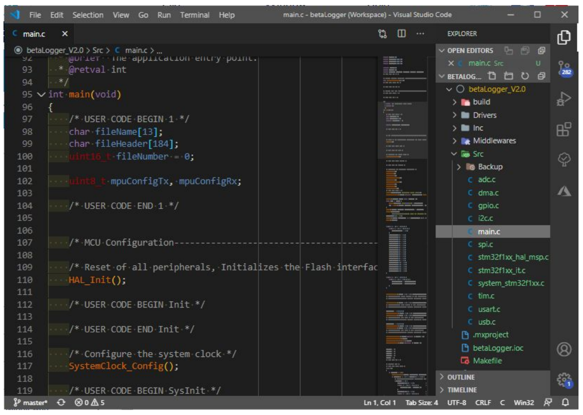
All these modules are soldered on a custom PCB and are packaged in small plastic case along with a 2200mAh battery, capable of supplying enough charge for the system to run continuously for several hours without the need of external power. The PCB and the case can be seen in the pictures below.



Firmware

The code for this project was developed using ***STM32CubeMX*** and ***Visual Studio Code*** (images below). STM32CubeMX was used to configure all of the microcontroller peripherals, while VS Code was used to develop the rest of the code.

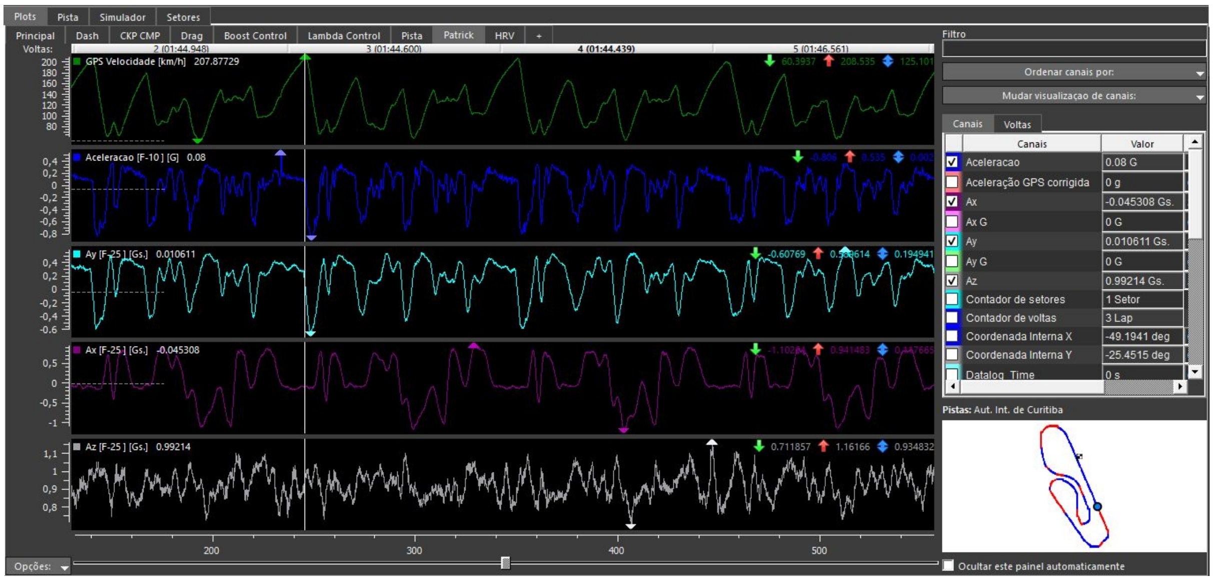




```
  * @retval int
  */
int main(void)
{
    /* USER CODE BEGIN 1 */
    char fileName[13];
    char fileHeader[184];
    uint16_t fileNumber = 0;

    uint8_t mpuConfigTx, mpuConfigRx;

    /* USER CODE END 1 */


    /* MCU Configuration--------------------------------------------------------

    /* Reset of all peripherals, Initializes the Flash interfac
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */
```

In short, the code found inside ***betaLogger_V2***.0 folder will configure the ***STM32F103C8Tx*** to make 7 analog to digital conversions and request 14 bytes of I2C data every 5 milliseconds, as well as receive 51 bytes via USART every 100 milliseconds. All this data will be stored in its memory until a buffer is full and ready to be sent to an SD card via SPI.

The data is saved in the SD card as a ***DLF file***, similar to a .csv file, in order to be compatible with ***ProTuneAnalyzer***. This program allows the user to view the logged data as seen in the image below.



	Canais	Valor
✓	Aceleracao	0.08 G
	Aceleração GPS corrigida	0 g
✓	Ax	-0.045308 Gs.
	Ax G	0 G
✓	Ay	0.010611 Gs.
	Ay G	0 G
✓	Az	0.99214 Gs.
	Contador de setores	1 Setor
	Contador de voltas	3 Lap
	Coordenada Interna X	-49.1941 deg
	Coordenada Interna Y	-25.4515 deg
	Datalog_Time	0 s

Tests done to ensure the system works

- Tests to ensure the correct measurement of time:
    - Generate a square wave using the interruptions of the microcontroller timer and analyze it with the oscilloscope (short term);
    - Save the GPS time stamp in a file and compare it to a counter also saved into the same file (long term).
- Ensure the correct sampling rate of the system:
    - Use the analog channels to log a known signal, generated by a calibrated instrument, and analyze the data in MATLAB.
- Take the system to track day events and compare the lap time data recorded by betaLogger with the events' oficial lap times.

Areas where this project can be improved

- Separate the code into functions to make it easier to read, maintain and expand;
- Substitute all bidimensional arrays for single dimensional arrays;
    - This will ensure the data to be stored in sequence and will improve the speed of the code, possibly eliminating the need to have 3 different ***sdBuffers***;
    - It is also possible to use a single circular buffer. Tests should be made to choose an appropriate size.
- The data is stored as a DLF file for practicality reasons. It is possible to store the raw data in a DAT file and use other programs to analyze the data;
- The documentation of this project is insufficient and can be improved;
- The tests can be automated in order to make it easier and safer to implement new features;
- The file can be named according to the GPS date and time when it is available.