Hes·so VALAIS WALLIS

School of Engineering

Hes·so

Haute Ecole Spécialisée de Suisse occidentale

Fachhochschule Westschweiz

University of Applied Sciences and Arts Western Switzerland

Degree Programme

Systems Engineering

Major Infotronics

BACHELOR'S THESIS

DIPLOMA 2023

Sylvestre van Kappel

Telemetry for the Formula Student

Professor
Prof. Medard Rieder

Expert
Yoan Rossier

Submission date of the report
18.08.2023

Données du travail de diplôme
Aufgabenstellung der Bachelorarbeit

Filière / Studiengang **SYND**	Année académique / *Studienjahr* **2022-23**	No TB / *Nr. BA* **IT/2023/91**
Mandant / *Auftraggeber* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*	Etudiant / *Student* **Sylvestre van Kappel** Professeur / *Dozent* **Medard Rieder**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja ☒ non / nein	Expert / *Experte* *(données complètes)* Yoan Rossier, Route de Vissigen 68, 1950 Sion yoan.rossier@insiders.live, +4178 6719441	

Titre / *Tite* **Telemetry for the Formula Student**

L'objectif de ce travail est de développer et de tester un système de télémétrie pour la voiture de course Formula Students de la HES-SO. Les tâches suivantes doivent être accomplies :

- Définition des données à transmettre. Dans ce contexte, il faut également choisir les capteurs correspondants. Il faut également déterminer la quantité de données qui en résultera.
- Choisir une technologie de transmission sans fil appropriée.
- Sélectionner et définir le backend du système de télémétrie.
- Création du schéma et de la mise en page de l'appareil de télémétrie sur le véhicule.
- Programmation du dispositif de télémétrie.
- Mise en place du backend.
- Test du système de télémétrie en termes de portée et de fiabilité.

Les éléments suivants doivent être remis :

- Rapport technique
- Tous les codes développés
- Présentation du système
- Prototype du système de télémétrie.

Signature ou visa / *Unterschrift oder Visum*

Responsable de l'orientation /
Leiter der Vertiefungsrichtung:

.....

[1] Etudiant / Student:

.....

Délais / *Termine*

Attribution du thème / *Ausgabe des Auftrags:*
15.05.2023

Présentation intermédiaire / *Zwischenpräsentation*:
19 - 20.06.2023

Remise du rapport final / *Abgabe des Schlussberichts:*
18.08.2023, 12:00

Expositions / *Ausstellungen der Diplomarbeiten:*
25.08.2023 – HEI
28.08.2023 – Monthey
31.08.2023 – Visp

Défense orale / *Mündliche Verfechtung:*
Semaine/Woche 36 (04-07.09.2023)

[1] *Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme. Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.*

Rapport reçu le / *Schlussbericht erhalten am* ……..…….. Visa du secrétariat / *Visum des Sekretariats* ……..

Hes·so VALAIS WALLIS

Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Telemetry for the Formula Student

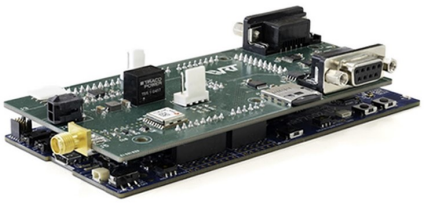
Graduate Sylvestre van Kappel

Objectives

This project aims to develop and test a telemetry system for the Formula Student car of the HES-SO Valais-Wallis. This thesis deals with the telemetry system's onboard device and the communication with the PC.

Methods | Experiences | Results

Telemetry is a technology that enables remote measurement and monitoring. This technology is interesting for a race vehicle as it allows live readings from the car's sensors to be monitored directly from the side of the track. With such a system, the data from all the sensors are easily accessible, and the engineers can adjust the car's parameters during the test sessions to increase the car's performance.
The objective of this project is to select the optimum transmission technology and to design and test the hard- and software of the telemetry system's onboard device.
The telemetry device is connected to the sensors on the car's CAN bus. The device also includes a GPS module. The CAN bus and GPS module data are transmitted to the base station via Wi-Fi. The data are also saved on an SD card. The system is designed to be generic and completely configurable. New sensors can easily be added to the system by only changing the configuration file.
The device has been successfully developed and tested, and all the objectives have been reached. The tests showed that the system can transmit data over a 750 meters distance.



Telemetry onboard device

Valais-Wallis Racing Team's Car

Bachelor's Thesis
| 2023 |

Degree programme
Systems Engineering

Field of application
Infotronics

Supervising professor
Medard Rieder
medard.rieder@hes-so.ch

Partner
Valais-Wallis Racing Team

Information about this report

Contact information

Author:	Sylvestre van Kappel Bachelor Student HES-SO//Valais Wallis Switzerland
Email:	sylvestrevk@gmail.com

Declaration of honor

I, undersigned, Sylvestre van Kappel, hereby declare that the work submitted is the result of a personal work. I certify that I have not resorted to plagiarism or other forms of fraud. All sources of information used and the author quotes were clearly mentioned.

Place, date: August 16, 2023

Signature: 

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

1.1 Formula Student

Formula Student is an international student engineering competition. Teams from all over the world design and build a small-scale Formula-style racing car. During the competitions, the vehicles are judged on static events: Engineering Design, Cost and Manufacturing, Business Presentation, Lap Time Simulation, and Technical Inspection. The vehicle must comply with the rules [1] for the technical inspection. If the car passes the technical assessment, it can participate in the dynamic events: Skidpad, 1 km autocross/sprint, 75 m acceleration, and 22 km endurance.

1.2 Valais-Wallis Racing Team

The Valais-Wallis Racing team [2] is the Formula Student team of the HES-SO Valais Wallis. It was created in the spring of 2022 by a group of students. The team's first car will take part in the races of summer 2023, and the telemetry system developed in this thesis will be used on the next car for the races of summer 2024. Telemetry is not implemented on the first car. This project deals with the development of the team's first telemetry system.

Figure 1.1: Valais-Wallis Racing Team's Car [3]

1.3 Objectives

Telemetry is a technology that enables remote measurement and monitoring.This technology is interesting for a race vehicle as it allows live readings from the car's sensors to be monitored directly from the side of the track. With such a system, the data from all the sensors are easily accessible, and the engineers can adjust the car's parameters during the test sessions to increase the car's performance. A telemetry system is also helpful in improving the driver's skills, providing measurements such as GPS, speed, pedal level, steering angle, etc. Direct visualization of measurements also allows problems to be identified before they can cause an accident or damage the car.

This project aims to develop and test a telemetry system for the Formula Student car of the HES-SO Valais-Wallis. This thesis deals with the embedded part of the system and the communication with the remote PC. This project will be carried out in collaboration with a Business Information Technology student working on the software to display the telemetry system's data.

For this work, the reference race is Formula Student Alpe Adria [4] in Croatia. On this circuit, the maximum distance between the car and the base station will be less than 300 meters. The minimum range of the system must therefore be 300 meters, ideally 500 meters.

Figure 1.2: Formula Student Alpe Adria Circuit

1.4 State of the Art

Telemetry is used in many systems that require real-time remote measurement and monitoring. The telemetry needs to be embedded in a vehicle for this application. Racing cars often incorporate telemetry systems, but these are proprietary systems unavailable on the market. That's why the Valais-Wallis Racing Team must develop its own telemetry system.

Sensors are the fundamental elements of a telemetry system. All sensors must be connected to the telemetry system. The CAN bus is the standard protocol used in the automotive industry for this purpose. CAN bus is a communication protocol that enables reliable and efficient data transfer between ECUs and sensors. It is a multiplexed bus and was introduced to reduce the amount of copper in cars by wiring all the ECUs and sensors on the same bus. It has been standardized with ISO 118987.

The transmission process is the other key point of the telemetry system. Many transmission processes could be used for this purpose.

- A standard 433 MHz RF transmission can send data through the air at a reasonable range. Many wireless devices use this transmission method with a dedicated protocol.
- Otherwise, there are many transmission protocols used for remote data transmission. For example, Wi-Fi is one of the most common transmission protocols. Every laptop, smartphone, etc., use it.
- LoRaWAN is a protocol used with LoRa technology that enables long-range communication. Many IoT applications use this protocol to communicate.
- 4G/5G cellular networks can transmit data with an unlimited range as long as the car is in network coverage.

The last component of a Telemetry system is the processing unit. The processing unit on almost every telemetry system is a PC with dedicated software. The processing unit displays the sensors' measurements and allows the user to analyze the data transmitted by the telemetry system. The processing units can also be linked with a database to store the data at the end of every test session or race.

1.5 Structure of this Report

This report is divided into five parts. The first is an analysis to select the transmission technology. The second part is a general system overview. The third part describes the hardware development, and the fourth part describes the software. The final part details the various tests that have been carried out.

1.6 Git

All the files referring to the project (codes, altium PCB project ...) are available on a Git on the following link :

blablabla

2 | Transmission Technology

A wireless transmission must be established from the car to the computer. The range of the telemetry system must be greater than 300 meters (ideally 500 meters) with no obstacle between the transmitter and the receiver.

For this transmission, two technologies were studied. A Wi-Fi connection and a 433 MHz RF transmission.

LoRa protocol and 4G/5G cellular networks were not considered, as the bandwidth is not enough for the first, and there is no guarantee of network coverage in the race locations for the second.

2.1 RF 433 MHz

The first option is a standard RF transmission. This solution is easy to implement, the range is wide enough, and the bandwidth is sufficient for the small amount of data to be sent. Moreover, there is no need to establish a connection. The transmitter emits the data on a frequency, and the receiver listens to the same frequency. If the transmission is interrupted, the data transmission will instantly work once the signal is recovered.

The 433 MHz band is the best compromise between the range and the bandwidth. Higher frequencies will have better bandwidth, but the range will be shorter. Lower frequencies will have a better range, but the bandwidth will not be able to carry enough data.

However, there will be many teams during the competition, and if other teams use the same transmission method, all systems will disturb each other, and the transmission will not be guaranteed.

For an RF transmission, the RFM69HCW [5] module from HopeRF is one of the best options for this use case. It is available in 433 MHz and 868 MHz.

Figure 2.1: RFM69HCW module

This module has an output power of +20 dBm and an input sensitivity of -120 dBm at 1.2 kbps. The theoretical range can be calculated with the Friis transmission equation [6] (with omnidirectional antennas → Gain of antenna = 0 dbi):

$$P_r = P_t + 20 \cdot \log\left(\frac{\lambda}{4\pi d}\right)$$

For the 433 MHz module, the received power is -54 dBm at 500m, which is more than enough for the input sensitivity of -120 dBm.

2.2 Wi-Fi

The second option is a Wi-Fi connection. It is a robust protocol that allows the transfer of large amounts of data. With a Wi-Fi connection, a classic router can connect the telemetry system to the computer and directly send IP packets. Moreover, the system will not have to manage interference from other teams because the Wi-Fi protocol uses carrier-sense multiple access with collision avoidance (CSMA/CA). This mechanism is based on the LBT (Listen Before Talk) principle. When a station needs to send something, it will wait for nobody else to emit on the same channel. This permits having many Wi-Fi networks in the same area. Rather than disturb each other, the different stations will lower their transmission speed.

Disadvantages of Wi-Fi communication are:

- The range, which will be more challenging to reach than with a 433 MHz transmission.
- In case of loss of connection, the car will need to reconnect to the hotspot, which takes a little time.

Two Wi-Fi module were studied. The Digi XBee Wi-Fi and the Nordic nRF7002.

The XBee Wi-Fi [7] is a module working with the 802.11b/g/n [8] standards and supporting WPA and WPA2 . It has SPI and UART interfaces to relate to a host SoC. It works on the 2.4 GHz band, and its maximum output power is 16 dBm (13 dBm for the EU versions). At 1 Mbps, the input sensitivity is the best and reaches -93 dBm.

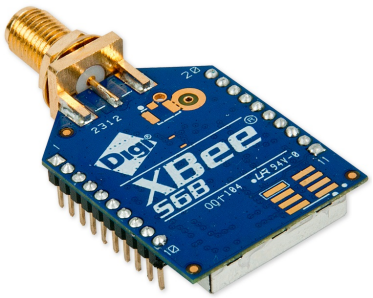


Figure 2.2: Digi XBee Wi-Fi Module

The nRF7002 is a chip from Nordic Semiconductors [9] working with the 802.11a/b/g/n/ac standards and supporting WPA, WPA2, and WPA3. It is a dual-band (2.4 GHz and 5GHz) Wi-Fi module with a QSPI interface to relate to a host SoC. Its maximum output power is 21 dBm, and the input sensitivity is -98.6 dBm at 1 Mbps.

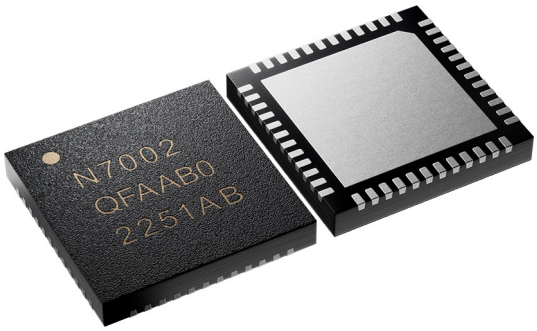


Figure 2.3: nRF7002 chip

Based on the specifications, nRF7002 is clearly better and more modern on all points. It supports more IEEE 802.11 (Wi-Fi) standards and security protocols, supports dual-band, and consumes less current. Its output power and input sensitivity are better, which will result in a better range. The QSPI interface enables up to 61 Mbps data transfer, compared with 6 Mbps for Xbee's SPI interface. Moreover, the nRF7002 offers a strong compatibility with the Nordic SoC, which could be an excellent choice for this project.

The theoretical range of the nRF7002 module can be calculated with the Friis transmission equation [6] (with omnidirectional antennas → Gain of antenna = 0 dbi):

$$P_r = P_t + 20 \cdot \log\left(\frac{\lambda}{4\pi d}\right)$$

For the 2.4 GHz band, the received power is -73 dBm at 500 m, which should be enough for a traditional router. As these equations give theoretical results, tests must be carried out to determine whether the range will be sufficient in practice.

2.2.1 Tests

This test aims to determine if a Wi-Fi connection is suitable for the required range. The tests have been carried out in the 2.4 GHz frequency band because it offers the best range.

The router used in this test is an Asus RT-AC68U. The router's technical specifications do not mention the output power, but it does have the CE mark, which means that the output power should not exceed 20 dBm. On the opposite side, a laptop was used. It is equipped with an Intel Wi-Fi 6 AX201 module [10], which has similar performances (in terms of range) as the nRF7002. The test was realized with the default omnidirectional antennas, so there is room for improvement using sector antennas if needed.

The tests were performed with the iPerf software [11], using the UDP protocol at 1 Mbps.

The laptop and the router communicated successfully at 400 meters distance. The minimal objective of 300 meters is reached. Should a better range be necessary, the performances can be improved by using sector antennas.

```
C:\Users\sylve\OneDrive\Bureau\iperf-3.1.3-win64>iperf3.exe -c 192.168.2.35 -u -b1m -i1 -p 6000
Connecting to host 192.168.2.35, port 6000
[  4] local 192.168.2.35 port 52345 connected to 192.168.2.35 port 6000
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.01   sec   120 KBytes   970 Kbits/sec  15
[  4]   1.01-2.00   sec   128 KBytes  1.06 Mbits/sec  16
[  4]   2.00-3.01   sec   120 KBytes   976 Kbits/sec  15
[  4]   3.01-4.00   sec   120 KBytes   989 Kbits/sec  15
[  4]   4.00-5.01   sec   120 KBytes   975 Kbits/sec  15
[  4]   5.01-6.00   sec   128 KBytes  1.06 Mbits/sec  16
[  4]   6.00-7.01   sec   120 KBytes   975 Kbits/sec  15
[  4]   7.01-8.01   sec   120 KBytes   989 Kbits/sec  15
[  4]   8.01-9.01   sec   120 KBytes   975 Kbits/sec  15
[  4]   9.01-10.01  sec   128 KBytes  1.06 Mbits/sec  16
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.01  sec  1.20 MBytes  1.00 Mbits/sec  0.156 ms  0/152 (0%)
[  4] Sent 152 datagrams
```

Figure 2.4: iPerf test - Client side

```
-----------------------------------------------------------
Server listening on 6000
-----------------------------------------------------------
Accepted connection from 192.168.2.35, port 51901
[  5] local 192.168.2.35 port 6000 connected to 192.168.2.35 port 52345
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.01   sec   112 KBytes   904 Kbits/sec  0.057 ms  0/14 (0%)
[  5]   1.01-2.00   sec   128 KBytes  1.06 Mbits/sec  0.105 ms  0/16 (0%)
[  5]   2.00-3.01   sec   120 KBytes   976 Kbits/sec  0.152 ms  0/15 (0%)
[  5]   3.01-4.01   sec   120 KBytes   989 Kbits/sec  0.201 ms  0/15 (0%)
[  5]   4.01-5.01   sec   120 KBytes   976 Kbits/sec  0.208 ms  0/15 (0%)
[  5]   5.01-6.01   sec   128 KBytes  1.06 Mbits/sec  0.193 ms  0/16 (0%)
[  5]   6.01-7.01   sec   120 KBytes   975 Kbits/sec  0.220 ms  0/15 (0%)
[  5]   7.01-8.01   sec   120 KBytes   989 Kbits/sec  0.233 ms  0/15 (0%)
[  5]   8.01-9.00   sec   120 KBytes   990 Kbits/sec  0.222 ms  0/15 (0%)
[  5]   9.00-10.01  sec   128 KBytes  1.04 Mbits/sec  0.156 ms  0/16 (0%)
[  5]  10.01-10.01  sec  0.00 Bytes  0.00 bits/sec  0.156 ms  0/0 (0%)
```

Figure 2.5: iPerf test - Server side

The same test was also successfully carried out on a smartphone at the same distance.

2.3 Conclusion

The following table shows the differences between a 433 MHz RF transmission and a Wi-Fi communication:

	Wi-Fi	**RF 433 MHz**
Bandwidth	More than enough	Just enough
Range	enough	More than enough
Disturbed by interference	No (managed by CSMA/CA)	Yes
Connection to the computer	Wi-Fi / Ethernet	Serial port
Base station module	Wi-Fi Router	To be designed
Protocol	TCP/UDP	To be designed

Table 2.1: Comparison between Wi-Fi and RF 433 MHz

The critical issues are the range and the ability to operate in an environment with much interference.

The Wi-Fi solution is clearly better regarding its ability to deal with interference. The 433 MHz RF solution should be avoided because there will be many teams during the event, and communication without collision avoidance will likely fail.

For the range, the 433 MHz RF solution is better. However, the tests showed that the minimum required range was reachable with a classic Wi-Fi router.

The Wi-Fi solution is the best option based on these two main criteria. Moreover, on the other criteria, the Wi-Fi solution has advantages.

The system will therefore use Wi-Fi communication with the Nordic Semiconductors nRF7002 module [12].

3 System Overview

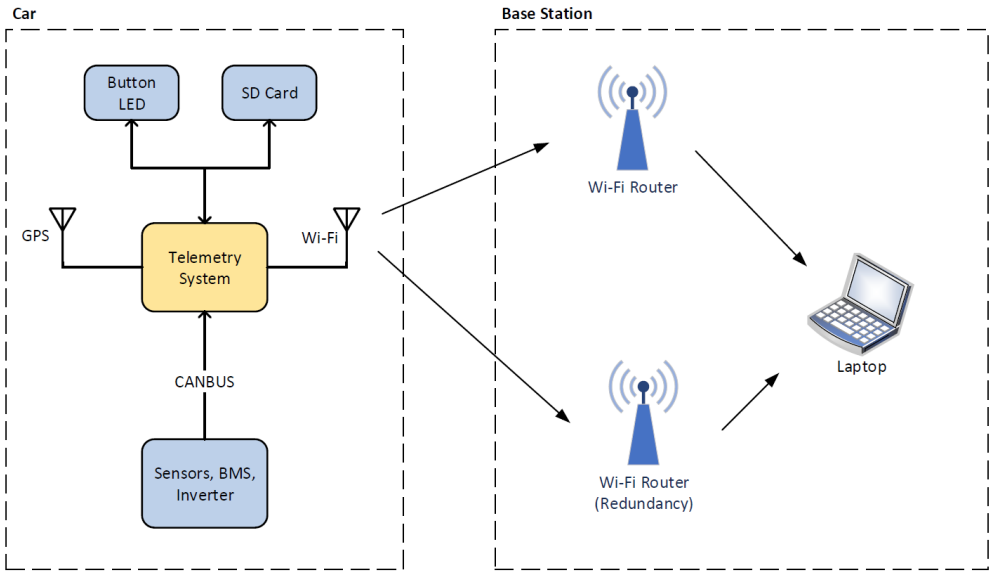


Figure 3.1: System Overview

This diagram shows the general functioning of the telemetry system. The data coming from the sensors are transmitted by Wi-Fi to the base station.

3.1 CAN Bus

The CAN bus is a widely used communication protocol in automotive applications. It was developed in the 1980s by Bosch to allow reliable communication between ECUs in vehicles. The CAN bus protocol is robust and fast. It can handle multiple devices on a single network while maintaining reliability and data integrity.

The CAN bus connects all the different devices of the car that need to communicate. The messages contain various parameters, such as engine speed, pedal status, temperature of the different components, etc. The BMS and the inverter in the actual car include a CAN interface. These two components provide some measurements such as voltage and current of the High Voltage system, power, temperatures, and angular speed. The car will also include more sensors also connected to the CAN bus.

The telemetry system is connected to the CAN bus and retransmits the data from the sensors.

3.2 SD Card

The system includes an SD card. The SD card has two functions.

The first function is to hold a configuration file. It is a JSON file that contains all the information the system needs to work correctly. The configuration file contains the settings for every sensor. A new sensor can be easily added by putting its CAN settings in the configuration file.

The second function is to register the measurements made by the sensors. When the button is pressed, the recording is activated, and the LED on the button blinks. When the button is pressed again, the recording is stopped. The data on the SD card provide a backup in case the transmission is interrupted. The data from all sensors are recorded at a high frequency. These data are saved in CSV files.

3.3 Wi-Fi Transmission

The telemetry system uses the Wi-Fi protocol to communicate, as seen in the Transmission Technology chapter.

The Wi-Fi transmission includes an optional redundancy. When it is activated, the system tries to connect to the first router. If it cannot connect, it tries with the second router. It then retries with the first if the second is unreachable, etc.

The settings of the Wi-Fi connection (SSID, password) are set in the configuration file. The system can use any standard Wi-Fi router. It employs the WPA2 security protocol but can be easily migrated to WPA3.

The data of only a selected number of sensors shall be live transmitted to the base station. A boolean field in the configuration file indicates whether the data of each sensor shall be transmitted or not. The transmission frequency is also defined in the configuration file. The defined frequency is lower than the recording frequency.

3.4 GPS

The telemetry device includes a GPS. It provides the GPS coordinates and the speed of the car. As for the sensors, the data are saved in the CSV files and can also be sent by the Wi-Fi transmission.

3.5 Front End

The software on the base station's PC displays the data sent by the telemetry device. The data saved on the SD card can also be loaded on the software. This software was developed by a student from the Business Information Technology program.

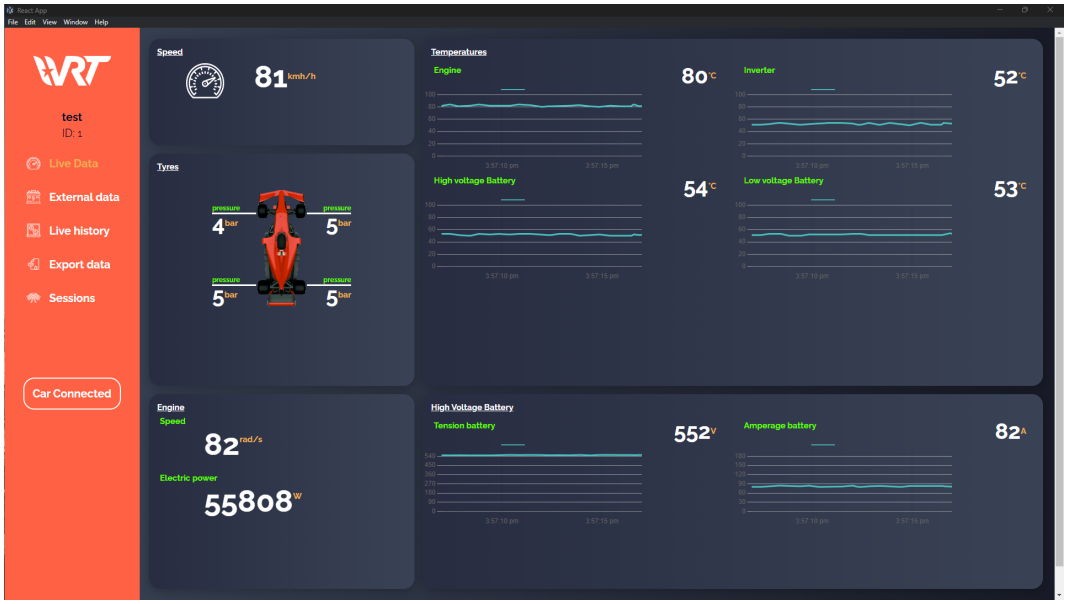


Figure 3.2: Telemetry System Display

The software can be downloaded at the following link :

https://vrt.simonbeaud.ch/

4 Hardware Development

4.1 Bloc Diagram

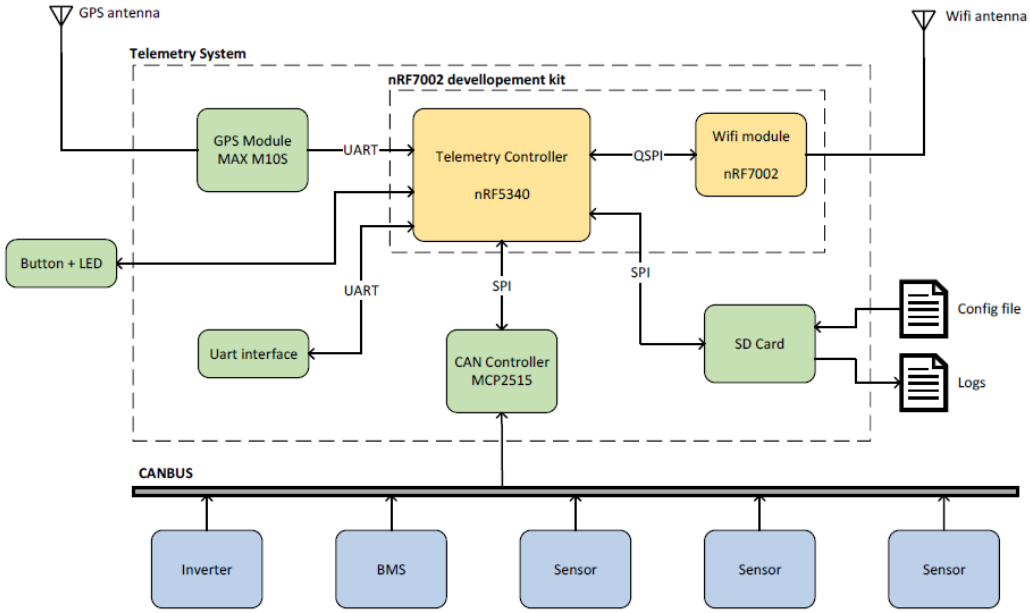


Figure 4.1: Bloc Diagram

This diagram shows the general architecture of the telemetry system's onboard device.

The main controller (nRF5340) is the central element of the system.

The CAN controller is connected to the nRF5340 via an SPI bus. On the other side, it is connected to the CAN bus to communicate with the BMS, the inverter, and all the sensors.

Another peripheral is the GPS module. It is connected to the nRF5340 via a UART communication. The GPS module uses an external antenna because it needs to be pointed to the sky, and the telemetry system will be placed in a case in the car.

The microSD card is connected to the nRF5340 via an SPI bus. The microSD card serves to store the measurements of the telemetry system. The memory card also holds a configuration file.

The nRF7002 chip handles Wi-Fi communication. It is connected to the nRF5340 via a QSPI bus. As for the GPS module, the Wi-Fi module uses an external antenna to improve the transmission.

The system also includes a button with an intern LED. The button permits to start and stop the recording of the logs, and the LED indicates the system's status. The button will be placed on the dashboard of the car.

The last peripheral is a UART interface. The telemetry system does not use it. It is just there for future system improvements (to connect an onboard computer, for example).

4.2 Main SoC

An SoC needs to be chosen for the project. The SoC must be available on a development kit to be able to start to program the telemetry system before the hardware is manufactured. The Nordic nRF7002-DK [12] is a development kit that integrates the Wi-Fi chip chosen in the Transmission Technology section. The host SoC of this kit is the nRF5340.

The nRF5340 [13] is a high-performance SoC from Nordic Semiconductors. It integrates two Arm Cortex-M33 processors, Bluetooth, high-speed SPI/QSPI, and other features. The application processor can be clocked at 64 or 128 MHz and is optimized for performance. It has 1 Mbyte Flash and 512 Kbyte RAM. The network processor is clocked at 64 MHz and is optimized for low power and efficiency. It has 256 Kbyte Flash and 64 Kbyte RAM. The nRF5340 can integrate the Zephyr RTOS.

This processor is powerful enough for the system, and its availability in a development kit makes it the perfect choice.

The telemetry device will use the development kit with a custom PCB directly plugged into it, in order to avoid spending too much time on hardware development.

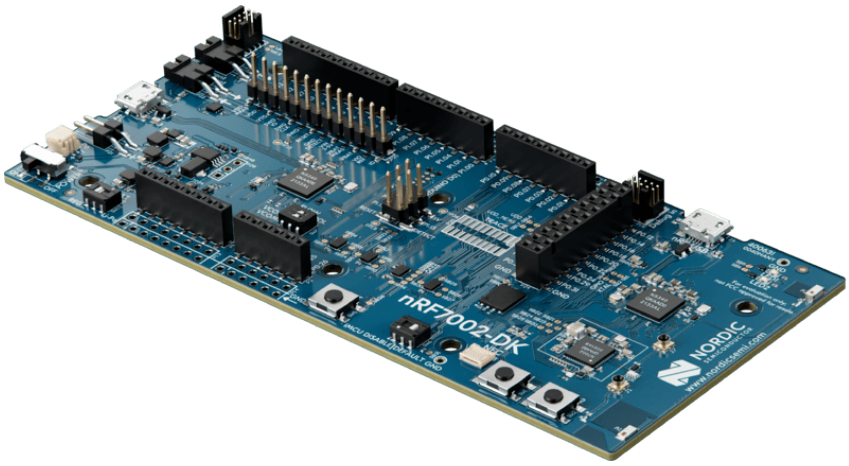


Figure 4.2: nRF7002-DK

4.3 GPS Module

4.3.1 Overview

The telemetry system needs to include a GNSS module. U-Blox is a leading provider of GNSS solutions. They produce high-quality positioning chips, modules, and antennas. Three GNSS solutions were studied.

The first is the SAM-M10Q [14] module. It is an all-in-one module with an internal antenna. It has a UART and an I2C interface. Integrating this module into the design is easy and straightforward.

Figure 4.3: SAM-M10Q module

The second option is the MAX-M10S [15] module. The specifications of this module are similar to the first module. It also has a UART and an I2C interface. However, it does not have an internal antenna. This makes it less easy to integrate into the design because GNSS external antennas are active, and HF circuitry needs to be designed.

Figure 4.4: MAX-M10S module

The last considered option is the UBX-M9140 [16]. It has two UART interfaces, one SPI, one I2C, and one USB interface. Unlike the two first options, this is not a module but a chip. It is much smaller than the modules, but the integration of this chip in the design is much more complicated.

Figure 4.5: UBX-M9149 module

These solutions are equivalent in terms of precision (standard precision). They all support concurrent reception of four GNSS (GPS, GLONASS, Galileo, and BeiDou). As the system will be placed in a housing in the car, the GNSS module needs to have an external antenna because it needs to be pointed at the sky. The first option

is, therefore, not possible. As the system does not need to be extremely small, the second option (MAX-M10S) is chosen to avoid spending too much time developing the electronic schematics.

The ANN-MB antenna from the same manufacturer is the best option to work with this module.

Figure 4.6: GPS Antenna

4.3.2 Schematics

This schematic shows the circuitry of the GPS module. The module is connected to the nRF5340 using a UART communication. This schematic is taken from the MAX-M10S integration manual [17].

On the RF side of the module, a circuit permits activating the antenna's power supply. The VCC_RF pin provides the antenna's power supply, and the LNA0_EN pin provides the activation signal. The connection between the RF_IN and the SMA connector (J5) is a 50 ohm microstrip transmission line.

The jumper on the left side permits to isolate the module and links the UART port to a connector. This allows to connect the module directly to a computer and to configure the module with the u-center2 [18] software. Shorting SB7 permits to generate a hardware reset. SB8 allows to enter in safeboot mode.

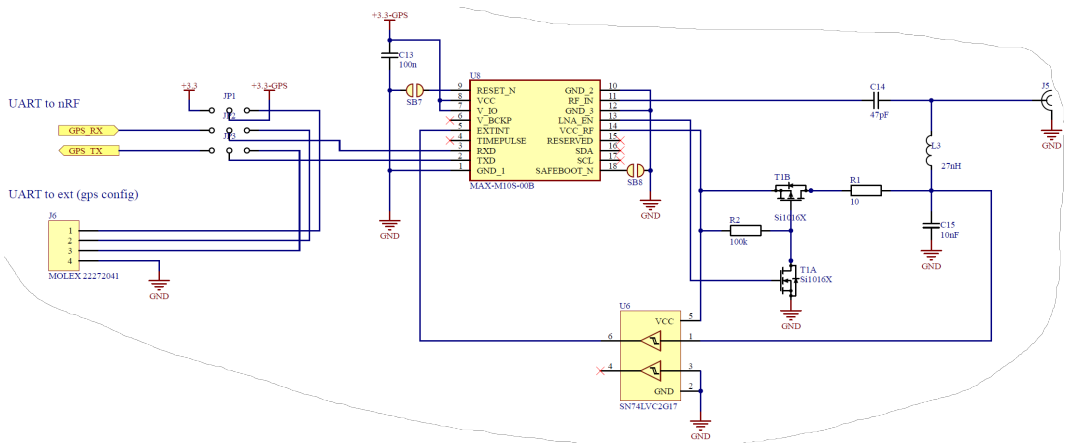


Figure 4.7: GPS Schematics

The TX and RX pins are connected to the nRF7002-DK by passing via a level shifter because the nRF provides 1.8 V signals, and the GPS module works with 3.3 V signals. Another UART interface also uses the level shifter. The telemetry system does not use it. It is just there for debugging purposes and future system improvements (to connect an onboard computer, for example).

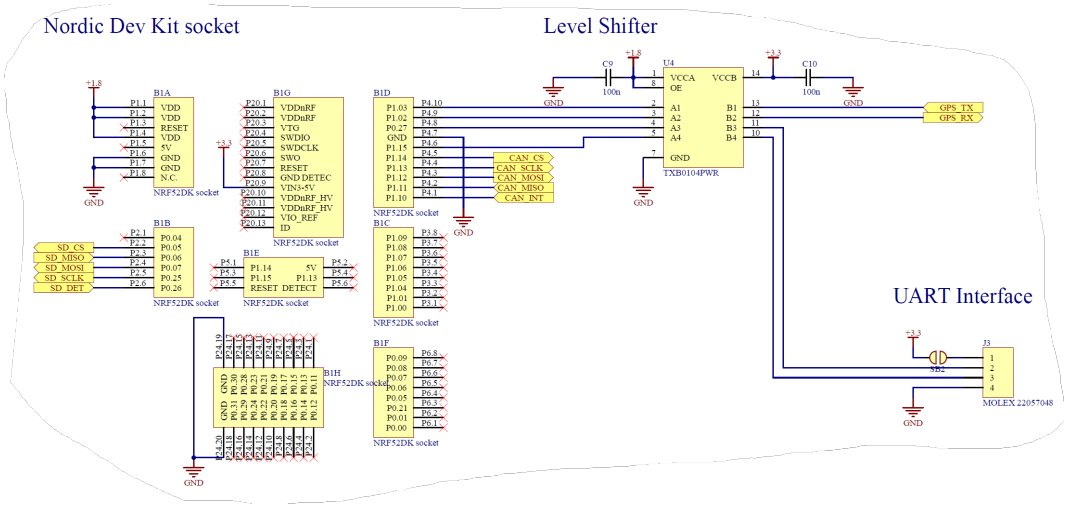


Figure 4.8: Nordic Dev Kit Socket and Level Shifter Schematic

4.4 CAN Bus Controller

4.4.1 Overview

The telemetry system communicates with the sensors by the CAN bus. For this purpose, a CAN Controller must be selected. The MCP2515 [19] is a CAN controller manufactured by Microchip. It implements the CAN specification, Version 2.0B. It is capable of transmitting and receiving both standard and extended data frames. It has an SPI interface to relate to the main SoC. This module handles all functions for receiving and transmitting messages on the CAN Bus.

A CAN transceiver must be used to generate the CANH and CANL signals. The SN65HVD232 [20] is a standard CAN transceiver. It just needs the TXCAN and RXCAN signals to generate the CANH and CANL signals.

The car environment is noisy and harsh, so the CAN must be protected against interference using EMC protections. All CAN connections must be made with shielded twisted pair cables in the car to reduce noise [21].

4.4.2 Schematics

As for the GPS module, the CAN controller also works with 3.3 V signals and needs a level shifter to communicate with the nRF7002-DK.

The CAN Controller (MCP251) is a tiny microprocessor requiring its own oscillator. The system uses a 16 MHz crystal oscillator. The TXCAN and RXCAN pins are directly connected to the CAN transceiver (SN65HVD232), which generates the CANH and CANL signals.

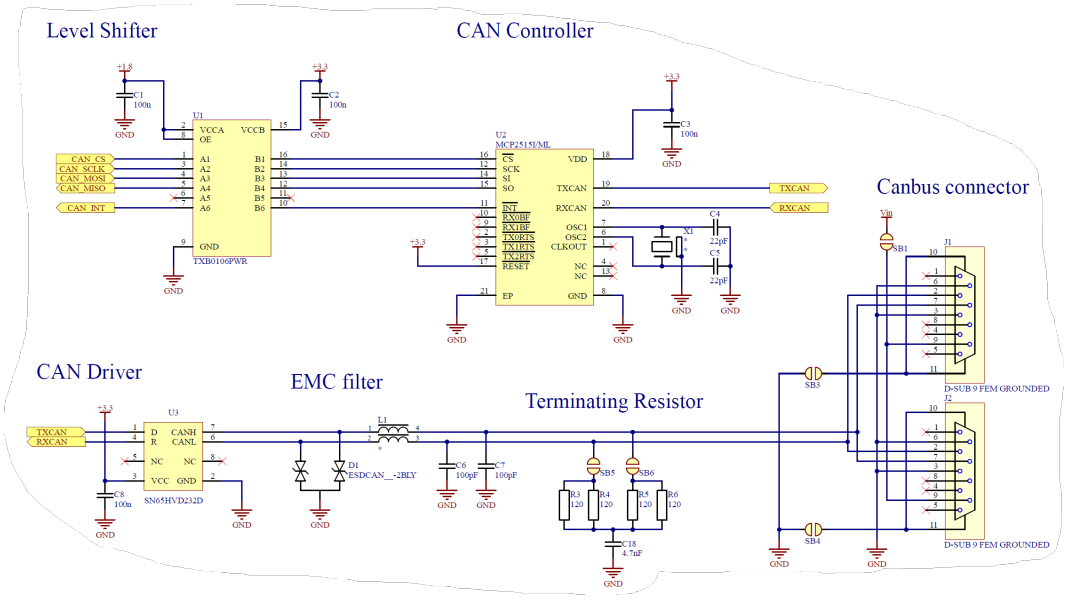


Figure 4.9: Can Controller Schematic

The system uses a common mode choke to protect the transmission against interference. Common mode chokes increase the CMRR by providing high impedance for the common mode signals (perturbation) and low impedance for the differential signals (CAN frames) [22].

Two tiny capacitors help to filter high frequencies [21].

The last elements of EMC protection are the two TVS diodes. TVS (Transient Voltage Suppressor) diodes permit to suppress the transient voltage created by the common mode choke to avoid damaging the CAN transceiver [22].

The design also includes a terminating resistor for the CAN Bus. Shortening the two solder bridges (SB5 and SB6) can connect the resistor. The system should use this terminating resistor because it is split and includes a capacitor that acts like a filter and permits to increase EMC immunity [21, 22].

The CAN Bus uses nine pins D-sub connectors (DE-9). The design implements the standard pinout. Two connectors permit placing the telemetry system in the middle of the bus. However, to increase EMC immunity, the system should be placed at one end of the bus and use the inbuild optimized terminating resistor.

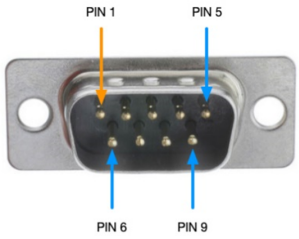
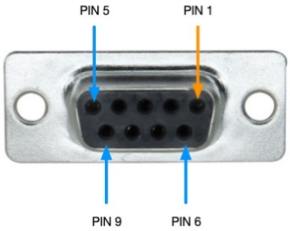
DE-9 Male	DE-9 Female	Pin	Function
		1	-
		2	CAN L
		3	GND
		4	-
		5	-
		6	GND
		7	CAN H
		8	-
		9	Power Supply

Figure 4.10: DE-9 Standard CAN Pinout

Shorting SB1 permits connecting the board's input power supply with the CAN Bus's power supply line. This allows powering up the sensors with the telemetry system power supply by the CAN cable. It can also be used to power up the telemetry system with the CAN cable.

Shorting SB3 and SB4 permits connecting the shield of the CAN cable to the ground. All CAN cable shields must be connected to the ground at only one end to avoid ground loops.

4.5 SD Card

The micro-SD card can communicate using the SPI protocol. As in the previous parts, the SD card also works with 3.3 V signals and needs a level shifter to communicate with the nRF7002-DK. The micro-SD card slot also provides a signal that detects when a micro-SD card is inserted (DETECT). The SPI BUS and DETECT signal are connected to the GPIO of the nRF7002-DK.

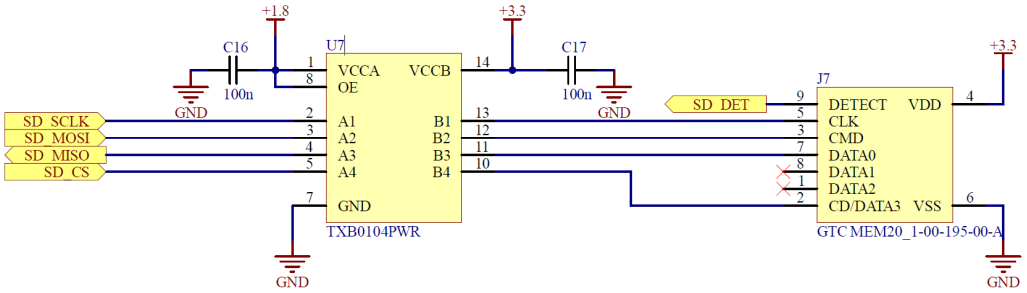


Figure 4.11: SD Card Slot Schematic

4.6 Power Supply

The LVS (Low Voltage System) 12 V battery of the car powers the telemetry system. The system can also be powered up by any power supply that provides a continuous voltage between 4.75 V and 36 V. The power supply must be able to provide 1.2 W. The power supply connector is a vertical Molex Microfit. All the LVS devices of the car use this model.

The components on the board need a 3.3 V power supply. A Traco Power DC/DC converter (TSR-1-2433) [23] provides the 3.3 V power supply. The design implements the input filter described in the component application note of the DC/DC converter [24].

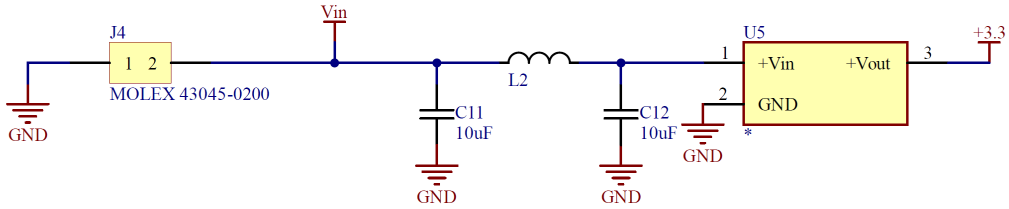


Figure 4.12: Power Supply Schematic

The nRF7002-DK is powered up by the 3.3 V on the Vin pin. It works with 1.8 V signals and has an onboard DC/DC regulator. It also provides a 1.8 V power source used by the level shifters.

4.7 Level shifters

The design implements many level shifters to interface the nRF7002-DK, which works with 1.8 V signals, with the other components, which work with 3.3 V signals. The TXB0104 (4 channels) and TXB0106 (6 channels) from Texas Instruments were selected for this application [25]. These level shifters are powered at the A port with the 1.8 V power supply and at the B port with the 3.3 V power supply. They allow a bidirectional translation between the nRF7002-DK and the other components.

4.8 Hardware Implementation

The full schematics and the bill of materials are available in the appendix. The Altium PCB project is available in the Git. Two Altium projects are available. The v1.0 is the version of the PCB used in the hardware device. However, this version uses the wrong CAN connector and the CANH and CANL signals were inverted. This error is corrected in the v1.1 version.

4.8.1 PCB Design

A dedicated PCB has been designed to implement all the subsystems described in the previous chapters. The PCB is designed to be plugged into the nRF7002-DK development kit. The following image shows the design of the PCB.

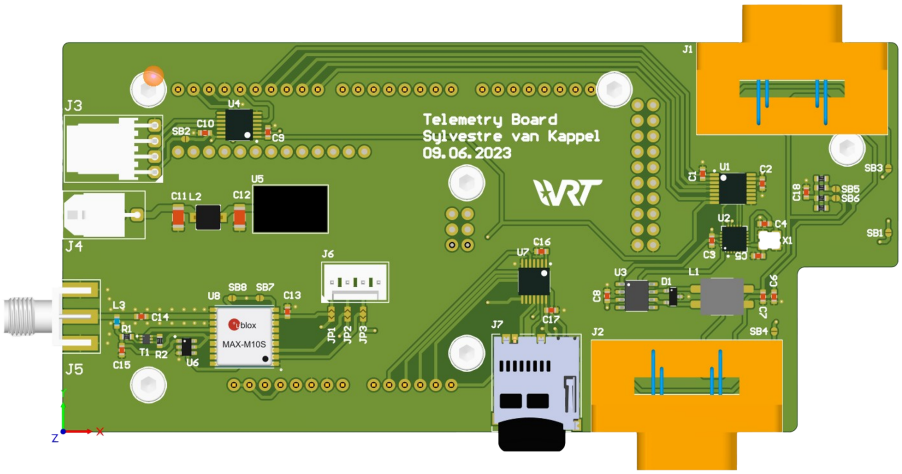


Figure 4.13: PCB Design

The PCB is a standard four-layer board from Eurociruits. The circuitry is simple and could be implemented on a two-layer PCB. However, the 50 ohm microstrip transmission line between the GPS module and the SMA connector requires a ground plane close to the top layer. This is why the design uses a four-layer PCB. The 50 ohm microstrip line is surrounded by vias connected to the ground. This permits you to get a good-quality transmission line.

The cutout on the bottom right side of the PCB permits access to the antenna connector for the Wi-Fi antenna on the nRF7002-DK development kit.

The following image shows the PCB plugged on the development kit :

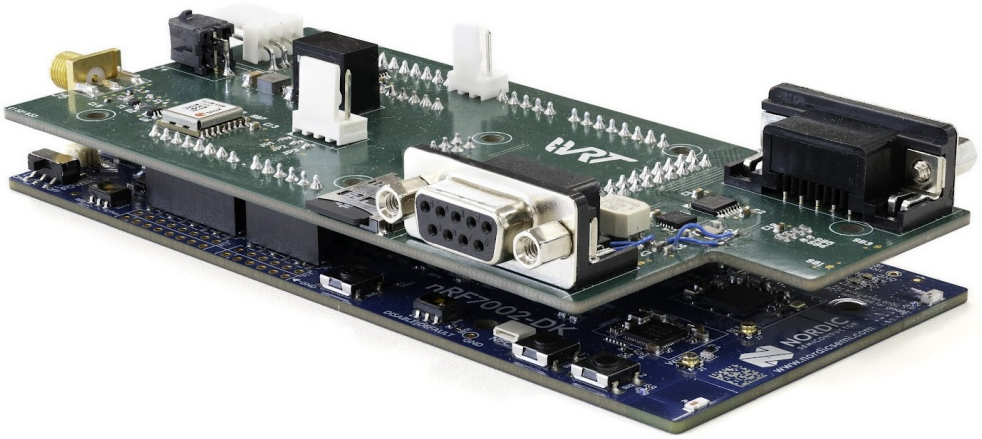


Figure 4.14: PCB Photo

4.8.2 Wi-Fi Antenna

The nRF7002-DK uses a Murata SWD connector for an external Wi-Fi antenna. This connector is also a switch. It redirects the signal on the internal antenna when the external antenna is not plugged in. Unfortunately, the male connector is not well held in the socket and could be unplugged by the vibrations. A small 3D-printed part was designed to hold the connector in place and avoid unfortunate disconnection.

(a) Holder *(b) Murata Adapter*

Figure 4.15: Wi-Fi Antenna Holder and Adapter

Two M3 threaded inserts in the bottom piece permit to attach the two pieces together with M3 screws. The parts pinch the PCB, and a third screw goes through a hole in the PCB, which locks the pieces.

The following image shows the assembly of the connector holder:

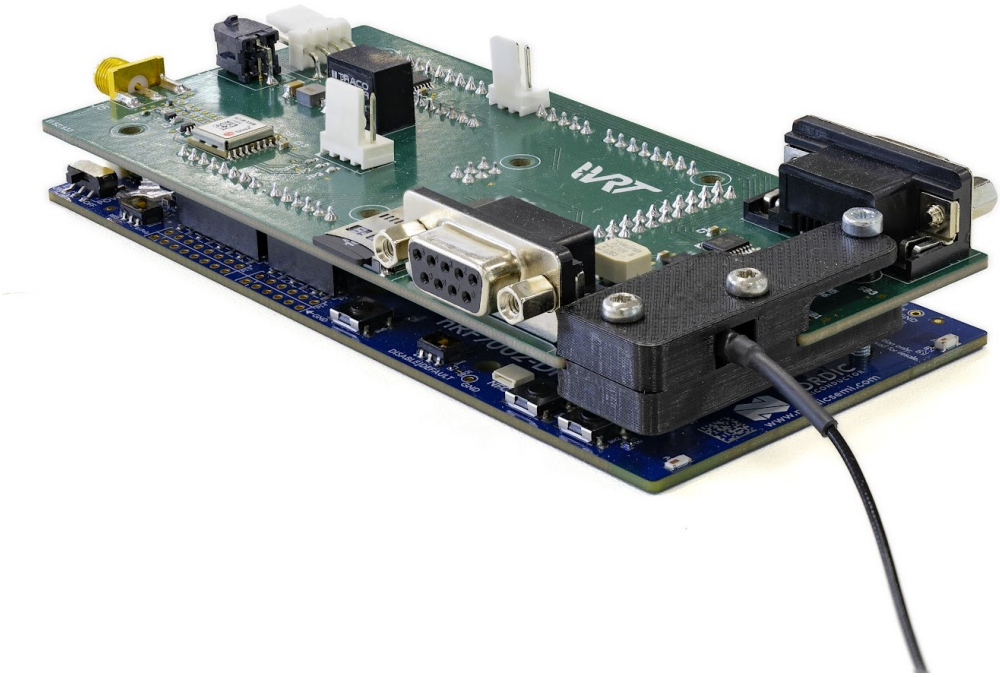
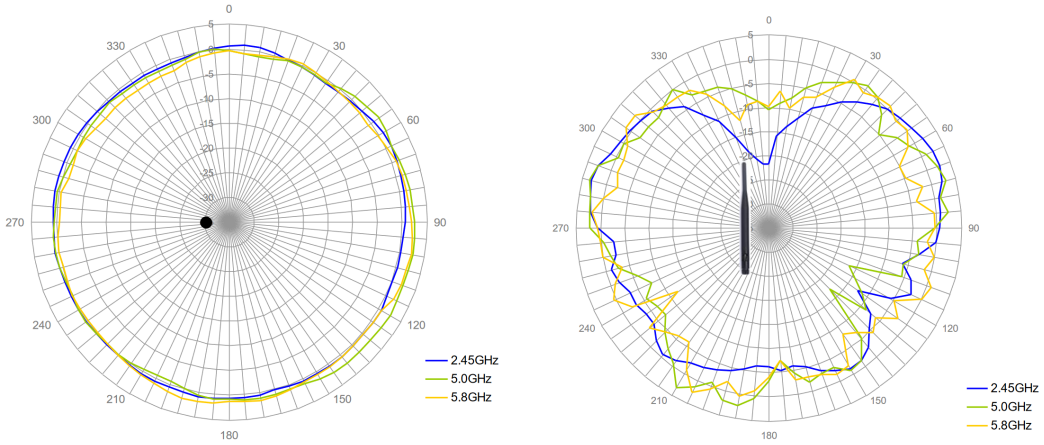


Figure 4.16: Wi-Fi Antenna Connector Holder Assembly

The antenna is then connected to the Murata adapter using an SMA coaxial cable. The antenna used in this design is a standard omnidirectional Wi-Fi antenna :

Figure 4.17: Wi-Fi Antenna

The Wi-Fi antenna must absolutely be placed in vertical position because optimum omnidirectional emission is obtained in the H-plane. The gain of the antenna is very bad (about -20 dBi) when the receiver is in the axis of the antenna. These two charts show the radiation pattern of the antenna [26]:

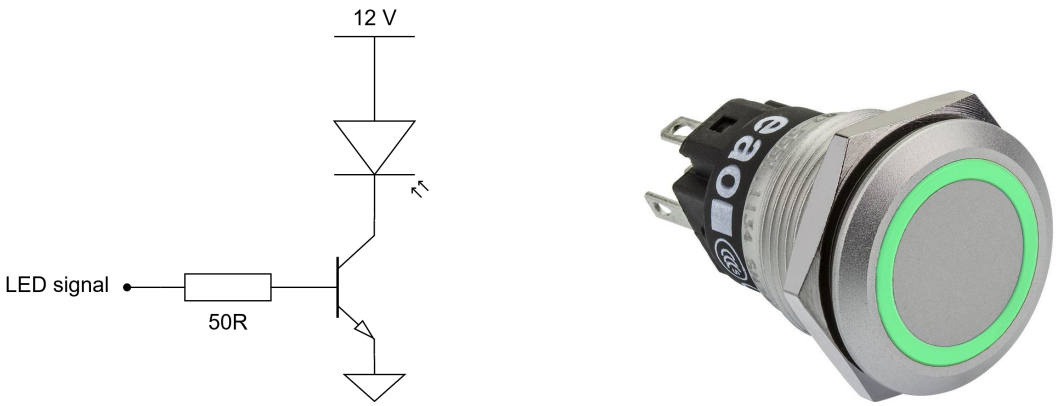


(a) H-Plane *(b) E-Plane*

Figure 4.18: Radiation Pattern of the Wi-Fi Antenna

4.8.3 External Button

A button with an integrated LED is connected to the system. The button is connected in parallel to the button 1 of the nRF7002-DK. The command for the LED signal is the same as for the LED 2 of the nRF7002-DK. The LED must be powered by a 12 V power supply. The button and the telemetry system must have the same ground. A small circuit with a NPN transistor and a resistor permits to activate the LED with the 1.8 V signal from the nRF7002-DK.



(a) LED Activation Circuit *(b) Button*

Figure 4.19: Button

4.8.4 System Description

The following figure shows the placement of all the peripherals of the telemetry system.

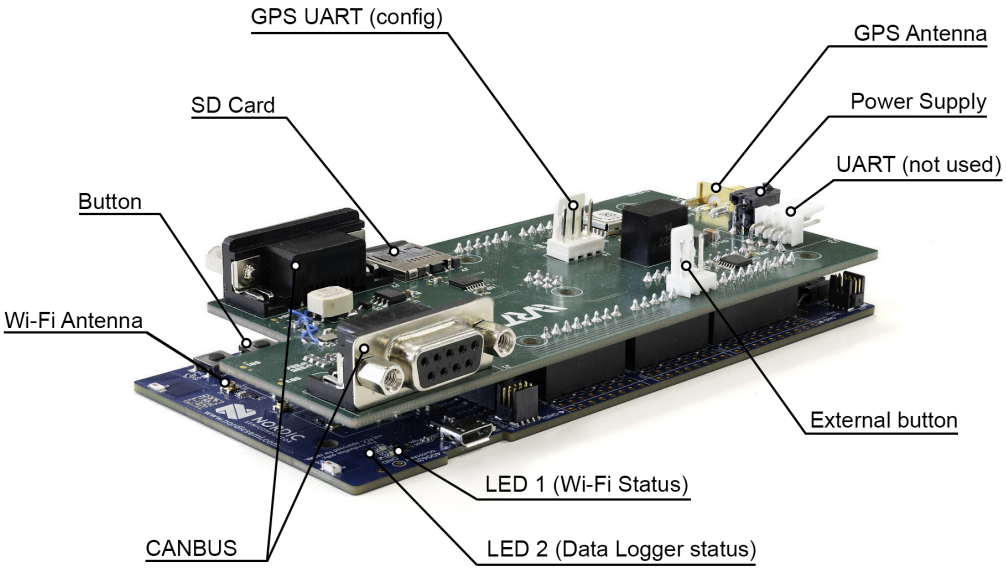


Figure 4.20: System Description

4.9 Wi-Fi Router

The Wi-Fi Router used by the telemetry system is the Asus RT-AX86U Pro. It was chosen because it is the latest router version used for the tests. The system can actually work with any Wi-Fi router.

Figure 4.21: Asus RT-AX86U Pro

5 Software Development

The software implemented on the nRF5340 uses the Zephyr RTOS. Zephyr RTOS is a compact operating system designed for devices with limited resources. Nordic's advanced hardware, renowned for its performance and efficiency, fits absolutely with Zephyr's lightweight and modular design.

The software is implemented on the application processor of the nRF5340. It was developed with the nRF Connect SDK v2.3.0.

The software implements the following tasks working in parallel:

- Wi-Fi Station
- UDP Client
- Data Sender
- Data Logger
- GPS Controller
- CAN Controller
- Button Manager
- LED Controller

The *main* function starts the Button Manager and the LED Controller first. Then the configuration file is read, and all the other tasks are started if the reading of the file is successful.

The design includes two buffers. The first is the sensor buffer. It contains the value and the configuration of every sensor present in the configuration file. The CAN Controller writes the values received by the sensors on this buffer. The second buffer is the GPS buffer. It works like the sensor buffer but takes its data from the GPS Controller.

The Data Sender periodically reads the sensor and GPS buffers and creates a JSON message containing the sensor and GPS values configured to be sent on the live transmission. Then it puts it in the UDP queue.

The UDP Client reads the UDP queue and sends the messages to the base station via the Wi-Fi using a UDP socket. The UDP protocol was chosen because it is faster than TCP, and the transmission does not need to be 100% reliable because the messages are periodically re-transmitted. Consequently, it is not a problem if sometimes a transmission contains an error.

When the button is pressed, the Data Logger creates a new CSV file on the SD card. It then periodically reads the sensor and GPS buffers and writes a new line on the CSV file with the read values. When the button is pressed again, the recording is stopped. The LED on the button is blinking when the Data Logger is recording.

The Wi-Fi Station manages the Wi-Fi connection.

The following diagram shows the system's general architecture:

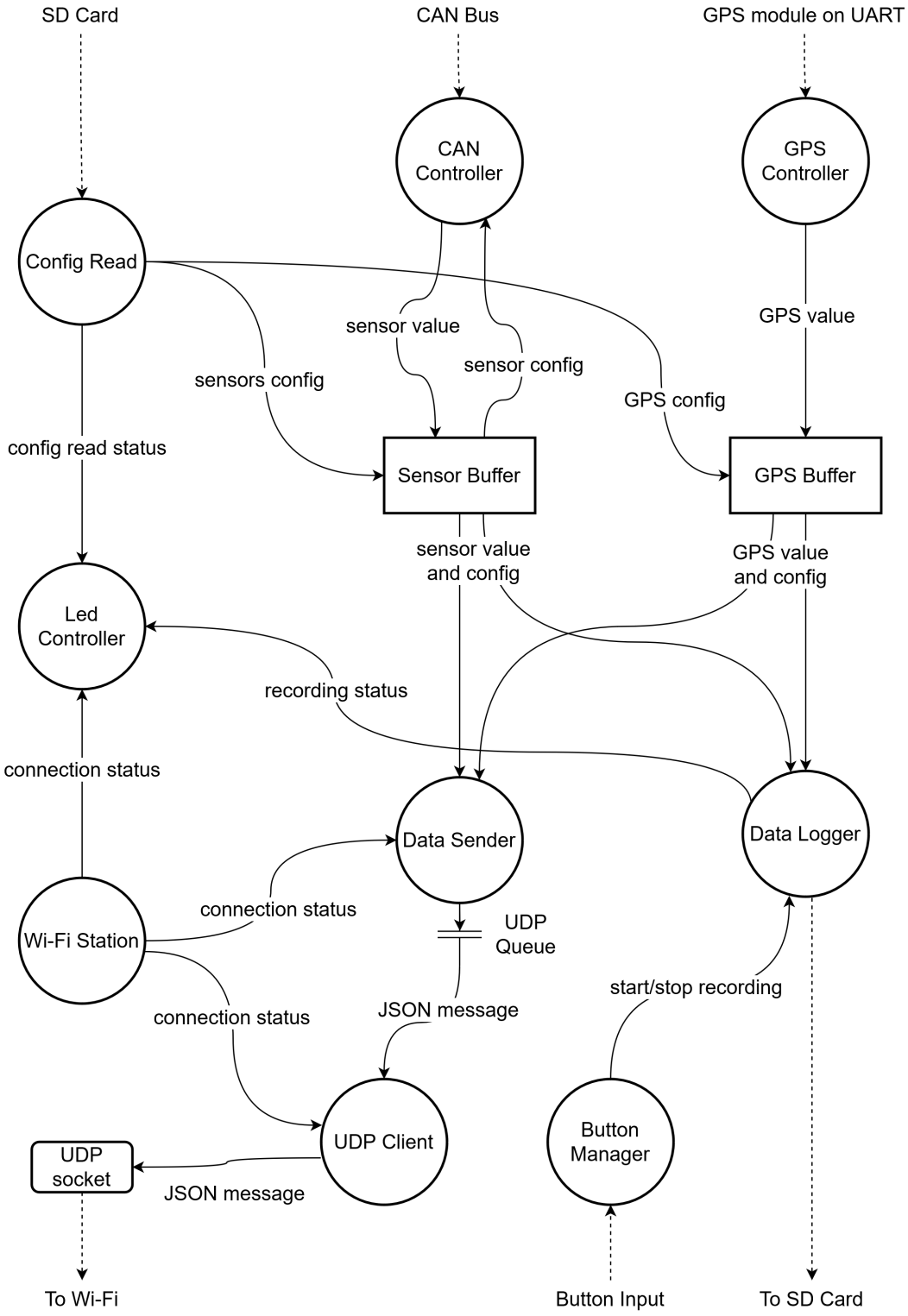


Figure 5.1: Data-Flow Diagram

5.1 Configuration File

First, the program reads the configuration file. If the SD card is not inserted, the configuration file is not present or not correctly established. The program does not start and the LED on the button remains ON. The LED switches to OFF if the system has started correctly.

The configuration file is a JSON file and contains the following configurations:

- Wi-Fi SSID and password
- Redundancy Wi-Fi SSID, password, and a boolean field to activate the redundancy
- UDP Server address and port. This information is contained in an array so that multiple servers can be selected (max 5)
- Telemetry data send rate in messages per second
- Logging data rate in measurement per second
- GPS parameters. Three data are treated. The coordinates, the GPS speed, and the fix status. For these three data, three parameters are configurable :
  - Name of the data on the live transmission
  - Name of the data on the logs
  - A boolean field to activate the data on the live transmission
- Sensor parameters. This information is contained in an array so that multiple sensors can be configured (max 100). For all sensors, five parameters are configurable :
  - Name of the sensor on the live transmission
  - Name of the sensor on the logs
  - A boolean field to activate the sensor on the live transmission
  - The CAN ID of the message containing the sensor's value
  - The format of the CAN message containing the sensor's value.

More details about the configuration file are available in the user guide and in the configuration file example in the appendix.

First, the SD card is mounted, and the base directory is opened. Then a loop searches for the configuration file. Once the file is found, it is opened and read. The code uses the disk access [27] and filesystem [28] libraries provided by Zephyr. The JSON string is parsed with the JSON library [29] provided by Zephyr, and the data are registered in a *struct*.

If the SD card cannot be read, if the configuration file is not present on the SD card or if the configuration file contains errors, an error code is returned, which stops the program.

The last task of the *config_read* function is to initialize the sensor buffer and GPS buffer.

The SD card is connected to the nRF5340 with an SPI bus. The overlay file requires the following configuration to work with the SD card:

```
&spi1 {
    status = "okay";
    cs-gpios = <&gpio0 5 GPIO_ACTIVE_LOW>;
    pinctrl-0 = <&spi1_default>;
    pinctrl-names = "default";

    sdhc0:sdhc@0 {
        compatible = "zephyr,sdhc-spi-slot";
        reg = <0>;
        status = "okay";
        mmc {
            compatible = "zephyr,sdmmc-disk";
            status = "okay";
        };
        spi-max-frequency = <24000000>;
    };
};
```

Listing 5.1: SD Card configuration

5.2 CAN Controller

The CAN Controller receives and puts the CAN data in the sensor buffer. The CAN bus uses the MCP2515 driver controlled by SPI. The CAN Controller works with the CAN v2.0 specification. It is compatible with 11 bits and 29 bits identifiers. The bus uses a 500 kbit/sec speed. The overlay file requires the following configuration to work with the CAN bus driver:

```
&spi2 {
    status = "okay";
    cs-gpios = <&gpio1 14 GPIO_ACTIVE_LOW>;
    pinctrl-0 = <&spi2_default>;
    pinctrl-names = "default";

    canbus:mcp2515@0 {
        compatible = "microchip,mcp2515";
        spi-max-frequency = <1000000>;
        int-gpios = <&gpio1 10 (GPIO_ACTIVE_LOW)>;
        status = "okay";
        label = "CAN_1";
        reg = <0x0>;
        osc-freq = <16000000>;
        bus-speed = <500000>;
        sjw = <1>;
        prop-seg = <2>;
        phase-seg1 = <7>;
        phase-seg2 = <6>;
        #address-cells = <1>;
        #size-cells = <0>;
    };
};
```

Listing 5.2: CAN Controller Configuration

The CAN Controller task is implemented in an independent thread. The code uses the CAN Controller library [30] provided by Zephyr. The system only needs to receive data. It does not send data. When the system gets a CAN message, it searches in the sensor buffer which sensors have the same CAN ID. A CAN message can contain the values of many sensors. The data from the CAN message are then put in the proper field of the sensor buffer.

The following diagram shows how the CAN Controller task work:

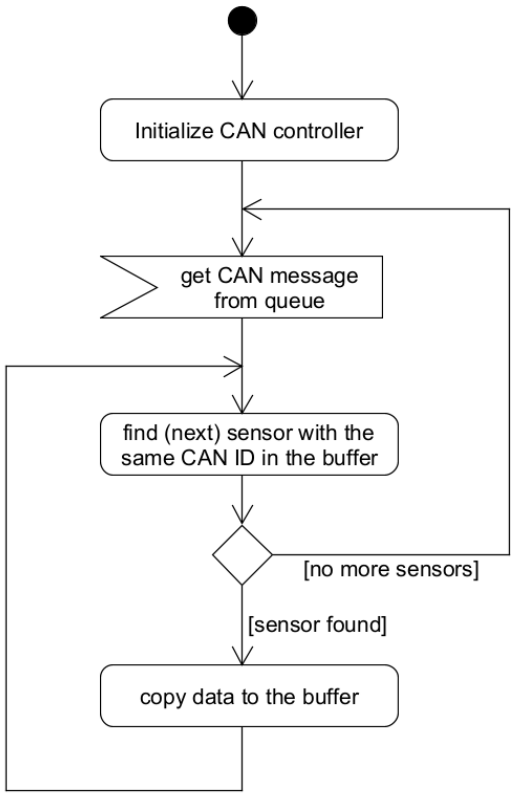


Figure 5.2: CAN Controller Activity Diagram

The library [30] puts the received messages in a message queue. A filter needs to be set up to receive the data. As the system needs to receive all the messages, the mask is set to 0.

In the infinite loop of the thread, the message queue is read. Then a loop goes through the sensor buffer. For every sensor, the message's ID is compared with the sensor's ID. If the IDs are the same, the value in the message is copied to the sensor buffer value field. As one CAN message can contain many sensor values, a field of the sensor buffer includes the information of which byte(s) of the message needs to be copied to the value field of the sensor buffer.

The following diagram show how the data are stored when a CAN frame is received :

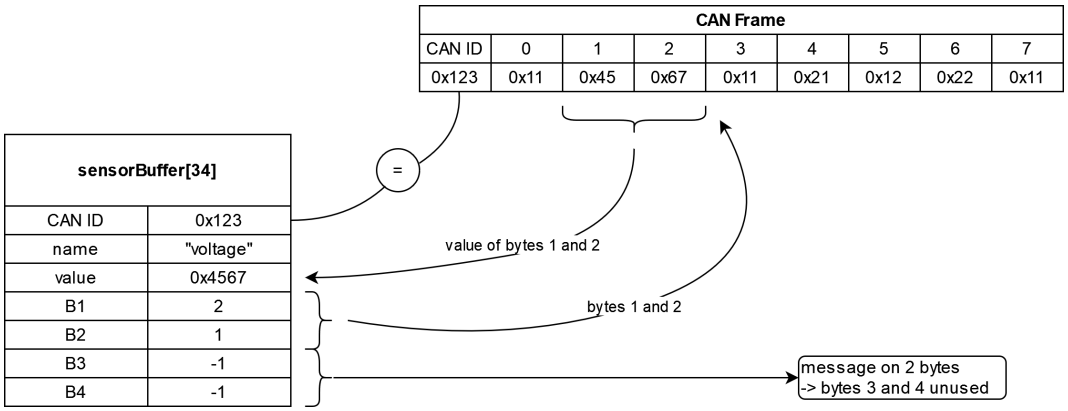


Figure 5.3: Can Receive Example

In this example, the box 34 of the sensor buffer contains the value of a voltage sensor. The message's ID is 0x123, and the value (0x4567) of the sensor is stored on two bytes, the bytes 1 and 2 of the CAN frame.

5.3 GPS Controller

The GPS Controller is implemented in an independent thread. It reads the UART port connected to the GPS module and fills the GPS buffer with the data received.

The JP1 jumper permits connecting the UART port of the GPS module to the external connector. The GPS module can be configured with the u-center2 software by connecting the external connector to the PC via a UART/USB converter.

The first set of parameters configures the module according to the hardware design. These parameters are taken directly from the component's integration manual [17].

	Keyname (Key ID)	Layer	Value (raw)
1	CFG-HW-ANT_CFG_SHORTDET	RAM (0)	1
2	CFG-HW-ANT_CFG_SHORTDET_POL	RAM (0)	0
3	CFG-HW-ANT_SUP_SHORT_PIN	RAM (0)	5
4	CFG-HW-ANT_CFG_PWRDOWN_POL	RAM (0)	0
5	CFG-HW-ANT_CFG_PWRDOWN	RAM (0)	1
6	CFG-HW-ANT_CFG_RECOVER	RAM (0)	1
7	CFG-HW-ANT_SUP_SWITCH_PIN	RAM (0)	7
8	CFG-HW-ANT_CFG_VOLTCTRL	RAM (0)	1

Figure 5.4: GPS Hardware Configuration

Then, the second set of parameters is the vehicle tracker preset available in the u-center2 software. This preset provides continuous tracking with a high position update rate, accuracy, and availability.

Only the 12 first parameters of the preset are used. The MSGOUT parameters are treated in the last set of parameters.

	Keyname (Key ID)	Layer	Value (raw)
1	CFG-SIGNAL-QZSS_ENA	RAM (0)	0
2	CFG-SIGNAL-QZSS_ENA	BBR (1)	0
3	CFG-SIGNAL-GLO_ENA	RAM (0)	1
4	CFG-SIGNAL-GLO_ENA	BBR (1)	1
5	CFG-SIGNAL-GLO_L1_ENA	RAM (0)	1
6	CFG-SIGNAL-GLO_L1_ENA	BBR (1)	1
7	CFG-NAVSPG-DYNMODEL	RAM (0)	4
8	CFG-NAVSPG-DYNMODEL	BBR (1)	4
9	CFG-ODO-PROFILE	RAM (0)	3
10	CFG-ODO-PROFILE	BBR (1)	3
11	CFG-RATE-MEAS	RAM (0)	100
12	CFG-RATE-MEAS	BBR (1)	100

Figure 5.5: GPS Vehicle Tracker Preset

The last set of parameters configures the rates of the messages the module sends. The module is configured to send messages containing the latitude, longitude, speed, and

the fix status every second. A message containing the number of connected satellites is sent every 5 seconds. The other messages are not interesting for the telemetry system, so they are not sent.

	Keyname (Key ID)	Layer	Value (raw)
1	CFG-MSGOUT-NMEA_ID_RMC_UART1	RAM (0)	1
2	CFG-MSGOUT-NMEA_ID_GGA_UART1	RAM (0)	1
3	CFG-MSGOUT-NMEA_ID_DTM_UART1	RAM (0)	0
4	CFG-MSGOUT-NMEA_ID_GBS_UART1	RAM (0)	0
5	CFG-MSGOUT-NMEA_ID_GLL_UART1	RAM (0)	1
6	CFG-MSGOUT-NMEA_ID_GNS_UART1	RAM (0)	0
7	CFG-MSGOUT-NMEA_ID_GRS_UART1	RAM (0)	0
8	CFG-MSGOUT-NMEA_ID_GSA_UART1	RAM (0)	1
9	CFG-MSGOUT-NMEA_ID_GSV_UART1	RAM (0)	5
10	CFG-MSGOUT-NMEA_ID_VLW_UART1	RAM (0)	0
11	CFG-MSGOUT-NMEA_ID_VTG_UART1	RAM (0)	1
12	CFG-MSGOUT-NMEA_ID_ZDA_UART1	RAM (0)	0

Figure 5.6: GPS MSGOUT Configuration

The messages are sent to the nRF5340 using a UART bus. The GPS module uses the NMEA standard [31] for the output messages. The module transmits the RMC, GLL, and GGA messages for the car's position. The VTG message contains the speed of the vehicle. The last message used by the telemetry system is the GSA message. It contains the Fix status of the module, which indicates if the position is valid.

The GPS module uses the uart3 peripheral of the nRF5340. It uses a 9600 Baud rate. The overlay file contains the following configuration:

```
&uart3 {
    status = "okay";
    current-speed = <9600>;
    pinctrl-0 = <&uart3_default>;
    pinctrl-names = "default";
};
```

Listing 5.3: GPS UART Configuration

In the code, an independent thread controls the GPS. During the initialization, an interrupt with a callback method is assigned to the UART receiver. The callback method reads the UART FIFO buffer and puts the message(s) in a message queue.

The following diagram shows how the GPS Task works:

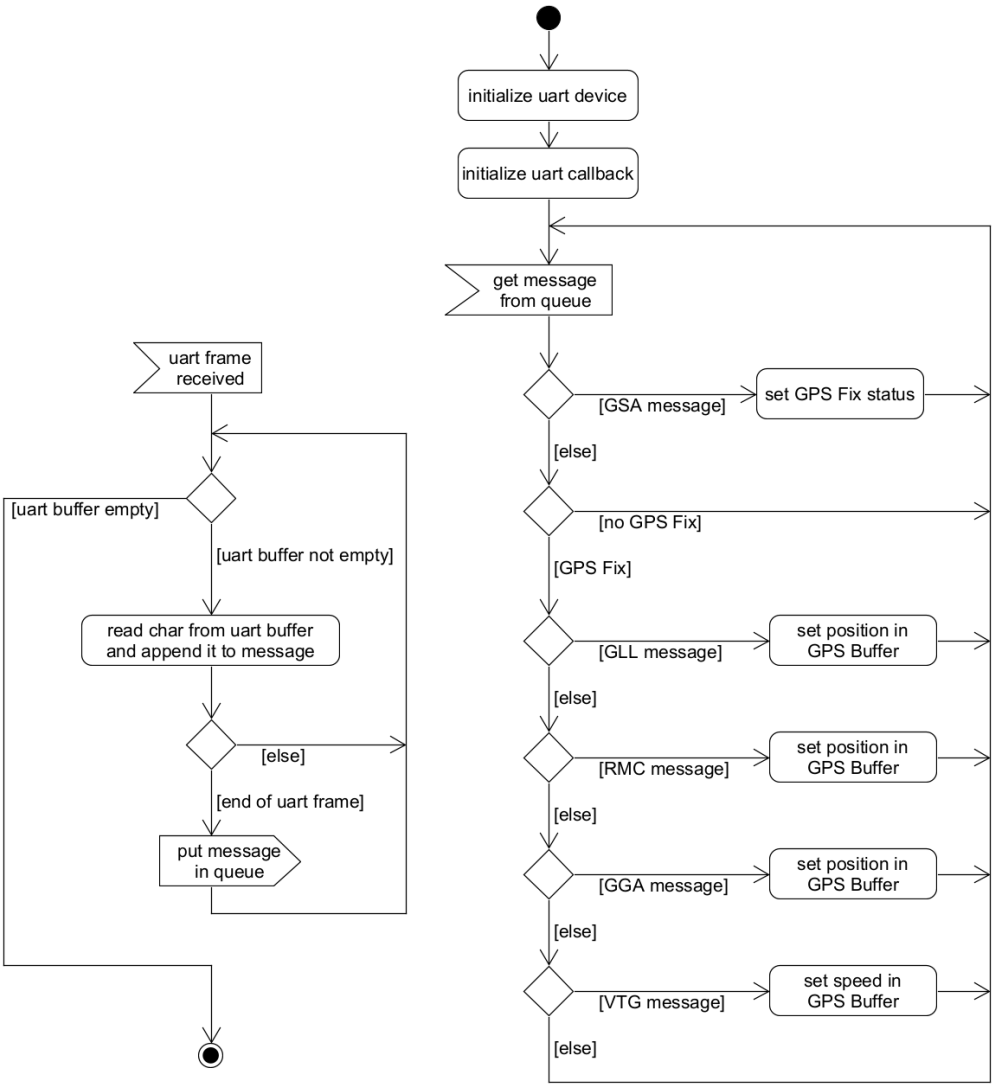


Figure 5.7: GPS Controller Activity Diagram

In the infinite loop of the thread, the messages are read from the message queue. Then the messages are analyzed according to the message type:

- GSA messages: The Fix status is read, and the result is stored in a boolean variable.
- GLL, RMC, and GGA messages: If the Fix status is set to "true", the position is read, and the result is stored in the GPS Buffer.
- VTG messages: If the Fix status is set to "true", the speed is read, and the result is stored in the GPS Buffer.

The NMEA messages are formatted as follows:

$GPRMC,161229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598, ,*10

In this RMC frame example, the coordinates are 37°23.2475N 12°158.3416W. The coordinates given by the module are in degree-arcminutes (*ddmm.mmmm* for the latitude and *dddmm.mmmm* for the longitude), and the sign is given by letters (N,S,E,W). In order to improve the reading, the coordinates are converted into decimal degrees.

To get the GPS position from the message, it is first separated at commas. The coordinates are then converted to decimal degrees. The following formula permits to convert degree-arcminutes into decimal degrees:

$$(d)dd + (mm.mmmm/60)(* - 1\ for\ W\ and\ S)$$

To work only with integer variables, the formula has been slightly changed. A coordinate is stored in 2 variables. One for the characteristic and one for the mantissa. Only the degree part is taken for the characteristic[1] during the parsing operation. For the mantissa[2], only the arcminute part is taken, the decimal point is removed (equivalent to an x10000 multiplication), and the number is divided by six.

Then, the coordinates are printed in the coordinate string of the GPS buffer with the *sprintf* function. The following scheme shows how the NMEA coordinates are converted into decimal coordinates:

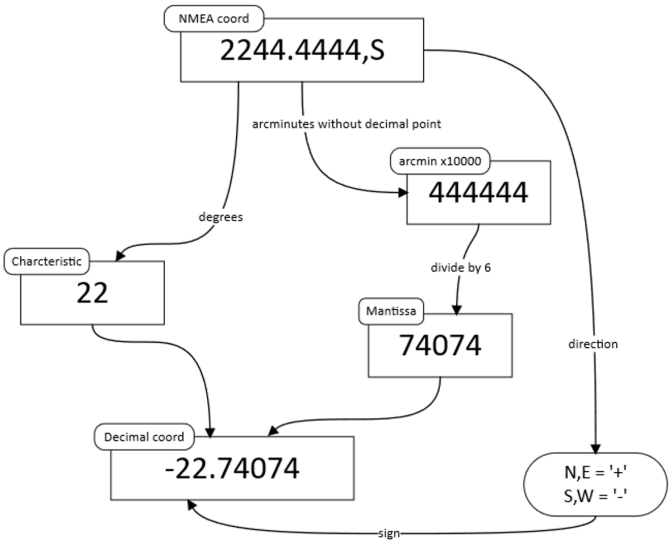


Figure 5.8: Conversion of NMEA to Decimal Coordinates

[1] Characteristic: Integer part of a decimal number
[2] Mantissa: Fractional part of a decimal number

5.4 Data Logger

The Data logger task records the measurement from the sensors on the SD card. The system communicates with the SD card across the SPI bus. The configuration of the SPI is detailed in the Configuration File chapter. The data from all the sensors and the GPS are saved on a CSV file. Each column contains one sensor's value, and each line corresponds to a time sample. The first column contains a timestamp. The record frequency is set in the configuration file. The maximum frequency depends on the number of sensors and the send frequency of the sensors on the CAN bus. The system was successfully tested with 30 measurements on 12 CAN frames sent 40 times per second with a record frequency of 40 measurements per second. More details about the maximum recording frequency are available in the Tests chapter.

The CSV file uses the following structure:

Timestamp [ms]	Sensor1	Sensor2	Sensor3
50	10	34567	156
100	12	30678	876
150	15	22344	459
...	...	...	...

Table 5.1: CSV File Structure

The button permits to start and stop the recording of the logs. When the LED on the button is blinking, the recording is enabled.

A timer periodically calls the Data Sender task. It takes the data from the sensor and GPS buffers. Then it formats a line with all the values, which is appended at the end of the CSV file. When the button is pressed, the recording is started. If the button is pressed again, the recording is stopped.

The code uses the Disk Access [27] and File System [28] libraries provided by Zephyr to work with the SD card.

During the initialization, the maximum length of a line is calculated. This length is then used when a line is created. Then the timer that periodically calls the Data Logger is started. The system uses a timer provided by the Zephyr environment [32].

In the timer interruption and the button callback method, works [33] that call the functions are submitted. Using work objects [33] permits spending as little time as possible in the interruption to avoid interrupting high-priority processes.

When a recording is started, the SD card is mounted, and a new file is created and opened. The first line of the CSV is written. It contains the name of all the sensors. Then the timestamp variable is set to 0.

When the recording is stopped, the file is closed, and the SD card is unmounted.

The following diagram shows how the Data Logger works:

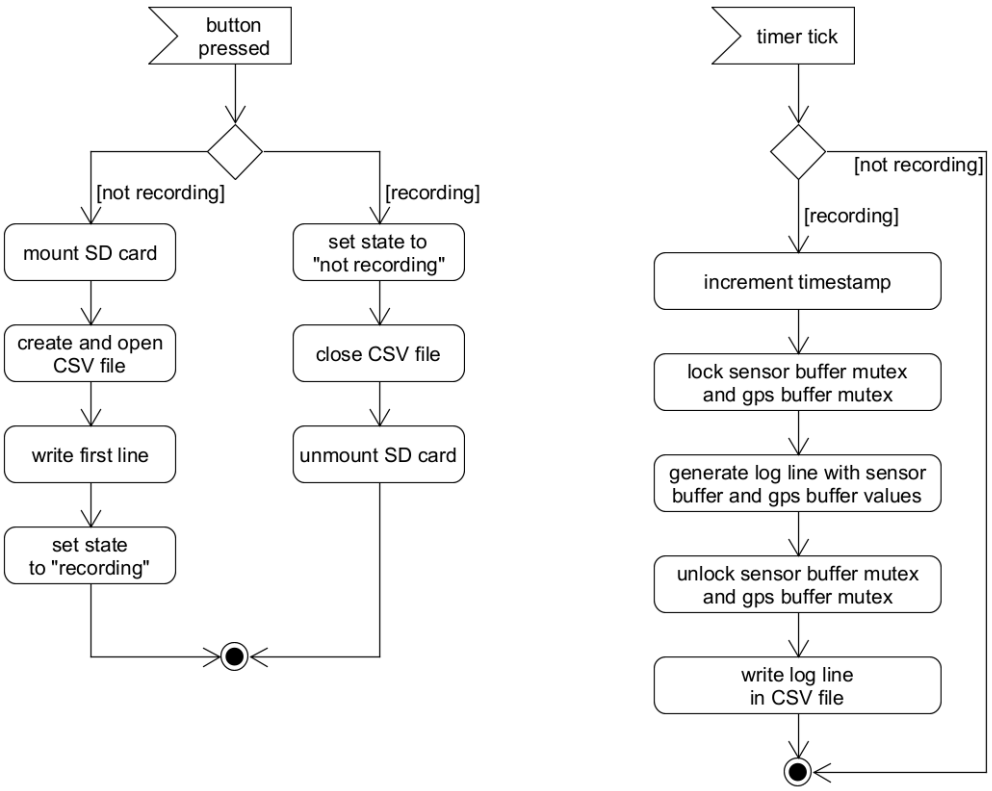


Figure 5.9: Data Logger Activity Diagram

The name of the file is *LOG_XXXX.CSV* where *XXXX* is a number that is incremented for every new file. The number is saved in the flash memory to avoid restarting at 0 when the system is powered off. The NVS library [34] from Zephyr is used to store the number in the flash memory. Every time a new file is created, the number is read from the memory, incremented, and rewritten on the memory.

In the periodic function, if the recording is enabled, a line is formed with the values from the sensor buffer, GPS buffer and timestamp variable. Mutexes [35] protect the buffers, as they are used in different threads. The line is then appended in the file and the timestamp variable is incremented.

At the end of the periodic function, the timestamp is incremented, and the synchronization function from the file system library is called. It flushes the open file's cache, ensuring data gets written to the storage media immediately to avoid data loss if the SD card is removed without stopping the recording.

5.5 Data Sender

The Data Sender task sends the data from the sensors to the remote PC. The send frequency is set in the configuration file. However, the send rate should be at maximum two transmissions per second. The system is technically capable of sending data at higher frequencies, but the live transmission is not designed for that purpose. Higher frequencies will negatively impact the system's general performance, and the receiving software could not be able to deal with high frequencies. However, the logs saved on the SD card by the Data Logger are designed for higher frequencies.

The Data Sender task is periodically called by a timer. The task takes the data in the sensor and GPS buffers. Then it formats a JSON string with the sensor values and names. Two fields are added at the end of the JSON string. One is monitoring a variable that indicates if the Data Logger is currently recording logs, and the other is a keepalive counter. It increments its value at every transmission and is restarted to 0 when it reaches 100. This allows the receiving software to detect when the system disconnects. The message is then added to a queue which is read by the UDP Client.

The JSON string uses the following structure:

```
{
    "Sensor1":42,
    "Sensor2":768,
    "Sensor3":76890,
    ...
    "KeepAliveCounter":78,
    "LogRecordingSD":true
}
```

Listing 5.4: Live Data JSON Structure

The following diagram shows how the Data Sender works:

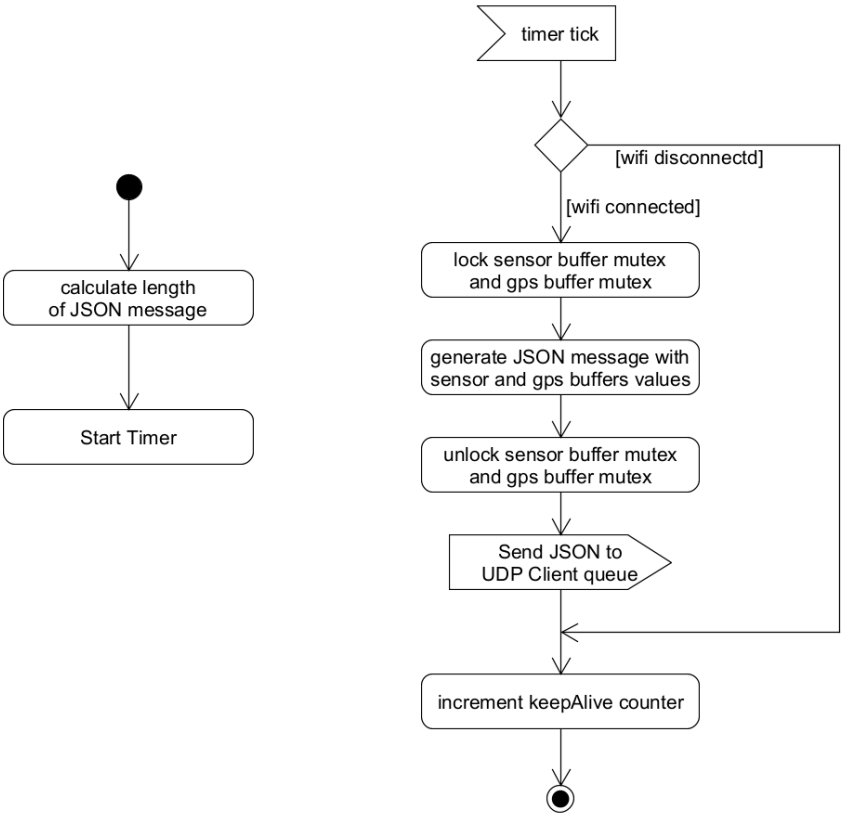


Figure 5.10: Data Sender Activity Diagram

During the initialization, the maximum length of the message is calculated. This length is then used when the message is created. Then the timer that periodically sends the data is started. The system uses a timer provided by the Zephyr environment [32].

In the timer interruption, a work [33] that calls the periodic function is submitted. Using a work object [33] permits spending as little time as possible in the interruption to avoid interrupting high-priority processes.

In the function, if the Wi-Fi is connected, memory is allocated to store the message. The size of the memory allocated is the maximum length of a message, which was calculated during the initialization of the task. The message is formed with the names and values in the sensor buffer and GPS buffer. Mutexes [35] protect the buffers, as they are used in different threads. The pointer on the message is then appended to the queue.

Finally, the keepAlive counter is incremented.

5.6 UDP Client

The UDP Client Task sends the messages to the network using a UDP socket. It is implemented in an independent thread. The system can send the data to different addresses and ports. This feature could be used to monitor the car on multiple PCs or send the data to multiple softwares. It is also useful if the redundancy Wi-Fi router uses different IP addresses. The configuration file contains the addresses and ports.

The messages that need to be sent are placed in a queue by the Data Sender Task. In the infinite loop of the thread, the queue is read, and the messages are sent via a UDP socket. If the Wi-Fi is disconnected, the UDP socket is closed, and the thread waits for the Wi-Fi to reconnect before reconnecting the socket and sending the data.

The following diagram shows how the UDP Client thread works:

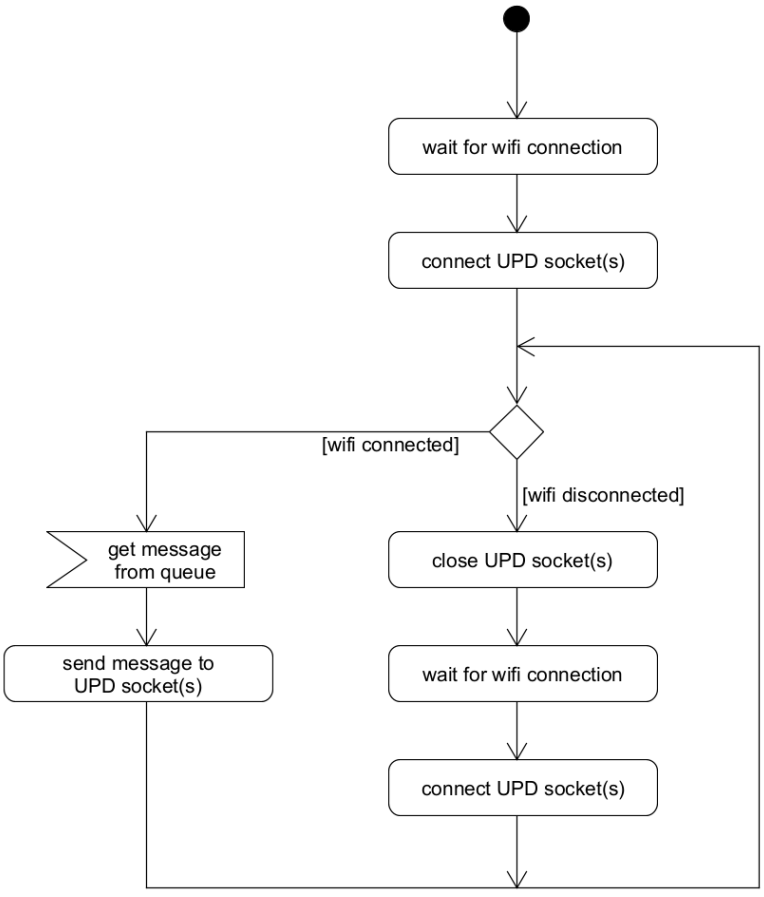


Figure 5.11: UDP Client Activity Diagram

The code uses the socket library [36] provided by the Zephyr environment. This library permits the creation of BSD sockets to send TCP or UDP packets using the Wi-Fi chip. A socket is created for each address/port pair if multiple addresses and ports are used.

The messages sent by the Data Sender are JSON strings that follow the Live Data Structure. The Data Sender dynamically allocates memory on a memory heap [37] to store messages to be sent. Then the pointer on the string is appended to the queue. In the UDP Client thread, the pointer is gotten from the queue, and the message is copied from the memory heap. The characters are copied one after the other, and the end of the message is detected with the '}' character, closing the JSON string. The memory allocated by the message on the heap is then freed.

5.7 Wi-Fi Station

The Wi-Fi Station task is a modified version of the Wi-Fi Station example from Nordic Semiconductors [38]. It is implemented in an independent thread. This task manages the connection with the Wi-Fi Router(s). A second router can offer a redundancy if the connection with the main router is lost. The SSID and passwords are set in the configuration file.

The security protocol is set to WPA2. It can be easily changed by commenting the WPA2 line and uncommenting the wanted protocol in the *prj.conf* file.

```
# Below configs need to be modified based on security
# CONFIG_STA_KEY_MGMT_NONE=y
CONFIG_STA_KEY_MGMT_WPA2=y
# CONFIG_STA_KEY_MGMT_WPA2_256=y
# CONFIG_STA_KEY_MGMT_WPA3=y
```

Listing 5.5: Security Parameters Configuration

When the redundancy is activated, the system tries to connect to the main router for ten seconds. If the connection succeeds, the thread goes to sleep mode. Otherwise, the system tries to connect to the redundancy router. If the connection succeeds with the redundancy router, the thread goes to sleep mode. Otherwise, the system retries with the main router, etc.

When the redundancy is not activated, the system only tries to connect to the main router.

When the router is connected, the thread goes to sleep mode. If the connection is lost, the thread retries to connect.

The following diagram shows how the Wi-Fi Station task works:

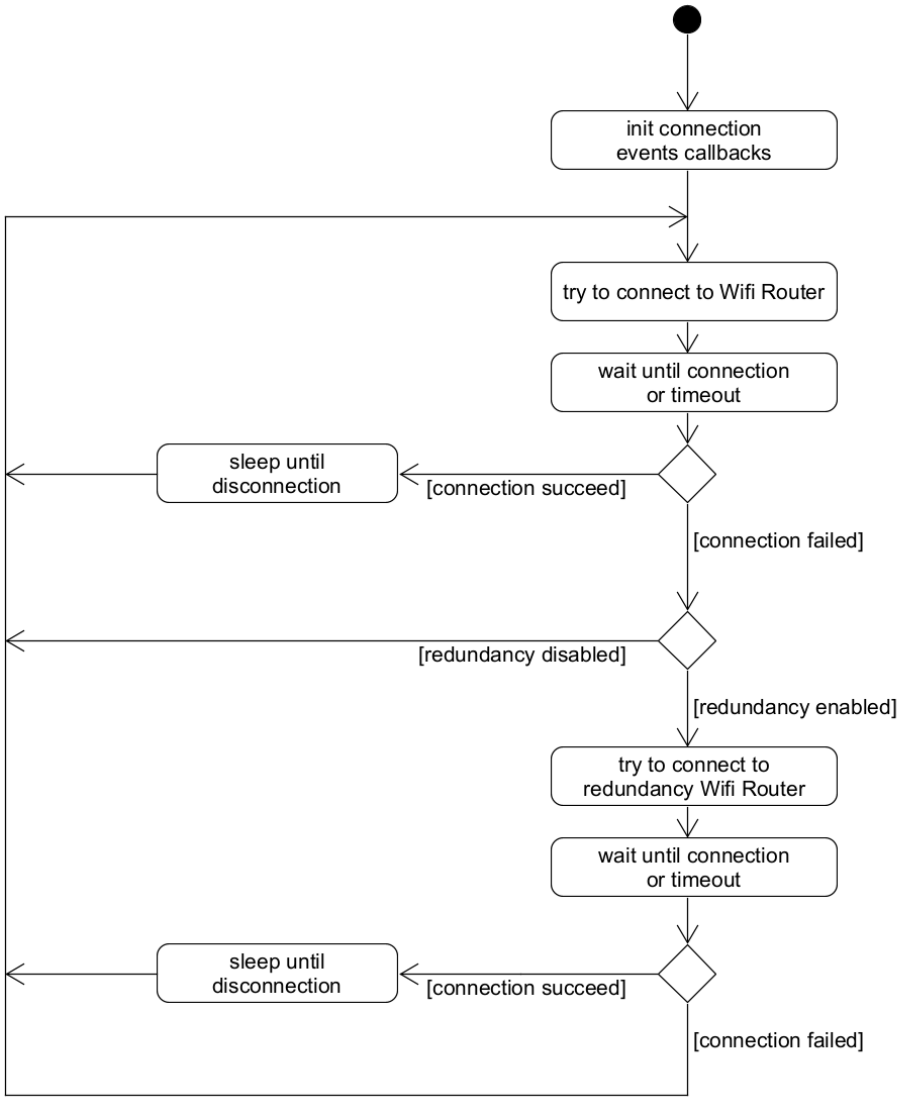


Figure 5.12: Wi-Fi Station Activity Diagram

The *context struct* contains the connection status of the Wi-Fi. It is declared as *extern* in the *deviceInformation.h* file. Every task that requires Wi-Fi connection status includes this file.

The code uses the network management library [39] provided by Zephyr. When a connect request is made, the thread waits some time. The *wifi_mgmt_event_handler* method is called when the Wi-Fi is connected. Then, the context struct *connected* attribute is set to 1, and the thread goes to sleep mode. The *net_mgmt_event_handler* method is called when an IP address has been assigned. The context struct *ip_assigned* attribute is then set to 1.

When the router is disconnected, the *wifi_mgmt_event_handler* method is called, which resets the context struct. Then the system retries to connect.

5.8 Button Manager

The button manager controls the button and the SD Detect signal, which detects whether the SD card is inserted. The button permits to start and stop the recording of the logs.

According to the Disk Access documentation [27], Zephyr does not support inserting or removing cards while running the system. The cards must be present at boot and must not be removed. The system is then rebooted when the SD card is inserted or removed.

During the initialization, the interruptions are configured. Then, the system works on the interruptions. The button input includes a debounce. The interruption triggers a work [33] after a small timeout. The interruption is recalled at every button bounce, and the timeout is restarted. The callback function is executed when the timeout expires, so when the state of the button signal is stable. In the callback function the button handler of the Data Logger task is called. The following diagram shows how the button manager works:

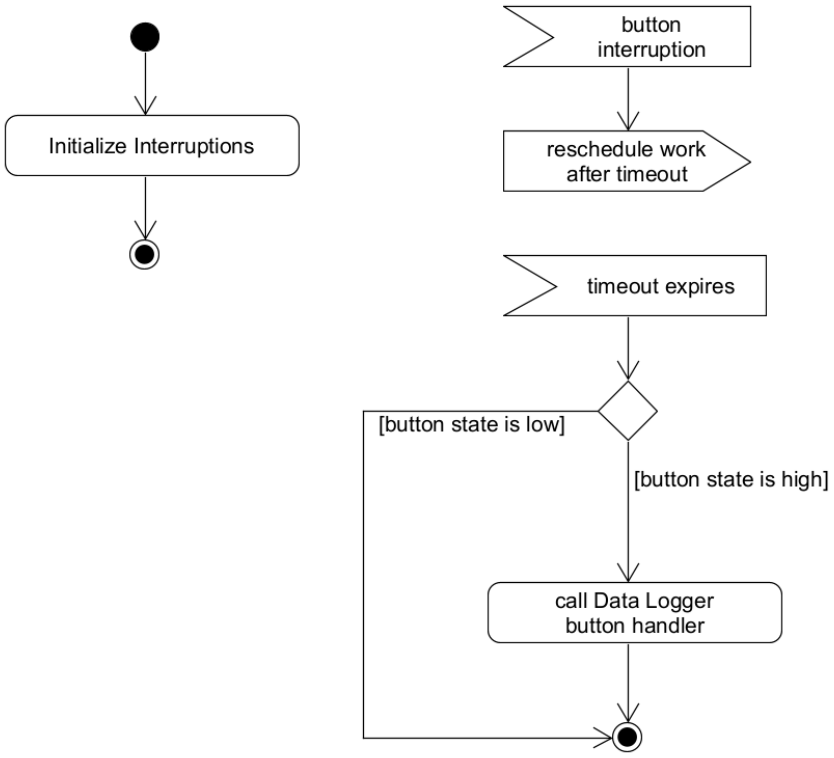


Figure 5.13: Button Manager Activity Diagram

5.9 LED Controller

The LED controller indicates the state of the system on the LEDs. There are two LEDs on the board. The first one is blinking when the Wi-Fi is connected and stays OFF if it is not connected. The second one remains ON after the boot if the configuration file has not been correctly read. Then during the system's normal functioning, it blinks when the logs are recording. The signal of this second LED also commands the LED of the button.

The LED task is implemented in an independent thread. The following diagram show how the LED task works:

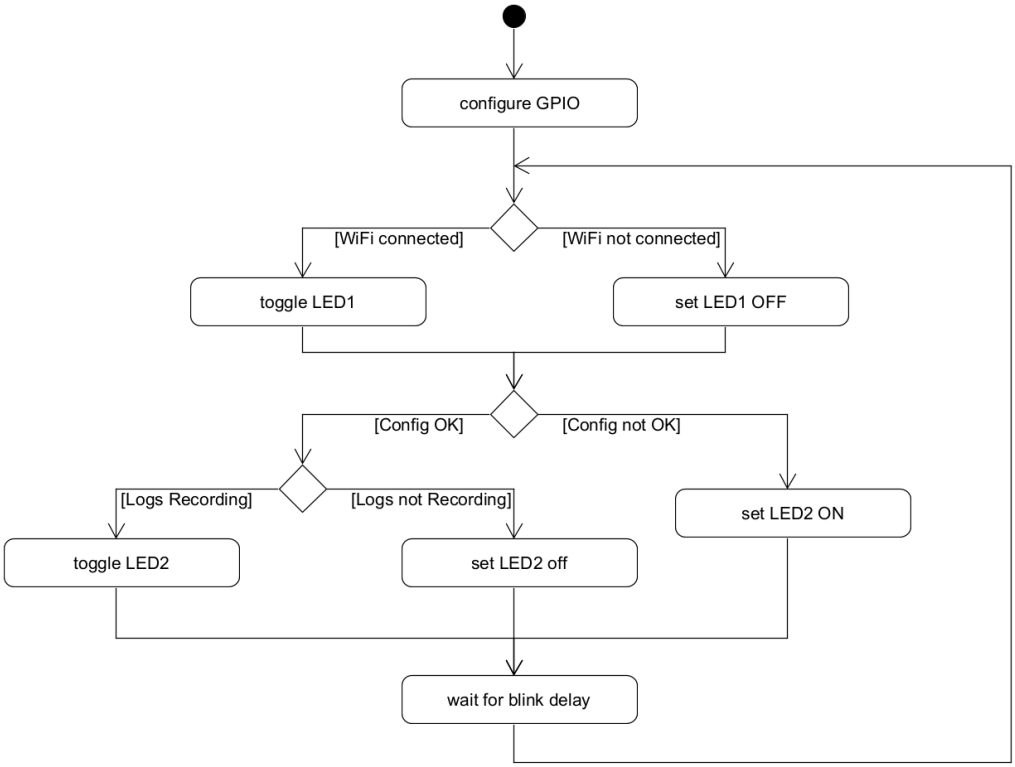


Figure 5.14: LED Controller Activity Diagram

6 | Tests

6.1 Range Tests

The objective of the range test is to determine the system's maximum range. Two tests were performed. The system was set up for both tests with a frequency of two transmissions per second. The tests were performed on a regular car because the Formula Student car was not available during the tests and is not authorized to drive on open roads. The embedded board used the external antenna placed vertically on the car's roof. The GPS antenna was also placed on the roof. The Wi-Fi router on the base station was set up with the default antenna.

The first test aimed to determine the absolute maximum range of the telemetry system. For this test, the car was driven away from the base station in a straight line until the connection was interrupted. The telemetry system was able to transmit the data over 750 meters.

The objective of the second test was to use the car in a realistic situation. For this test, the car was driven on the following route:

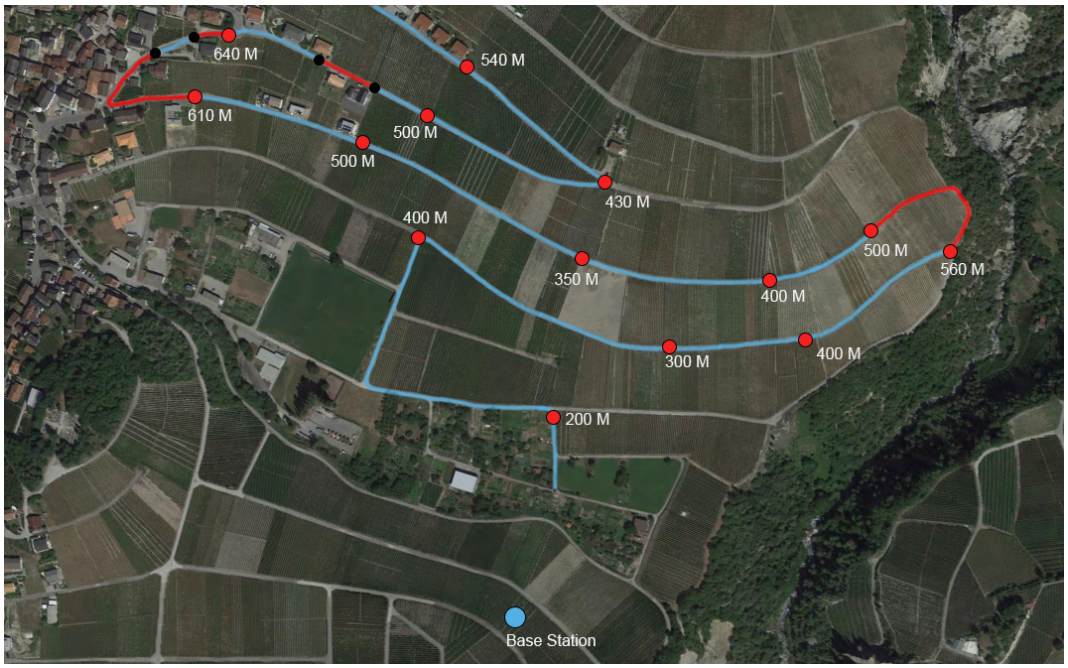


Figure 6.1: Range Test Route

The base station was placed on the blue dot at the image's bottom. The red points indicate the distance to the base station. The blue sections represent the zones where the telemetry system was connected during the test, and the red sections represent the zones where the transmission was interrupted.

The first disconnection occurred between the 560 M point and 500 M point on the right side of the map. This disconnection is normal because the line of sight was interrupted by the relief. The system is not supposed to work with obstacles between the car and the base station.

Three disconnections occurred around the 640 M point, which is, like the first disconnection, normal because the line of sight was interrupted by houses.

These two tests showed that the live transmission is working correctly in realistic conditions, and the system fulfilled the 500 meters objective with a transmission working up to 750 meters, without using sector antennas.

6.2 Connection Time Tests

At startup, the system takes about 10 seconds to connect to the Wi-Fi router.

As seen in the range tests, the connection can be lost if there is an obstacle between the car and the base station. This test aims to determine the time the system needs to reconnect once the system is able to communicate again. The telemetry system was placed in an anechoic chamber [1] , and the Wi-Fi outside the chamber to perform this test. With the door opened, the system could communicate correctly, and with the door closed, the communication was broken.

When the communication is broken, there is a 20 seconds delay. If the communication is reestablished during this delay, the system immediately reconnects. If the disconnection lasts over 20 seconds, the system takes about 6 seconds to reconnect once the signal is recovered.

If the redundancy is activated, the system could take up to 10 seconds longer to reconnect if only one of the two routers is reachable.

Since the line of sight should be guaranteed on the race circuits (except for small obstacles), there should be no long-time disconnections.

[1] An anechoic chamber is a room designed to stop reflections or echoes of either sound or electromagnetic waves. In this case, it is used as a Farraday cage.

6.3 Speed Test

The speed test aims to determine how many CAN messages the system can receive per second. Two tests were performed. The first one is to determine the absolute maximum frequency of CAN messages. The second one is to test the system in real-life conditions. For these tests, the CAN messages were sent with the Busmaster software [40] and a Tiny Can adapter [41]. The sensors on the car were simulated by Busmaster scripts sending random values. An example of these scripts is available in the appendix.

The maximum frequency of CAN messages the system can receive depends on the load of the nRF5340 processor. The processor is executing many different tasks in parallel, so there are many parameters that impact the maximum frequency of CAN messages the system can handle. All the tests were performed in the same real-life conditions and configurations to guarantee the most accurate results.

The system was configured as follows:

- Thirty sensors configured
- System connected to the Wi-Fi router
- Live Telemetry enabled for 12 sensors with a rate of 2 transmissions per second
- Log Recording activated with a rate of 20 records per second

For the first test, a single CAN message was sent periodically with increasing frequencies until the CAN receive buffer was overflowing. At 200 messages per second, the CAN buffer overflowed sometimes. However, it was sporadic, and the system was still able to work correctly. At 250 messages per second, the buffer overflowed more often, which sometimes caused system functioning problems. The maximum frequency is, therefore, 200 messages per second with the described configuration.

For the second test, multiple CAN messages were sent at different frequencies to simulate the real sensors on the car. The results of the second test were similar to the first, with a system working correctly until 200 messages per second.

These tests are, however, only valid for the configuration mentioned above. The same tests were done with a log recording rate of 40 measurements per second, and the CAN buffer overflowed at about 150 messages per second. As there are too many factors that have an impact, it is impossible to give a fixed maximum frequency. During the installation and configuration, the system must be tested, and if the buffer overflows, the sending frequencies of the sensors must be decreased. The system can be tested by connecting it to a PC via USB using PuTTY or any other serial monitor at 115200 bauds. The system will print errors on the serial monitor if the buffer overflows.

7 Conclusions

7.1 Project summary

This project aimed to develop and test a telemetry system for the Formula Student car of the HES-SO Valais-Wallis. This thesis deals with the onboard device of the system and the communication with the PC. The device transmits the data from the sensors connected on the CAN bus to a PC on the base station. The data are also saved on an SD card.

The project was separated into four different phases.

The first phase was an analysis to determine the best transmission technology for this application. The study concluded that a Wi-Fi transmission was the most convenient technology for this application.

The second phase was the hardware development. During this phase, the onboard device of the telemetry system was designed. The device is based on the nRF7002-DK, a development kit from Nordic Semiconductor designed for Wi-Fi application devices. A PCB was designed to be plugged into the nRF7002-DK. The PCB includes a CAN interface, a GPS module, and an SD card slot.

The third phase of the project was the software development. During this phase, a program was developed for the processor of the development kit (nRF5340). The program uses the Zephyr RTOS to implement all the tasks the system needs.

The last phase of the project was the testing. The first test measured the maximum range and the reliability of the system. The second test measured the time the system needed to connect to the base station. The last test aimed to determine the maximum data rate the system can treat. These tests were carried out in real-life conditions.

An analysis of the project's sustainability was carried out. It is available in the appendix.

7.2 Comparison with the initial objectives

The first objective was to define the data to transmit and choose the sensors. As the car on which the system will be integrated does not yet exist and the sensors needed are not yet defined, the system has been designed to be generic and completely configurable. New sensors can easily be added to the system by only changing the configuration file.

The second objective was to choose a transmission technology. An analysis was carried out, and the Wi-Fi protocol was selected. The tests have shown that the system is able to transmit the required volume of data over the required range of 500 meters.

The main objective was to develop the telemetry device's hardware and software. The tests have shown that the device is working correctly in real-life conditions. This objective can therefore be considered as fulfilled.

7.3 Encountered difficulties

In this project no insurmountable problems have been encountered. However some parts of the project were challenging.

The first challenge was to get familiar with the Nordic nRF and Zephyr environments. The necessary competencies have been acquired by study of the documentation.

The second challenge was to develop a generic and configurable system to which new sensors can easily be added. The CAN sensors can send their data in many forms, and it was challenging to find a way to make the system configurable for the different forms of the CAN frames.

7.4 Future perspectives

For future improvements, three points can be mentioned.

In the coming years, the car will include an onboard computer. The telemetry device has a UART port, which is currently unused. This port could be used by the onboard computer to start and stop the recording on the SD card and/or to monitor data on the dashboard.

Another possible would be to use the second core of the nRF5340 processor. Currently, only the application core is used. Some tasks could be executed by the network core, which will lower the load of the application core and improve the system's general performance. The frequency of CAN messages and the recording frequency could be increased.

The redundancy could also be improved using two Wi-Fi modules on the onboard device. This would allow the system to be connected to both routers simultaneously. If one connection is interrupted, the second one will still handle the data transmission without delay. This would also permit positioning the two Wi-Fi antennas in two different spots on the car to guarantee a line of sight with at least one antenna from all around the vehicle.

A | User Guide

A.1 System Overview

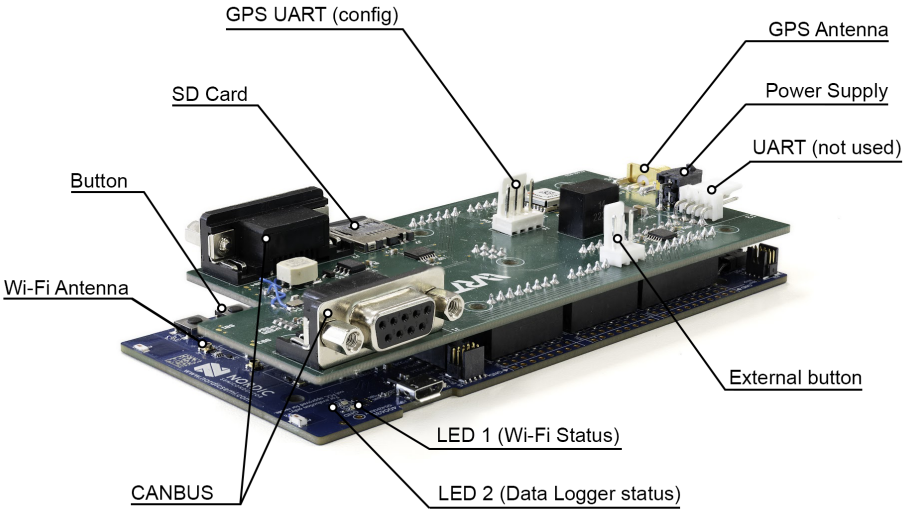
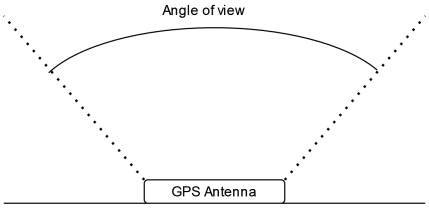


Figure A.1: System Description

This figure shows all the peripherals of the telemetry system. The GPS UART (config) and UART (not used) connectors are present for development purposes. These connectors need not to be used in the normal operation.

A.2 GPS Antenna

The GPS antenna must be placed flat and visible from the sky. The angle of view must be as wide as possible. The greater the angle, the greater the accuracy of the GPS. The GPS antenna shall be placed at at least 20 cm from the Wi-Fi antenna.



(a) Angle of View

(b) GPS Antenna

Figure A.2: GPS Antenna placement

A.3 Wi-Fi antenna

The Wi-Fi antenna must be placed vertically and visible from all around the car. It is connected to the board using an SMA cable and a Murata SWD adapter. The Murata adapter must be locked with the connector holder.

(a) Wi-Fi Antenna *(b) Murata Adapter*

Figure A.3: Wi-Fi Antenna and Adapter

The following image shows the assembly of the connector holder:

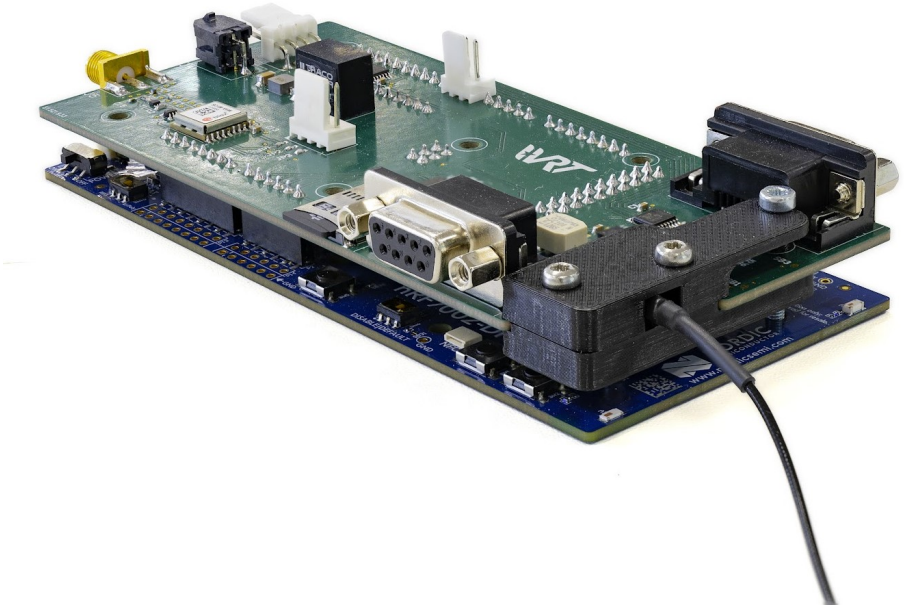


Figure A.4: Wi-Fi Antenna Connector Holder Assembly

A.4 Power Supply

The telemetry system requires the following power supply:

- 4.75 to 36 V DC
- 1.2 W

Shorting SB1 permits connecting the board's input power supply with the CAN Bus's power supply line. This allows powering up the sensors with the telemetry system power supply by the CAN cable. It can also be used to power up the telemetry system with the CAN cable.

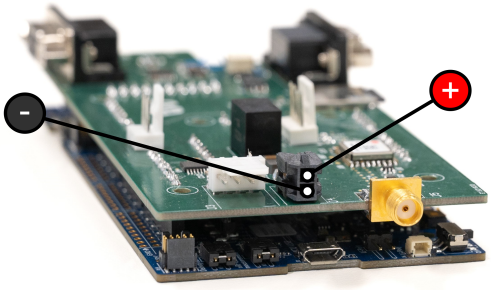


Figure A.5: Power Supply Connector

A.5 CAN Bus

The CAN interface works with the following settings:

- CAN 2.0
- Compatible with both standard or extended CAN ID
- 500 kbit/sec

The CAN Bus uses nine pins D-sub connectors (DE-9). The design implements the standard pinout. Two connectors permit placing the telemetry system in the middle of the bus. In this case, SB5 and SB6 must be opened. However, to increase EMC immunity, the system should be placed at one end of the bus and use the inbuild optimized terminating resistor by shorting SB5 and SB6.

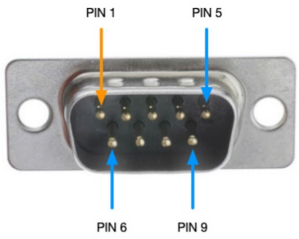
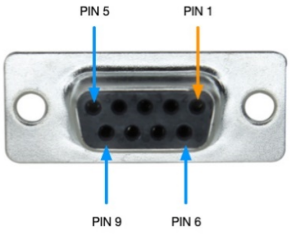
DE-9 Male	DE-9 Female	Pin	Function
		1	-
		2	CAN L
		3	GND
		4	-
		5	-
		6	GND
		7	CAN H
		8	-
		9	Power Supply

Figure A.6: DE-9 standard CAN pinout

Shorting SB1 permits connecting the board's input power supply with the CAN Bus's power supply line. This allows powering up the sensors with the telemetry system power supply by the CAN cable. It can also be used to power up the telemetry system with the CAN cable.

Shorting SB3 and SB4 permits connecting the shield of the CAN cable to the ground. All CAN cable shields must be connected to the ground at only one end to avoid ground loops.

A.6 External Button

The button on the board and the external button start and stop the recording on the SD Card. When the recording is activated, the LED on the button and the LED 2 on the board blink. If it stays ON after boot, there is an error with the configuration file.

The button requires the following power supply :

- 12 V DC
- The ground must be common with the telemetry system's ground.

A.7 SD Card

The SD Card contains a configuration file and the recordings. The system only works when the SD card is plugged in and when it contains the configuration file.

A.7.1 Recordings

The recordings are saved in CSV files. Recordings are started and stopped by pressing the button. Recordings must be stopped before the system is powered down.

A.7.2 Configuration File

The configuration file must be named *config.json* and be placed at the root of the SD card. If the LED on the external button stays ON after boot, there is an error with the file.

The file contains the following configurations:

Wi-Fi Router Informations

The *WiFiRouter* field contains the SSID and the password of the router that the system needs to connect:

```
"WiFiRouter":
{
    "SSID":"VRT-Telemetry",
    "Password":"TJJC2233"
},
```

Listing A.1: Wi-Fi Router Configurations

Redundancy Wi-Fi Router Informations

The *WiFiRouterRedundancy* field contains the SSID and the password of the redundancy router. The *Enabled* field permits to enable redundancy.

```
"WiFiRouterRedundancy":
{
    "SSID":"VRT-Telemetry",
    "Password":"TJJC2233",
    "Enabled": false
},
```

Listing A.2: Redundancy Wi-Fi Router Configurations

Server Informations

The *Server* field contains the PC's IP address and the telemetry software's UDP port. This information is contained in an array, so multiple (Max 5) address and port couples can be configured, for example if multiple PCs monitor the telemetry.

```
"Server":
[
    {
        "address":"192.168.50.110",
        "port":7070
    }
],
```

Listing A.3: Server Configuration

Frequencies

The *LiveFrameRate* parameter is the frequency at which data are transmitted to the PC in transmissions per second. This parameter should be set at 2 transmissions per second.

The *LogFrameRate* parameter is the frequency at which data are recorded in the SD card in samples per second. This parameter must not exceed 50 samples per second.

```
"LiveFrameRate":2,

"LogFrameRate":20,
```

Listing A.4: Frequencies Configurations

GPS

The *GPS* field contains three GPS parameter sets for the GPS coordinates, the speed measured by the GPS module and the fix status, which indicates if the measurements are valid.

The three parameter sets include a *NameLive* field, which contains the name of the data in the live transmission. The *NameLog* field indicates the name of the data in the CSV file, and the *Live Enable* field permits to activate the live transmission of the data.

```
"GPS":
{
    "Coordinates":
    {
        "NameLive":"GPSCoords",
        "NameLog":"GPSCoords",
        "LiveEnable":true
    },
    "Speed":
    {
        "NameLive":"GSPSpeed",
        "NameLog":"GSPSpeed",
        "LiveEnable":true
    },
    "Fix":
    {
        "NameLive":"GSPFix",
        "NameLog":"GSPFix",
        "LiveEnable":true
    }
},
```

Listing A.5: GPS Configurations

CAN Sensors

The *Sensors* array contains the information about all data provided by the sensors on the CAN bus. Each data includes a structure of 5 entries. The *NameLive* field contains the name of the data in the live transmission, and the *NameLog* field indicates the name of the data in the CSV file. The *Live Enable* field permits to activate the live transmission of the data. The *CanID* field provides the CAN ID of the message in a string with the *0x00* format.

The last field is the *CanFrame*. This field indicates which bytes contain the variable that needs to be taken from the received CAN frame. The system can treat up to 32 bits variables, which correspond to 4 bytes. The *CanFrame* field contains a string representing the frame on the bus. The number of bytes (DLC) must be the same in the representation and the real CAN frame. In the representation, every byte is separated by a ':'. The bytes unused by the variable are represented by an 'X'. The bytes containing the variable are represented by the 'B1', 'B2', 'B3', and 'B4' symbols, where 'B1' corresponds to the LSB. If the variable is only on 1 byte, only 'B1' is present in the representation. Only 'B1' and 'B2' are present for two bytes, Etc.

The following example shows two 16-bits (2 bytes) variables transported on the same frame (0x18):

```
"Sensors":
[
    {
        "NameLive":"EnginePower",
        "NameLog":"EnginePower_NL",
        "LiveEnable":true,
        "CanID":"0x18",
        "CanFrame":"B2:B1:X:X:X:X:X:X"
    },
    {
        "NameLive":"EngineTemperature",
        "NameLog":"EngineTemperature_NL",
        "LiveEnable":true,
        "CanID":"0x18",
        "CanFrame":"X:X:B2:B1:X:X:X:X"
    }
]
```

Listing A.6: Can Sensor Configurations Example 1

The *CanFrame* field can also include conditions. If one or more byte(s) contain(s) a hexadecimal number, the variable will only be taken if the CAN message contains the same number(s) as in the representation (at the same position). This permits a sensor to send different data on the same frame with one byte containing a function code.

In this example, the pressure of each tire is transmitted on an equivalent frame. The first byte permits to indicate which tire pressure is transmitted:

```
"Sensors":
[
    {
        "NameLive":"PressureTireFL",
        "NameLog":"PressureTireFL_NL",
        "LiveEnable":true,
        "CanID":"0x45",
        "CanFrame":"0x01:X:X:B2:B1"
    },
    {
        "NameLive":"PressureTireFR",
        "NameLog":"PressureTireFR_NL",
        "LiveEnable":true,
        "CanID":"0x45",
        "CanFrame":"0x02:X:X:B2:B1"
    }
]
```

Listing A.7: Can Sensor Configurations Example 2

The system can treat approximately 200 CAN messages per second. However, this limit can change a lot depending on all the settings. When configuring the sensors, the system must be tested in real conditions (Transmission activated and Log Recording activated), and if the CAN buffer overflows, the send frequencies of the sensors or the record frequency of the logs must be reduced. The system must be tested for several minutes.

To monitor the CAN buffer, the system can be connected to PuTTY or any other serial monitor with the USB connector under the power supply connector. The system prints warnings to indicate how full the buffer is and errors if the buffer overflows.

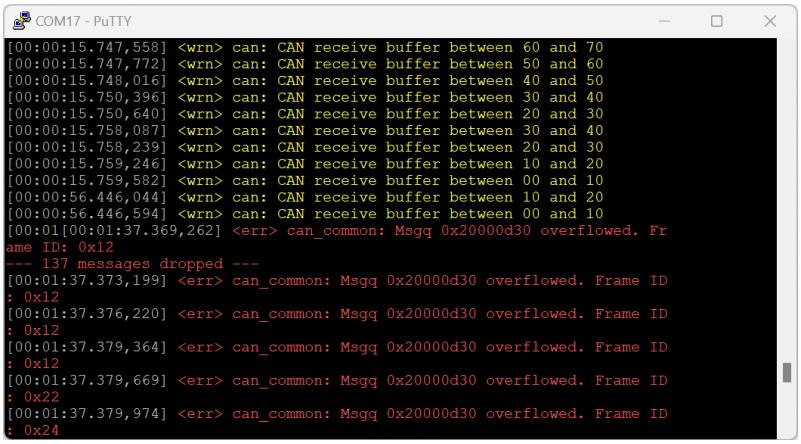


Figure A.7: Buffer Overflow Example

A.8 Solder Bridges

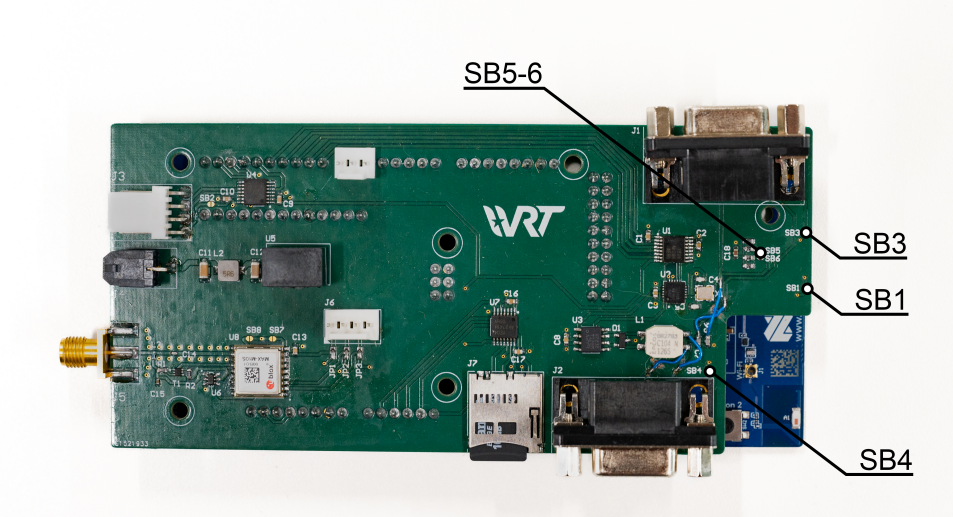


Figure A.8: Solder Bridges

Shorting SB1 permits connecting the board's input power supply with the CAN Bus's power supply line. This allows powering up the sensors with the telemetry system power supply by the CAN cable. It can also be used to power up the telemetry system with the CAN cable.

SB3 permits connecting the shield of the J1 connector with the ground.

SB4 permits connecting the shield of the J2 connector with the ground.

SB5 and SB6 permit connecting the CAN bus's internal terminating resistor. Both solder bridges must be shorted to connect the resistor, and they must both be opened to disconnect the resistor.

SB2, SB7, and SB8 are there for development purposes and must stay open.

B | Configuration File Example

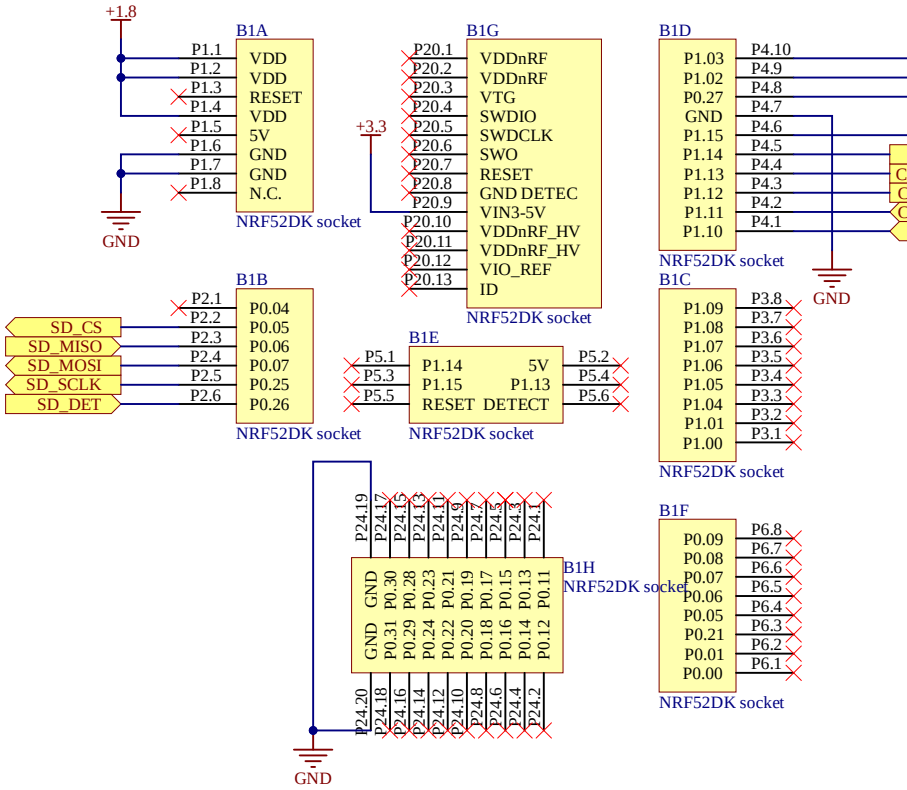
This is an example of the configuration file with five sensors configured. This example is working with the software developed in 2023. The entire file with all the sensors is available in the Git.

```
{
    "WiFiRouter":
    {
        "SSID":"VRT-Telemetry",
        "Password":"TJJC2233"
    },
    "WiFiRouterRedundancy":
    {
        "SSID":"VRT-Telemetry",
        "Password":"TJJC2233",
        "Enabled": false
    },
    "Server":
    [
        {
            "address":"192.168.50.110",
            "port":7070
        }
    ],

    "LiveFrameRate":2,

    "LogFrameRate":20,

    "GPS":
    {
        "Coordinates":
        {
            "NameLive":"GPSCoords",
            "NameLog":"GPSCoords",
            "LiveEnable":true
        },
        "Speed":
        {
            "NameLive":"GSPSpeed",
            "NameLog":"GSPSpeed",
            "LiveEnable":true
        },
        "Fix":
        {
            "NameLive":"GSPFix",
            "NameLog":"GSPFix",
```

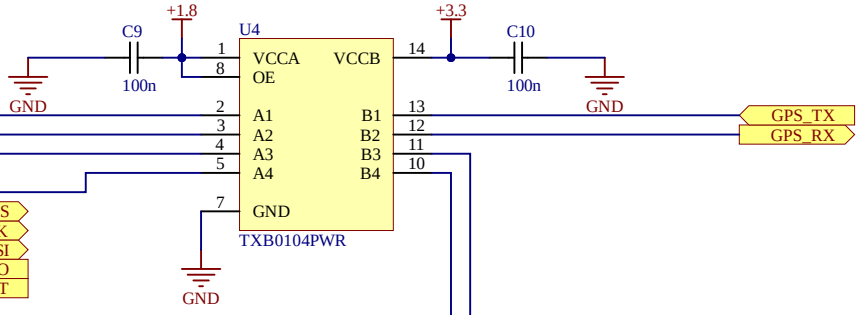
```
            "LiveEnable":true
        }
    },

    "Sensors":
    [
        {
            "NameLive":"TensionBatteryHV",
            "NameLog":"TensionBatteryHV_NL",
            "LiveEnable":true,
            "CanID":"0x12",
            "CanFrame":"B2:B1:X:X:X:X:X:X"
        },
        {
            "NameLive":"AmperageBatteryHV",
            "NameLog":"AmperageBatteryHV_NL",
            "LiveEnable":true,
            "CanID":"0x12",
            "CanFrame":"X:X:B2:B1:X:X:X:X"
        },
        {
            "NameLive":"TemperatureBatteryHV",
            "NameLog":"TemperatureBatteryHV_NL",
            "LiveEnable":true,
            "CanID":"0x12",
            "CanFrame":"X:X:X:X:B2:B1:X:X"
        },
        {
            "NameLive":"EnginePower",
            "NameLog":"EnginePower_NL",
            "LiveEnable":true,
            "CanID":"0x18",
            "CanFrame":"B2:B1:X:X:X:X:X:X"
        },
        {
            "NameLive":"EngineTemperature",
            "NameLog":"EngineTemperature_NL",
            "LiveEnable":true,
            "CanID":"0x22",
            "CanFrame":"X:X:B2:B1:X:X:X:X"
        }
    ]
}
```

Listing B.1: Configuration File Example

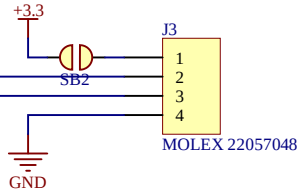
Nordic Dev Kit socket



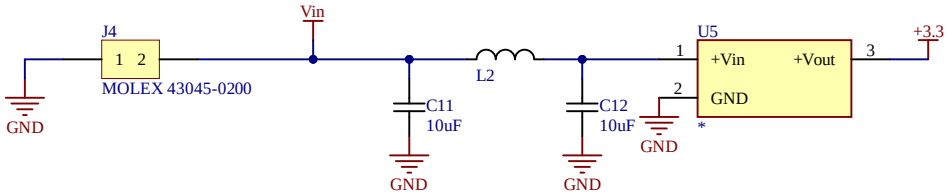
Level Shifter

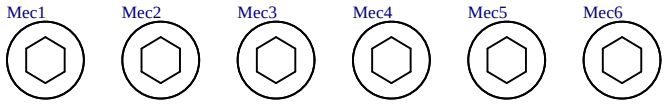


UART Interface

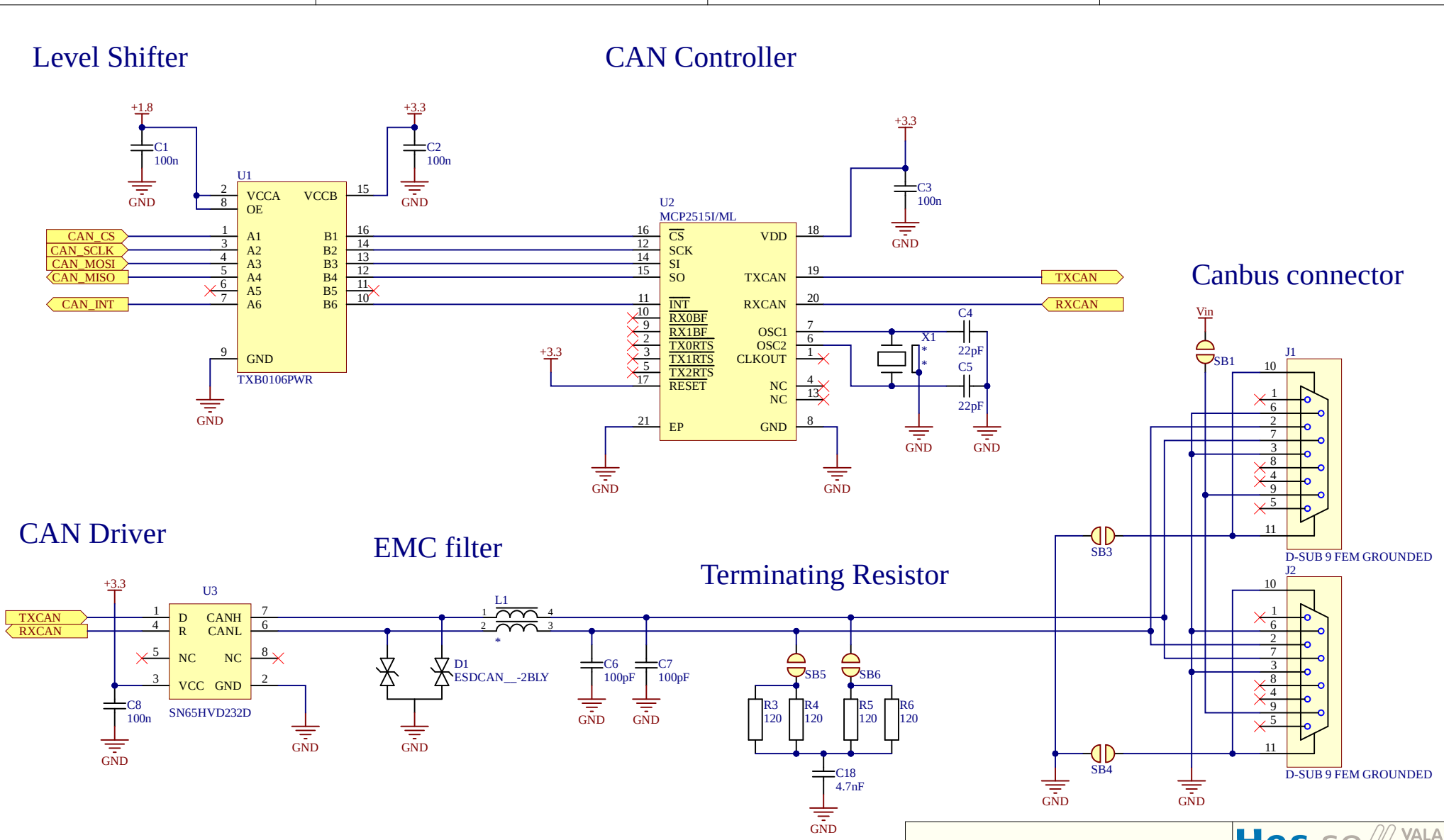


Power Supply



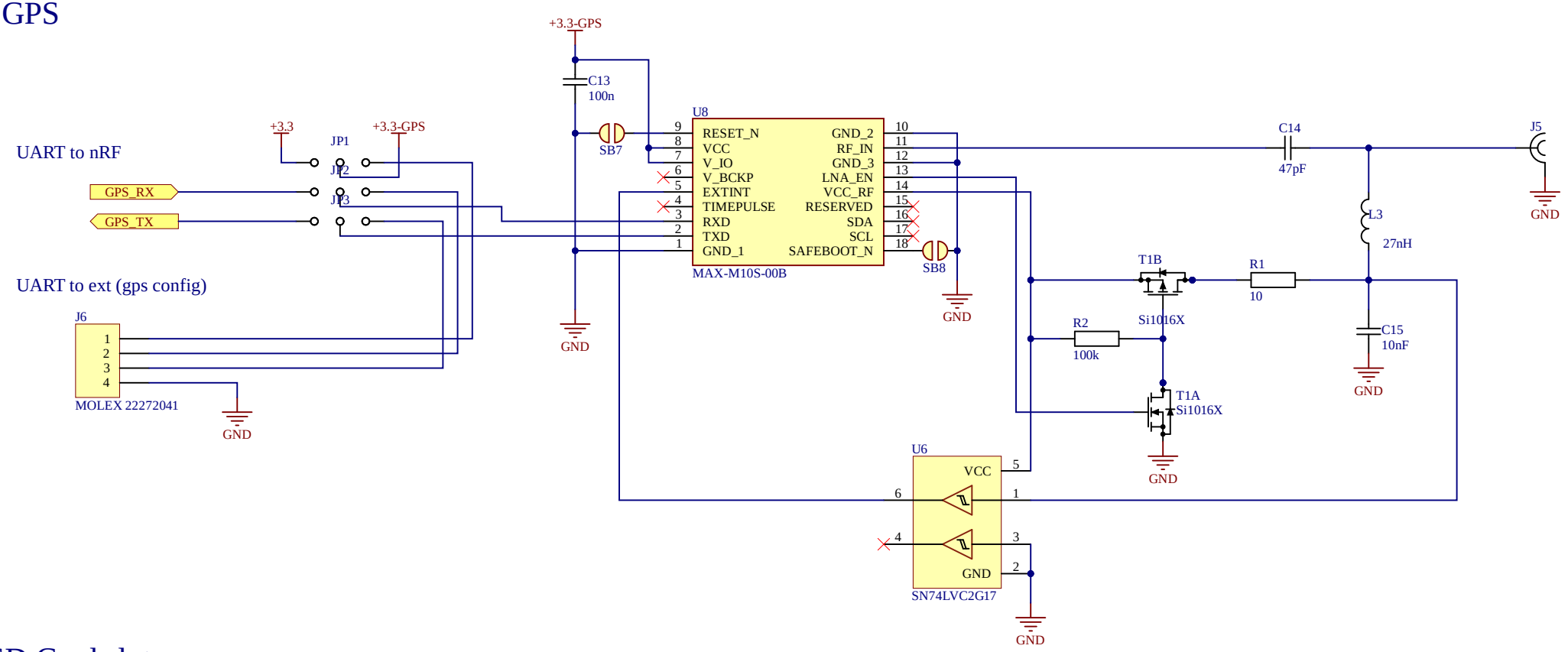


TelemetrySystem.PrjPcb		Hes·so VALAIS WALLIS
nRF7002-DK.SchDoc		Date : 02.06.2023
Revision : 1.1	Sheet 1 of 3	Design by : Sylvestre van Kappel
C:\Users\sylvestr.vankappe\Documents\Telemtetry-for-the-Formula-Student\Hardware\TelemetrySystemAltiu		

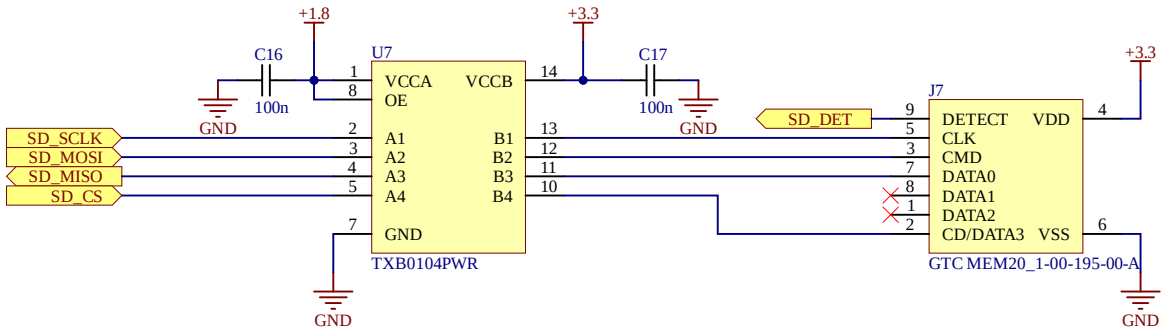


TelemetrySystem.PrjPcb		Hes·so VALAIS WALLIS
CanController.SchDoc		Date : 02.06.2023
Revision : 1.1	Sheet 2 of 3	Design by : Sylvestre van Kappel
C:\Users\sylvestr.vankappe\Documents\Telemtetry-for-the-Formula-Student\Hardware\TelemetrySystemAltiu		

GPS



SD Card slot



TelemetrySystem.PrjPcb		Hes·so VALAIS WALLIS
GPS-SD.SchDoc		Date : 02.06.2023
Revision : 1.1	Sheet 3 of 3	Design by : Sylvestre van Kappel
C:\Users\sylvestr.vankappe\Documents\Telemtetry-for-the-Formula-Student\Hardware\TelemetrySystemAltiu		

Supplier	Nbr.	Supplier's Number	Manufacturer	Description	unit price	total price	ordered
Bill of Material - Telemetry System					**Last update : 29.06.2023**		
Passive components							
Mouser	9	80-C603C104K5RAC3121	KEMET	Capacitor 1608 - 100 nF	0,07	0,63 CHF	from stock
Mouser	2	80-CBR06C220J5GAUTO	KEMET	Capacitor 1608 - 22 pF	0,14	0,28 CHF	from stock
Mouser	2	80-CBR06C101J5GAUTO	KEMET	Capacitor 1608 - 100 pF	0,46	0,92 CHF	from stock
Mouser	1	80-C0603X472K5GAUTO	KEMET	Capacitor 1608 - 4.7 nF	0,40	0,40 CHF	from stock
Mouser	1	810-C1005X7R1H103KBE	TDK	Capacitor 1005 - 10 nF	0,14	0,14 CHF	from stock
Mouser	1	810-C1005C0G1H470G	TDK	Capacitor 1005 - 47 pF	0,35	0,35 CHF	from stock
Mouser	2	80-C1206C106Z4VACTU	KEMET	Capacitor 3216 - 10 uF	0,41	0,82 CHF	from stock
Mouser	1	71-TNPW060310R0DEEA	Vishay	Resistor 1608 - 10 Ohm	0,44	0,44 CHF	from stock
Mouser	1	71-TNPW0603100KBEEA	Vishay	Resistor 1608 - 100 kOhm	0,50	0,50 CHF	from stock
Mouser	4	71-CRCW0603-120-E3	Vishay	Resistor 1608 - 120 Ohm	0,09	0,36 CHF	from stock
Mouser	1	871-B82793C104N201	EPCOS/TDK	Common mode choke	2,07	2,07 CHF	15.06.2023
Mouser	1	495-TCK-141	TRACO Power	EMC Choke	6,23	6,23 CHF	15.06.2023
Mouser	1	994-0402DC-27NXGRW	Coilcraft	RF Inductor, Wirewound, 1005 - 27 nH	1,41	1,41 CHF	15.06.2023
Mouser	1	581-CX3225SB16D0FLJ	Kyocera	Quartz 16 MHz	1,17	1,17 CHF	15.06.2023
Digikey	1	497-13262-1-ND	STMicroelectronics	Automotive dual-line TVS diode	0,41	0,41 CHF	15.06.2023
Integrated Circuits							
Digikey	1	296-23759-6-ND	Texas Instruments	6 channels bidirectionnal level shifter	1,50	1,50 CHF	15.06.2023
Digikey	2	296-21929-1-ND	Texas Instruments	4 channels bidirectionnal level shifter	1,03	2,06 CHF	15.06.2023
Mouser	1	781-SI1016X-T1-GE3	Vishay	Complementary N- and P-Channel 20 V (D-S) MOSFET	0,42	0,42 CHF	15.06.2023
Mouser	1	579-MCP2515-I/ML	Microchip	Stand-Alone CAN Controller with SPI Interface	2,50	2,50 CHF	15.06.2023
Mouser	1	595-SN65HVD232D	Texas Instruments	CAN Interface IC 3.3-V CAN TRANSCEIVER	2,97	2,97 CHF	15.06.2023
Mouser	1	495-TSR-1-2433	TRACO Power	Non-Isolated DC/DC Converters 3.3v	5,75	5,75 CHF	15.06.2023
Mouser	1	377-MAX-M10S-00B	Ublox	Standard precision GNSS module	20,67	20,67 CHF	15.06.2023
Mouser	1	595-SN74LVC2G07DCKR	Texas Instruments	Buffers & Line Drivers Dual w/ Open-Drain	0,43	0,43 CHF	15.06.2023

Connectors							
Mouser	1	538-22-05-7048	Molex	Headers & Wire Housings 4C RA F/L HEADER	1,17	1,17 CHF	from stock
Mouser	1	538-22-27-2041	Molex	Headers & Wire Housings 4 POS HEADER	0,39	0,39 CHF	from stock
Mouser	1	538-43045-0200	Molex	Headers & Wire Housings 2 CKT R/A HEADER	1,10	1,10 CHF	from stock
Mouser	1	538-73251-1150	Molex	SMA connector	4,10	4,10 CHF	15.06.2023
Mouser	1	640-MEM20510019500A	GCT	Memory Card Connectors Micro SD	1,12	1,12 CHF	15.06.2023
Digikey	2	A132510-ND	TE Connectivity	CONN D-SUB PLUG 9POS R/A SLDR	1,92	3,84 CHF	from stock
Miscellaneous							
Digikey	1	1568-1504-ND	Sparkfun	USB UART Serial breakout	14,85	14,85 CHF	15.06.2023
Mouser	1	949-NRF7002-DK	Nordic Semi	nRF-7002 DK	57,44	57,44 CHF	02.06.2023
Mouser	1	377-ANN-MB-00	Ublox	Antennas Multi-band active GNSS antenna - SMA	55,84	55,84 CHF	15.06.2023
Digitec	1	23079748	Asus	Wifi Router - Asus RT-AX86U Pro	247,00	247,00 CHF	15.06.2023
Distrelec	1	301-27-198	EAO	Light button	18,95	18,95 CHF	29.06.2023
Distrelec	1	302-20-253	Taoglas	Dual Band wifi antenna	15,64	15,64 CHF	15.06.2023
Digikey	1	ADP-SMAM-RPSF-G-ND	Linx Technologies	SMA to RP-SMA adapter	6,72	6,72 CHF	15.06.2023
Distrelec	1	301-31-595	Nedis	Antenna cable SMA 2m	10,29	10,29 CHF	15.06.2023
Digikey	1	490-4982-ND	Murata	Microwave Coaxial adapter	27,70	27,70 CHF	15.06.2023
PCB							
Eurocircuit	1	-	Eurocircuit	4 Layer PCB	130,00	130,00 CHF	15.06.2023

Total :	**648,58 CHF**

E | Busmaster Example Script

```
/* This file is generated by BUSMASTER */
/* VERSION [1.2] */
/* BUSMASTER VERSION [3.2.2] */
/* PROTOCOL [CAN] */

/* Start BUSMASTER include header */
#include <Windows.h>
#include <CANIncludes.h>
/* End BUSMASTER include header */


/* Start BUSMASTER global variable */
STCAN_MSG tx;
int voltageHVS;
int currentHVS;
int tempHVBat;
/* End BUSMASTER global variable */


/* Start BUSMASTER Function Prototype  */
GCC_EXTERN void GCC_EXPORT OnTimer_bmsSend_100( );
/* End BUSMASTER Function Prototype  */

/* Start BUSMASTER Function Wrapper Prototype  */
/* End BUSMASTER Function Wrapper Prototype  */


/* Start BUSMASTER generated function - OnTimer_bmsSend_100 */
void OnTimer_bmsSend_100( )
{
    /* TODO */
    voltageHVS= rand()%5 + 548;
    currentHVS= rand()%5 + 80;
    tempHVBat= rand()%5 + 50;
    tx.id=0x12;
    tx.dlc=8;
    tx.data[0]=voltageHVS >> 8;
    tx.data[1]=voltageHVS;
    tx.data[2]=currentHVS >> 8;
    tx.data[3]=currentHVS;
    tx.data[4]=tempHVBat >> 8;
    tx.data[5]=tempHVBat;
    tx.data[6]=0;
    tx.data[7]=0;
    SendMsg(tx);
}/* End BUSMASTER generated function - OnTimer_bmsSend_100 */
```

Listing E.1: Busmaster BMS Script

F | Sustainability Analysis

This analysis aims to measure the project's impact concerning the 17 global objectives for sustainable development established by the OECD. The Organisation for Economic Co-operation and Development (OECD) is an international organization that works to build better policies for better lives. Its goal is to shape policies that foster prosperity, equality, opportunity and well-being for all (www.oecd.org).

Figure F.1: OECD Goals

This project can be analyzed on the two following objectives:

- 9 : Industry, Innovation and Infrastructure
- 12 : Responsible Consumption and Production

F.1 Industry, Innovation and Infrastructure

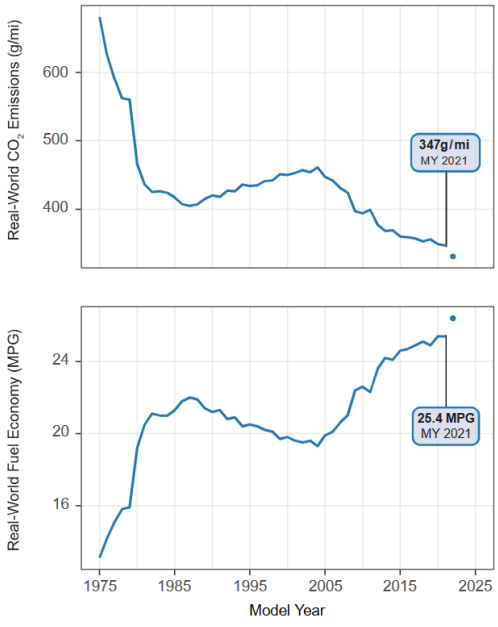


Figure F.2: Estimated Real-World Fuel Economy and CO_2 Emissions

This project is in line with innovation and development of sustainable vehicles. Telemetry systems do not have a direct impact on the car's performance, but they allow to collect many data during the development of vehicles, which are then used to improve the performance and the efficiency.
This chart is taken from the 2022 Automotive Trends report from the United States Environmental Protection Agency. It shows the average CO_2 emission rate for all new vehicles. It is clearly visible that the CO_2 emissions and fuel consumption of vehicles were reduced since 1975. In this period, the use of electronics in vehicles has increased. Electronics, including telemetry systems, have contributed to the improvement of the fuel efficiency and reduction of emissions of cars.

F.2 Responsible Consumption and Production

The telemetry system itself only consumes a very low amount of energy. Its consumption never exceeds one Watt, which is negligible compared to the car's consumption. However, car racing can not be considered as sustainable, even if it is electric. A racing electric vehicle does not emit CO_2, but it still consumes a lot of energy, which needs to be produced and will emit CO_2. Moreover, the primary source of pollution is not the race itself but the travelling to the race circuits. For example, the Valais-Wallis Racing Team travelled to 4 races during the 2023 season and had to travel to Switzerland, Austria, Spain, and Croatia with many people and equipment.

Therefore, Formula Student cannot be considered as responsible and sustainable in line with objective 9 of the sustainable development goals from the OECD.

F.3 Conclusion

To conclude this analysis, there are two ways to see this project.

The first is the telemetry system itself, a great device that enables increasing the efficiency of vehicles, which clearly enters into sustainability development goals.

The second is Formula Student, which is not sustainable regarding responsible energy consumption.

Bibliography

[1] *FS-Rules_2023_v1.1.pdf*. URL: https://www.formulastudent.de/fileadmin/user_upload/all/2023/rules/FS-Rules_2023_v1.1.pdf (visited on 05/26/2023).

[2] *Valais Wallis Racing Team*. URL: https://www.vrt-fs.ch/ (visited on 05/26/2023).

[3] @eteclan_fotomotor. *VRT Car*. Formula Student Spain. URL: https://mediaformulastudentspain.smugmug.com/FSS-2023/Saturday-12/i-GmpWz5c (visited on 08/15/2023).

[4] *FS Alpe Adria - Home*. URL: https://fs-alpeadria.com/ (visited on 06/15/2023).

[5] *RFM69HC 20dBm Programmable 315-915Mhz RF Transceiver Module_ Sub 1G Programmable 315-915Mhz RF Transceiver Module | HOPERF*. URL: https://www.hoperf.com/modules/rf_transceiver/RFM69HCW.html (visited on 05/26/2023).

[6] *Friis transmission equation*. In: *Wikipedia*. Page Version ID: 1157035076. May 25, 2023. URL: https://en.wikipedia.org/w/index.php?title=Friis_transmission_equation&oldid=1157035076 (visited on 05/26/2023).

[7] *Digi XBee® Wi-Fi*. URL: https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-rf-modules/xbee-wi-fi (visited on 08/11/2023).

[8] *IEEE 802.11*. In: *Wikipédia*. Page Version ID: 204512477. May 22, 2023. URL: https://fr.wikipedia.org/w/index.php?title=IEEE_802.11&oldid=204512477 (visited on 08/16/2023).

[9] Nordic Semi. *nRF7002 Product Brief*. URL: https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF7002-IC_Product-Brief-v1.0.pdf (visited on 06/16/2023).

[10] *Intel® Wi-Fi 6 AX201 (Gig+) - Caractéristiques du produit*. Intel. URL: https://www.intel.fr/content/www/fr/fr/products/sku/130293/intel-wifi-6-ax201-gig/specifications.html (visited on 05/26/2023).

[11] *iPerf - The TCP, UDP and SCTP network bandwidth measurement tool*. URL: https://iperf.fr/ (visited on 05/26/2023).

[12] Nordic Semi. *nRF7002 Product Specification*. URL: https://infocenter.nordicsemi.com/pdf/nRF7002_OPS_v0.7.pdf (visited on 06/01/2023).

[13] Nordic Semi. *nRF5340 Product Specification*. URL: https://infocenter.nordicsemi.com/pdf/nRF5340_PS_v1.3.pdf (visited on 06/07/2023).

[14] U-blox. *SAM-M10Q ProductSummary*. URL: https://content.u-blox.com/sites/default/files/documents/SAM-M10Q_ProductSummary_UBX-22004868.pdf (visited on 06/17/2023).

[15] U-blox. *MAX-M10 Product Summary*. URL: https://content.u-blox.com/sites/default/files/MAX-M10_ProductSummary_UBX-20017987.pdf (visited on 06/17/2023).

[16] U-blox. *UBX-M9140 ProductSummary*. URL: https://content.u-blox.com/sites/default/files/UBX-M9140_ProductSummary_UBX-19027230.pdf (visited on 06/17/2023).

[17] U-blox. "MAX-M10S Integration manual". In: (). URL: https://content.u-blox.com/sites/default/files/MAX-M10S_IntegrationManual_UBX-20053088.pdf.

[18] U-blox. *u-center*. u-blox. June 25, 2015. URL: https://www.u-blox.com/en/product/u-center (visited on 08/16/2023).

[19] Microchip Technology Inc. *MCP2515 Datasheet*. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf (visited on 05/31/2023).

[20] Texas Instruments. *SN65HVD23x Datasheet*. URL: https://www.ti.com/lit/ds/symlink/sn65hvd232.pdf?ts=1687982856942&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FSN65HVD232 (visited on 06/29/2023).

[21] Microchip Technology Inc. *Controller Area Network Physical Layer Protection*. Oct. 2006. URL: https://www.microchip.com/stellent/groups/sitecomm_sg/documents/training_tutorials/en528477.pdf (visited on 06/07/2023).

[22] Texas Intruments. "Common-Mode Chokes in CAN Networks: Source of Unexpected Transients". In: (2008).

[23] Traco Power. *TSR1 Datasheet*. URL: https://www.tracopower.com/sites/default/files/products/datasheets/tsr1_datasheet.pdf (visited on 08/16/2023).

[24] Traco Power. *TSR1 EMI Consideration*.

[25] Texas Instruments. *TXB0104 Datasheet*. URL: https://www.ti.com/lit/ds/symlink/txb0104-q1.pdf?ts=1685708331679&ref_url=https%253A%252F%252Fwww.google.com%252F (visited on 06/02/2023).

[26] Taoglas. *GW.71.5153 Specification*. URL: https://media.distrelec.com/Web/Downloads/_t/ds/GW.71.5153_eng_tds.pdf (visited on 08/07/2023).

[27] *Disk Access — Zephyr Project Documentation*. URL: https://docs.zephyrproject.org/latest/services/storage/disk/access.html (visited on 08/04/2023).

[28] *File Systems — Zephyr Project Documentation*. URL: https://docs.zephyrproject.org/latest/services/file_system/index.html (visited on 07/28/2023).

[29] *Zephyr API Documentation: JSON*. URL: https://docs.zephyrproject.org/apidoc/latest/group__json.html (visited on 07/31/2023).

[30] *CAN Controller — Zephyr Project Documentation*. URL: https://docs.zephyrproject.org/latest/hardware/peripherals/canbus/controller.html (visited on 07/31/2023).

[31] SiRF Technology Inc. “NMEA Reference Manual”. In: (2007). URL: https://d1.amobbs.com/bbs_upload782111/files_10/ourdev_390384.pdf.

[32] *Timers — Zephyr Project Documentation*. URL: https://docs.zephyrproject.org/latest/kernel/services/timing/timers.html (visited on 07/26/2023).

[33] *Workqueue Threads — Zephyr Project Documentation*. URL: https://docs.zephyrproject.org/latest/kernel/services/threads/workqueue.html#work-item-lifecycle (visited on 07/26/2023).

[34] *Non-Volatile Storage (NVS) — Zephyr Project Documentation*. URL: https://docs.zephyrproject.org/latest/services/storage/nvs/nvs.html (visited on 08/03/2023).

[35] *Mutexes — Zephyr Project Documentation*. URL: https://docs.zephyrproject.org/latest/kernel/services/synchronization/mutexes.html (visited on 07/28/2023).

[36] *BSD Sockets — Zephyr Project Documentation*. URL: https://docs.zephyrproject.org/latest/connectivity/networking/api/sockets.html (visited on 07/26/2023).

[37] *Memory Heaps — Zephyr Project Documentation*. URL: https://docs.zephyrproject.org/latest/kernel/memory_management/heap.html (visited on 07/26/2023).

[38] *Wi-Fi Station — nRF Connect SDK 2.2.99-dev3 documentation*. URL: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/2.2.99-dev3/nrf/samples/wifi/sta/README.html (visited on 07/26/2023).

[39] *Zephyr API Documentation: Network Management*. URL: https://docs.zephyrproject.org/apidoc/latest/group__net__mgmt.html (visited on 07/26/2023).

[40] *BUSMASTER*. URL: https://rbei-etas.github.io/busmaster/ (visited on 08/14/2023).

[41] *MHS Tiny-CAN I-XL*. URL: https://www.mhs-elektronik.de/index.php?module=artikel&action=artikel&id=16 (visited on 08/14/2023).

Acronyms

BMS Battery Management System. 15

CAN Controller Area Network. 3, 15, 20

CSV Comma-Separated Values. 12

ECU Electronic Controlled Unit. 3

GNSS Global Navigation Satellite Systems. 17

I2C Inter-Integrated Circuit. 17

IoT Internet of Things. 3

IP Internet Protocol. 6

JSON JavaScript Object Notation. 12

PCB Printed Cirtuit Board. 23

QSPI Quad SPI. 7, 15

RF Radio Frequency. 5

RTOS Real-Time Operating System. 16

SoC System on Chip. 16, 20

SPI Serial Peripheral Interface. 6, 7, 15, 20, 22, 81

UART Universal Asynchronous Receiver Transmitter. 6, 15–17

UDP User Datagram Protocol. 8

WPA Wi-Fi Protected Access. 6, 7