Summary

[1 Introduction 2](#_Toc141278617)

[1.1 Formula Student 2](#_Toc141278618)

[1.2 Valais-Wallis Racing team 2](#_Toc141278619)

[1.3 Objectives 2](#_Toc141278620)

[1.4 State of the art 3](#_Toc141278621)

[1.5 Timeline -> à supprimer ? 4](#_Toc141278622)

[2 Problem Description 5](#_Toc141278623)

[3 Transmission technology 6](#_Toc141278624)

[3.1 433 MHz RF 6](#_Toc141278625)

[3.2 Wi-Fi 7](#_Toc141278626)

[3.2.1 Tests 8](#_Toc141278627)

[3.3 Conclusion 9](#_Toc141278628)

[4 Technologies 10](#_Toc141278629)

[5 Hardware development 11](#_Toc141278630)

[5.1 Bloc diagram 11](#_Toc141278631)

[5.2 Main SoC 12](#_Toc141278632)

[5.3 GPS Module 13](#_Toc141278633)

[5.4 CAN Bus Controller 15](#_Toc141278634)

[5.5 SD card 18](#_Toc141278635)

[5.6 Power supply 18](#_Toc141278636)

[5.7 Level shifters 18](#_Toc141278637)

[6 Software development 19](#_Toc141278638)

[6.1 Configuration file 19](#_Toc141278639)

[6.2 CAN Controller 20](#_Toc141278640)

[6.3 GPS Controller 20](#_Toc141278641)

[6.4 Data Logger 24](#_Toc141278642)

[6.5 Data Sender 24](#_Toc141278643)

[6.6 UDP Client 26](#_Toc141278644)

[6.7 Wi-Fi Station 27](#_Toc141278645)

[7 Bibliography 27](#_Toc141278646)

# Introduction

## 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 & 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.

Une image contenant plein air, route, véhicule, sport mécanique

Description générée automatiquement

Figure 1: Formula Student Endurance race

## 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. The current and first car does not implement telemetry. This project deals with the development of the team's first telemetry system.

## 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 read directly from the side of the track. With such a system, the data from all the sensors is 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 [3] 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.

Une image contenant Photographie aérienne, ligne, Vue plongeante, plein air

Description générée automatiquement

Figure 2: Alpe Adria Circuit

## 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 (Control Area Network Bus) is a communication protocol that enables reliable and efficient data transfer between ECUs (Electronic Control Units) 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 (Internet of Things) 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.

## Timeline -> à supprimer ?

Une image contenant texte, capture d’écran, ligne, nombre

Description générée automatiquement

This schedule shows the project's progress over time. The first two weeks were devoted to the transmission technology study. The next three weeks were dedicated to developing the electronic board. The fourth week in the "Hardware" section corresponds to the time to manufacture the electronic board. The remaining time was devoted to programming the telemetry system. A detailed schedule is available in the appendix.

# Problem Description

The car electrical

Expliquer bus can bms capteurs …. Comment est la voiture – unidirectionnel

The telemetry device must communicate via the car's CAN bus with the sensors, the BMS, and the Inverter. A GPS must be integrated into the system. A transmission technology must be chosen to directly visualize the data, and the system must also store the data on a microSD card.

Une image contenant texte, diagramme, capture d’écran, ligne

Description générée automatiquement

# 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.

## 433 MHz RF

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 [4] module from HopeRF is one of the best options for this use case. It is available in 433 MHz and 868 MHz.



Figure 3: 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 [5] (with omnidirectional antennas → Gantenna = 0 dbi):

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.

## 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 method shares the band between the stations to avoid collisions between frames. développer

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.

Plusieurs choix

The best option for Wi-Fi communication is the nRF7002 [6] chip from Nordic Semiconductors. It is a dual-band Wi-Fi module with an SPI/QSPI interface to relate to a host SoC.

Une image contenant texte

Description générée automatiquement

Figure 4: nRF7002 chip

On the 2.4 GHz band at 1 Mbps, the maximum output power is 21 dBm, and the input sensitivity is -98.6 dBm.

The theoretical range can be calculated with the Friis transmission equation [5] (with omnidirectional antennas → Gantenna = 0 dbi):

For the 2.4 GHz band, the received power is -70 dBm at 300 m, which is close to the input sensitivity of a traditional router.

As these equations give theoretical results, tests must be carried out to determine whether the range will be sufficient in practice.

### 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 [7], which has similar performances (in terms of range) as the nRF7002 [6]. 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 [8], using the UDP protocol at 1 Mbps.

The laptop and the router could communicate successfully at 400 meters distance. The minimal objective of 300 meters is reached. It has not been demonstrated that the system also works at 500 meters. Should a better range be necessary, the performances can be improved by using sector antennas.

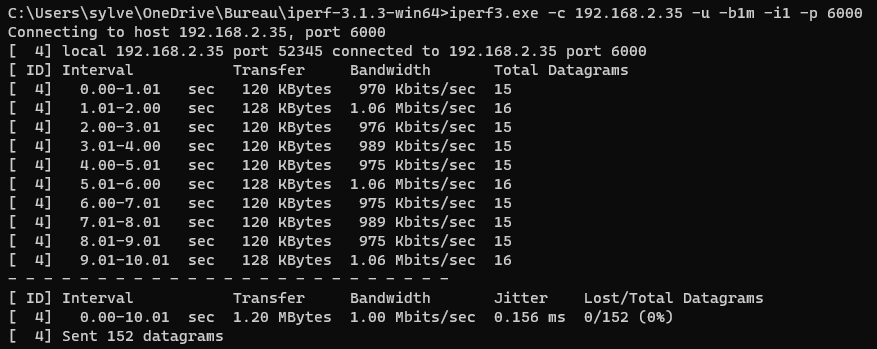


Figure 5: iPerf test - Client side

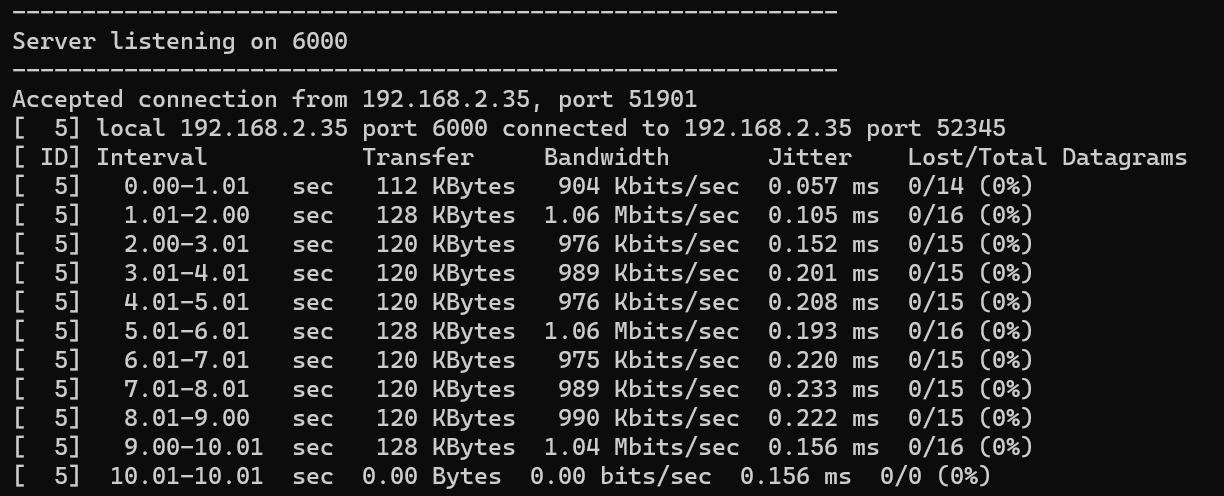


Figure 6: iPerf test - Server side

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

## Conclusion

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

|  |  |  |
| --- | --- | --- |
|  | **Wi-Fi** | **433 MHz RF** |
| 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 1: Comparison between Wi-Fi and 433 MHz RF

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 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 [6] module.

# Technologies

CANBUS

GPS

SPI

UART

# Hardware development

## Bloc diagram

Une image contenant texte, diagramme, Plan, capture d’écran

Description générée automatiquement

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 last peripheral is a UART interface. 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).

## Main SoC

An SoC (System on Chip) 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 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 is a high-performance SoC from Nordic Semiconductors. It integrates two Arm Cortex-M33 processors, Bluetooth, high-speed SPI/QSPI, and more great 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 system will use the development kit with a custom PCB directly plugged into it to avoid spending too much time on hardware development.

Une image contenant Composant électronique, Composant de circuit, Ingénierie électronique, Composant de circuit passif

Description générée automatiquement

Figure 7: Development Kit

## GPS Module

Overview

The telemetry system needs to include a GNSS (Global Navigation Satellite Systems) 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 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.

Une image contenant Appareils électroniques, texte, Composant électronique, Composant de circuit

Description générée automatiquement

Figure 8: SAM-M10Q module

The second option is the MAX-M10S 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.



Une image contenant texte, Rectangle

Description générée automatiquement

Figure 9: MAX-M10S module

The last considered option is the UBX-M9140. 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.

Une image contenant texte, Rectangle

Description générée automatiquement

Figure 10: UBX-M9140 chip

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) can be 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 11: GPS Antenna

Schematics

This schematic shows the circuitry of the GPS module. The module is connected to the nRF5340 using a UART communication.

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 LNA\_EN pin provides the activation signal.

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 configuring the module with the u-center2 software.

Shorting SB7 permits to generate a hardware reset. SB8 allows to enter in safeboot mode.

Une image contenant texte, diagramme, Plan, schématique

Description générée automatiquement

Figure 12: GPS schematic

The TX and RX pins are connected to the nRF 7002-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).

Une image contenant texte, diagramme, Plan, ligne

Description générée automatiquement

Figure 13: Nordic Dev Kit socket and GPS level shifter schematic

## CAN Bus Controller

Overview

The telemetry system communicates with the sensors by the CAN bus. For this purpose, a CAN Controller must be selected. The MCP2515 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 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.

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.

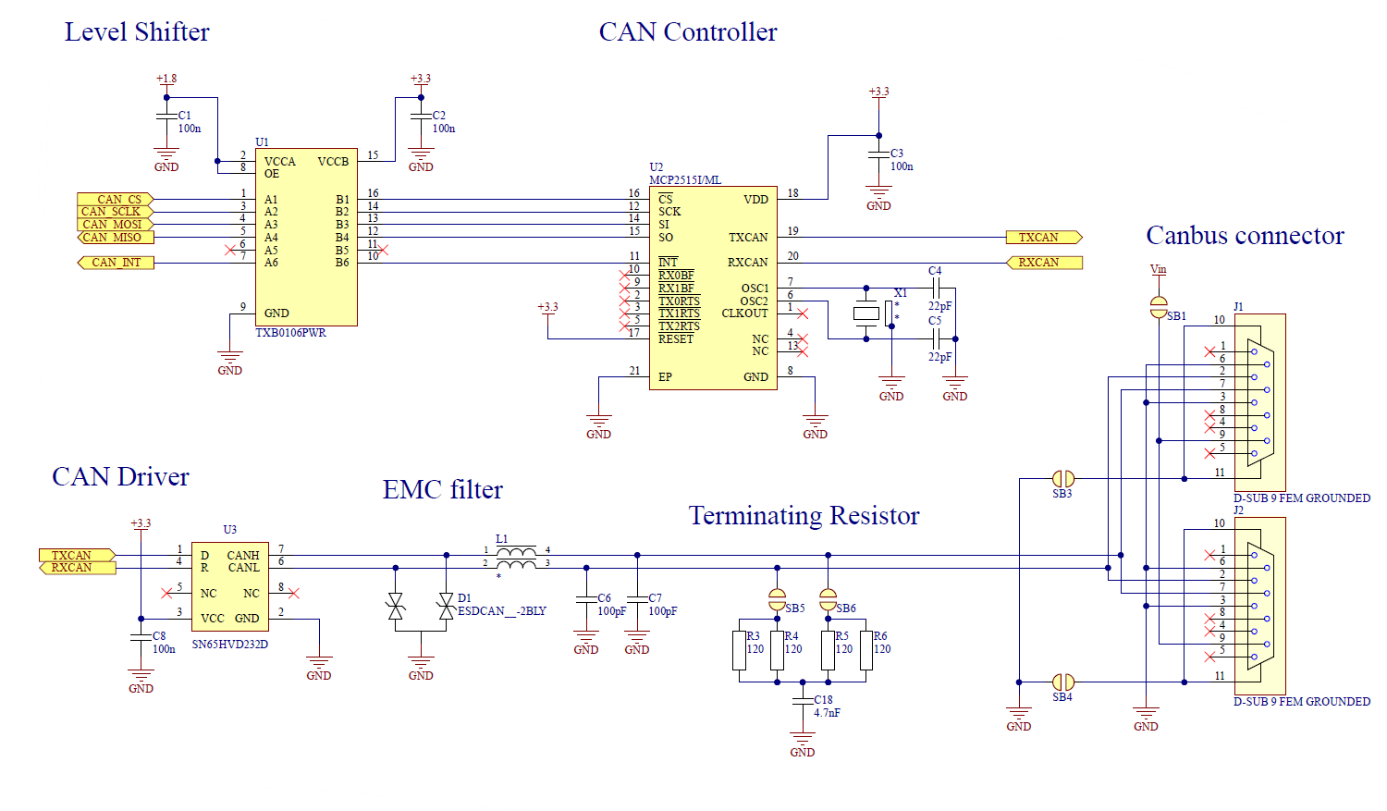


Figure 14: Can controller schematic

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).

Two tiny capacitors help to filter high frequencies.

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.

The design also includes a terminating resistor for the CAN Bus. Shortening the two solder bridges (SB5 and SB6) can connect it. 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.

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** |
| DB-9 Connector Pinout - Decisive Tactics, Inc. | DB-9 Connector Pinout - Decisive Tactics, Inc. | 1 | - |
| 2 | CAN L |
| 3 | GND |
| 4 | - |
| 5 | - |
| 6 | GND |
| 7 | CAN H |
| 8 | - |
| 9 | Power Supply |

Table 2 : 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 also could 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.

## 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.

Une image contenant texte, diagramme, ligne, capture d’écran

Description générée automatiquement

Figure 15: SD card slot schematic

## Power supply

The LVS (Low Voltage System) 12 V battery powers the telemetry system. 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) provides the 3.3 V power supply. The design implements the input filter described in the component application note.

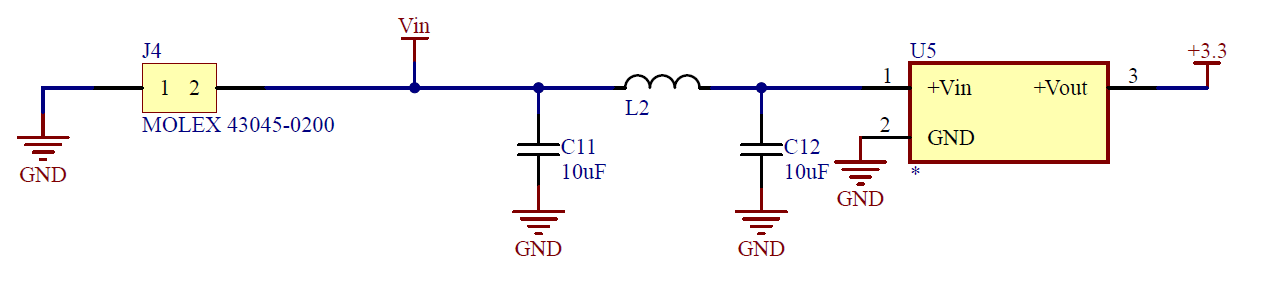


Figure 16: Power supply schematic

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.

## 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. 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.

# Software development

Zephyr

The software is implemented on the application processor of the nRF5340. It is separated into multiple tasks working in parallel. The *main*function starts the button manager task and led manager task first. Then the configuration file is read, and all the tasks are started if the read succeeds.

The software implements the following tasks :

* Wi-Fi Stationing
* UDP Client
* Data Sender
* Data Logger
* GPS Controller
* CAN Controller

Memory management - buffers

Data flow diagram

## Configuration file

The first thing the program needs to do is to read the configuration file. If this task fails, the program does not start.

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 measurement send rate in messages per second.
* Logging measurement rate in measurement per second.
* GPS parameters
* Sensor parameters

First, the SD card is mounted, and the base directory is opened. Then a loop search for the config file. Once the file is found, it is opened and read. The JSON string is parsed with the JSON library provided by the Zephyr environment, and the data are registered in a struct.

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

The last task of the *config\_read* function is to initialize the sensor buffer and GPS buffer.

led

## CAN Controller

## 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 [9].

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 17: 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.

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Figure 18: GPS Vehicle tracker configuration

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 satellite-connected is sent every 5 seconds. The other messages are not interesting for the telemetry system, so they are not sent.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 19: GPS MSGOUT configuration

The messages are sent to the nRF5340 using a UART bus. The GPS module uses the NMEA standard [10] 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:



Figure 20 : 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.

Une image contenant texte, diagramme, capture d’écran, Police

Description générée automatiquement

Figure 21: 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).

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:

To work only with integer variables, the formula has been slightly changed. A coordinate is stored in 2 variables. One for the characteristic[[1]](#footnote-2) and one for the mantissa[[2]](#footnote-3). Only the degree part is taken for the characteristic during the parsing operation. For the mantissa, 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 schema shows how the NMEA coordinates are converted into decimal coordinates :

Une image contenant capture d’écran, texte, conception

Description générée automatiquement

Figure 22: Conversion of NMEA to decimal coordinates

## Data Logger

The Data logger task records the measurement from the sensors on the SD card. 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 test section.

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 3: 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, if the logs are not recording, the SD card is mounted, and the file is created and opened. If the logs are recording, the file is closed, and the SD card is unmounted.

The following diagram shows how the Data Logger works:

Une image contenant texte, diagramme, Plan, Dessin technique

Description générée automatiquement

Figure 23: Data Logger Activity Diagram

The code uses the File System library [11] 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 [12].

In the timer interruption and the button callback method, works [13] that call the functions are submitted. Using a work objects [13] permits spending as little time as possible in the interruption to avoid interrupting high-priority processes.

addon flash name

When the 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.

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 [14] 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.

## 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 most two sends per second. The system is technically capable of sending data at higher frequencies, but the live transmission is not meant for that. Higher frequencies will negatively impact the system's general performance, and the receiving software will not necessarily be able to deal with high frequencies. 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. 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 send 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:



Figure 24: Live Data Structure

The following diagram shows how the Data Sender works:

Une image contenant texte, diagramme, croquis, dessin

Description générée automatiquement

Figure 25: 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 [12].

In the timer interruption, a work [13] that calls the periodic function is submitted. Using a work object [13] 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 [14] 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.

## 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 software. 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:

Une image contenant texte, diagramme, Plan, Dessin technique

Description générée automatiquement

Figure 26: UDP Client Activity diagram

The code uses the socket.h library [15] 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 *Figure 23* structure. The Data Sender dynamically allocates memory on a memory heap [16] 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 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.

## Wi-Fi Station

The Wi-Fi Station task is a modified version of the Wi-Fi Station example from Nordic Semiconductors [17]. 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.

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. Else, the system tries to connect to the redundancy router. If the connection succeeds with the redundancy router, the thread goes to sleep mode. Else, 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:

Une image contenant texte, diagramme, capture d’écran, Parallèle

Description générée automatiquement

Figure 27: 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 [18] provided by the Zephyr environment. 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, the context struct is reset, and the system retries to connect.

# Sources

[1] ‘FS-Rules\_2023\_v1.1.pdf’. Accessed: May 26, 2023. [Online]. Available: https://www.formulastudent.de/fileadmin/user\_upload/all/2023/rules/FS-Rules\_2023\_v1.1.pdf

[2] ‘Valais Wallis Racing Team’. https://www.vrt-fs.ch/ (accessed May 26, 2023).

[3] ‘FS Alpe Adria - Home’. https://fs-alpeadria.com/ (accessed Jun. 15, 2023).

[4] ‘RFM69HC 20dBm Programmable 315-915Mhz RF Transceiver Module\_Sub 1G Programmable 315-915Mhz RF Transceiver Module | HOPERF’. https://www.hoperf.com/modules/rf\_transceiver/RFM69HCW.html (accessed May 26, 2023).

[5] ‘Friis transmission equation’, *Wikipedia*. May 25, 2023. Accessed: May 26, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Friis\_transmission\_equation&oldid=1157035076

[6] ‘nRF7002 - Low-power, advanced security, seamless coexistence’. https://www.nordicsemi.com/Products/nRF7002 (accessed May 26, 2023).

[7] ‘Intel® Wi-Fi 6 AX201 (Gig+) - Caractéristiques du produit’, *Intel*. https://www.intel.fr/content/www/fr/fr/products/sku/130293/intel-wifi-6-ax201-gig/specifications.html (accessed May 26, 2023).

[8] ‘iPerf - The TCP, UDP and SCTP network bandwidth measurement tool’. https://iperf.fr/ (accessed May 26, 2023).

[9] ‘MAX-M10S Integration manual’.

[10] S. Jose, ‘NMEA Reference Manual’, 2007.

[11] ‘File Systems — Zephyr Project Documentation’. https://docs.zephyrproject.org/latest/services/file\_system/index.html (accessed Jul. 28, 2023).

[12] ‘Timers — Zephyr Project Documentation’. https://docs.zephyrproject.org/latest/kernel/services/timing/timers.html (accessed Jul. 26, 2023).

[13] ‘Workqueue Threads — Zephyr Project Documentation’. https://docs.zephyrproject.org/latest/kernel/services/threads/workqueue.html#work-item-lifecycle (accessed Jul. 26, 2023).

[14] ‘Mutexes — Zephyr Project Documentation’. https://docs.zephyrproject.org/latest/kernel/services/synchronization/mutexes.html (accessed Jul. 28, 2023).

[15] ‘BSD Sockets — Zephyr Project Documentation’. https://docs.zephyrproject.org/latest/connectivity/networking/api/sockets.html (accessed Jul. 26, 2023).

[16] ‘Memory Heaps — Zephyr Project Documentation’. https://docs.zephyrproject.org/latest/kernel/memory\_management/heap.html (accessed Jul. 26, 2023).

[17] ‘Wi-Fi: Station — nRF Connect SDK 2.2.99-dev3 documentation’. https://developer.nordicsemi.com/nRF\_Connect\_SDK/doc/2.2.99-dev3/nrf/samples/wifi/sta/README.html (accessed Jul. 26, 2023).

[18] ‘Zephyr API Documentation: Network Management’. https://docs.zephyrproject.org/apidoc/latest/group\_\_net\_\_mgmt.html (accessed Jul. 26, 2023).

1. Characteristic: Integer part of a decimal number [↑](#footnote-ref-2)
2. Mantissa: Fractional part of a decimal number [↑](#footnote-ref-3)