Automotive data acquisition system - FST

David Rua Copeto
Instituto Superior Técnico
Technical University of Lisbon
Lisbon, Portugal

September 28, 2009

Abstract - This thesis addresses the design, implementation and validation of a telemetry system for a Formula Student prototype vehicle, having in mind an existing CAN-bus network of sensors in it. To achieve this, the proposed system is divided in two blocks: a mobile station and a base station. The first, is placed on the vehicle and is connected to its sensors through a CAN-bus. This mobile station has the function of recording locally the data generated by the bus activity and also of transferring this data wirelessly to the base station located off-track. The second has the function of picking up the data sent through the wireless link and presenting it in an attractive and comprehensible way to the user. Besides the "online" operation, the base station is also capable of presenting data from previous logging sessions for analysis.

Given the type of vehicle (and competition) this work is applied to, certain requirements are set in terms of both system capabilities and project management. In one hand the system must be able to sustain a harsh environment, namely vibration, heat, liquids and electromagnetic interference, and on the other hand it must be lightweight, cheap and user-friendly.

The developed telemetry system was successfully deployed on a practice track, being able to reliably record CAN-bus data of a variable number, sample rate and type of sensors. The performance of the radio link between stations was also evaluated both in controlled and real environments in order to understand the limits of the system. The results show a maximum wireless data rate of 4.28 kB/s for a typical vehicle position on track.

Keywords: Telemetry, CAN-bus, Zigbee, Formula Student, automotive data logger, graphical user interface.

1 Introduction

A great deal of time, effort and expense goes into the construction of a prototype. As it represents the tangible result of the breakthroughs accomplished after many hours of research, it is of major importance to have the best enlightenment possible on it’s performance and maintenance conditions, and in this way nowadays sensor networks and data gathering systems are a mandatory part of any fairly complex machine.

An important field where data loggers have been used extensively is the automotive area. Specially in competition, data logging plays an important role, allowing a much easier tuning of vehicles, monitoring of critical parts (such as the engine) and as a useful tool for assessing driver performance. Current solutions found on the market offer the ability to record wheel speed, temperature, pressure, engine RPM’s, suspension travel or acceleration.

By adding wireless transmission hardware to a data logger, gathered data can be analyzed right after being generated, and even in real-time, allowing actions to be taken much quicker in the case of dangerous or failure situations; in the case of two-way communication it may also be used to adjust some parameters.

This work is primarily aimed at a particular prototype vehicle (figure 1), built at Instituto Superior Técnico (IST), to participate in the Formula Student competitions. In these competitions, held every year in different countries, teams of students compete with their prototypes in many different ways, from the dynamic performance of the vehicles on track to the way they manage their team.

Figure 1: The prototype FST03 during a test run.

The project presented here has the objective of designing and building a fully working CAN-bus data logger with telemetry capabilities for a Formula student prototype vehicle. The system should be able to connect with an existing CAN-bus network of sensors and assure the recording of all the data put on the CAN-bus. Besides this, it should be able to transmit live sensor data through a radio link while the vehicle is running on the track to a location off-track, and making it available.
for the user in an attractive way. Finally, the built system should be validated in a real environment, that is, working in the prototype vehicle its meant for (figure 1).

2 Mobile station

2.1 Microcontroller and Real-Time Clock

At the core of the mobile station a microcontroller is used to interconnect and control all the other modules. The device used is a Microchip’s PIC18F4685 8 bit microcontroller operating at 5 V and running with an oscillator frequency of 40 MHz, provided by the external 10 MHz crystal in combination with the internal 4xPLL.

In order to have a time base for the files created on the memory module (section 2.4) a real-time clock (RTC) is added to the mobile subsystem. Connected to the microcontroller, using an I²C bus, is a Maxim’s DS3232 RTC, which has high accuracy and incorporates in its package an oscillator, a temperature sensor and age compensating logic, achieving in this way a very robust device that can withstand temperature swings [7].

For continuous timekeeping a 3 V (200 mAh) coin cell battery (CR2032) is connected to the RTC. When the supply voltage drops below the battery’s voltage, the power supply changes to battery mode. In this mode the typical current consumption for timekeeping is 1.5 µA and therefore the cell’s life is more than 15 years, which is large enough for this application.

2.2 CAN-bus Interface

The Controller Area Network (CAN) is a serial communications protocol developed by Bosch [1] widely used in automotive applications. The CAN communication protocol is a Carrier Sense Multiple Access/Collision Detection (CSMA/CD) protocol. This means that every node on the network must monitor the bus for a period of no activity before trying to send a message on the bus (Carrier Sense). Also, once this period of no activity occurs, every node on the bus has an equal opportunity to transmit a message (Multiple Access). As for the Collision Detection, if two nodes on the network start transmitting at the same time, the nodes will detect the collision and take the appropriate action. In the CAN protocol, a non destructive bitwise arbitration method is utilized, which means that messages will remain intact after arbitration, even if collisions are detected. All this arbitration takes place without corruption or delay of the higher priority message (the node transmitting the highest priority message will continue and the others will wait) [2].

In order to interface the CAN-bus controller of the MCU to the physical medium, a transceiver is used – Microchip’s MCP2551 CAN transceiver. This IC translates the TTL level signals output by the microcontroller into signals conforming to ISO-11898 up to a speed of 1 Mbit/s [3]. According to physical layer specification ISO-11898-2 the CAN-bus should be fitted with a terminating resistor of 120 Ω on both ends with the purpose of eliminating signal reflections and ensure the bus has the correct DC values [5], as can be seen in figure 2.

Figure 2: CAN-bus physical layer block diagram.

2.3 Wireless Link

To implement the wireless communication between the mobile and the base station the chosen radio modules are MaxStream’s XBee-Pro OEM. The key features of the devices, as given by the manufacturer [8, page 5], are presented in table 1.

<table>
<thead>
<tr>
<th>Specifications</th>
<th>XBee-PRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indoor/Urban Range</td>
<td>Up to 100 m</td>
</tr>
<tr>
<td>Outdoor line-of-sight Range</td>
<td>Up to 1500 m</td>
</tr>
<tr>
<td>Transmit Power Output</td>
<td>60 mW (18 dBm)</td>
</tr>
<tr>
<td>RF Data Rate</td>
<td>250,000 bps</td>
</tr>
<tr>
<td>Serial Interface Data Rate</td>
<td>1200-115200 bps</td>
</tr>
<tr>
<td>Receiver Sensitivity</td>
<td>-100 dBm</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>2.8-3.4 V</td>
</tr>
<tr>
<td>Transmit Current (18 dBm)</td>
<td>215 mA</td>
</tr>
<tr>
<td>Idle / Receive Current</td>
<td>55 mA</td>
</tr>
<tr>
<td>Power-down Current</td>
<td>&lt; 10 μA</td>
</tr>
<tr>
<td>Operating Frequency</td>
<td>ISM 2.4 GHz band</td>
</tr>
</tbody>
</table>

In order to validate the specifications given by the manufacturer, measurements were done (table 2) for the center frequency, the channel power, the bandwidth occupied by a channel and the Adjacent Channel Power Ratio (ACPR), for different power output settings to look for possible changes in performance.

Taking advantage of the radio modules automatically outputting the Received Signal Strength (RSS) value,
Table 2: Radio modules measurements results for 18 dBm output power setting.

<table>
<thead>
<tr>
<th>Measurements (expected)</th>
<th>Result</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center Freq. (2.410 GHz)</td>
<td>2.410401 GHz</td>
<td>0.017</td>
</tr>
<tr>
<td>Channel Power (18 dBm)</td>
<td>16.4 dBm</td>
<td>8.89</td>
</tr>
<tr>
<td>Bandwidth (2 MHz)</td>
<td>2.41977 MHz</td>
<td>21</td>
</tr>
<tr>
<td>ACPR</td>
<td>-41.73 dB</td>
<td>n/a</td>
</tr>
</tbody>
</table>

measurements for power/distance ratio are also made and compared with the theoretical loss of signal strength of an electromagnetic wave in a line-of-sight propagation through free space - Free Space Path Loss (FSPL). In this way, having the devices at a distance in meters, \(d\), communicating using a carrier of frequency, \(f\), and with \(c\) being the speed of light in vacuum:

\[
FSPL = \left( \frac{4\pi d}{\lambda} \right)^2 = \left( \frac{4\pi f}{c} \right)^2
\]

which expressed in \(dB\) is:

\[
FSPL_{dB} = 20 \log_{10} (d) + 20 \log_{10} (f) - 147.55.
\]

The result can be seen in figure 3. Although the results are quite far from the theoretical value (in blue), still at the tested distance the transceivers are working above the sensibility limit of -100 dBm (table 1). The reasons for the values being much lower than the theoretical expectations are believed to be related with the way the test was conducted. The number of samples of the RSS value per distance is very low and since between successive packets there are variations in path and reflections in surrounding objects then, likewise, the power varies a lot.

The RF modules interface with the MCU is done through a TTL level asynchronous serial port at 115200 baud and are configured to work at 18 dBm (60 mW), in frequency channel 2.410 GHz, with broadcasting addressing mode, no sleep mode, retries, acknowledge packets nor encryption of the messages.

2.4 Data storage

Since a wireless link is being used, there is the possibility of transmission interruption or fail (range shortage or electromagnetic interference). To ensure that no data gathered by the sensors is lost, it is necessary to have a storage component. This is chosen to a USB memory drive together with FTDI’s Vinculum VNC1L-1A USB Host Controller. This IC is able to, besides handling the USB Host Interface, deal, in a transparent way, with the FAT file structure. The Vinculum is integrated in the system with the help of the VDIPI development module, as can be seen in figure 4.

![Figure 4: VDIPI module schematic detail.](image)

Although there is an extra cost introduced by the use of one more module to interface the memory, substantial improvements in operating speed and implementation time are made, as the overhead of handling the FAT structure with the microcontroller (in the case of a memory card) is eliminated, as it is transferred to the dedicated USB Host Controller. Other advantages of this solution are, besides the very high endurance of USB flash drives, their low price and high availability compared to memory cards.

The connection between the microcontroller and the VDIPI module is done through a parallel FIFO interface using 8 data lines and 4 control lines.

2.5 Power supply

The prototype vehicle has a 12 V battery that powers all the electrical systems on it and since two different voltage levels are needed on the circuit, 5 V and 3.3 V, two regulators are used.

During the test of the complete mobile station’s circuit it was noted that when the system would be powered without a USB drive and then the drive was be connected, the MCU would reset. This behaviour was given to the sudden current draw by the USB drive to which the regulator could not respond and caused a drop on the MCU supply voltage enough to make it reset. To mitigate this fault a high value capacitor was added to the system.

The power consumption by the mobile station was measured in 2 different states. In the Standby state,
when the system is waiting on orders from the base station, the DC current, as measured with a multimeter, is 145 mA which is slightly higher than expected since the major contributors should be the MCU, that according to [4, page 424] draws typically 28 mA (and maximum 44 mA), the Vinculum that needs 25 mA [6, page 14] and the XBee-PRO radio transceivers, that are configured for no sleep, and so their power consumption, from [8, page 8], is 55 mA. The difference between the expected value and the result for the current consumption in this state is believed to be related with both the USB Host controller module circuits and the USB memory drive.

For the measurement when the station is in the Log state, a different method had to be used in order to be able to characterize the consumption in the most power intensive phase, that is when a message shows up in the CAN-bus and it is recorded in the memory and sent through the wireless link. A 0.1 ± 5% Ω resistor was fitted in series with the positive supply line and the voltage drop between its ends was observed in the difference of 2 channels in an oscilloscope to be approximately 30 mV. From this, the value of the current burst is computed to be 300 mA, a value that is due to the high current consumption of the radio transceiver, according to [8, page 8] of 215 mA, while it is in transmit mode.

2.6 Prototype construction

The layout of the mobile station’s printed circuit board was done in Altium Designer, and it’s design had the size, the easy assembly and the high versatility of the prototype as main concerns. The size was reduced by using SMD components as much as possible but still keeping enough space for an easy assembly, as well as for the wires and connections of the board to it’s exterior connectors. The versatility and expandability of the board was assured by providing access to some extra wireless radio’s pins (RSS signal and 3.3 V regulated supply), to all the microcontroller’s ports, and also by fitting extra placement holes for “big” capacitors (seen before in section 2.5).

To house the PCB board, a commercial box was modified in order to support the circuit’s connections to the CAN-bus, the USB memory drive, the antenna and the power switch. The CAN-bus cabling running across the vehicle uses IP67 sealed connectors, so a flanged plug is mounted on one of the enclosure’s side panels for superior resistance and isolation as can be seen in figure 5. In the same figure the extra aluminium piece that was fitted to the bottom so that fixation holes could be added can also be seen. These provide safe holding points when the unit is placed on the vehicle.

![Figure 5](image5.png)

Figure 5: Final assembly of the PCB board inside the enclosure fitted with cables to the connector and power switch. The transceiver adapting cable and the main regulator heatsink can also be seen.

Since is a system which is going to be used by other people, an informative sticker and information LED’s were added. They provide provides basic information on power and signal connections, as well as, antenna type, and allow the user to easily find out the state in which the mobile station is, respectively. The finished mobile station’s prototype can be seen in figure 6.

![Figure 6](image6.png)

Figure 6: Final mobile station’s prototype.

2.7 Software

The program state diagram running in the mobile station’s microcontroller is shown in figure 7. At power up,

\footnote{1According to [8, page 22] when the modules are configured for no sleep, their power consumption in idle mode is the same as in receive mode.}

\footnote{2Ingress Protection rating. The first number concerns solid materials and the second liquids: 6-Totally protected against dust, 7-Protected against the effect of immersion up to 1 m.}
there is first an initialization stage. The MCU’s ports, UART communication channel and the CAN-bus controlling hardware is configured, the Vinculum USB Host controller is initialized and the presence of a USB drive is checked for. The program then stays in Standby till a Start logging command arrives through the wireless link. From here, a new file is started on the USB drive and the program enters the Log loop, being interrupted for each arriving message: messages on the CAN-bus or commands from the base station on the serial line (coming from the radio link). The interrupt routine for the CAN-bus takes care of the incoming messages, putting together a data frame from the meaningful CAN message data and sending it both to the storage device and the base station (through the radio link). If a Stop logging command is received through the serial line, meaning the user issued a stop order in the base station software, the file (on the USB drive) is closed and the environment is reset for a new session. In the following paragraphs a deeper insight into the main routine of the MCU’s software is presented, along with the interrupt routines and the MCU’s peripherals access routines and implementation details.

The messages (or frames) sent from the mobile station to the base station through the wireless link have always the same size and format although they can be data frames or feedback frames. Data frames are sent when the mobile station is in the Log state, and they contain the meaningful fields of the CAN messages received - the CAN_ID of the sensor and the CAN 4 byte data field, plus a frame number. The feedback frames use reserved CAN_ID numbers to signal Start, Info or Error conditions to the base station. The frames are composed by 7 bytes and their format is depicted in figure 8.

3 Base Station

The base station is the part of the system that is located off-track (figure 9). Its hardware is formed by a wireless transceiver and a PC that runs the user interface software to the system, allowing both “online” data collection and “offline” data analysis. The software is written in C++ with Qt graphical libraries.

The application is structured according to figure 10. Since it is possible to configure the software for the kind of session and network it will work with, an important piece is the startup wizard. This pre-application configuration tool guides the user into the sort of session he wants, asking for all the information needed to build the environment. This results in the main application window and the database being built. Depending on the type of session the database is filled up with data values, in the case of an Offline session, or is empty, in the case of an Online session. Likewise, the data source and structure of the main window also depends on the type of session. The data is completely loaded at startup if its source is the hard drive or the USB memory drive, or, in the case of an Online session it is received by the serial interface “block” coming from the Zigbee radio. After the main window is built, the user has the possibility to change a few details in the visualization of the items and also control the state of the mobile station.

The wizard is composed of a series of input fields.
designed to help start the application, and cannot be skipped. The user must complete the forms in order to properly build the application for the desired purpose, although in an Online session some of the information entered can be later modified.

The database is organized as a list of sensors. For each sensor, besides the data values, details on name, CAN-bus ID and type are kept, and whether the values should be plotted or not (Online session). There are 8 types of sensor available: Tire temperature, Throttle position, Steering position, Wheel speed, Suspension displacement, Engine coolant temperature, Engine RPM and Sine Wave (for testing purposes). For each type a different display widget exists.

In an online session the main window (figure 11) is built dynamically, in accordance with the options entered by the user, which means that not all sensor display widgets are viewable every time, only if they are being used, and the same for the Values Plot area, only if this output is chosen for (at least) one of the sensors it will be shown. There are though, some elements that are always built and integrated in the main window. The Status Information and the Sensors List items have this behaviour, but can, like all the items in window, be hidden if necessary.

There is a sensor display widget for each type of sensor in the system for displaying its current value, and the Values Plot item, displays the incoming data for the selected sensors and enables zooming and axis dragging using the mouse buttons, besides showing coordinate value with just a click. The current plot image can be saved as an image file (.jpg, .png or .bmp) by choosing this option on the File menu.

The Status Information item, besides holding the log state control button, from which the user can issue the commands, shows the name of the file being written in the USB memory drive, the current wireless link data rate in kB/s and the current state at which the mobile station is. It also includes an activity log so that the user has feedback of what is “going on” in the embarked subsystem.

Everytime the log “Start” button is pressed a new run begins. Consequently a new file for each sensor to hold the data from the run is created and the Values Plot is reset.

For both modes of the offline session type (load data from hard drive or load from USB drive) the main window has a static structure. There are only two elements, the Sensors List and the Plot Area, which make available all the controls over each trace on the plot and show the enabled traces respectively.

4 System performance

In order to evaluate the entire system’s performance, tests are run in two different environments - the bench and the track.

4.1 Bench tests

In the bench, a CAN-bus network emulation platform is used so that the interaction with the network this work connects to could be emulated. This testing platform is composed of simple modules that can be connected through a CAN-bus and generate data in a similar way as the CAN-bus network nodes of the vehicle do.

The built platform has nodes that can be freely connected/disconnected to the CAN-bus. Each node of the testing network is programmed to output (by putting a CAN-bus message on the bus) a sine wave\(^3\) with a certain frequency. By using a sine wave, the data generated in each node is known \textit{a priori} and so it can be easily spotted if the logging of the CAN-bus messages is being successful.

\(^3\)Constructed out of a sample of 91 values per quarter of period
4.1.1 CAN-bus interruption routine timing

Each time a new message is put on the CAN-bus, the mobile station’s CAN interruption routine picks it up and sends the meaningful data through the wireless link (using the UART to connect to the radio transceiver). Since the MCU can only deal with one CAN-bus message at a time, the amount of time this process takes is crucial to understand what is the least time spacing that messages put in the CAN-bus can have between them. In order to quantify the time a function takes, a random MCU i/o pin is set high throughout its duration and captured with an oscilloscope. For this test, one node of the CAN-bus network emulation board is connected to the CAN-bus, and it sends messages periodically as explained above.

The two important actions made by this routine are the recording of the data frame on the USB memory drive and sending it through the serial link to the radio module. Likewise the time taken by each of these function calls is measured and in the CAN interruption routine the call to function sendFrame() takes 448 $\mu$s and the call to vinculumWriteToFile() takes 8 $\mu$s. This result is in discordance with the approximated transmission time computation that can be done based on the baud rate setting of the UART communication channel. As mentioned before, the baud rate is set to 115200 baud, which means that the line switches state 115200 times per second. Since each switch in state is equivalent to 1 bit, the time a bit takes to be transmitted is:

$$t_{bit} = \frac{1}{115200} = 8.681 \mu s,$$

and for a data frame with 7 bytes (section 2.7) it makes the time to transmit a message:

$$t_{message} = 7 \times 8 \text{ bit } \times t_{bit} = 486.1 \mu s,$$

which is higher than the measurement made. The difference is found to be due to the MCU baud rate setting. According to [4, page 232], for a device’s oscillator frequency of 40 MHz and a UART baud rate setting of 115200 baud, the actual rate has a positive error to the set value and is 125000 baud. If the time per bit and the time per message are computed again:

$$t_{bitREAL} = \frac{1}{125000} = 8 \mu s,$$

$$t_{messageREAL} = 7 \times 8 \text{ bit } \times t_{bitREAL} = 448 \mu s.$$  

4.2 Wireless link maximum throughput

In order to obtain the effective maximum data transmission rate achieved by the wireless link between the mobile and base stations, the test setup of figure 12 is used, where $d = 1 \text{ m}$.

Since this test has the objective of studying the behaviour of the wireless connection between stations, the mobile station is disconnected from the testing CAN-bus and the main program is changed so that data frames containing the sine wave (used in the same way as explained before) values are sent to the base station at a certain frequency. This way, by changing the frequency at which the frames are sent and computing the amount of bytes arriving per second\(^4\) at the receiving end (base station), the throughput of the wireless link can be obtained as can be seen in figure 13.

\(^4\)The throughput value is computed by making an average of the bytes/s arriving in a period of 60 seconds.
A maximum throughput of 9.61 kB/s is achieved by the wireless link with a relative error to the expected of 0.04.

5 Track tests

The system was also tested in a real environment, that is, connected to the vehicle’s CAN-bus network of sensors, with the vehicle running on the practice track. The objective was to evaluate the performance of the system in the ambience in which it was designed to work and also to understand if its enrollment in an electrically noisy system would cause failure. Besides the functioning considerations, overall mechanical design and resistance was also evaluated.

The CAN-bus network of sensors was composed of 3 nodes placed throughout the vehicle as can be seen in figure 14.

Figure 14: Sensors network on the vehicle. The nodes (green) are connected to the mobile station (red) through the CAN-bus (yellow). Sensors: tire temperature (orange), wheel speed (blue), suspension displacement (dark red), throttle (yellow), engine RPM (red), engine coolant temperature (pink) and steering angle (gray).

5.1 Performance for different base station positioning

With the vehicle running on the track, the distance between the mobile and the base station is the key factor, for the good operation of the system, being changed with time. In this way, it is of interest to study what is the best location around the course to place the mobile station. Two different locations are used and they are chosen for both being out of the limits of the track and being believed to turn out the best results in terms of wireless propagation, due to having less obstacles or less distance to the track. In figure 15, the two locations can be seen. Location 1 (blue square) is on top of a small garages building, approximately 4 meters high, from where all the track can be seen, and Location 2 (red square) is by the course’s start line, at approximately 1.5 meters in height.

Figure 15: Practice track image with points of interest. Location 1 (blue square), Location 2 (red square), Track point 1 (green circle), Track point 2 (yellow circle), Trees (green ellipse) and Low area (orange ellipse).
In order to evaluate the wireless link, the emulation platform (explained above in section 4.1) is added to the CAN-bus network already in the vehicle, with just one node generating a sine wave, in order to, as before, have “known” data being generated on the embarked system. By looking for imperfections on the sine wave shape (figure 16, captured with the Save plot image function described in section 3), it was possible to identify the spots on the track where the wireless connection between the mobile and the base station was temporarily broken as the vehicle performs several laps. The “higher” connection loss spots of the track are marked on figure 15. In this figure, it can be seen that with the base station at Location 1 the critical Track points are 1 and 3, which is believed to be due to trees in the station’s line-of-sight propagation, and for Location 2 the critical Track point is number 2, due to the track going through a lower ground area at that point.

Figure 16: Example of a sine wave with short imperfections (black circle).

Figure 17: Data acquired at the practice track. Temperatures of the tire’s inside (pink) and outside (red), the throttle’s position (blue) and the engine’s RPM (green).

Overall, the performance in both locations is similar, as the position of objects in the middle of the track area creates difficulties for both in different spots, and the fact that from one Location to the other, the distance to the vehicle is exchanged by height (which translates in line-of-sight propagation between stations).

Although the method used enabled some weaker connection spots on the practice track to be identified, the performance/coverage of the connection is good, as, considering the average speed at which the vehicle performs the laps (measured to be always above 40 km/h on this track), the losses in “live” sensor data are acceptable and barely noticed, also due to the frequencies at which data is acquired from each sensor (in the case of the test realized: 10 Hz for all sensors except temperature - 1 Hz). In figure 17 a saved image of the plot area shows an example of the data gathered by the system at the practice track.

5.2 Wireless link maximum throughput

The effective maximum data transmission rate on the track was obtained for what is thought as one of the worst case scenarios, when the car is passing by Track point 1 (figure 15) and the base station is at Location 3, which differs from Location 2 on the height, being this approximately 80 cm off the ground. The testing scheme is in all similar to the one of figure 12 for the bench test, with the difference being the distance that separates the two stations \( d \approx 300 \text{ m} \), as can be seen in figure 15. The result of the throughput capability of the wireless link can be seen in figure 18.

Figure 18: Results of the measurements of the wireless link throughput in the track. Expected value in red and the system’s value in green.

The maximum throughput achieved by the system in this case was 4.45 kB/s with a relative error of 0.36 to the expected, and 4.28 kB/s with a 0.08 relative error to the expected. The first value is obtained with a high relative error, meaning a great amount of losses, and so the second value is considered as the maximum with the system in acceptable working conditions.
6 Conclusions

In this project, the development, construction and deployment of a CAN-bus data logging and telemetry capable system was proposed. To reach this objective both COTS parts and original circuit design were put together with software developed for two different platforms and the result was tested in a real world scenario. The work was divided in two parts: an embarked sub-system (the mobile station in chapter 2), composed of a microcontroller, a CAN-bus interface and a storage unit and an interface sub-system (the base station in chapter 3), composed of a PC running an application with a graphical user interface (GUI). In order to connect the two sub-systems, a wireless link was studied (table 2) and used (section 2.3).

Both stations are original contributions of this thesis and the mobile station hardware was developed having in mind the balance between cost, performance and ease of integration, particularly the use of the Vinculum USB host controller (section 2.4) together with a USB memory drive, proved a good option when compared to more classic approaches like, SD or MMC cards and EEPROMs. In the mobile station’s embedded software, the time taken by the critical routines (section 4.1.1) was minimized in order to collect all the messages put in the CAN-bus and, to the extent tested, CAN-bus messages were not lost.

Regarding the base station, its hardware was kept simple in opposition to its software, which grew in complexity because of the project objective of having a dynamic and versatile user interface. The GUI was developed in a Linux OS but using libraries that allow an easy port to other operating systems.

Of note is the fact that this work was developed to interface with other systems being developed at the same time - the CAN-bus network of sensors (section 5). This situation, which is not easily found in the academic course, proved very challenging.

The objective, presented in section 1, was completely fulfilled by the delivery of a working system with reliable CAN-bus message recording and wireless transmission of data up to 4.28 kB/s (value that corresponds to the situation tested at the practice track used). This throughput proved enough to transmit the data generated by an existing CAN-bus network of 20 sensors with the vehicle running several regular practice laps and reaching an approximate 300 m maximum distance between both ends of the wireless link.

The wireless link has, none the less, difficulty to comply with obstacles in the transmission path, being line-of-sight propagation the ideal for the XBee-PRO radios, which means a track completely free of obstacles. Due to this, and the fact that no flow control or acknowledge schemes are used in the wireless transmission of data in either direction, live data from the mobile station can be lost as seen in section 5.1.

Concerning the data recorded on the USB memory drive, its safety is only compromised if proper closing of files does not takes place, which can (as proved by experience on test runs) happen, when the vehicle, for example, undergoes a problem and its power needs to be swiftly cut. The solution would pass by modifying the recording method to make it more robust in the presence of sudden power failures.

6.1 Future work

As future enhancements of this system, it is proposed the further exploitation of the degree of freedom presented by the antenna type. By using a more directional antenna on the base station, propagation problems could be mitigated. Also the use of flow control and the implementation of a minimal handshaking protocol would contribute to less losses on the transmission. Another option would be to move to WiFi technology which would provide a larger bandwidth and also make easier the implementation of a transport layer protocol like TCP. In order to reduce the power consumption of the mobile station a sleep state can be implemented and the telemetry scheme could be changed in order to run in tracks with a lot of obstacles by using the information on the radio signal strength to choose when the mobile station should transmit.

On the base station’s software, it is proposed an enhancement of the serial routine, so that it becomes less CPU “intense”, the inclusion of a “driver comparison mode” and also the possibility to do further configurations of the mobile station.

References