BE C++ : F1 vehicule tracking

Réalisé par
Santiago Andrés Pantano Calderón
Javier Sánchez Félix

As part of the course
Software computing and networks(I4AEIL11)

Travail présenté à
David Gauchard
Thierry Monteil

From the Department of Electrical and Computer Engineering, of Automatic-Electronics
Institute National des Sciences Apliquées Toulouse
26 may 2020

1 Introduction

With the upcoming arrival of 5G, there is an exponential trend in the development of technologies such as the Internet of Things. The best known use of the IoT is the remote control of intelligent houses, autonomous cars and machines, but in fact it has more important functionalities to mention, especially in the field of engineering. Nowadays the processing of huge amounts of data has allowed to optimize the execution of a process in real time thanks to its acquisition through sensors.

In Formula 1 racing, the factors that define the success of a team are associated to the performance of the driver and the technology of the vehicle. In this sector where the result of a competition is defined in milliseconds, the quickness of a good decision making is primordial. In this way we see the relevance of developing a system for monitoring a high competition vehicle, by using of sensors where the processing of the acquired data must be efficient, something that is proposed to be achieved with the use of object oriented programming making use of polymorphism aspiring to a real time response.

1.1 Literature Review

Telemetry was the first way the engineers knew the values registered by all sensors in the cars, and even in this days this is the monitoring method used by all teams. It consist in a host of electronic devices which transmits different measurements from the car to the pit garage to help the engineers to make decisions or plans in order to improve the performance during the competition. Besides, it records all the measurements of the sensors, including the engine, suspension and transmission status data, fuel, temperature in all components, lap time and g-forces, with the purpose of having a foundation to determine better car setups for future races.[4]

The use of this technology started in the late 1980s, when teams only could send and receive data near the pit line because of the antenna range[4]. It was proposed by Williams and McLaren teams, as the microprocessors became smaller, with the objective of collect data to get a better control over the fuel usage. The McLaren MP4/2B was the first car to cross the finish line with this technology, drove by Alain Prost, who won the San Marino Grand Prix over Senna and Johanson after they ran out of fuel, before being unqualified when his car was found to be underweight. Throughout these seasons, many teams decided to follow suit, and the ones who did not do it began to lose time and to feel left behind.[5]

Despite all the advances made in the las 40 years, in the 1990s the computing capability suffered a boom, and it was not used just in cars but throughout the whole team[5]. As mentioned, by 1993 the teams had really exploited the computing advances, until the FIA banned them. After Senna's death, and the acceptance of the using of analogue technology in the competition, the telemetry evolved rapidly, until in the early 2000s the data was able to be bidirectional : it could go from the car to the pit garage and vice versa. In 2003, the FIA finally decided to limit the transmitted data, in order to avoid the remote control of the cars from the garages, and since this year the send of information became unidirectional again, so only the driver could manage his car.[3]

The transmission of data was not completely effective until it was sent to the constructor factory, and with this aid, the analysis of data could be performed during all the week, while the teams were moving up to the next Grand Prix, wherever it was. This is the moment when the Internet of Things was added into the teams. This was an idea of the Mercedes3 AMG Petronas, on the hunt of IoT solutions to boost performance in all the Grand Prix around the world. Actually, it began with a contest made by Tata Communications, requested to all the Formula One fans, and anyone who could develop a solution using IoT technologies and real-time data analytics to make the team excel in human performance, race operations and logistics management would win the grand prize : $50.000. One of the judges of the competition was the driver Lewis Hamilton.[6] But it was just the first step. The contest was called the Formula 1 Connectivity innovation Prize, and after the first challenge, Tata announced the next one : connect the cars directly to the F1 fans mobile with an app[6]. And this is the way the Internet of things began to be important to make differences between teams. Of course, constructors work in their own projects, completely apart from the Tata Communications' contest, and one very good example is the use that is giving Williams Martini Racing team to this technologies : with the use of biometrics of the pit crew, they achieved to make the perfect pit stop, analyzing data to train each member individually[7]. Another example is the use of the IoT by the McLaren F1 Team to boost their engines with more than 160 IoT sensors[8].

2 Methodology

We propose a high composition vehicle tracking system, which uses sensor measurements to activate actuators and keep the driver and support team constantly informed of these measurements. To simulate the captures we go to read text files. The value contained for each item within the file represents the instantaneous measured value. So we created a text file with values for each sensor to demonstrate the operation of the system and its reaction to risk situations by activating actuators. We use the terminal to display the measurement update on the driver's screen and the display of the support equipment using a different i2C output bus for each. Another device was added that particularly simulates a camera capturing images from time to time that distinguishes the color captured.

2.1 Device Class

Following the simulation of a development board, we initially approach the program structure as a modular library, incorporating a base class `Device` that has essential attributes for the configuration of any device such as pointers to the designated pin number, and its definition such as input or output, the i2C address as an integer, an i2C object for the bus and an integer for the delay time as attributes. As methods it has the setters of the memory and configuration pins of the i2C for the device and an instruction execution method called `Run`.

2.2 Sensor Class

We decided to make three classes that derive from Device, the first one is the Sensor class that agglomerates all the common characteristics of a sensor in its attributes ; an integer val as the captured value and an alea integer as the resolution of the sensor.

In the proposed project most of the devices used are analog sensors so the polymorphism of this class is quite present.

2.2.1 EngineTemperature Class

In order to monitor the engine temperature to ensure the safety of the driver and good performance of the vehicle, a temperature sensor is implemented that constantly emits the captured value to the LCD instance corresponding to the pits, in case of detecting a risk value the driver is notified by an indicator LED and a message on its LCD screen, instance corresponding to the vehicle.

2.2.2 Speedometer Class

This class is implemented to measure the speed of the vehicle which will be constantly monitored by the pits through its LCD screen, we propose that this data could be stored in the future to determine the performance of the driver in each stage of the race to determine obstacles and areas of improvement.

2.2.3 RPMS Class

Likewise, this class manages the pressure sensor measurement data focused on the vehicle's tires, throwing the result to the LCD of the pits, in order to monitor the performance and also if a risky level is detected in any of the tires, the driver is notified through its LCD screen. It also has an integer variable `wheel` to distinguish each wheel.

2.2.4 Tachometer Class

This class manages the angular velocity of the engine measured by a tachometer sensor, looking for an optimal performance of the vehicle, the data is displayed in the LCD of the pits to indicate to the driver the moment to shift.

2.2.5 Potentiometer Class

Commonly the level of fuel available in the tank of a car is measured with a float type level sensor that can be represented as a potentiometer, this class captures the data donated by the sensor and transmits it to the LCD screen of the pits, in case of detecting a low level of fuel notifies to the pilot through its LCD screen and an indicator LED.

2.2.6 WheelTemperature Class

This class is similar to `EngineTemperature`, it monitors the temperature in each tire, throwing the results to the LCD screen of the pits and in case of a low level in any tire, the driver is notified by its LCD screen, unlike `EngineTemperature`. It also notifies the driver by means of an indicator LED. It also has an integer variable `wheel` to distinguish each tire.

2.3 Camera Class

The camera class is derived from the device class and is tasked with processing images captured by making their distinction in color to simulate a real-time transmission of the race to the pits. As attributes it has an Image object and a character to distinguish the color.

2.4 Actuator Class

This class derives from the Device class and is intended for the actuators of the device as the integer state indicating its status.

2.5 LED Classes

The LED class is derived from the actuator and contains as an attribute an entire`state` and in turn is ancestor of the TempLED and FuelLED classes that are responsible for lighting an indicator LED in the vehicle when a high temperature in the engine or low fuel level is measured.

2.6 Image Class

Image is a class apart from the heritages described above that has as its attribute an entire pointer to the color of its object. It has methods to randomly generate the color of the pixel from 0 to 255 and the operator modification = for the comparison of the Image objects. These objects are generated in this class and are processed in the Camera class.

3 Diagrams

3.1 Class diagram

This class diagram includes the object-oriented programming of the program, but it must also be considered that there are numerous instances of some objects.

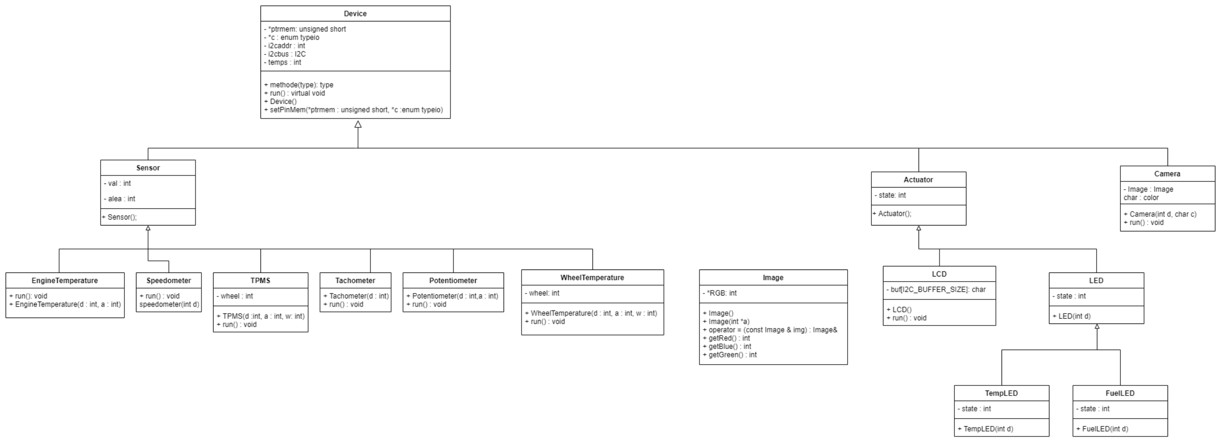

FIGURE 1 – Class diagram

3.2 Sequence diagram

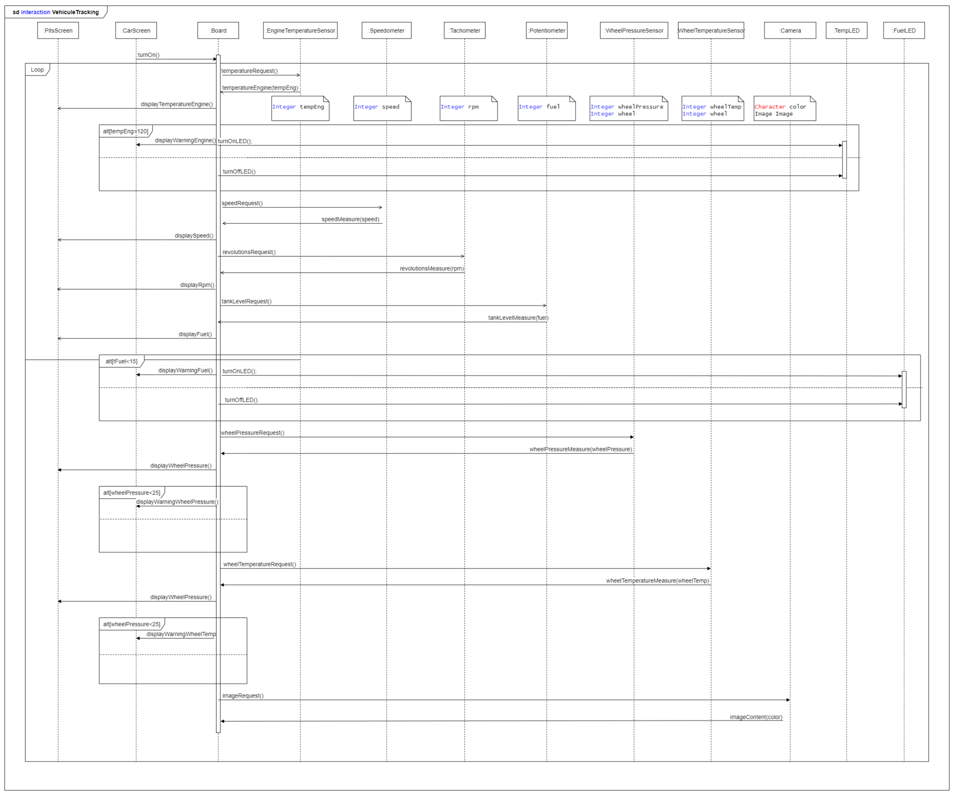

FIGURE 2 – Sequence diagram

3.3 Use case diagram

The following diagram describes the activities that the system performs in order to be physically implemented.

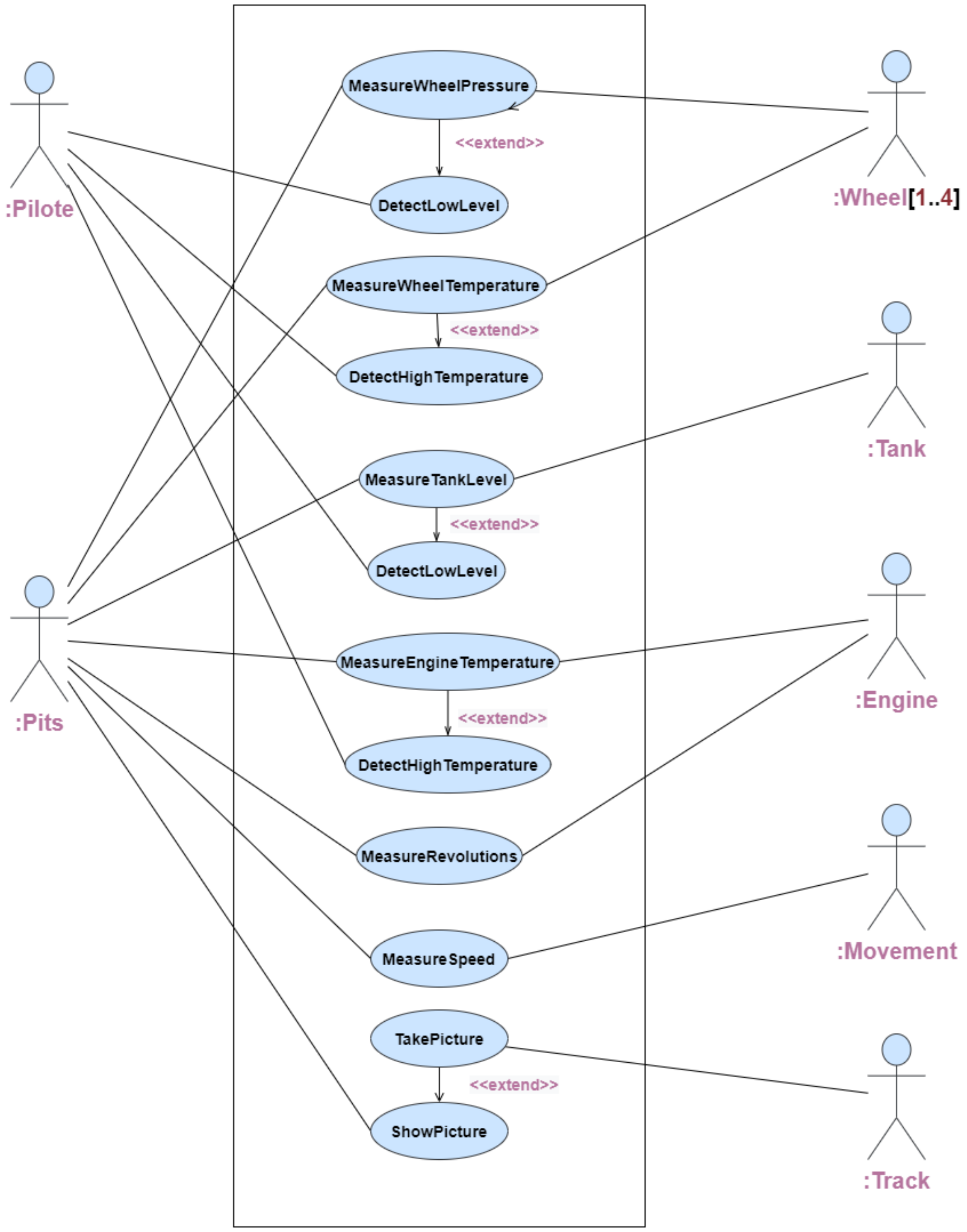

FIGURE 3 – Use case diagram

4 Conclusion

During this design study, we were able to make a functional set based on the simulator provided. We managed to structure the application in an object-oriented programming by optimizing code through polymorphism. Difficulties were encountered when using the 100 character bus due to interference so it was decided to create 3 buses for each i2C bus used, also when implementing the i2C bus input for the camera had difficulties but it was enough to understand the operation of it. According to our project where the speed of operation is important, it is limited to the simulation considering it slow for the application.
The next step would be to physically implement the system, acquire the sensors and actuators and modify the coding as to contemplate electrical factors corresponding to the operating currents of each device. We consider the project as a beginning of it because, nowadays these applications need around 150 to 200 sensors to be competent but we believe that it is enough to exemplify diverse measurement sectors that are made in these competitions.