Politecnico di Torino

Politecnico di Torino

Master's degree in Mechatronic Engineering
A.a. 2021/2022
April 2022

EVduino – an Arduino-like Open Source Vehicle Management Unit

Supervisors:
Stefano Carabelli
Massimo Violante

Candidates:
Michele Galli
Stefano Loreti

Index

Introduction

Nowadays the usage of embedded systems is widespread. They find applications in every aspect of our life: from consumer electronics to industrial equipment, medical devices to security systems, aerospace systems and entertainment devices to weapon control systems, and so on. The drastic rise in the number of such systems, happened in the last two decades, especially in the automotive field. Thanks to their versatility and flexibility, embedded systems are replacing mechanical systems in vehicles equipment, just think about collision sensors, anti-lock braking system (ABS), airbags, emission control, traction control, climate control, and so on. For this reason, they can be considered the heart of a vehicle's electronic system and, consequently, of the automotive market.

This great demand in the automotive field, come to meet the increasing demand for automation, the increasing focus on safety features, and the more and more restrictive pollution standards. Thanks to the safety systems, the number of accident and traffic fatalities dropped down in the developed countries in recent years. Systems like pedestrian recognition, adaptive speed control, merging assistance, lateral collision warning, and many others, help reducing driver's effort, minimizing errors and inattention. Moreover, the growing vehicle sales and the increasing demand of connected car devices are contributing to the global spread of the automotive embedded system market.

Embedded systems allow to develop more advanced control strategies with respect to the traditional mechanical systems, but the increasing number of functionalities required, led to a considerable growth of code complexity. This, together with the stricter safety standards required to avoid software failures, led to a paradigm change in the software development, from hand-coded to model-based development.

Model-Based design provides a mathematical and visual approach to develop systems. It makes use of a model, that can be seen as an abstraction of the physical system, throughout all the verification and

validation phases of the development flow. The advantage of a model is that, used with simulation tools, allows rapid prototyping and software testing and validation, in such a way errors can be spotted and removed before hardware testing. The quality of the product improves without the need for expensive prototypes. Then, code is automatically generated starting from the model, saving time, and avoiding manual errors in the code. This approach is particularly exploited in the automotive field to master software complexity, which is one of the dominant cost items in automotive.

The concept of reducing complexity is why this project is inspired to Arduino. Arduino is one, if not the most, famous embedded system: it is open-source and easy to program. Usually, inexperienced people who want to approach for the first time the world of embedded systems, start from Arduino. It does not require any particular skill because the functions to program its peripherals together with its Integrated Development Environment (IDE) are simple and intuitive. Moreover, Arduino's source code is freely accessible for study, modifications, and eventually redistribution. An Arduino support package is also available on the well-known platform Simulink, to allow a Model-based design. However, Arduino has limited power since it was born for home electronics purposes.

This work of thesis aims at developing an *Arduino-like* toolchain to program an embedded system intended for Automotive applications. The embedded system concerned is a custom board by Ideas&Motion (I&M), called EVduino, equipped with an Infineon Tricore as a processor. EVduino is *Arduino-like* both from a hardware point of view, since it is compatible with the Arduino shields, and from a software point of view since it is open-source and user-friendly.

However, EVduino overcomes Arduino, not only for the power of its multicore processor but also for the possibility to implement multitask applications with a Rate Monotonic Scheduling (RMS), thanks to the integration of Erika Enterprise v2 Real-Time Operating System (RTOS) on the IDE.

The first chapter of this thesis talks about the MultiTask IDE where a hand-written application code can be developed, exploiting an *Arduino-like* set of functions to communicate with the peripherals easily and intuitively.

The second chapter shows how to program the hardware through a Model-Based approach, starting from building a model on Simulink generating the code and exporting it on the IDE where it is deployed on the hardware. To make this possible a Simulink Support Package for EVduino was created starting from the *Arduino-like* functions and provided to the user.

Everything is documented, starting from installation of the needed software to a user manual for both traditional and Model-Based approaches, for supporting the user step-by-step along with the entire procedure. Also, a set of examples are provided to test the toolchain with already working applications. All the documentation, regarding EVduino, was produced by the authors and is accessible through hyperlinks.

Concluding, EVduino replicates the main Arduino's advantages, but at the same time, it is definitely better in terms of power, number, and type of peripherals.

Chapter 1

MultiTask IDE

The MultiTask IDE is the software where the application source code for EVduino can be developed. EVduino mounts a powerful 32-bit Tricore™ AURIX™ TC277 system architecture, belonging to Infineon's new family of microcontrollers. The Integrated Development Environment (IDE) chosen to program this microcontroller is HighTec IDE, as it is one of the software suggested from Infineon's website (Compilers - Infineon Technologies).

Since EVduino is intended for Automotive applications, this software is particularly appropriate for the possibility to integrate a Real-Time Operating System (RTOS). The most suitable RTOS for EVduino is Erika Enterprise v2 because it is open-source and OSEK-compliant, an important requirement in the Automotive field.

A *Project Starting Folder* to be imported on HighTec IDE, already containing all the configuration files for the RTOS, was provided to the user as the starting point of each application.

All the software to be installed, together with a wizard for the installation, can be consulted in the EVduino MultiTask IDE - Getting Started guide.

A library of *Arduino-like* functions was also created in order to communicate with the peripherals of the hardware in an easy and intuitive fashion. It was added to the *Project Starting Folder* so that the user can freely use and consult it in his applications.

The procedure to program EVduino through the MultiTask IDE starts from importing the *Project Starting Folder* on HighTec IDE, writing the application code, building it, and finally deploying the generated executable on the hardware platform.

The full procedure, as well as the structure of the *Project Starting Folder* and the library of *Arduino-like* functions, are documented in the EVduino MultiTask IDE - User Manual. It explains how to create multithreaded

applications from scratch, with tasks at different frequencies scheduled according to a full preemptive (Preemption (computing) - Wikipedia) Rate Monotonic Scheduling strategy. The tasks can be programmed in a file similar to Arduino's IDE, which contains an initialization task (corresponding to the Arduino setup function) and a set of tasks with different execution times (each of which corresponds to an Arduino loop function). In *Figure 1* are shown some of the available tasks for EVduino in comparison with the Arduino's ones.

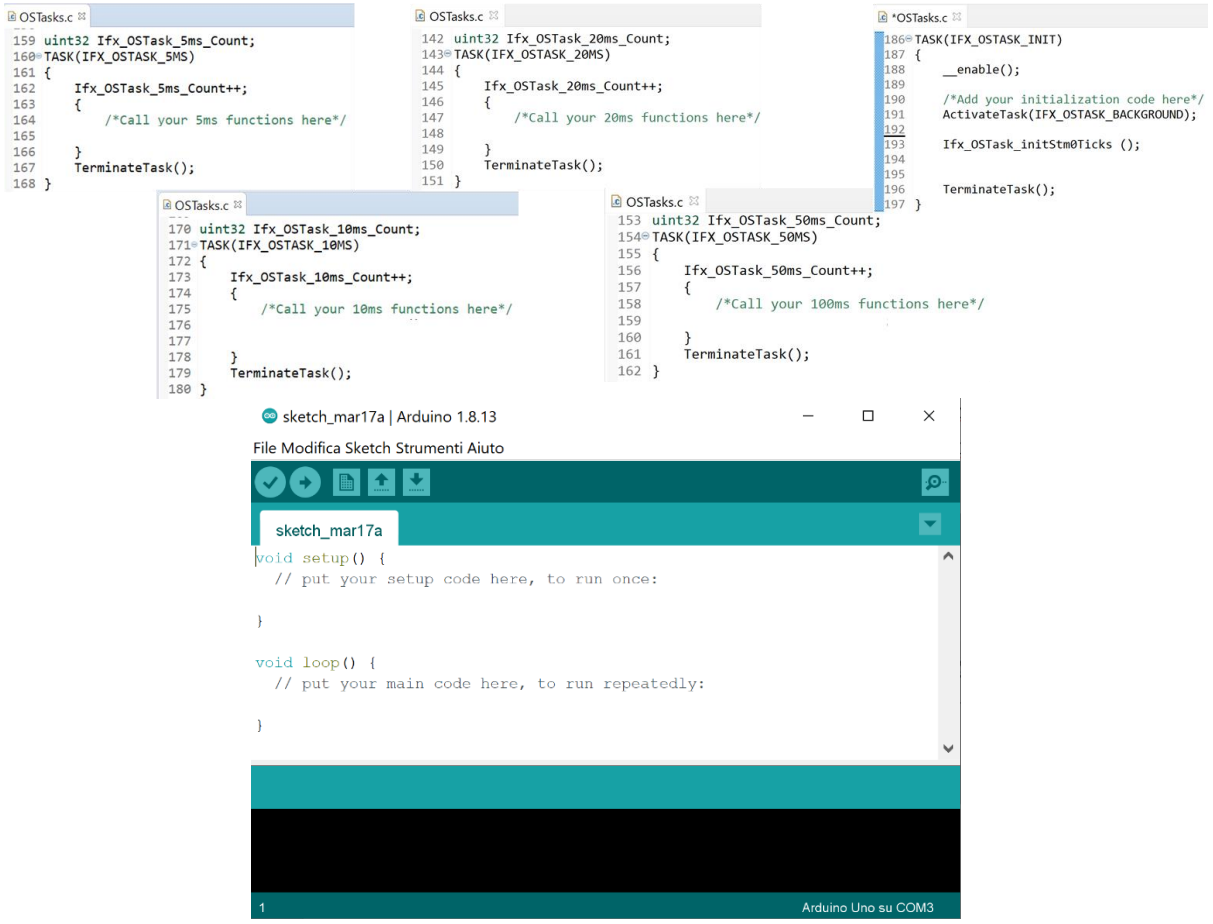

Figure 1 – Some of the available tasks on HighTec IDE (above) and the setup and loop function on Arduino IDE (below)

A multitask application on Arduino would mean using as many boards as the number of tasks to be implemented. EVduino is able to manage more than one task, with the possibility to choose the priority of execution in the

case they are activated concurrently. This is a very powerful tool, extremely important in automotive field.

In this first version of the software, the *Arduino-like* functions allow to program the following peripherals of EVduino:

- Digital I/O
- ADC
- PWM output

Two working applications in the form of tutorials were provided to show the user step-by-step the full procedure, included the coding part where the library of *Arduino-like* functions is exploited.

The first tutorial shows how to make two LEDs blink with different frequencies, using two tasks:

EVduino MultiTask IDE – Blink LEDs Tutorial

In the second tutorials two digital filters, running at different frequencies, are implemented:

EVduino MultiTask IDE – Digital Filters Tutorial

The aim is to cut an analog signal and provide the filtered signals as PWM outputs. Then, two physical RC low-pass filters cut the PWM outputs and reconstruct the analog filtered signals. This is a typical procedure when D/A converters are not available on the hardware.

Chapter 2

Simulink Support Package

From the Multitask IDE we moved to the MATLAB/Simulink environment in order to allow a model-based design approach to develop an application.

The idea was born from another powerful tool to program Arduino, the ***Simulink Support Package for Arduino***, which provides a set of blocks standing for Arduino's physical I/O. This is very useful because it allows everyone, even with low programming skills, to easily program Arduino with a graphical representation of their applications. Moreover, and not less important, the usage of the model-based design approach is mandatory in almost every automotive standard.

For this purpose, a set of Simulink blocks was created, starting from the *Arduino-like* functions, and collected in a Simulink library called ***EVduino library***. This was possible thanks to the ***MATLAB Legacy Code Tool*** which allows to create and customize a set of blocks starting from the definition of the functions contained inside a header and a source file written in C code, as explained in this document by MathWorks.

As well as the first part of the project, the created blocks are intended for the following I/O of EVduino:

- **Digital I/O:**

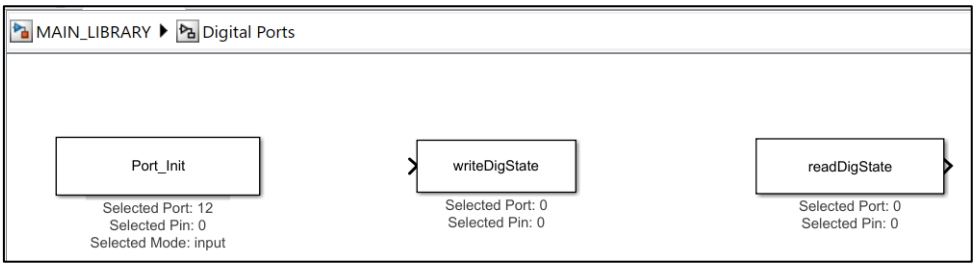

Figure 2 – Digital I/O blocks in 'EVduino Library'

- **ADC**

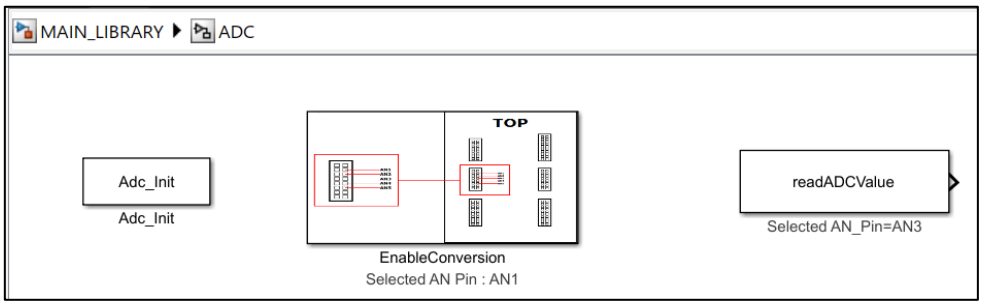

Figure 3 – ADC blocks in 'EVduino Library'

- **PWM output**

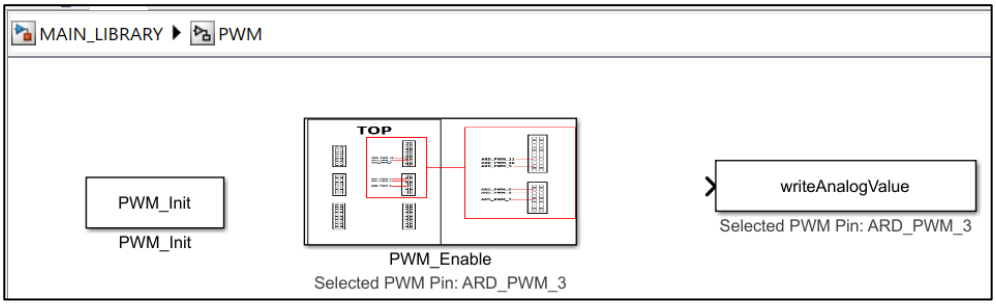

Figure 4 – PWM blocks in 'EVduino Library'

A guide on how to download and integrate this Simulink library is provided inside EVduino Simulink Support Package – Getting Started.

Then, the general rules to design a Simulink application for EVduino and deploy it are explained step-by-step in EVduino Simulink Support Package – User Manual.

Reading this manual, one can observe that once the model of the application is ready, the user cannot deploy it directly from Simulink, but he has to generate the code using ***Embedded Coder***, integrate it on the *Project Starting Folder* and exploit the procedure seen in *Chapter 1* in order to flash the application into the processor. This is the worst drawback with respect to Arduino's Support Package, which provides tools to directly *Build & Deploy* the model on the board, by allowing to exploit some other

very useful tool like *External Mode* (for real-time tuning of the relevant parameters). This is for sure one feature to work on for the next releases of this product.

But on the other hand, with minimal manual intervention on the generated code, it can be uploaded on the hardware with the same procedure seen for **Multitask IDE**. This means that all the features described in *Chapter 1* are still valid.

The applications explained in the **MultiTask IDE** chapter are developed in the form of tutorials even for this second approach. They are available at the following link:

- EVduino Simulink Support Package – Blink LEDs Tutorial
- EVduino Simulink Support Package – Digital Filters Tutorial

It can be noticed that for simpler applications, like the blinking LEDs, is more convenient working directly on the IDE. While for more complex applications where the code to be developed can be challenging, as in the case of digital filters, a model-based approach is preferrable. However, both products work for every application and the user just have to choose the one that suits his needs and capabilities the most.

Future Works

Both products are valid and works properly, but they can still be improved. One way could be expanding the library of the *Arduino-like* functions and consequently the EVduino library of Simulink blocks. For example, adding the functions for sending and receiving data using CAN communication, which is fundamental in the automotive field.

Another powerful tool that can be object of future studies is how to build and deploy the generated code directly from Simulink, without passing through HighTec IDE. This is a necessary step to achieve the final target, which is the real-time monitor and tuning of the parameters, which are the characteristics of the *External Mode*.