Robotic car-like vehicles: a case study for cyberphysical systems

Federico Moro[1], Tizar Rizano[1], Daniele Fontanelli[1] and Luigi Palopoli[1]

DISI, Universitá degli Studi di Trento, Italy

1 Introduction

The technological achievements of the Information and Communication Technology are reshaping our life. The constant access to the network offered by the new generation of mobile devices discloses important and unprecedented opportunities for business users and for "simple" consumers alike. The next thrust is expected to come from the so-called cyberphysical systems (CPS) [4].

A CPS is in the common lingo a device or a system where the computation units are deeply interconnected with the physical system they control. This definition is apparently very close to that of a "classic" embedded system (ES). However, CPSs differ for a number of reasons, some of which are: 1) the number, the complexity and the interconnection between the different control functions integrated, 2) the variety and the level of sophistication of the sensing devices and of the related perception algorithms, 3) the sophisticated Human Machine Interface (HMI), 4) the degree of openness to changes dictated by unanticipated environment condition or user requests, 5) the ability to share information and services with other CPS disseminated in the environment setting the basis for an "internet of things" [1].

We have just outlines some of the requirements posed to the upcoming generation of CPSs, but they are sufficient to suggest the challenging difficulty of the development. We believe that this level of complexity requires out-and-out science of CPSs, where some of the traditional methodologies developed for ESs will be revisited and integrated with new ideas and paradigms [3]. This science is taking its first steps, and it requires realistic examples and benchmarks to measure the efficacy of the approaches it produces on a concrete ground.

The objective of this paper is to offer one of these examples, whose level of complexity has been chosen as a compromise between featuring many of the requirements mentioned above and being easy to replicate with a moderate effort for all researchers interested to use it as a benchmark for their methodologies.

2 System Description

The case study considered in this paper is deeply inspired by the automotive industry. We consider a robotic car, which we have prototyped using a scaled model in our laboratory. In the following text, we will first describe the main functional components of the car. Then we will move on to describing the hardware/software architecture used for its implementation.

Functional View. The system can be considered as a particular instance of a much larger class of similar cyberphysical systems. It is interconnected to a physical environment, which is comprised of the road where the car moves (along with external objects and agents) and by the physical component of the system. Sensors collect information (in our case position, attitude and velocity of the car), while actuators are the means that the system has for environment manipulation (in our case the engine and the steering wheels).

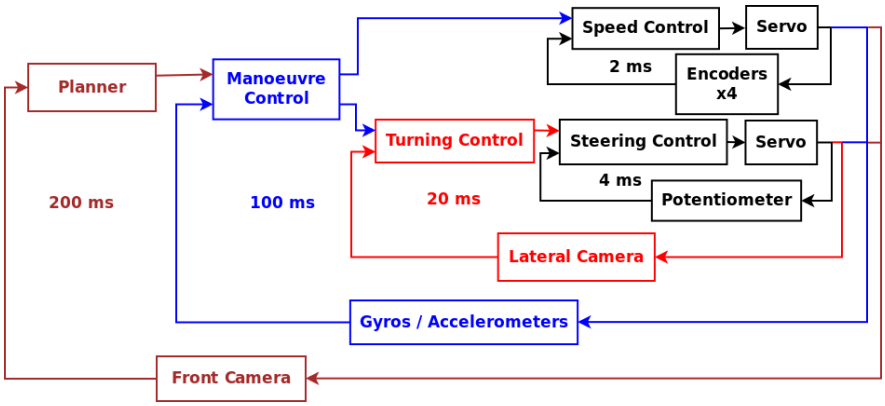

Figure 1: Functional organization of the system. Sensing and actuating blocks are involved in control loops. Contollers are connected in a nested structure, with the planner being the driving block of the entire system. Each loop is associated with a period.

Sensors and actuators are involved in computational activities that implement the feedback control loops for system stabilisation, which in our case enable the system to follow a predefined line. The complexity of the system requires a *planner*, which decides the system goals and sees to their fulfilment. In our case, the planner decides the line to follow, in essence a sequence of manoeuvres such as go straight with speed X, turn with radius Y and speed Z etc. This decision is determined by different considerations, such as saving time, saving energy, overtaking slower vehicles, avoiding fixed obstacles etc. The decided path is tracked by the low level controllers.

From a high level perspective, the system is comprised of a set of nested control loops. Each control loop is activated periodically and has a different frequency, as shown in Fig. 1.

Looking at the model from the bottom up, at the low level the robot is equipped with two servos: one is the engine used to move the car (which operates on the four wheels), and one is for controlling the steering angle. The two servos are controlled by a PWM signal. Each wheel has a relative encoder for speed monitoring, and, on the front of the car, a potentiometer is mounted in order to get a feedback of the steering position. Such sensors are used by two low level feedback loops (activated with a period of $2ms$ and $4ms$), which implement PID controllers used to regulate the speed and the turning angle.

A basic Inertial Platform, composed by gyros and accelerometers, completes the set of low level sensors and is used to improve the estimate of the car position. This information is used by a line following algorithm that controls the position of the car with respect to its ideal trajectory. The algorithm utilises a high frame rate camera, pointing sideways and used to estimate position and attitude of the car with respect to the road line. The line following algorithm and the speed controller receive set points from a manoeuvre controller that decides the sequence of manoeuvres and monitors their execution using encoders and the inertial platform to estimate the progress along the planned line.

A second camera, mounted on the front of the vehicle, is used for path reconstruction and obstacles detection. This camera is activated with a relatively low rate (5 frame per seconds). The Planner receives an image captured through the camera, reconstructs the path and extracts other meaningful information (e.g., on the presence of obstacles) The Planner, therefore, decides which manoeuvre the vehicle has to perform and communicates it to the Manoeuvre Controller.

The vision algorithms used for each camera use a combination of Randomised algorithms

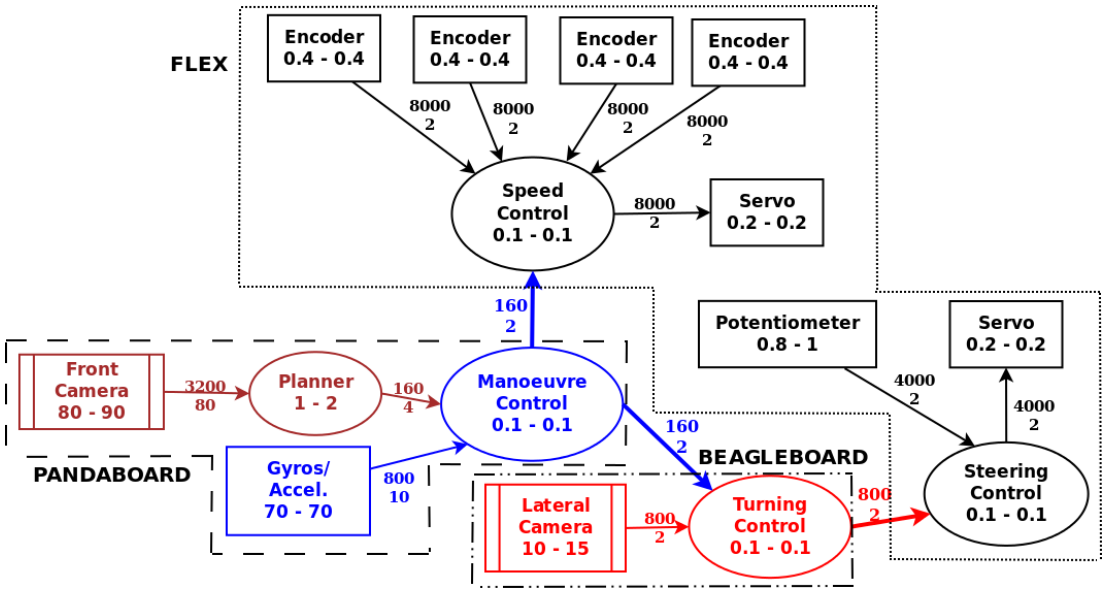

Figure 2: The task set generated from the functional diagram. Each node represents a task and is labeled with its average and worst case execution, both in ms. Edges represent communication between tasks, and bitrate (bit/s) and payload (bytes) are specified. The diagram also shows the allocation of the tasks in the computing units used in the system.

(RANSAC) and Kalman Filtering [5, 6]. Such sensing activities generate a widely changing computing workload, a situation difficult to manage with the standard tools of digital control [2].

Hardware Architecture. The computing system is a mix of microcontrollers and microprocessors. There are three main components. The first one is the FLEX: a development board based on a 16 bit dsPIC. The board is provided with a complete software infrastructure based on Erika which is a RTOS OSEK compliant. The PIC technology allows developers to interface the microcontroller with external objects by means of common digital interfaces like low power communication systems (SPI or I^2C), or PWM. The FLEX also supports advanced communication technologies like Ethernet and CAN bus.

The other two components are two ARM evaluation boards: Beagleboard and Pandaboard. Both are Texas Instruments products and are based respectively on OMAP3 and OMAP4 processors version. The RTOS used for these components is a Linux kernel modified with RT-preempt patches, which improve its real–time performance. Such export SPI and I^2C interfaces for connection with sensors and other low level peripheral; other connectivity solutions are the classic USB, Ethernet, Bluetooth and IEEE802.11 which facilitate remote control and telemetry. Such boards have a sufficient computing power to support the functional blocks described above, but have a limited power consumption and a low cost, both desirable features for robots used in laboratory activities.

Fig. 3 shows the overall picture of the architecture. The processing units are connected through a CAN BUS, which offers a sufficient bit-rate for our applications without incurring the cost of an Ethernet switch in terms of power consumption. Other communication technologies are used for sensors and actuators interface. In general, the FLEX board is adopted for basic functionalities of the system: motor and steering controllers, and therefore for the commu-

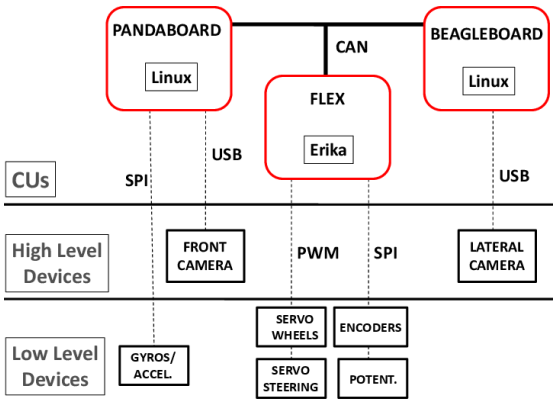

Figure 3: The hardware architecture of the system. Pandaboard, Beagleboard and FLEX are connected with a CAN bus, while each board has specific connections for communication with sensors and actuators.

nication with low level sensors and actuators. An exception is the communication with gyros and accelerometers that requires the connection with the Pandaboard, otherwise the FLEX would not be able to sustain all the traffic.

The two cameras, seen as high level devices, have an USB connection to the two ARM boards. Usually, ARM processors design integrates in the system some co-processing units, like GPUs or DSPs, which extend the capabilities of this kind of processing units. In particular, it is possible to increase the performance of algorithms, such as the ones used in our study case. This reinforces the choice of using microprocessors to define a level of sensors and actuators interaction, and exploit more advanced processors for the high level intelligence of the system. A common bus between the processing units facilitates the data flow for all the tasks running in the system.

Software Architecture and Mapping. Model based approaches recommend to use such models as the primary design primitives to fine tune the design of the planner and of the control algorithms. In addition to highlighting essential mathematical and physical aspects related to the stability of the system and to the correctness of the design, the models composing the functional diagram also suggest a possible decomposition into subsystems, a definition of their relations and of the timing constraints for their execution.

After the system functionalities and time constraints are defined, the computational entities are generated through automated tools or by a manual coding process. This is a refinement step that produces a set of concurrent tasks: each one with a period related to the control loop for which the task is involved. The set of tasks is easy to represent with a directed acyclic graph (DAG), where the nodes correspond to the tasks and the edges are the involved communications. An edge is associated with a weight which specifies the amount of data that need to be sent (bitrate requirements). Fig. 2 shows how a DAG could be derived from the previous diagram in Fig. 1. In our simple hypotheses, every functional block generates a task which receives and sends messages to the tasks derived from the other functional blocks involved in the same control loop. If required, some tasks may be further refined into subtasks. For instance, the Lateral Camera task could be split in two tasks: one for frame capturing and one for image processing. The splitting increases the complexity of the scheduling problem, but could give benefits from the overall computational power utilization. Furthermore, the allocation of the

two tasks in different computing units adds a new message in the system. In the example it would be a whole frame.

Once a refinement of the system into tasks has been produced, the next steps is to map them into our hardware architecture in order to take advantage of the physical parallelism enabled by the different computation units and by their interconnection buses. An optimized task allocation guarantees that all the tasks respect their deadlines and that the traffic generated by tasks communication is sustainable by the system. For this reason, the allocation of tasks to a specific computing unit not only should consider the computational power of the CPU, but also the bandwidth required by output emission of the tasks. Tasks that need to communicate, if allocated in the same unit, will not increase the traffic in the system because the data exchange will happen internally at the computational unit. Whatever the communication link between the two tasks, a fundamental problem at this point is to ensure that the system obtained has the same semantics of its abstract counter part.

3 Possible applications of the case study

The case study described above is a very good work bench for testing different design approaches for CPSs. Possible applications include (but are not limited to):

- Automated mapping procedures for functional models onto a software/hardware architecture (e.g., as shown by Zheng et al. [7].
- End-to-end design of distributed real–time systems, where the different tasks communicate through shared memory or CAN Bus;
- Specification languages and planning for autonomous robots;
- Resource aware control.

References

[1] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[2] D. Fontanelli, L. Palopoli, and L. Greco. Deterministic and stochastic qos provision for real-time control systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE*, pages 103–112. IEEE, 2011.

[3] E. A. Lee. Cyber-physical systems-are computing foundations adequate. In *Position Paper for NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap*, volume 2. Citeseer, 2006.

[4] E. A. Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369. IEEE, 2008.

[5] F. Moro, D. Fontanelli, and L. Palopoli. Vision-based robust localization for vehicles. In *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, pages 553–558. IEEE, 2012.

[6] D. Nistér. Preemptive ransac for live structure and motion estimation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 199–206. IEEE, 2003.

[7] W. Zheng, M. Di Natale, C. Pinello, P. Giusto, and A. S. Vincentelli. Synthesis of task and message activation models in real-time distributed automotive systems. In *Proceedings of the conference on Design, automation and test in Europe*, pages 93–98. EDA Consortium, 2007.