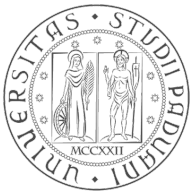


Università degli Studi di Padova

School of Engineering

First Cycle Degree in
Mechanic and Mechatronic Engineering

Control of an Autonomous Vehicle

Degree Dissertation

Graduand

Orazio Scarlatella

Supervisor

Professor Roberto Oboe

Department of Management and Engineering

Academic Year 2013/2014

Contents

Abstract

This thesis wants to give trace of and explain my work for The Freescale Cup University Programme. The Freescale Cup involves the design and control of an autonomous vehicle and culminates in final races on a track for speed.

I joined this international competition held by Freescale as a member of the team "The Jaiton Broules" of University of Padua. Since my University took part to the competition for the second time this year, my teammates and I inherited the work done in the past, setting it to a precious starting point to our development.

In the following chapters, along with a brief description of the system parts, some of the solutions involved in the design and control of the Freescale vehicle will be shown. The ideas to solve specific problems and their implementation were subjected to many changes during the development of the project, but, in general, this thesis won't follow a chronological order, thus presenting what we found to be the best solution only. Although they were not part of the software we used in the final race, some algorithms that could be interesting for further developments are included, too. This dissertation doesn't mean to be exhaustive in the description of the whole work. Instead, it is focused on some of the problems I personally faced, only giving a glance at others and leaving the detailed explanation of what concerns my teammates to their thesis works.

1. Introduction

In the Freescale Cup, students have to find their best strategies to control an autonomous vehicle in the purpose of competing on a track for speed. Freescale organized the challenge on three different levels, from regional qualify races to EMEA[1] finals, to global ones. My team came 2nd at the selection races at Politecnico di Torino on 4th February 2014 and achieved, with great satisfaction, the 4th place at EMEA finals on 30th April in Erlangen.

In the attempt to make the problem clearer, I will describe here the main features of the car and its environment. Freescale provides teams with a standard kit, which is the very starting point for developments and customisations. Students have to take care of problems relating different fields, such as vehicle mechanics, electronics and software design. The racing track consists of a white background with a narrow black line in the middle. In order to complete a successful lap, the vehicle has not to get out of the track with any part. The vehicle is equipped with sensors to gather information from its environment. The most important of them is the camera used to detect the black line, in order to work out vehicle position. A microcontroller processes the information acquired by sensors, takes decisions on its basis and coordinates actuators.

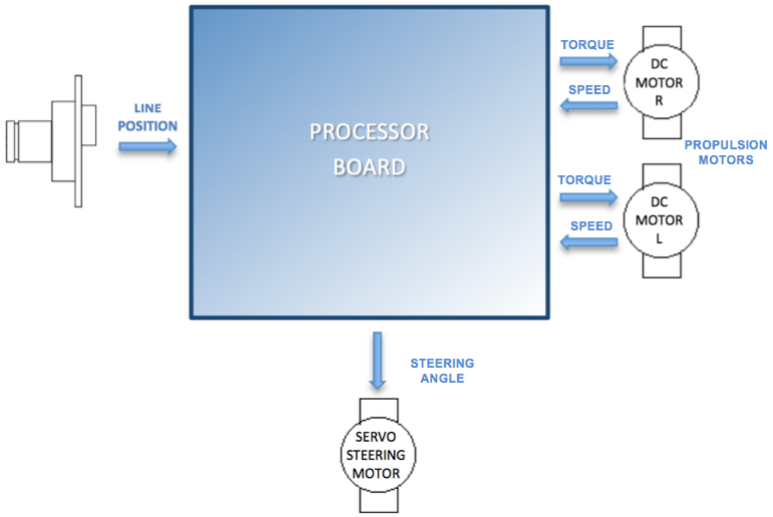


Figure 1: System architecture

Our work started from the comprehension of hardware. This has been my first experience of embedded programming and dealing with a modern microcontroller was a great challenge itself. Once familiar with the microprocessor, we set up a serial wired and wireless communication to monitor data acquisition and software execution. You will find detailed explanation

[1] Europe, Middle East and Africa

of these features in my teammates' theses. At this stage, the challenge consisted of finding effective solutions in the field of data processing and control, to design algorithms and develop small parts easily connected with each other.

Among the instrumentation the University provided us, we took familiarity with digital oscilloscopes, signal generators and digital multimeters. We carried on software development through CodeWarrior IDE[2], Matlab and Visual Studio.

[2]Integrated Development Environment

2. System Overview

In the following sections, each part of the vehicle is briefly described, with the aim of introducing the reader to the system and to the problems related to it.

2.1. Components

University of Padua joined the Freescale Cup event in the academic year 2012-2013, with three teams taking part to the competition. These "pioneers" enhanced the basic kit provided by Freescale with the addition of new sensors and circuitry. We inherited their vehicles and started our development from their achievements. As a goal, we tried to reduce the hardware to the strictly essential parts: the car had to be as light and functional as possible. All the structural changes pursued this aim. The components of our car, as it was in its last version at the EMEA finals, are listed below.

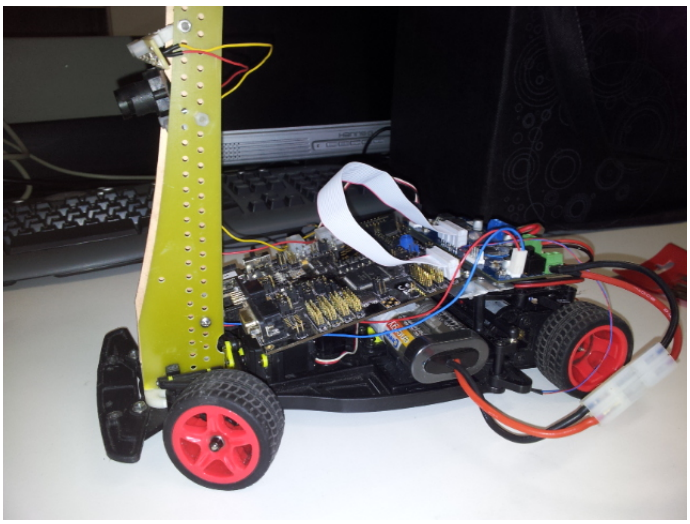


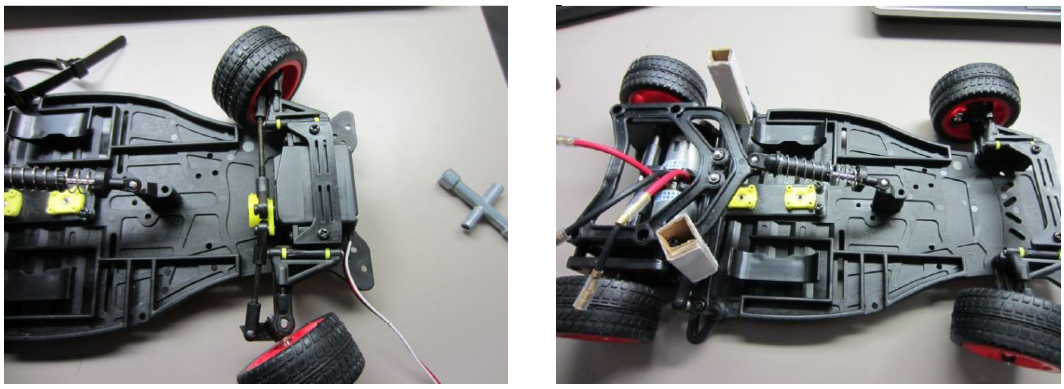
Figure 2: The Jaiton Broules' Vehicle

- Official chassis provided by Freescale
- Custom chassis components, camera support.
- 2 Traction DC motors Standard Motor RN260C
- 1 Servo motor FUTABA S3010 driving the steering mechanism
- 1 Battery pack CONRAD ENERGY NiMH 7.2 V 1300mAh
- 1 Power stage board provided by Freescale Rev0

- 1 5 V power supply custom designed board
- 1 Custom designed 7.2 V LED board
- 1 Microcontroller unit Qorivva MPC5604B
- 1 Line Scan Camera sensors featuring TAOS 1401
- 2 Encoder sensors, each composed of a custom ring and a HOA1405002 IR reflective sensor

2.2. Mechanics

The mechanics of the vehicle is subjected to strict rules in terms of geometry and mechanisms for the transmission of motion. Figure 3 shows the chassis and the steering mechanism provided by Freescale, which we integrally used in the competition.



(a) The steering mechanism (b) The chassis of the vehicle

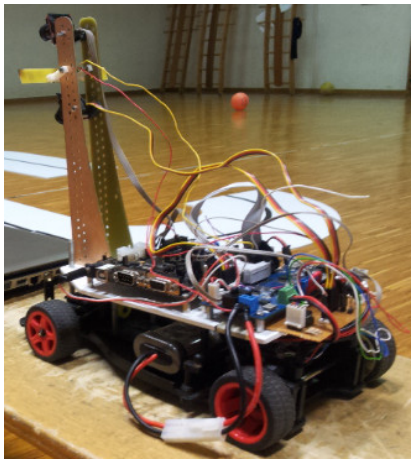
Figure 3: The mechanics of the vehicle

There are two independent traction motors on the rear side of the car, coupled with transmission gears to each rear wheel. The shock absorber in the centre allows two rotational degrees of freedom, the first about the axis of symmetry of the vehicle and the second around the direction of motion of the rear wheels. The front wheels have small shock absorbers too and are driven by the steering mechanism, in turn actuated by the servomotor.

2.2.1. Bodywork Revision

Here is shown how the car passed through different stages of development. As mentioned before, all the changes focused on weight reduction and centre of mass lowering. It's worth to

say that not every part of the vehicle could be modified as there were strict rules about. For example, it was not allowed to replace motors, or to use lithium batteries.



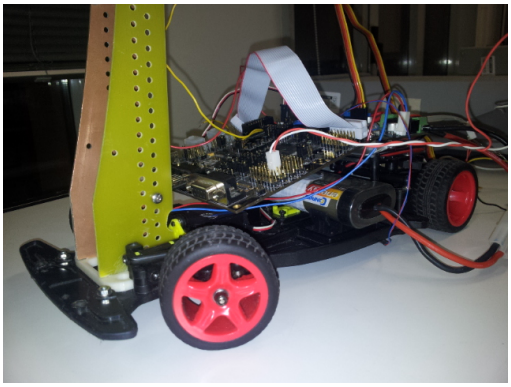


Figure 4: The vehicle in different configurations

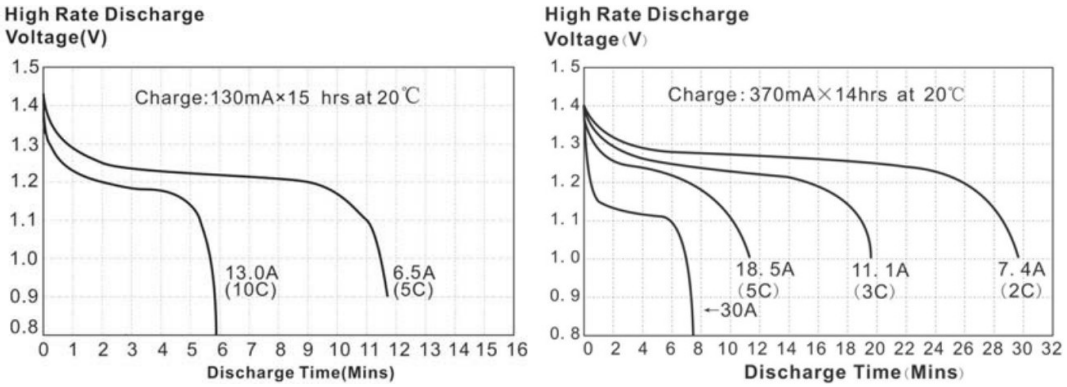
By reducing the frame holding electronic boards and rearranging connections, the structure shifted from a massive to a lighter, more compact one. The weight reduction involved in these changes was about 200 g, taking the overall mass from 1250 g to 1050 g, including batteries. The advantages of this new configuration were tested on road, resulting in a gain of 0.2 s accelerating on a straight track of 2.5 m.

2.2.2. The battery choice

The battery is one of the most massive components of the car. The pack included in the kit was a CONRAD ENERGY NiMH 7.2 V 2400 mAh, weighing 300 g. The challenge rules don't allow to change either the battery technology or its voltage, but they only set a maximum value of capacity. The goal was to find the best trade-off between weight and available power of the battery, which are, in general, related to the capacity of the accumulator itself. Estimating the peak power consumption about 60 W, or 8.3 A at 7.2 V, it seemed useful to reduce the battery capacity giving up some available power, yet obtaining a sensible cut in weight. The choice was a CONRAD ENERGY NiMH 7.2 V 1300 mAh, weighing 150 g. A comparison between the two batteries mentioned above in terms of available power is shown in the parametric curves of Figure 6. This choice shrunk the total mass of the car from 1050 g to 900 g, with all the advantages in terms of acceleration. The weight reduction influences sustainable turning speed as well [3]. Experimental results showed a sensible increase of sustainable turning speed while

[3] It is not straight to understand how a change of mass influences the steering behaviour of the vehicle. The less the weight, the less centripetal force required. Nevertheless, lower weight leads, in turn, to a decrease of maximum

reducing the mass from 1250 g to 1050 g, but the difference was not so evident between the 1050 g and 900 g configurations. The great advantage, in this case, was in terms of acceleration.



(a) Conrad NIMH 1.2 V 1300 mAh (b) Conrad NIMH 1.2 V 2400 mAh

Figure 5: Cells discharge rate

2.3. Actuators

The car is equipped with two independent traction motors and a servomotor for steering control. The traction ones are DC motors Standard Motor RN260C. The servomotor is a Futaba S3010.

(a) Standard Motor RN260C (b) Futaba S3010

Figure 6: Actuators

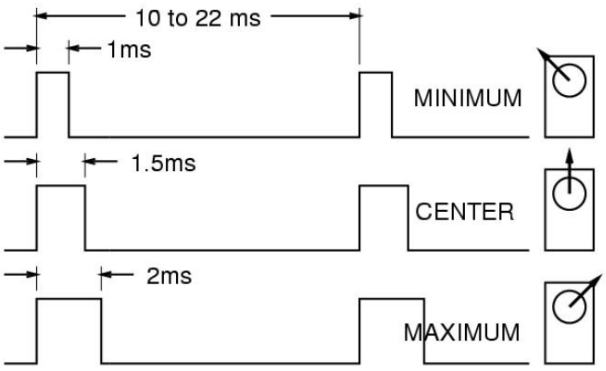
friction forces that allow the car to steer. These two contributions tend to compensate each other, but it's hard to say which is dominant. The relation between friction forces and weight is non-linear for tyres and the calculation need a model of tyres and loads, which was beyond our goals

2.3.1. The DC traction motors

The traction motors are fed by two H-bridges integrated on the power stage board and work at 7.2 V. Low power, 5 V PWM[4] signal, generated by the micro-controller unit, drives the switches of the H-bridge, which produces an output with the same shape of the PWM control signal, but with an amplitude of 7.2 V. This output PWM signal drives the motors exploiting the low-pass capacity of motors inductance: since it is a high frequency signal with a continuous component, to the effects of armature current, it is equivalent to the continuous component only, which is calculated as the mean of the PWM signal.

2.3.2. The servomotor

Servomotors are particular motors that include a control circuitry, allowing the position control of the shaft, which is associated with the control signal. A servomotor has three wires: power supply, ground and signal. It consists of the proper motor, a reduction drive, some kind of an encoder sensor and a control circuitry. The power board provides stabilized 5 V power supply, while the PWM control signal comes straight from the micro-controller unit. There is a one-to-one correspondence between the PWM input signal and the position of the rotor. The control circuitry is sensitive to the ON time of the signal: an ON of 1.5 ms is associated with the central position, and varying it from 1 ms to 2 ms, makes the rotor moves from the extreme left position to the extreme right one. Generally, the total excursion is mechanically limited to 90 degrees.



(a) Servomotor control signal

Basic Information

Modulation:	Analog
Torque:	**4.8V:** 72.0 oz-in (5.18 kg-cm) **6.0V:** 90.0 oz-in (6.48 kg-cm)
Speed:	**4.8V:** 0.20 sec/60° **6.0V:** 0.16 sec/60°
Weight:	1.45 oz (41.0 g)
Dimensions:	Length: 1.57 in (39.9 mm) Width: 0.79 in (20.1 mm) Height: 1.50 in (38.1 mm)
Motor Type:	3-pole
Gear Type:	Plastic
Rotation/Support:	Single Bearing

(b) Futaba S3010 Datasheet

Figure 7: Servomotor specs

[4]Pulse-width-modulation

2.4. Microcontroller Unit

The MCU[5] installed on my team's vehicle is a Qorivva MPC5604B. Its main features from the data sheets are:

CORE

- PowerPC e200z0 core running 48-64 MHz
- VLE ISA instruction set for superior code density
- Vectored interrupt controller
- Memory Protection Unit with 8 regions, 32byte granularity

MEMORY

- 512Kbyte embedded program flash
- 64Kbyte embedded data Flash
- Up to 64MHz non sequential access with 2WS
- ECC-enabled array with error detect/correct
- 48Kbyte SRAM

COMMUNICATION

- 3x enhanced FlexCAN
- 64 Message Buffer each, full CAN2.0 spec
- 4x LINFlex
- 3x DSPI, 8-16 bits wide and chip select
- 1x12C

ANALOG

- 5V 10-bit resolution ADC

TIMED I/O

- 16bit eMIOS module

OTHER

- CTU (Cross Triggering Unit) to sync adc with PWM Channels

[5]Micro-controller unit

Module-Subfunction(if any)	Application
STM (System timer module)	Implement delays
eMIOS-GPIO (General Purpose Input Output)	General Purposes
eMIOS-ADC (Analog to Digital Converter)	Read camera signal
eMIOS-IPM (Impulse Period Measurement)	Read wheels speed
eMIOS-MC (Modulus Counter)	Estimate position
eMIOS-PWMB (Output Pulse Width Modulation Buffered)	Drive the H-bridges
PIT (Periodic Interrupt Timer)	Generate periodic events
INTC (Interrupt Controller)	Handle interrupts
LINFlex	Serial Communication

Table 1: Modules and their applications

- I/O: 5V I/O, with selecting GPIO functionality

Many modules were used for accomplishing different tasks. For example, some were useful to read external signals, others to generate interrupts. Not going in further detail, some of them and their specific task in our case are reported in Table 1.

2.5. CodeWorrior IDE

CodeWarrior is an integrated development environment, which supports Qorivva MPC5604B. It includes an editor, a compiler, a linker and a debugger. In particular, its powerful debugger allows the user to connect the MCU and monitor variables as well as registers. It is also possible to change the value of a variable and to connect the micro-controller in debug mode without turning it off. This was useful for troubleshooting.

2.6. Power Stage Board

The power board used was a dedicated Freescale Rev0. It includes 2 H-Bridges for controlling the DC motors, 2 5 V stabilizers to feed the MCU and the servomotor, and a circuitry for conditioning camera signal. A small 5 V power supply board was added and connected to the signal 5 V stabilizer in order provide power supply for encoder sensors.

Figure 8: Freescale Rev 0

2.7. Sensors

Sensors are core parts of an automatic system. In order to control an autonomous vehicle, some information about the position of the car and its speed is needed. Position and speed estimations are the feedback signals for steering and speed control. The electronics involved with sensors were strictly connected to signal acquisition, while the whole processing stage was left to the micro-controller. Two kind of sensors were used: a line scan camera and reflective sensors for encoders.

2.7.1. Line Scan Camera

The line scan camera is the main sensor involved in the control of motion. It is able to capture sort of an image of a line, which was processed to recognise particular patterns. The sensor used was a TAOS TSL1401C. Apart from power supply and ground connections, there are three wires coming out of the camera: AO, SI and CLK. The sensor consists of a 128x1 array of photodiodes, associated charge amplifier circuitry and an internal pixel data-hold function that provides simultaneous-integration start and stop times for all pixels.

Light, impinging on a photodiode, generates a current, which is integrated by the active integration circuitry associated with each pixel. During the integration period, a sampling capacitor connects to the output of the integrator through an analogue switch. An output cycle is initiated by clocking in a logic 1 on SI. This causes all 128 sampling capacitors to be disconnected from their respective integrators and starts an integrator reset period. As the SI pulse is clocked, the charge stored on the sampling capacitors is sequentially connected to an output amplifier that generates a voltage on analogue output AO at CLK rate.

AO is connected to an ADC port of the MCU. If a minimum integration time is desired, the next SI pulse may be presented after a minimum delay of t_{qt} (pixel charge transfer time)

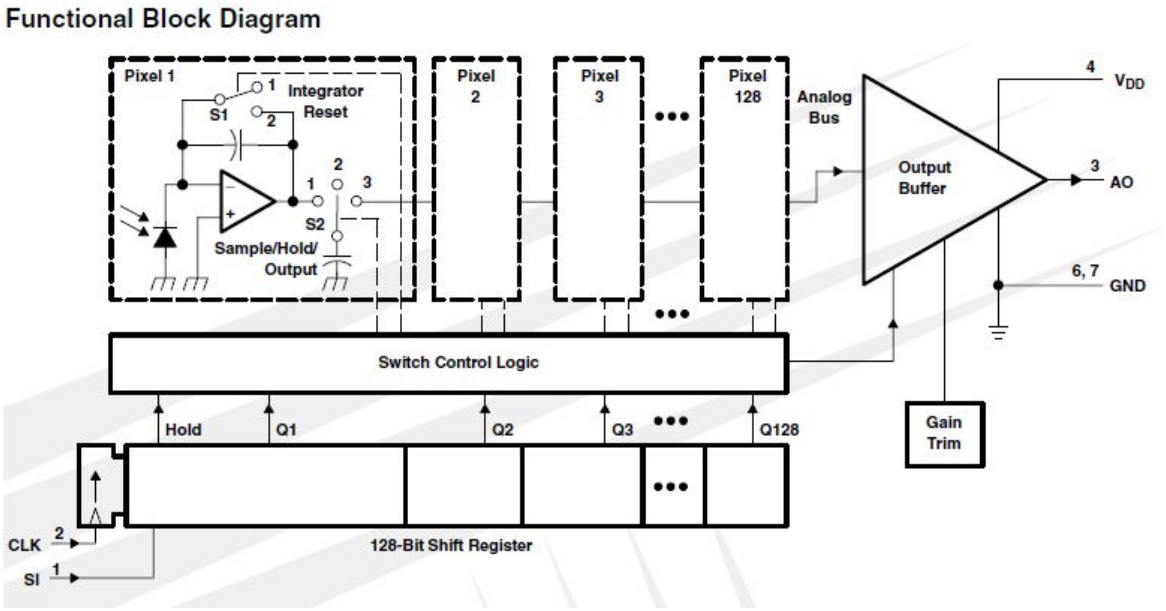


Figure 9: TAOS TSL1401C Functional Diagram

after the 129th clock pulse. The choice of integration time is related to light intensity and sight range. Reducing integration time allows to sample at higher speeds but produces poorer images. A trade-off between speed and image quality led to a sampling rate of 100 Hz. Figure 11 shows a sample picture from the camera.

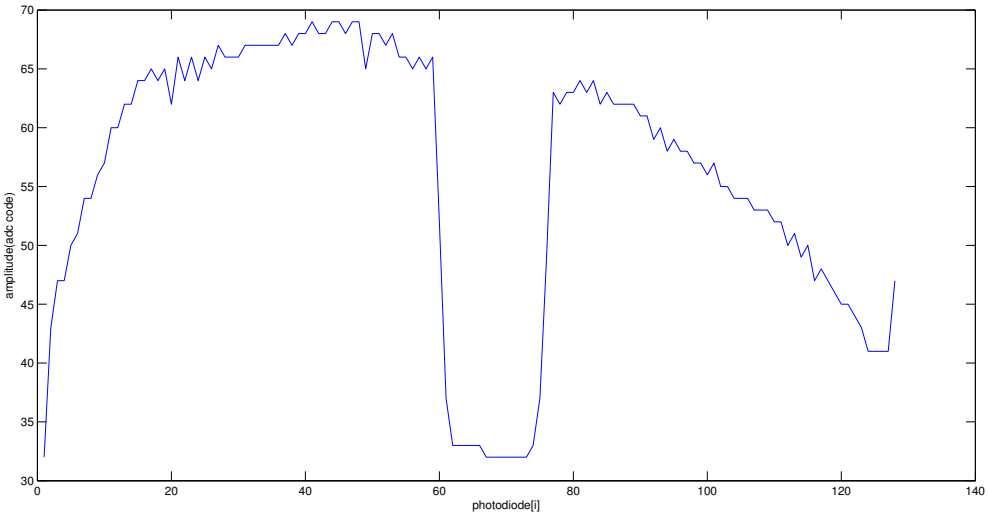


Figure 11: Camera sample

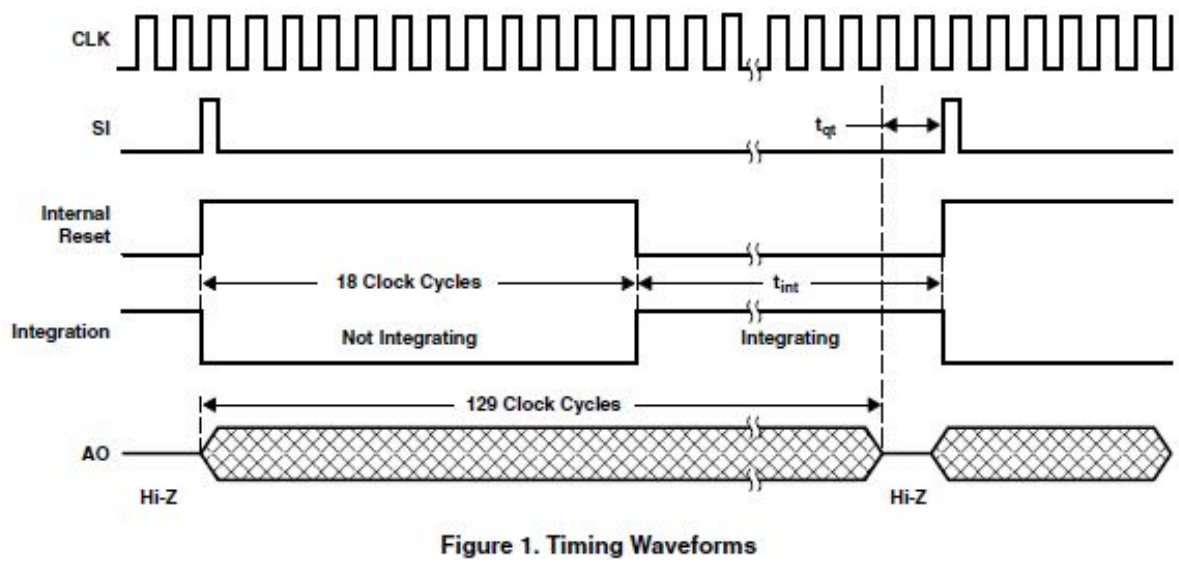
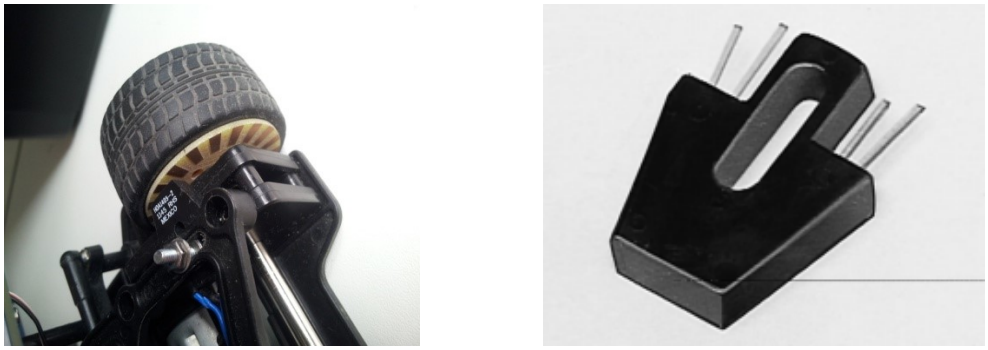


Figure 10: TAOS TSL1401C Time Diagram

2.7.2. Encoder Sensors

Encoder sensors were placed on each traction wheel (rear wheels) in order to read and control its speed. The sensor consists of a custom-made encoder ring alternating reflective and non-reflective sections and of an IR reflective sensor. The HOA1405002 IR reflective sensor



(a) Encoder ring (b) HOA1405002

Figure 12: Encoder sensor

is composed of an IR emitter and a NPN phototransistor. The emitter generates an IR beam towards a surface. According to the surface properties, a part of the incident power is reflected back towards the transistor, prompting a collector current. The HOA1405002 employs an IR filter to minimize undesired effects of visible light. For the encoder application, the expected output signal is a square wave modulated in frequency. The phototransistor was connected in a common emitter configuration (Figure 13).

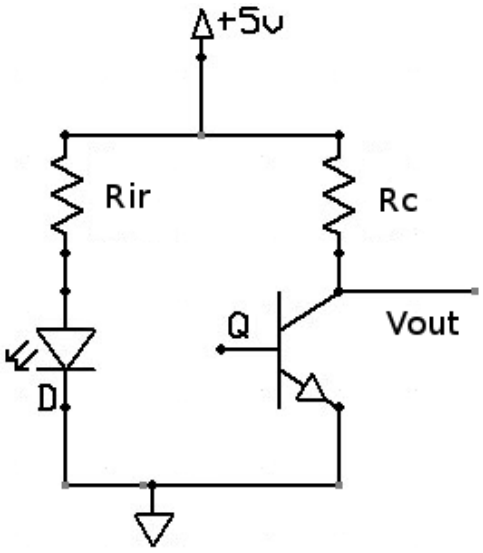


Figure 13: Common emitter configuration

The device was designed to work in interdiction-saturation mode. A characterization of the couple sensor-ring material was needed to adjust the saturation collector current and to ensure that high and low levels match the requirements of the micro-controller input logic. At this point, two opposite needs arise. In order to obtain the sharpest transition, that is the best square wave, it would be suitable to set a low saturation collector current. This would need a high collector resistance R_c[6]. In turn, the higher R_c, the greater the voltage drop across it: this reduces the high level voltage. As a result, R_c should be as high as possible as long as the high level voltage is accepted by the logic. From the data sheets of the micro-controller (Figure 14), the lower limit for the high logic level is 3.25 V at 5 V power supply and the upper limit for the low logic level is 1.75 V, in the same configuration.

Table 15. I/O input DC electrical characteristics

Symbol		C	Parameter	Conditions[1]	Value			Unit
					Min	Typ	Max	
V_{IH}	SR	P	Input high level CMOS (Schmitt Trigger)	—	$0.65V_{DD}$	—	$V_{DD}+0.4$	V
V_{IL}	SR	P	Input low level CMOS (Schmitt Trigger)	—	−0.4	—	$0.35V_{DD}$	

Figure 14: Logic Levels

A trade off was made setting $R_{ir} = 100\ \Omega, R_c = 2.7\ k\Omega$, thus providing a measured high out-

[6]Recall that, in order to enter saturation mode, base-collector and base-emitter junctions must be polarized directly. The transition linear-saturation mode can be roughly set at $V_{ce} = V_{be}$, with saturation for $V_{ce} < V_{be}$. See Ref [9] for further details.

put voltage of $V_{o,high} = 3.86\ V$ and a low output level of $V_{o,low} = 1.22\ V$. It's worth to remember that in common emitter configuration, the phototransistor output is '0' when the IR beam hits a reflective surface thus generating a collector current and '1' when no IR radiation reaches the phototransistor. At last, $V_{o,low} = 1.22\ V$ is not far from the 1.75 V threshold: ambient light fluctuations could create undesired levels transition. Fortunately and thanks to the IR filter, the collector current measured turning off the photodiode is almost null. This means that the signal provided is robust to light variations.

2.8. Led Board

A custom led board, composed of six OSRAM Luw CN5M LEDs, was designed in order to ensure optimal light conditions for camera reading.

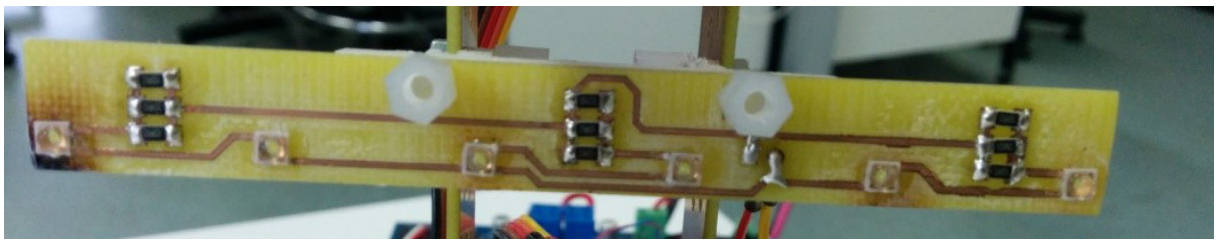
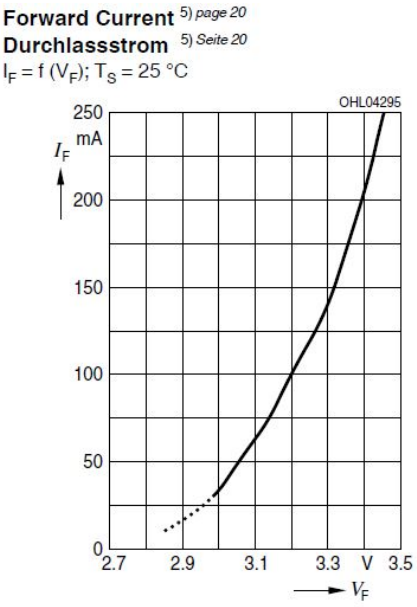
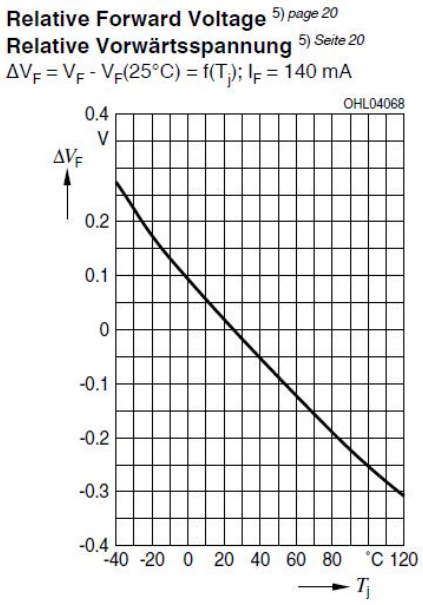


Figure 15: Led Board

The voltage-current curve from the datasheet was useful to determine the working point with a first attempt of 160 mA DC, thus 3.35 V per led as can be found in Figure 16a, since 180 mA is the limit for DC currents. The 7.2 V battery powers three parallel branches, each of them consisting of a series of two leds and a $R = 3.33\ \Omega$ resistor (parallel of 3 10 Ω resistors). Even if this application is not critical for temperature, it's worth to remember the drawbacks of putting diodes in parallel configuration. It is know that the voltage-current characteristic shifts to lower voltages at temperatures that are higher than the standard ambient temperature (25 $°C$). This means that if one of the three diodes in parallel drains more current than the others (e.g. because of tolerances), it heats more too. This, in turn, reduces the voltage drop across the diode, increasing that across the series resistor (their sum equals the 7.2 V power supply) and producing more current, thus prompting a positive feedback effect that will definitively break the diode. In our case, differences in temperature are not appreciable because each led can freely exchange power with its surroundings, leading to steady state conditions of heat flux.



(a) Voltage-current characteristics



(b) Voltage-temperature characteristics

Figure 16: OSRAM Luw CN5M curves

3. Steering Kinematics and Kinematic Model

Consideration about vehicle dynamics are useful with the aim of control. Tyres response is complicated and requires a laborious characterization. The model considered here is simple and it does not provide a reliable description of the system dynamics, but it is enough to make some calculations. We make the hypotheses that the chassis of the vehicle is a rigid body mov-

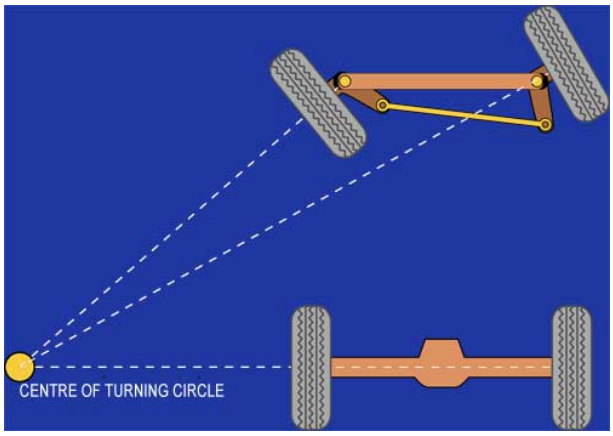


Figure 17: Kinematic steering conditions

ing in a plane and that "kinematic steering condition" is respected. This is the same as saying that all the wheels are in pure rolling motion and that they are rigid and rigidly connected to the chassis in 4 points. Because of these hypotheses, the speed vectors of each of these points is constrained to be in the direction of the plane of rotation of each wheel[7]. Moreover, the 4 points to which wheels are connected, belong to a rigid body, thus the normal directions to the speed of each of them intersect in the instantaneous centre of rotation, as shown in Figure 17. Since the directions of the speeds of rear wheels is fixed under the considered hypotheses, setting that of one more point defines the instantaneous centre of rotation. In this way, the front wheels directions is determined as well. As a result, we can steer controlling the direction of speed of one point only. A strict geometrical law governs front wheels directions, since the theoretical direction of one wheel is not independant from the other. The steering mechanism obeying this law is known as Ackermann's. Since the mechanism of our vehicle doesn't respect Ackermann's geometry, one or more wheels have to slip in order to respect kinematic laws.

Our steering mechanism, shown in Figure 19, can be draft by reducing it to one plane as in Figure 18, thus introducing a reasonable approximation for the model.

[7]Since tyres are not rigid, side forces applied on them produce side strains and slips affect the actual direction of the speed.

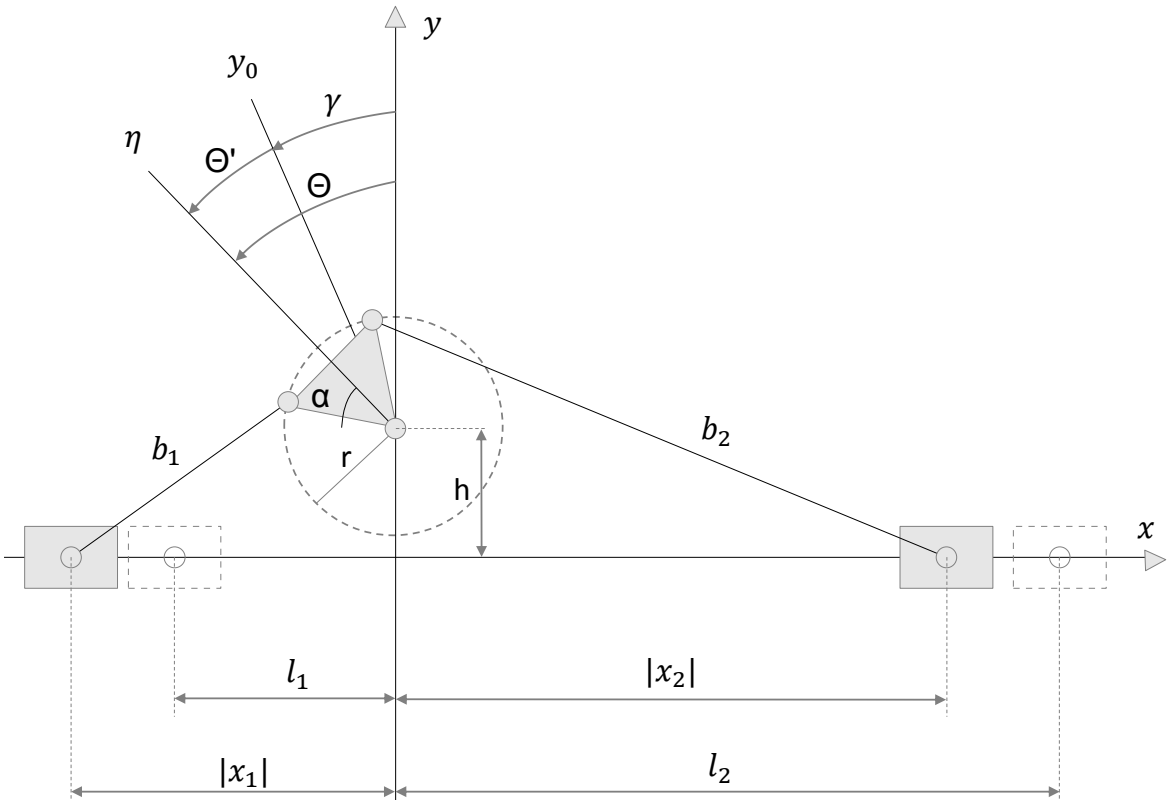


Figure 18: Steering schematic

γ, h, r and α can be adjusted in order to find the best configuration.

$$|x_1| = \sqrt{b_1^2 - [h + r\ \cos(\alpha + \Theta' + \gamma)]^2} + r\ \sin(\alpha + \Theta' + \gamma) \tag{1a}$$

$$|x_2| = \sqrt{b_2^2 - [h + r\ \cos(\alpha - \Theta' - \gamma)]^2} + r\ \sin(\alpha - \Theta' - \gamma) \tag{1b}$$

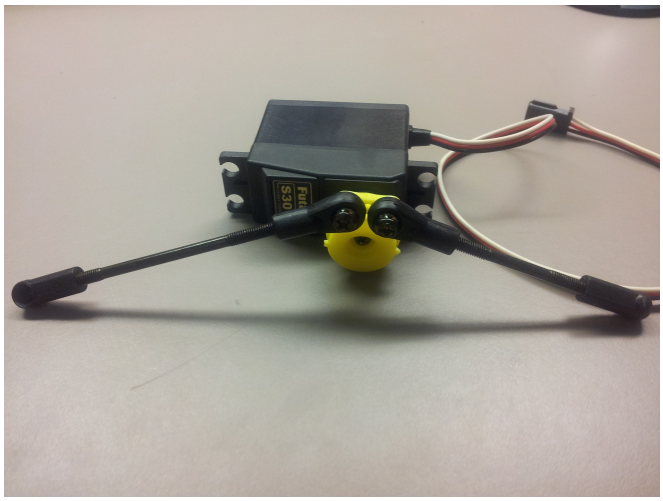
When $\Theta' = 0$, then $|x_1| = l_1, |x_2| = l_2$, with l_1, l_2 known and fixed, and $b_1(\gamma, \alpha), b_2(\gamma, \alpha)$ can be computed. In the plane x-z, with respect to the reference system in Figure 18, the steering angle of each wheel is given by[8]

$$\beta_{left} = \frac{l_2 - |x_2|}{R}, \quad \beta_{right} = \frac{|x_2| - l_1}{R} \tag{2}$$

where R is the distance shown in Figure 19b.

Since ours is not an Ackermann's mechanism, the normal lines to wheels directions don't converge in a single point. For a given couple of angles $(\beta_{left}, \beta_{right})$ it's possible to find $\Delta\beta$, such that the lines drawn considering the new couple $(\beta_{left} + \Delta\beta, \beta_{right} - \Delta\beta)$ converge in a virtual single point. In this way, the non-ideal behaviour of the mechanism is equally split

[8]The equations (2) for $\beta_{left}, \beta_{right}$ are derived in the configuration with the servomotor on the other side of the mechanism with respect to Figure 19b.



(a) Servomotor transmission system

(b) x-z plane

Figure 19: Steering mechanism

between the two wheels. The equations for the lines from front wheels are:

$$y = tan(\beta_{left} + \Delta\beta)x + L \tag{3a}$$

$$y = tan(\beta_{right} - \Delta\beta)x + L - tan(\beta_{right} - \Delta\beta) \tag{3b}$$

Introducing the condition that the interception must belong to the x axis, a transcendental equation in $\Delta\beta$ comes out, which can solved numerically.

With this approximation, the coordinates of the CIR can be found. From the CIR, the steering angle β and the curvature radius can be calculated as well. In particular, β is arbitrarily defined as the angle between the direction of the speed of the point M of Figure 20 and the plane of symmetry of the vehicle, while the curvature radius is the distance between M and the CIR.

At this point, it is also possible to estimate the errors committed for the approximation to the Ackermann's mechanism. Calculating these errors was useful to adjust the position of the servomotor on the chassis in order to minimize them. Figures 21, 22 show a comparison between ideal and real wheels angles and an estimation of the curvature radius for a given servo angle.

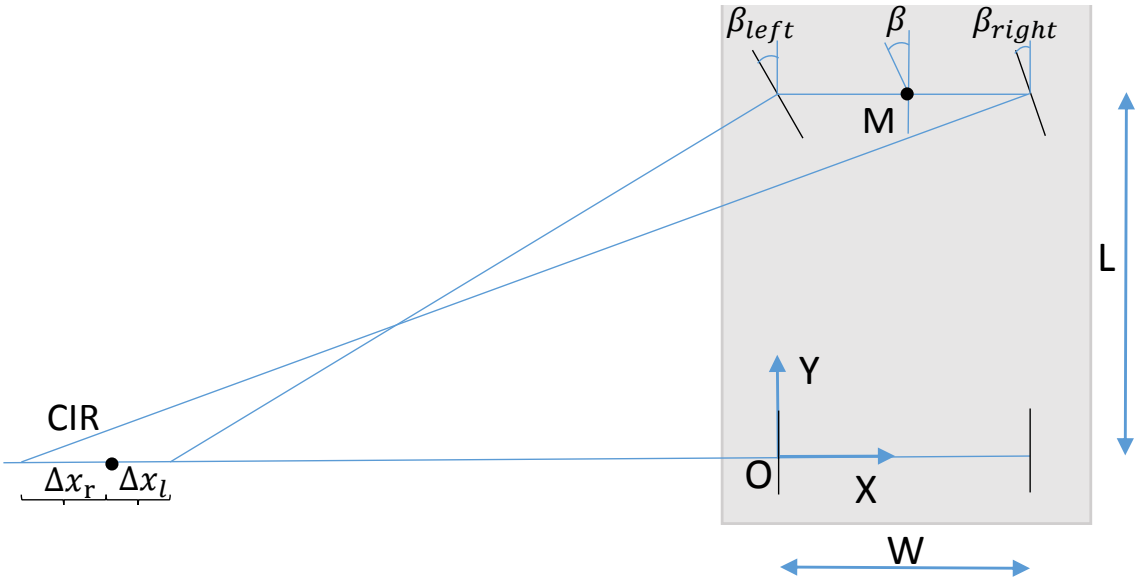


Figure 20: Steering kinematics approximation

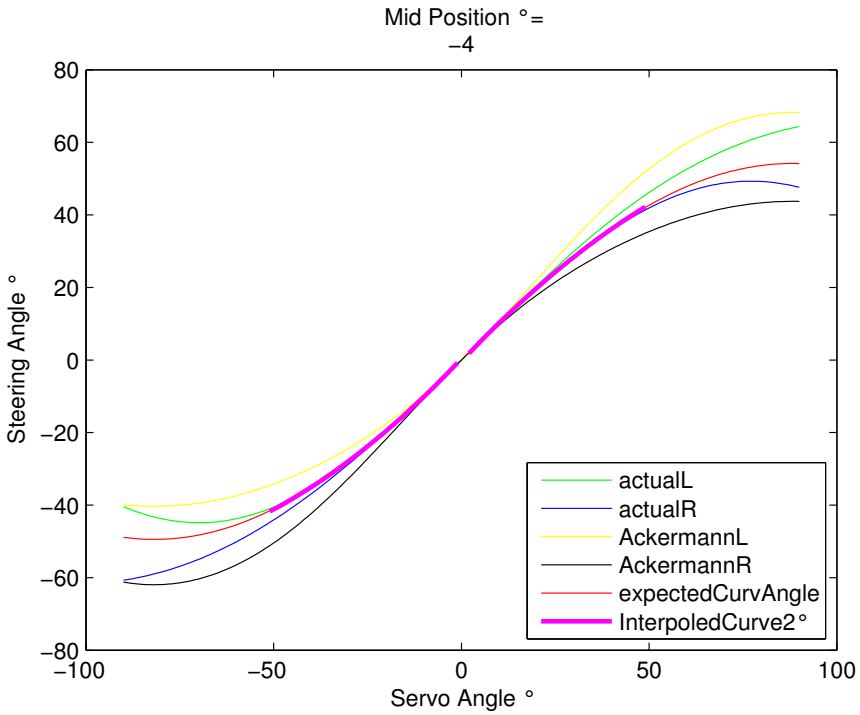


Figure 21: Steering non-ideal behaviour

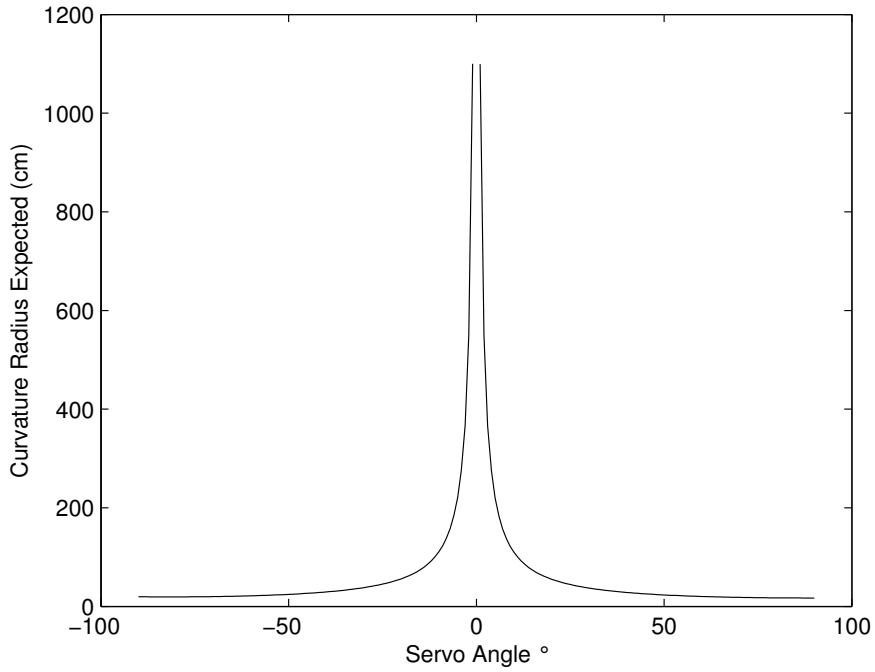


Figure 22: Curvature radius expected

This approach is as simple as inaccurate for some reasons:

- Slips always occur when there are forces on the tire, so the hypothesis of kinematic steering conditions holds for very low speeds only.
- Curvature radius depends heavily on speed, but this model can't show this dependence.
- Equally splitting errors between the two wheels is a trick to define a steering angle, but it has no physical value. In fact, slips are related to the distribution of mass on the vehicle as well as to dynamic conditions.

4. Software Design

One of the main activities which the Freescale Cup involves is software design. Embedded programming of the microcontroller was carried on through CodeWarrior IDE. The programming language used for the MCU is C. Tools like Matlab and Visual Studio were useful for developing auxiliary software and for testing code that would have ran in the MCU. Many times, algorithms were developed in Matlab first, then translated in C and downloaded into the microcontroller. Sets of experimental data were stored from sensors, which were useful to plan and test the algorithms behaviour. The great advantage of this approach is starting from real data sets, which helps a lot in the comprehension of critical points of a problem. Moreover Matlab is a higher level programming language than C, thus allowing to focus on ideas and not to care about implementation matters.

4.1. Software Concept

One of the most important things that concerns teamwork is organization. The basic idea to sort the software out was to build small packages that could easily work together, be independently modified and which could easily adapt for different needs. In order to achieve these goals, we organized the code in different sources, each coupled with a header. Routines are

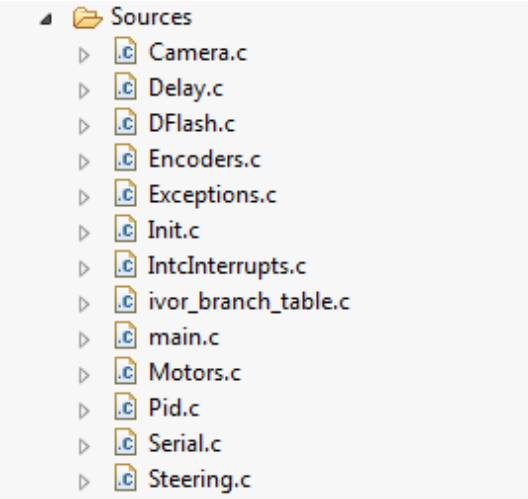


Figure 23: Source files

grouped according to the microsystem they belong to. Signal data and processing results are stored and shared through global structures and variables. In an object-like approach, each microsystem has its own structure which embodies in one or more instances or "objects". Each object is generally associated with a physical or an abstract identity (e.g. a sensor or the results

of data processing). In Figure 24 you can see an example of how encoder sensors are managed by software.

```
#ifndef ENCODERS_H_
#define ENCODERS_H_

typedef enum Position {
    AHEAD_LEFT,
    AHEAD_RIGHT,
    BACK_LEFT,
    BACK_RIGHT,
}Position;

typedef struct Encoder{
    Position position;
    uint32_t speed;
    uint32_t dist; // mm
    uint32_t distCounter;
    uint32_t lastCADRValue;
    uint32_t lastCBDRValue;
    uint32_t notUpdatingEdgesCounter;
} Encoder;

int16_t getSpeed(Encoder *encoder, uint16_t speedSamplingPeriod /*us*/);

uint32_t getDist(Encoder *encoder);

//this function is called by interrupt signals when an encoder's MC reaches
//its maximum value
void incrDistCNT(void);

void installEncoderPeriodicInterrupts(Encoder *encoder,uint16_t clockPeriod);

uint8_t getMCUC(Encoder *encoder);
```

Figure 24: Encoders header

Two encoder sensors are installed on the rear wheels, which are represented by two instances of the structure “Encoder”. The variable “position” is an enumerative type which is associated with the physical port one sensor is connected to. Each function declared in Figure 24 takes an encoder object as an input and do its calculation independently from which sensor is chosen. This means that ports can be easily interchanged and other speed sensors can be added by creating new instances of “Encoder” and describing new connections. Furthermore, encoders information is accessible by each other piece of software, as ”Encoder” instances are declared as global variables. With the aim of increasing flexibility and reducing redundancy, complex algorithms are split in sub-functions that need each other to complete the calculation and it’s up to the user to call them in the right order. For example, the function “findLineCenter”, has to be called after “updateCamera” and “indentifyPatternElements”. This seems confusing, because there are many items to take care about, but it makes the code simpler to

debug and easier for the programmer to keep in mind as a complex piece of software works. Moreover, some sub-functions could be useful for many calculations and not just for one, thus limiting code repetitions.

4.2. Software Architecture

The software can be split on three levels, starting from low level control of peripherals to user interface.

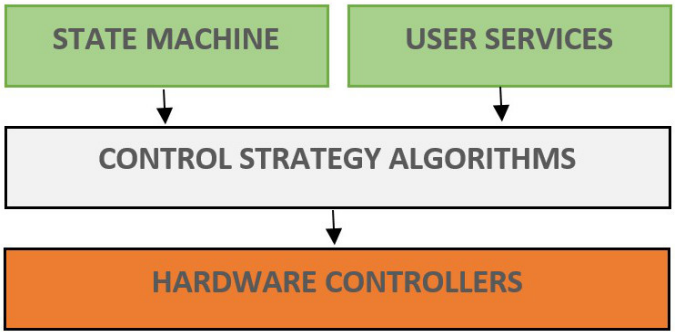


Figure 25: Software architecture

Hardware interface is that piece of software which deals with initiating peripherals and microcontroller modes, but also with interpreting data from sensors and with making them available for control algorithms. Apart from initialization routines, it includes software for interfacing with traction motors, servomotor, camera, encoders and serial communication. Control algorithms are the core of the system, having the aim of processing data and controlling actuators. They consist of temporized routines called by periodic interrupts and include, for instance, line recognition, servo controller, traction motors controllers and estimation of motion. The highest level of software is that of state machine and user interface. By pushing buttons on the microcontroller board, different modes of operation are selected. This was useful for testing and essential for the race; in fact you were not allowed to download different programs for the three attempts you had to complete a lap and state machine was the only way to change parameters after an unsuccessful lap. Finally, an user interface was developed for real time monitoring, downloading data and change parameters through wired and bluetooth serial communication. This piece of software was written in C++ using Visual Studio.

Figure 26 shows the modules the software is composed of. The dashed blocks were not mature enough to be used in the final race.

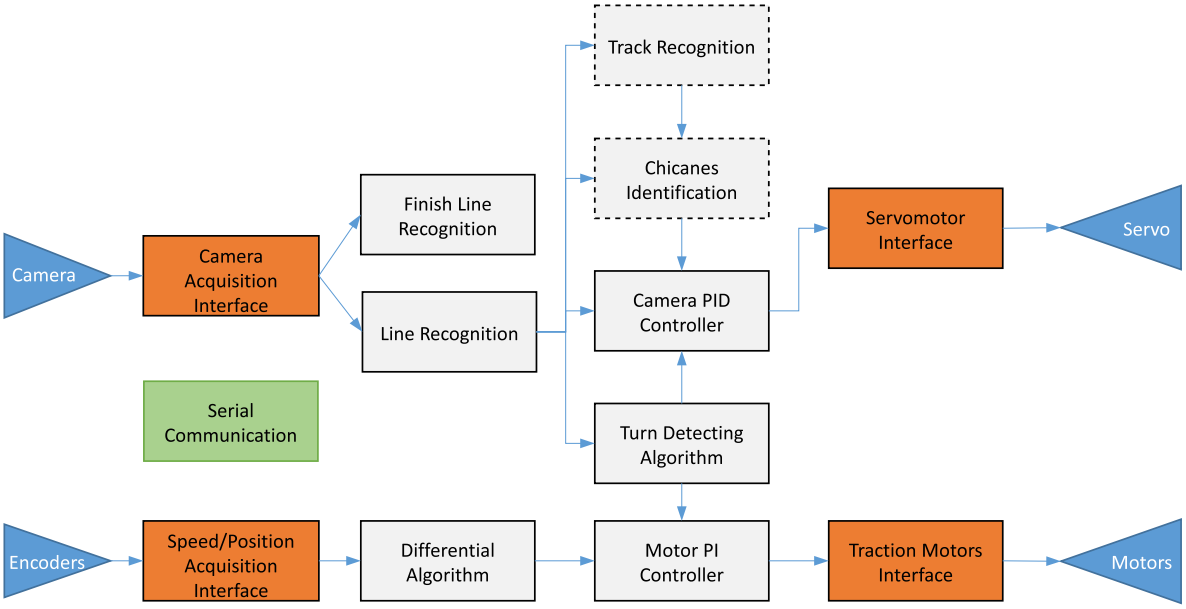


Figure 26: Software modules

5. Control Algorithms

5.1. Patterns Detection Algorithm

Patterns detection is essential for controlling the Freescale vehicle. The relative position of the vehicle to the black line in the middle of the track is the feedback signal for steering and its correct measurement is decisive to setup a good control. The requirements for this algorithm are:

- Robustness
- Versatility
- Computational cost

Robustness is here the most important feature. The first reason lies in light variability and shadows, which the track can be subjected to: as the camera can distinguish contrasts by measuring the power of incident light hitting each photodiode it is composed of, variations in ambient light cause changes in signal amplitude and contrasts become less sharp in conditions of poor light. Moreover, shadows, track borders and intersections, which are normally present on the racing track, can produce patterns that look similar to a black line, leading to ambiguous information. As we used to look far in front of the car to make sorts of predictions using a second camera, the sensor involved in this operation caught a very low signal because of its big sight range. Moreover, the image read by this sensor was often completely meaningless since, while the vehicle was turning, it saw out of the track, gathering dangerous random images.

The second requirement for line detection algorithm is versatility. The goal was to use an algorithm that should be able to detect any kind of pattern, aid by other software blocks, specialized in finding a single black line or the finish line pattern.

Finally, this algorithm had to be as light as possible to allow fast control. When it was worth, only integers have been used in its implementation.

The central idea of the line detecting algorithm is to find the peaks of the signal derivative, which stand for strong slopes in the camera signal or sharp contrasts in the image. One of the main advantages of this approach is that derivative reflects the shape of the signal and not it's magnitude. A scheme of the building blocks of the whole algorithm for patterns detection is shown in Figure 27.

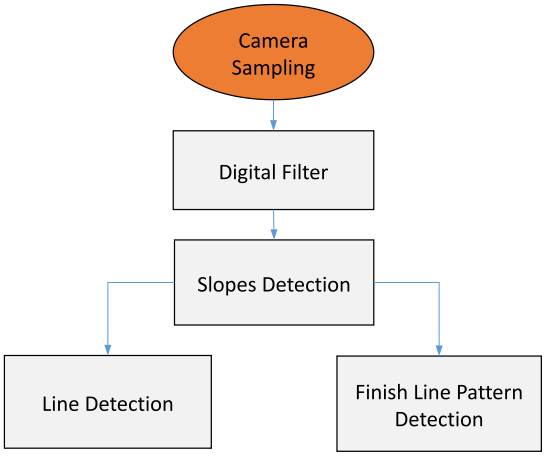


Figure 27: Patterns recognition blocks

5.1.1. Signal Filtering

The first step to extract information from camera signal is to understand which frequency band carries the information. Figure 11 shows a sample signal which is very contrasted and smooth apart from the central dip, which is clearly visible and represents the black line in the centre. Problems occur when contrasts are not sharp and there are disturbing elements in the image. Figure 28 is a good example of a disturbed camera image and highlights some critical points for the robustness of the algorithm. It is a camera sample in non-uniform light conditions and in presence of strong disturbs: the left side of the image represents random things outside the track. On the bottom of the picture, there is the derivative of the signal, that is what the algorithm analyses. Derivative peaks in the left side are produced by high harmonics, but they don't carry any information: it is important to distinguish between information and overlapped noise. The Discrete Fourier Transform[9] (DFT) was useful to understand how to filter the signal. The Discrete Time Fourier Transform (DTFT) of a signal is given by:

$$V(\nu) = \sum_{k=-\infty}^{\infty} v(k) e^{-j2\pi\nu k} \tag{4}$$

where $\nu \in \mathfrak{Re}$. $V(\nu)$ is periodic with respect to ν, with period 1. It holds that $V(\nu) = V(-\nu)'$ if $v(k) \in \mathfrak{Re}$, which means $|V(\nu)| = |V(-\nu)|$. The DFT is essentially the sampling of a period of the DTFT at a proper sampling rate, which allows not to lose information in the discretization:

$$V(h) = \sum_{k=0}^{N} v(k) e^{-j2\pi \frac{h}{N} k} \tag{5}$$

[9] In this case, the signal is seen as a wave packet in space, $v = v(x)$, and space is the starting dominion for the transform.The frequency ν and its discrete equivalent h, are to be intended as wave numbers: $f(x) = Ae^{j\nu x}$

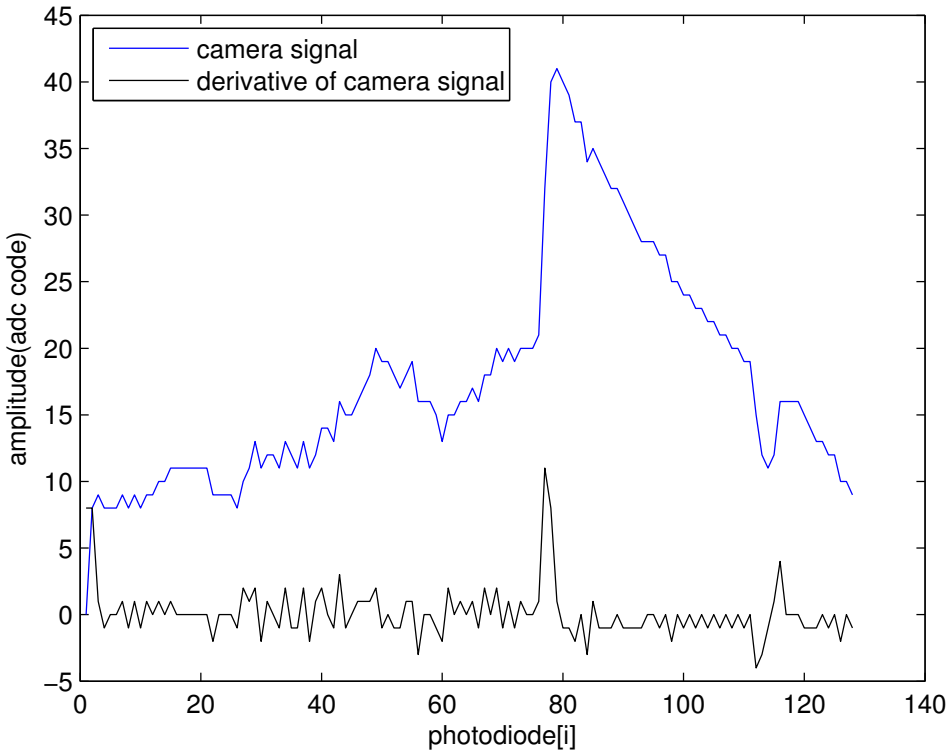


Figure 28: Disturbed camera signal and its derivative

where $N = 128$ is the number of sampling photodiodes and $h \in [0, 127]$. For both the DTFT and the DFT it holds that $V(h) = V(N-h), h \in [0, N/2]$, which means that the DFT can be analysed for $h \in [0, 64]$. It is interesting to see what happens to the signal cutting down $h \in [30, 64]$ frequencies, thus focusing on a low and mid-range spectrum. It can be done roughly setting to zero the desired frequencies coefficients and antitrasforming. The results of this experiment are shown in Figure 29.

Derivative peaks in the left region are lower than before as the result of cutting down high frequencies. Disturbs at high frequencies are even more evident for signals that are small with respect to the input range of the ADC digitalizing it. In this case, small errors due to quantization become evident, introducing high harmonics in the signal.

As long as low-frequency components are not carrying information neither disturbing, a simple low pass filter was implemented, discretizing a first order continuous low pass filter with Tustin approximation.

$$F(s) = \frac{\omega_p}{s + \omega_p} \qquad with \quad s = \frac{2}{X_s} \frac{1 - z^{-1}}{1 + z^{-1}} \tag{6}$$

$$F(z) = \frac{\omega_p X_s (1 + z^{-1})}{(\omega_p X_s + 2) + (\omega_p X_s - 2) z^{-1}} \tag{7}$$

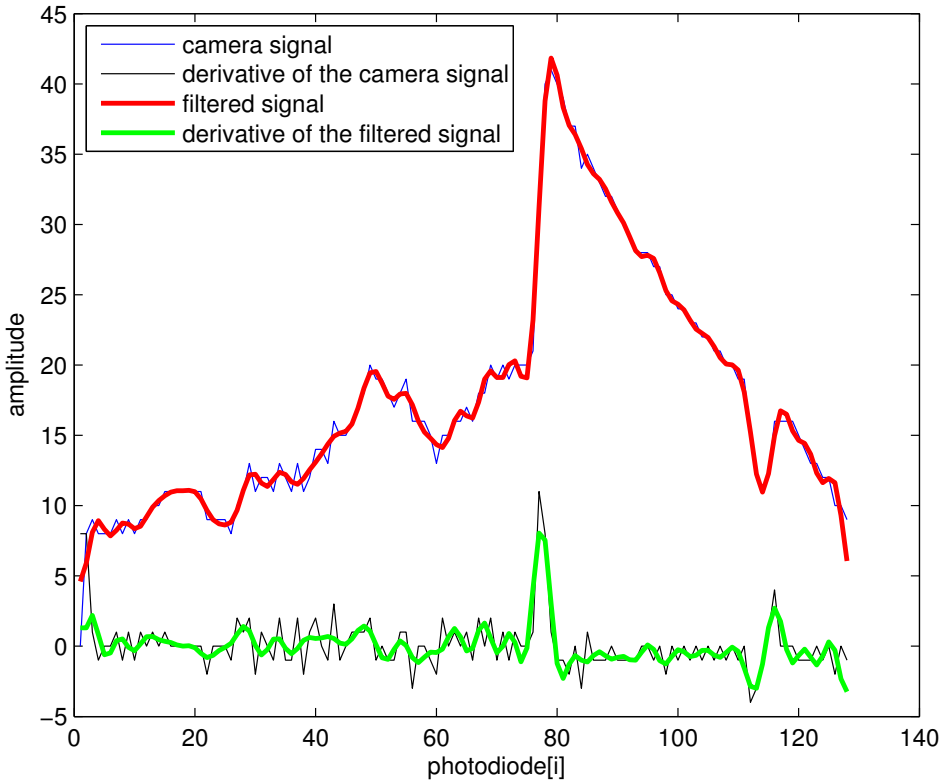


Figure 29: Camera sample without $h \in [30, 64]$ frequencies

In equation 7, $v(k)$ is the input signal and $v_F(k)$ the output one. In k dominion

$$v_F(k) = \frac{\omega_p X_s v(k) + \omega_p X_s v(k-1) - (\omega_p X_s - 2) v_F(k-1)}{\omega_p X_s + 2} \tag{8}$$

The discretization of a signal with sampling length X_s causes a repetition of its transform with period $\omega_s = \frac{2\pi}{X_s}$. The frequency range $\omega \in [0, \frac{\omega_s}{2}]$ matches the discrete interval $h \in [0, \frac{N}{2}]$. ω_p is given by

$$\frac{\omega_p}{\omega_s} = \frac{h_p}{N} \tag{9}$$

By setting a cutoff discrete frequency $h_p = 30$, as in the previous example of Figure 29, ω_p is given by $\omega_p X_s = \frac{2\pi h_p}{N}$. For $h_p = 30$, $w_p X_s \approx 1.5$.

In the implementation of the filter, a gain coefficient of 1000 was introduced to perform filtering and following operation with 32bit integers, in place of doubles. This accomplished to save a lot of computational time, but needed caution to arrange arithmetic operations in the right way to avoid overflow and excessive loss of precision. For example, amplified camera derivative values are of the order of 10^4 and are stored in signed int32 variables: multiplying two derivative values produces an overflow.

5.1.2. Slopes Detection

Once filtered, the signal has to be analysed in order to find the black line pattern or the finish line one. This task can be split in two steps, the first of them being pattern independent. The derivative of the signal is computed using Euler approximation:

$$\frac{dv_F(x)}{dx} \approx \frac{v_F(k) - v_F(k-1)}{X_s} \tag{10}$$

At this point, the algorithm shown in Figure 30 scans the array of the derivative one element at a time and compares its absolute value with a threshold. When a derivative value overcomes the threshold, the algorithm understands he found a new peak. Until values keep being higher than the threshold, the algorithm stores the maximum value seen in that peak and once the peak is off, it recognizes that value as the height of the peak and its index as the position of the maximum in the array. Peaks positions are all stored in a vector.

A last note about the threshold is due. Instead of being set to a fixed value, the threshold is computed as the average of the modulus of the derivative, weighed with an experimental coefficient. In normal conditions, the black line produces the highest derivative peaks. When no line is on the background, no peak has to be found, but the average derivative is dangerously low, because the signal looks like a smooth hill and there are no peaks. The computed threshold could be very low and, to avoid meaningless peaks to be considered, a constant quantity is added to the threshold, which sets the lower limit for its value.

```
//the first value of the derivative can't be in a peak because is conventionally put to 0.
//If it hadn't been, i should have checked wether it is in a peak or not.

for(i=cam->wasteBits+1; i<(128-cam->wasteBits) && j<MAX_PEAKS_NUMBER; i++)
{
    if (!imInAPeak && (abs(cam->derivative[i]) > cam->thresholdDerivative))
    {
        imInAPeak =1;
        currAbsMaxIndex = i;
        currAbsMax = abs(cam->derivative[i]);
    }
    else if( imInAPeak && (abs(cam->derivative[i]) > cam->thresholdDerivative))
    {
        //if the derivative is positive at i-1 and becomes negative at i, while it's abs value
        //is always above the threshold,
        //I have to recognise the end of a positive peak and the beginning of a new, negative one.
        if((cam->derivative[i]>0)!=(cam->derivative[i-1]>0) && j<MAX_PEAKS_NUMBER)
        {
              cam->derivativeExtremsIndices[j]=currAbsMaxIndex;
              j++;
              currAbsMaxIndex = i;
              currAbsMax = abs(cam->derivative[i]);
        }
        else if(abs(cam->derivative[i]) > currAbsMax)
        {
            currAbsMaxIndex = i;
            currAbsMax = abs(cam->derivative[i]);
        }
    }
    else if ( imInAPeak && (abs(cam->derivative[i]) < cam->thresholdDerivative) && j<MAX_PEAKS_NUMBER)
    {
          imInAPeak =0;
          cam->derivativeExtremsIndices[j]=currAbsMaxIndex;
          currAbsMax=0;
          j++;
    }
}
```

Figure 30: Slopes detecting algorithm

5.1.3. Line Detection

At this stage, the program seeks for two peaks in the derivative array that satisfy particular requirements. For the detection of the middle black line the rules below apply, as it can be realized by looking at Figure 31.

- The first peak is negative, the second positive.
- The peaks can be neither too distant, nor too close.
- Among the possible lines, the closest to the line detected in the previous sample is chosen.
- If the line is too far from that of the previous sample, it is discarded.
- If no line is detected, one of the peaks could be hidden because is it out of camera range, but the other can still be seen.

If no line is found, the last meaningful output is held and a flag is set as a warning.

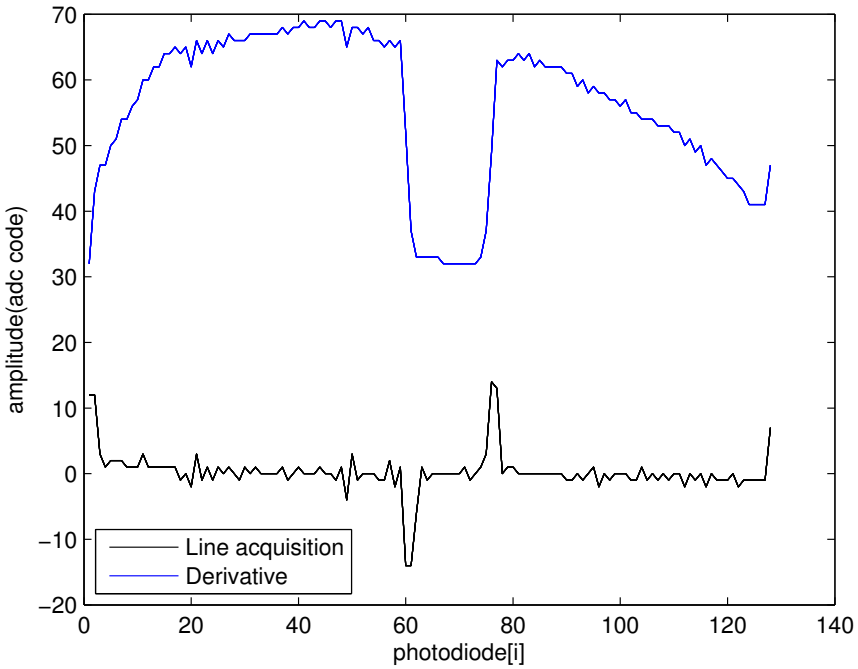
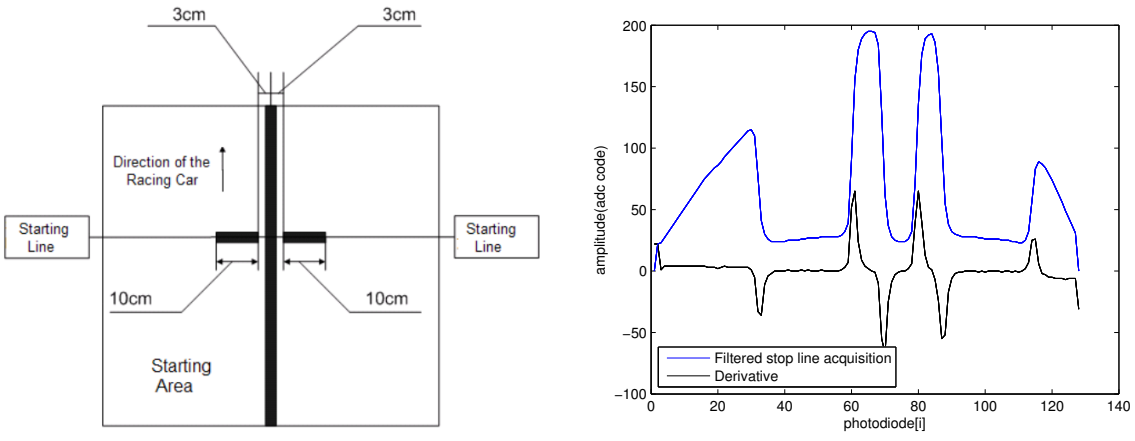


Figure 31: Line acquisition

5.1.4. Finish line detection

Detecting the finish line is similar to detecting the central line. The finish line pattern is shown in Figure 32. In this case, the algorithm looks for a sequence of four peaks properly spaced. Derivative peaks have to respect signs in such an order: positive, negative, positive, negative.



(a) Finish Line Pattern

(b) Finish Line Camera Sample

Figure 32: Finish Line

5.2. PID Controller

A closed loop algorithm controls steering as well as traction motors. All the controllers are implemented with the structure of a time independent PID.

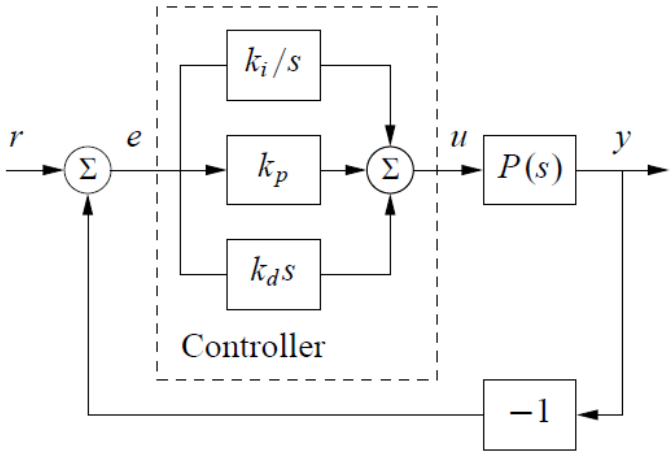


Figure 33: Time independant PID Blocks

The output is barely the sum of three terms, proportional, derivative and integral applied to the difference of the reference input signal and the output, at a given time. The derivative part embeds a filter for high frequencies. Discretization of the PID using Tustin method produces:

$$y_p(k) = e(k)K_p \tag{11}$$

$$y_d(k) = \frac{[e(k) - e(k-1)]K_d + y(k-1)T_l}{T_c + T_l} \tag{12}$$

$$y_i(k) = y_i(k-1) + e(k)T_cK_i \tag{13}$$

An important feature that had to be included in the controller, is the anti reset windup (ARWU), which is important to cope with saturation of actuators. ARWU is a clever strategy to avoid integrator overshooting. As long as the control signal produces a saturation of the output, the integrator term continue to grow and when the input error changes sign, it takes a lot of time before the output exits saturation condition, because the integrator value is high. This problem is reduced by setting a maximum value for the integrator term, but this still produces an overshooting with respect to linear operating conditions. In fact, in linear conditions, the sum of the three components of the PID equals the output. When the output saturates, the sum of the three components is higher than the actual output. Once the input error changes sign, the proportional and derivative terms continue to operate the same way as if saturation has never occurred, but the integral term (which saturates to the output maximum value) is still higher

than in linear conditions, producing an overshooting. ARWU introduces a closed loop that force the sum of the three terms to be always equal to the output, even in saturation condition, by discharging the integral term.

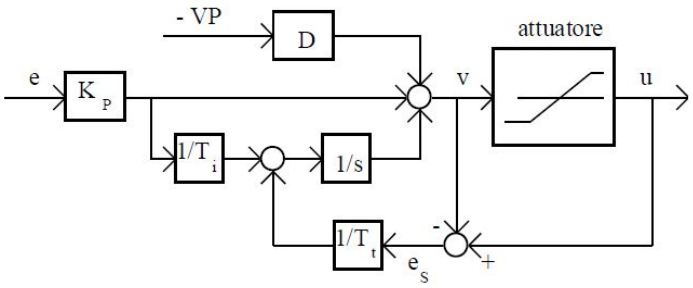


Figure 34: Anti reset windup

The ARWU implementation is shown in Figure 35.

```
// Integration ARWU
pureIntegralValue = auxKi*error*controlPeriodSec + myPid->ValueI;
purePIDOutput = myPid->ValuePS + myPid->ValueD + pureIntegralValue;


outputSatPos = (purePIDOutput + auxKarwu * myPid -> saturationValuePos * controlPeriodSec)
        /(1+auxKarwu* controlPeriodSec);
outputSatNeg = (purePIDOutput + auxKarwu * (myPid->saturationValueNeg) * controlPeriodSec)
        /(1+auxKarwu* controlPeriodSec);
if(purePIDOutput > myPid -> saturationValuePos)
{
    myPid->ValueI += auxKi*error * controlPeriodSec + auxKarwu *
            (myPid -> saturationValuePos - outputSatPos) * controlPeriodSec;
    myPid->outputValue = myPid -> saturationValuePos;
}
else if(purePIDOutput < myPid->saturationValueNeg)
{
    myPid->ValueI += auxKi*error * controlPeriodSec + auxKarwu *
            (myPid->saturationValueNeg - outputSatNeg) * controlPeriodSec;
    myPid->outputValue = myPid->saturationValueNeg;
}
else
{
    myPid->ValueI += auxKi*error  * controlPeriodSec;
    myPid->outputValue = purePIDOutput;
}
return myPid->outputValue;
```

Figure 35: ARWU implementation

5.2.1. Steering Control

The input to the steering controller represents how far the detected line is from the central photodiode of the camera, which sees straight in front of the vehicle. The output is the servo-motor angular position with respect to the rest position, associated with a null steering angle.

The parameters K_p, K_d, K_i and T_i were found experimentally.

Even if its effects are difficult to figure out, the ARWU produced great advantages when exiting from turns, killing dangerous oscillations and contributing to a smooth control of steering.

5.2.2. Speed control of traction motors

The same software has been used to implement a PI controller for speed control of traction motors. This time, the PI parameters were found analytically by reducing the inertia of the vehicle to the axis of the traction motors. Supposing the wheel is not slipping and mass is evenly distributed on the two traction motors, from forces equilibrium, shown in Figure 36, equations 14 are found.

$$\tau = J_{eq}\alpha \qquad J_{eq} = J + \frac{R^2 M_{tot}}{2} \tag{14}$$

where J is the inertia of the rotor and can be neglected. Detailed explanation of this controller can be found in my teammates' theses.

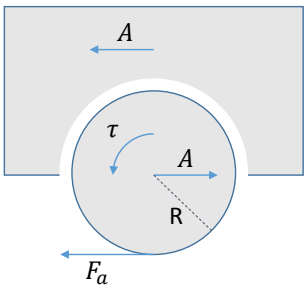


Figure 36: Forces equilibrium for traction wheels

5.3. Differential Algorithm

Since traction motors are completely independent, it was useful to control the speed of rear wheels according to the angle of the servomotor controlling the steering. Differential algorithm relies on the kinematic model of Section 3. The result of this model is a one-to-one correspondence between the servo position and the theoretical steering angle. The curve that defines the steering angle as a function of servo position was interpolated with a second order polynomial. Setting the desired speed for M (Figure 37), which has been chosen arbitrarily, the velocity of the rear wheels can be found using geometrical relations linked to the kinematic model. The position of the centre of instantaneous rotation (x_c, y_c) is given by solving one of the equations

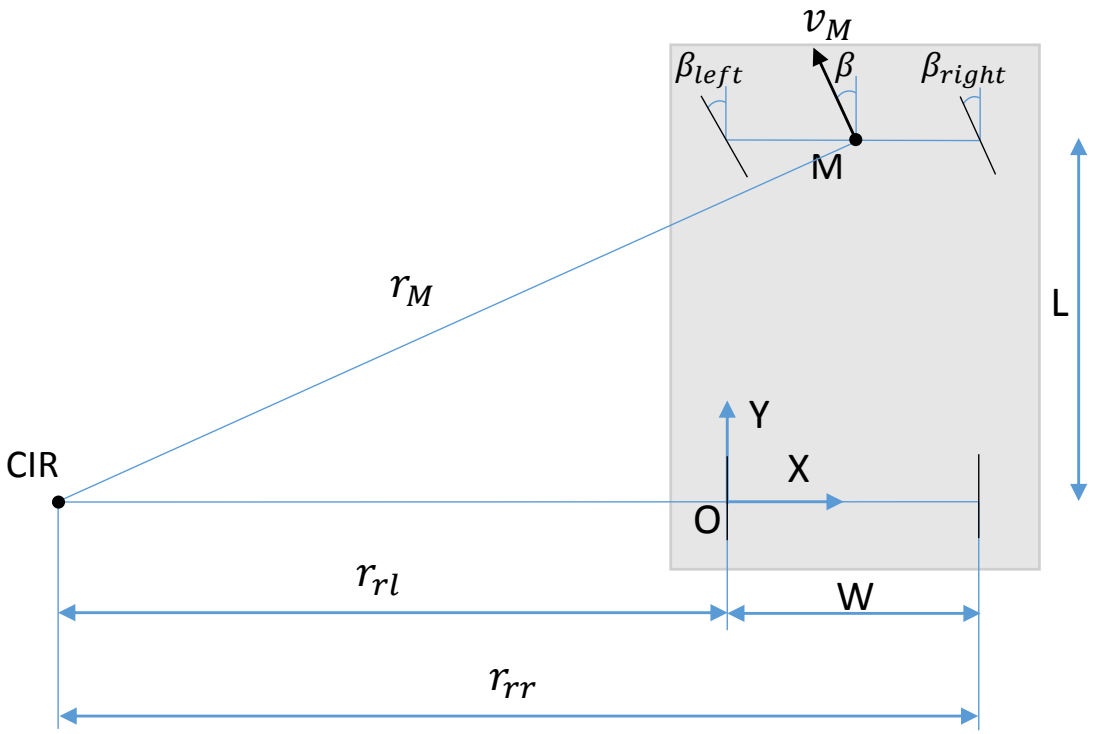


Figure 37: Differential Principle

(3a), (3b). The angular speed of the vehicle is given, for each point P, by:

$$\omega = \frac{v_p}{r_p} \tag{15}$$

with r_p the distance from P to (x_c, y_c). Being r_M, r_{rr}, r_{rl} the distances from (x_c, y_c) to M and to the right and left rear wheels respectively (Figure 37), the speeds of the wheels are found by:

$$r_M = \frac{L}{\sin(\beta)} \qquad \omega = \frac{v_M}{r_M} \tag{16}$$

$$v_{rr} = \omega r_{rr} \qquad v_{ll} = \omega r_{rl} \tag{17}$$

The angular speed of the wheels is computed by $\Omega = v_{wheel} R_{wheel}$

The active control of speed of the rear wheels forces the vehicle to work in "ideal-like" conditions, that are not guaranteed by the steering only because of the non ideal behaviour of the steering mechanism and of the tyres. To stress this point, say that for a given servo angle the actual trajectory of the vehicle doesn't match that computed by the model. If motors are off, the velocities of the wheels are different from those worked out from the model. Turning the motors on, the rear wheels are forced to their ideal speeds. As a result, the actual trajectory in this case will be closer to the ideal one. For what it concerns with implementation matters, data about kinematics are updated periodically and stored in a dedicated structure which is accessible to any function needing these information, including the differential algorithm.

5.4. Track recognition algorithm

This piece of software was not running at EMEA final because it didn't reach a good robustness level: it should be integrated with sensors of motion in order to give reliable results. Nevertheless, it is interesting because it leads to a higher understanding of the track and adds possibilities in terms of control strategies.

The problem leading to the development of this solution is that, with a line scan camera, you can only see the position of a point belonging to the black line at a given time, but you have no idea of the shape of the track you are running on. Our control algorithm allows the car to drive trying to keep the vehicle as close as possible to the black line, judging the relative distance of the vehicle from it. By contrast, a human driver sees the whole track coming after. This allows him to make considerations about the track such as curvature radius and length of a turn, estimate distances and so on. The distance at which the camera sees can be arbitrarily set. The idea is using subsequent acquisitions from the line scan camera to create a bi-dimensional image of the slice of the track extending from the front wheels to the range of the camera, at a given time.

The problem consists of transforming points measured with respect to a moving reference frame to a fixed one. The moving reference frame is the car intrinsic system. The fixed one is arbitrary. The blue rectangle in Figure 38 represents camera position at different times. With reference to Figure 38, let's focus on one of the two instants, for example the second. The point detected by the camera is the intersection of the orange line, representing the width of camera sight W, and the black line on the track. The coordinates of that point in the system of the camera $(x_c^{''}, y_c^{''})$ are:

$$x_c^{''} = \frac{pixel\ index - 64}{N_{pixel}} \cdot W \tag{18}$$

$$y_c^{''} \equiv R \tag{19}$$

where $pixel\ index$ is the index of the photodiode corresponding to the centre of the line, R is the range of the camera and N_{pixel} the total number of photodiodes. In polar coordinates:

$$R_c^{''} = \sqrt{x_c^{''2} + y_c^{''2}} \tag{20}$$

$$\Theta^{''} = atan2(x_c^{''}, y_c^{''}) \tag{21}$$

Expressing the point $(x_c^{''}, y_c^{''})$ in the fixed reference frame:

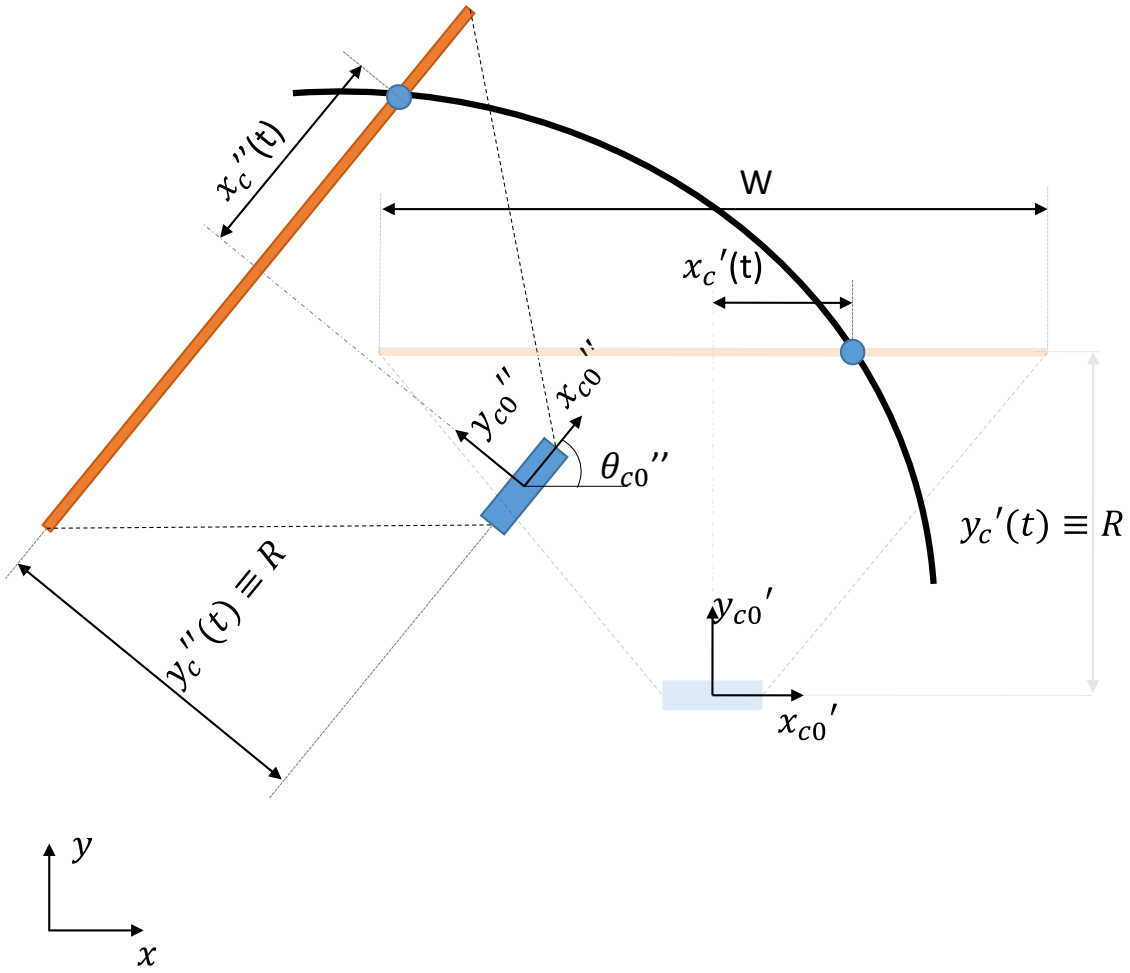


Figure 38: Intrinsic reference system

$$x_c = R_c'' \cos(\theta'' + \theta_{c0}'') + x_{c0}'' \tag{22a}$$

$$y_c = R_c'' \sin(\theta'' + \theta_{c0}'') + y_{c0}'' \tag{22b}$$

$(x_{c0}'', y_{c0}'', \Theta_{c0}'')$ are the coordinates of the moving frame with respect to the fixed one. $(x_{c0}'', y_{c0}'', \Theta_{c0}'')$ can be written with respect to position in the previous sample $(x_{c0}', y_{c0}', \Theta_{c0}')$.

Defining

$$\alpha = \Theta_{c0}'' - \Theta_{c0}' \tag{23a}$$

$$\Delta x = x_{c0}'' - x_{c0}' \tag{23b}$$

$$\Delta y = y_{c0}'' - y_{c0}' \tag{23c}$$

the vector $(\alpha, \Delta x, \Delta y)$ can be estimated by considering the kinematics of the vehicle. The best we can do, is to shrink the time interval between two subsequent observations to the camera

sampling period T_s. From Figure 39, it follows that:

$$\omega = \frac{v}{r_M} \tag{24a}$$

$$\alpha = \omega T_s \tag{24b}$$

$$\Delta x = -\omega T_s r_M [1 - cos(\alpha)] \tag{24c}$$

$$\Delta y = \omega T_s r_M sin(\alpha) \tag{24d}$$

As a convention, $\omega > 0$ stands for a counter clockwise rotation. β is the steering angle defined

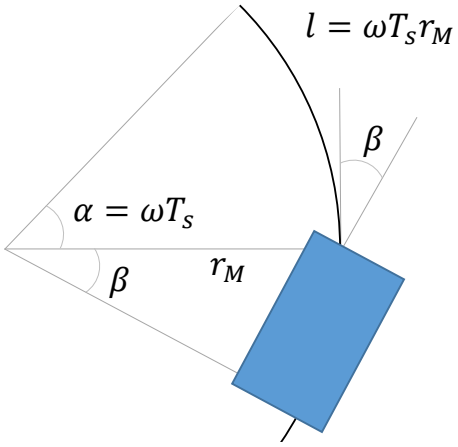


Figure 39: Position estimation

in Section 3. In our case $\omega = \omega(\beta)$ and $r_M = r_M(\beta)$ through the relations of the kinematic model: here is the limit of this algorithm. In particular, problems occur because of tyres slips becoming non negligible at high speeds. A solution to this problem should be integrating the algorithm with a gyroscope sensor in order to measure ω, thus resulting unaffected by the inaccuracies of the model. Figure 40 shows an example of successful estimations of a 90 degree turn and a chicane at low speed.

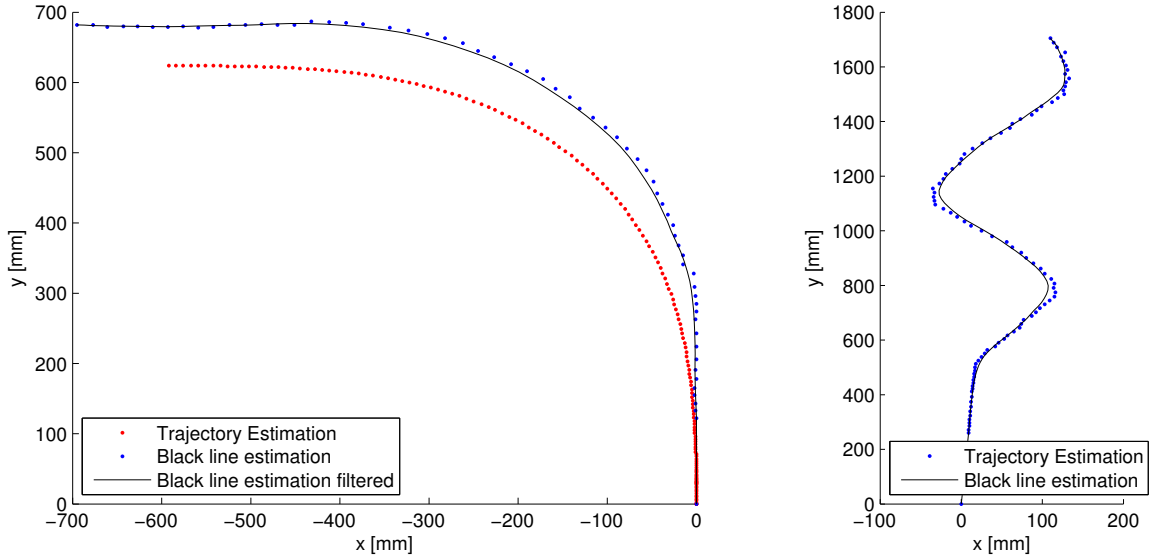
A filter has been implemented in order to obtain a smooth image. Its implementation consists of a discretization of a first order low pass filter using Tustin approximation. Both x and y values are filtered. The cutoff frequency can be set by considering that the maximum frequency component for x and y arises when the car is running at the maximum speed through a turn of minimum radius. This can be figured out by looking at the parametric description of a turn.

$$x = R\cos(\omega t) \tag{25a}$$

$$y = R\sin(\omega t) \tag{25b}$$

$$\omega = \frac{v_{max}}{R} \tag{25c}$$

For the challenge rules, the minimum radius of curvature of a turn is $R = 0.5\ m$. Considering these to be the highest harmonics contained in $x(t), y(t)$ and knowing the maximum speed of



(a) 90 degrees turn estimation

(b) Chicane estimation

Figure 40: Track estimation samples at low speed

the car to be about 5 m/s, $\omega = 10\ rad/s$ is worked out. To be even more cautious, a cutoff frequency $\omega = 20\ rad/s$ was chosen.

Figure 41 shows an example of sampling at high speed: the car is decisively understeering, thus not producing a correct estimation.

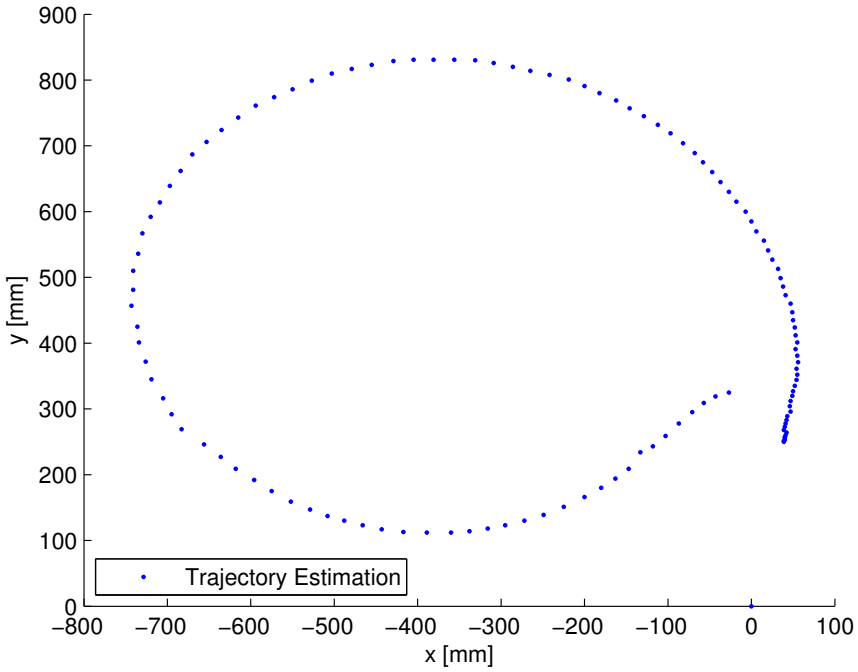


Figure 41: 180 degrees turn estimation at high speed

5.5. Chicane Detection Algorithm

As an application of the track analysis, an attempt to recognize chicanes was made. Of course, as track recognition was not precise enough, this algorithm resulted very unstable and dangerous, thus not taking part to the software running at races. Its implementation is pretty laborious, but the idea is simple. Chicanes are approximated a sequence of two segments with a difference of orientation of γ from one to another, as shown in Figure 42. When the algorithm detects such a pair of segments, it turns on a very low pass filter between the camera output and the servo controller: the controller will have the illusion that the chicane has never been on the track.

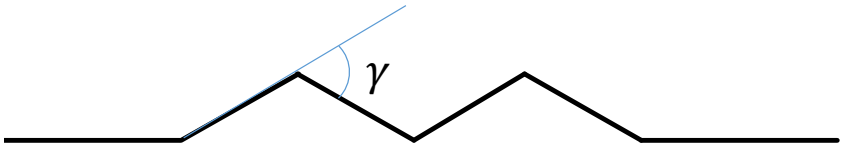


Figure 42: Chicane simplification

6. Conclusions

The Freescale Cup University Programme has been a very interesting experience. It provided us the occasion to deal with a complex, unknown system, leading to an intense activity of bibliographical research. It was a way to deepen and apply our knowledge, develop teamwork abilities, compete with universities from all over the world and stimulate creativity. I talked more about ideas than implementations in the hope of focusing the core of solutions and I invite the reader to give a glance at the referenced documents for further details. I finally recommend to take a look at my teammates theses to have a complete overview of our work for the Freescale Cup. I hope this dissertation will be useful for students who will join the next editions of The Freescale Cup, both from my and other universities.

References

[1] F. Biral, Lectures of Vehicle Dynamics and Control, University of Trento, 2013

[2] Conrad, NIMH 2400mAh 1.2 V Cells Data Sheet

[3] Conrad, NIMH 2400mAh 1.2 V Cells Data Sheet

[4] Freescale Semiconductors, MPC5604B/C Microcontroller Reference Manual, Rev. 8.1, 05/2012

[5] Freescale Semiconductors, MPC5604B/C Microcontroller Data Sheet, Rev. 11.1, 10/2013

[6] Freescale Semiconductors, Motor Diver Rev A, August 2011

[7] Futaba, S3010 - Standard High-Torque BB Servo Datasheet

[8] Honeywell, HOA1405 Reflective Sensor Data Sheet

[9] R. C. Jaeger, T. N. Blalock, Microelettronica, quarta edizione, Milano, McGraw-Hill,2009

[10] B. W. Kernighan, D.M. Ritchie, C Programming Language, Englewood Cliffs, New Jersey, Prentice Hall Software Series, 1988

[11] G. Lilli, Hardware e Software di Controllo del Veicolo "Freescale Cup", Dipartimento di tecnica e gestione dei sistemi industriali, Università di Padova, 2013

[12] G. Marro, Controlli automatici, quinta edizione, Bologna, Zanichelli, 2004

[13] F. Meneguzzo, Freescale Cup: Studio del Controllo dello Sterzo di un Veicolo Autonomo, Dipartimento di tecnica e gestione dei sistemi industriali, Università di Padova, 2013

[14] MIT Electric Vehicle Team, A Guide to Understanding Battery Specifications, December 2008

[15] R. Oboe, Dispense del corso di Controlli Automatici, Dipartimento di tecnica e gestione dei sistemi industriali, Università di Padova, 2014

[16] OSRAM Opto Semiconductors, OSLON LUW CN5M Data Sheet Version 1.0, July 2013

[17] G. Ricci, M.E. Valcher, Segnali e Sistemi, Padova, Libreria Progetto, 2004

[18] A. Rindi, S. Papini, L. Pugi, J. Auciello, M. Ignesti, Appunti del Corso di Meccanica del Veicolo, Università di Firenze, 2012

[19] T. Scaletta, Freescale Cup: Studio del Controllo della Trazione di un Veicolo Autonomo, Dipartimento di tecnica e gestione dei sistemi industriali, Università di Padova, 2013

[20] Standard Motor, RN260-CN-18130 Data Sheet, August 2003

[21] A. Stella, Regolatori e Sistemi di Controllo per una Smart Car, Tesi di laurea, Dipartimento di tecnica e gestione dei sistemi industriali, Università di Padova, 2013

[22] TAOS, TSL1401CL 128 × 1 Linear Sensor Array with Hold Data Sheet, July 2011

[23] M. Zigliotto, Dispense del corso di Fondamenti di Macchine e Azionamenti Elettrici, Dipartimento di tecnica e gestione dei sistemi industriali, Università di Padova, 2014