



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

STUDY OF AUTOMATIC CONTROL ALGORITHMS FOR A SLOT CAR

Author: Matías Nicolás Correa Barceló

Advisor: Ramon Sarrate Estruch



Bachelor's Degree in Industrial
Electronics and Automatic Control
Engineering

INDEX:

1. INTRODUCTION	9
1.1. Previous documentation	10
1.2. Purpose of the project	11
1.3. Resources	11
1.3.1. Workspace	12
1.3.2. Slot circuit	12
1.3.3. FireWire camera	14
1.3.4. Varifocal lens	16
1.3.5. Central system (PC)	17
2. ORIGINAL ANALOG SYSTEM	18
2.1. Power source	18
2.2. Analog controller	19
2.3. Motor	20
3. COMPUTER VISION SYSTEM	23
3.1. Introduction to computer vision	23
3.2. Computer vision. Typical architecture	24
3.3. Getting data with computer vision	26
3.4. Now, how do I have to do to get slot car position?	26
3.4.1. Taking pictures	27
3.4.2. Region of interest (ROI)	27
3.4.2.1. Sizing ROI	29
3.4.2.2. Circuit zoning	32
3.4.3. Get reference picture	35

3.4.4.	Colour filter	35
3.4.5.	Tracking car system	37
3.5.	Adjustments workspace and lighting	38
4.	PC-RACETRACK INTERFACE	40
4.1.	PC-Racetrack interface. What's inside and how it works?	41
4.2.	Microchip software	45
5.	JSLOT PROJECT. CLASSES AND LIBRARIES	47
5.1.	LTI-CIVIL	48
5.2.	Rxtx Comm	48
5.3.	JSlot Screen	49
5.4.	Camera Access	50
5.5.	Car Position	51
5.6.	Calculator	52
5.7.	SerialPort Access	54
5.8.	Lighting Adjustment	55
5.9.	Chronometer	55
6.	HOW JSLOT WORKS?	56
7.	TESTS AND ADJUSTMENTS	63
7.1.	Hardware	63
7.1.1.	PC-Racetrack interface device	64
7.1.2.	Camera	68
7.2.	Software	70
7.2.1.	Resized ROI	70

8. AUTOMATIC CONTROL ALGORITHM	78
8.1. Scan time	78
8.1.1. Image processing task	79
8.1.2. Race data calculation task	82
8.1.3. Send an integer data to the control device	84
8.1.4. Refresh GUI	86
8.1.5. Conclusions about Scan time	87
8.2. Irregular ROI zoning	89
8.3. Data sent to PIC-Speed Ratio	92
8.4. Speed limit for irregular zoning	98
8.5. ROI size to irregular zoning	100
8.6. Look-up table	101
9. USER GUIDE	103
9.1. Settings	103
9.2. Calibration wizard	112
9.3. Getting started	115
9.4. Troubleshooting	123
10. CONCLUSIONS	133
11. FUTURE RESEARCH TRENDS	135
12. BIBLIOGRAPHY	137

FIGURES INDEX:

<i>Fig.1 Super series NINCO slot circuit.</i>	13
<i>Fig.2 Bayer DBK21AU04 Camera.</i>	15
<i>Fig.3 Varifocal lens with CS Mount.</i>	16
<i>Fig.4 Panel racetrack connections.</i>	18
<i>Fig.5 NINCO 55 plus Analog Controller.</i>	19
<i>Fig.6 Wiring diagram of the analog controller.</i>	20
<i>Fig.7 Nc-5 speed motor.</i>	21
<i>Fig.8 Shunt motor running. Circuit diagram and electromechanical waveform.</i>	21
<i>Fig.9 Nc-5 speed motor with 22 ohms resistance.</i>	22
<i>Fig.10 Equivalent circuit diagram racetrack.</i>	22
<i>Fig.11 ROI concept example.</i>	28
<i>Fig.12 Location of the singular circuit regions.</i>	33
<i>Fig.13 Circuit zoning.</i>	34
<i>Fig.14 Lexus 430 sc orange (original version).</i>	35
<i>Fig.15 Lexus 430 sc orange (current version).</i>	36
<i>Fig.16 Binarization concept.</i>	36
<i>Fig.17 Flowchart of the tracking car system.</i>	37
<i>Fig.18 Interface to calibrate the slot car position.</i>	38

<i>Fig.19 PC-Racetrack interface. Outside.</i>	40
<i>Fig.20 PC-Racetrack interface. Blocks diagram system.</i>	41
<i>Fig.21 PC-Racetrack interface. Circuit diagram.</i>	42
<i>Fig.22 Power supply when occurs positive transition of the PWM signal</i>	43
<i>Fig.23 Power supply when occurs negative transition of the PWM signal</i>	43
<i>Fig.24 Testing. PWM waveform when load is constant.</i>	44
<i>Fig.25 PIC software flowchart.</i>	45
<i>Fig.26 JSlot GUI.</i>	49
<i>Fig.27 JSlot flowchart.</i>	56
<i>Fig.28 Connect button event flowchart.</i>	57
<i>Fig.29 Disconnect button event flowchart.</i>	58
<i>Fig.30 Database manager events flowchart.</i>	59
<i>Fig.31 Circuit menu item event flowchart.</i>	60
<i>Fig.32 Lighting menu item event flowchart.</i>	61
<i>Fig.33 Help menu item event flowchart.</i>	62
<i>Fig.34 Control device. PWM waveform signal.</i>	65
<i>Fig.35 Control device. Optodriver waveform signal.</i>	65
<i>Fig.36 Control device. Gate MOSFET waveform signal.</i>	66
<i>Fig.37 Repeteability study of capture time.</i>	69

<i>Fig.38 Performance test of slot car location finder for ROI size 45 px.</i>	71
<i>Fig.39 Performance test of slot car location finder for ROI size 60 px.</i>	74
<i>Fig.40 Performance test of slot car location finder for ROI size 84 px.</i>	77
<i>Fig.41 Scan cycle map.</i>	79
<i>Fig.42 Time spent by image processing task to locate the slot car.</i>	80
<i>Fig.43 Time spent to calculate race data using calculator class.</i>	82
<i>Fig.44 Time spent to calculate race data using calculator class. Boxplot.</i>	83
<i>Fig.45 Time spent to sent data to the controller device (PIC).</i>	84
<i>Fig.46 Time spent to sent data to the controller device (PIC). Boxplot.</i>	85
<i>Fig.47 Time spent to refresh the visual interface.</i>	86
<i>Fig.48 Time spent to refresh the visual interface. Boxplot.</i>	87
<i>Fig.49 Scan time.</i>	88
<i>Fig.50 ROI size using the results of Scan time.</i>	90
<i>Fig.51 Irregular zoning idea.</i>	91
<i>Fig.52 Study of the speed-SP ratio.</i>	95
<i>Fig.53 What you need to use control and monitoring slot system?</i>	104

<i>Fig.54 Interface PC-SlotCircuit drivers (CDC drivers) (1).</i>	<i>106</i>
<i>Fig.55 Interface PC-SlotCircuit drivers (CDC drivers) (2).</i>	<i>107</i>
<i>Fig.56 Interface PC-SlotCircuit drivers (CDC drivers) (3).</i>	<i>108</i>
<i>Fig.57 Interface PC-SlotCircuit drivers (CDC drivers) (4).</i>	<i>108</i>
<i>Fig.58 Interface PC-SlotCircuit drivers (CDC drivers) (5).</i>	<i>109</i>
<i>Fig.59 Interface PC-SlotCircuit drivers (CDC drivers) (6).</i>	<i>110</i>
<i>Fig.60 Java web.</i>	<i>111</i>
<i>Fig.61 Workspace adjustment.</i>	<i>113</i>
<i>Fig.62 JSlot visual interface. Components.</i>	<i>120</i>

1. INTRODUCTION

As the technology advances, more and more areas can use these technological improvements favorably. So much so, that slot racing world has evolved from using simple electronic command systems to include increasingly sophisticated improvements.

What's more, the amount of professional slot car racing has increased significantly over the years. For this reason it is considered important to use technological advances to improve slot racing. It is very important keep in mind that a small improvement in operation mode of driving can provide substantial benefits.

Therefore, this project aims to develop a platform for a slot car control using artificial vision. The control system priority will carry out the most efficient driving at all times keeping the car safety. Slot car control system will be managed from a Java application. This Java application, besides to managing slot car control, will be used to provide useful racing data to the user.

The monitoring data task can provide to the user some interesting data, "*in situ*" or "*a posteriori*". These data can include diverse types: top speed, average speed, fastest lap, total race time, energy use, car position, and so on.

1.1. Previous documentation

Part of this project is based on the integration of the various studies conducted in previous projects. It is for this reason that during their development can be found, in some sections of this document, explicitly mention to conclusions drawn from those projects.

Firstly was performed, in 2007-2008, a mini project titled "*DISEÑO DE UNA PLATAFORMA DIDÁCTICA DE MONITORIZACIÓN Y CONTROL DE UNA CARRERA DE SLOT*". It was a work of three industrial electronics and automatic students, in the ETSEIAT School. In the bibliography, this document is referenced Ref: BFP.01.

Later, during 2009-2010 academic year, Mr. Daniel Sanchez Rodriguez made his Final Project to graduate at industrial electronics engineering. Titled "*DISSENY D'UNA PLATAFORMA DIDÀCTICA DE MONITORATGE I CONTROL D'UN CIRCUIT DE SLOT*" was responsible for developing a Java application through which you can get some information very useful in performing slot control and monitoring. In the bibliography, this document is referenced Ref: BFP.02.

Then, during 2011-2012 academic year, was made a Bachelor's Thesis by Mr. Edgar Jiménez López, entitled "*DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE UN CIRCUITO SLOT*". Through this project the author was graduated as industrial electronics engineer. During this project was designed a device for converting digital data from the PC (where is run the application responsible to take the slot car control) into an analog voltage signal. The voltage signal generated is the energy source used by slot car. In the bibliography, this document is referenced Ref: BFP.03.

1.2. Purpose of the project

The purpose of this project is to implement a Java application that can perform an efficiently control algorithm for the Slot Car. This control application must keep safe the car at all times. Moreover is very important to make the path in the shortest time as possible. Besides this is imperative not only make control of the slot car, but also monitoring some data that may be important in a race. These data type relate to maximum speeds, lap time, car position, top speed, average speed of the race, and so on.

1.3. Resources

When deciding which items are to be used for Slot Car control, you have to know and keep in mind the your design needs, the product features on the market that are compatible with the project and mainly the elements that are already at your complete disposal and can use them immediately.

In the following sections you can see the most important elements that will be used on this project.

1.3.1. Workspace

The workspace conditions are a fact to keep in mind because it affects the features of the other elements. If some of the components of the control assembly are not designed according to the workspace conditions may be affected the correct operation of the system.

The workspace is located in TR11 building of the UPC Terrassa, specifically in Industrial Computing Laboratory. The race track is located on a square table 160x160 cm. The camera that is responsible for taking the necessary images for control and monitoring of the vehicle is located on the roof of the laboratory. 2m is the height distance between the camera and the circuit. Laboratory lighting conditions can be adjusted by combining pull up/down of two blinds and switching on/off of four rows of fluorescent lights.

1.3.2. Slot circuit

The Circuit Kit NINCO Super Series has been selected for use in this project. This circuit was in the Industrial Computing Laboratory and therefore we do not have to buy. This is a conventional set of tracks for 1/24 and 1/32 scale models. The driving of the vehicle on the circuit is performing with the 55 Plus analog controller. Lexus 430 sc orange 1/32 with nc-5 Speed motor is the car that we will use in the development of this project. Later we will study in detail this car and their motor.

The next picture that you can see shows the circuit kit that we will use in this project, Circuit Kit NINCO Super Series:



Fig. 1. Super series NINCO slot circuit

In order to select the slot circuit best suited to the needs of this project was decided looking for the different options that we offered the market.

The comparative study of the options offered by the market currently is done in the document referenced in the bibliography Ref: BFP.01. In addition in that reference we can see in detail the reasons for the choice of NINCO circuit.

In conclusion, we can say that the reasons for the choice of NINCO circuit are set out in the following list:

- Circuit with isolated tracks and more depth guide. This way you get to avoid short-circuits, reduces wear of the rails and get a more stable and higher vehicle speeds. The depth of the tracks is 6.5 mm.

- The rails of the tracks are completely covered and therefore the introduction of dirt inside decreases. In this way we can get greater speeds than with other types of circuits and we can achieve more stability.
- The width of the tracks is higher than the other circuits on the market. This fact allows doing races with different car models and with different scales.
- Easyclip system facilitates its assembly and subsequent changes in the distribution of parts that we want to do. Moreover, being a strong and secure mounting, enables the user can ignore the correct fastening of the track segments because cause an uncoupling of parts.
- And last but not least, NINCO circuit has a return system to facilitate braking competition cars.

1.3.3. FireWire camera

When controlling a slot car manually with the analog control, vision is used as a way to obtain information about the state of the race. After that, When the brain processes visual information that reaches the circuit is able to determine the speed at which the vehicle must travel at all times. That is the way in which a human being performs closed loop control of the drive slot.

In the project at hand, in charge of making the control loop is the machine and not the driver of the car. It is therefore must have a member

who carries out the equivalent function to the view to provide information to the controller located on the machine. This element is the camera.

A wrong choice of the type of camera to be used can cause the control system is seriously affected and cannot succeed. It is for this reason that the search for the camera that best suits the needs of the project has to be done slowly and carefully analyzing the characteristics of each of the options. Document Ref: BFP.02, in the bibliography, you can see a business case where it is concluded that the most appropriate camera for the application to be developed is DBK21AU04 Bayer USB Camera Imaging Source. Is presented below the aforementioned camera:



Fig. 2. Bayer DBK21AU04 Camera

Bayer DBK21AU04 camera has three important characteristics: medium resolution, high and variable ratio sampling rate and "Bayer" filter. The following list enumerates its main aspects:

- USB Connection (400 Mbit/s)
- 1/4" progressive Scan CCD.
- Fixed resolution 640x480.
- Configurable Frame Rate depending on the video format:

- 640x480 UYVY @ 30, 15, 7.5, 3.75 fps.
- 640x480 BY @ 60, 30, 15, 7.5, 3.75 fps.
- Sensitivity: 0.5 lx to 1/30s. Gain: 20 dB.

1.3.4. Varifocal lens

Since the Bayer BDK21AU04 camera does not incorporate any type of lens, we have to choose a camera lens that fits the needs of the project to make the shots necessary to Slot car control.

The author of the Bachelor's Thesis referenced in the bibliography Ref: BFP.02 presents a set of calculations in order to determine the needed features that is imperative include in the used lens. In short, on the document cited above is concluded that a data to keep in mind is the fact that "focal length" of selected lens is 2.82 mm approximately. Beside this, any needed features are: working distance at 2 m, height object 1700 mm and CS connection in order to be compatible with the camera.

The selected lens is a generic varifocal with CS Mount and 2.8-12 mm focal length.



Fig. 3. Varifocal lens with CS Mount

The most important characteristics that are provided by the manufacturer are presented below:

- Focal length: 2.8-12 mm.
- View angle: 80-20°.
- Manual iris.
- Image format 1/3.
- CS Mount Connection.

1.3.5. Central system (PC)

The PC is in front the circuit, in Industrial Computing Laboratory, is one of the most important elements for the correct operation of the system. This computer is used to do testing, image processing, production control and monitoring software, and so on.

Here are most important characteristics of the PC:

- Intel Core2Duo E7300 @ 2.66 GHz.
- 2 GB DDR2 667 Mhz.
- Windows XP Professional.
- Software: NetBeans, Java Development Kit (JDK), IC Capture, GIMP.

2. ORIGINAL ANALOG SYSTEM

To better understand the changes that will be made in the circuit when we put the new control system, has decided to do an explanation of the original analog system implemented. In the following sections you will find an overview of the main system components. Such components are: power source, analog controller and slot car motor.

2.1. Power source

The power system is based on a circuit power source of 14.8 V DC and 1A maximum load. The system has a switch to change the running direction of the vehicle if you wish. This system also offers the option of connecting two different power sources for the current peaks of a lane does not affect the other lane.



Fig. 4. Panel racetrack connections

2.2. Analog controller

Order to perform slot car driving the user has to use NINCO 55 plus Analog Controller. Depending on the variation of the pressure on the trigger, the driver can control the voltage at the motor and therefore the speed of the car. This controller is equivalent to a potentiometer where you can change the resistance. According to tests in previous projects the equivalent electrical resistance of the Analog Controller is 53.6Ω .



Fig. 5. NINCO 55 plus Analog Controller

It is also important to note that the Analog Controller is connected to the circuit through a 3.5 mm Jack Connector.

The potentiometer changes the motor voltage. When we move the slider trigger changed the value for the electrical resistance. If the trigger is off, the resistance is with a low. When that happens the slot car does not move because the voltage between the tracks, and therefore between motor terminals, is zero. By contrast if we press the trigger electrical resistance changes. As the trigger moves the electrical resistance increases, thus making the voltage increases on the

tracks and the vehicle starts to move, to the moment that the maximum voltage 14.8 V is achieved.

The picture below represents the operation scheme of the system:

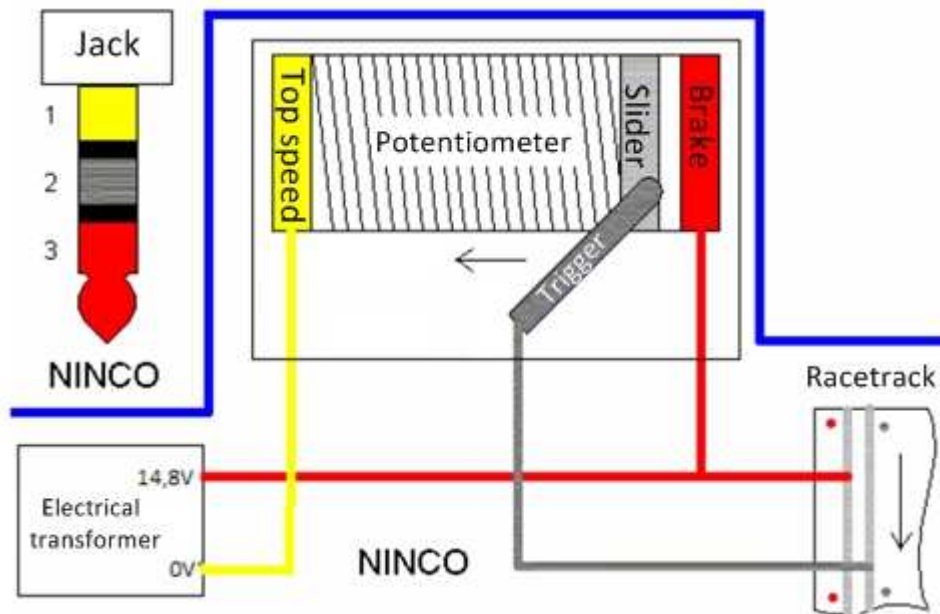


Fig. 6. Wiring diagram of the analog controller

2.3. Motor

The slot car motor that we will use is responsible for transmitting the angular motion to the wheels through the bevel gear system. Lexus 430 sc orange slot car uses a high revolution DC motor. The motor to be used in this project is

known as Nc-5 Speed motor, with 12/32 bevel gear ratio. Document Ref: BFP.01, in the bibliography, you can see more details of the motor.

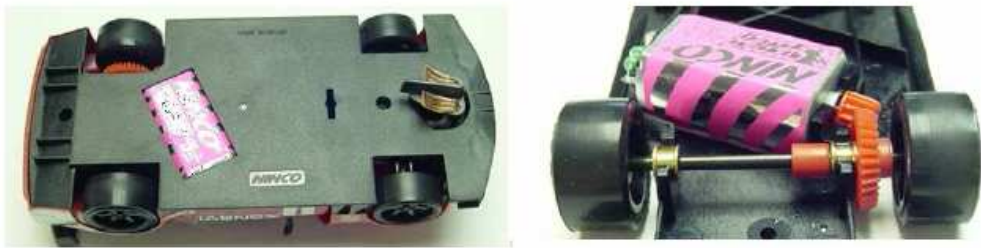


Fig. 7. Nc-5 speed motor

Following you can see the most important characteristics of the NC-5 Speed DC motor (Shunt):

$$V_N = 14,8V \quad \omega_0 = 20000 \text{ rpm} \quad I = 140\text{mA} \quad M = 90\text{g}\cdot\text{cm}.$$

Is shown below graphically the running of a shunt motor:

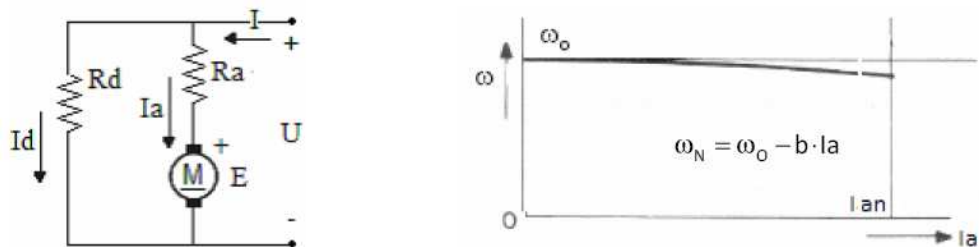


Fig. 8. Shunt motor running. Circuit diagram and electromechanical waveform

Where ω_n is the nominal angular speed, ω_o is the angular speed into the void, I_a is the armature current and b is a constant motor.

In order to prevent the occurrence of too high current overshoot has decided to put an electrical resistance of 22 ohms wired in series with the motor. In this way we get starting currents less than 500 mA.



Fig. 9. Nc-5 speed motor with 22 ohms resistance

In the annexed picture below you can see the equivalent circuit diagram of the system:

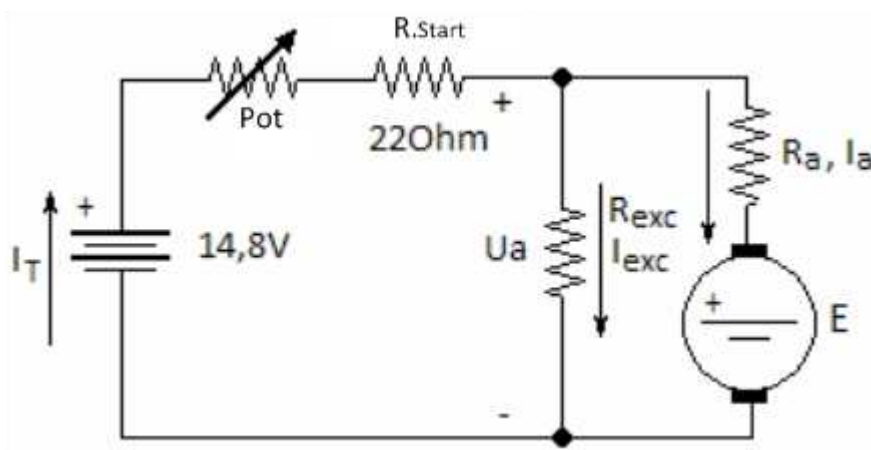


Fig. 10. Equivalent circuit diagram racetrack

3. COMPUTER VISION SYSTEM

During this chapter you can see a generalized concept of how is the operation way of the computer vision system. You can find more information about this section on Document Ref: BFP.01, in the bibliography.

For acquire some parameters required to make the Slot Car control will be used a computer vision system. *DBK21AU04 Bayer USB Camera Imaging Source* will be responsible for taking the pictures that contain the needed information. When the camera has completed its task, the system must do the image processing task.

3.1. Introduction to Computer Vision

Computer vision is the science of endowing computers or other machines with vision, or the ability to see. But what exactly does it mean to see? Most computer vision scientists would agree that seeing is more than the process of recording light in a form that can be played back, like the recording of a video camera. But what, exactly, is needed in addition to the detection or recording of light in order to say that a device, be it natural or manufactured, is seeing?

Perhaps we wish to say that vision is the interpretation of images that leads to actions or decisions, as in the navigation of an autonomous robot. But would we then exclude as vision the process of gazing at the night sky or a beautiful ocean vista, processes in which we may have no intention of making any decision? Processes such as recognition, interpretation, learning, or just enjoyment may be occurring when we see that have no immediate bearing on a decision. On the other

hand, something we see may affect a decision we make years later. How do we then know if we are currently seeing or not?

Since vision is a core component of intelligence, its definition encounters many of the same philosophical issues raised when trying to define intelligence itself. Like intelligence, there are many components to vision, including but not limited to memory, retrieval, reasoning, estimation, recognition, and coordination with other senses. It would be odd to insist that all of the above elements be present before we would consider a system to have some degree of vision. At the same time, a system with only one of these abilities might not be promoted to the rank of having vision. To some extent, we define vision by the familiar processes of our own visual systems, and thus, there may be some subjective judgement about the degree to which a system can see by comparing it to our own capabilities.

Erik G. Learned-Miller, "Introduction to Computer Vision" (University of Massachusetts, Amherst)

3.2. Computer Vision. Typical architecture

The most important parts in a computer vision system are firstly the acquiring methods and finally the processing image task. As regard to the acquiring methods, this process includes lighting subsystem, image capture and data transfer to the computer. However, regarding to the processing image task, after the PC has received data, this information is transformed into high-level data through the use of algorithms. The high-level data can be used to send to the device that needs it such as: robot, PLC, or any device where the correct operate is based on artificial vision.

Therefore, we can say that typical computer vision architecture comprises:

- **Lighting subsystem:** This subsystem takes care to provide the necessary luminosity so that the Artificial Vision System can take the pictures correctly. Some components of this subsystem are lighting lamps, halogen lights, light filters, lasers and so on.
- **Pickup subsystem:** are transducers that convert the light reflected from the objects into an electrical signal. This subsystem gathers CCD cameras.
- **Acquisition subsystem:** At this stage, electrical signals coming from cameras make up the video signal. The video signal can be digital signal or analog signal (CCIR, RS170, PAL, NTSC...). In the case of an analog signal, the system has to make a sampling and quantization step for later use.
- **Processing subsystem:** At this stage, different types of algorithms are used to getting information that is useful to the system. The type of information depends to the system characteristics that you want control. Such algorithms implemented a number of changes in pictures taken to get the necessary information.
- **Peripherals devices subsystem:** This subsystem contains all devices to receive high-level information. This group contains any device that needs the information obtained by taking pictures to carry out partial or total operation.

3.3. Getting data with computer vision.

When it comes to setting up a control system based on computer vision, the designer has to keep in mind the type of data that the system has to obtain through the camera.

Specifically, when working in the Slot Car Control design, you should to consider two parameters: vehicle position and speed. Car speed can be calculated using current position, previous position and elapsed time. For this reason, the speed is not considered a type of information that the camera has to provide. Therefore, the computer vision system only has to provide the vehicle position at every moment.

3.4. Now, how do I have to do to get Slot Car position?

Through this section is to explain which is the way, which are the steps to follow, for acquire the position of the Slot Car. Moreover, note that in this section only there is a brief explanation about how you have to do to get Slot Car position. On the other hand, if you want more information about this you can see the document referenced in the bibliography Ref: BFP.02.

3.4.1. Taking pictures

One of the most important components in taking pictures is the communication bus with camera. The communication camera is via a communication buffer with specific characteristics for processing image. The camera sends taking pictures to the buffer and then processing image task can use these whenever it want.

3.4.2. Region Of Interest (ROI)

According to reached conclusions in the final project “DISSENY D’UNA PLATAFORMA DIDÀCTICA DE MONITORATGE I CONTROL D’UN CIRCUIT DE SLOT” referenced in the bibliography Ref: BFP.02 about camera capture time, is known that to achieving an enough picture quality we also have a very brief capture time.

As you can see in the aforementioned document, the lowest time between a picture and the next is 33 ms because the camera takes pictures to 30 FPS (frames per second). On the other hand, if image resolution is 640 x 480 pixels, processing image task take 600 ms to extract data. Because of all this, you may come to believe that it is impractical to do the Slot Car control correctly in this way. The solution to this problem: use only the picture region that interests us, i.e. using ROI.

ROI (Region Of Interest) is to define a particular area of the overall picture. In the ROI has to be the region and information that is important for the control system. By this method, using a small picture region, we can

achieve decrease processing time and therefore sampling time of the control system. In contrast, if ROI is not correctly defined, the system may lose very important and useful data.

Below is a graphical description of the ROI concept:



Fig. 11. ROI concept example

In the case at hand, the author of the final project Ref: BFP.02 has decided to divide slot circuit in a few regions. These regions have the same extent that ROI. In order to know the slot car position, the system will check if car is in any circuit regions or not and return a binary answer. This task is done in all regions and thus can know the slot car position.

3.4.2.1. Sizing ROI

It is very important to estimate correctly the ROI size. If ROI size is too large, the system wasted unnecessarily so much computation time. On the other hand, if ROI size is too small the system can lose race data and consequently will lose slot car control.

ROI sizing process involves two parameters. The first one is camera frame rate and the last one is top speed of the slot car on the circuit.

In the following tables you can see a practical study done three years ago, reflected in REF: BFP.02 final project:

Top speed on the circuit straights			
Lap	Frame	Time	Top speed(cm/s)
1	280	9,333	266,667
	289	9,633	
2	321	10,700	300,752
	329	10,966	
3	363	12,100	240,240
	373	12,433	
4	403	13,433	299,625
	411	13,700	
5	443	14,766	266,667
	452	15,066	
6	484	16,133	299,625
	492	16,400	
7	523	17,433	299,625
	531	17,700	
8	563	18,766	299,625
	571	19,033	
9	600	20,000	343,348
	607	20,233	
10	637	21,233	299,625
	645	21,500	
11	680	22,666	266,667
	689	22,966	
12	721	24,033	266,667
	730	24,333	

Top speed on the circuit bends			
Lap	Frame	Time	Top speed(cm/s)
1	268	8,933	300,000
	280	9,333	
2	309	10,300	300,000
	321	10,700	
3	350	11,666	276,498
	363	12,100	
4	392	13,060	321,716
	403	13,433	
5	431	14,366	300,000
	443	14,766	
6	472	15,733	300,000
	484	16,133	
7	511	17,033	300,000
	523	17,433	
8	551	18,366	300,000
	563	18,766	
9	589	19,633	326,975
	600	20,000	
10	626	20,866	326,975
	637	21,233	
11	668	22,266	300,000
	680	22,666	
12	709	23,633	277,136
	722	24,066	

As you can see in the above tables, the Lexus 430 sc orange top speed is 343.3 cm/s on the circuit. Remember that DBK21AU04 camera Frame Rate is 30 fps. Based on these two parameters, we have to do the following process to estimate the ROI size:

1. Calculate the distance traveled by the car from one frame to the next:

$$Distance = \frac{Top\ speed}{Frame\ rate} = \frac{343.3\ cm/s}{30\ fps} = 11.44\ cm$$

2. Estimate Distance/Pixel ratio:

Circuit size → 1640 x 840 mm

Circuit size on the photo → 612 x 323 pixels

$$\left\{ \begin{array}{l} DistPix Ratio1 = \frac{1640}{612} = 2.68 \text{ mm/px} \\ DistPix Ratio2 = \frac{840}{323} = 2.60 \text{ mm/px} \end{array} \right.$$

The most restrictive result: **1 Pixel → 2.60 mm.**

3. Calculate ROI size based on the results of the previous steps:

$$ROI \text{ size} = \frac{Distance}{DistPix Ratio} = \frac{11.44 \text{ cm}}{2.60 \text{ mm/px}} = 44.01 \text{ px}$$

Considering the results, we can say that appropriate ROI size is 45x45 pixels. Actually, a 45x45 pixels ROI represents in the real world an 11.70 cm square region. The increment between real ROI selected (11.70 cm) and the distance traveled by the car from one frame to the next (11.44 cm) is a margin of safety to ensure the correct operating of the control system task.

WARNING:

If you ever decide to regulate the lens objective of the camera you have to keep in mind that the results obtained in previous sections are invalid. In this case you will have to recalculate all parameters of this section.

3.4.2.2. Circuit zoning:

The circuit zoning concept is to situate ROI regions on the circuit. In the previous section we can estimate that ROI size is 45x45 pixels.

The circuit zoning concept is to situate ROI regions on the circuit. In the previous section we can estimate that ROI size is 45x45 pixels. In order to perform this circuit zoning is very important keep in mind the location (pixels) of some singular regions of the circuit.

On the following picture you can see these singular regions mentioned above:

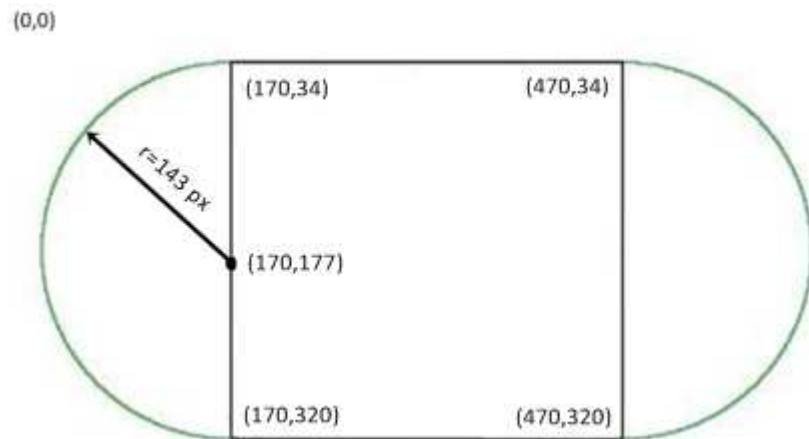


Fig. 12. Location of the singular circuit regions.

When we know the location of the singular regions, we are ready to place correctly the ROI regions on the circuit. First, we will place the first ROI in the starting line (painted in red on the following picture). Then we will put the other ROI regions (painted in green) in clockwise order. We had to put 33 ROI regions to complete the racetrack.

In the picture below you can see the final circuit zoning:



Fig. 13. Circuit zoning.

To conclude this section, we have considered necessary to draw a chart where you can see a locations list of the centers of the ROIs.

ROI index	Location	ROI index	Location	ROI index	Location
Pos. 0	(405, 320)	Pos. 11	(40, 143)	Pos. 22	(464, 34)
Pos. 1	(360, 320)	Pos. 12	(55, 109)	Pos. 23	(409, 46)
Pos. 2	(315, 320)	Pos. 13	(78, 78)	Pos. 24	(542, 69)
Pos. 3	(270, 320)	Pos. 14	(109, 50)	Pos. 25	(577, 95)
Pos. 4	(225, 320)	Pos. 15	(149, 34)	Pos. 26	(597, 135)
Pos. 5	(180, 320)	Pos. 16	(194, 34)	Pos. 27	(603, 180)
Pos. 6	(135, 313)	Pos. 17	(239, 34)	Pos. 28	(597, 225)
Pos. 7	(96, 296)	Pos. 18	(284, 34)	Pos. 29	(577, 270)
Pos. 8	(64, 270)	Pos. 19	(329, 34)	Pos. 30	(540, 304)
Pos. 9	(40, 233)	Pos. 20	(374, 34)	Pos. 31	(495, 313)
Pos. 10	(30, 188)	Pos. 21	(419, 34)	Pos. 32	(450, 320)

3.4.3. Get reference picture

In order to know slot car position, the system has to compare two pictures to find difference. Thus, the first picture will be taken as a reference. This picture is saved because it contains the initial conditions of the slot circuit. Later, when the camera takes new pictures, the system will compare this picture with reference picture. The changes between reference picture and new picture may denote appear or disappear of the slot car.

To reduce the information that has to be stored is used colour filter. This is done because it is only necessary to know the elements of the picture that are the same colour as the car, i.e. orange.

3.4.4. Colour filter

In order to detect the car easily we have to use a strong color car. For this project it was decided to use Lexus 430 sc Orange. In the original version the car had a black top, but to facilitate the detection, the car top was painted orange.



Fig. 14. Lexus 430 sc orange (Original version)



Fig. 15. Lexus 430 sc orange (Current version)

Colour filter process involves analyzing every pixel of the picture and check where it corresponds to a given value (the value represented by the orange colour). Given below is a binarization converting pixel white or black according to whether the previous check was positive or negative.



Fig. 16. Binarization concept

3.4.5. Tracking car system

At all times the system has to know the current position of the vehicle and the next position. Also you have to keep in mind the circuit zoning and ROI concept implemented in this process. Having made a correct ROI sizing, you can be assured that the camera will always have enough time to see the car at each ROI.

In the annexed diagram below you can see the operating blocks diagram of the tracking car system:

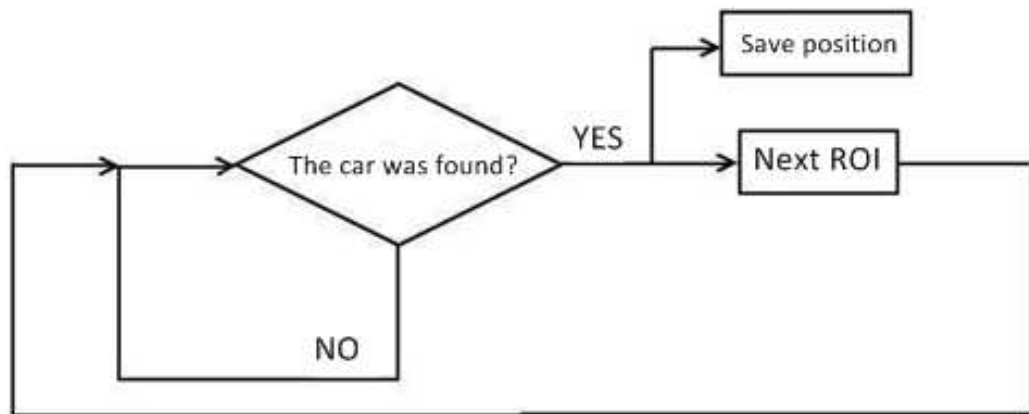


Fig. 17. Flowchart of the tracking car system

3.5. Adjustments workspace and lighting

One of the most important aspects to ensure the correct operation of the system is the correct placement of the circuit. The camera has to focus at all times in the complete circuit to keep any information about the slot car.

Therefore, before you start the race, you have to adjust the location of the circuit. IC Capture software provides an interface through which you can adjust the position of the car.

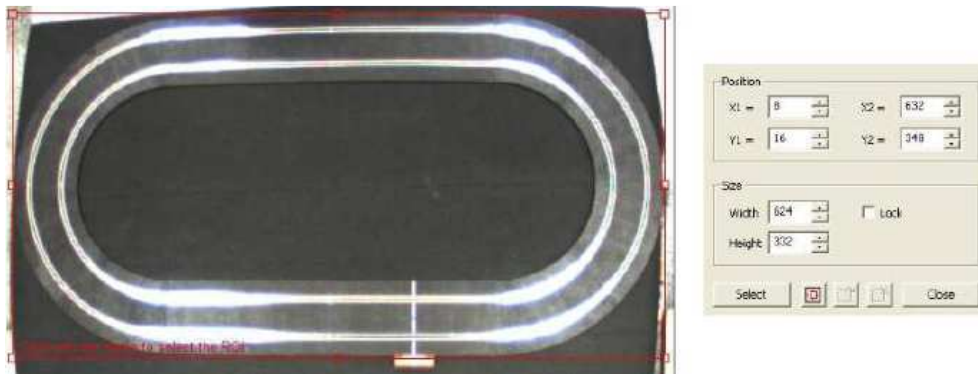


Fig. 18. Interface to calibrate the slot car position

In the picture above you can see a red box around the area that has to be placed the circuit. The circuit has to hold all the area inside the red box but in no case may overhang of the square.

To adjust the red zone size you have to enter these parameters: X1= 8, X2= 632, Y1= 16, Y2= 348, WIDTH=624 and HEIGHT=332.



Keeping with very important aspects to ensure the correct operation of the system we have an obligation to mention the lighting of the room. Adjust the lighting will be the next step after placing the circuit.

Considering that the control system is based on computer vision, adjust the lighting is essential to see the car with the correct colour shade. To run this process can be used three elements: camera diaphragm, fluorescent lights and blinds.

Section "*Lighting conditions calibration*" located on page 113 of this project is a step by step tutorial that explains how to adjust lighting.

4. PC-RACETRACK INTERFACE

Last year an UPC student, Edgar Jiménez López, made his final project Ref: BFP.03 entitled “*DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE UN CIRCUITO SLOT*”. Due to this project today we have the hardware to connect PC with the racetrack. It is a critical piece of the control system because it is the physically governing the voltage that is provided to the circuit.

Note that the following sections are not a thorough analysis of the PC-Racetrack interface, only are explained the most important issues for the project development at hand. If you want to see a more comprehensive report about this, you can read the above document (Ref: BFP.03).



Fig. 19. PC-racetrack interface. Outside

Below is presented in a schematic way the hardware operation:

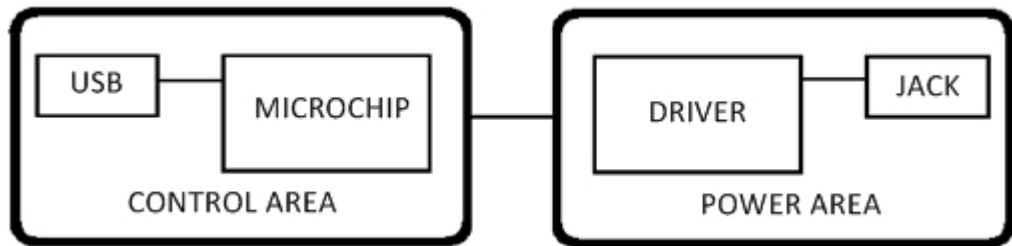


Fig. 20. PC-Racetrack interface. Blocks diagram system

4.1. PC-Racetrack interface. What's inside and how it works?

If we were a data signal and we travelled from the PC to the racetrack the first element that we would find along the way would be the USB wire. The USB wire has a dual role: transmit data from the PC and provides power to the microprocessor.

Microprocessor is the brain of the system and its role is to read the information that comes from the computer, process it and act accordingly. We have used PIC18f2550 Microchip. To execute correctly the work of this microprocessor it has to be ruled with a clock frequency. This clock frequency is derived from the intrinsic frequency of a 20 MHz quartz crystal.

Before further explaining the interface operation we should show the electrical schematic:

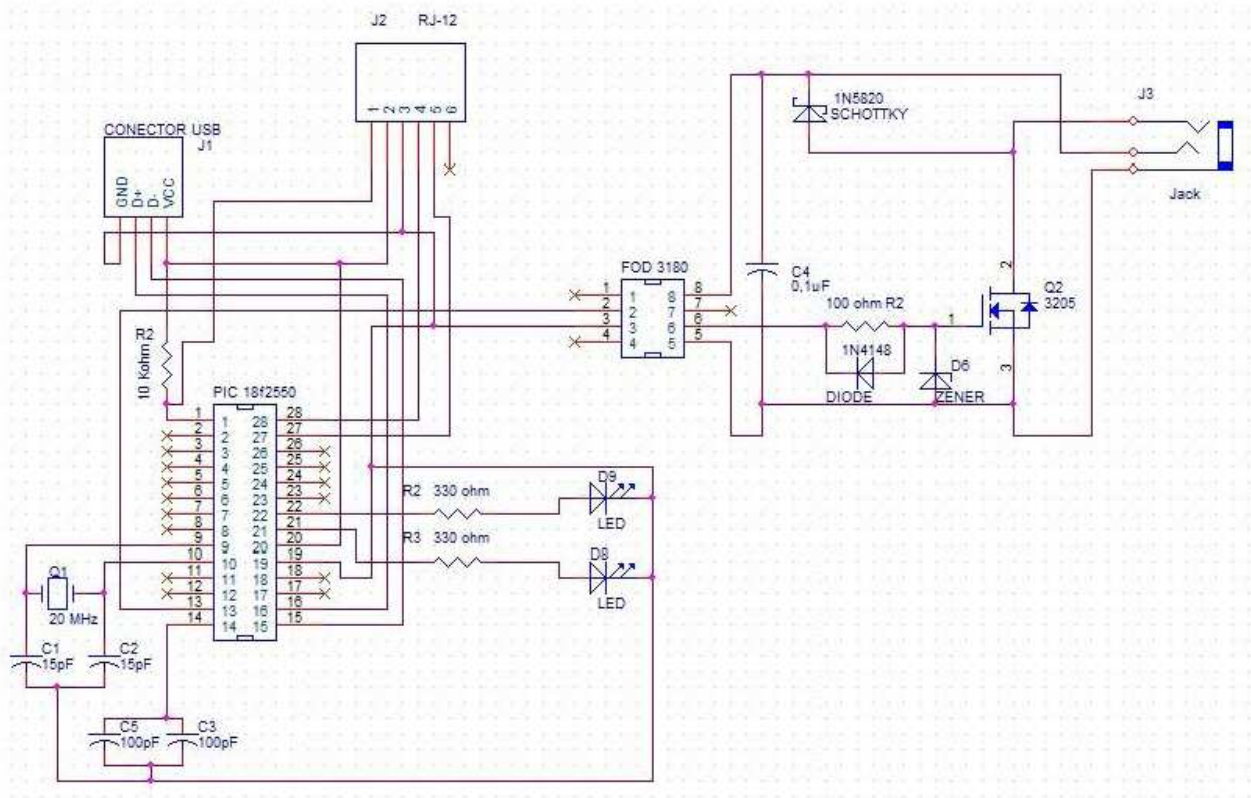


Fig. 21. PC-Racetrack interface. Circuit diagram

Following the study about the most important components we cannot forget about the driver. The circuit is powered from the power grid and, as we saw in previous sections, slot car motor has 14.8V. To control these 14.8V is used transistor and a PWM signal. However, the problem is the microprocessor provides 0-5V and the circuit requires 14.8 V for speed limit. To solve this problem the developer has decided to include a driver.

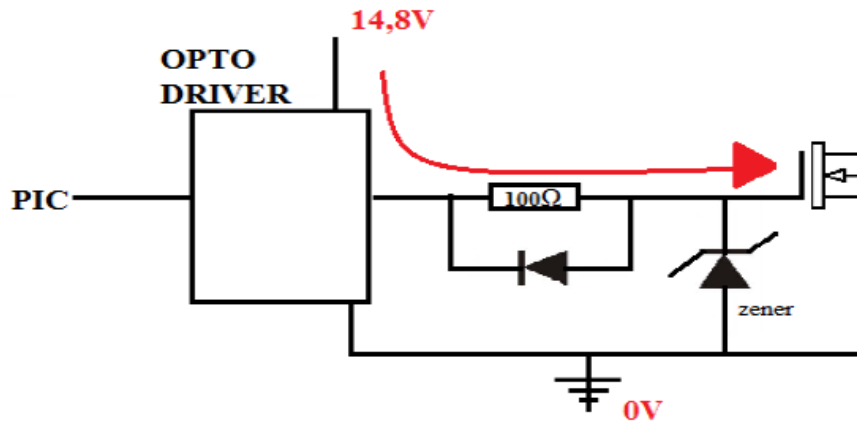


Fig. 22. Power supply when occurs positive transition of the PWM signal.

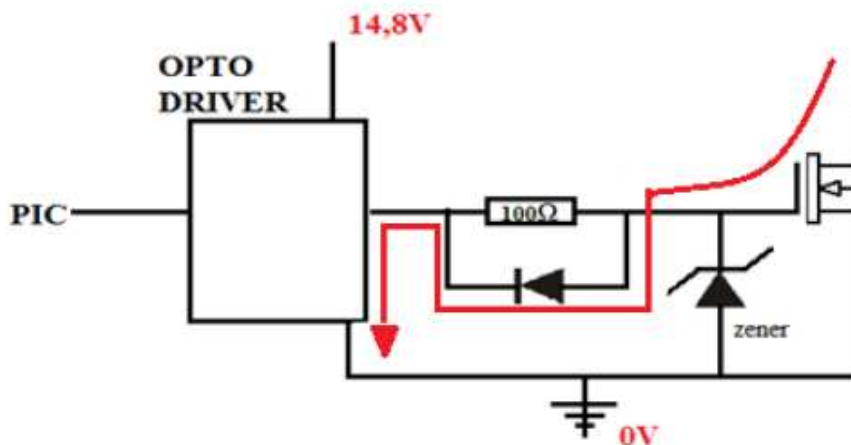


Fig. 23. Power supply when occurs negative transition of the PWM signal.

In the schemes shown above, you can see the PWM signal path according if it is a positive or negative transition. In the first case, the slot car is connected to 14.8 V during the time width of the PWM signal. When the first negative transition happens the motor is disconnected and driver is connecting directly to ground. It is also important to note that a Zener diode is used to ensure the correct operation of MOSFET.

In the next picture you can see PWM waveform when load is constant:

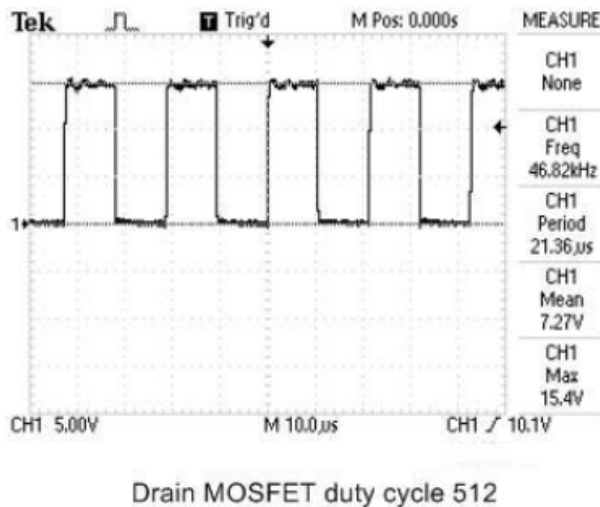


Fig. 24. Testing. PWM waveform when load is constant

WARNING:

It is very important to note an aspect of interface operation that can be problematic: Jack connector shorts optocoupler terminals and MOSFET. For this reason, when connected the interface via jack connector we have to make sure the wire has no voltage.

The steps to prevent shorting are:

- 1) Plug in jack connector to the interface.
- 2) Plug in jack connector to the racetrack.
- 3) Connect power to the circuit.

4.2. Microchip software

It is recalled that the microprocessor has the role to generate PWM signal based on the read data via USB. This involves the PIC must have a software that is able to run this task. In the annexed diagram below you can see blocks diagram system.

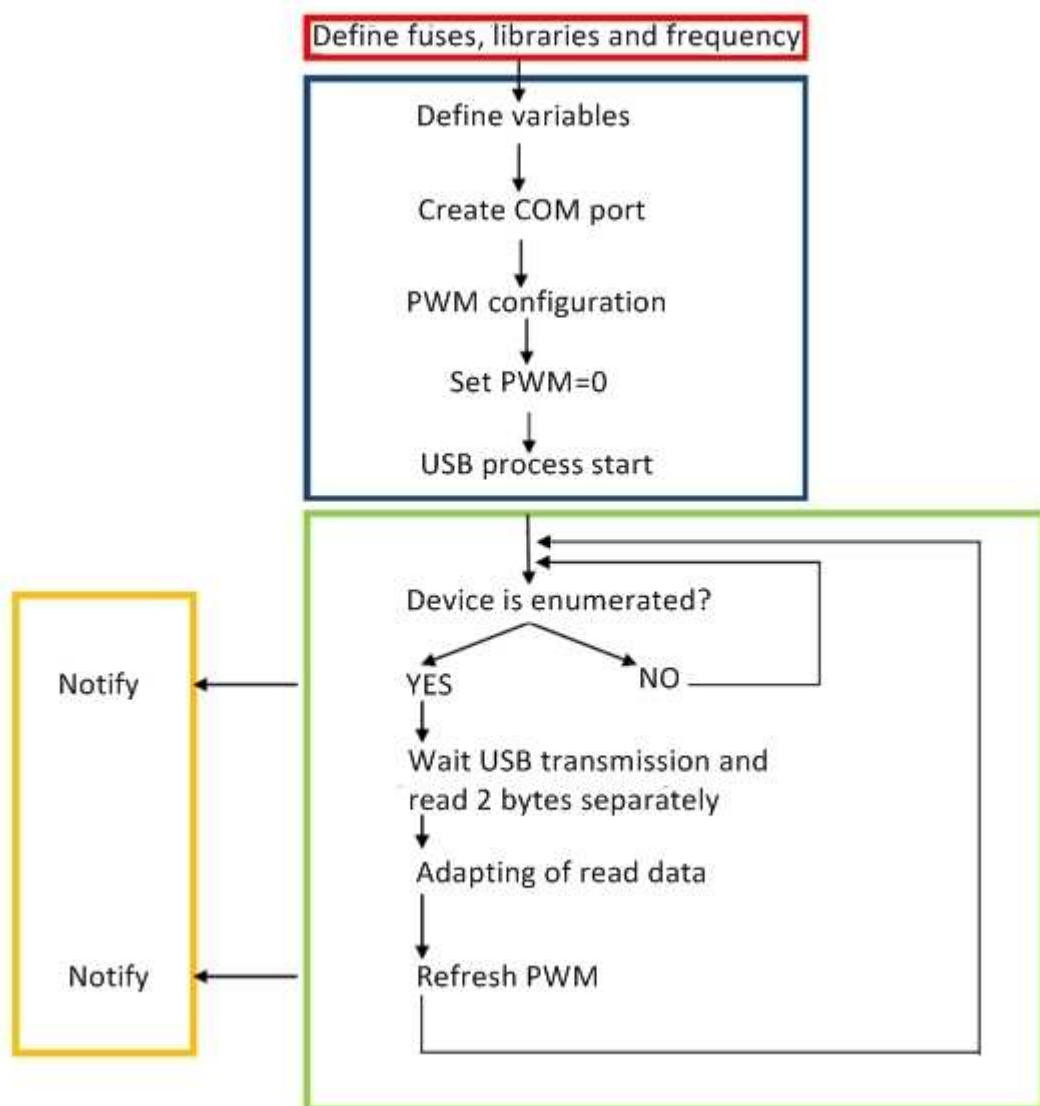


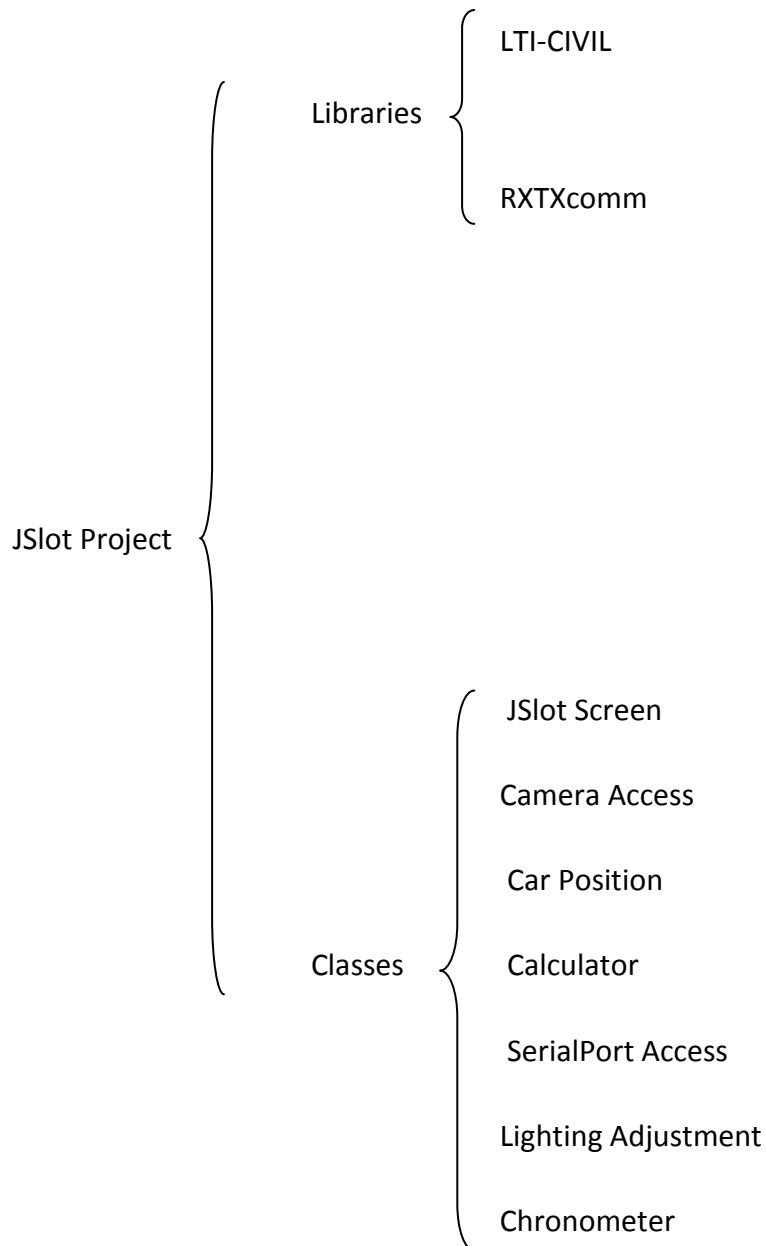
Fig. 25. PIC software flowchart

As you can see in the above flowchart the process comprises three tasks. The first, marked in red, is to define fuses, libraries and frequency. The second one, marked in blue, is to define local variables and to realize USB and PWM configuration. Finally, the last one, marked in green, is to refresh PWM signal according to read data.

The notifications are made to the red and green LEDs. If the red LED is ON, it means that the interface is connected to the PC. In the same way, if the green LED is ON, it means that the interface has read new data.

5. JSLOT PROJECT. CLASSES AND LIBRARIES

JSlot Project is the platform created to allow users to interact with the control and monitoring system of the slot car. Below you have a chart with all Java classes and libraries that make up the project:



5.1. LTI CIVIL.

LTI-CIVIL is a Java library for capturing images from a video source such a USB camera. LTI-CIVIL stands for **Larson Technologies Inc. Capturing Images and Video In a Library**. It provides a simple API and does not depend on or use JMF. Beside this, it is totally free and is available for all OS.

If you want to know more about LTI-CIVIL project, you have to go to <http://lti-civil.org>. Also you can access to <http://lti-civil.org/doc/index.html> in order to know more about each class integrated in LTI-CIVIL project.

In contrast to JMF, LTI-CIVIL not use streaming, instead it use events for taking pictures task.

5.2. RXTXComm

RXTX is a Java library for Serial Port control. Note that when you installed the drivers to control the device connected via USB, a serial port was assigned to the device by PC. Therefore, JSlot Project has to control a serial port and conversely does not have to control USB connection, as you can think at first.

If you want to know more about RXTX libraries, you have to go to <http://rxtx.org/>. If you don't find what you need on this web, you can go to http://rxtx.qbang.org/wiki/index.php/Main_Page

5.3. JSlot Screen

JSlot Screen is a Java Class developed in order to provide a graphical interface to control slot car and display monitoring data. Besides this, JSlot Screen is responsible for managing all events and to interconnect all used Java objects. Following image shows the visual appearance of JSlot Project application. If you want know more about any fields or buttons of this visual window you should go to user guide chapter on page 120.



Fig. 26. JSlot GUI

JSlot Screen includes the next methods: **JSlot_Screen()** (the builder), **initComponents()** (is a generated code automatically by NetBeans and includes all main java commands to design the visual window), **screenAdjusting()** (includes any java commands that are not includes on initComponents method),

clearFields(), **toString()** (return the most important race data using String form), and all **actionPerformed** methods.

5.4. Camera Access

Camera Access is a Java Class developed in order to provide a way between taking pictures task and main class of JSlot Project. Camera Access class implements CaptureObserver interface and therefore this class is required to contain all methods implemented in CaptureObserver. Note that CaptureObserver is an interface developed for LTI-CIVIL project.

OnNewImage method has to provide acquired pictures to CameraAccess class. When a picture is acquired, it is converted into a BufferedImage and is temporarily stored with attribute form. It is very important to note that OnNewImage method is based on events, not on streaming. Is possible get the last image stored using **myImage()** getter method.

Another methods included on Camera Access java class are: **CameraAccess()** (class builder), **onError(CaptureStream stream, CaptureException ce)** (called when an error has occurred during capture) and **disconnect()** (used to stop the stream).

5.5. Car Position

Car Position is a java class developed in order to perform image processing tasks to pictures taken of the race and get, through binarization tasks, location of slot car on the circuit.

This class includes the following methods:

1. public **CarPosition**(int ROIsize,int[] pixel_X,int[] pixel_Y)

This method is used when a new CarPosition object is created (class builder). When this class builder is executed is imperative insert three inputs. The first needed input is ROI size measured in pixels. Second and third inputs are coordinates in pixels of ROIs location.

For example, in case of ROIs are placed in (200,32) , (400,50) , (300,118) and (55,200) the correct inputs are: pixel_X={200,400,300,55} and pixel_Y={32,50,118,200}.

2. public void **calibration**(BufferedImage picture)

```
public void calibration(BufferedImage picture) {
    for(int i=0;i<640;i++){
        for(int j=0;j<480;j++){
            if((picture.getRGB(i,j) & 0xFF0000) > 0xC80000) && ((picture.getRGB(i, j) & 0x0000FF) < 0x80) {
                Reference_Picture.setRGB(i,j,Color.WHITE.getRGB());
            }
            else{
                Reference_Picture.setRGB(i,j,Color.BLACK.getRGB());
            }
        }
    }
}
```

As you can see in above image, where is presented the font code used in calibration method, orange areas are searched pixel by pixel. When an orange pixel is found, the same pixel is painted with white colour in a reference picture. However, if pixels are not orange then reference picture is painted with black colour for these pixels. In short, it is a binarization task

used to store where are located the orange areas. It is necessary to avoid confusion slot car (orange) with these orange areas.

3. public int[] **check_OutBoundsPIXELS()**

This method is called from newFrame method. Check_OutBoundsPIXELS provides to newFrame the bounds in order to avoid perform the binarization task for coordinates that exceed the picture size (640x480).

4. public void **newFrame**(BufferedImage Original_Picture)

newFrame method is used when a new capture appear from the camera. This code font looks for orange areas, pixel by pixel for a single ROI, in order to find the slot car. When an orange pixel is found, the same pixel of reference picture is analyzed to determine whether was previously orange (part of the circuit or workspace) or is a new orange pixel (part of the slot car). Finally, if the number of orange pixels related to the slot car exceed a fixed value (in this case forty) will be determined that slot car is placed on the analyzed ROI. When slot car is placed on the current ROI then “*Location*” attribute is increased to ensure when newFrame method will be called again, the search will be on the next ROI.

5.6. Calculator

Calculator is a java class developed in order to calculate all monitoring data related to the race. This class includes two builders. You have to use the first class builder when ROI distribution is regular. In contrast, you have to use the second class builder when ROI distribution is irregular.

Some of the most important methods of calculator class are:

1. public void **setRaceData**(int position, long positionTimer)

Whenever will be necessary recalculate the current race data, setRaceData must be called from main method of JSlot Screen. This method is responsible for decide which data needs to be recalculated depending on the car position. For example: if it is the first lap and the car is located on the first bend of the circuit, setRaceData will recalculate the current speed. However, if it the slot car is placed in last ROI, setRaceData will recalculate current speed, lap time and average speed.

2. public void **setSpeed**(int position, long positionTimer)

When this method is used are available two options depending whether or not ROI distribution is regular. In case that ROI distribution is regular, the current speed will be calculated in the following manner:

```
speed=(double) ((ROI*pixelSize)/(positionTimer-Timer1));
```

Instead, when ROI distribution is irregular, the current speed will be calculated using the following math expression:

```
speed=(double) ((LengthSections[position]*pixelSize)/(positionTimer-Timer1));
```

Where:

ROI: ROI size measured in pixels

pixelSize: Actual size in mm of each pixel.

Timer1: time point for the last previous speed calculation.

If this speed exceeds topSpeed value, this speed value will be stored in the topSpeed attribute.

3. public void **setLapTime**(long positionTimer)

This method is called from `setRaceData` every lap when the slot car is placed on the last ROI. If current `lapTime` exceeds `fastLap` value, this `lapTime` will be stored in the `fastLap` attribute.

4. public void **setAverageSpeed()**

In the same way that `setLapTime`, this method is called from `setRaceData` every lap when the slot car is placed on the last ROI. The average speed will be calculated in the following manner:

```
averageSpeed=(double) ((averageSpeed*(laps-1)+(lengthTrack/lapTime))/laps);
```

*`lengthTrack` represents the travel distance in pixels by slot car for each lap.

5.7. SerialPort Access

SerialPort Access is a java class developed to manage the use of the serial port and thus allow, send control data to the microprocessor connected via USB .In this manner, the microprocessor will generate the proper PWM signal to control at any given time the supply voltage of slot car motor. It is important to note that this class relies on RXTX library.

This class has three methods: **Connect** (String COM), **write** (int data) and **close** (). The first one looks for the serial port and connects to it. On the other hand, the second one is used to send control data to the device responsible for reading this data and manage the supply voltage of slot car motor. As a result, you can achieve proportional voltages respect to control data sent previously. Finally, the last one is used to disconnect the current serial port.

5.8. Lighting adjustment

Lighting adjustment class was developed to carry out the lighting calibration task. The operation mode is simple: using camera Access platform this class get a `BufferedImage` through `newFrame(BufferedImage picture)` method. Then, is performed a processing task in order to binarize the orange regions. Finally the new binarized picture is stored on *"Lighting adjustment"* folder. Lighting adjustment class will be used whenever you click in *"Edit"* tab and then in *"Lighting"* of visual interface JSlot Screen.

5.9. Chronometer

Chronometer is a simple java class to measure the race time. The most important methods of this class are: **start()** (used to start the timer), **stop()** (stop the timer), **reset()** (this method have to be used to reset the timer), **elapsedTime()** (return the elapsed time in milliseconds since the timer was started) and **elapsedTime_string()** (return the elapsed time in hh:mm:ss mode).

6. HOW JSLOT WORKS?

When all classes and libraries used by JSlot are listed, in the above section, is time to know how works JSslot application, i.e. which tasks are running, how is the sequence to do it, the events that can appear, and so on. For this reason, it was decided to make a flow chart where is collected this information in a visual manner.

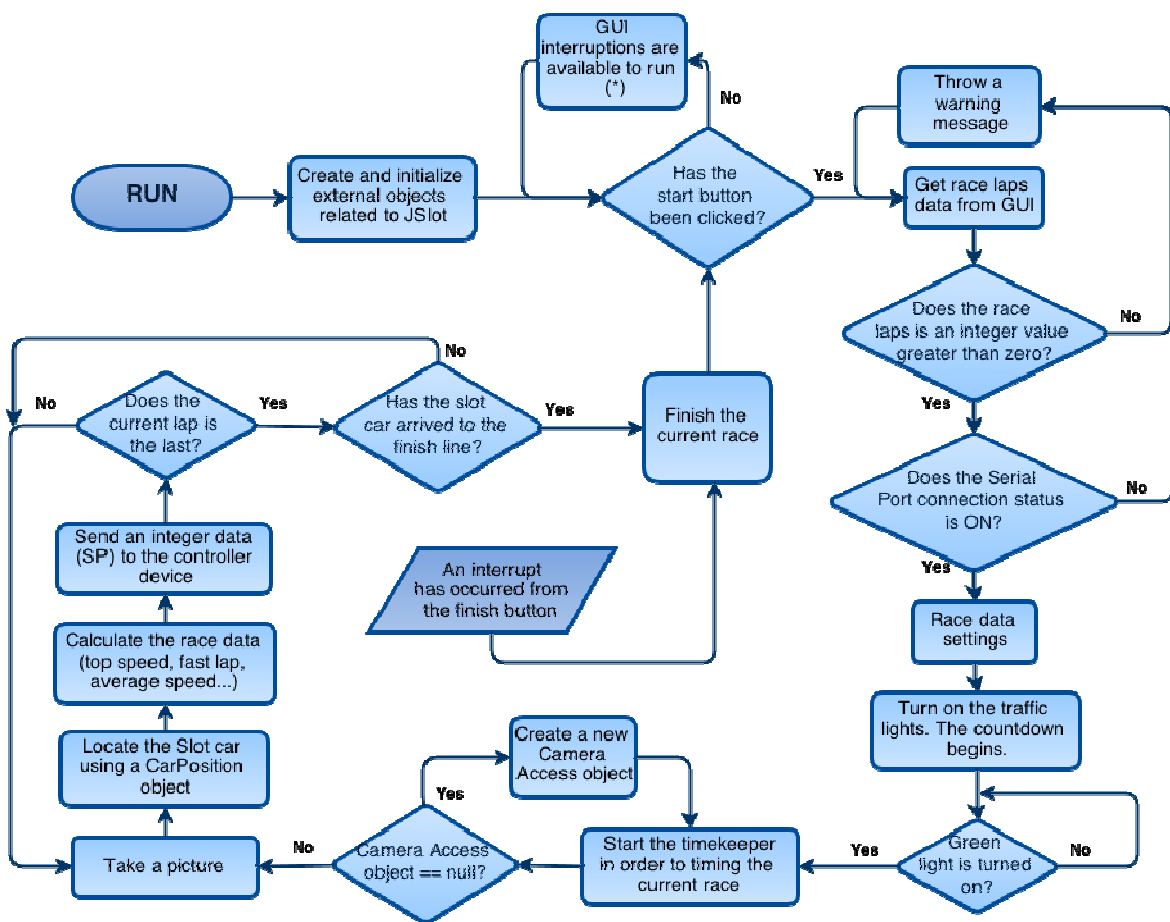


Fig. 27. JSslot flowchart

(*) These interruptions can appear from several events. These events can be thrown from the Menu items or from buttons. As menu item there are several options: lighting adjustment, save/open/new DB file, open the user guide and workspace

adjustment. On the other hand, as button group, the options are "connect" or "disconnect" the serial port.

It is very important keep in mind that these interruptions (*) cannot run when the race was started. In contrast, the "finish" button event can run when the race was started and their work is stop the current race and make ready JSIot application for the next slot race.

During this section will be discussed the tasks performed by the interruptions (*) when it appear.

1. **Connect button event:** When connect button is clicked, JSIot application get the chosen COM port from ComboBox components whether the race was not started. Then, JSIot will connect to the selected serial port using a SerialPort Access object. If an error appear, during the connection task, a warning message will be displayed in a new notify window.

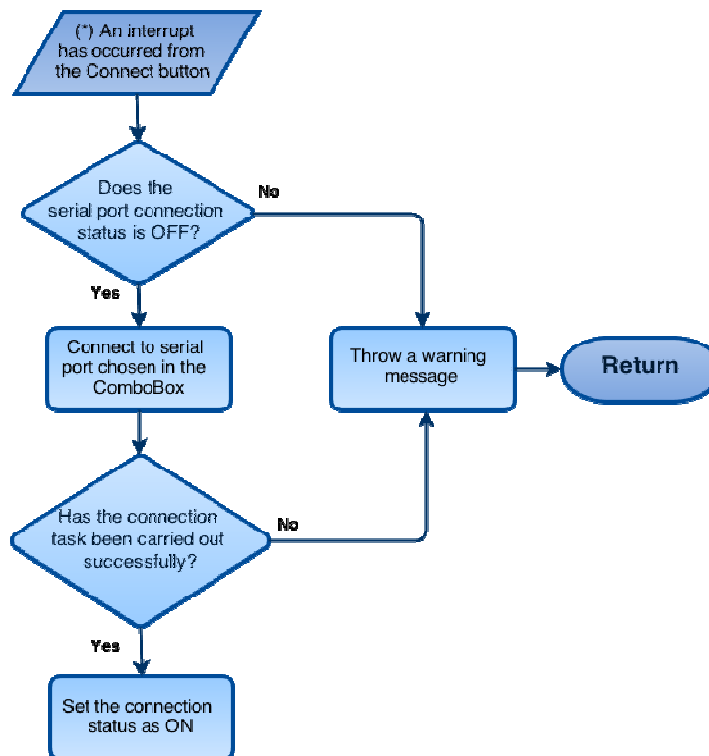


Fig. 28. Connect button event flowchart

2. **Disconnect button event:** When disconnect button is clicked, JSlot application close the current serial port connection using close() method of the SerialPort Access object. If an error happens, during the disconnection task, or when the connection status be "OFF", a warning message will be displayed in a new notify window.

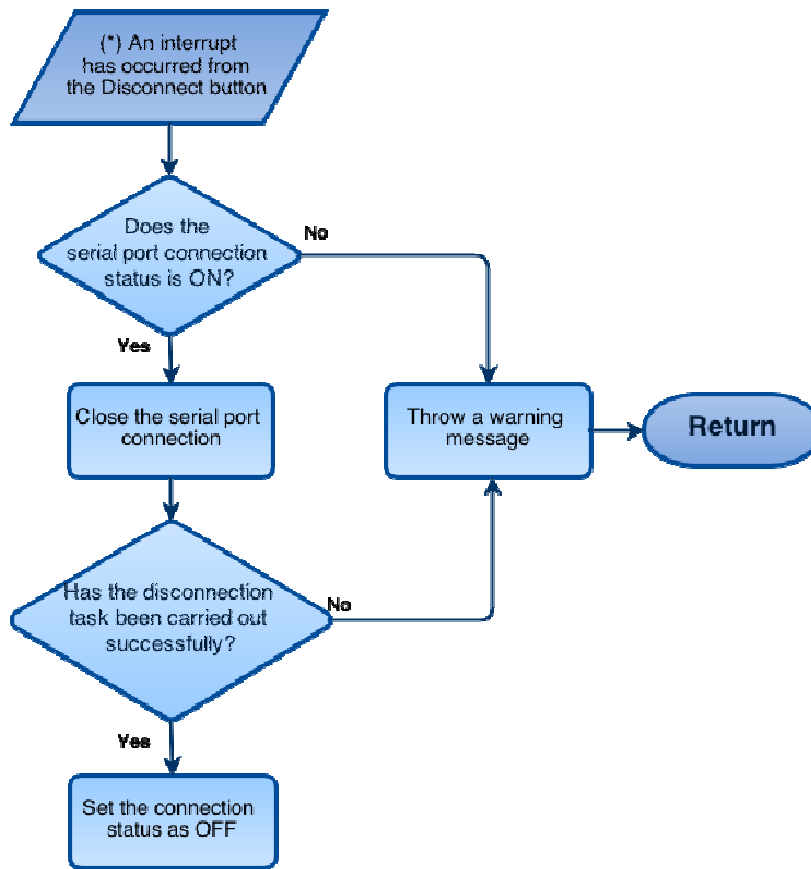


Fig. 29. Disconnect button flowchart

3. **Database manager interruptions:** These interruptions will occur when the user wants to create, open or save a database file from the application. It includes the menu items placed on the File tab. All database files will be managed from DB folder.

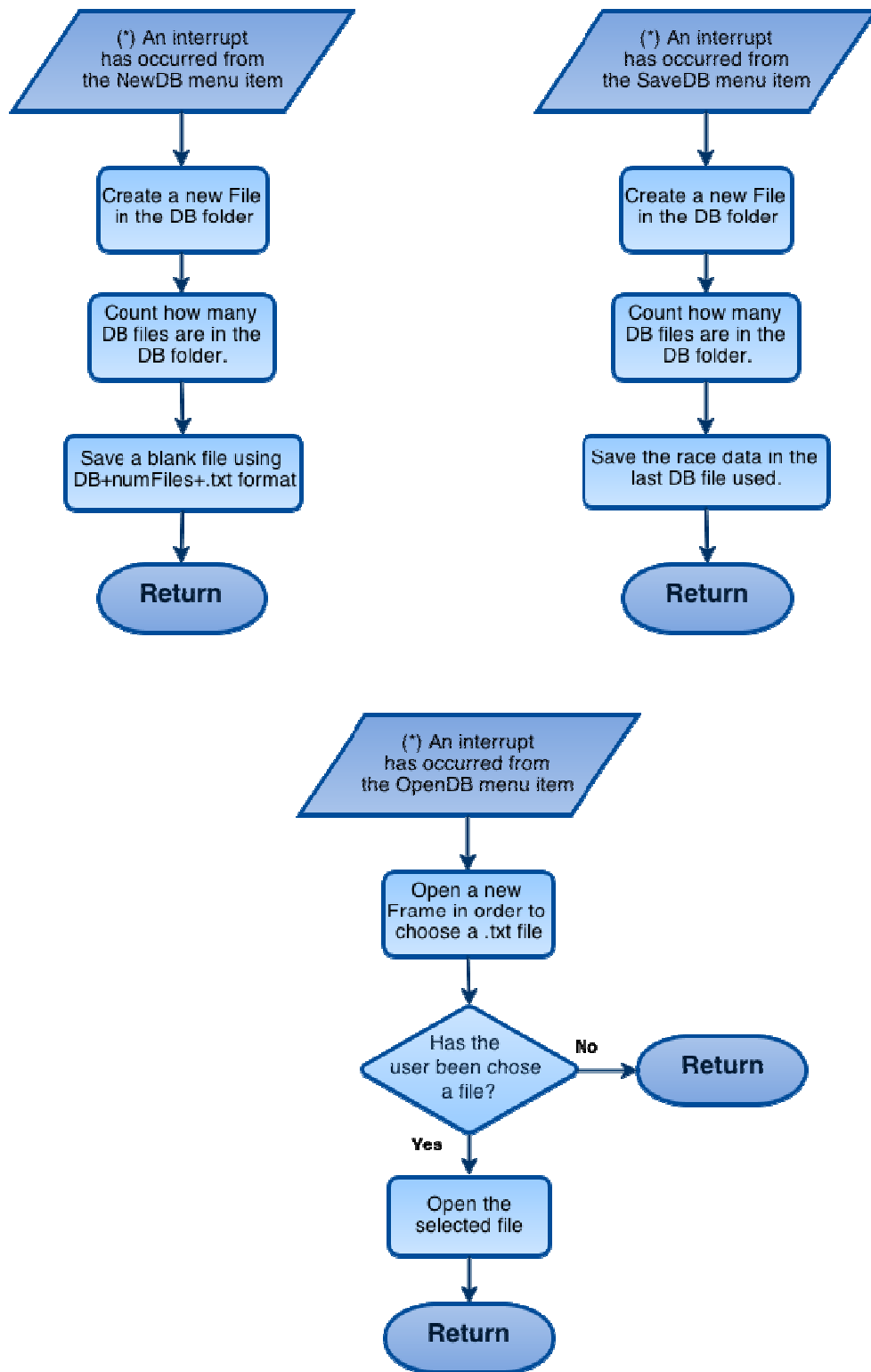


Fig. 30. Database manager events flowchart

4. **Workspace adjustment event:** This event appear when the user clicks on Edit tab and then on Circuit menu item (Edit > Circuit). When this task is run an IC Capture shortcut will be executed in order to place correctly the slot circuit. Once the workspace adjustment process is finished the user has to close IC Capture. If IC Capture software is not available at the moment, the user will be notified using a warning message.

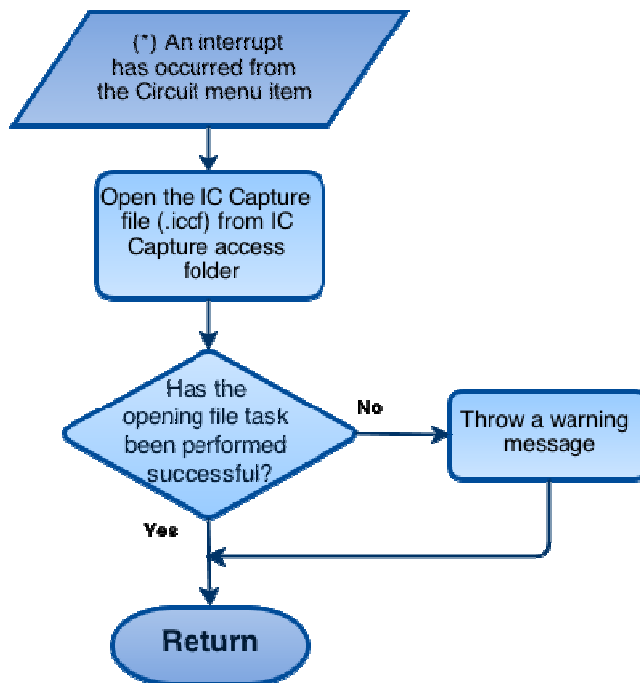


Fig. 31. Circuit menu item event flowchart

5. **Lighting adjustment event:** First, the user has to place some orange post-it in the circuit. Then, the application has to take 5 pictures using a Camera Access object. Later, a binarization task is performed using a Lighting Adjustment object in order to emphasize the orange areas. If lighting level is correct, the orange areas are white now. In contrast, if lighting level is too high or not enough, you can see some black stains in regions where the post-it are placed.

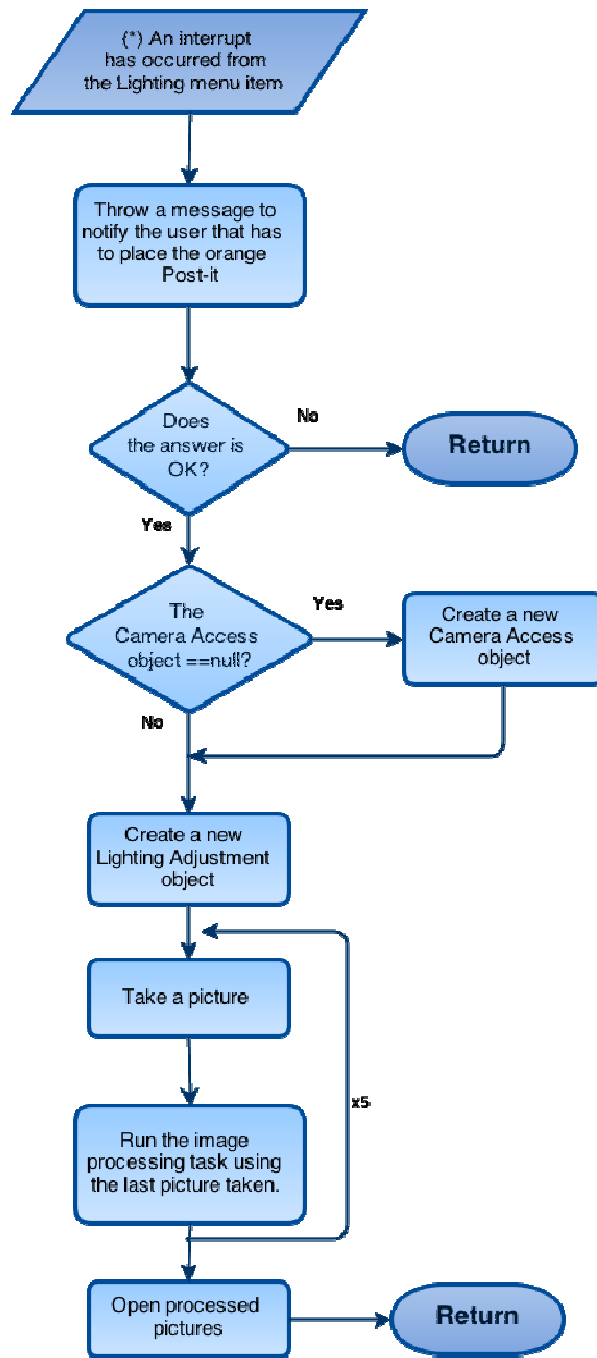


Fig. 32. Lighting menu item event flowchart

To validate the lighting level as correct, the user has to compare the images taken for this task with the reference pictures presented below:



6. **Help menu item event:** This event happens when the user choose the Help tab. Two options are possible: the user can chose open the user guide in english or in spanish. For that, the user has to click on "User Guide" for the first option or "Manual de instrucciones" for the second.

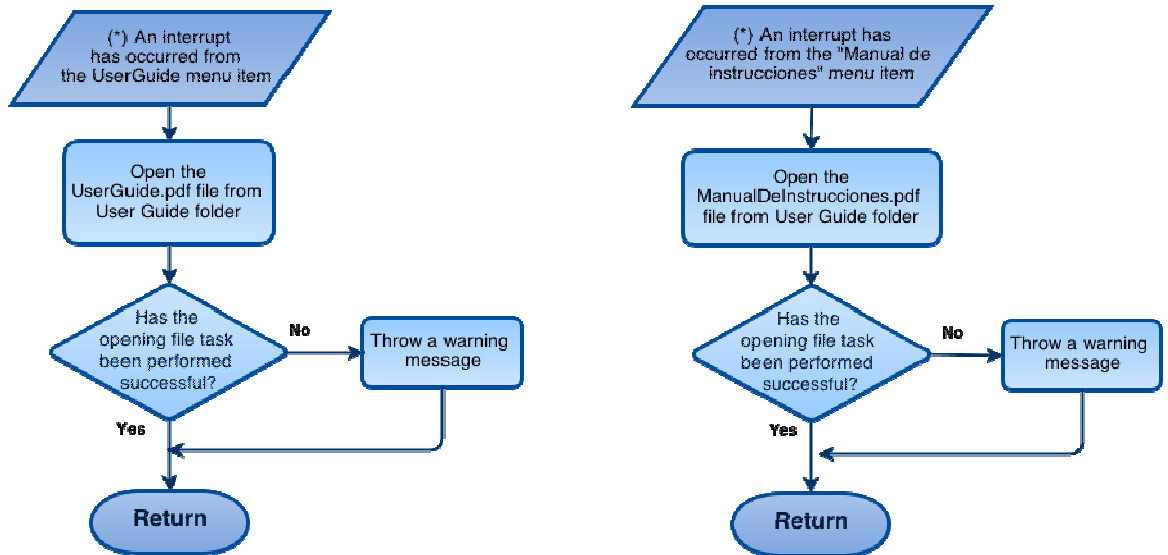


Fig. 33. Help menu item event flowchart

7. TESTS AND ADJUSTMENTS

Tests and final adjustments section contains a set of tests that have been performed to JSlot Project, both hardware and software, and solutions that have been taken to the problems that have emerged when it was found that theoretical calculations of previous sections did not correspond to reality.

7.1. Hardware

As regards the hardware part of the system, during project development process have been two important problems.

The first one is a component failure of the device responsible for generating PWM signal according to voltage level request from PC through USB communication. This device was manufactured for REF: BFP.03 project. In order to simplify device reparation task in the future, it was decided to include on this document an instruction list about how to find the problem and how to solve it.

The second one has to do with camera. The manufacturer claims that frame rate camera is 30 fps. To carry out calculation task to determine ROI size, this frame rate was used for it. Later it was found that, even though camera takes 30 frames per second, it doesn't regularly, i.e. time between a picture and the next is not constant.

7.1.1. PC-racetrack interface device

PC-racetrack interface device is a more important component to ensure the system correct operation. For this reason, in case you notice a malfunction of the device you have to find the problem immediately and solve it as quickly as possible.

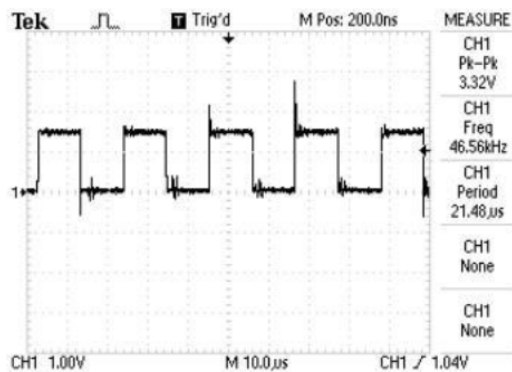
In order to find the problem that is causing a malfunction of the device you has to follow a list of steps. In this section only are presented the main steps and the results you should get. If you want to see a detailed instruction list you must go to page 125.

Before you start testing, remember that to connect the device to the PC and to the circuit, you have to make the connections in the following order:

- a. Connect USB cable to PC and then to the device.
- b. Connect the first plug of Jack cable to device.
- c. Connect the other plug of Jack cable to circuit.
- d. Connect the circuit to grid power.

First, you have to verify that the device is recognized by PC. Then you have to send an integer value, from 0 to 1023, to PIC. Next you have to check that PWM signal generated by PIC is right.

PWM signal generated by PIC has to look like following picture:

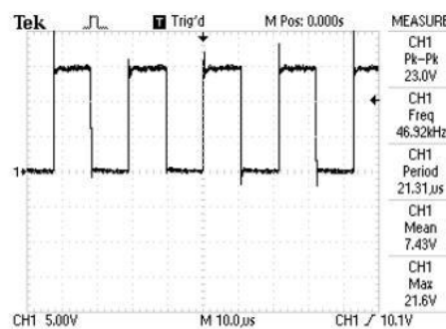


Salida PWM PIC duty cycle 512

Fig. 34. Control device. PWM waveform signal.

NOTE: Is very important to note that PWM signal amplified by optocoupler driver, in case reflected in the picture above, has a voltage mean value 7.43 V. Given that voltage value of slot car varies between 0-14.8 V and that the send data was 512 (50 %), it is expected that the measured voltage take values around 7.4 V.

Subsequently you have to check that amplified PWM signal is right. If optocoupler driver is ok, signal displayed on oscilloscope screen has to look like below picture:



Salida OPTO-DRIVER duty cycle 512

Fig. 35. Control device. Optodriver waveform signal

Finally, you just need to check that PWM signal is correctly rectified by zener diode. This electronic component has to delete the overshoot signal. The signal rectified by zener diode has to look like next picture:

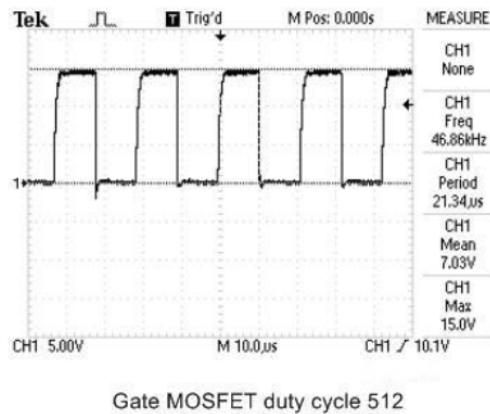


Fig. 36. Control device. Gate MOSFET waveform signal.

Before disconnecting the device, remember to follow a specific order to carry out disconnection task in order to not damage their electronic components. The order that you have to follow is presented below:

- a. Disconnect the power grid.
- b. Disconnect the first plug of Jack cable to the circuit.
- c. Disconnect the other plug of Jack cable to the device.
- d. Disconnect USB cable to PC and then disconnect it to the device.

SOLUTIONS:

- If device hasn't recognized by PC, you have to install the device driver. You can see a step by step to install this driver on page 106, titled "*Interface PC-SlotCircuit drivers (CDC Drivers)*".
- If PWM signal generated by PIC is not right, you can be sure that PIC is damaged. You have to check connections state (loose wires, broken welding, broken wire, and so on) and mend all damages that you can find. If the problem isn't solved, you have to buy a new PIC and change it.
- If PWM amplified signal by optocoupler driver is not right, you can be sure that optocoupler driver is damaged. You have to check connections state (loose wires, broken welding, broken wire, and so on) and mend all damages that you can find. If the problem isn't solved, you have to buy a new optocoupler driver and change it.
- If zener diode doesn't delete overshoot signal, you can be sure that zener diode is damaged. You have to check connections state (loose wires, broken welding, broken wire, and so on) and mend all damages that you can find. If the problem isn't solved, you have to buy a new zener diode driver and change it.

WARNING: If you have to change any electronic component of your device is very important use the same type component used when the device was assembled. Thus, you can be sure there will be full compatibility between all components of the device. The most important electronic components used for assembled task of the device are: PIC (18f2550 made by Microchip), driver

optocoupler (FOD3180 made by Fairchild Semiconductor), Zener diode (1N5333B made by Multicomp) and MOSFET (IRF3205 made in International Rectifier).

7.1.2. Camera

The delay time between a picture and the next, is a determining factor to calculate ROI size. As this delay time increases, also should increase ROI size because travelled distance by slot car for this time is longer.

Remember that camera manufacturer claims that camera frame rate is 30 fps (frames per second). The problem appears when it was found that, even though camera takes 30 frames per second, it doesn't regularly, i.e. time between a picture and the next is not constant. In next chapter you can see the results when are used different ROI sizes.

For this reason, it was decided to carry out a repeatability study of camera capture time. The results achieved were as follows:

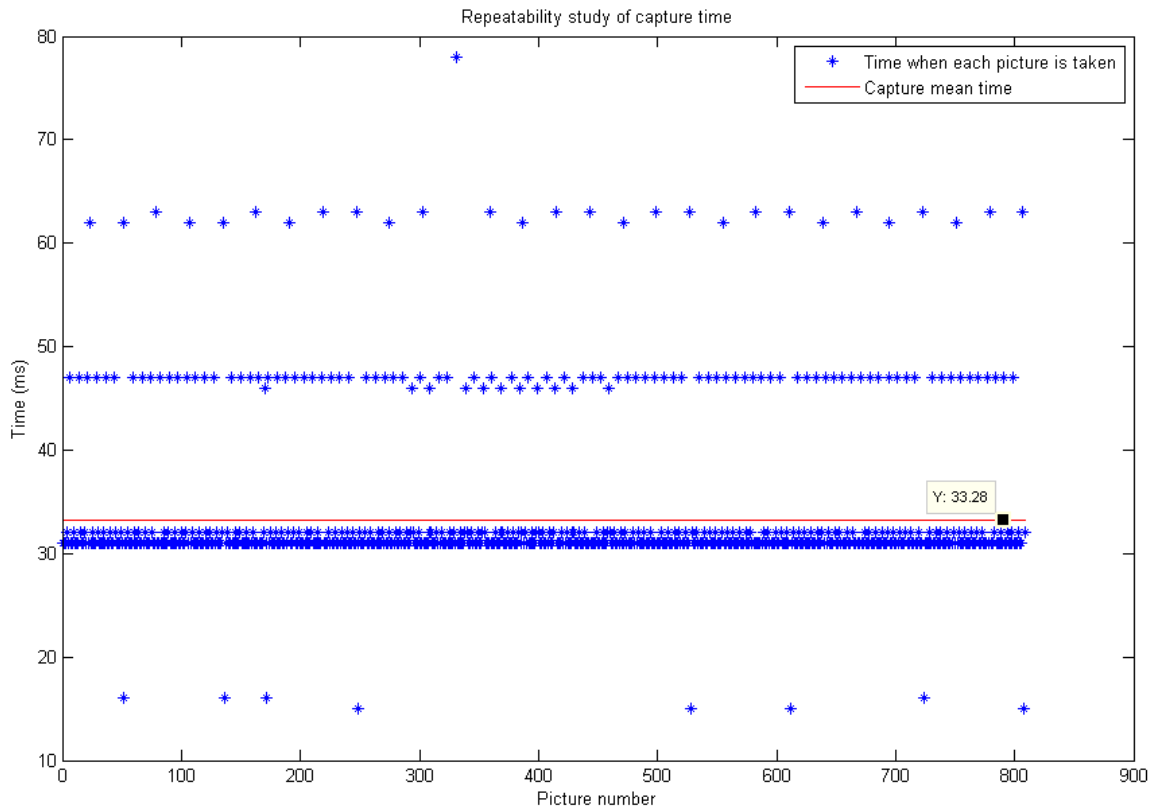


Fig. 37. Repeatability study of capture time

The repeatability study of capture time was carried out for image sequence where total pictures are 810. This experiment was performed for 27 seconds.

As you can see in the picture above, capture mean time is 33.28 ms. Therefore, this value indicates that camera frame rate is around 30 fps. Even so, some capture times are around 16 ms but others capture time are around 64 ms. This variability in capture times, means that when capture time exceeds 33 ms, is not enough time to get the slot car position through image processing task. In order to solve it, resize the ROI is essential.

7.2. Software

Regarding software part of the project, changes of the program code must be made gradually. The most critical part is the running time for execute all needed informatic tasks to carry out control and monitoring process of slot car. When main software executes the image processing task and decides the SP to be used to control slot car, is very important to do it fast enough. Otherwise it can happen that the SP generated is not adequate because current conditions have changed from the previous ones.

In short, the optimal solution is the application that combines the most important advantages of a simple and quick execution, but otherwise complete and robust enough to perform the needed tasks.

In order to develop an application that is as close as possible to the optimal solution mentioned above has decided to make a series of studies about execution times for main tasks of the project. Moreover, as was seen in previous section, camera doesn't take pictures regularly to 30 FPS and, it is for this reason that was decided resize ROI.

7.2.1. Resized ROI

ROI size is a determinate value to ensure the correct running of main application. If ROI size is too small, is very likely that slot car skip some ROI when camera take a picture and take the next. If this happens, slot car tracking will be lost and, what is more, it should be noted that slot car tracking will not recover until next lap. However, if ROI size is too large, computation time for image processing task will be too long and, consequently, the

application cannot process all images because image processing rate is less than camera rate. If this happens, slot car tracking will be lost also.

In order to prove what has been said above and to determine which ROI size is the most suitable for the system, is decided carry out tests for several ROI sizes.

All tests were carry out on the same conditions to avoid that external disturbances, compared to what really matters in these tests, affect the results. Thus, all tests have been carried out for an 8 laps at a 63.5 % of total voltage, 650 when the data is sent as an integer value. Before doing these tests it was decided to perform an initial test to determine the lap time when slot car is driving to aforementioned speed. The results were: for 40 laps test, mean lap time was 1750 ms.

First test was for a 45px ROI size. Remember that theoretical calculations done in “Sizing ROI” chapter on this document indicate that this ROI size is the optimal.

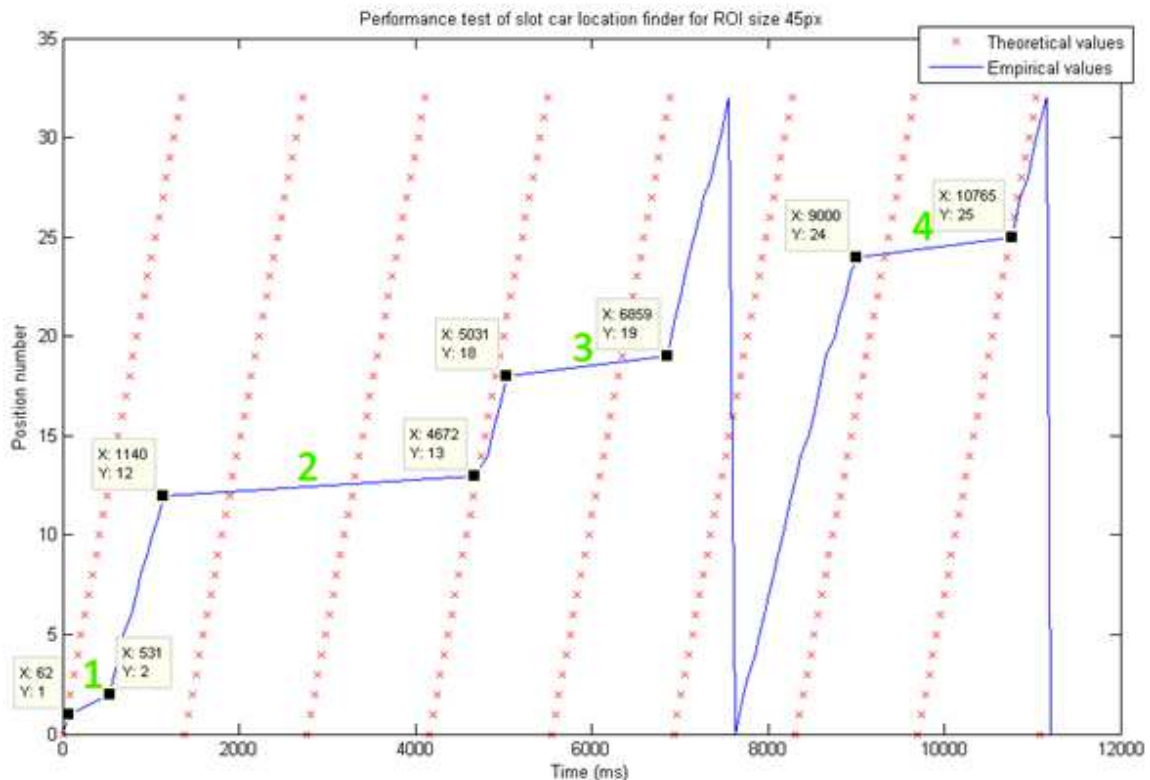


Fig. 38. Performance test of slot car location finder for ROI size 45 px

When results of performance test of slot car location finder for ROI size 45 pixels are presented visually or using schemes, you can draw more successful conclusions. This makes it much easier to make the right decisions for change the current operating mode.

Is very important you know that, in the picture presented above, car location finder using image processing task (blue points) is not the same, sometimes, that theoretical values (red points). What's more there are excessive delay times (green numbers) between the find of slot car in a specific position and the find in next position. When excessive delay times appear this is indicative that slot car tracking was lost and, on following laps, the car was found again. In order to facilitate understanding all information presented by graph above, it is decided include below a table where more important data is collected.

Loss point	ROI index where tracking car was lost	Time elapsed when tracking car was lost
1	1	469 ms
2	12	3532 ms
3	18	1828 ms
4	24	1765 ms

Given that mean lap time is 1750 ms and using data presented on table above you can draw some conclusions. Despite the fact that on table above you can find 4 tracking car loss, when delay times are analyzed in detail you can conclude that:

- Regarding first lost, since the delay time between the find of slot car in a position number 1 and the find in next position is

far less than theoretical lap time ($469 \ll 1750$) you can conclude that this delay is not caused by a position car loss. Likely to the delay is due to the moment when slot car started the race is not the same that the moment when the application was run.

- In respect to the second case you can conclude that the delay time between the find of slot car in a position number 12 and the find in next position (3532 ms) is roughly twice the theoretical lap time. For this reason, you can conclude that the position car loss occurs during 2 laps.
- Finally, regarding to 2 last cases, delay times between the find of slot car in a position number 18 and 24 respectively and the find in next position are similar to theoretical lap times. For this reason, you can conclude that both the first and second case, the loss occurred during 1 lap.

Remember that 45 pixels is the ROI size calculated for “Sizing ROI” chapter on this document using theoretical analysis. Even so, having analyzed the results when main application was run for 45 pixels ROI size, you can conclude that this value is not valid for slot car location finder task. Later, you can see some additional tests about different values of ROI size. In this way, you can determinate which one is most suitable for this project.

The second test was for a 60 pixels ROI size. Prior to seeing the test results, it was considerate appropriate draw a table where you can see a locations list of the ROI centers.

ROI index	Location	ROI index	Location	ROI index	Location
Pos. 0	(413, 320)	Pos. 8	(35, 158)	Pos. 16	(456, 33)
Pos. 1	(353, 320)	Pos. 9	(54, 98)	Pos. 17	(516, 47)
Pos. 2	(293, 320)	Pos. 10	(100, 56)	Pos. 18	(567, 76)
Pos. 3	(233, 320)	Pos. 11	(156, 33)	Pos. 19	(600, 33)
Pos. 4	(173, 320)	Pos. 12	(216, 33)	Pos. 20	(600, 193)
Pos. 5	(113, 309)	Pos. 13	(276, 33)	Pos. 21	(582, 253)
Pos. 6	(64, 278)	Pos. 14	(336, 33)	Pos. 22	(536, 310)
Pos. 7	(35, 218)	Pos. 15	(396, 33)	Pos. 23	(473, 320)

On the following picture you can see the results of this performance test:

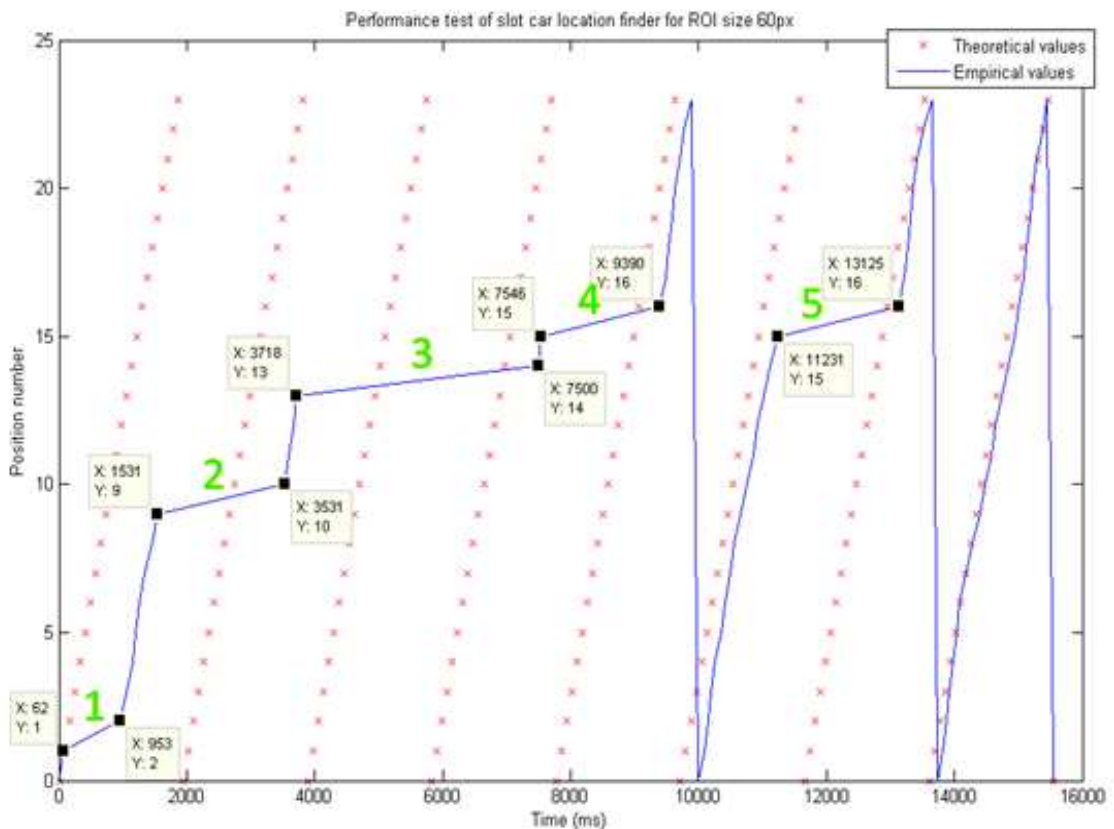


Fig. 39. Performance test of slot car location finder for ROI size 60 px

The same way that performance test for 45 pixels ROI size, car location finder using image processing task (blue points) is not the same, sometimes, that theoretical values (red points). What's more there are excessive delay times (green numbers) between the find of slot car in a specific position and the find in next position. When excessive delay times appear this is indicative that slot car tracking was lost and, on following laps, the car was found again. In order to facilitate understanding all information presented by graph above, it is decided include below a table where more important data is collected.

Loss point	ROI index where tracking car was lost	Time elapsed when tracking car was lost
1	1	891 ms
2	9	2000 ms
3	13	3782 ms
4	15	1844 ms
5	15	1894 ms

Given that mean lap time is 1750 ms and using data presented on table above you can draw some conclusions. Despite the fact that on table above you can find 5 tracking car loss, when delay times are analyzed in detail you can conclude that:

- Regarding first lost, since the delay time between the find of slot car in a position number 1 and the find in next position is far less than theoretical lap time ($891 \ll 1750$) you can conclude that this delay is not caused by a position car loss. Likely to the delay is due to the moment when slot car started the race is not the same that the moment when the application was run.

- In respect to the third case you can conclude that the delay time between the find of slot car in a position number 13 and the find in next position (3782 ms) is roughly twice the theoretical lap time. For this reason, you can conclude that the position car loss occurs during 2 laps.
- Finally, regarding to the cases number 2, number 4 and number 5, delay times between the find of slot car in a position number 9, 15 and 15 respectively and the find in next position are similar to theoretical lap times. For this reason, you can conclude that both the first and second case, the loss occurred during 1 lap.

As you can see in the previous test, in spite of the ROI size was increased from 45 pixels to 60 pixels, is still not enough. For this reason, was decided to increased ROI size to 84 pixels. Prior to seeing the test results, it was considerate appropriate draw a table where you can see a locations list of the ROI centers.

ROI index	Location	ROI index	Location	ROI index	Location
Pos. 0	(422, 320)	Pos. 6	(44, 152)	Pos. 12	(467, 44)
Pos. 1	(338, 320)	Pos. 7	(70, 84)	Pos. 13	(551, 68)
Pos. 2	(254, 320)	Pos. 8	(131, 44)	Pos. 14	(595, 135)
Pos. 3	(170, 320)	Pos. 9	(215, 44)	Pos. 15	(595, 219)
Pos. 4	(86, 295)	Pos. 10	(299, 44)	Pos. 16	(555, 285)
Pos. 5	(44, 236)	Pos. 11	(383, 44)	Pos. 17	(506, 320)

Then you can see the results about this performance test:

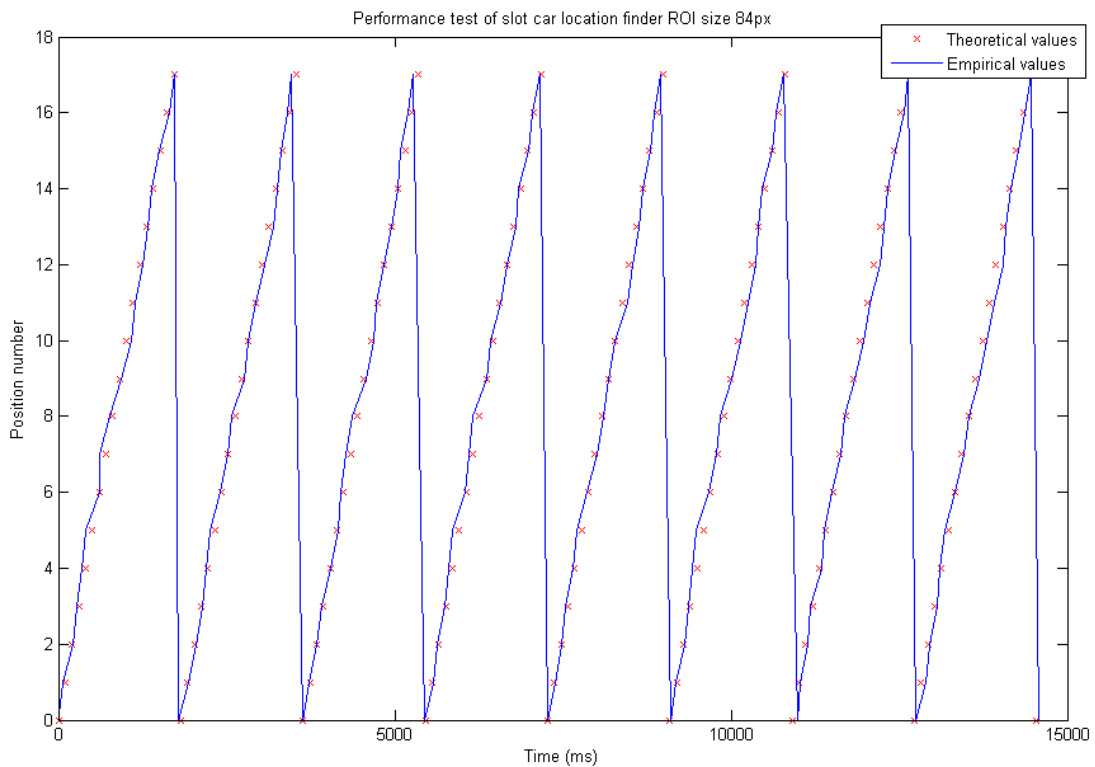


Fig. 40. Performance test of slot car location finder for ROI size 84 px

When test of slot car location finder for ROI size 84 pixels was performed, the results indicated that at no point the car position was lost. For this reason, 84 pixels is the most appropriate ROI size value to use in order that ensure the correct operating of slot car location finder task.

This study only includes some ROI size and, for this reason, is possible improve the value of the appropriate ROI size slightly. Even so, it is important keep in mind that, the ROI size value is changed in later sections of this project, when additional tasks are included.

8. AUTOMATIC CONTROL ALGORITHM

The purpose of this chapter is the research of the most suitable automatic control algorithm to drive the car automatically. Note that, during the development of this project, here it was absolutely imperative make again a lot of the work made in the final project Ref. BFP.02. As shown in previous sections, the author of the project Ref. BFP.02 did not take into account the fact that the camera takes pictures irregularly. Consequently the calculated ROI size is wrong and their application had many operational problems. For this reason, it was imperative invest much time in carry out studies that were not included on the purpose of this project. This is compounded by the fact that some classes developed on the abovementioned project were not valid for JSLOT application. For all these reasons, the time to study the automatic control algorithms possibilities is less than the recommended time to do it.

It is noteworthy that, for the reasons abovementioned, the study of automatic control algorithms performed on this project is not final. It can be update in the future research projects.

8.1. Scan time

For choosing the suitable control algorithm is very important know, before that, the runtime for all tasks executed from the main application. In this way, it is possible calculate the scan time of the control process. For this reason, a thorough analysis of the runtime for the main tasks is included on this project.

Are listed below the tasks included on this study:

1. Image processing task (newFrame(), Car Position)
2. Race data calculation task (setRaceData(), Calculator)
3. Send an integer data to the device control (write(), SerialPort Access)
4. Refresh GUI (JSIlot Screen)

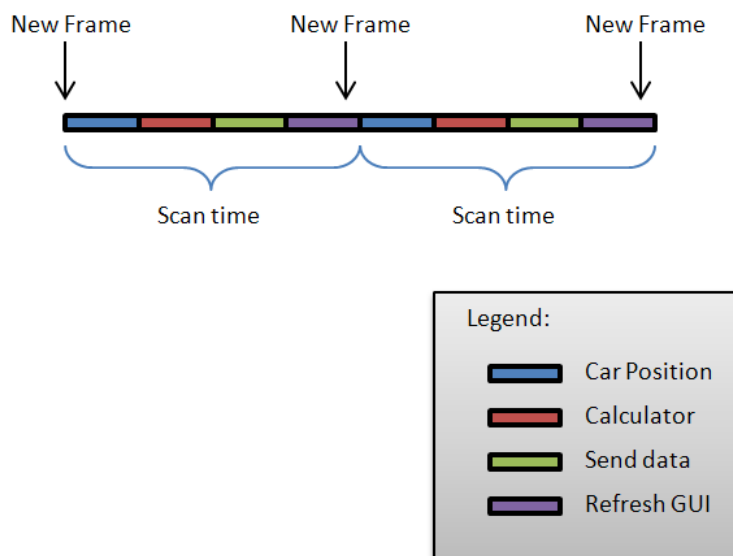


Fig. 41. Scan cycle map

8.1.1. Image processing task

Image processing task is used to locate de slot car in the circuit. These data can be use later to calculate some race data related to the speed. These data can also be used to create a SetPoint (SP) to control the slot car.

It is important to note that the ROI size affect directly to the runtime of the image processing task.

It is important to note that the ROI size affect directly to the runtime of the image processing task. Thus, if ROI size is too large, the runtime for image processing task increases and consequently so does the Scan time. Beside this, when the ROI size increases so does the speed limit to avoid losing the trace of the slot car. This is because by increasing the ROI size so does the distance traveled by the car to go from a ROI to the next.

Due to the ROI size dependence with the runtime of the image processing task, it has decided to do a study about the runtime to carry out this task for several ROI size.

Below is the aforementioned study. For this study was took 200 samples of the runtime to execute the image processing task , for a ROI size between 40 and 90 pixels, increasing the size to 5 pixels for the next sampling.

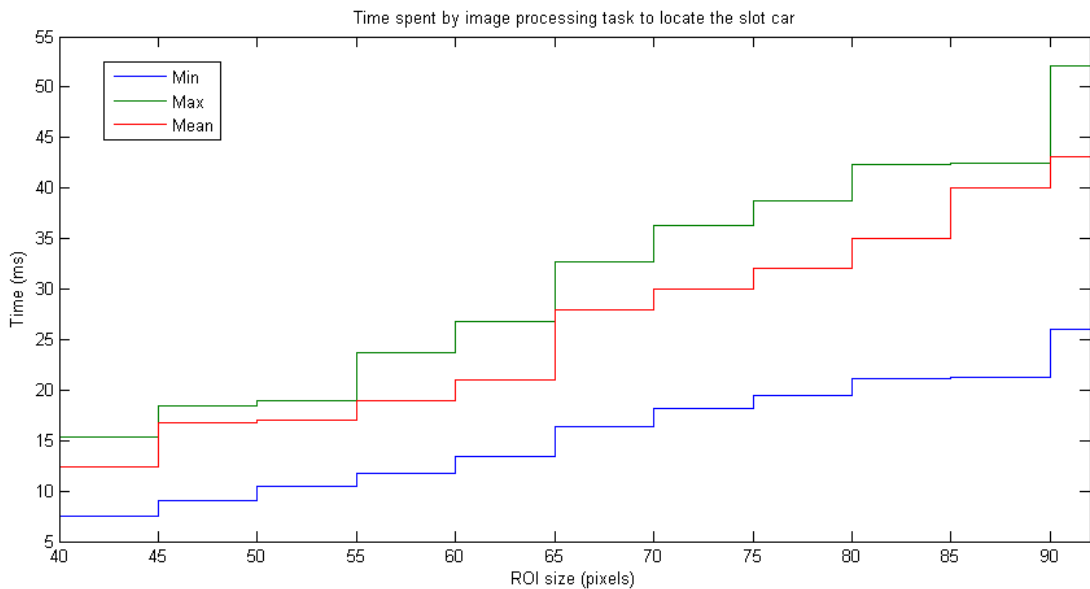


Fig. 42. Time spent by image processing task to locate the slot car

As you can see in the above figure, indeed when ROI size increases does so the runtime of the image processing task.

Following is a table which lists the most important data in the graph above:

ROI size (px)	Min (ms)	Mean (ms)	Max (ms)
40	7.5	12.35	15.36
45	8.9	16.8	18.48
50	10.5	17.06	19
55	11.7	19.1	23.65
60	13.38	21.03	26.76
65	16.38	28.12	32.76
70	18.15	30	36.29
75	19.41	32.02	38.76
80	21.18	34.88	42.4
85	21.25	39.97	42.5
90	26.05	43.12	52.1

Once the data presented above have been analyzed, both the chart and the table, it is concluded that the runtime increase is not proportional for the ROI size increases. Beside this, it is not there any pattern for that.

This is because the PC working conditions may change slightly, when this study was performed, between the moment when was took the samples for a ROI size and the moment when was took it for the next ROI size. This is related to the available RAM memory for use by Java. This available RAM change depending to the external tasks that the PC is executing at that time. If runtime value of the image processing task had been large (hundreds milliseconds or seconds units), this changes would have not affected to the results. In contrast, when the runtime to study is around tens or units of milliseconds, a small change of the available RAM value may increase the runtime.

8.1.2. Race data calculation task

Unlike to the image processing task, in this case, the runtime to calculate the race data not depend of the ROI size. For this reason, it is not imperative make the study for several ROI size as has been done in the previous section. Therefore, the results of this study are valid to any ROI size.

Following you can see the results of this study graphically. To carry out the study, 200 samples were taken. For each sample is included the runtime to calculate the most important data of the current race (speed, average speed, top speed, fast lap, and so on).

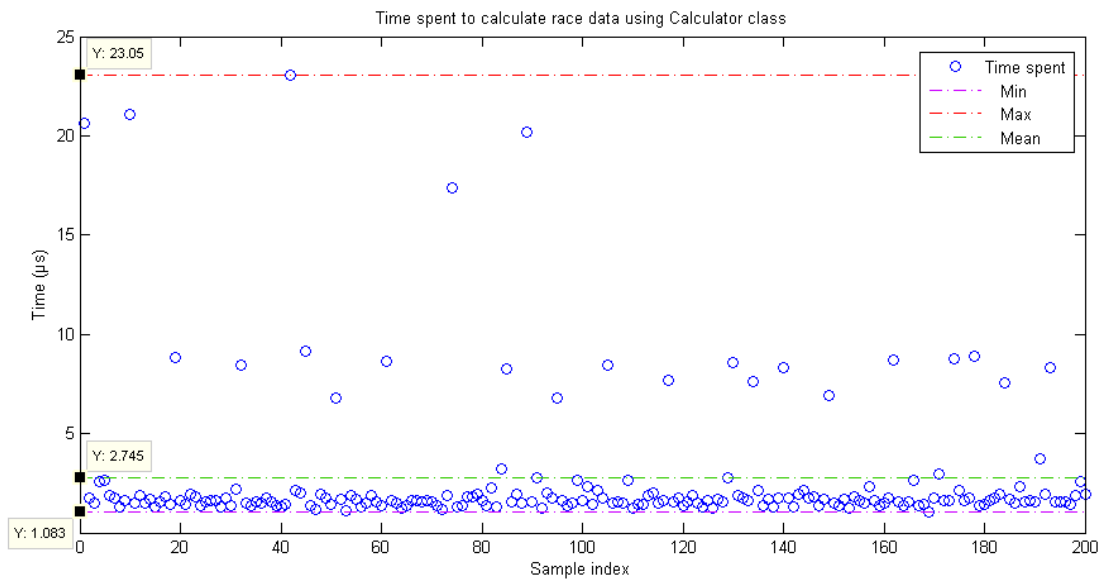


Fig. 43. Time spent to calculate race data using Calculator class.

It is important to note that, as you can see in the picture above, the bulk of the samples are located around the mean value. Therefore, the scatter of data is not high. Even so, some samples values are to 8 times larger than the mean value.

When a sample is located far from the mean value, this sample is called outlier. To calculate the Scan time is very important keep in mind the outliers, specifically the maximum value, because otherwise is possible that slot car tracking will be lost.

Following is shown a box chart where you can see the scatter data level for this study.

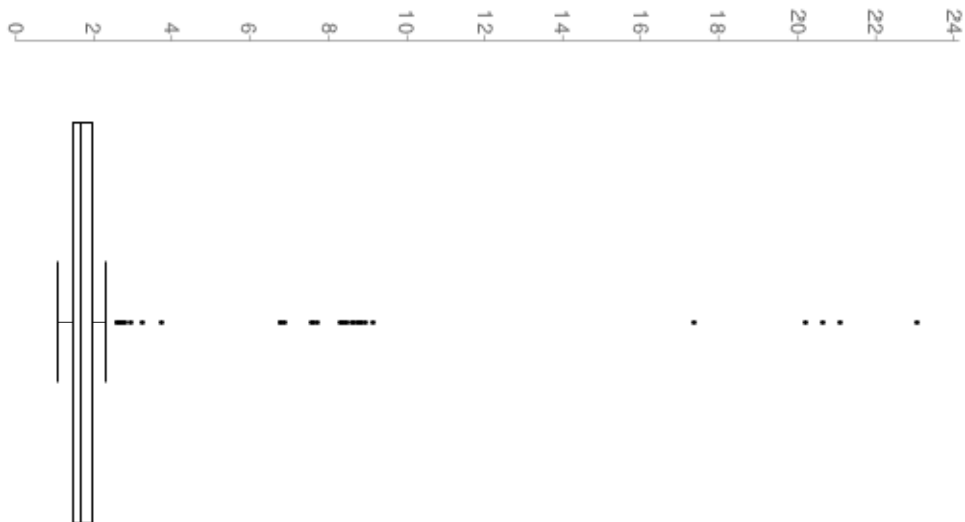


Fig. 44. Time spent to calculate race data using Calculator class. Boxplot

The maximum runtime value is, as you can see in the results above, 23 μ s. Because this runtime is too small compared with the other tasks, it can be neglected for the Scan time.

8.1.3. Send an integer data to the control device

In order to continue the runtime study for the several tasks executed for each scan cycle, through this section is carried out the study to get the runtime of the task responsible for send an integer data to the control device. The same way as a previous section, this runtime not depend of the ROI size. Therefore, the results of this study are valid to any ROI size. To make this study was took 200 samples. Following is presented the results of this study:

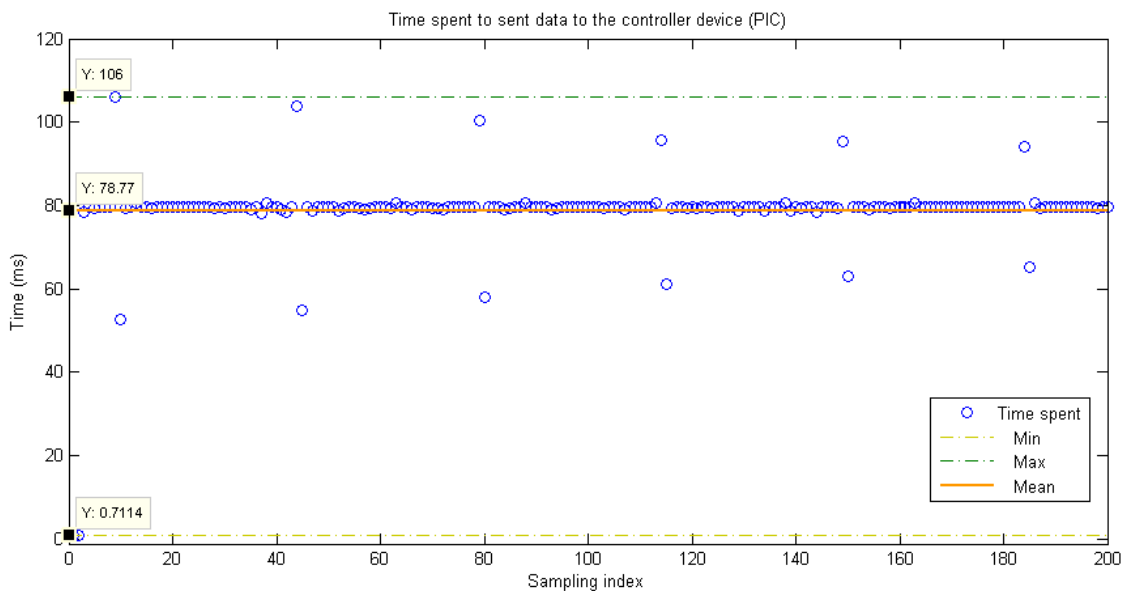


Fig. 45. Time spent to sent data to the controller device (PIC).

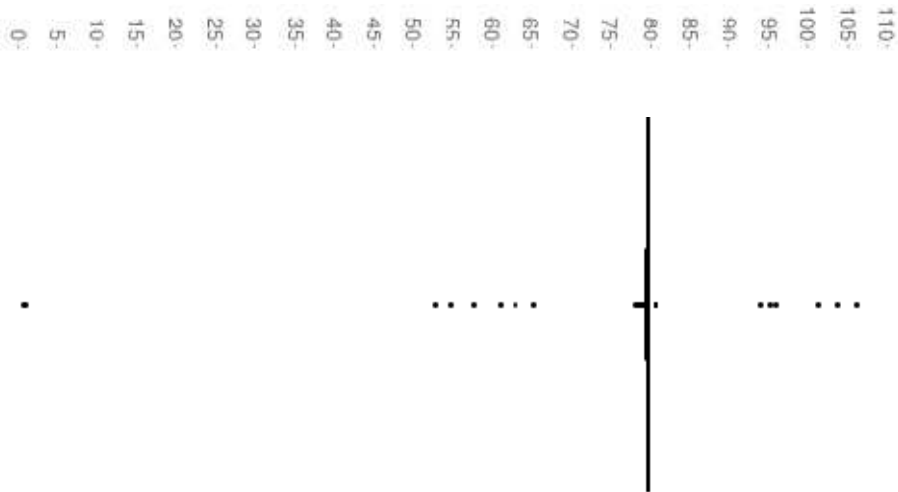


Fig. 46. Time spent to sent data to the controller device (PIC). Boxplot.

As you can see on the above charts, the bulk of samples are placed around the mean value. The mean value for the runtime of this task is 79 ms and, the maximum value is 108 ms. In order to make a trusted control application, it is imperative use the maximum value (108 ms) for calculate the Scan time.

8.1.4. Refresh GUI

Finally, to complete the Scan time calculation, it is necessary to measure the runtime for Refresh the visual interface. This task is the most adaptable for removing some task, if it is necessary, in order to reduce the runtime.

It is very important keep in mind that this study was performed for the current visual components of the application. Therefore, it is know that when some visual components are added or deleted to the visual interface, this change affect to the runtime of the task and consequently to the Scan time.

Below are the results of the abovementioned study presented graphically:

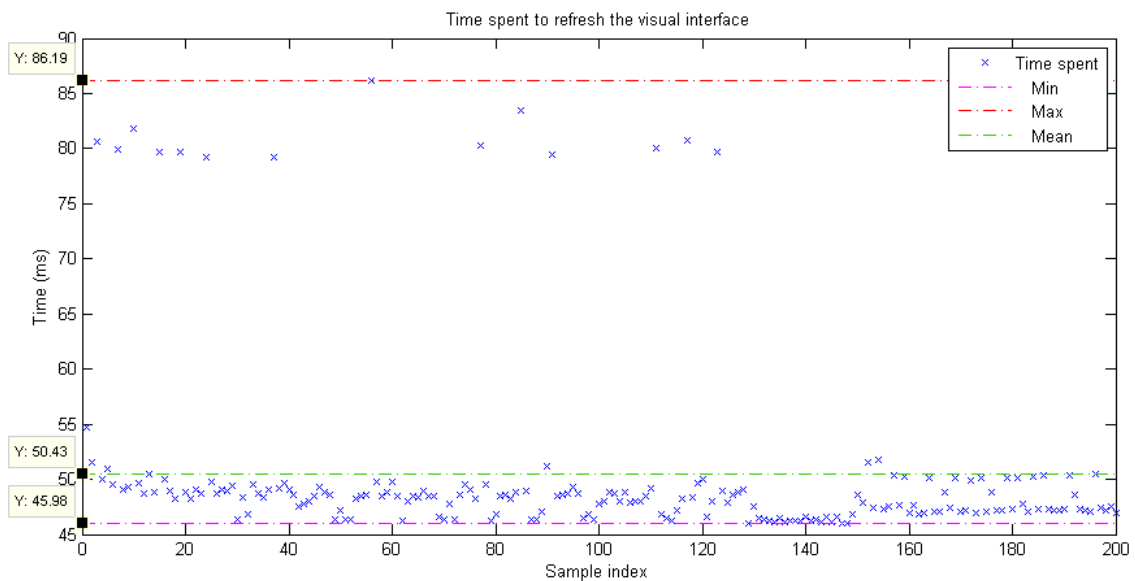


Fig. 47. Time spent to refresh the visual interface.

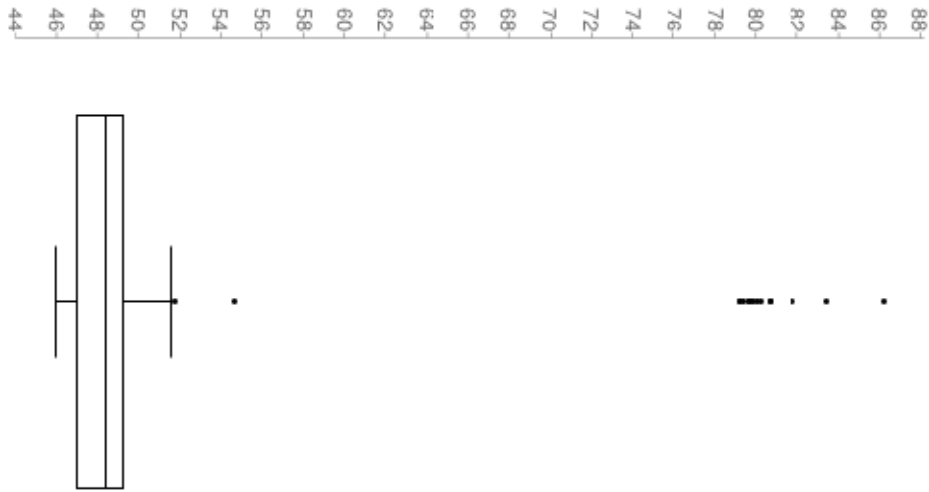


Fig. 48. Time spent to refresh the visual interface. Boxplot.

As you can see on the above charts, the bulk of samples are placed around the mean value. The mean value for the runtime of this task is 50 ms and, the maximum value is 86 ms. In order to make a trusted control application, it is imperative use the maximum value (86 ms) for calculate the Scan time.

8.1.5. Conclusions about Scan time

Once the study of runtime of the several tasks has been finished, now is time to collect the results and draw conclusions. For this reason, it is presented above a table where these results are collected. Beside this, it is included the Scan cycle map where you can see some important data, for example: the tasks performed for a scan cycle, the runtime for each task or the scan time value.

ROI size (px)	Car position (ms)	Calculator (μ s)	Send data (ms)	Refresh GUI (ms)	Scan time (ms)
40	15.36	23.05	106	86.19	207.57
45	18.48				226.05
50	19				226.57
55	23.65				231.22
60	26.76				234.33
65	32.76				240.33
70	36.29				243.86
75	38.76				246.33
80	42.4				249.97
85	42.5				250.07
90	52.1				259.67

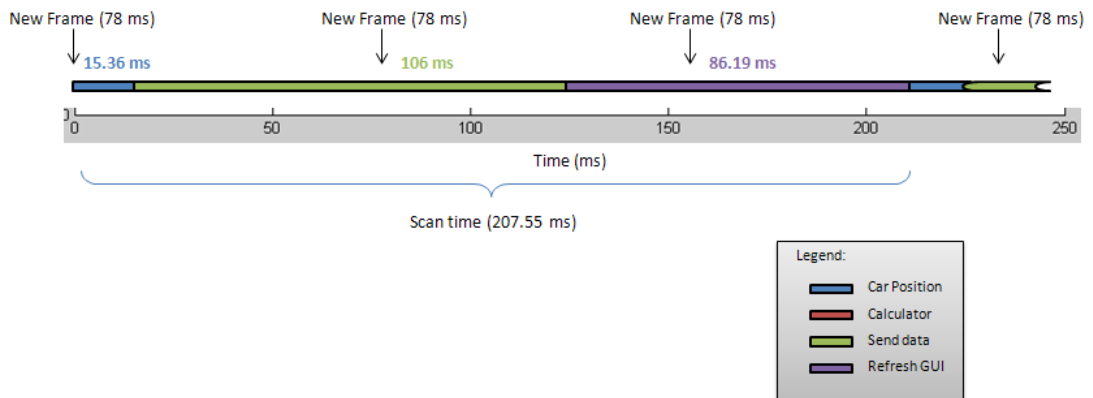


Fig. 49. Scan time.

It is important to note that the Scan cycle map presented above was performed for a unique case (40 pixels). Even so, the Scan cycle map for other ROI sizes only changes, respect to the abovementioned Scan cycle map, the runtime of the image processing task (Car Position). Also note that, these values are collected in the table above and is not a problem imagine the other scan cycle maps.

8.2. Irregular ROI zoning

Until now a regular ROI distribution in the circuit was used. During development of section number 7.2.1 of this project, on page 77 it was concluded that the appropriate ROI size, to avoid loss tracking of the slot car, was 84 pixels.

Even so, it is important keep in mind the fact that, when these abovementioned tests were done, it has not been included the task responsible for sending the control data to the control device (PIC). As a result, the scan time for the case where it was not included the control task is less than the current case where this task is included. Thus, some visual components were added later of the test performing. Therefore, the results of the test (ROI size = 84 px) can be not enough to implement the added tasks.

In order to know the ROI size for the current Scan time, it is important calculate previously the maximum distance traveled by the slot car without losing their tracking.

$$\underbrace{d}_{40 \text{ px}} = v \cdot t = V_{max} \cdot ScanTime = 343.3 \frac{cm}{s} * 207.55 \text{ ms} = 71.25 \text{ cm}$$

$$ROI \text{ size} = \frac{71.25 \text{ mm}}{2.6 \text{ mm/px}} = 274 \text{ px}$$

Once the ROI size is calculated using the current Scan time, one can say that the result is too large. Furthermore, you have to keep in mind that for this calculation has been used the runtime of the image processing task for 40 pixels. Therefore, if we want to use the ROI size equals to 274 pixels, the Scan time increases and so do the ROI size again. It necessary repeats this iterative calculation again and again until the ROI size as a result is the same value that used ROI size used to calculate the Scan time.

Because the calculated ROI size is too large is not feasible implement it because a single ROI region needs too much space and consequently the information for any ROI area is inaccurate. Below you can see the zoning for regular ROI distribution using ROI size equals to 274 pixels:

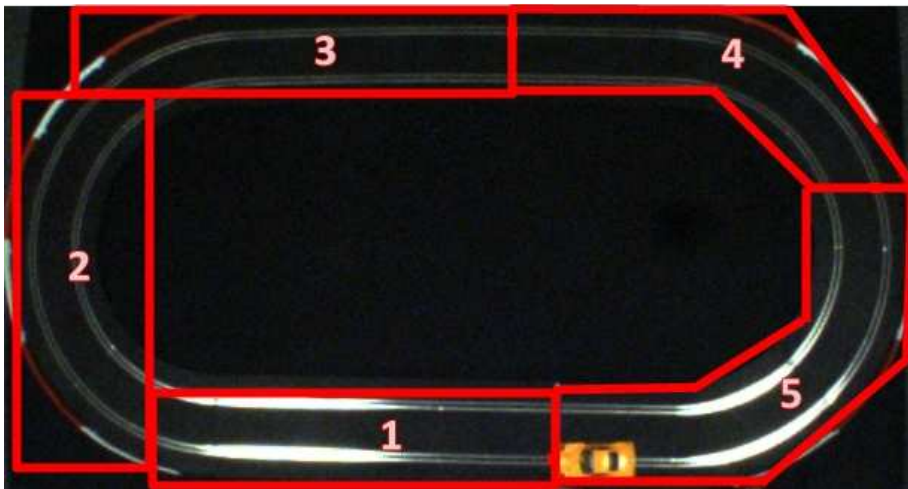


Fig. 50. ROI size using the results of Scan time.

To avoid using such large ROI sizes it is possible change the way to place the ROI areas. Rather than using regular ROI distribution, it has decided to use an irregular zoning.

The irregular zoning or irregular ROI distribution is based on the idea of taking information only in the most important areas. In this case, the most important areas are related to the finish line and related to the start of racetrack bends and the final of this bends.

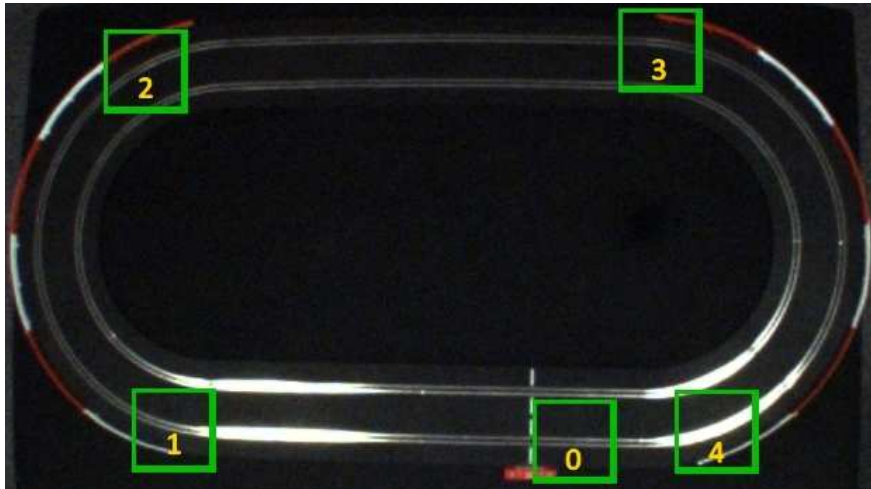


Fig. 51. Irregular zoning idea.

ROI index	Location
Pos. 0	(413, 320)
Pos. 1	(128, 312)
Pos. 2	(108, 56)
Pos. 3	(474, 41)
Pos. 4	(514, 313)

The most important advantage of the irregular zoning respect to the regular is related to the fact that the distance between a ROI and the next is increased and, therefore, the Scan time is less restrictive. In this way, the application has more time available to execute the tasks when the slot car is traveled from a ROI to the next. It is important to note that the distance between a ROI and the next is not constant and, consequently the speed limit depends directly to the racetrack section where is placed the car at that time. If the distance between a ROI and the next is small, the slot car has to move slowly but, instead, if this distance is not small the slot car can move fast.

8.3. Data sent to PIC – Speed ratio

Throughout this project, in previous sections, to speak about the control action to drive the car it was referred to it as an integer value (0-1023) that is sent to the PIC of the control device. The purpose of this section is to know the ratio of this integer data sent to the control device and the slot car speed.

In order to find this abovementioned ratio, it was performed a statistic study using 200 samples for it. For this study the following steps were carried out: first an integer data is sent to the control device (PIC) and when the slot car is traveling in the circuit, the camera records this travel and stores the video. Then, sent data value is added to 50 and the camera records it again. This process is repeated again and again until the sent value is equals to 1023 (maximum value). When this process is finished, it is necessary analyze these videos using an editor video software, frame by frame. VideoPad has been the editor video used for this study. Using the distance traveled (pixels) by the car and the time spent to do it, you can calculate the speed car, on straight sections and bends, for each sent value.

As expected, for the same integer value sent to the control device, the speed on bends is not the same that the speed on straight sections. For this reason, it is necessary carry out the abovementioned study both for straight sections and bends.

This data is based on the spent time to travel on a straight or a whole bend, 350 and 440 pixels for each. Below are the tables where are collected the previous results of the abovementioned research:

Study of speed-SP connection for straight sections		
SP value	Route time (s)	Speed (cm/s)
0 (0%)	∞	0
50 (5%)	∞	0
100 (10%)	∞	0
150 (15%)	∞	0
200 (20%)	6.6	13.8
250 (24%)	2.6	35
300 (29%)	1.5	60.7
350 (34%)	1.3	70
400 (39%)	1	91
450 (44%)	0.8	113.8
500 (49%)	0.7	130
550 (54%)	0.5	182
600 (59%)	0.4	227.5
650 (64%)	0.38	239.47
700 (68%)	0.3	303.3
750 (73%)	0.28	325
800 (78%)	0.24	379.17
850 (83%)	0.23	395.65
900 (88%)	0.22	413.6
950 (93%)	0.21	433.3
1023 (100%)	0.2	455

Study of speed/SP connection for bends			
SP value	Route time (s)	Speed (cm/s)	The Slot car crashed?
0 (0%)	∞	0	NO
50 (5%)	∞	0	NO
100 (10%)	∞	0	NO
150 (15%)	∞	0	NO
200 (20%)	∞	0	NO
250 (24%)	∞	0	NO
300 (29%)	∞	0	NO
350 (34%)	3 *	14.7	NO
400 (39%)	3	38.1	NO
450 (44%)	1.4	81.7	NO
500 (49%)	1	114.4	NO
550 (54%)	0.8	143	NO
600 (59%)	0.7	163.4	NO
650 (64%)	0.6	190.7	NO
700 (68%)	0.55	208	NO
750 (73%)	0.42	272.4	NO
800 (78%)	0.39	293.3	YES (11th lap)
850 (83%)	0.38	301.1	YES (5th lap)
900 (88%)	0.36	317.9	YES (1st lap)
950 (93%)	0.42	272.4 **	YES (1st lap)
1023 (100%)	0.5	228.8 **	YES (1st lap)

* The applied voltage of the car was not enough to complete the whole bend.

** The speed decreases in despite of the applied voltage is higher. As the car skidded during the whole travel, it has rubbed the bend of the protective band. Then, the speed of the car has been reduced significantly.

In order to facilitate the understanding of the previous data it has been performed a graphic where you can see the evolution of the slot car speed according as the applied voltage increases.

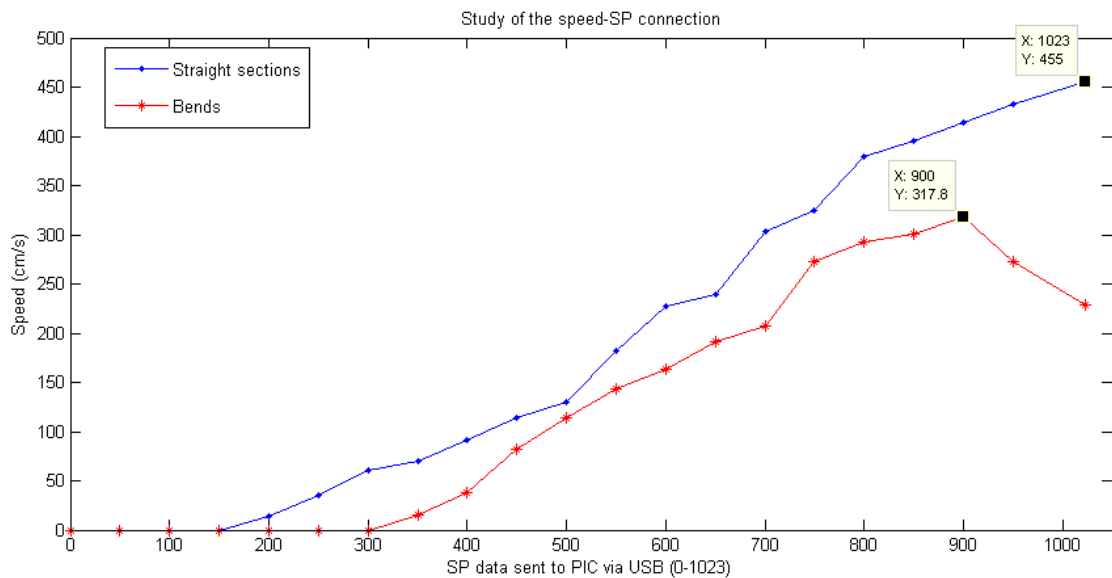


Fig. 52. Study of the speed-SP ratio

After analyzing the previous graphic, here it has been drawn the following conclusions:

- On a straight, it is necessary to apply, at least, the 20% of the voltage range to move the car (data sent to the PIC equals to 200).
- Instead, at a bend, in order to move the slot car it is necessary to apply more than 34% of the maximum voltage (data sent to the PIC equal to 350).
- On a bend, if the integer data sent to the control device (PIC) is higher than 900 (applied voltage higher than the 88%), the car speed decreases. This is due to, as the speed is so high, the car skids and rubs with the protector band placed on the bends of the circuit.
- The results are not proportional but, even so, there is linear progress for some sections. Because of this linearity for sections, it is possible calculate

an approximate ratio to estimate theoretically the speed of the car when the maximum voltage is applied.

- As expected, the speed on straights is higher than the speed on bends.

Below is a table that collect the results as a ratio that relate the integer data sent to the control device and the slot car speed.

SP value	Ratio (SP/speed)* for straights	Ratio (SP/speed)* for bends
0 (0%)	-	-
50 (5%)	-	-
100 (10%)	-	-
150 (15%)	-	-
200 (20%)	3.62	-
250 (24%)	2.86	-
300 (29%)	2.47	-
350 (34%)	2.86	3.4
400 (39%)	2.75	2.62
450 (44%)	2.64	1.84
500 (49%)	2.69	1.75
550 (54%)	2.2	1.75
600 (59%)	1.98	1.84
650 (64%)	2.09	1.84
700 (68%)	1.81	1.92
750 (73%)	1.85	1.65
800 (78%)	1.71	1.70
850 (83%)	1.77	1.83
900 (88%)	1.81	1.89
950 (93%)	1.85	2.39**
1023 (100%)	1.92	3.16**

* To calculate the ratio has been divided the SP data sent to the control device (PIC) by the speed value. Previously, the offset caused by the force of static friction between circuit and the car was subtracted.

** These data cannot be used because, as explained above, in these sections the car rubbed with the protection band.

After analyzed the ratios calculated above, it can be conclude that the way to estimate the car speed for a data sent to the control device will be as follows:

On a straight:

- If the sent data is equal or less to 200 (applied voltage <20% of the voltage range), the car will not move or will stop soon, then, $v = 0 \text{ cm/s}$
- If the sent data ranges is between 200 and 500 (20% - 49% of the motor voltage), the speed will be calculated in the following way: $v \text{ (cm/s)} = \frac{\text{data}}{2.67} \pm 7.5\%$
- If the sent data is higher to 500 (applied voltage >49% of the voltage range), the speed will be calculated in the following way: $v \text{ (cm/s)} = \frac{\text{data}}{1.9} \pm 15\%$

On a bend:

- If the sent data is less than 350 (<34% of the voltage), the car will not move, then, $v = 0 \text{ cm/s}$
- When the data is between 350 and 400 (34% - 39%), there not sufficient linearity to ensure that is possible to calculate the estimated value of speed with a minimum of accuracy.
- If the data is between 450 and 900 (44% - 88%), the theoretical estimate of the speed will be done with the following expression: $v \text{ (cm/s)} = \frac{\text{data}}{1.79} \pm 7.6\%$
- Finally, if data is higher than 900 (>88%) cannot do an estimation because, as already mentioned before, the friction with the protection band influences on the slot car speed.

8.4. Speed limit for irregular zoning

In contrast to regular ROI distribution, when irregular zoning is used on the circuit, the distance between a ROI and the next is not constant. This fact affects to the speed limit of the slot car. Now, this speed limit depends not only whether the slot car is located in a straight or in a bend but depends to the circuit stretch where is located the car at the time as well. Thus, when a ROI is placed near to the next ROI, the speed limit will be less than when these ROIs are placed farther away.

As you can see in the graphic documentation presented in section 8.2 of this document, page 91, the stretch where the distance is the smaller is the 4-1 stretch, which includes the area from the end of the last bend to the finish line. Therefore, this stretch is the most restrictive as to the speed limit.

Using the length values for each stretch and the scan time data, it is possible calculate the speed limit for these sections. To carry out these calculations it is necessary use the next math expression:

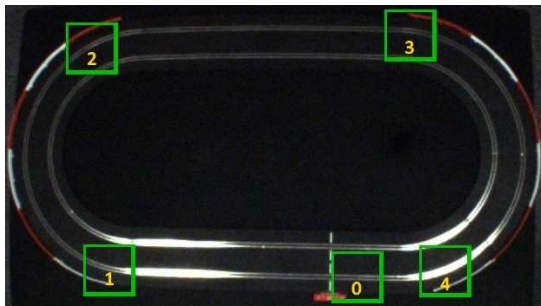
$$v (m/s) = \frac{d(mm)}{Scan\ time\ (ms)}$$

$$1\ pixel \rightarrow 2.6\ mm$$

$$Scan\ time = 207.55\ ms$$

Below is a table where is collected the results to calculate the speed limit for each stretch using the abovementioned math expression:

Stretch	Length (px)	Limit speed (cm/s)
0-1	285	357
1-2	373	467
2-3	370	463
3-4	402	503
4-0	105	131



It is important to note that these speed limit values are related to the speed to avoid the tracking loss of the slot car. Thus, these values are not related to the speed of their motor. Noteworthy that the speed of the car on the circuit, provided by their motor, was studied in other sections of this project.

8.5. ROI size to irregular zoning.

When an irregular ROI distribution or irregular zoning is used to locate the slot car, the ROI size not depends to the Scan time. In this case, it depends only on the runtime of the tasks done to locate the car. In contrast, when a regular ROI distribution was used to locate the slot car, the ROI size depends only to the Scan time because the distance between a ROI and the next is always the same. In short, for regular ROI distribution the ROI size is always the same but, instead, for the irregular zoning the ROI size can change depending to the distance between ROIs.

Therefore, in the present case, it is important to design the ROI size regardless the scan time. You have to ensure that for each ROIs will be possible take a picture at least. Once a picture is taken, at the time as the slot car travel from a ROI to the next ROI it is possible run the other tasks (calculate race data, send data to control device, refresh GUI, and so on). Remember that, the maximum time spent to take a picture using a camera Access object was, according to a study presented in section 7.1.2 of this project on page 69, 78 ms. In order to calculate the ROI size for irregular zoning you have to use the following math expressions:

$$\underbrace{d}_{i \text{ px}} = v \cdot t = v \cdot 78 \text{ ms}$$

$$ROI \text{ size} = \frac{d}{2.6 \text{ mm/px}}$$

It is important to note that, in this case, it is necessary have a different ROI size for each ROI index because the speed limit is not the same between a stretch and other stretch.

Following is a table where is collected the results to calculate the ROI size for each circuit area using the speed limit value for the respective stretch:

ROI index	Speed1 (cm/s)*	Speed 2 (cm/s)**	Speed (cm/s)***	ROI size
0	357	455 (recta)	357	108
1	467	318 (curva)	318	96
2	467	455 (recta)	455	137
3	503	318 (curva)	318	96
4	357	455 (recta)	357	108

* This is the speed limit to avoid the tracking loss of the slot car.

** This speed is related to the speed limit that can achieve the slot car physically on the circuit.

*** Both abovementioned types of speed values are compared and, then, the smallest value (the most restrictive value on the circuit) is used to calculate the ROI size.

8.6. Look-up table

At this moment all the necessary information has been obtained to determine the control algorithm to use. In this case it was thought appropriate to use an open-loop automatic control algorithm based on the look-up table idea.

The look-up table idea is simple. It is used an indexed table where through an input value (key) is obtained an output value related to that input index. In this case the input value will be related with the current position of the slot car. On the other hand, the output will be the integer data to send to the control device (PIC).

The major advantage of this system is the runtime to generate a set point (SP) is smaller. The fact that it is not necessary any math calculation is the cause of the previously mentioned advantage. The system only has to do a search on the look-up table for that.

A disadvantage of this algorithm is, due to the loop is not closed, it is not possible calibrate the SP depending to the slot car speed. For this reason, when the look-up table is build, it is very important keep in mind that if the Setpoint (SP) is so high the car may crash. If this happens, it is not possible adjust the speed automatically and solve the problem.

Following you can see the look-up table look:

ROI index	Speed (cm/s)*	sector	SP data**
0	357	Straight	750
1	467	Bend	900
2	463	Straight	1023
3	503	Bend	900
4	131	Straight	500

* Data taken from the table of the section 8.4 of this project, on page 99.

** Data taken from the tables of the section 8.3, on pages 93-94.

The data of the first column (ROI index) are the input values (keys) to access to look-up table and the last column (SP data) are output values that will be sent to the control device (PIC).

9. USER GUIDE

This chapter was performed in order to facilitate the use of JSlot Project application. On this section, you can find how to do the previous settings needed to use this application, how to perform the calibration tasks, the correct way to run JSlot and some questions to keep in mind about this and, finally, a troubleshooting designed to solve the most likely problems that can appear when JSlot platform will be used.

9.1. Settings

What you need?:

To use control and monitoring slot system, you need:

- A. Super Series Circuit Kit of NINCO.
- B. Bayer DBK21AU04 USB Camera.
- C. Varifocal lens with CS Mount.
- D. Lexus 430 SC Orange Slot Car.
- E. USB 2.0 Cable with type A connector and type B connector.
- F. Interface PC-SlotCircuit Device.
- G. Cable with two Jack 3.5 mm plugs.
- H. PC to run Java Project.



Fig. 53. What you need to use control and monitoring slot system.

The devices identifiers that you can see in the picture above will be used in the following sections.

In addition to physical items above mentioned, to use control and monitoring slot system, you have to install a few software and libraries on the PC.

- DBK21AU04 Camera drivers of TheImagingSource.
- IC Capture.
- Interface PC-SlotCircuit drivers (CDC Drivers).

- JAVA.
- Libraries to connect to the camera from Java (LTI-CIVIL).
- Libraries to connect to the Serial Port from Java (RXTX).
- GIMP or similar (optional).
- NetBeans or similar (optional).

WARNING: Optional software provides an easy way to make changes in the Slot java project.

INSTALLATION:

DBK21AU04 Camera drivers of TheImagingSource:

If this is the first time you connect the camera (B) to the PC (H), the first step you have to do is install the drivers. The camera manufacturer (TheImagingSource) makes available the drivers you need. One way to get the drivers is through the installation CD that you have purchased with the camera. If for any reason you don't have the installation CD you can download the drivers from the manufacturer's website:

http://www.theimagingsource.com/downloads/icwdmuvccamtis.en_US.zip

IC Capture:

When you have installed the DBK21AU04 Camera drivers, you have to install IC Capture software on the PC. IC Capture provides a way to access the real-time camera and save single pictures, a sequence of images or video streams. IC

Capture software, As same as camera drivers, is provided in the installation CD. If you don't find the installation CD you can download IC Capture software from the following website link:

http://www.theimagingsource.com/downloads/iccapturetrial.en_US.zip

Anyway keep in mind that if you need technical support using the camera or updates software you have to go to following website link:

http://www.theimagingsource.com/en_US/support/downloads/

Interface PC-SlotCircuit drivers (CDC Drivers):

The installation of these drivers will be required only if you haven't installed NTPVista.inf drivers on your computer and, when you connect interface PC-SlotCircuit device (F) to the PC (H) using the USB Cable (E), the computer doesn't recognize the device. If this happens, you will get the following message:

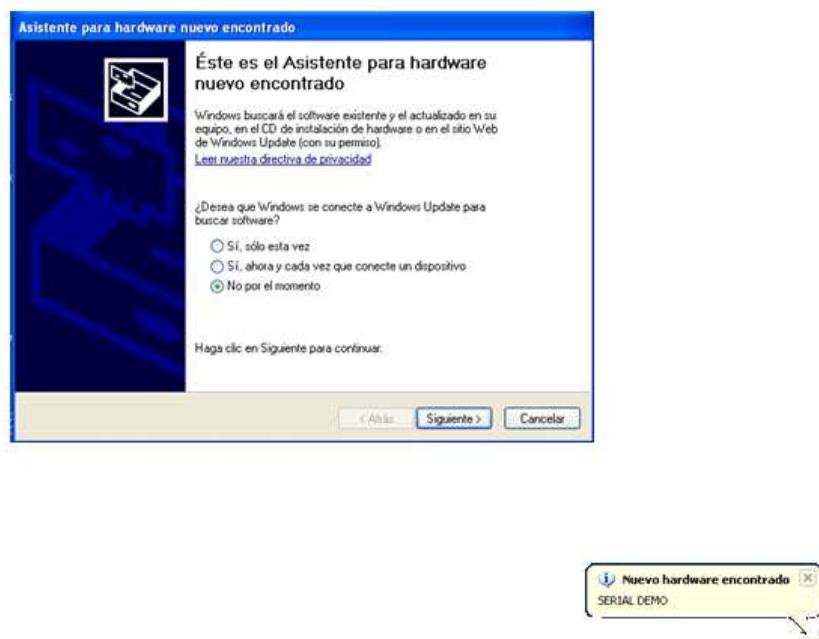


Fig. 54. Interface PC-SlotCircuit drivers (CDC Drivers) (1).

In case that PC doesn't recognize the device connected via USB, as you can see in the picture above, you have to carry out the following steps in order to install the needed drivers. The drivers you need to install can be found on the CD of this project.

To install the drivers you have to select ***“No por el momento”*** on the ***“Asistente para hardware nuevo encontrado”*** and click ***“Siguiente”***.

Then the wizard lets you choose if you want to install the driver automatically or instead you prefer to install the driver from a specific path. In this case you have to choose the second one: ***“Instalar desde una lista o ubicación específica (avanzado)”***.

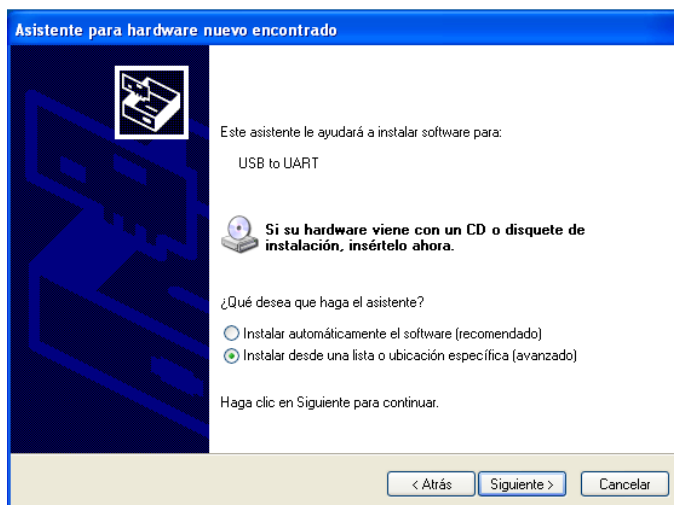


Fig. 55. Interface PC-SlotCircuit drivers (CDC Drivers) (2).

The next step is to specify the path where is the needed driver. You must select the folder containing the **cdc_NTXPVista.inf** file.

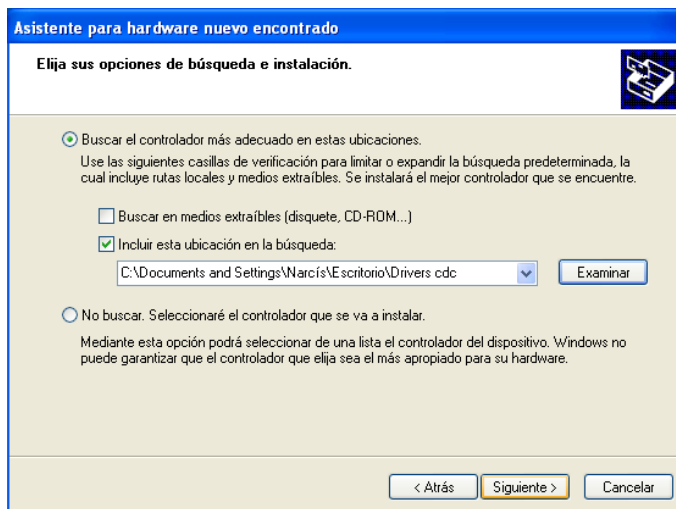


Fig. 56. Interface PC-SlotCircuit drivers (CDC Drivers) (3).

When installing you will see a notice telling you that the software you are installing has not passed Windows Logo testing. You have to ignore this message and click **“Siguiente”** because it is a usual warning and is not dangerous for your system.

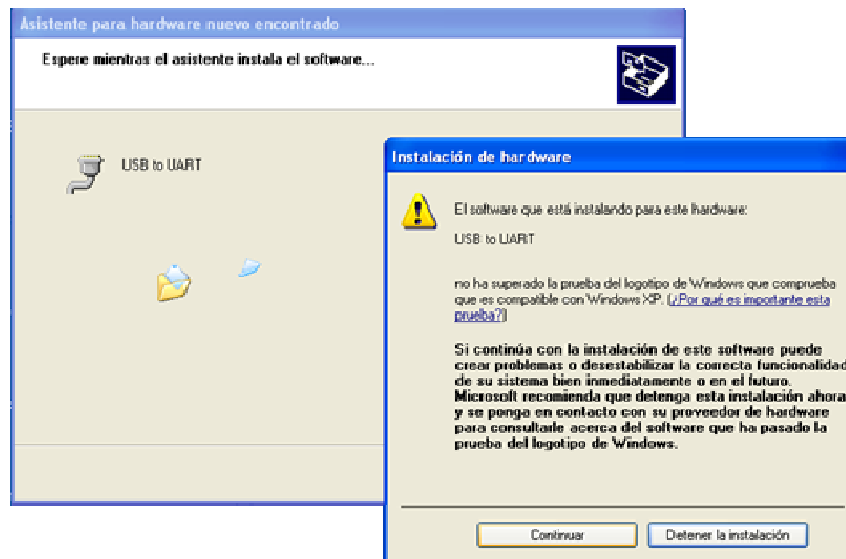


Fig. 57. Interface PC-SlotCircuit drivers (CDC Drivers) (4).

When the installation is complete you will see the following message on your screen:

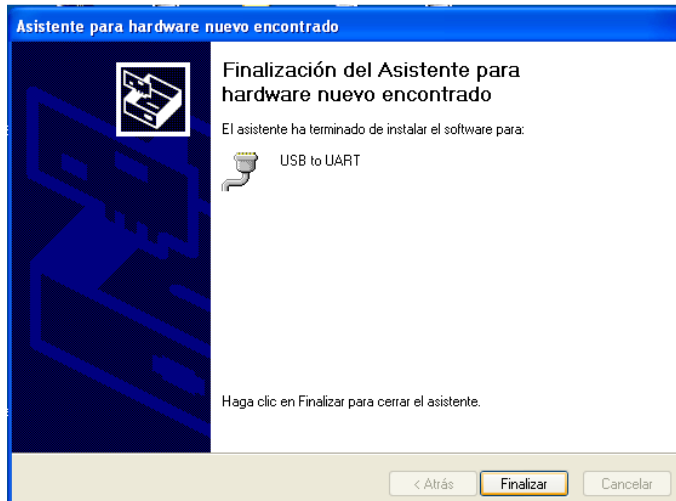


Fig. 58. Interface PC-SlotCircuit drivers (CDC Drivers) (5).

Finally, only need to check that the driver has been installed correctly and that now the system recognizes the Interface PC-SlotCircuit (F) when connected via USB. To do this we need to right click on *“Mi PC”*, select *“Administrar”*, and then go to *“Administrador de dispositivo”* (in the left side of the window). Finally we select the option *“Puertos COM & LPT”* and, if the device has been recognized correctly, you will see the serial port assigned.

You can see that in the case of the image below is assigned COM3 serial port. It should be noted that knowing the serial port that connects the device is required to run the Java application of the project.

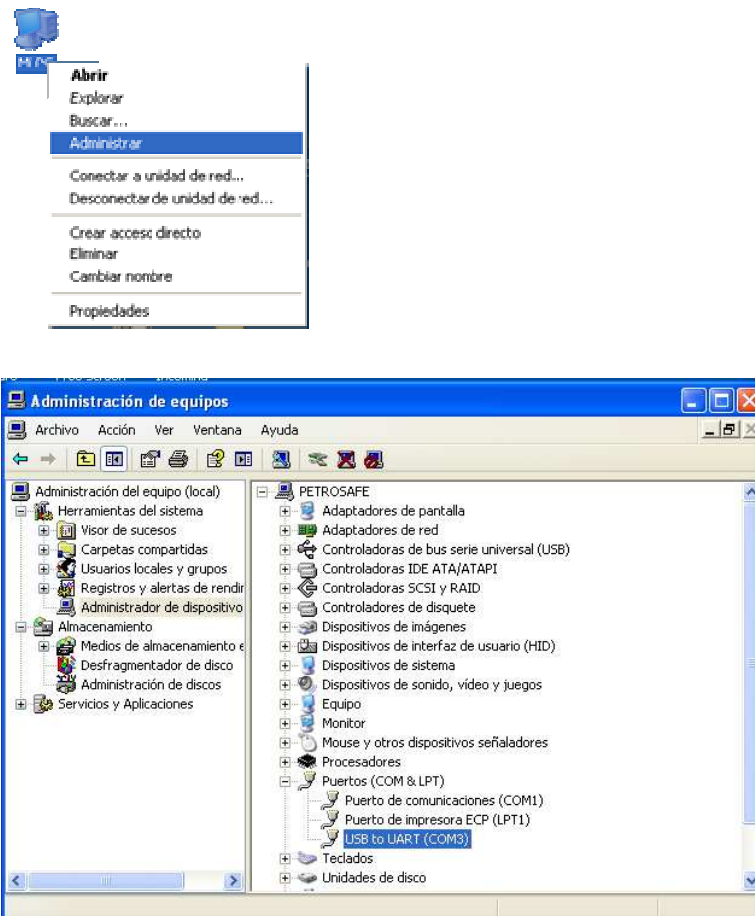


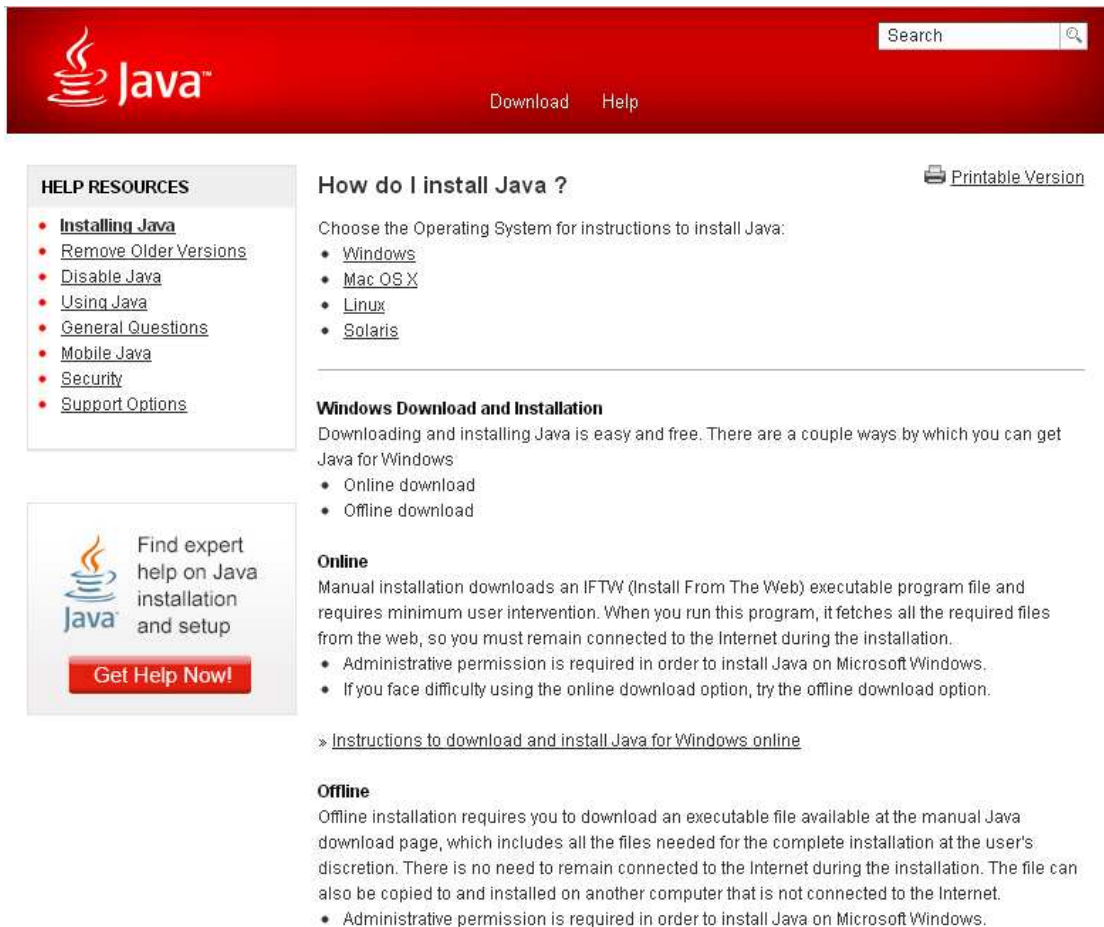
Fig. 59. Interface PC-SlotCircuit drivers (CDC Drivers) (6).

JAVA:

To run control and monitoring slot system project is essential to have installed on your computer the appropriate version of Java. To make the project was used java version 7 update 21. In order to ensure correct running of the Java classes you are advised to use java version 7 update 21 or later.

On the JAVA website you can see a step-by-step installation guide:

http://www.java.com/en/download/help/download_options.xml



HELP RESOURCES

- [Installing Java](#)
- [Remove Older Versions](#)
- [Disable Java](#)
- [Using Java](#)
- [General Questions](#)
- [Mobile Java](#)
- [Security](#)
- [Support Options](#)

How do I install Java ? [Printable Version](#)

Choose the Operating System for instructions to install Java:

- [Windows](#)
- [Mac OS X](#)
- [Linux](#)
- [Solaris](#)

Windows Download and Installation

Downloading and installing Java is easy and free. There are a couple ways by which you can get Java for Windows

- Online download
- Offline download

Online

Manual installation downloads an IFTW (Install From The Web) executable program file and requires minimum user intervention. When you run this program, it fetches all the required files from the web, so you must remain connected to the Internet during the installation.

- Administrative permission is required in order to install Java on Microsoft Windows.
- If you face difficulty using the online download option, try the offline download option.

» [Instructions to download and install Java for Windows online](#)

Offline

Offline installation requires you to download an executable file available at the manual Java download page, which includes all the files needed for the complete installation at the user's discretion. There is no need to remain connected to the Internet during the installation. The file can also be copied to and installed on another computer that is not connected to the Internet.

- Administrative permission is required in order to install Java on Microsoft Windows.

Get Help Now!

Fig. 60. Java web.

9.2. Calibration wizard

Throughout this chapter is intended to guide the user through the steps to be followed to correctly position the slot circuit (workspace) and calibrate lighting conditions.

WORKSPACE ADJUSTMENT:

In order to ensure that the Java application responsible for slot car control has all needed information at all times, you have to be carefully when executing workspace adjustment step. You have to keep in mind that the camera have to take pictures of the full circuit and is not valid a picture of a partial zone of the circuit.

In order that you can carry out this work easily the camera manufacturer offers IC Capture. If you want to know how to install IC Capture go to page 112 of Chapter 1 of this document.

The steps to position the slot circuit are:

1. Run JSlot Project App.
2. Click on “*Edit*” and next in “*Circuit Location*”.
3. When IC Capture is open you have to place the slot circuit inside the capture window.

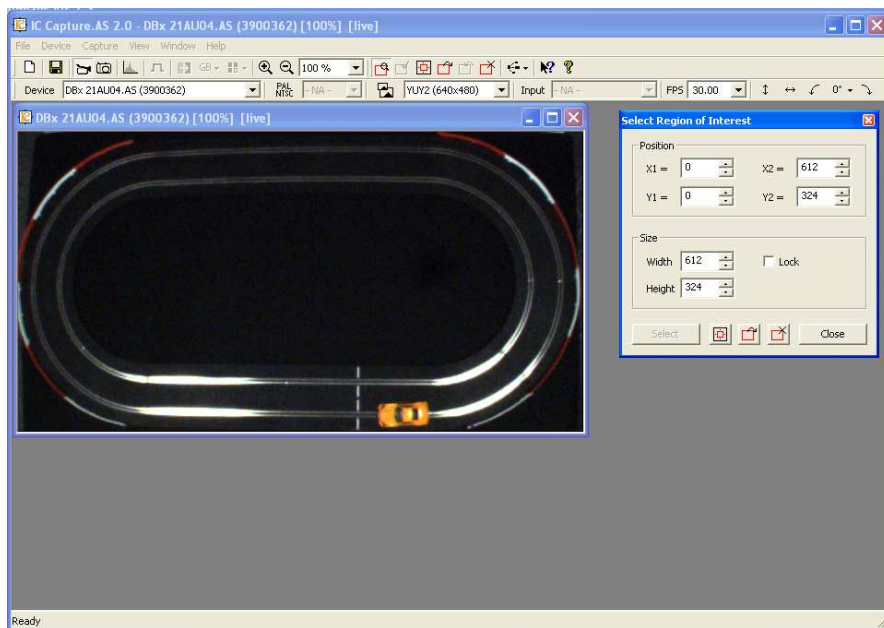


Fig. 61. Workspace adjustment

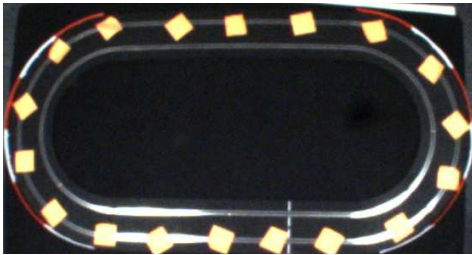
4. Close IC Capture software and return to JSlot Project App.

LIGHTING CONDITIONS CALIBRATION:

A wrong calibration of the lighting conditions may involve the loss of slot car control. It is for this reason that this section is critical to ensure correct system operation.

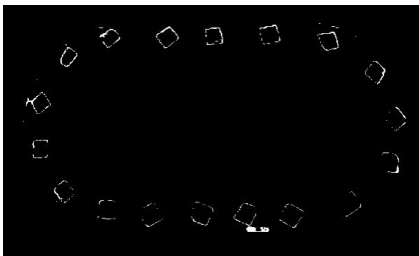
To allow the user to calibrate the lighting conditions was developed, in an earlier project to it, an easy protocol to execute. The steps to follow are:

1. Distribute evenly "Post-it" the same colour as the Slot car, orange.

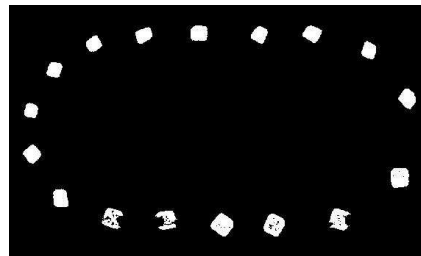


2. Pull down all blinds of the room.
3. Turn on the lights you see fit.
4. Run JSIot Project App if you have not done before.
5. Click on “Edit” and next in “Lighting”.
6. When calibration pictures appear you have to check that calibration is correct using the next standard presented below. You have to check 5 generated pictures doing click on next frame.

Incorrect calibration



Correct calibration



7. If the calibration is incorrect you will have to repeat steps 4-9 changing lights on chosen. Repeat this step as many times as you need before the next step.
8. If the calibration is incorrect you will have to repeat steps 4-9 changing the diaphragm opening of the camera so that it passes more or less light. Repeat this step as many times as you need before the next step. *Read the warning at the end of this chapter before execute this step.*

9. If the calibration is incorrect you will have to repeat steps 4-9 opening and pulling down the blinds as you see fit. You can repeat this step as many times as required.

In tests carried out in the laboratory we have used the following conditions:

- Turn on first and third rows of fluorescent lights (counting from just above row of the slot circuit).
- All blinds pulled down.

WARNING: You have to be carefully when opening or closing diaphragm of the camera because next to this part is the zoom adjustment component. If you ever change the zoom of the camera then you will need to resize ROI region and recalculate the positions of all ROI regions.

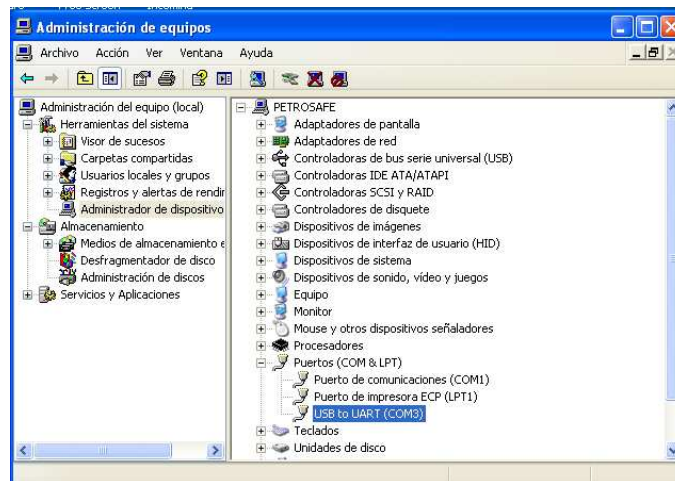
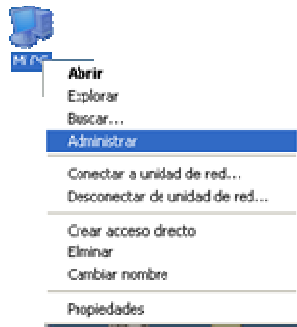
9.3. Getting started

This section aims to show the right way to start JSlot Project application. To run the main application is imperative that you have carried out the processes mentioned in the preceding section.

To start monitoring and control process of the Slot Car, we recommend you take the following steps:

1. Connect “*Interface PC-SlotCircuit Device*” (F) to PC (H) using the USB cable (E).
2. Connect the first plug of the Jack cable (G) to “*Interface PC-SlotCircuit Device*” (F).
3. Connect the other plug of the Jack cable (G) to the outer rail command connector of the circuit.
4. Connect the power grid to the circuit.
5. Check that the selector to choose the race direction is set on clockwise direction.
6. Make sure that are not obstacles in the path of the slot cars.
7. Ensure that the slot car controlled by JSlot Project application, Lexus 430 SC Orange (D), is not on the circuit.
8. Run JSlot Project App if you have not done before.
9. Perform calibration tasks (presented in chapter 2, page “XX” of this user guide).
10. Check the serial port where the device “*Interface PC-SlotCircuit Device*” (F) is connected and insert it into the COM ports dropdown that incorporates the main application.

For this purpose: right click on **Mi PC**, select **Administrar** and follow that go to **Administrador de dispositivo** (on the left side of the window). Finally select Puertos **COM & LPT** and get the COM port assigned.



Now you have to enter the serial port number assigned in the main application and click **“Connect”** button.



11. Place the slot car Lexus 430 SC Orange (D) in the start line. The car has to be located in the outside lane and positioned to move at the clockwise direction.
12. Enter the number of laps you want to assign to the race in the TextField LAPS.
13. Connect the analogue controller of the rival to the circuit. Then put the car in the start line. The car has to be located in the inside lane and positioned to move at the clockwise direction.
14. Press **“Start”** button.



15. Wait at start notified by the set of traffic lights.



16. Enjoy the race.

WARNING: It is very important that you always connect the device “Interface PC-SlotCircuit Device” (F) to other devices in the manner provided on this user guide. Specifically it is the steps number 2, 3, 4 and 5. Otherwise the device may be damaged.

PARTS OF THE VISUAL INTERFACE:



Fig. 62. JSLOT visual interface. Components.

1. Serial ports list where is connected the device "Interface PC-SlotCircuit Device" (F). You have to select the COM port assigned by the PC.
2. Label indicating the total number of laps that comprise the race.
3. Button to connect to the serial port.
4. Label where you can see the status of serial port connection. The value of connection can switch between "ON" and "OFF".
5. Button to close the serial port connection. You only need to press it if you have already set up a serial port and you want to change.
6. Button to start the race.

7. Button to finish the race. If you do not press this button, the race will end when all the race laps are completed.
8. On this label you can see the top speed reached by Lexus 430 SC Orange slot car.
9. On this label you can see the current speed of the slot car controlled by JSlot Project application.
10. On this label you can see the average speed for the last lap.
11. On this label you can see the less lap time achieved by Lexus 430 SC Orange slot car.
12. On this label you can see the elapsed time since the start of the race.
13. On this label you can see the number of laps remaining to finish.
14. This bar indicates the proportional value of the speed of the car in real time.
15. You can use File tab to manage database files. You can choose 3 options: New, save or open. If you click on "New DB" new .txt file will be generated in DB folder. Instead, if you click on "Save DB" the most important data of the current race will be stored in last .txt file generated. Finally, if you choose "Open DB" a file chooser window will appear and you have to select the DB file that you prefer.
16. You can use Edit tab to perform the calibration tasks or clear fields. 3 options are available: Circuit location, Lighting and Clear fields. If you select "Circuit location" option IC Capture will run in order to calibrate the workspace. Instead, if you click in "Lighting" the camera will take 5 pictures. You have to analyze these pictures to determine if lighting conditions are acceptable or not. However, if you choose "Clear fields" option, the data of last race will be removed from the window fields. If you

want know more about calibration tasks and how do it you have to read chapter number 9.2. “*Calibration wizard*” of this project, on page 113.

17. Using “*Help*” tab, you can open this user guide from JSLOT Project App. You can choice this document in English (user guide) or Spanish (manual de instrucciones).

HOW TO STOP USING THE APPLICATION?

When you want to finish using JSLOT Project application you have to follow the next steps:

1. Click on the “red cross” located in the right-up corner of the visual interface window in order to close the application. If the race isn’t finished yet, we recommend you click finish button before close the window.
2. Disconnect the circuit to the power grid.
3. Disconnect the first plug of the Jack cable (G) that is connected to the circuit.
4. Disconnect the other plug of the Jack cable (G) that is connected to the device “Interface PC-SlotCircuit Device” (F).
5. Disconnect the USB cable (E) to PC (H) and then to the device “Interface PC-SlotCircuit Device” (F).
6. To finish, disconnect the other components of the system that are still pending.

WARNING: It is very important that you always disconnect the devices of the system in the manner provided on this user guide. Specifically it is the steps number 3, 4, 5 and 6. Otherwise the device “*interfaz entre PC y circuito*” (F) may be damaged.

9.4. Troubleshooting

This chapter is a troubleshooting to help you to solve some frequent problems of the control and monitoring system of the slot car.

Trough this section you can see solutions for the next problems:

- **Problem 1:** When IC Capture software is running and I want carry out the workspace adjustment process, I cannot watch the camera filming in real time.
Solution on page 123.
- **Problem 2:** The slot car controlled via JSlot Project application doesn't move.
Solution on page 124.
- **Problem 3:** “*Interface PC-SlotCircuit Device*” (F) isn't working correctly.
Solution on page 125.

PROBLEM 1:

If when IC Capture software is running and you want carry out the workspace adjustment process, you cannot watch the camera filming in real time, you have to take next solutions:

1. Check that camera is connected via USB cable to PC
2. Ensure that camera diaphragm is not closed.
3. Make sure that the visual field of the camera is not blocked.

If after check the above tips the problem persists you can contact the manufacturer of the camera:

http://www.theimagingsource.com/en_US/company/contact/

PROBLEM 2:

If slot car controlled via JSlot Project application doesn't move, you have to take next solutions:

1. Check all connections.
2. Ensure that metal brushes, located on the bottom-front zone of the slot car, are plugging to the rails metal zone.
3. Make sure that your PC has installed a correct Java version. For this project is needed version 7 update 21 or higher. If version Java installed on your PC is above that this, you have to update it. You can see more information about that on page 110.

If you have not successfully removed the problem, is possible that "*Interface PC-SlotCircuit Device*" (F) has a physical damage. If this happens, you have to follow instructions on page 125 in order to test the device components and find the problem.

PROBLEM 3:

If “*Interface PC-SlotCircuit Device*” (F) is not working correctly, you have to make a set of tests in order to find the problem.

Note that, if when you execute any instruction listed below the tests results aren’t expected, you have to go to end of list in order to read the recommendations to solve this specifically problem.

Here you can see a list of actions that you have to do for device testing process:

1. Close JSlot Project application.
2. Disassemble “*Interface PC-SlotCircuit Device*” (F) removing screws located on back side.
3. Separate the mainboard from device housing. You have to do this process in the following order:
 - a. Connect USB cable to PC and then to the device.
 - b. Connect the first plug of Jack cable to device.
 - c. Connect the other plug of Jack cable to circuit.
 - d. Connect the circuit to grid power.
4. Run MATLAB software on your PC.
5. Make sure that your PC has recognized the device and a serial port number have been assigned.

For this purpose: right click on **Mi PC**, select **Administrar** and follow that go to **Administrador de dispositivo** (on the left side of the window). Finally select Puertos **COM & LPT** and get the COM port assigned.

6. Write the next set of instructions on MATLAB console to send data to the PIC of the device. In this way, you can know if PIC gets this data or not.

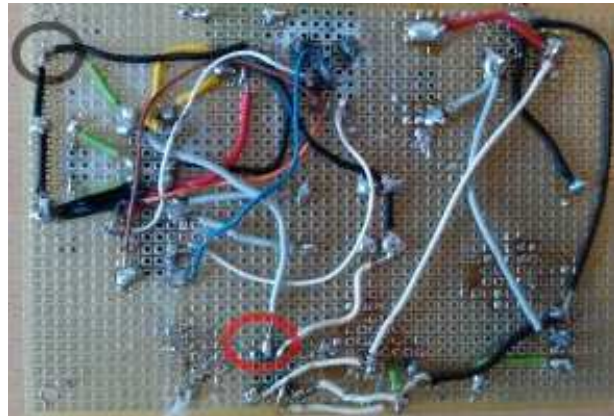
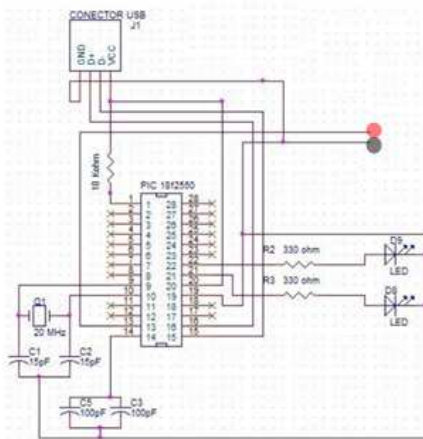
```
PS=serial('COMX');           % "COMX" is the serial port number  
                             assigned by PC.  
  
fopen(PS);  
  
fwrite(PS,data,'uint16');    % "data" has to have a value  
                             between 0-1023.
```

When the PIC read data via USB cable the green LED turn on at the moment.

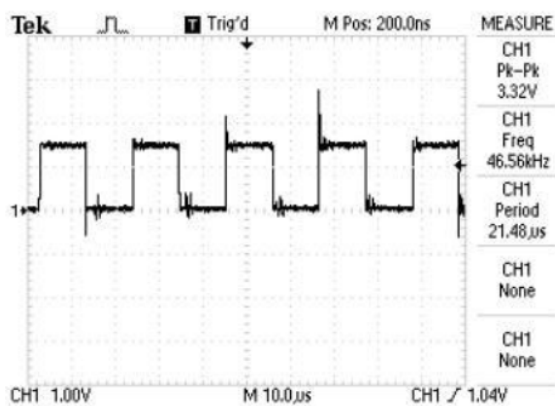
7. Send data 512 to PIC using console `fwrite(PS,512,'uint16');` on MATLAB console.

From now on, you have to keep in mind that in the pictures presented on steps number 8, 9 and 10, the point where you have to put the positive plug of oscilloscope is represented by red mark and, on the other hand, the point where you have to put the negative plug is represented by black mark (ground).

8. Check that PWM signal generated by PIC is correct. For this purpose you have to check voltage signal between PIN 13 and PIN 19.

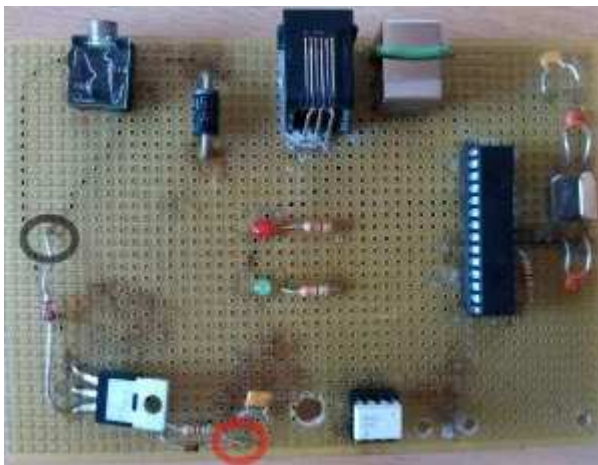
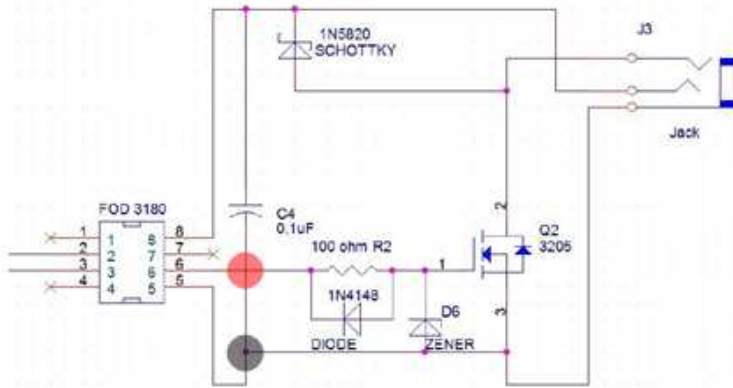


If the PWM signal generation task is working correctly, signal displayed on oscilloscope screen, when you send data 512 to PIC, has to look like following picture:

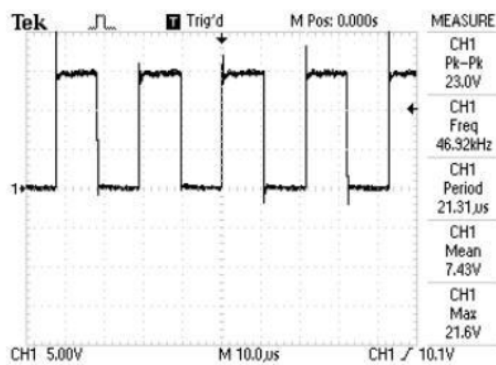


Salida PWM PIC duty cycle 512

9. Check that PWM signal generated by PIC is correctly amplified by optocoupler driver. For this purpose you have to check voltage signal between PIN 5 and PIN 6 of this optocoupler driver.



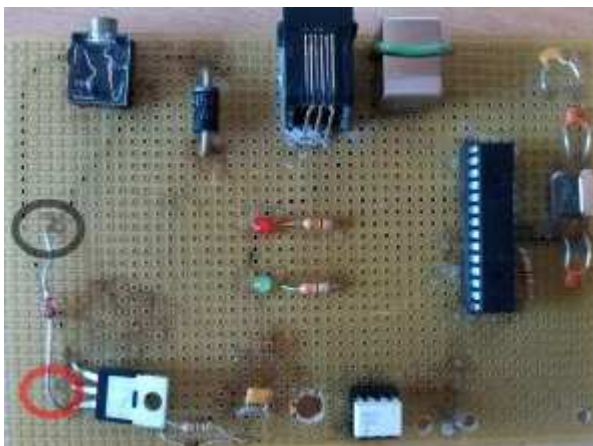
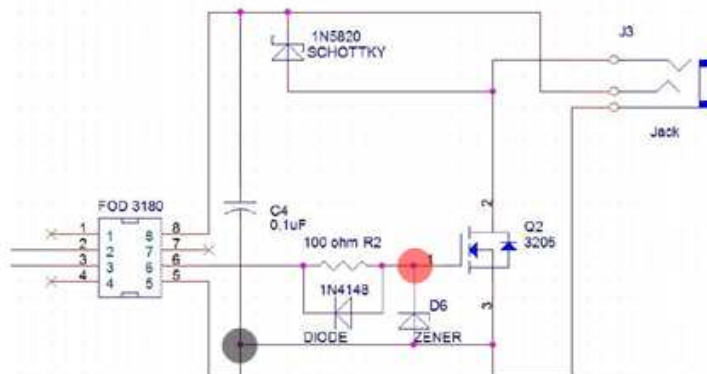
If PWM signal generated by PIC is correctly amplified by optocoupler driver, signal displayed on oscilloscope screen, when you send data 512 to PIC, has to look like following picture:



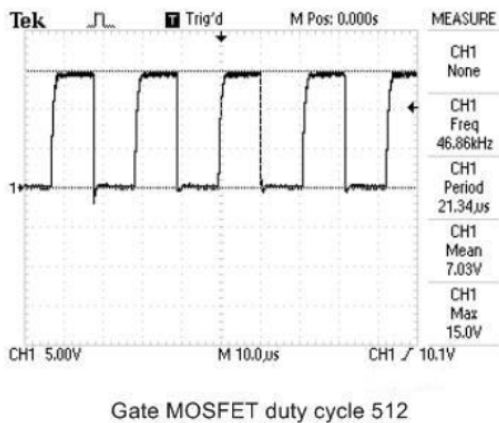
Salida OPTO-DRIVER duty cycle 512

NOTE: Is very important to note that PWM signal amplified by optocoupler driver, in case reflected in the picture above, has a voltage mean value 7.43 V. Given that voltage value of slot car varies between 0-14.8 V and that the send data was 512 (50 %), it is expected that the measured voltage take values around 7.4 V.

10. Make sure that Zener diode has reduced the voltage overshoot. For this purpose you have to check voltage signal between both sides of Zener diode.



If Zener diode is working correctly on the device, you can see that new signal has not overshoot voltage. When you send data 512 to PIC, signal displayed on oscilloscope screen has to look like following picture:



11. You can redo steps 8, 9 and 10 so many times as you see fit, changing the value sent to PIC. You can use relevant values such as 256 (25%), 768 (75 %) or 1023 (100 %).

12. Close and delete serial port created in MATLAB software. For that, you have to enter the follow lines code in the MATLAB console:

```
fclose(PS);
delete(PS);
```

13. Close MATLAB software.

14. Disconnect all cables to the device in the reverse order presented on step number 3.

Therefore, the order to disconnect peripheral hardware to device is:

- a. Disconnect the power grid.
- b. Disconnect the first plug of Jack cable to the circuit.
- c. Disconnect the other plug of Jack cable to the device.
- d. Disconnect USB cable to PC and then disconnect it to the device.

When you have finished the test presented above, you will have the knowledge to determine what is the problem of your device and what is the way to solve it.

- If when executing step number 4 the device hasn't recognized by PC, you have to disconnect USB cable and then connect it again. If the problem isn't solved, you have to install again the device driver. You can see a step by step to install this driver on page 106, titled "*Interface PC-SlotCircuit drivers (CDC Drivers)*".
- If when executing step number 8 the signal obtained on oscilloscope doesn't look like at picture presented on step 8, you can be sure that PIC is damaged. You have to check connections state (loose wires, broken welding, broken wire, and so on) and mend all damages that you can find. If the problem isn't solved, you have to buy a new PIC and change it.
- If when executing step number 9 the signal obtained on oscilloscope doesn't look like at picture presented on step 9, you can be sure that optocoupler driver is damaged. You have to check connections state (loose wires, broken welding, broken wire, and so on) and mend all damages that you can find. If the problem isn't solved, you have to buy a new optocoupler driver and change it.
- If when executing step number 10 the signal obtained on oscilloscope doesn't look like at picture presented on step 10, you can be sure that Zener diode is damaged. You have to check connections state (loose wires, broken welding, broken wire, and so on) and mend all damages that you can find. If the problem isn't solved, you have to buy a new Zener diode and change it.

WARNING: If you have to change any electronic component of your device is very important use the same type component used when the device was assembled. Thus, you can be sure there will be full compatibility between all components of the device. The most important electronic components used for assembled task of the device are: PIC (18f2550 made by Microchip), driver optocoupler (FOD3180 made by Fairchild



Semiconductor), Zener diode (1N5333B made by Multicomp) and MOSFET (IRF3205 made in International Rectifier).

10. CONCLUSIONS

Through the development of this project has been successfully integrated into a single Java application the control slot car and the computer vision. Moreover, has been included a data storage system as a database. This DB can be managed from the JSlot App. Furthermore, as if that were not enough, the calibration tasks, both workspace location and lighting calibration, can be carry out easily from JSlot App, in a single click for each.

Regarding to monitoring of the race car, the application developed in this project is as complete as effective. JSlot App is able to calculate interesting race facts such as: current speed, average speed, top speed, current position or the lap time. What's more, the user can watch the video of the race in real time.

On the other hand, regarding to the hardware part, the PC-Racetrack interface device has been modified. Some electronic components of this device have been replaced to another area in order to put all external connections in the same area. Furthermore has been placed within a housing to protect the electronic components from bumps and improve their appearance.

Moreover, as regards to computer vision part was necessary to design it again. This is so because the application developed on Ref: BFP.02 project was malfunctioning. Once the new system to acquire and process pictures was finished, was decided to make a comparative study using different values of ROI (Region Of Interest) size. From this analysis it was concluded that ROI size calculated theoretically, during final project referenced above, is too small. Consequently it was decided to change the ROI size.

For this project, also has been made a complete user guide, written both in Spanish and English, in order to facilitate the use of JSlot Project system. Moreover, the user guide includes a step by step about how to find and fix the breakdowns (hardware) or malfunctioning (software) that may appear in the future.



Through the development of this Bachelor's thesis the control and monitoring platform, started in previous projects, is ready for use. Even so, there are some areas which this platform can be improved. These improvements are listed in the following section: Future research trends.

11. FUTURE RESEARCH TRENDS

Once the project is completed, is very important keep in mind future research where it is possible work in order to improve their software and hardware features.

Following is shown below all recommended changes or improvements to do:

- Assemble the electronic components of the PC-Racetrack interface device on a PCB. All wires and welds are removed. In this way, the whole system will be safer because the likelihood that an electrical fault happens (short circuit, any wire disconnected ...) will be less than now. You can go to bachelor's thesis Ref: BFP.03 to find the PCB designed of the PC-Racetrack interface device.
- Include a tool for the computer vision software to achieve memorize the circuit before the race starts. This tool has to recognize the blends of the circuit (blend type, where are placed the start and the final of the blend ...) in order to determine the circuit zoning (ROI locations).
- Make the look-up table control using the circuit zoning data.
- Consider using a corrective control system. This algorithm has to use the tilts angle of the slot car (yaw) to determine when is possible increase the speed car or, instead, when is imperative reduce their speed.
- Include an improvement at JSlot Project App to control two or more slot cars using computer vision.
- Include the chance to control a slot car using the behaviour of the other car.
- Save in the database the set points (SP) generated during the race to control the slot car. In this way, will be possible repeat the same travel behaviour on the racetrack (ghost car mode). Thus, you can compete versus yourself as a driver of the past. This improvement is very useful to train for slot car racing.



- Use the race data stored on DB to make statistical studies.
- Change the current database type for a better way to store data (e.g. SQL).
- Install a new lighting system above the slot racetrack. Additionally, it would also be necessary to place a diffuser fabric. In this way the lighting calibration task is not necessary.

12. BIBLIOGRAPHY

BACHELOR'S THESIS:

Jiménez López, E. (2012). *Diseño e implementación de un sistema de control de un circuito de slot*. Bachelor's Thesis, UPC, EET, Terrassa. (Ref: BFP.03).

Osete Herraiz, Azucena; Peñalver Núñez, Andrés and Portell Blanch, Gabriel. (2008). *Diseño de una plataforma didáctica de monitorización y control de una carrera de slot*. Miniproyecto, UPC, ETSEIAT, Terrassa. (Ref: BFP.01).

Sánchez Rodríguez, D. (2010). *Disseny d'una plataforma didàctica de monitoratge i control d'un circuit de slot*. Bachelor's Thesis, UPC, EUETIT, Terrassa. (Ref: BFP.02).

SERIAL PORT:

Cantón Ferrero, J. (2010, February 02). *El despacho de los Jorges*. Retrieved from Conectarse con el puerto Serie/paralelo desde Java:

<http://eldespachodelosjorges.blogspot.com.es/2010/02/conectarse-con-el-puerto-serieparalelo.html>

Hell-Desk. (2011, December 26). Retrieved from Acceder al puerto serie programando en Java:

<http://www.hell-desk.com/programacion/acceder-al-puerto-serie-programando-en-java/>

Hell-Desk. (2012, January 31). Retrieved from Acceder al puerto serie programando en Java, 2ª parte:

<http://www.hell-desk.com/programacion/acceder-al-puerto-serie-programando-en-java-2a-parte/>

Rtx Project. (2007, November). Retrieved from

<http://www.rtx.org/>

Rtx wiki. (2011, July). Retrieved from

http://rtx.qbang.org/wiki/index.php/Main_Page

COMPUTER VISION:

Java: LTI-CIVIL, usando video desde cámara USB, adiós a JMF. (2010, July 03).

Retrieved from

<http://cmop17.wordpress.com/2010/07/03/java-lti-civil-usando-video-desde-camara-usb-adios-a-jmf/>

Learned-Miller, E. G. (2012, September 04). Retrieved from

http://people.cs.umass.edu/~elm/Teaching/Docs/IntroCV_9_4_12.pdf

LTI-CIVIL Powering video applications in Java. (2007, Oct.). Retrieved from

<http://lti-civil.org/>

Osama Oransa's Blog. (2010, December 18). Retrieved from

<http://osama-oransa.blogspot.com.es/2010/12/capture-photo-from-web-cam-usb-cam.html>

SUPPORTING DOCUMENTATION:

Java. (n.d.). Retrieved from

<http://www.java.com/en/>

Microchip. (1998-2013). Retrieved from

<http://www.microchip.com/>

Oracle. (1993-2013). *API Java.* Retrieved from

<http://docs.oracle.com/javase/7/docs/api/>

The Imaging Source. (1991-2013). Retrieved from

http://www.theimagingsource.com/en_US/



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

ANNEXES

STUDY OF AUTOMATIC CONTROL ALGORITHMS FOR A SLOT CAR

Author: Matías Nicolás Correa Barceló

Advisor: Ramon Sarrate Estruch



Bachelor's Degree in Industrial
Electronics and Automatic Control
Engineering

```
/**
 *
 * @author Matías Nicolás
 */

import com.lti.civil.*;
import com.lti.civil.awt.AWTImageConverter;
import com.lti.civil.utility.FPSCounter;
import com.sun.image.codec.jpeg.*;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.Calendar;
import java.util.List;
import javax.imageio.ImageIO;

public class CameraAccess implements CaptureObserver {

    CaptureStream captureStream = null;
    BufferedImage myImage=null;

    public CameraAccess() {

        CaptureSystemFactory factory = DefaultCaptureSystemFactorySingleton.instance();
        CaptureSystem system;
        try {
            system = factory.createCaptureSystem();
            system.init();
            List list = system.getCaptureDeviceInfoList();
            int i = 0;
            if (i < list.size()) {
                CaptureDeviceInfo info = (CaptureDeviceInfo) list.get(i);
                captureStream = system.openCaptureDeviceStream(info.getDeviceID());
                captureStream.setObserver(CameraAccess.this);
                captureStream.start(); //Conectarse a la cámara.
            }
        } catch (CaptureException ex) {
        }
    }

    public synchronized void onNewImage(CaptureStream stream, Image image) {

        try{
            myImage=AWTImageConverter.toBufferedImage(image);

        }

        catch (Exception e){
            e.printStackTrace();
            return;
        }

    }
}
```

CameraAccess.java

```
public void onError(CaptureStream stream, CaptureException ce) {
System.out.println("Error!");
}

public void disconnect(){
    try {
captureStream.stop();
} catch (CaptureException ex) {
ex.printStackTrace();
}
}

public synchronized BufferedImage myImage(){
    BufferedImage img=null;
    if (myImage!=null) {
        img=myImage;
        myImage=null;
    }

    return img;
}

}
```

SerialPort_Access.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Matías Nicolás
 */

import gnu.io.*;
import java.util.Enumeration;
import java.io.*;

public class SerialPort_Access {

    private OutputStream outputStream;
    private SerialPort puertoSerie;

    public SerialPort_Access(){

        System.loadLibrary("rxtxSerial"); //Cargar librería
        puertoSerie=null;

    }

    public void Connect(String COM) throws Error{

        //Búsqueda del puerto serie introducido en este método. (COM).
        Enumeration listaPuertos = CommPortIdentifier.getPortIdentifiers();
        CommPortIdentifier idPuerto = null;
        boolean encontrado = false;
        while (listaPuertos.hasMoreElements() && !encontrado) {
            idPuerto = (CommPortIdentifier) listaPuertos.nextElement();
            if (idPuerto.getPortType() == CommPortIdentifier.PORT_SERIAL) {
                if (idPuerto.getName().equals(COM)) {
                    encontrado = true;
                }
            }
        }

        try{
            puertoSerie = (SerialPort)idPuerto.open("vid_0461&pid_0033",2000 );
        }
        catch(PortInUseException e){
            System.out.println("Error abriendo el puerto serie");
        }

        try{
            outputStream = puertoSerie.getOutputStream(); //Abrir canal de salida de
datos.
        }
        catch(IOException e){
        }

    }

    public void write(int data){

        byte[] sendData=new byte[2];
        sendData[0]=(byte) (data & 0xFF);
    }
}
```

```

                                SerialPort_Access.java
sendData[1]=(byte) ((data >>8) & 0xFF);

if(!(outputStream==null)){
    try{
        outputStream.write(sendData,0,2);
        outputStream.flush();
    }
    catch(IOException e){
        System.out.println("Error al intentar escribir sobre el PIC");
    }
}

public void close() throws Error{
    puertoSerie.close();
}

}

```

CarPosition.java

```
/**
 *
 * @author Matías Nicolás
 */

import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;

public class CarPosition {

    int Location,WhitePixels,BlackPixels,ROI;
    int[]Pixel_X,Pixel_Y;
    long Timer;
    BufferedImage Reference_Picture;

    //41pi:
    //static final int[]
    Pixel_X={347,306,265,224,183,142,102,65,36,41,41,41,41,65,102,142,183,224,265,306,3
    47,388,423,463,496,517,535,539,535,517,496,463,423,385};
    //static final int[]
    Pixel_Y={278,278,278,278,278,278,272,254,226,189,150,111,74,46,41,41,41,41,41,41
    ,41,41,41,46,74,111,150,189,226,254,272,278,278};

    //45pi:

    //84pi:
    //static final int[]
    Pixel_X={422,338,254,170,86,44,44,70,131,215,299,383,467,551,595,595,555,506};
    //static final int[]
    Pixel_Y={320,320,320,320,295,236,152,84,44,44,44,44,44,68,135,219,285,320};

    //60pi:
    //static final int[]
    Pixel_X={413,353,293,233,173,113,64,35,35,54,100,156,216,276,336,396,456,516,567,60
    0,600,582,536,473};
    //static final int[]
    Pixel_Y={320,320,320,320,320,309,278,218,158,98,56,33,33,33,33,33,33,47,76,133,193,
    253,310,320};

    //Usando puntos singulares, ROI=60pi.

    public CarPosition(int ROIsize,int[] pixel_X,int[] pixel_Y){

        ROI=ROIsize;

        Pixel_X=new int[pixel_X.length];
        Pixel_Y=new int[pixel_Y.length];
        for(int i=0;i<pixel_X.length;i++){
            Pixel_X[i]=pixel_X[i];
            Pixel_Y[i]=pixel_Y[i];
        }

        Location=0;
        WhitePixels=0;
    }
}
```



```

BlackPixels=0;
Reference_Picture = new BufferedImage(640,480,BufferedImage.TYPE_INT_RGB) ;
}

public void calibration(BufferedImage picture){
    for(int i=0;i<640;i++){
        for(int j=0;j<480;j++){
            if(((picture.getRGB(i,j)&0xFF0000)>0xC80000)&&((picture.getRGB(i,
j)&0x0000FF)<0x80))){
                Reference_Picture.setRGB(i,j,Color.WHITE.getRGB());
            }
            else{
                Reference_Picture.setRGB(i,j,Color.BLACK.getRGB());
            }
        }
    }
}

public int[] check_OutBoundsPIXELS(){
    //El código que viene a continuación es para evitar que se intente
binarizar píxeles
//fuera del tamaño de la imagen 640x480

    int[] bounds=new int[4];//0=Xmin,1=Ymin,2=Xmax,3=Ymax
    if(Pixel_X[Location]>ROI)
    {
        bounds[0]=Pixel_X[Location]-ROI;
    }
    else {bounds[0]=10;}
    if(Pixel_Y[Location]>ROI)
    {
        bounds[1]=Pixel_Y[Location]-ROI;
    }
    else {bounds[1]=15;}

    if(Pixel_X[Location]<(640-ROI))
    {
        bounds[2]=Pixel_X[Location]+ROI;
    }
    else {bounds[2]=624;}
    if(Pixel_Y[Location]<(480-ROI))
    {
        bounds[3]=Pixel_Y[Location]+ROI;
    }
    else {bounds[3]=342;}

    return bounds;
}

public void newFrame(BufferedImage Original_Picture){

```

```

CarPosition.java
int[] Bounds=new int[4];
Bounds=check_OutBoundsPIXELS();
int Xmin=Bounds[0];
int Ymin=Bounds[1];
int Xmax=Bounds[2];
int Ymax=Bounds[3];

BufferedImage Edited_Picture = new
BufferedImage(640,480,BufferedImage.TYPE_INT_RGB) ;

for (int i = Xmin ; i <Xmax ; i ++ ) {
    for ( int j = Ymin; j <Ymax ; j ++ ) {

        if ( ((Original_Picture.getRGB(i,j)&0xFF0000) >0xC80000)&&
((Original_Picture.getRGB(i,j)&0x0000FF) <0x80)) {
            Edited_Picture.setRGB(i,j,Color.WHITE.getRGB());

            if (Reference_Picture.getRGB(i,j)==Color.WHITE.getRGB()){
                Edited_Picture.setRGB(i,j,Color.BLACK.getRGB());
                BlackPixels++;
            }

            else {
                WhitePixels++;
            }

        }
        else {
            Edited_Picture.setRGB(i,j,Color.BLACK.getRGB());
            BlackPixels++;
        }
    }
}

if (WhitePixels>40){//El coche realmente ocupa unos 400 px.
    Location++;
    Timer=System.currentTimeMillis();
}

if(Location>((Pixel_X.length)-1)){
    Location=0;
}

WhitePixels=0;
BlackPixels=0;
}
}

```

calculator.java

```
/**
 *
 * @author Matías Nicolás
 */

public class Calculator {

    long Timer0,Timer1,lastPosition,fastLap,lapTime;
    int[] LengthSections;
    double speed,averageSpeed,topSpeed;
    int laps,maxROIindex,ROI;
    boolean firstLap,isRegularROIdistribution;

    static final double pixelSize=2.6; //longitud expresada en milímetros
    (representa la medida de cada pixel).
    static final double lengthTrack=1531; //longitud de cada vuelta expresada en
    pixeles.

    public Calculator(int totalROIs,int ROIsize){

        maxROIindex=totalROIs-1;
        ROI=ROIsize;
        isRegularROIdistribution=true;
        lastPosition=maxROIindex;
        speed=0;
        averageSpeed=0;
        topSpeed=0;
        fastLap=0;
        lapTime=0;
        laps=0;
        firstLap=true;

    }

    public Calculator(int totalROIs,int ROIsize,int[] lengthSections){
        //Este constructor se utilizará únicamente en el caso de que no se realice
        una distribución
        //regular de las distintas ROIs a lo largo del circuito (es decir, que la
        distancia entre ellas
        //no sea constante). lengthSections[] deberá contener las distancias
        expresada en píxeles entre los distintos tramos.
        //lengthSections[0] deberá tener la distancia entre la ROI número 0 (la
        meta) y la anterior.
        //lengthSections[1] deberá tener la distancia entre la ROI número 1 y la
        anterior (la 0). Y así sucesivamente.

        maxROIindex=totalROIs-1;
        ROI=ROIsize;
        isRegularROIdistribution=false;
        lastPosition=maxROIindex;
        speed=0;
        averageSpeed=0;
        topSpeed=0;
        fastLap=0;
        lapTime=0;
        laps=0;
        firstLap=true;

        LengthSections=new int[lengthSections.length];
        for(int i=0;i<lengthSections.length;i++){
            LengthSections[i]=lengthSections[i];
        }

    }

}
```

```

Calculator.java
public void setRaceData(int position, long positionTimer){

    if(lastPosition!=position){
        if(firstLap){
            if(Timer0==0){
                Timer0=positionTimer;
            }
            else{
                setSpeed(position,positionTimer);
                if(position==maxROIindex){
                    firstLap=false;
                }
            }
        }
        else{
            setSpeed(position,positionTimer);
            if(position==0){
                laps++;
                setLapTime(positionTimer);
                setAverageSpeed();
            }
        }

        Timer1=positionTimer;
        lastPosition=position;
    }
}

public void setSpeed(int position,long positionTimer){
    if(isRegularROIDistribution){
        speed=(double) ((ROI*pixelSize)/(positionTimer-Timer1));
    }
    else{
        speed=(double)
((LengthSections[position]*pixelSize)/(positionTimer-Timer1));
    }
    speed=rounding(speed);
    if(speed>topSpeed){
        topSpeed=speed;
    }
}

public void setLapTime(long positionTimer){
    lapTime=positionTimer-Timer0;
    Timer0=positionTimer;
}

```

```

Calculator.java
    if(lapTime<fastLap || fastLap==0){
        fastLap=lapTime;
    }
}

public void setAverageSpeed(){
    averageSpeed=(double) ((averageSpeed*(laps-1)+(lengthTrack/lapTime))/laps);
    averageSpeed=rounding(averageSpeed);
}

public String TimetoString(long time){
    long ss=time/1000;
    long ms=time%1000;

    return ss+"s "+ms+"ms ";
}

public double rounding(double num){
    return Math rint(num*100)/100;
}

}

```

JSlot_Screen.java

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

import com.lti.civil.CaptureException;
import java.awt.Color;
import java.awt.Desktop;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileNameExtensionFilter;

/**
 *
 * @author Matías Nicolás
 */
public class JSlot_Screen extends javax.swing.JFrame {

    private CameraAccess camera;
    private SerialPort_Access port;
    private CarPosition carFinder;
    private Calculator raceDataFactory;
    private chronometer Timer;
    private int laps;
    private long tempStart;
    private boolean carIsRacing;

    public JSlot_Screen() {

        initComponents();
        screenAdjusting();

        port=new SerialPort_Access();

        //45 px distribución regular
        //int[]
        Pixel_X={405,360,315,270,225,180,135,96,64,40,30,40,55,78,109,149,194,239,284,329,374,419,464,509,542,577,597,603,597,577,540,495,450};
        //int[]
        Pixel_Y={320,320,320,320,320,320,313,296,270,233,188,143,109,78,50,34,34,34,34,34,34,34,34,46,69,95,135,180,225,270,304,313,320};

        //60 px discretos:
        int[] Pixel_X={413,128,108,474,514};
        int[] Pixel_Y={320,312,56,41,313};

        //60px distribución regular:
        //int[]
        Pixel_X={413,353,293,233,173,113,64,35,35,54,100,156,216,276,336,396,456,516,567,600,600,582,536,473};
        //int[]
    }
}

```

```
Pixel_Y={320,320,320,320,320,309,278,218,158,98,56,33,33,33,33,33,33,47,76,133,193,253,310,320};
```

```
//84 px continuo:
//final int[]
```

```
Pixel_X={422,338,254,170,86,44,44,70,131,215,299,383,467,551,595,595,555,506};
//final int[]
```

```
Pixel_Y={320,320,320,320,295,236,152,84,44,44,44,44,68,135,219,285,320};
carFinder=new CarPosition(45,Pixel_X,Pixel_Y);
```

```
//60 px discretos:
int[] lengthSections={102,290,363,378,397};
raceDataFactory=new Calculator(Pixel_X.length,45,lengthSections);
```

```
Timer=new chronometer();
```

```
tempStart=0; //Esta variable guardará el momento en que se pulsa el boton
START. cuando pasen 5 segundos comenzará la carrera.
```

```
laps=0;
carIsRacing=false;
```

```
}
```

```
/**
```

```
* This method is called from within the constructor to initialize the form.
* WARNING: Do NOT modify this code. The content of this method is always
* regenerated by the Form Editor.
*/
```

```
@SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate="collapsed" desc="Generated
Code"> //GEN-BEGIN: initComponents
```

```
private void initComponents() {
```

```
    jMenu1 = new javax.swing.JMenu();
    jLabel10 = new javax.swing.JLabel();
    jLabel11 = new javax.swing.JLabel();
    jPanel4 = new javax.swing.JPanel();
    jPanel1 = new javax.swing.JPanel();
    jPanel2 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    labelTopSpeed = new javax.swing.JTextField();
    labelSpeed = new javax.swing.JTextField();
    labelAverageSpeed = new javax.swing.JTextField();
    labelFastLap = new javax.swing.JTextField();
    labelTimekeeper = new javax.swing.JTextField();
    jSeparator1 = new javax.swing.JSeparator();
    jLabel6 = new javax.swing.JLabel();
    jSeparator2 = new javax.swing.JSeparator();
    progressBarGas = new javax.swing.JProgressBar();
    jLabel7 = new javax.swing.JLabel();
    jLabel14 = new javax.swing.JLabel();
    labelLapsToGo = new javax.swing.JTextField();
    jLabel12 = new javax.swing.JLabel();
    jPanel3 = new javax.swing.JPanel();
    comboBoxPort = new javax.swing.JComboBox();
    jLabel8 = new javax.swing.JLabel();
    buttonConnect = new javax.swing.JButton();
    buttonDisconnect = new javax.swing.JButton();
    labelConnection = new javax.swing.JTextField();
    jLabel15 = new javax.swing.JLabel();
    textFieldLaps = new javax.swing.JTextField();
    jLabel13 = new javax.swing.JLabel();
    jLabel9 = new javax.swing.JLabel();
    buttonStart = new javax.swing.JButton();
```

```

JSlot_Screen.java
ButtonFinish = new javax.swing.JButton();
jMenuBar1 = new javax.swing.JMenuBar();
jMenu2 = new javax.swing.JMenu();
jMenuItemNew = new javax.swing.JMenuItem();
jMenuItemSave = new javax.swing.JMenuItem();
jMenuItemOpen = new javax.swing.JMenuItem();
jMenu3 = new javax.swing.JMenu();
jMenuItemCircuit = new javax.swing.JMenuItem();
jMenuItemLighting = new javax.swing.JMenuItem();
jMenuItemClear = new javax.swing.JMenuItem();
Help = new javax.swing.JMenu();
jMenuItemUserGuide = new javax.swing.JMenuItem();
JItemManual = new javax.swing.JMenuItem();

jMenu1.setText("jMenu1");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("JSLOT Project -FINAL PROJECT 2012-2013- By Matías Correa");
setBackground(new java.awt.Color(51, 255, 255));
setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
setResizable(false);

jPanel4.setBackground(new java.awt.Color(227, 231, 234));

jPanel2.setBackground(new java.awt.Color(0, 51, 102));
jPanel2.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

jLabel1.setForeground(new java.awt.Color(255, 255, 255));
jLabel1.setText("Top Speed (m/s):");

jLabel2.setForeground(new java.awt.Color(255, 255, 255));
jLabel2.setText("Speed (m/s):");

jLabel3.setForeground(new java.awt.Color(255, 255, 255));
jLabel3.setText("Average Speed (m/s):");

jLabel4.setForeground(new java.awt.Color(255, 255, 255));
jLabel4.setText("Fast Lap:");

jLabel5.setForeground(new java.awt.Color(255, 255, 255));
jLabel5.setText("Timekeeper (hh:mm:ss):");

LabelTopSpeed.setEditable(false);
LabelTopSpeed.setBackground(new java.awt.Color(0, 51, 102));
LabelTopSpeed.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
LabelTopSpeed.setForeground(new java.awt.Color(255, 255, 255));

LabelSpeed.setEditable(false);
LabelSpeed.setBackground(new java.awt.Color(0, 51, 102));
LabelSpeed.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
LabelSpeed.setForeground(new java.awt.Color(255, 255, 255));

LabelAverageSpeed.setEditable(false);
LabelAverageSpeed.setBackground(new java.awt.Color(0, 51, 102));
LabelAverageSpeed.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
LabelAverageSpeed.setForeground(new java.awt.Color(255, 255, 255));

LabelFastLap.setEditable(false);
LabelFastLap.setBackground(new java.awt.Color(0, 51, 102));
LabelFastLap.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
LabelFastLap.setForeground(new java.awt.Color(255, 255, 255));

LabelTimekeeper.setEditable(false);
LabelTimekeeper.setBackground(new java.awt.Color(0, 51, 102));
LabelTimekeeper.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
LabelTimekeeper.setForeground(new java.awt.Color(255, 255, 255));

jSeparator1.setForeground(new java.awt.Color(204, 204, 0));

```


JSlot_Screen.java

```

jLabel6.setFont(new java.awt.Font("Segoe Media Center Semibold", 0, 18));
// NOI18N
jLabel6.setForeground(new java.awt.Color(226, 226, 167));
jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel6.setText("Slot Car Data:");

jSeparator2.setForeground(new java.awt.Color(204, 204, 0));

ProgressBarGas.setMaximum(1023);

jLabel7.setForeground(new java.awt.Color(255, 255, 255));
jLabel7.setText("GAS");

jLabel14.setForeground(new java.awt.Color(255, 255, 255));
jLabel14.setText("Laps to go: ");

LabelLapsToGo.setEditable(false);
LabelLapsToGo.setBackground(new java.awt.Color(0, 51, 102));
LabelLapsToGo.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
LabelLapsToGo.setForeground(new java.awt.Color(255, 255, 255));

javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jSeparator1)
    .addComponent(jSeparator2)
    .addGroup(jPanel2Layout.createSequentialGroup()
        .addComponent(jLabel6,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jLabel11)
            .addComponent(jLabel12)
            .addComponent(jLabel13)
            .addComponent(jLabel14)
            .addComponent(jLabel15)
            .addComponent(jLabel14))
            .addGap(40, 40, 40)

        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addComponent(LabelLapsToGo)
            .addComponent(LabelTopSpeed)
            .addComponent(LabelSpeed)
            .addComponent(LabelAverageSpeed)
            .addComponent(LabelFastLap)
            .addComponent(LabelTimekeeper,
javax.swing.GroupLayout.DEFAULT_SIZE, 90, Short.MAX_VALUE))
            .addGap(0, 0, Short.MAX_VALUE))
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jLabel7)
            .addGap(110, 110, 110))
        .addComponent(ProgressBarGas,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,

```

```

Short.MAX_VALUE))
        .addContainerGap())
    );
    jPanel2Layout.setVerticalGroup(
jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
            .addGap(9, 9, 9)
            .addComponent(jLabel6)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE,
10, javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel11)
            .addComponent(LabelTopSpeed,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel12)
            .addComponent(LabelSpeed,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel13)
            .addComponent(LabelAverageSpeed,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel14)
            .addComponent(LabelFastLap,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel15)
            .addComponent(LabelTimekeeper,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel14)
            .addComponent(LabelLapsToGo,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE,
5, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(jLabel17)

```

```

                                JSlot_Screen.java
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(ProgressBarGas,
javax.swing.GroupLayout.PREFERRED_SIZE, 19, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addContainerGap(30, Short.MAX_VALUE)
);

    javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 272, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
        .addGap(0, 0, Short.MAX_VALUE)
        .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
    );
    jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 447, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addContainerGap()))
    );

    jPanel3.setBackground(new java.awt.Color(0, 51, 102));
    jPanel3.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

    ComboBoxPort.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
"COM1", "COM2", "COM3", "COM4", "COM5", "COM6", "COM7", "COM8", "COM9", "COM10",
"COM11", "COM12", "COM13", "COM14", "COM15", " " }));
    ComboBoxPort.setSelectedIndex(2);

    jLabel8.setForeground(new java.awt.Color(255, 255, 255));
    jLabel8.setText("PORT:");

    ButtonConnect.setText("Connect");
    ButtonConnect.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            ButtonConnectActionPerformed(evt);
        }
    });

    ButtonDisconnect.setText("Disconnect");
    ButtonDisconnect.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            ButtonDisconnectActionPerformed(evt);
        }
    });

    LabelConnection.setEditable(false);
    LabelConnection.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
    LabelConnection.setForeground(new java.awt.Color(255, 51, 0));
    LabelConnection.setHorizontalAlignment(javax.swing.JTextField.CENTER);
    LabelConnection.setText("OFF");

    jLabel15.setForeground(new java.awt.Color(255, 255, 255));
    jLabel15.setText("LAPS:");

```



```

        JSlot_Screen.java
        .addComponent(ComboBoxPort,
javafx.swing.GroupLayout.PREFERRED_SIZE, javafx.swing.GroupLayout.DEFAULT_SIZE,
javafx.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(javafx.swing.GroupLayout.Alignment.TRAILING,
jPanel3Layout.createSequentialGroup())
        .addGap(2, 2, 2)
        .addComponent(TextFieldLaps,
javafx.swing.GroupLayout.PREFERRED_SIZE, javafx.swing.GroupLayout.DEFAULT_SIZE,
javafx.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javafx.swing.LayoutStyle.ComponentPlacement.RELATED, 2,
javafx.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap(javafx.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    );

    ButtonStart.setText("START");
    ButtonStart.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            ButtonStartActionPerformed(evt);
        }
    });

    ButtonFinish.setText("FINISH");
    ButtonFinish.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            ButtonFinishActionPerformed(evt);
        }
    });

    javafx.swing.GroupLayout jPanel4Layout = new
javafx.swing.GroupLayout(jPanel4);
    jPanel4.setLayout(jPanel4Layout);
    jPanel4Layout.setHorizontalGroup(

jPanel4Layout.createParallelGroup(javafx.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel4Layout.createSequentialGroup())
        .addContainerGap()

.addGroup(jPanel4Layout.createParallelGroup(javafx.swing.GroupLayout.Alignment.TRAILING)
        .addGroup(jPanel4Layout.createSequentialGroup())
        .addGap(14, 14, 14)
        .addComponent(jLabel12,
javafx.swing.GroupLayout.PREFERRED_SIZE, 100,
javafx.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javafx.swing.LayoutStyle.ComponentPlacement.RELATED, 43,
Short.MAX_VALUE)
        .addComponent(jLabel13,
javafx.swing.GroupLayout.PREFERRED_SIZE, 468,
javafx.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(jPanel4Layout.createSequentialGroup())

.addGroup(jPanel4Layout.createParallelGroup(javafx.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(jPanel3,
javafx.swing.GroupLayout.Alignment.LEADING, javafx.swing.GroupLayout.DEFAULT_SIZE,
javafx.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(javafx.swing.GroupLayout.Alignment.LEADING,
jPanel4Layout.createSequentialGroup())
        .addGap(26, 26, 26)

.addGroup(jPanel4Layout.createParallelGroup(javafx.swing.GroupLayout.Alignment.LEADING, false)
        .addComponent(jLabel9,
javafx.swing.GroupLayout.DEFAULT_SIZE, javafx.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addGroup(jPanel4Layout.createSequentialGroup())

```



```

        JSlot_Screen.java
        .addComponent(ButtonStart,
javax.swing.GroupLayout.PREFERRED_SIZE, 139,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(ButtonFinish,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addGap(0, 0, Short.MAX_VALUE)))
        .addGap(6, 6, 6)
        .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap())
    );
    jPanel4Layout.setVerticalGroup(
jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel4Layout.createSequentialGroup()
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(jLabel13, javax.swing.GroupLayout.DEFAULT_SIZE,
95, Short.MAX_VALUE)
                .addComponent(jLabel12, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addGroup(jPanel4Layout.createSequentialGroup()
                    .addComponent(jPanel3,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(39, 39, 39)
                    .addComponent(jLabel9,
javax.swing.GroupLayout.PREFERRED_SIZE, 237,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(28, 28, 28)
                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(ButtonFinish,
javax.swing.GroupLayout.PREFERRED_SIZE, 41, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(ButtonStart,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGroup(jPanel4Layout.createSequentialGroup()
                    .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addContainerGap()))))
    );
    jMenu2.setText("File");

jMenuItemNew.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_N, java.awt.event.InputEvent.CTRL_MASK));
jMenuItemNew.setText("New DB");
jMenuItemNew.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItemNewActionPerformed(evt);
    }
});
jMenu2.add(jMenuItemNew);

```

JSlot_Screen.java

```

jMenuItemSave.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_S, java.awt.event.InputEvent.CTRL_MASK));
jMenuItemSave.setText("Save DB");
jMenuItemSave.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItemSaveActionPerformed(evt);
    }
});
jMenu2.add(jMenuItemSave);

jMenuItemOpen.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_O, java.awt.event.InputEvent.CTRL_MASK));
jMenuItemOpen.setText("Open DB");
jMenuItemOpen.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItemOpenActionPerformed(evt);
    }
});
jMenu2.add(jMenuItemOpen);

jMenuBar1.add(jMenu2);
jMenu3.setText("Edit");

jMenuItemCircuit.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_C, java.awt.event.InputEvent.CTRL_MASK));
jMenuItemCircuit.setText("...Circuit location");
jMenuItemCircuit.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItemCircuitActionPerformed(evt);
    }
});
jMenu3.add(jMenuItemCircuit);

jMenuItemLighting.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_L, java.awt.event.InputEvent.CTRL_MASK));
jMenuItemLighting.setText("... Lighting");
jMenuItemLighting.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItemLightingActionPerformed(evt);
    }
});
jMenu3.add(jMenuItemLighting);

jMenuItemClear.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_C, java.awt.event.InputEvent.SHIFT_MASK | java.awt.event.InputEvent.CTRL_MASK));
jMenuItemClear.setText("Clear fields");
jMenuItemClear.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItemClearActionPerformed(evt);
    }
});
jMenu3.add(jMenuItemClear);

jMenuBar1.add(jMenu3);
Help.setText("Help");

jMenuItemUserGuide.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_U, java.awt.event.InputEvent.CTRL_MASK));
jMenuItemUserGuide.setText("User Guide");
jMenuItemUserGuide.addActionListener(new java.awt.event.ActionListener() {

```

```

        JSlot_Screen.java
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenuItemUserGuideActionPerformed(evt);
        }
    });
    Help.add(jMenuItemUserGuide);

```

```

JMenuItemManual.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_M, java.awt.event.InputEvent.CTRL_MASK));
JMenuItemManual.setText("Manual de instrucciones");
JMenuItemManual.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItemManualActionPerformed(evt);
    }
});
Help.add(JMenuItemManual);

jMenuBar1.add(Help);

setJMenuBar(jMenuBar1);

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED_SIZE,
100, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(20, 20, 20)
            .addComponent(jLabel11, javax.swing.GroupLayout.DEFAULT_SIZE, 505,
Short.MAX_VALUE)
            .addGap(20, 20, 20)
            .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(20, 20, 20)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel10, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel11, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGap(20, 20, 20)
        );
pack();
} // </editor-fold> // GEN-END: initComponents

```

```

private void screenAdjusting(){
    setLocationRelativeTo(null); // Sirve stop centrar la ventana.
    jLabel12.setIcon(new ImageIcon("Pictures/UPC.png"));
    jLabel13.setIcon(new ImageIcon("Pictures/JSlotProject.png"));
    jLabel19.setIcon(new ImageIcon("Pictures/Busy.png"));
    ProgressBarGas.setMinimum(0);
    ProgressBarGas.setMaximum(1023);
    pack();
    setVisible(true);
}

```



```

}

private void clearFields(){
    LabelSpeed.setText("");
    LabelAverageSpeed.setText("");
    LabelTopSpeed.setText("");
    LabelTimekeeper.setText("");
    LabelFastLap.setText("");
    LabelAverageSpeed.setText("");
    LabelLapsToGo.setText("");
}

public String toString(){
    DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    Date date = new Date();
    String Date=dateFormat.format(date)+" --> ";
    String laps="Laps: "+TextFieldLaps.getText()+" ";
    String topSpeed="Top speed: "+LabelTopSpeed.getText()+" m/s; ";
    String averageSpeed="Average speed: "+LabelAverageSpeed.getText()+" m/s;
";
    String fastLap="Fast lap: "+LabelFastLap.getText();
    return Date+laps+topSpeed+averageSpeed+fastLap;
}

private void ButtonFinishActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ButtonFinishActionPerformed
    // TODO add your handling code here:

    port.write(0);
    carIsRacing=false;
    camera.disconnect();
    jLabel9.setIcon(new ImageIcon("Pictures/Busy.png"));
    LabelSpeed.setText("");
    LabelLapsToGo.setText("0");
    Timer.stop();

}

private void ButtonStartActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ButtonStartActionPerformed

    try{
        laps=Integer.parseInt(TextFieldLaps.getText());
    }
    catch(Exception e){
    }

    if(laps>0){//Se pone esta condición para evitar que comience una carrera de
0 vueltas..

        if(!LabelConnection.getText().equals("OFF")){

            clearFields();
            jLabel9.setIcon(new
ImageIcon("Pictures/TrafficLightAnimation.gif"));
            tempStart=System.currentTimeMillis();
            Timer.reset();
            if(camera==null){
                camera=new CameraAccess();

```

```

        JSlot_Screen.java
    }
    carIsRacing=true;

}

else{
    JOptionPane.showMessageDialog(null,"JSlot is offline. Please,
select a COM port and connect to serial port before start the
race","ERROR",JOptionPane.ERROR_MESSAGE);
}

}

else{
    JOptionPane.showMessageDialog(null,"LAPS field must be an integer value
greater than zero. Please, insert laps race value again.
","ERROR",JOptionPane.ERROR_MESSAGE);
}

}

}

private void ButtonConnectActionPerformed(java.awt.event.ActionEvent evt)
{
//GEN-FIRST:event_ButtonConnectActionPerformed

// TODO add your handling code here:

if(LabelConnection.getText().equals("OFF")){
    try {
        port.Connect((String)ComboBoxPort.getSelectedItem());
        LabelConnection.setText("ON");
        LabelConnection.setForeground(Color.green);
    } catch (Error e) {
        JOptionPane.showMessageDialog(null,"Serial port connection failed.
Please, check that selected COM port is correct and try again
later.", "ERROR",JOptionPane.ERROR_MESSAGE);
    }
}

else{
    JOptionPane.showMessageDialog(null,"JSlot is connected to a serial
port. If you want to change the selected COM port you have to click Disconnect
previously", "ERROR",JOptionPane.ERROR_MESSAGE);
}
}

private void ButtonDisconnectActionPerformed(java.awt.event.ActionEvent evt)
{
//GEN-FIRST:event_ButtonDisconnectActionPerformed

// TODO add your handling code here:

if(!LabelConnection.getText().equals("OFF")){
    try{
        port.close();
        LabelConnection.setText("OFF");
        LabelConnection.setForeground(Color.red);
    }
    catch(Error e){
        JOptionPane.showMessageDialog(null,"An error has occurred during
disconnect the serial port. Please, unplug the USB cable and reset JSlot
App.", "ERROR",JOptionPane.ERROR_MESSAGE);
    }
}

else{

```



```

                                JSlot_Screen.java
{//GEN-FIRST:event_jMenuItemNewActionPerformed

    // TODO add your handling code here:
    try {
        File f=new File("DB");
        int numFiles=f.list().length;
        FileWriter fw=new FileWriter("DB/DB"+numFiles+".txt",false);
        BufferedWriter out=new BufferedWriter(fw);
        out.write("");
        out.close();

    } catch (IOException e) {
        e.printStackTrace();
    }

}//GEN-LAST:event_jMenuItemNewActionPerformed

private void jMenuItemSaveActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jMenuItemSaveActionPerformed
    // TODO add your handling code here:

    try {
        File f=new File("DB");
        int numFiles=f.list().length;
        int pos=0;
        if(numFiles!=0){
            pos=numFiles-1;
        }
        FileWriter fw=new FileWriter("DB/DB"+pos+".txt",true);//Guardamos
siempre en el último fichero en uso
        BufferedWriter out=new BufferedWriter(fw);
        out.write(toString());
        out.newLine();
        out.newLine();
        out.close();

    } catch (IOException e) {
        e.printStackTrace();
    }

}//GEN-LAST:event_jMenuItemSaveActionPerformed

private void jMenuItemOpenActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jMenuItemOpenActionPerformed

    // TODO add your handling code here:

    JFileChooser fileChooser=new JFileChooser("DB");
    fileChooser.setFileFilter(new FileNameExtensionFilter("TXT Files","txt"));
    JFrame f=new JFrame();
    f.add(fileChooser);
    setLocationRelativeTo(null);
    f.setSize(300,200);

    f.setVisible(true);

    int status=fileChooser.showOpenDialog(null);
    if(status==JFileChooser.APPROVE_OPTION){
        try {
            File selectedFile=fileChooser.getSelectedFile();
            Desktop.getDesktop().open(selectedFile);

        } catch (IOException ex) {
            Logger.getLogger(JSlot_Screen.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
}

```

```

} //GEN-LAST:event_jMenuItemOpenActionPerformed

private void jMenuItemCircuitActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_jMenuItemCircuitActionPerformed

    // TODO add your handling code here:

    try {
        Desktop.getDesktop().open(new File("IC capture
access/workspace_calibration.iccf"));
    } catch (IOException ex) {

        Object[] buttonText={"OK"};
        JOptionPane.showOptionDialog(null,"IC Capture is not available. Please,
solve the problem and try
again. ","WARNING",JOptionPane.DEFAULT_OPTION,JOptionPane.INFORMATION_MESSAGE,null,b
uttonText,buttonText[0]);

    }

} //GEN-LAST:event_jMenuItemCircuitActionPerformed

private void jMenuUserGuideActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_jMenuUserGuideActionPerformed

    // TODO add your handling code here:

    try {
        Desktop.getDesktop().open(new File("User Guide/UserGuide.pdf"));
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(null,"UserGuide.pdf file was not found in
the User Guide folder","ERROR",JOptionPane.ERROR_MESSAGE);
    }

} //GEN-LAST:event_jMenuUserGuideActionPerformed

private void JItemManualActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_JItemManualActionPerformed
    // TODO add your handling code here:

    try {
        Desktop.getDesktop().open(new File("User Guide/Manual de
instrucciones.pdf"));
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(null,"Manual de instrucciones.pdf file
was not found in the User Guide folder","ERROR",JOptionPane.ERROR_MESSAGE);
    }

} //GEN-LAST:event_JItemManualActionPerformed

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {

```

```

        JSlot_Screen.java
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(JSlot_Screen.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(JSlot_Screen.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(JSlot_Screen.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(JSlot_Screen.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

```

```

JSlot_Screen AppSlot=new JSlot_Screen();
BufferedImage img;

```

```

while(!AppSlot.carIsRacing);

```

```

do{
    img=AppSlot.camera.myImage();
}
while(img==null);

```

```

AppSlot.carFinder.calibration(img);

```

```

while((System.currentTimeMillis()-AppSlot.tempstart)<5000 ||
AppSlot.tempstart==0);
//El bucle de arriba es stop esperar hasta que transcurran los 5
segundos necesarios stop
//empezar la carrera (cuando el semáforo se pone en verde).

```

```

AppSlot.Timer.start(); //Ponemos el cronómetro en marcha
int SP=0;

```

```

while(AppSlot.carIsRacing){

```

```

AppSlot.LabelTimekeeper.setText(AppSlot.Timer.elapsedTime_string()); //Actualizamos
etiqueta cronómetro

```

```

do {
    img=AppSlot.camera.myImage();//Obtener imagen.
}
while (img==null);

```

```

AppSlot.carFinder.newFrame(img);//pasarla por CarPosition.

```

```

AppSlot.raceDataFactory.setRaceData(AppSlot.carFinder.Location,AppSlot.carFinder.Timer);
//pasarla por Calculator.

```

```

switch(AppSlot.carFinder.Location){
    case 0: SP=570;
            break;

    case 1: SP=500;
            break;

    case 2: SP=570;
            break;

    case 3: SP=500;
            break;

    case 4: SP=500;
            break;
}

    if(SP!=AppSlot.ProgressBarGas.getValue()){
        AppSlot.port.write(SP);//Ejectutar write() de
SerialPort_Access (stop dar consigna).
        AppSlot.ProgressBarGas.setValue(SP); //Se deberá poner
dentro de setValue el entero que se envió al PIC. (LA CONSIGNA).
    }

        AppSlot.jLabel9.setIcon(new
ImageIcon(img.getSubimage(10,15,614,327).getScaledInstance(295,157,BufferedImage.SC
ALE_AREA_AVERAGING))); //enviamos imagen de la carrera en tiempo real a la
pantalla.

AppSlot.LabelTopSpeed.setText(String.valueOf(AppSlot.raceDataFactory.topSpeed));
AppSlot.LabelAverageSpeed.setText(String.valueOf(AppSlot.raceDataFactory.averageSpe
ed));

AppSlot.LabelFastLap.setText(String.valueOf(AppSlot.raceDataFactory.TimetoString(App
Slot.raceDataFactory.fastLap)));
AppSlot.LabelLapsToGo.setText(String.valueOf((AppSlot.laps-AppSlot.raceDataFactory.
laps)));
AppSlot.LabelSpeed.setText(String.valueOf(AppSlot.raceDataFactory.speed));

        if((Integer.valueOf(AppSlot.LabelLapsToGo.getText())==0)&&
(AppSlot.carFinder.Location==1)){
            AppSlot.carIsRacing=false;
        }

    }

    AppSlot.jLabel9.setIcon(new ImageIcon("Pictures/Busy.png"));
    AppSlot.port.write(0);
}

// variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton ButtonConnect;
private javax.swing.JButton ButtonDisconnect;

```

```

JSlot_Screen.java
private javax.swing.JButton ButtonFinish;
private javax.swing.JButton ButtonStart;
private javax.swing.JComboBox ComboBoxPort;
private javax.swing.JMenu Help;
private javax.swing.JMenuItem JMenuItemManual;
private javax.swing.JTextField LabelAverageSpeed;
private javax.swing.JTextField LabelConnection;
private javax.swing.JTextField LabelFastLap;
private javax.swing.JTextField LabelLapsToGo;
private javax.swing.JTextField LabelSpeed;
private javax.swing.JTextField LabelTimekeeper;
private javax.swing.JTextField LabelTopSpeed;
private javax.swing.JProgressBar ProgressBarGas;
private javax.swing.JTextField TextFieldLaps;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JMenu jMenu1;
private javax.swing.JMenu jMenu2;
private javax.swing.JMenu jMenu3;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JMenuItem jMenuItemCircuit;
private javax.swing.JMenuItem jMenuItemClear;
private javax.swing.JMenuItem jMenuItemLighting;
private javax.swing.JMenuItem jMenuItemNew;
private javax.swing.JMenuItem jMenuItemOpen;
private javax.swing.JMenuItem jMenuItemSave;
private javax.swing.JMenuItem jMenuItemUserGuide;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
// End of variables declaration//GEN-END:variables
}

```



```
import java.util.Date;

class chronometer
{

    private long t_start;
    private long elapsed;
    private boolean on;

    public void chronometer ()
    {

        t_start=0;
        elapsed=0;
        on=false;

    }

    public void start()
    {
        if(!on)
        {
            Date inici=new Date();
            t_start=inici.getTime();
            on=true;
        }
    }

    public void stop()
    {
        long t_stop=0;

        if(on)
        {
            Date parat=new Date();
            t_stop=parat.getTime();
            elapsed=elapsed+(t_stop-t_start);
            on=false;
        }
    }

    public void reset ()
    {
        t_start=0;
        elapsed=0;
        if(on)
        {
            Date inici=new Date();
            t_start=inici.getTime();
            on=true;
        }
    }

    public long elapsedTime ()
    {

        long elapsed_time=0;
```

chronometer.java

```
if (on)
{
    Date d=new Date();
    elapsed_time=(d.getTime()-t_start)+elapsed;
}

if(!on)
{
    elapsed_time=elapsed;
}

return elapsed_time;
}

public String elapsedTime_string (){

    long temps=elapsedTime()/1000;
    long hh=temps/3600;
    long mm=(temps%3600)/60;
    long ss=(temps%3600)%60;

    return hh+":"+mm+":"+ss;

}

}
```

Lighting_Adjustment.java

```
import com.sun.image.codec.jpeg.JPEGCodec;
import com.sun.image.codec.jpeg.JPEGImageEncoder;
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileOutputStream;
import javax.imageio.ImageIO;

/**
 *
 * @author Matías Nicolás
 */
public class Lighting_Adjustment {

    private int pictureCounter;

    public Lighting_Adjustment(){
        super();
        pictureCounter=0;
    }

    public void newFrame(BufferedImage picture){
        for(int i=0;i<640;i++){
            for(int j=0;j<480;j++){
                if(((picture.getRGB(i,j)&0xFF0000)>0xC80000)&&((picture.getRGB(i,
j)&0x0000FF)<0x80))){
                    picture.setRGB(i,j,Color.WHITE.getRGB());
                }
                else{
                    picture.setRGB(i,j,Color.BLACK.getRGB());
                }
            }
        }
        save(picture);
    }

    public void save(BufferedImage picture){
        try{
            File file=new File("Lighting
adjustment/picture"+pictureCounter+".jpg");
            ImageIO.write(picture,"jpg",file);
            pictureCounter++;
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }
}
```