Systems and Automation Engineering Department

BACHELOR THESIS

Wearable electronic orientation aid in straight paths for visually impaired people

Author: Jessica Páez Bonilla

Tutor: Irene Pérez Encinar
Director: Guillermo Prados Gimeno

Madrid, July 2015

To my beloved family,

and to you.. Jonathan

Acknowledgments

I would like to thank the people who supported and helped me in the development of this project:

Thank you Irene Pérez and Guillermo Prados for giving me the opportunity of being my Tutoress and Director. Your suggestions and advice were of great help.

I am also very thankful with the support given when needed by some of my previous colleagues at National Instruments's support department, Jesus Garrido, Ángel Sendín and their partner Daniel Murmman.

Thanks also to Dr. Umesh Rajashekar from New York University. His studies of Computational Vision using LabVIEW were very useful.

Finally, thanks to ONCE Aranjuez for making possible to do some tests with a visually impaired person.

Abstract

This project makes use of artificial vision, specifically of different image processing techniques in order to build a guidance aid for visually impaired people. While several navigation systems based on ultra-sound sensors have been developed, it is not the case of real time navigation systems based on edge detection techniques. Therefore the current project builds a prototype based on artificial vision, which is easily wearable. It is light and little cumbersome.

The main assumption of this guidance system is that there are many straight paths in different real world scenarios. These straight paths have parallel edges, which when captured as an image seem to converge to a single point called vanishing point. In the current project the vanishing point is calculated as a the crossing point between the two straight lines defining the path and a decision is taken based on the position of this point. This control algorithm has been developed using LabVIEW Real-Time software and has been deployed afterwards in a Real-Time target.

The prototype notifies (via an audio record) the visually impaired person about his deviation from the straight path. It also notifies if the person is completely out of the way with an alarm sound. This guidance system consists of an electronic device (which contains a real time processor along with an FPGA), a webcam, a battery and some earphones. In this project the FPGA is used for parallel playback of audio speech while vision processes take place in the Real-Time processor.

Keywords: guidance system, wearable electronics, artificial vision, image processing, LabVIEW Real-Time.

Resumen

Este proyecto hace uso de visión artificial, específicamente de distintas técnicas de procesamiento de imagen para construir un sistema de asistencia a la navegación para personas con discapacidades visuales. Mientras muchos sistemas de navegación basados en sensores de ultra sonido se han desarrollado, no es el caso de sistemas de navegación en tiempo real basados en técnicas de detección de bordes. Por tanto, en el presente proyecto se construye un prototipo basado en visión artificial, que es fácilmente portable. Es ligero y se puede llevar puesto cómodamente.

La principal suposición de este sistema de guiado es que hay muchos caminos rectos en distintos escenarios del mundo real. Estos caminos rectos poseen bordes paralelos, los cuales, al ser capturados en una imagen, parecen converger en un único punto llamado punto de fuga. En este proyecto el punto de fuga se calcula como el punto de corte entre las dos líneas rectas que definen el camino y se toma una decisión basándose en la posición de este punto. Este algoritmo de control se ha desarrollado usando el software LabVIEW Real-Time y se ha implementado posteriormente en un dispositivo con sistema operativo en tiempo real.

El prototipo notifica (a través de una grabación de audio) a la persona con discapacidad visual sobre su desviación del camino en línea recta. También notifica si la persona se ha salido por completo del camino con un sonido de alarma. Este sistema de guiado consiste en un dispositivo electrónico (el cual contiene un procesador en tiempo real junto con una FPGA), una webcam, una batería y unos auriculares. En el proyecto la FPGA se usa para la reproducción en paralelo de la grabación de audio mientras los procesos de visión tienen lugar en el procesador en tiempo real.

Palabras clave: sistema de guía, visión artificial, procesamiento de imagen, electrónica portable, LabVIEW Real-Time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

According to World Health Organization (WHO), latest records (august 2014), 285 million people are estimated to be visually impaired worlwide: 39 million are blind and 246 have low vision. Among them 90 per cent live in low-income settings [1]. An indepent and safe path following would have a great impact on these people's lifestyle, and this is the aim of this project.

Vision plays an important role in our life. People who lack a good vision face many troubles in their daily life, starting from knowing where they are going. Despite we are living in a high technological world, there aren't enough developed systems that help visually impaired people to acomplish their daily tasks like navigation yet. This is where the motivation for developing this project was born. The incentive is, therefore, to provide visually impaired people an aid for their navigation.

Most of the research that has been conducted up to now use external sensors and an infraestructure not easily wearable. Most of the mobility tools used include white canes and guide dogs. However, limited number of path finding systems have been developed on ergonomic mobile devices, and the ones developed with small devices were computationally challenged because of the limited hardware available. Very few use image processing for aiding low vision people to find their destination. Therefore achiving a simple, efficient and wearable device, free-hands, has been a key objective in the development of this project.

1.2 Objectives

The primary objective in this project has been to design and build a wearable system that aids people with visual impairments in their navigation in straight paths. This wearable system should work in real-time to ensure safety.

In order to achieve this primary goal, secondary objectives have been the following:

- The prototype should be light and easily wearable
- It should be free-hands and also easily usable, with no needs of previous training for its use
- It should operate in real-time

In order to achieve these objectives, it was necessary, first, to determine the proper electronic devices, and secondly, the peripherals most suitable for building this protoype. Afterwards, it was necessary to learn how to use and program them.

1.3 Problem statement

The challenge faced in the development of this prototype is getting a reliable guidance system using the images captured by a single camera as a mean for obtaining information about the real world. For the system to be reliable it is required that it provides outputs in real-time.

This project approaches the problem with a method for real-time navigation based on artificial vision, specifically, on edge and straigh line detection techniques. The proposed guidance system takes advantage of the existence of a large amount of straight paths in real world scenarios and of the existence of contrast differences in the lines defining these paths.

1.4 Structure of the document

The structure of this document is as follows:

- In chapter 2 there is an overview of the existing technology and the proposed solutions in the field of electronic navigation aids. Some techniques used in artificial vision and digital image processing for edge detection, straight path and vanishing point finding are taken into account as well.
- In chapters 3 and 4 the prototype's hardware and software are explained in detailed respectively. The reasons for choosing this device along with its programming language are mentioned. A global vision of LabVIEW environment and a definition of a Real-Time system are given.
- Chapter 5 describes the implemented system and analyzes it. It clears up every block of the main program: For each main part its flowchart and its block diagram are presented. Finally, there is a complete view of the whole developed program.
- In chapter 6 the results obtained in different scenarios are comented. Some initial tests are detailed and their conclusions. Prototype's appearance is shown. Restrictions of the system are detailed and finally it is possible to see the system functioning properly.
- Chapter 7 presents the different phases of this project. An estimation of the working hours invested in this project is made. Planification of the project over time is explained graphically with a Gantt's chart. In this chapter a cost estimation of the project is done.
- Finally, in chapter 8, some conclusions extracted from the project and future improvements for the prototype are mentioned.

Chapter 2

State of the art

Since 1960s, evolving technology has helped many researchers to build electronic devices for navigation. A first level categorization is as follows :

- Vision Enhancement
- Vision Replacement
- Vision Substitution (similar to vision enhancement but with the output being nonvisual, typically tactual or via audio).

In this work we are working in the vision substitution category. Subcategories of vision substitution, according to reference [2], are the following:

- **Electronic travel aids (ETAs)**: devices that transfrom information about the environment that would normally be relayed through vision into a form that can be conveyed through another sensory modality. (e.g. Obstacle detection systems)
- **Electronic orientation aids (EOAs)**: devices that provide orientation prior to, or during the travel. They can be external to or can be carried by the user (e.g. infrared transmitters and handheld receivers)
- **Position locator devices (PLDs)**: devices which include technologies like GPS.

This work can be placed in the first subcategory: ETAs but the information given has to do solely with orientation (second subcategory, EOAs). The developed prototype provides orientation to the user while travelling through audio speech. Similar works will be analyzed in the following sections.

2.1 Electronic Travel and Orientation Aids

It is important to do a brief review of some ETAs and EOAs, similar to the current project, developed until present. The presented systems appear in chronological order. However some of them are contemporary.

2.1.1 Travel Aids (ETAs)

Voice project

P.B.L. Mejier [3] based his project on the argument that human auditory system is capable of learning to process and interprete complex sound patterns. His prototype as shown in figure 2.1 consisted of a digital camera attached to some eyeglasses, a portable computer with the needed software and some earphones.

Figure 2.1: Voice Electronic Travel Aid

The digital camera captured images and the computer used an invertible one-to-one image to sound mapping. No filters were used, to reduce the risk of filtering important information, since he was based on the argument that human brain is powerful enough to process complex sound information. The sound patterns were sent to the earphones.

This system was lightweight, small and cheap. Lately, the software was embedded on a cellphone and thus the user can use cellphone's earphones and camera. In addition, a sonar extension for better representation of the environment was available. The results of this prototype were quite good but the individuals who used them required an extensive training because of the complicated sound patterns.

University of Stuttgart Project

A portable system that assisted visually impaired people in indoor navigation was developed by researchers in University of Stuttgart in Germany [4]. The prototype consisted of a sensor module with a detachable cane and a portable computer. The sensor as shown in figure 2.2 is equipped with two cameras, an

Figure 2.2: University of Stuttgart's ETA

small keyboard, a digital compass, a 3-D inclinator, and a loudspeaker.

It works as follows: By pressing the designated keys, different sequences and loudness options can be chosen and inquiries concerning object's features can be sent to the portable computer. After successful evaluation these inquiries are acoustically answered over a text-to-speech engine and the loudspeaker.

The computer had a software color and distance detection and could also measure the size of objects. It also had wireless local area network (WLAN) capabilities. The device worked almost in real-time.

Virtual Acoustic Space

Figure 2.3: Virtual Acoustic Space ETA

Virtual acoustic space was developed by researchers at Instituto de Astrofisica de Canarias (IAC) [5]. A sound map of the environment is created in such a way that the users can orientate themselves by building a perception of space at neuronal level. The prototype, as seen in figure 2.3, consists of two cameras attached to some conventional eyeglasses, a processor and headphones. The cameras, using stereoscopic vision, capture surroundings's information.

In this prototype the processor creates a depth map with attributes like color, distance, or texture and then generates sounds correspondingly. As shown previously the eyeglasses and the processor (size like a portable CD-player) are easily wearable. The experiments showed that low vision people in most cases could detect objects and their distances and they could even extract information of objects in small experimental rooms (like identifying windows, tables or doors). However this system has not been tested in real environments.

Navigation Assistance for Visually Impaired (NAVI)

Sainarayanan and Nagarajan from University Malaysia Sabah [6] developed an ETA (sound-based) to assist blind people in obstacle identification during their navigation. This prototype named NAVI, shown in figure 2.4 consisted of a digital camera, headgear (holds camera), stereo headphones, rechargeable batteries, a single-board processing system (SBPS), and a vest (that holds SBPS and batteries).

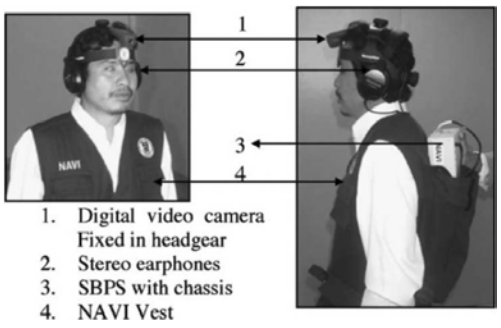


Figure 2.4: NAVI and its components

Developers's idea was that humans only focus on the objects that are in front of our center of vision, therefore it is important to distinguish between obstacles and background. In their prototype the camera captured grayscale video at a resolution of 32 x 32 pixels. Afterwards, they used fuzzy learning vector quantization (LVQ) to classify either background or objects using different gray level features. Finally, object pixels were enhanced, the background suppressed and image was splitted into left and right parts and transformed into an stereo sound sent to the user.

2.1.2 Orientation Aids (EOAs)

Existing orientation and wayfinding aids are limited by:

- the types, accuracy, and amount of information these aids can provide
- the types of environments in which they can function
- the user interface structure

One of the reasons for these limitations is that while there has been a great deal of research in the area of electronic travel aids for obstacle avoidance (in the previous section we saw a brief sample of it), there has been little comparable research and development of orientation and wayfinding devices.

What has been used up to now

Braille labels have been used as orientation aids for decades. They have been used to label important objects, such as pedestrian lights, doors, and to describe the layout of surrounding areas. However, this is not enough as there is no means of knowing where and when a braille label can be found and the information it may contain.

Also maps with braille labels are available to a limited extent. These maps provide an overview about the area along with walking routes. But this solution is not very practical as these maps are very cumbersome to carry and some people may have troubles translating these tactile maps into an aid that helps them to maintain their orientation to the physical environment as they travel through it [7]. In addition, the use of these maps is even more difficult for older people who experience some loss of touch sensitivity or cognitive loss.

Talking Signs, developed by the Smith-Kettlewell Eye Research Institute, provide the equivalent of visual signage orientation information [8],[9]. These devices employed visible light or infrared (IR), to transmit a message recorded in each sign of the environment. This message could be received via the use of a special hand-held receiver. These signs required a power source and could be installed in a variety of outdoor and indoor locations. The light beamed from each sign enabled the user to orient himself and walk in a desired direction. Specialized versions of these signs were integrated into pedestrian crossing signals to provide *Walk, Don't Walk* information to the blind.

Indeed the use of those signs is an advantage over the braille labels given the fact that they provide orientation information with time enough for the user to act. However the procedures for receiving the message make this device not

as easy to use as it should be. As explained in [8],[9] this procedure is time-consuming and awkard, mainly when the blind person is carrying something. In addition it assumes that the user is capable of walking in a straight path (which is not really the case). It would also require that the person is trained in the location of the signs in specific environments. A more usable receiver interface would not need to be held or scanned about to determine the presence or location of Talking Sign transmitters. Further, it would provide continuous feedback to help the user follow a straight line path to their intended destination.

In 1991 Blasch [7] completed a study of *Environmental information needs for orientation and wayfinding*. As a result of this study it was described the most usable form for wayfinding and orientation information. Blasch's study showed that it was very important to give the information in situ (in real time) in a concise and unobtrusive manner. Also, research in the area of virtual sonic environments by Jack Loomis, Ph.D., brought to initial prototypes and tests of an orientation/guidance system for the visually impaired [10].

Pedestrian navigation using vanishing points and lines

The method developed by Christoph Kessler, and Gert F. Trommer at [11], from the Institute of Systems Optimization (ITE) and Karlsruhe Institute of Technology (KIT), demonstrated that attitude determination is a key point in navigation systems, specially in the ones comprising microelectrical and micromechanical (MEMS) sensors. It was shown that attitude information can be extracted from images of a moving mono-camera. It also showed that advantage can be taken of many structures in man-made buildings, given the fact that those have parallel and perpendicular edges, and these edges can be extracted from captured video sequences.

The geometric relationship can be analyzed using the concept of Vanishing Point (VP) and of Vanishing Line (VL). Vanishing points are the virtual intersection points of some edges. Virtual lines are the intersection lines of planes. In this research they achieved the camera attitude determination from the fact that moving the camera leads to a change in the projected edges in the received image.

This approach uses line features instead of detecting point features, these lines arise from the general structure inside buildings. It was tested with recorded data form an Integrated Pedestrian Navigation System (IPNS), shown in figure 2.5 As it can be seen, it is composed of inertial measurement units (IMUs) in the foot and in the torso, as well as a magnetometer, a camera and a laser scanner.

Figure 2.5: Integrated Pedestrian Navigation System (IPNS)

Not all of the images acquired were processed for aiding, but only a subset, leading to a frame rate of 1 Hz. The reference attitude for subsequent vision-based attitude aiding was determined from the first video frame with a detected central VP and valid magnetometer measurements. In the following images, VLs and a central VP were detected. The scene orientation corresponding to an observed central VP was determined based on the IPNS attitude estimation. This is correct as long as IPNS attitude deviations due to magnetic disturbances remain below 90 degrees yaw angle error, which was always the case according to their experience. Every time a central VP and a VL were recognized, the torso IMU attitude was determined based on the camera attitude and the known alignment of the camera and the IMU.

Navatar: Low-cost accessible navigation system

At University of Nevada, some professors of the computer science engineering team, Kostas Bekris and Eelke Folmer presented their indoor navigation system for people with visual impairments [12].

Their navigation system uses some 2D architectural maps which, in many cases, are available for many buildings, and sensors as accelerometeres and compasses (located on a smartphone) to aid the navigation. This system locates and also tracks the user inside the building. If finds the most suitable path and gives the user step-by-step instructions to get to the destination. These instructions are given using synthetic speech and users confirm the receiving of this landmark (signal) by pressing a button on the phone or by verbal confirmation.

An important advantage of this approach is that leaves the user with both hands free for using a cane or for recognizing tactile landmarks.

Indoor navigation system using a smartphone

In Saud University, Mai A. Al-Ammar King and AbdulMalik S. Al-Salman from computer science department and Hend S. Al-Khalifa King from the Information Technology department developed a remarkable solution based on a smartphone device. The primary service of the proposed system was to find a path to a specific destination [13].

Their proposed system is an assistive technology that guides blind people to a specific destination inside a building. It assumes that the users help themselves to recognize their surroundings or use another assistive technology as white canes for obstacle avoidance.

It is composed of five components as it can be seen in figure 2.6: mapping, positioning, navigation, text to speech, and an interface. The most remarkable are the following:

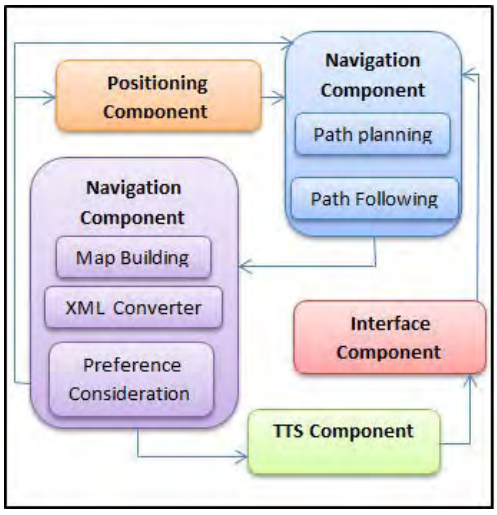


Figure 2.6: Overall system design of indoor navigation using a smartphone

- **Mapping**: It constructs a map of a building in which the system is operating. The system use XML to describe these maps.
- **Positioning**: It determines the position within the building. It uses a network based approach like Wi-Fi.
- **Navigation**: This determines the suitable path from the current position to a specific destination. It contains two subcomponents: path planning and path following. The prototype developed in this project performs just the path following, assuming the person is already in the right path. Therefore it could be included in a bigger system like this one.

2.2 Edge and straight line detection in navigation

In this section, firstly most remarkable techniques used for edge detection will be defined. Secondly, it will be shown how these techniques have been used in the development of different navigation systems. Thirdly different methods used to find straight lines will be mentioned. Finally, projective geometry will be defined, and its application for vanishing point's calculations.

2.2.1 Edges

Most known techniques for edge detection include *Canny, Sobel, Prewitt* and *Roberts*. In this project three of them have been used.

- *Canny*:
  This operator takes as input a gray scale image, and produces as output an image showing the positions of tracked intensity discontinuities. It works in a multi-stage process: image is smoothed by *Gaussian Convolution*, and afterwards, a derivative operator is applied to the smoothed image. More details of this technique can be seen in [14]

- *Sobel*:
  This operator performs a 2-D spatial gradient measurement on an image. Therefore, it emphasizes regions of high spatial frequency that correspond to edges. This operator consists of a pair of $3x3$ convolution kernels. It is used to find the approximate absolute gradient magnitude at each point in an input grayscale image. Further details can be read in [15].

- *Prewitt*:
  This operator calculates edges by using difference between vertical and horizantal pixel intensities of an image. It simply works as a first order derivate and calculates the difference of pixel intensities in a edge region. It applies horizontal and vertical convolution filters in sequence. After they have been applied results are summed to form the final image. More details can be found at [16].

- *Roberts*:
  This operator performs a 2-D spatial gradient measurement on an image. It then highlights regions of high spatial frequency which often correspond to edges. It consists of a pair of $2x2$ convolution kernels, one rotated 90 degrees with respect to the other. For more details consult [17].

Nimali Rajakaruna Iain Murray of Curtin University, Perth, conducted a research that focuses on methods for real time local navigation (indoor and outdoor) based on edge detection techniques using hand held devices [18].

Their protoype is mounted in the chest and hips of a person to determine an optimal position while the person is walking on sidewalks or stairways. When capturing images, they saw that orienting angles and the walking speed of the person affect to the image quality (may increase its noise). In order to determine an optimal solution, 3 frames per second are captured in their system for processing, as they consider it is sufficient for the average walking speed.

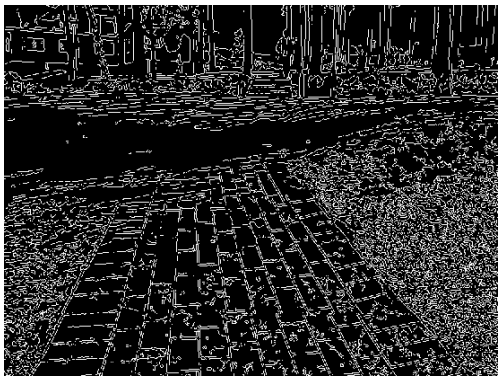


Figure 2.7: Image after applying DSCED filter

Under low light conditions the images captured by the camera are quite noisy and this noise should be removed for further processing. In this situation they use image enhancement and histogram equalization to enhance the main features of the image and remove the noise. Rajakaruna and Murray used the Discrete Singular Convolution Edge Detector (DSCED) [19] in order to obtain the initial edges. An example of edge detection using this method can be seen in figure 2.7.

In contrast with Canny edge detector (which depends on the computational bandwith, *W*, and standard deviation, sigma), DSCED has additional parameter alpha, which depends on the frequency selection.

The major challenge Rajakaruna and Murray encountered were the limited resources available in mobile devices. They found out that due to the processing power and memory limitations in mobile devices and the high-expected overall throughput, the complexity of the image processing algorithms had to be minimized and optimized.

2.2.2 Straight lines

Common methods being used for straight line detection are the following:

Hough Transform

The Hough Transform [20] is a detection method for parameterized geometric shapes based on an accumulation mechanism. Lines are expressed in polar form. The line parameters span up a space in which all parameter combinations of lines containing a particular edge pixel are accumulated. The peaks in this so-called accumulator space represent the actual lines in the image. The output of this method can be seen in figure 2.8

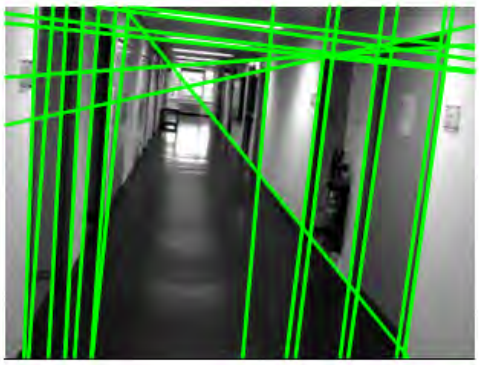


Figure 2.8: Hough method for line detection

According to [21] and [22] this method is robust to noise, but the accuracy may suffer from discretization errors and phantom lines may occur due to ambiguous peaks in the accumulator space.

IRLS-Iteratively Reweighted Least Squares

A line fitting method using IRLS algorithm is described in [23]. This method yields the edges extraction that can be seen in figure 2.9

Iteratively reweighted least squares is used for statistical estimation. It uses linear regression and non-linear parametrizations. It can be used with Gauss-Newton and Levenberg-Marquardt numerical algorithms [24].

Figure 2.9: IRLS method for line detection

Line Fitting using Connected Components

Another approach is taken in [25], where adjacent edge pixels with similar gradient direction are labeled using a Connected-Component algorithm. For the normalized [x, y] coordinates of each list of connected edge pixels, the eigenvalues and eigenvectors are analyzed, from which the line parameters are derived. The output is shown in figure 2.10

Figure 2.10: Line Fitting Method

It iteratively searches and linkes edges with points that are in the same direction. Lines are fitted by connecting endpoints of edge segments.

2.2.3 Projective geometry and vanishing points

Projective geometry describes the properties of projective transformations which take place for example in cameras. The projective action of a camera on points in 3D-space can be expressed as $x = P * X$ where P is the camera matrix and X the homogenous coordinates of points in 3D-space and on the 2D image plane [26]. A line in 3D-space is represented by

$$x(\lambda) = A + \lambda D$$

with A being a finite point on the line and D the line direction in homogenous coordinates. For better understanding see figure 2.11

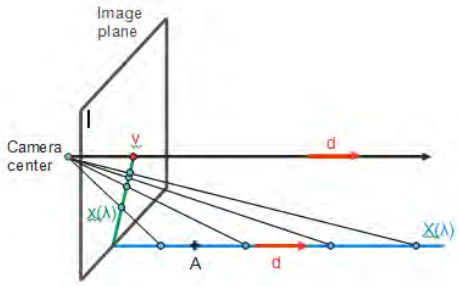


Figure 2.11: Projection of a line in 3D-space onto the camera image plane

Attitude information can be extracted from a VL detected in a camera image. A VL contains the VPs generated by lines within the plane. The benefit of extracting and tracking VPs and VLs for indoor navigation applications is that they contain information about attitude changes, which is used for navigation aiding. In this project VP extraction will be made and the guidance decisions will be taken upon this point, that's why this part is quite important.

Chapter 3

Prototype's Hardware

Having in mind the objectives defined in the development of the prototype, the requirements for the hardware are the following:

- A real time processor along with a FPGA, in order to make possible to have parallel processes executing at the same time. This will allow to have a system that responds in real-time.
- Wearable devices: Light and small.
- A powerful design platform that counts with the existence of a community of users would be of great help for making possible the further development in future of the prototype.
- Hardware with built-in vision capabilities for performing embedded vision processing.

After doing a research of the available devices that fulfill these needs, it was found that the best solution for the main hardware was NI myRIO. Reasons are the following:

- NI myRIO is an academic hardware device that allows to design different control systems. It contains an FPGA and a real-time processor in a single chip. This fulfills the first stated need.
- It is an small device, which allows to deploy on it control algorithms, these two are critical characteristics for a wearable (second need) control system.
- Its design platform is LabVIEW, which is a graphical design platform that allows to program and build complex control systems in relatively short periods of time. It also has a user's community that allows the sharing of knowledge for giving usability to this prototype. This satisfies third stated need.

In table 3.1 some comparisons made between different electronic devices available in market, that were considered for the development of the prototype, are presented. Analysis can be summarized as follows:

Electronic Device	**Real-time processor**	**FPGA**	**Built-in vision capabilities**	**Design Platform**
Arduino	Yes	No	No	Open-source Arduino Software
Blackberry Pi	General purpose processor	No	Yes	Open-source (C, C++, Java)
Spartan 3E Xilinx	No	Yes	Yes	VHDL
Beaglebone Black	No	Yes	Yes	Bonescript
myRIO-1900	Yes	Yes	Yes	LabVIEW

Table 3.1: Hardware Comparisons

Among the listed characteristic, the decisive feature to choose the myRIO device has been its graphical system design platform, LabVIEW. This will be analyzed in detail in next chapter. Having an FPGA is also a great advantage, as it allows to build hardware, just by creating it through software. Reusing software is always cheaper. It is also remarkable, the existing community of LabVIEW users, as it is possible to share ideas and projects. This gives visibility to the developed prototype and make possible that in future other myRIO and LabVIEW users can make use of it and add functionalities.

3.1 RIO Architecture

RIO stands for Reconfigurable Input/Output. This reconfigurable devices are based mainly on six components, as it can be seen in figure 3.1. This figure can be found at National Instruments website [27].

- A processor
- A renconfigurable FPGA
- An AXI bus for communicating the processor and the FPGA
- Analog inputs and outputs
- DIO Digital input/output
- UART lines

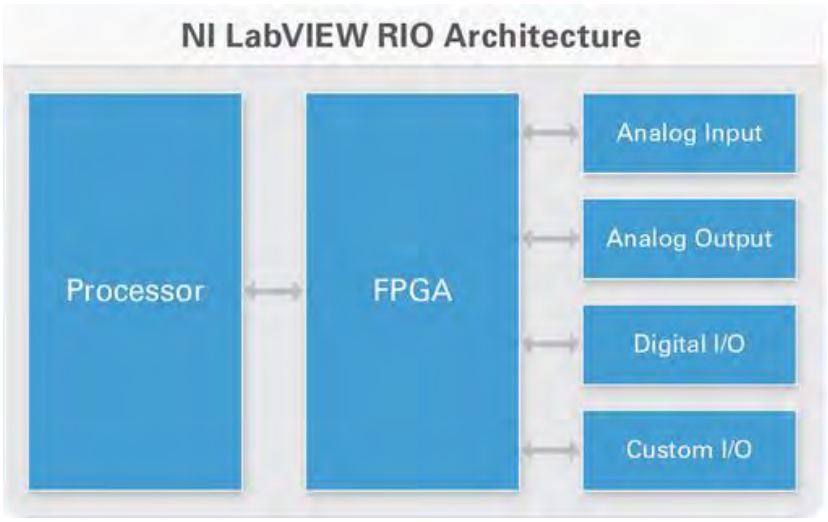


Figure 3.1: RIO components

These components provide the ability to create custom hardware circuitry with high flexibility. In this project the first four items of this architecture have been used.

There are several devices in this hardware family. Most of these devices are used for industrial purposes, and are depicted in figure 3.2 in increasing processing capacity order.

In the x-axis of this deployment curve, performance is represented, and cost is in the y-axis. MyRIO is not an industrial-oriented device, but academical, and

Figure 3.2: RIO Architecture

it would be situated in this graph between single-board RIO and Compact RIO.

MyRIO has specifications and performance quite similar to the Single-Board RIO, which is designed for high-volume and OEM (original equipment manufacturer) applications that require high reliability and performance. More information can be found at [27].

3.2 RIO and vision

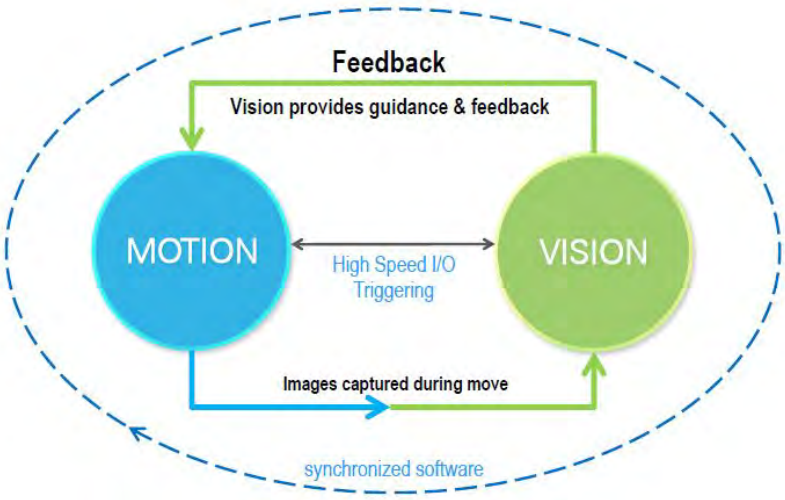


Figure 3.3: Vision combined with RIO hardware

RIO devices, like myRIO have built-in vision capabilities. The CompactRIO platform can accelerate embedded vision applications, satisfying in this way the last need (fourth) stated at the beginning of this chapter. A vision system which uses RIO hardware can provide information about motion, in the case of this project, it provides trajectory guidance. A diagram of this can be seen at figure 3.3, taken from [27].

This special characteristic of RIO devices, its built-in vision capabilities, has been another important factor for choosing this hardware among other alternatives.

Another important reason for choosing myRIO, as already mentioned, is its programming way. The control software for a vision application can be implemented in a RIO hardware device using the NI Vision Development Module, which includes many image processing functions that can run on both, a real-time processor, and an FPGA. This vision module will be explained in detail in the next chapter.

With this combination of characteristics this protoype could be built in an efficient way.

3.3 myRIO 1900

Figure 3.4: myRIO device

For the previous mentioned characteristics myRIO is a device suitable for the current project. It allows to have an embedded control system based on artificial vision.

In this section it will be explained in detailed its hardware architecture and its components, as well as its physical characteristics: dimensions and weight. The parts of the device that have been used in the project will be detailed. However, if a complete description of the device is needed, the *NI myRIO 1900: User guide and specifications* can be found in the annex.

As it will be shown, myRIO's physical characteristics allow to carry easily this device in a small bag which follows the waist, like a belt. Therefore it is easily wearable and allows the free-hands characteristic, which was one of the prototype's objectives.

As previously mentioned, myRIO 1900 is a reconfigurable device that provides analog input (AI), analog output (AO), digital input and output(DIO), audio in and out, and power output in a compact embedded device. Its hardware block diagram can be seen in figure 3.5. Figure and data are extracted from [28].

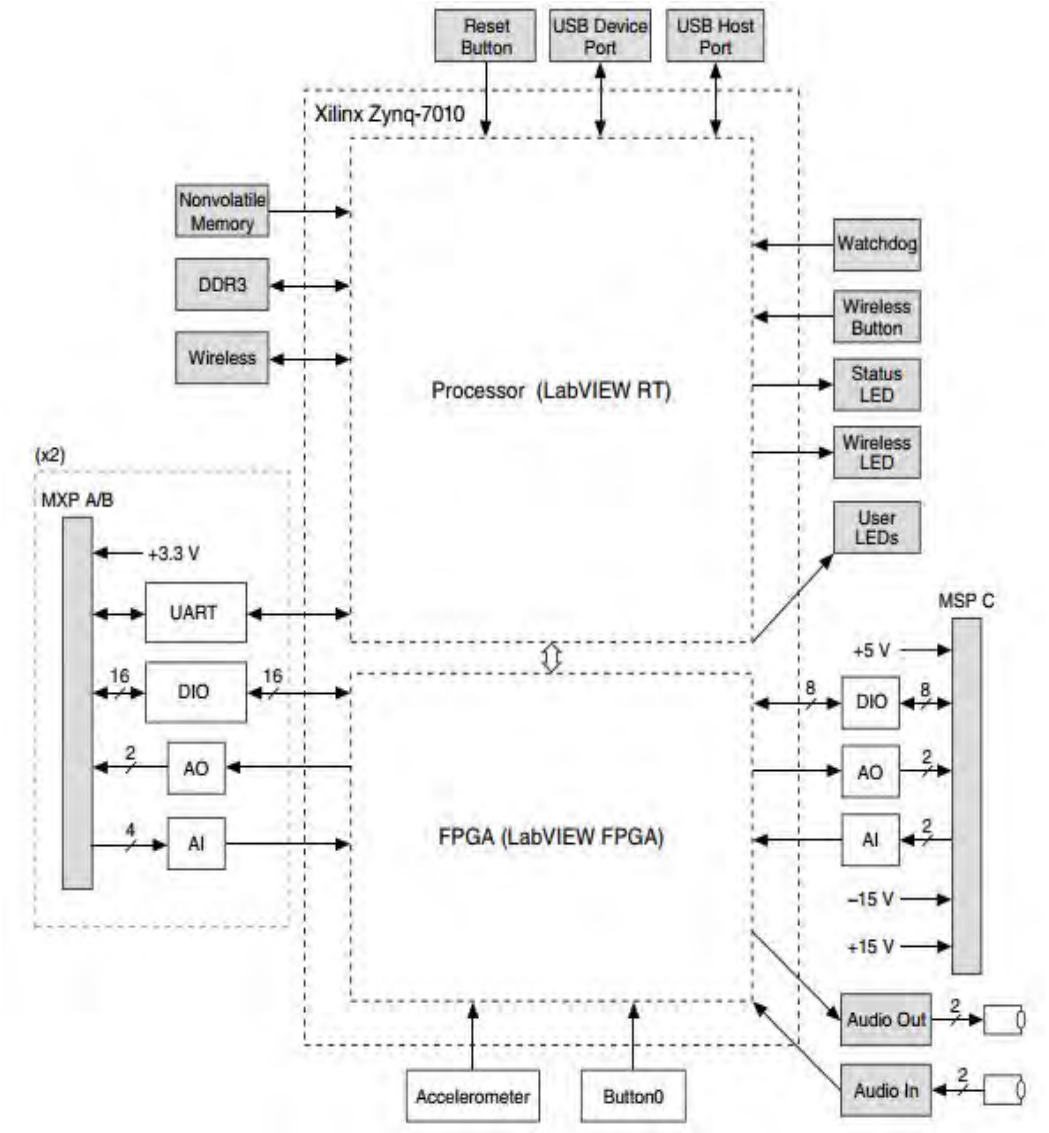


Figure 3.5: myRIO hardware block diagram

This device's specifications are the following:

- Real Time Processor:
  Type: Xilinx Z-7010
  Speed: 667MHz
  Processor cores: 2

- Memory:
  Nonvolatile memory: 256MB
  DDR3 memory: 512MB

- FPGA Type: Xilinx Z-7010

- USB Ports: USB host port and USB device port: both 2.0 Hi-Speed

- Analog Output (Audio Output):
  Configuration: One stereo output consisting of two AC-coupled, single-ended channels
  Output impedance: 100Ω.
  Bandwidth: 70Hz to values smaller than 50 kHz into 32Ω load, and 2Hz to values smaller than 50 kHz into high-impedance load.

- Power Requirements:
  Power supply voltage range: 6-16VDC
  Maximum power consumption: 14W
  Typical idle power consumption: 2.6W

- Physical Characteristics:
  Weight: 193g
  Dimensions: 120mm x 72mm x 25mm

3.4 Other devices used

In order to achieve the fulfilling of the objectives of the project, it was necessary to search for the complementary devices for building the prototype. Factors taken into account in the election process were compatibility with the main hardware device, NI myRIO, (for the webcam) or usability (earphones instead of headphones). Among several possibilities, the following were choosen:

Webcam

Figure 3.6: Webcam NGS

It was chosen the Xpress Cam-300 by NGS Webcam, whose main characteristics are:

- Resolution: VGA
- CMOS sensor: 300Kpx
- Video Resolution: 5Mpx (640x480 pixels)
- Image Resolution: 8Mpx
- 30 FPS
- Zoom

Not all USB cameras work with myRIO. The NI drivers for USB cameras use the USB Video Device Class (UVC) protocol to acquire images. Not all cameras expose full featured support at the UVC layer. But this webcam uses this protocol and therefore can be used in the prototype.

Earphones

These are conected to the audio out port of myRIO and are used for giving the guidance information to the user through speech. It is enough with the use of one of the pair. In this way the user can still receive other important audio information from the environment. This is not the case of headphones.

Figure 3.7: Headphones

Rechargable Battery

It was used a 2 cells lithium (Li-Po) battery of 7.4V and 2000 mAh which was enough to power the myRIO and the webcam.

Figure 3.8: Li-Po Battery

It was decided to use a rechargable battery so the prototype can be used several times. With the battery fully charged the system can be functioning for, approximately, an hour.

Chargers

For the battery recharging two different **types of chargers** can be used:

Figure 3.9: AC-DC converter (left) / DC-DC converter(right)

- **AC-DC Converter** (Figure 3.9 left):
  This charger can be plug in directly to the line current, therefore it receives an input voltage of 100-240 VAC at f=50 to 60Hz.
- **DC-DC Converter** (Figure 3.9 right):
  This charger has to be used with a DC power supply. It receives an input of 10V-15V.

Chapter 4

Software

As mentioned before, one of the main reasons for choosing myRIO was the way it can be programmed: through LabVIEW.

LabVIEW is a highly productive development environment for creating custom applications that interact with real-world data. Productivity is the key benefit of using LabVIEW: higher quality projects can be achieved in less time. This has made possible to achieve all the objectives set at the beginning.

LabVIEW is also unique because it makes posible that a wide variety of tools can be available in a single environment, ensuring compatibility.

4.1 LabVIEW Environment

LabVIEW, stands for *Laboratory Virtual Instrument Engineering Workbench*. It is a software development environment that contains numerous components, as can be seen in figure 4.1, taken from [29]

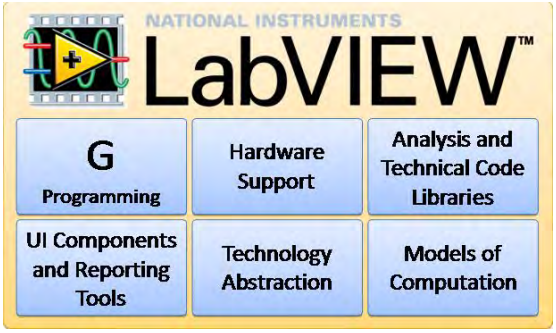


Figure 4.1: LabVIEW's components

In this project, advantage has been taken mainly of the G programming and of the technology abstraction components.

Graphical Programming

When writing a program in LabVIEW an user interface is created, named *Front Panel*. The code is represented graphically in the *Block Diagram*. LabVIEW's programs or subroutines are called *Virtual Instruments (VIs)*. *Front panel* is built with controls (inputs) and indicators (outputs). Indicators display the results of the functions performed inside the VI to the inputs.

Block diagram contains the graphical source code. Inputs and outputs placed on the front panel of a VI appear in the block diagram as terminals. The block diagram contains different structures and functions which perform operations on control values and supply the results to the indicators.

In figure 4.2 it is shown an VI used in this project for the edge detection process. There it can be seen the front panel with the inputs and outputs of the VI and the block diagram with the implemented code. In the block diagram all the controls placed on the front panel appear, as mentioned before, as terminals. It can also be seen that *wires* are used to transmit data values of each variable.

It is an scalable environment. Therefore inside a VI there may be several subVIs. SubVIs may be referred as nodes as well. Nodes are connected to each other using the above mentioned *wires*. In addition to the block diagram and the front panel, each VI has also a *connector panel*, used to represent which inputs it receives from the *calling VI* and which values it outputs. This connector panel is also shown in figure 4.2, it is the square on the upper right corner of the front panel window.

LabVIEW's scalability implies that, in most of the cases, each VI can be tested before being embeddes as a subroutine (subVI) into a larger program. Advantage of this feature has been taken in the development of this project.

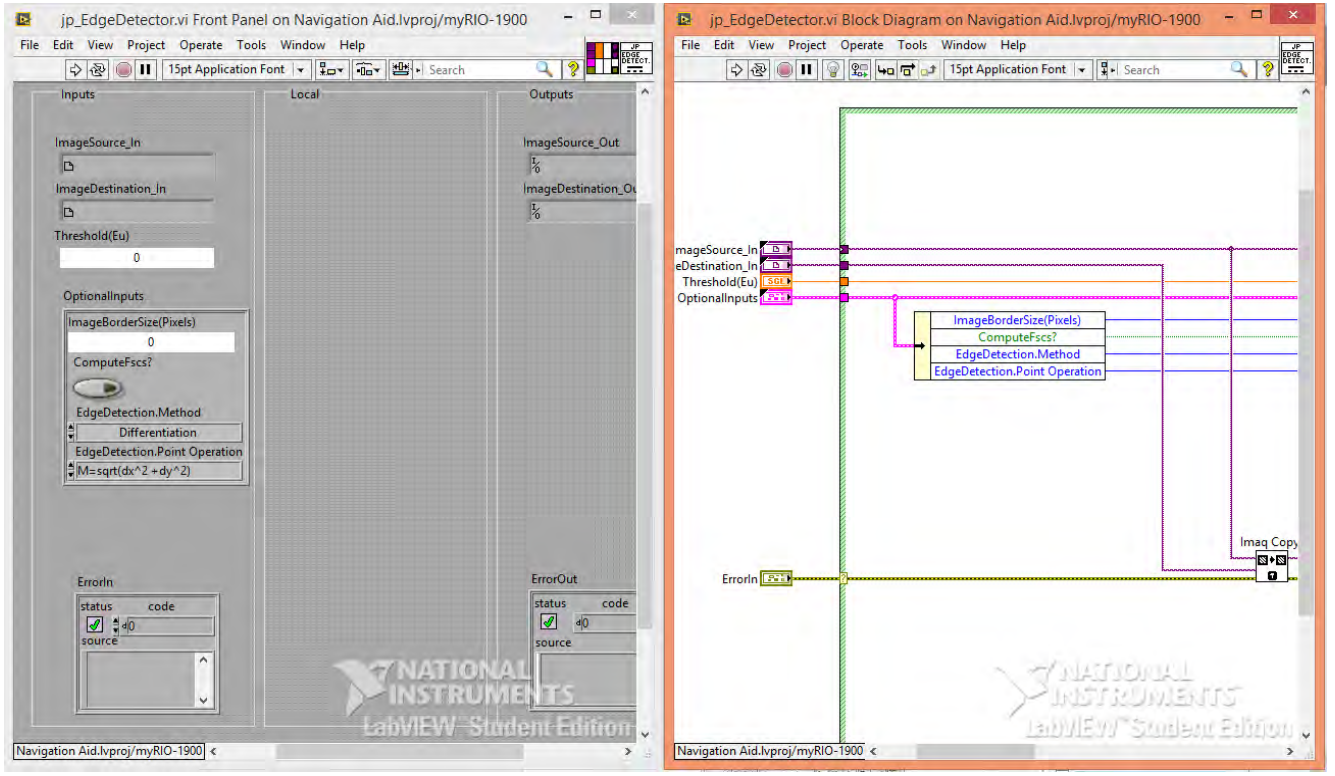


Figure 4.2: Example of a LabVIEW VI

Technology Abstraction

With LabVIEW writing low-level code is not neeeded. This allows to focus on the bigger problem and on finding the best solution.

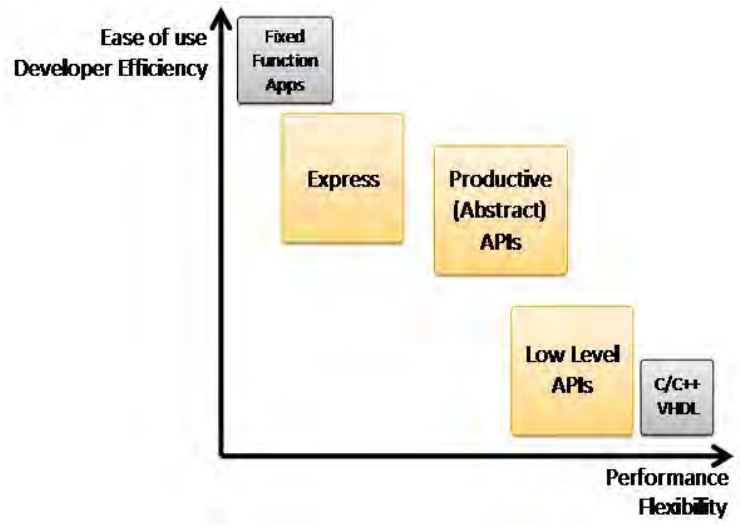


Figure 4.3: Ease of use vs performance flexibility graph.

One example of this is how LabVIEW is able to generate multithreaded code for execution on multicore processors or program FPGAs without having to write VHDL code, required for FPGA, explicitly.

In the LabVIEW environment, a program is as complex as the programmer wants it to be. In general, it is hard to find an equilibrium between simplicity of a program and its performance flexibility. If we want a high degree of personalization then it is necessary to go to very low-level, this is depicted in figure 4.3. This project has been programmed mainly with Productive (Abstract) APIs.

Abstract Programming Interface: With this type of APIs it is possible to work with high-level functions that keep a balance between abstracting the unnecessary tasks, such as memory management and some format conversions, but keep the flexibility of being able to customize almost every aspect of the project being carried out.

Dataflow Programming

Another important characteristic of LabVIEW is that it is a dataflow programming language, which means that execution of the program is determined by the flow of data. This causes to have a computional model inherently parallel. Text-based programming languages are not, which makes difficult to build parallel programs.

Wires propagate variables values and nodes can execute as soon as *all* of its inputs are available. This may be the case for multiple nodes at the same time, that's why LabVIEW is capable of parallel execution [30].

4.2 Building a Real-Time system

In the current project a real-time target was programmed. Therefore the programming was made in a RTOS (Real-Time Operating System). Firstly a RTOS definition will be made:

What a Real-Time OS is

A RTOS manages the hardware resources and hosting applications running on a device (as a general purpose OS does) but is specially designed to run applications with high degree of reliability and very precise timing.

It must have a known maximum time for each critical operation it performs. This can be very important in systems where a program delay could cause a safety threat, as is the case in this project. RTOS do this by providing programmers with a high degree of control over how tasks are prioritized. Therefore RTOS execute deterministic applications (applications whose timing can be guaranteed within a certain margin of error).

A real-time system consists of software and hardware components. The software components include LabVIEW, the RT Engine, and the LabVIEW project. The hardware components of a real-time system include a host computer and an RT target, which in this case is myRIO.

Host computer
This is the computer which has LabVIEW and LabVIEW Real-Time module installed and on which the deterministic application is developed. After the main VI is created it can be deployed (downloaded and runned) on the RT target.

Real-Time Module
It extends the capabilities of LabVIEW with additional tools for the creation of deterministic VIs. It also provides tools for the debugging and the deploying of these applications.

Real-Time Engine
It is a version of LabVIEW that runs on RT targets. It runs the VIs being deployed to RT targets. It runs on a RTOS, which ensures that the LabVIEW execution stick to real-time operation.

Real-Time Target
It refers to hardware that runs the RT Engine. It is the case of myRIO. Other examples of RT targets are the devices shown in figure 3.2.

4.3 Vision for LabVIEW software

Using LabVIEW Real-Time and the Vision Module, all the necessary tools for building a complete vision application on a embedded platform, are available. As mentioned in the previous section, Real-Time Module provides real-time execution capabilities. NI-IMAQ provides the vision acquisition components, and NI Vision the image manipulation and analysis functions.

In this section the **NI Vision Development Module** will be explained. This module is a library which contains VIs that can be used to develop machine vision and different imaging applications. NI Vision is organized into three main function palettes: Vision Utilities, Image Processing and Machine Vision. The groups of these palettes which are remarkable and the ones which have been used in this project will be briefly explained. A complete description of each palette can be found at the NI Vision Manual for LabVIEW Users [31].

4.3.1 Vision Utilities

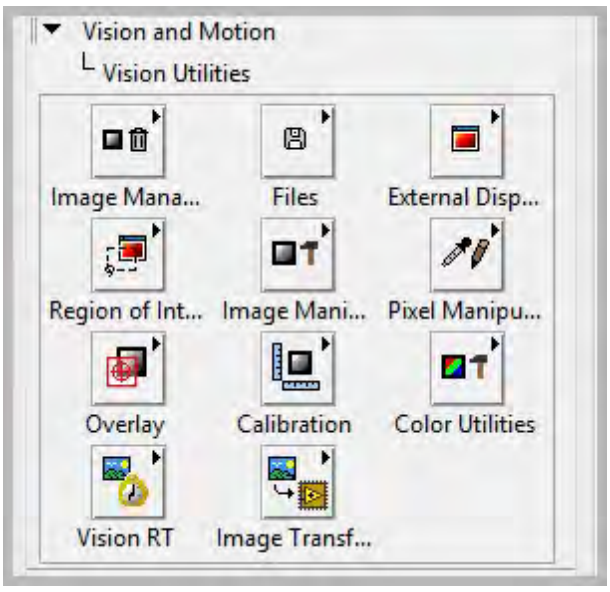


Figure 4.4: Vision Utilities Palette

Vision Utilities pallete, shown in figure 4.4, allows to manipulate and display images.

- **Image Management**: These VIs create and dispose images, set and read their attributes (such as their size and their offset), and copy one image to another.

- **Region of Interest (ROIs)**: These VIs are used for programatically define ROIs and convert ROIs to and from image masks. In the development of this prototype 2 ROIs are used. They are shown in next chapter (figure 5.16).
- **Image Manipulation**:Modify the spatial content of images (resample, parts extraction, rotation or shifting).
- **Pixel Manipulation**: These VIs read and set pixel values in an image or along a row or column, fill the pixels in an image with a particular value and convert an image to and from a 2D LabVIEW array.
- **Calibration**: Group of VIs that spatially calibrate an image to take accurate, real-world measurements. They can convert pixel coordinates to real-world coordinates.

4.3.2 Image Processing

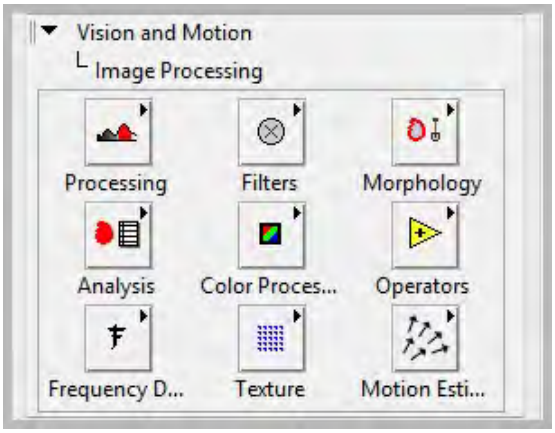


Figure 4.5: Image Processing Palette

Image Processing functions (shown in figure 4.5) allow to analyze, filter, and process the images. In this project functions from this palette have been used for performing different methods (prewitt, sobel or roberts) in order to get the edge detection.

- **Processing**: These VIs process grayscale and binary images. For example, it converts a grayscale image into a binary one using different thresholding techniques. They also invert values in images.
- **Filters**: Group of VIs that filter an image to enhance the information in it. They can smooth an image, remove noise, and highlight or enhance edges in it.

- **Morphology**: These VIs perform morphological operations. Some perform basic operations like dilation or erosion on grayscale/binary images. Others improve the quality of these images by removing particles or filling holes.
- **Analysis**: They analyze the content of the binary image. Compute histogram and make grayscale statistics. They also retrieve pixel information along any one-dimensional profile.
- **Operators**: Group of VIs that perform basic arithmetic (add, subtract, multiply,and divide an image with other images or constants) and logical operations (*AND/NAND, OR/NOR, XOR/XNOR*) and make pixel comparisons between an image and other images or a constant.

4.3.3 Machine Vision

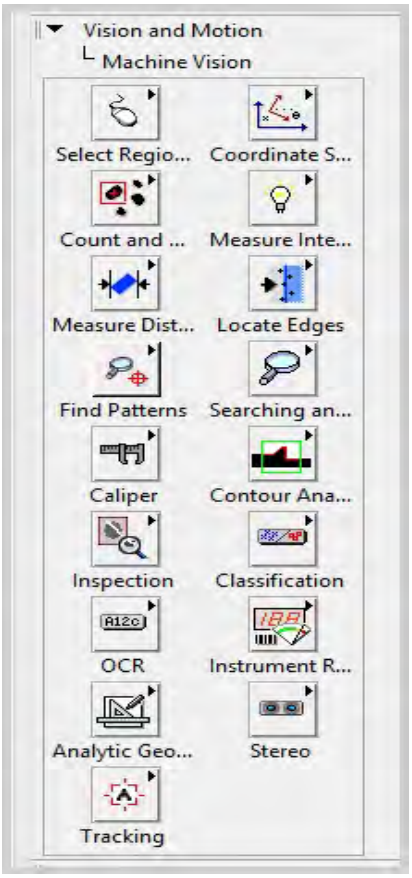


Figure 4.6: Image Processing Palette

Machine Vision palette, shown in figure 4.6, is formed by high-level VIs that perform common machine vision tasks.

- **Find Patterns**: Learns and searches for a pattern in an image.
- **Analytic Geometry**: These VIs perform analytical geometry computations. They can be used for fitting lines, circles and ellipses to a set of points in images. They also measure distances, find angles between lines or compute polygons's areas. In this project these VIs have been used for finding the intersection point of two lines.
- **Locate Edges**: This group of VIs that locate vertical, horizontal, and circular edges. One of this VIs has been key in the development of this project.

Chapter 5

Program's Design

In this chapter the methodology used for the design and the design itself of the developed program will be explained.

Best Practices and methodology for a good design

Design is made mainly with a top-down approach. The whole system has been broken down into its compositional sub-systems in order to gain insight on each part. First, an overview of the system is formulated, specifying but not detailing any first-level subsystems. Afterwards, each subsystem is then redefined in greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements.

Bottom-up methodology has also been used when getting to know the main tool for creating the system: LabVIEW. Courses of LabVIEW were taken and those were useful in order to learn good programming practices. These practices, like scalability, were used in the program's design. Several subVIs (subroutines) were created following a template in which inputs (at left) follow a dataflow execution and yield outputs (at right).

When detailing and redefining each subsystem, scientific method was applied. In the control algorith design, some conjectures (hypotheses) were made, and then experiments were carried out based on those hypotheses to determine whether the original conjecture was correct.

For a better comprehension of the program's design, it has been divided into 6 main sub-systems. Figure 5.1 shows the program's flowchart. Each subsystem has an identifying color, that will be maintained in its own flowchart and in its own VI, throughout the chapter.

Figure 5.1: Main program's flowchart

As it can be seen, it is a cyclic process, which is continually executing. Some of these main blocks are executed in parallel, while others are sequential. Steps 2 to 5 are sequential. After the first iteration (in which all steps are sequential), steps 1, 2 to 5, and 6 run in parallel. The complete block diagram is shown in figure 5.2.

Programming patterns used

The program's structure goes from left to right. In the VI of highest level (figure 5.2), an initilization takes place at left part, then, in central part, data is acquired and processed and, finally, at right closing takes place.

In this program the *Producer/Consumer design pattern* is used. This pattern is used to enhance data sharing between multiple loops running at different rates, as will be the case. One loop is the producer, the one that generates data, and another runnng in parallel will be the consumer. The reason for this parallelization is to optimize the program's performance. Communication between the loops is made using queues and shift registers. This will be explained in further detail in the following sections.

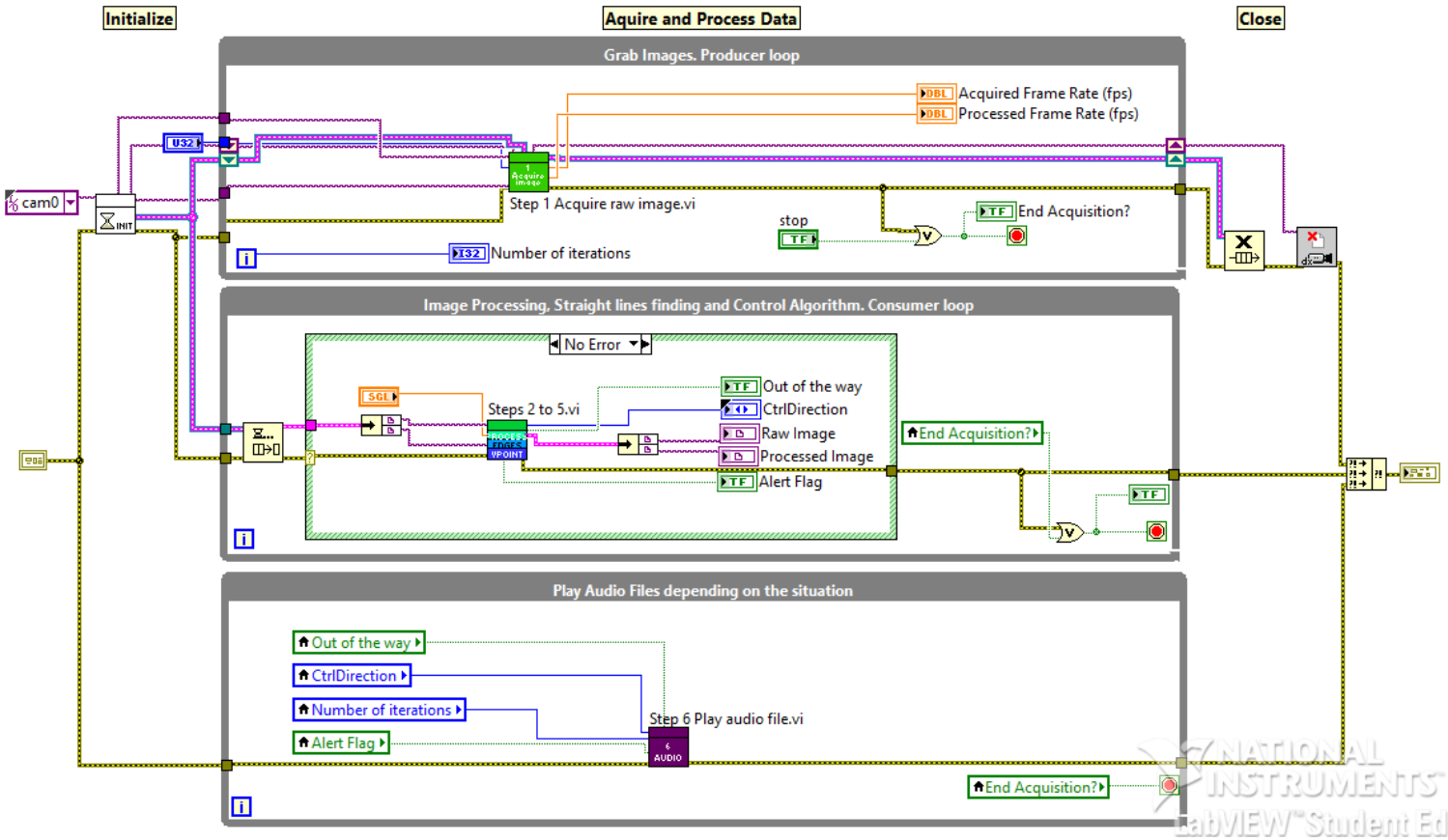


Figure 5.2: Main program's Block Diagram

5.1 Raw image acquisition

Figure 5.3: Image Acquisition's flowchart

This part is continually executing, according to the flowchart of figure 5.1. The image acquisition is programmed as follows: First the camera is initialized. In this inizialization main characteristics of the acquisition are defined. Here the program queries the camera for its capabilities, loads a camera configuration file and creates a unique reference to the camera. In this part also a queue and a buffer memory allocations are created. This buffer will contain both the raw and the processed images. LabVIEW code is shown in figure 5.4.

After the camera's inizialization takes place, and memory allocations are created, raw image acquisition takes place in the *Grab Images* loop. This is the producer loop. The Grab images loop, is shown in figure 5.5. Data to be transmitted between loops is a cluster, composed of two elements: raw and processed image. Here the raw image is acquired and enqueued while the memory allocation in buffer for the processed image remains empty in the first iteration.

Data queues are used to communicate data (in this case, images) between loops in the *Producer/Consumer design pattern* (producer loop as mentioned be-

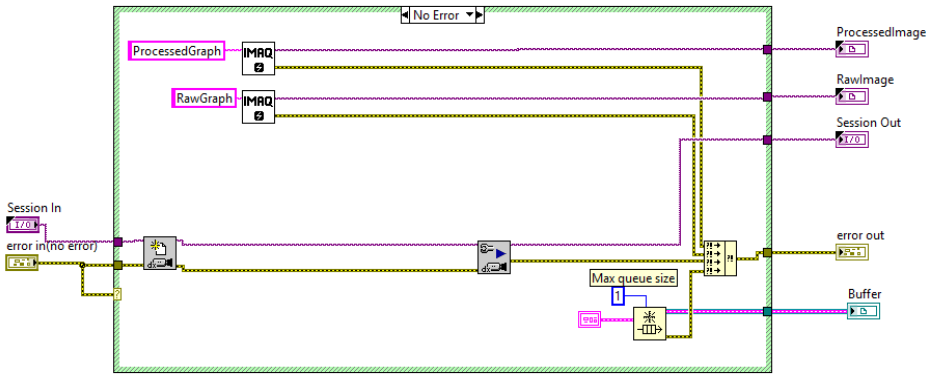


Figure 5.4: Camera's inizialization

fore is the *Grab Images* loop of figure 5.5, and consumer will be shown in next section). These queues offer the advantage of data buffering between producer and consumer loops. This queue works like a FIFO (First in, First Out) system, and is a circular queue, that means that data is being overwritten.

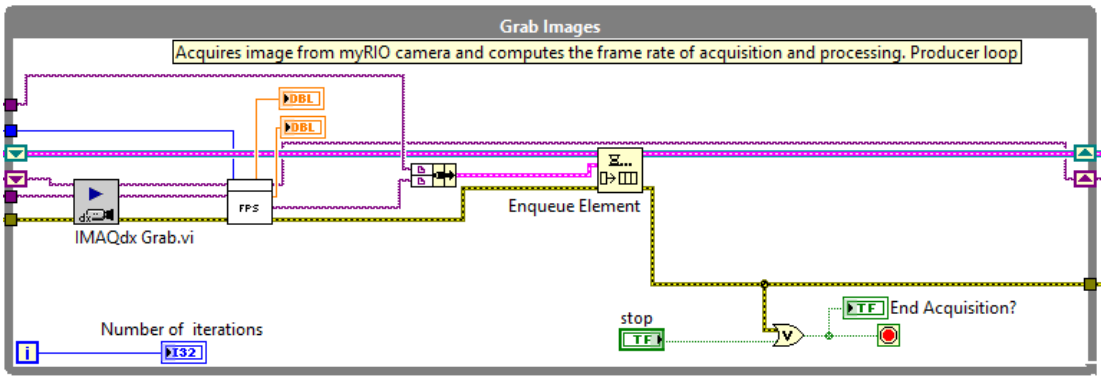


Figure 5.5: Grab Image Loop

In the Grab images loop, frames per second speed is calculated, both for the acquisition and the processing. As shown in the loop's borders (downward arrows) the program is using shift registers, in this way it is possible to get the fps speed of the processed images as well. Shift registers are useful for passing data from previous iterations throught the loop to next iterations.

The second loop (consumer loop) will consume raw image elements of the cluster that the queue sends it. This will be the loop that executes the image processing block.

5.2 Image processing

Flowchart of this section is as follows:

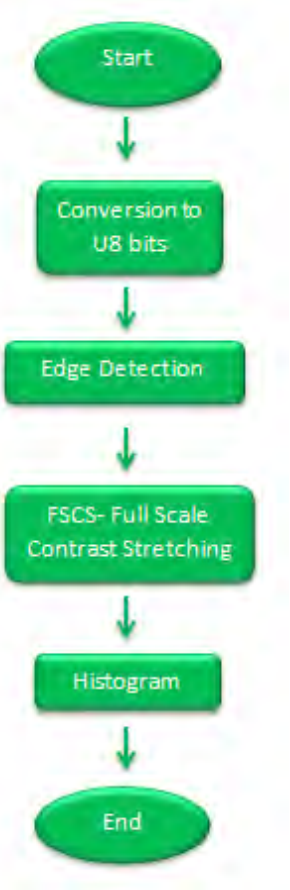


Figure 5.6: Image Processing's flowchart

First image format is changed from 32 bits-RGB image to U8-bit image. Consequently image's size also changes from 640x480 pixels to 480x480. This conversion is necessary for applying the edge detection method. In order to use some functions it is necessary that the image file is 8-bits. It is necessary to extract the edges in order to remove noise (aspects which are not important of the image) and leave just the relevant information. After edges of the image are extracted *full scale contrast stretching method (FSCS)* is applied to enhance these borders and a histogram is computed, in order to take into account a threshold value (explained later).

The VI performing image processing functions is named *jpEdgeDetect* and its design is shown in figure 5.7 . Inputs of this VI receive the following:
ImageSource In receives the image to process. In this case *ImageDestination In* is used for creating a copy of the image with a different file format: Unsigned 8

bits. It will be this copy the one that will be processed. It is necessary to create a different memory allocation for the processed image, otherwise raw image would be overwritten.

As detailed in flowchart after the processing is made, a histogram with a certain threshold value (this variable allows to control the minimum strength of an edge) is applied and the processed image is allocated in *ImageDestination Out*. In this image, edges whose gradient magnitudes are greater than the selected threshold will be shown. The original image is allocated in *ImageSource*.

An alternative in the image processing design was to apply some color filters, operations on image's pixels and contrast enhancement (instead of an algorithm for edge detection) for facilitating the searching of path borders. This was done and tested, as will be seen in chapter 6 with more details. As this alternative didn't work properly in most of the cases, even though the edge processing programming is further more complex, it was chosen.

There are some optional inputs in *jpImageProcessComputations* that can be modified, for example is possible to choose if FSCS method (explained in following subsection) will be applied or not, or which method will be used for the edge detection.

There are several ways of processing the image in order to get the edges present in it. The most of them (Canny, Sobel, Prewitt, Roberts and Singular Convolution Edge Detector) have been already mentioned in Chapter 2 (State of the art). Some of them, three, were programmed in LabVIEW. As their programming is very similar, it was decided to keep the three possibilities for image processing. Though there is a default case which is executed, it is possible and, easy to change, to one of the other two. These methods are explained in further detail in the next subsection.

There are also several alternatives for the point operation in the edge detection subblock. Point operation specifies the point operations to combine the horizontal and vertical gradient measurements to extract the edge strength at a point. In other words, the goal is to combine the x gradient (dx) and y gradients (dy) at each point, based on one of the following definitions:

- Eucledian distance: $\sqrt{dx^2 + dy^2}$
- Sum of absolute values: $(|dx| + |dy|)$
- $max(|dx|, |dy|)$

Even though one of the combination of gradients has been chosen to be executed by default (euclidean), it is possible to change it.

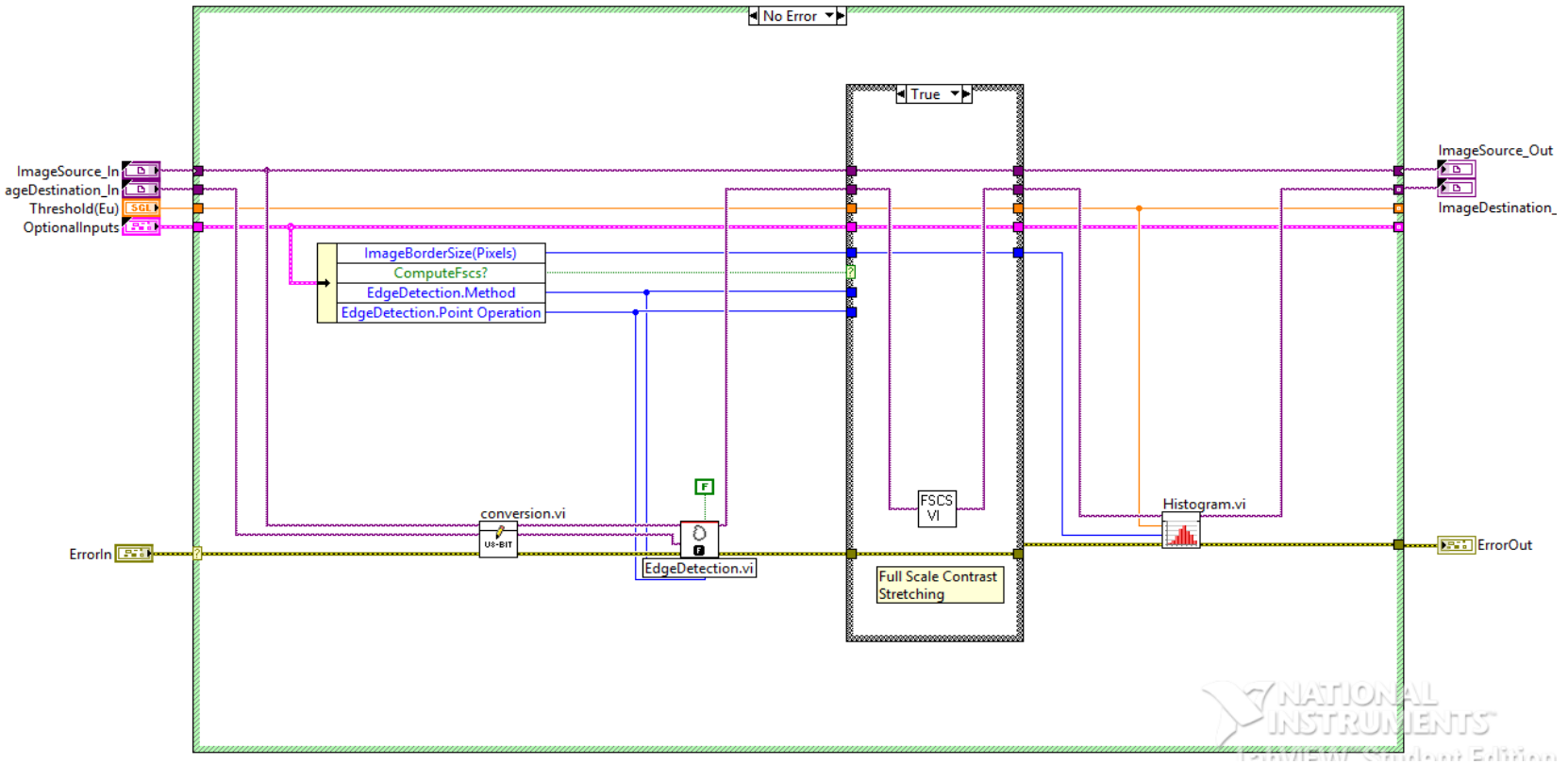


Figure 5.7: Image Processing's block diagram

5.2.1 Edge detection

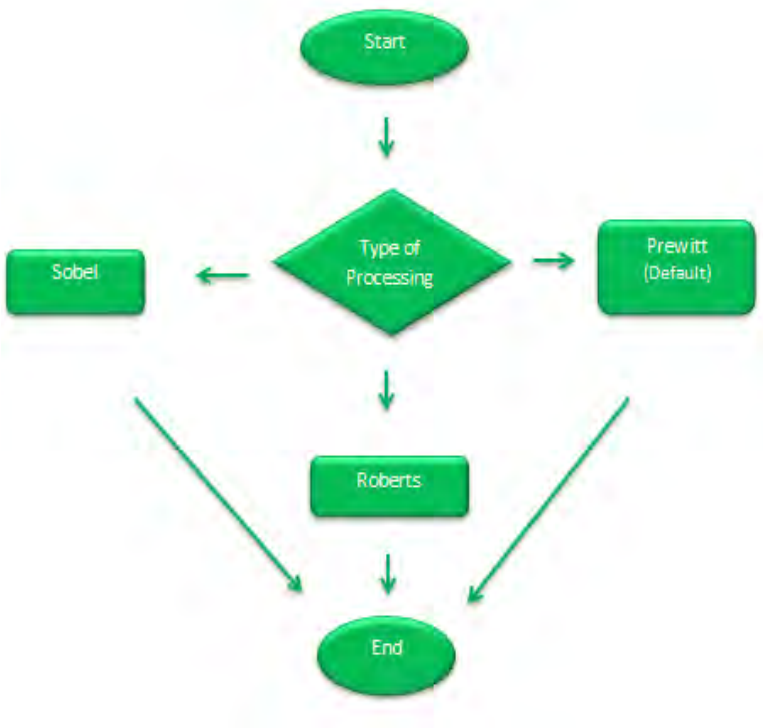


Figure 5.8: Edge Detection Flowchart

In this subsystem it is performed the edge detection algorithm. It is possible to choose one out of three different methods as shown in figure 5.8 The performance of the three types of edge filters is almost identical, with Prewitt and Sobel providing a little bit of smoothing to alleviate noise. However, the detected edges are thicker for the Prewitt and Sobel due to the smoothing operation.

This part is programmed using a *Stacked Sequence Structure*. A sequence structure contains several frames that execute in sequential order. Data dependency determines the execution order so the next frames only start execution after they have received all its inputs.

Figure 5.9 shows the VI performing *prewitt* method. Figures 5.10 and 5.11 show *roberts* and *sobel* methods respectively. In the left side of each figure is shown the VI inside its case structure, and in the right side of the image its corresponding implementation.

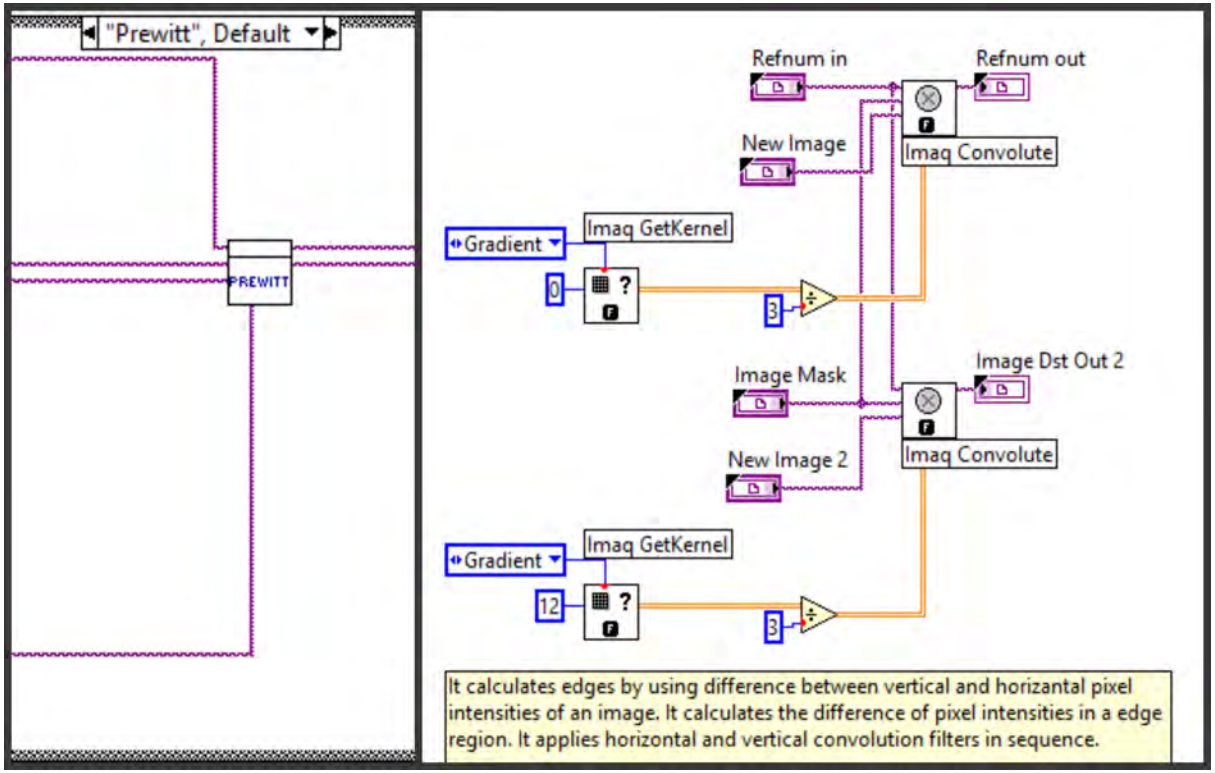


Figure 5.9: Prewitt edge detection method

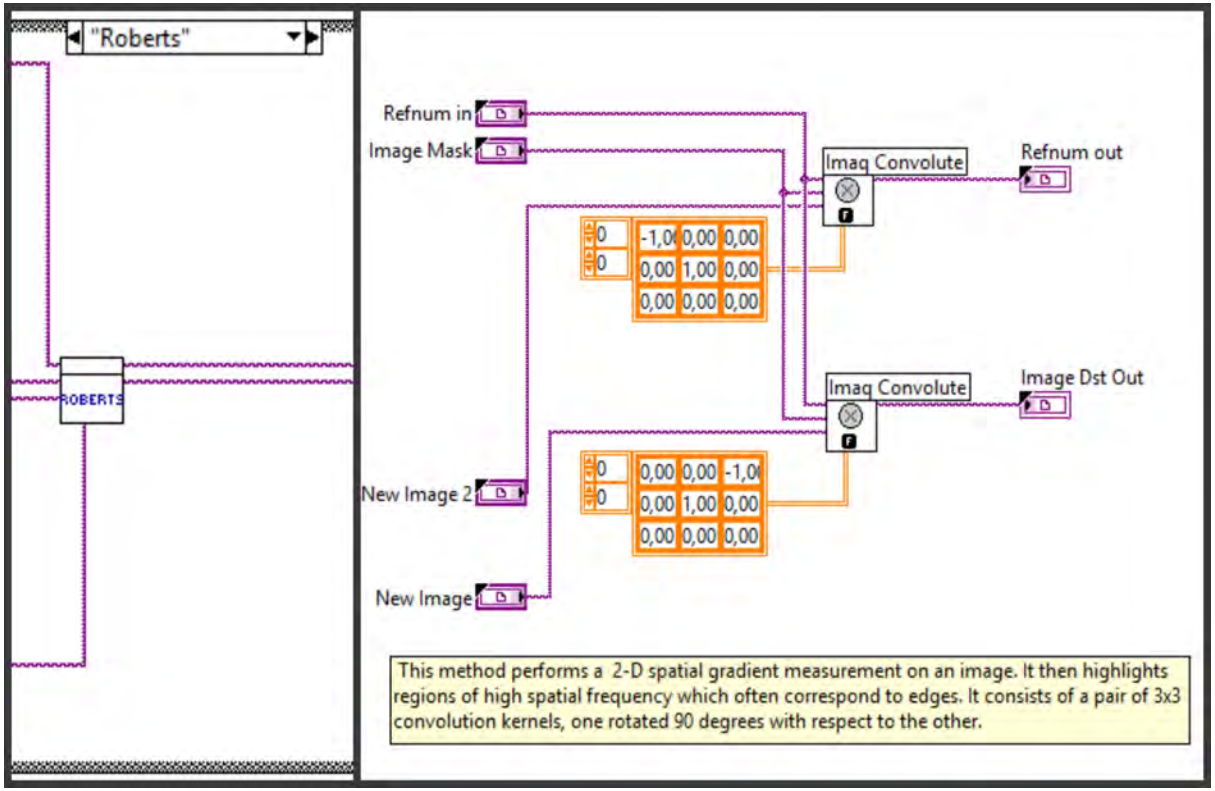


Figure 5.10: Roberts edge detection method

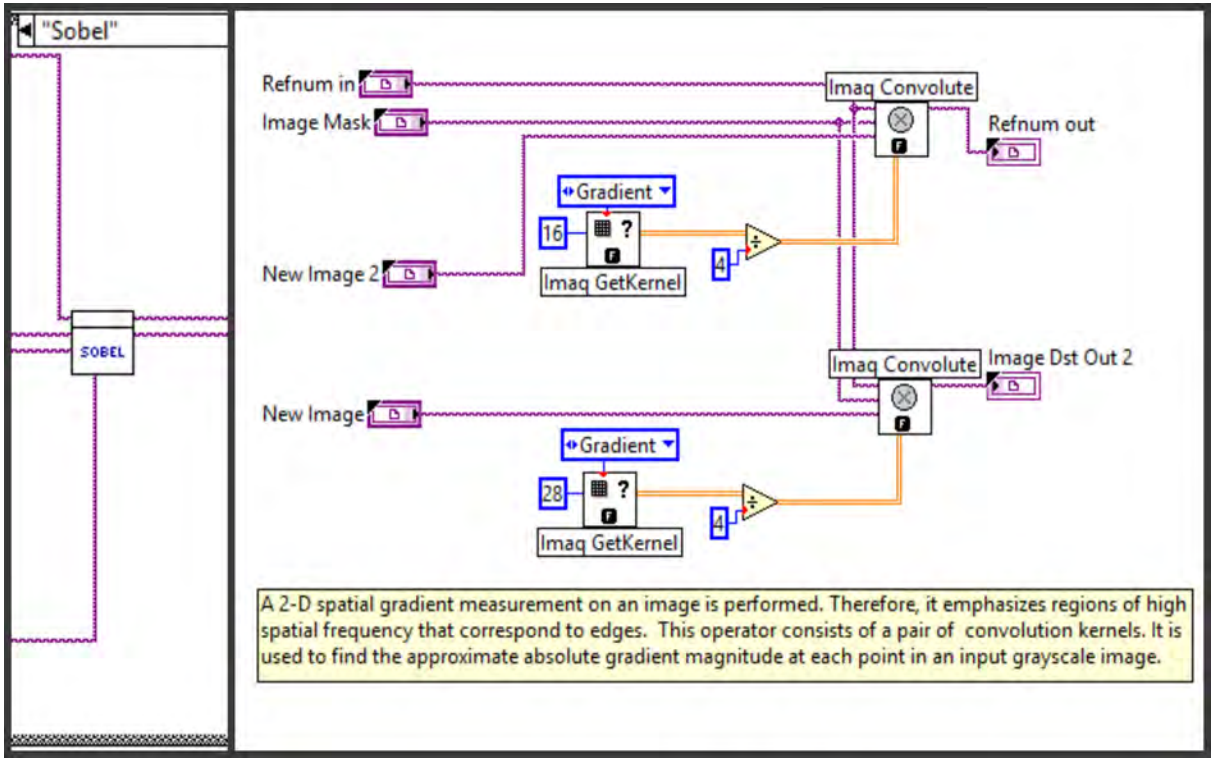


Figure 5.11: Sobel edge detection method

5.2.2 FSCS-Full Scale Contrast Stretching

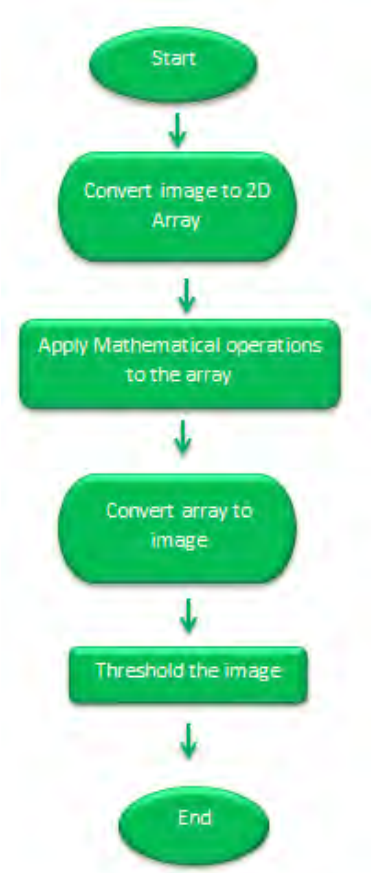


Figure 5.12: FSCS Flowchart

Contrast stretching, also called *Normalization* attempts to improve an image by spreading or stretching the range of intensity values the image contains. This is made in order to make full use of all values. In other words, it is a method for spreading out the grayscale values in an image.

It is used to improve the contrast in an image without distorting graylevel intensities too significantly. The contrast of an image can be defined as a measure of its dynamic range, or the 'spread' of its histogram. More details about this method can be found at reference [32]. In this program it is needed to improve image contrast as it allows to see the edges sharpest.

For 8-bit grayscale pictures, the limits over which intensity values are extended, are 0 and 255. The histogram of the image is examined to determine the intensity values and for each pixel, and the original intensity value is mapped to an output value. Given an original image $J(i, j)$ this method computes the

new images as $J(i,j) = P * I(i,j) + L$. The L is a constant, which just adds more brightness to the image. Multiplying an image by P increases the contrast. Implementation of it in LabVIEW can be seen at figure 5.13. More details about this formula and method can be checked at [33].

This method is actually applied twice in the program. The first time in the stacked sequence which performs the edge detection. The second time it executes in the calling VI which is the named *jpEdgeDetect* and whose diagram was shown in figure 5.7. The second execution is optional but recommended.

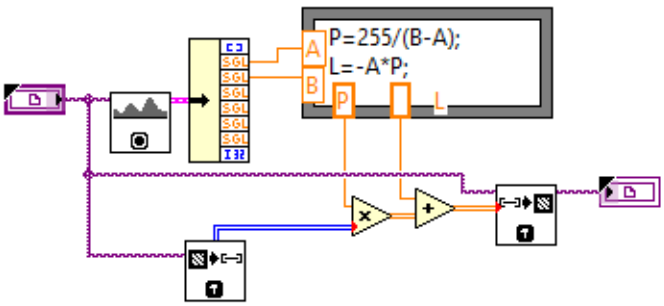


Figure 5.13: FSCS Block Diagram

In figure 5.14 it is possible to see the performance of the image processing block. Here both, edge detection (prewitt method) and FSCS method have taken place.

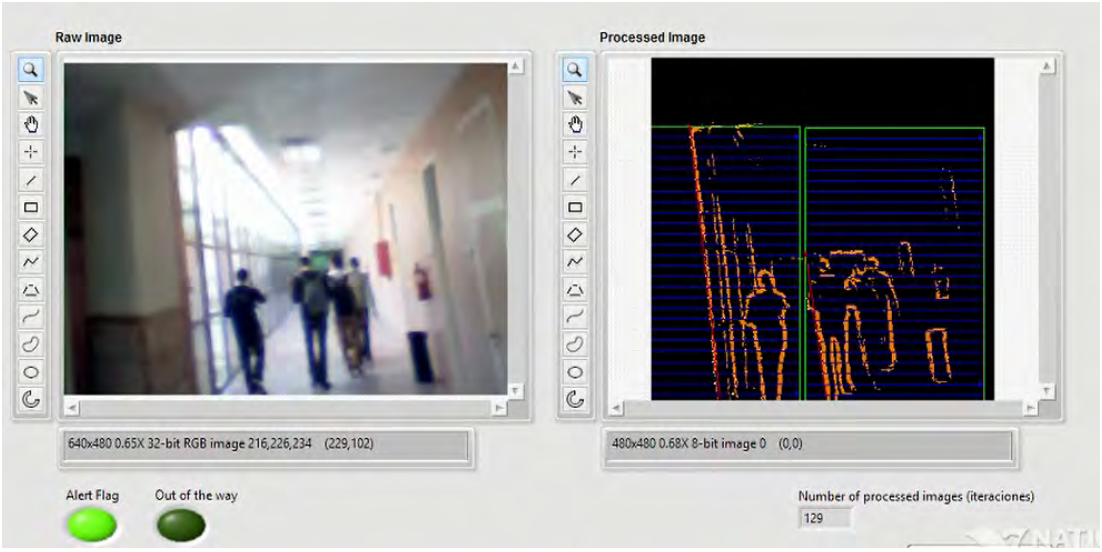


Figure 5.14: Example of the result of image processing

5.3 Straight line finding

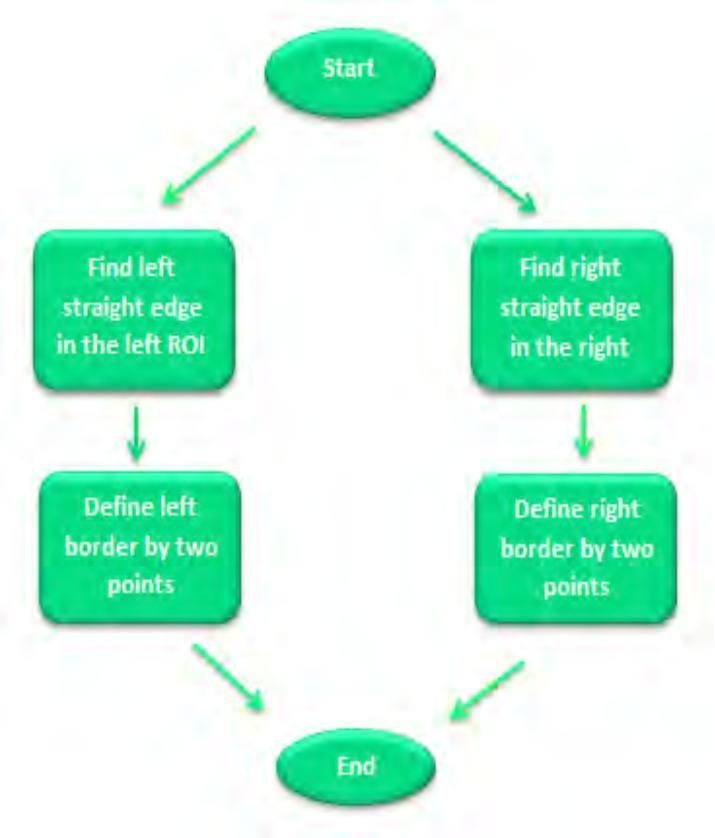


Figure 5.15: Straight Line Finding Flowchart

As shown in the flowchart of figure 5.15 the straight line finding is made using two regions of interest (ROIs). These two regions of interest appears as the two rectangles shown in figure 5.16.

It is in these two regions where the straight lines will be searched and found using the *IMAQ Find Edge VI*. In figure 5.19 inputs of this VI can be seen. A brief explanation of each will be made:

- *Options* is a cluster defining the parameters of the edge detection algorithm and the information that is overlaid on the result image:

  Search Direction specifies the direction in which the Region of Interest (ROI) is examined when straight edge detection is performed (left to right, top to bottom, etc).

  Edge Options specifies the parameters (edge polarity, minimum edge strenght, interpolation type, etc) that are used to compute the edge gradient information and detect the edges.

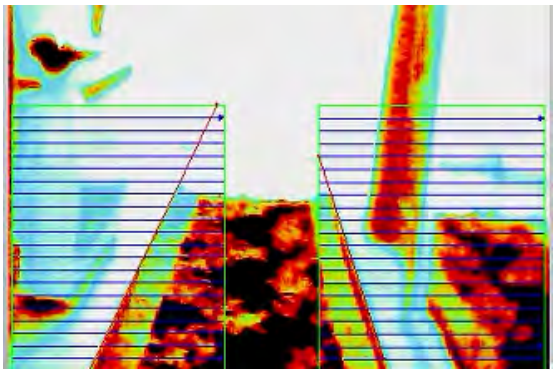


Figure 5.16: Regions of Interest (ROIs)

- *ROI Descriptor* is a descriptor that defines the rectangle or other four-side rectangular polygon within which the edge detection is performed.
- *Line Fit Options* specifies the options that are used to detect the straight edges. Some of them are:

  Type specifies the method used to find the straight edge. There are four possible ones: First Edge Rake (the one used in this project), Best Edge Rake, Hough Edge Rake, First Edge Projection and Best Edge Projection.

  Minimum Score specifies the minimum score of a detected straight edge (used 10).

  Angle Range specifies the positive and negative range around the orientation within which the straight edge is expected to be found (used 45 degrees).

The most important output of this function is the one named *Straight Edges*. It is a cluster that contains the following items:

- *Point1 (Pixels)* is the start point of the detected straight edge in pixel units.
- *Point2 (Pixels)* is the end point of the detected straight edge in pixel units. As with point 1 is possible to get this point in calibrated units but in this project only the coordinates in pixels are used.
- *Angle (Pixel)* is the angle the detected straight edge makes with the axis perpendicular to the search direction.
- *Score* is the score of the detected straight edge.
- *Straightness* is defined as the root mean squared error of the fitted line that represents the detected straight edge. A value of 0 indicates a perfectly straight line.

- *Average SNR (dB)* is the average signal-to-noise ratio along the detected straight edge.

After finding the straight edge, as the previously mentioned cluster of values, a line definition by two points, (each with x and y coordinates) is done. The code can be seen in figure 5.17 where two items (point 1 and point 2) of the *Straight Edges* cluster are extracted and form a new cluster of the line definition by two points.

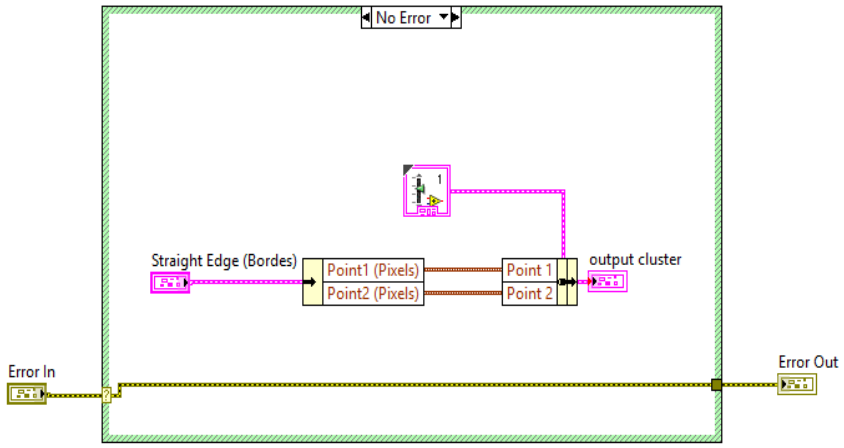


Figure 5.17: Line definition by two points

Figure 5.18 shows the implementation in LabVIEW of the above flowchart. It can be seen that blocks for searching left and right straight lines, delimiting the path, are set. Afterwards there is a line definition by two points for both lines. Figure 5.19 shows this same code in deeper detail.

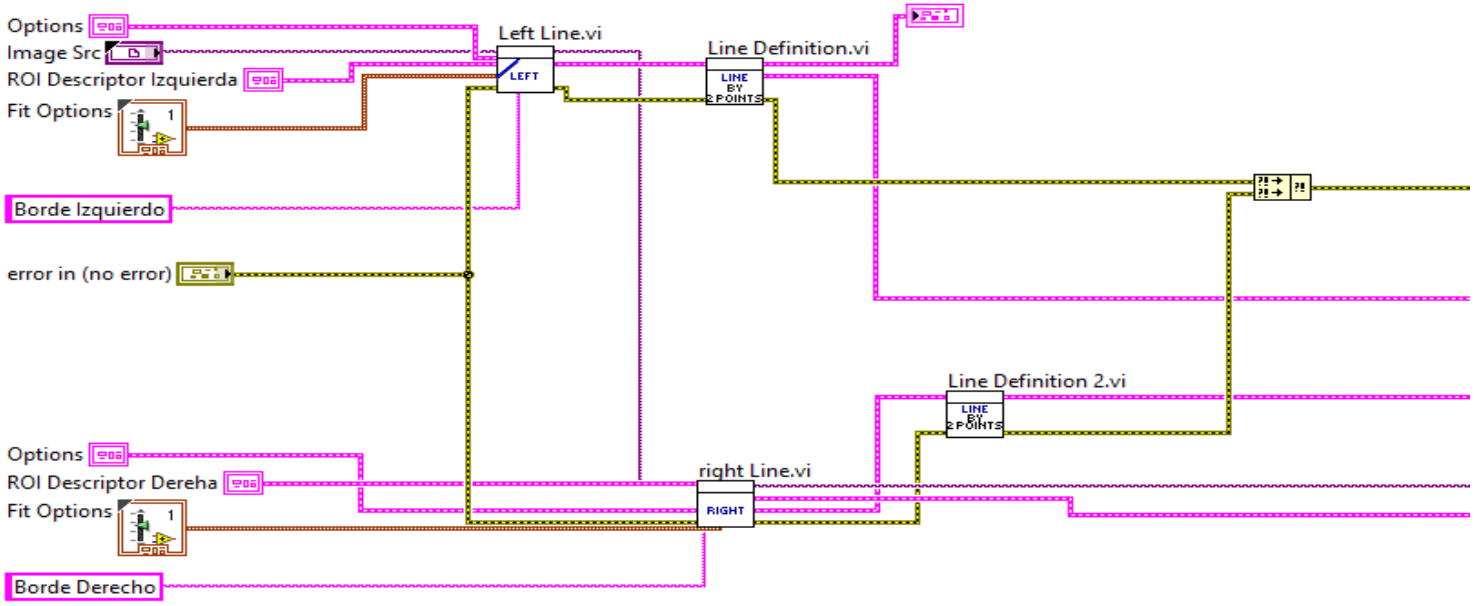


Figure 5.18: Straight line finding block diagram

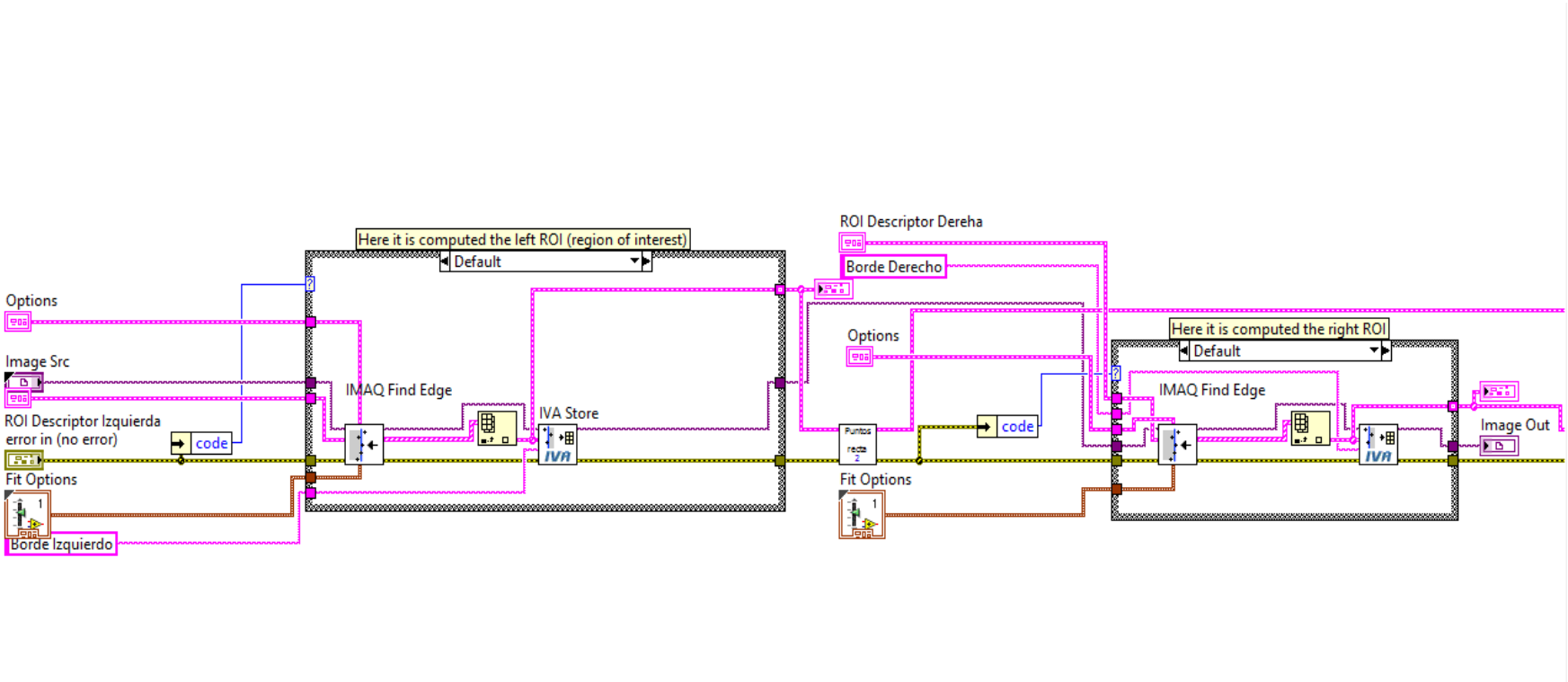


Figure 5.19: Straight line finding block diagram in detail

5.4 Vanishing point determination

The control algorithm is based on the position of the vanishing point, specifically on its x-coordinate.

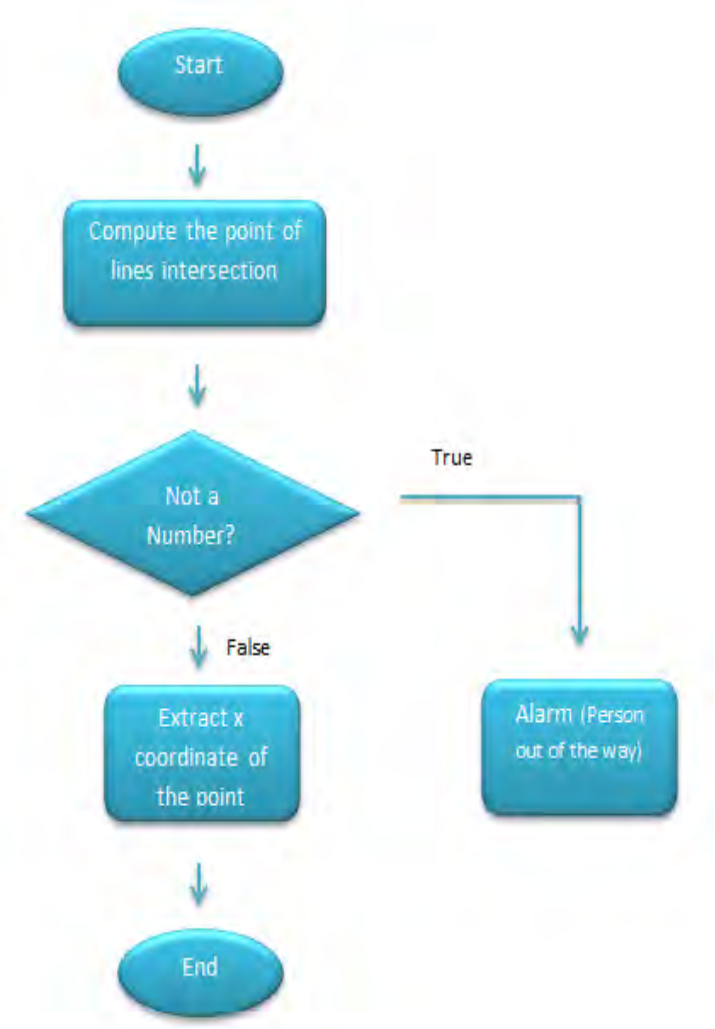


Figure 5.20: Vanishing Point Flowchart

For the computation of the crossing point of the two lines, *IMAQ Lines Intersection VI* is used. This VI computes the intersection point and the angle between two lines. Each line being specified by any two points that lie on it. The result is a point-coordinate cluster indicating the coordinates of the intersection point of both lines. *IMAQ Lines Intersection* code can be seen in figure 5.21.

After intersection point is got in the higher level VI, its x-coordinate is extracted. This value will be the base of the control algorithm, and given the fact that the webcam is placed in the head and assuming the person starts walking

inside the path, it should be near 240 pixel value. Y-coordinate of the point does not give us information related to how focused (on the center of the path) the person is.

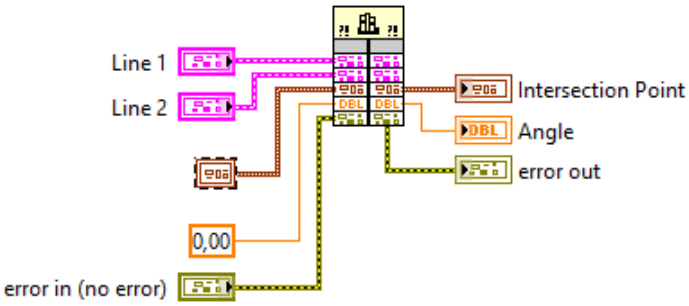


Figure 5.21: Lines Intersection Block Diagram

As shown in figure 5.20, if the result of the computation of the crossing point of the two lines is NaN (Not a number) then an alarm signal is turned on. If this happens it means that the person is out of the way, therefore this alarm signal will be used for playing an alarm sound that indicates it to the user. An example of this can be seen in figure 5.22. When the vanishing point is not found, the boolean named *Out of the way* turns on.

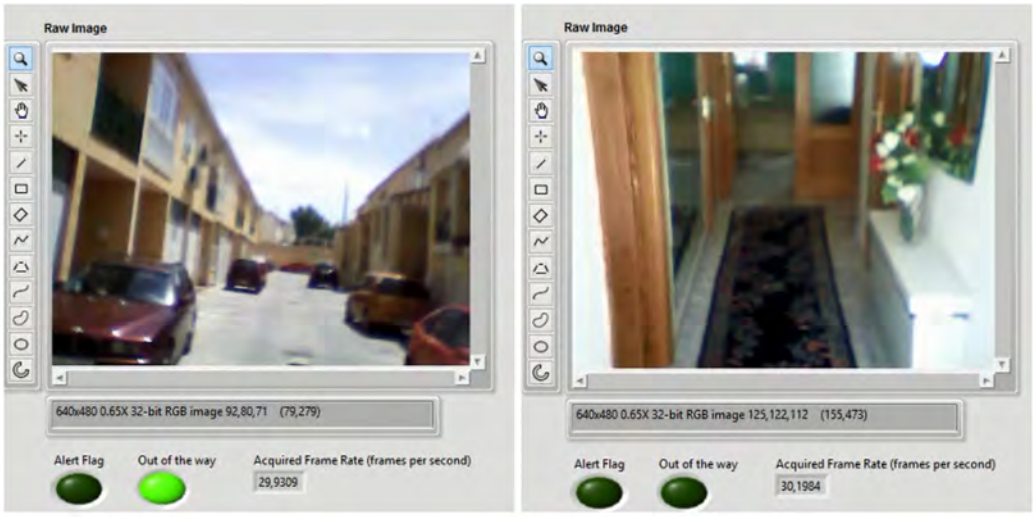


Figure 5.22: Out of the way alarm performance)

5.5 Control Algorithm

Once image has been processed, path borders extracted and their crossing point has been got, the control algorithm takes place. As mentioned in the beginning, in figure 5.1, these steps, from 2 to 5 are sequential. Step 5 (control algorithm) is the one that takes the guidance decision based on vanishing point position (in pixel values). Control algorithm has the following flowchart:

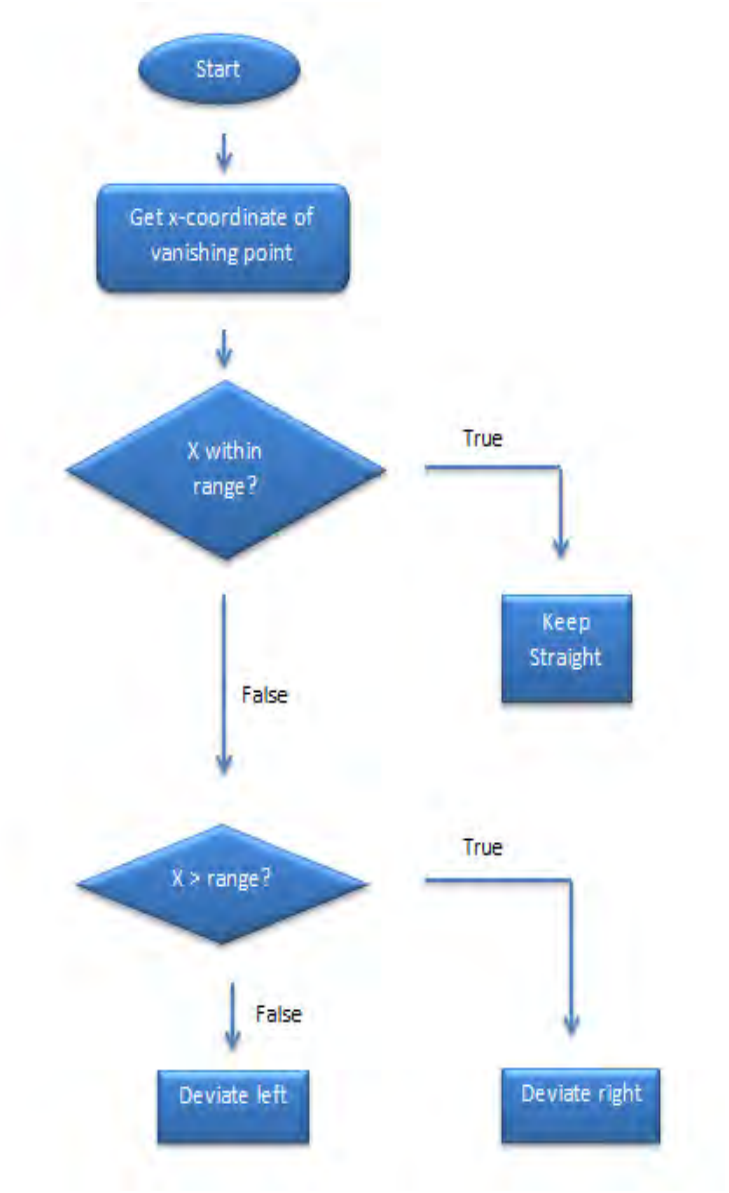


Figure 5.23: Control Algorithm Flowchart

Ranges defined in the pixel distribution

The U8 file format image has a pixel resolution of 480x480. According to it, in the ideal case in which the person is in the center of the way, the x-coordinate of the vanishing point should be 240 (pixel value). Based on the x-position of the vanishing point, the system will output three different decisions, as shown in figure 5.24.

Having this in mind three different cases are considered based on the urgency in which the person should act:

- **Correct:** VP position, (x) lies in the center of the image (pixel value = 240) with a margin error of 10 percent. In this case the person is walking in the right direction (case a).
- **Not urgent:** $24 < x < 216$ (case c) or $264 < x < 456$ (case b): In this case the person should correct the direction in which is walking but it is not critical.
- **Urgent:** $x < 240 - 0.9 * 240$ or $x > 240 + 0.9 * 240$: Here the person should deviate inmediately, otherwise he will leave the path. As it is a critical action, an alert flag is activated in this case.

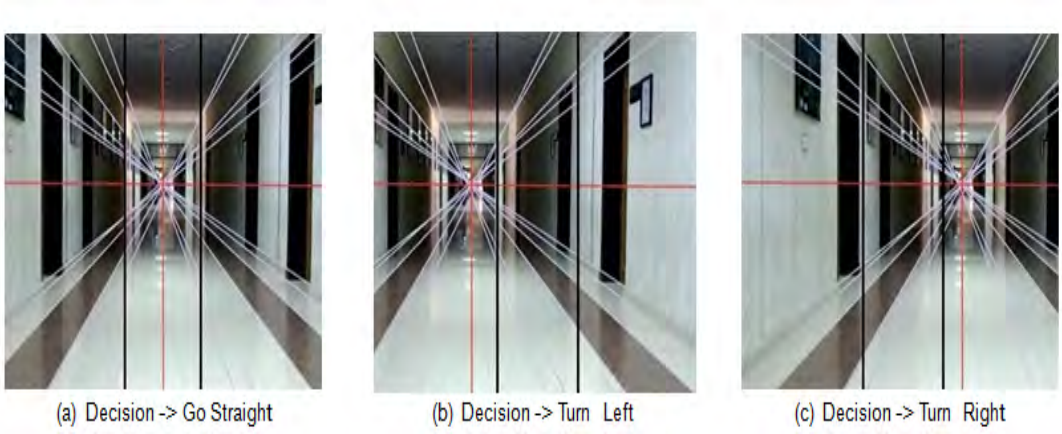


Figure 5.24: Possible outputs (guidance decisions)

The activation of an alert flag signal will affect the frequency at which audio files will be played. *Not urgent* case yields a faster audio playing than *Correct* case. *Urgent* case yields a faster audio playing than *Not urgent* case.

Implementation of control algorithm

Given the previously mentioned classification of the possible locations of the vanishing point, and having shown the flowchart (figure 5.23), now its implementation is detailed. Inputs and outputs of the *Control VI*, and the performed comparisons and computations, are explained. The corresponding block diagram can be seen at figure 5.27.

Algorithm's inputs are:

- x-coordinate of the vanishing point.
- Pixel value of the central point (in this case 240).
- Percentage for the allowed error (used 10).
- Percentage for an alert flag (90).

Algorithm's outputs are:

- *CtrlDirection*, which is an enumerated indicator with three possible values (right, left or straight)
- Two boolean variables, which are *Out of Way* and *Alert Flag*.

If not vanishing point has been found, *Out of Way* variable is assigned a *TRUE* value and no computations are performed. If an actual value of x-coordinate is got as input then two computations are done in parallel:

- x-coordinate is compared with the central image pixel value applied the allowed error percentage. If x value lies inside this range then the value of *CtrlDirection* will be 'straight'.
- x-coordinate is compared with with the central image pixel value applied the alert flag percentage. If x value lies inside this range then the value of *CtrlDirection* will be 'left' if x is greater than the upper limit of the range, and 'right if x is smaller than the lower limit of this range. If x value does not lie inside the range then *Alert Flag* boolean variable gets a true value.

In figure 5.25 it is shown the case in which the person is walking in the right direction, (as x-position of the vanishing point lies between the range $x > 240 - 0.1 * 240$ and $x < 240 + 0.1 * 240$). Therefore the boolean variable for alert is OFF. In this case the audio output is telling the person to keep straight.

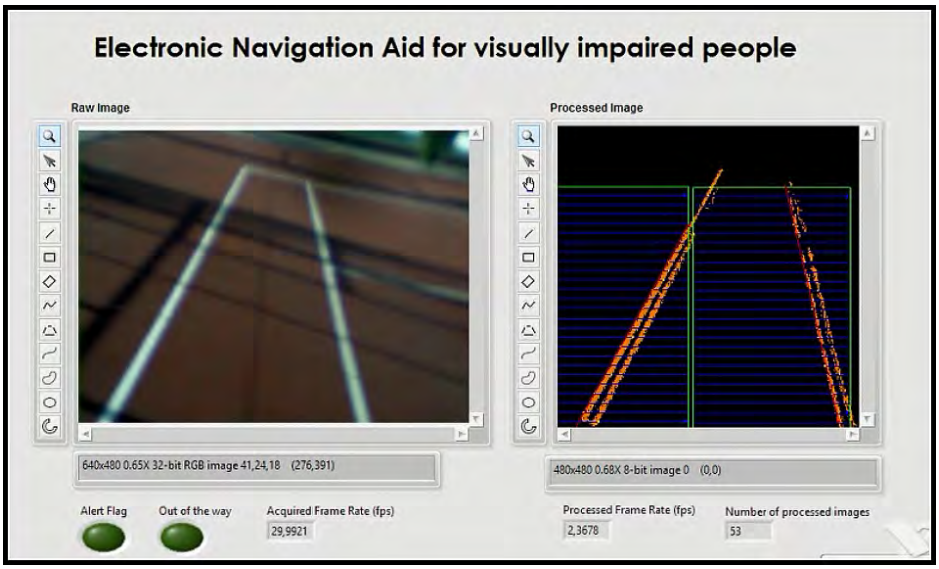


Figure 5.25: Alert Flag is OFF

The case in which the is about to leave the path (VP position lies in one of the ranges of the *Urgent* case) is shown in figure 5.26. Here the person should deviate right inmediately. X-position of vanishing point lies in the range $x > 240+0.9*240$ and therefore the boolean signal *Alert Flag* is ON. This will cause that the guidance output to be played played very often.

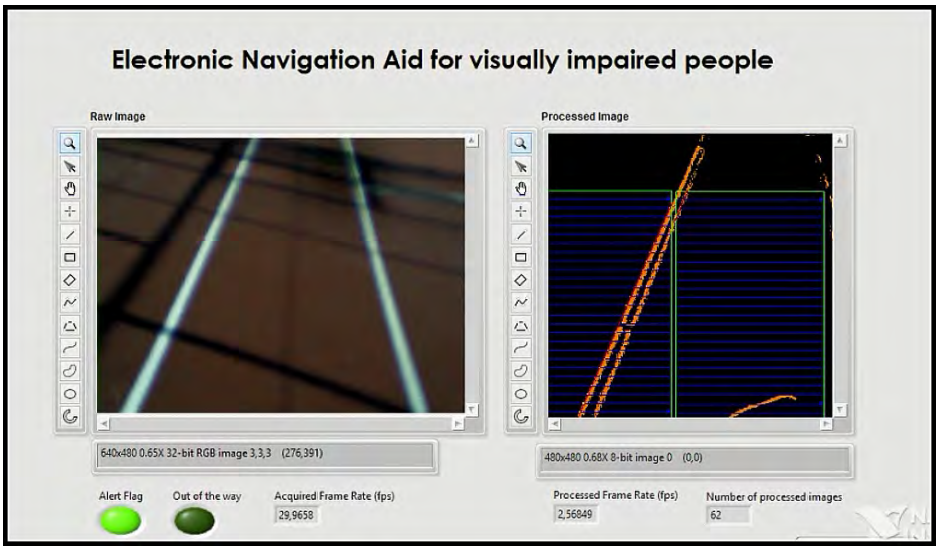


Figure 5.26: Alert Flag is ON

As previously mentioned, figure 5.27 shows the LabVIEW implementation of the control algorithm, with the explained inputs, outputs, comparisons, and computations.

In figure 5.28 is shown the block diagram of the calling VI, named *jpMainVisionProcessComputations*. This VI groups the image processing and the control computations. In this figure it can be seen how x-coordinate of the vanihising point is extracted, and this value is being received afterwards by the *Control Algorithm VI*.

The*CtrlDirection* enumerated control and the boolean variables are used as an inputs for the audio section.These variables are passed as local variables to the Audio VI.

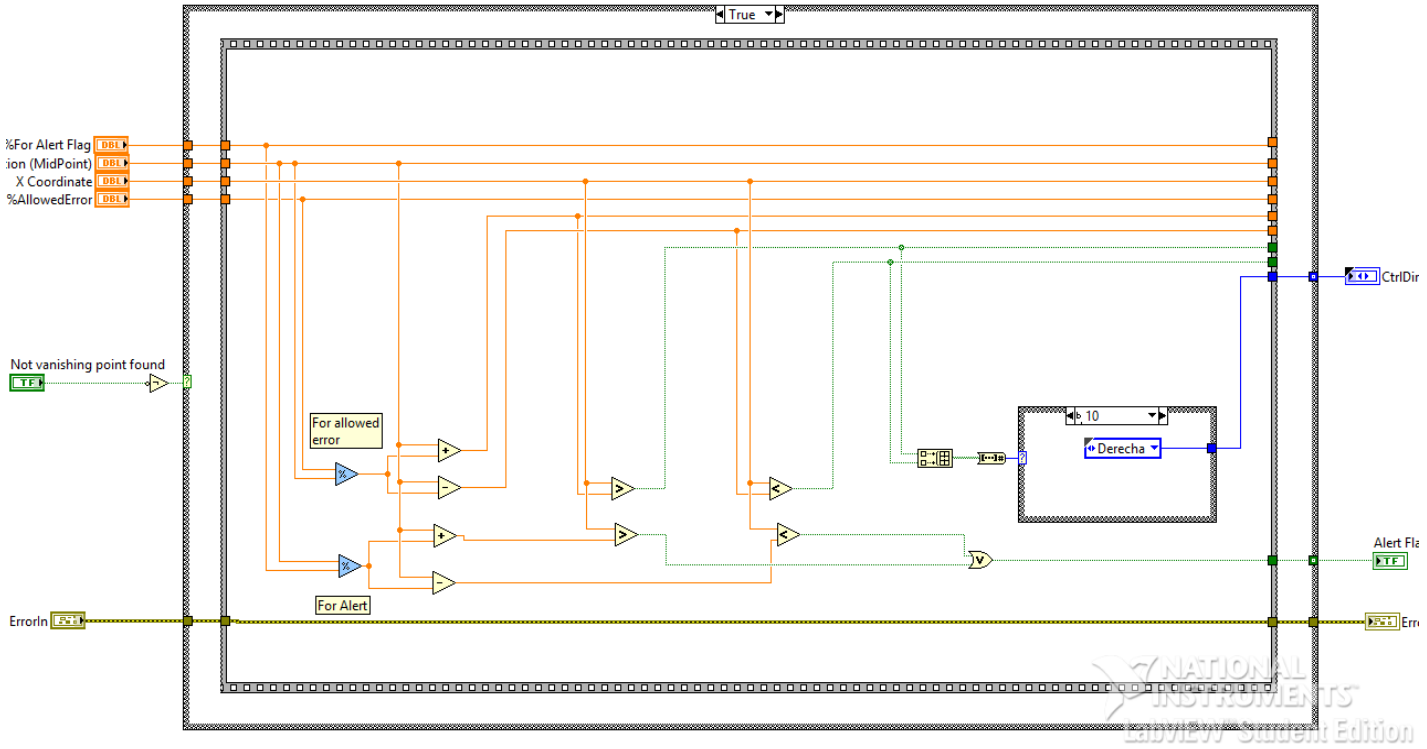


Figure 5.27: Control Algorithm Block Diagram

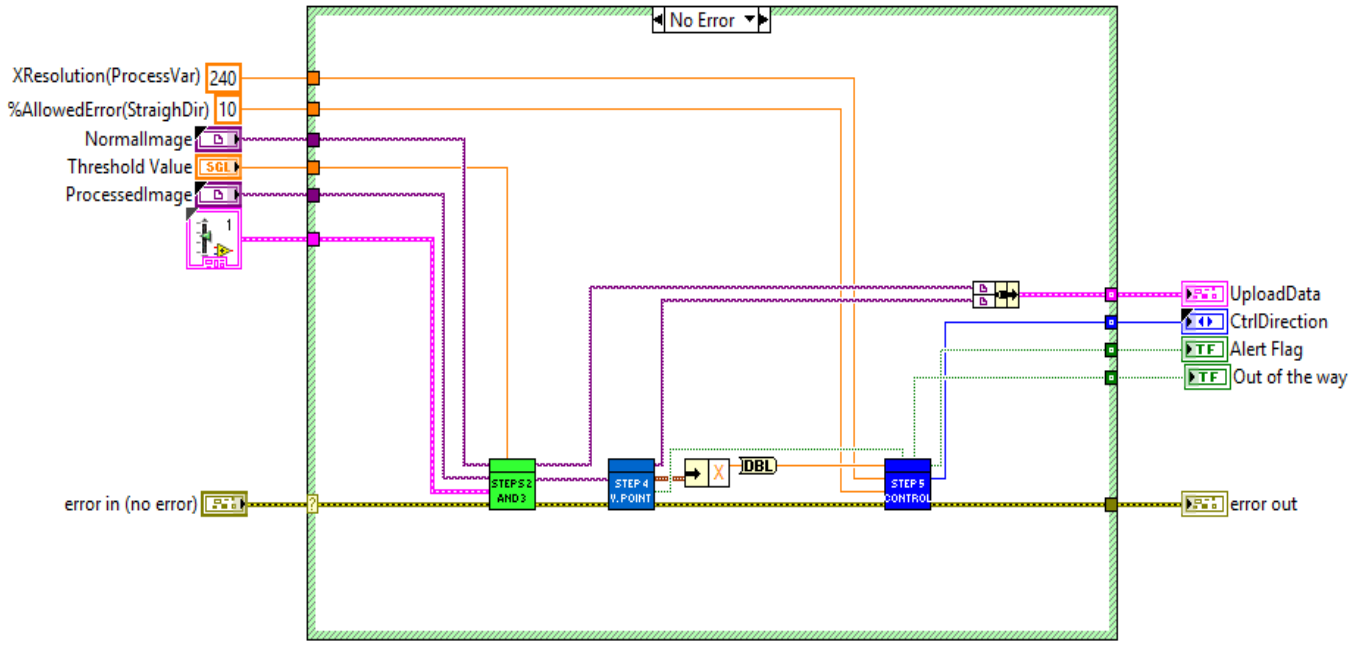


Figure 5.28: Vision and Control Block Diagram

5.6 Audio Output

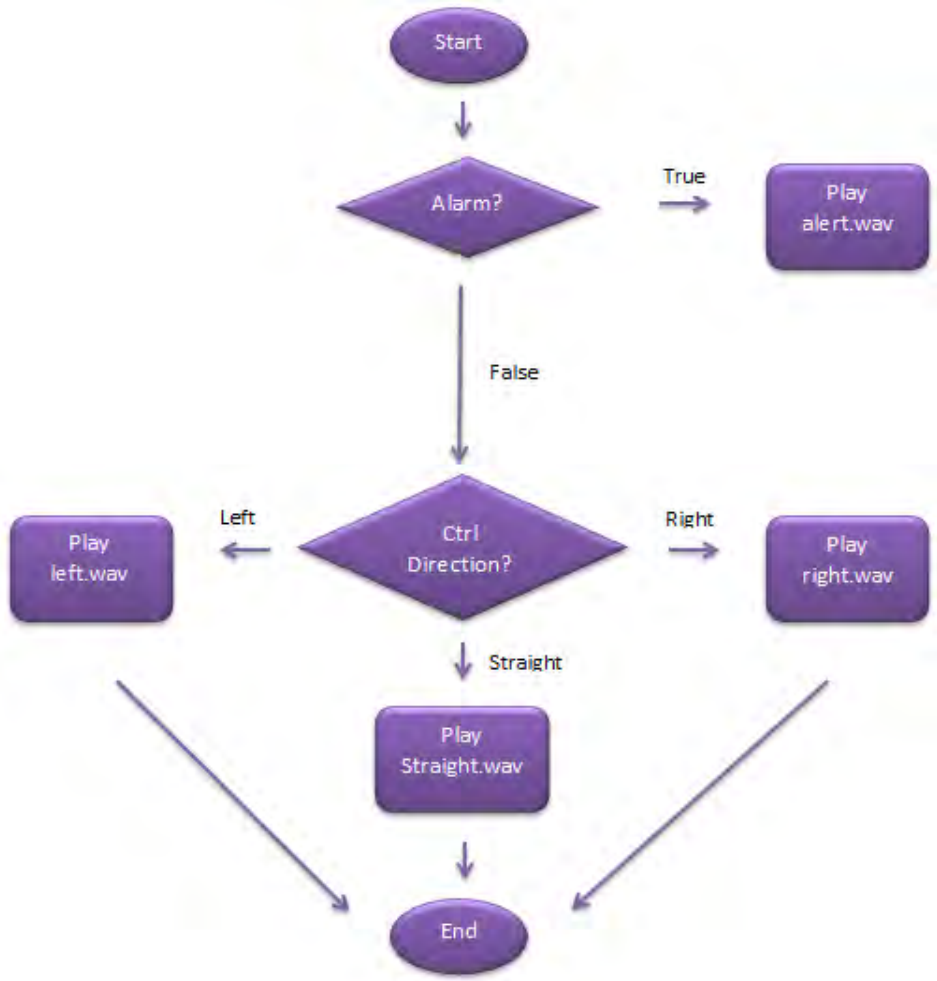


Figure 5.29: Audio Flowchart

Even though the flowchart shown in figure 5.29 does not take into account how often the audio feedback is received in each case, this frequency is not always the same. It depends on the severity of the action to be taken. The 'keep straight' guidance shouldn't be played as often as the ones advising to correct the path. Also it is not the same to be slightly deviated from the center of the path than to be about to leave it. These different rates at which audios are played don't affect the execution of other blocks of the program.

As previously mentioned, this VI takes as inputs the following local variables: enumerated variable *CtrlDirection*, integer *Number of iterations* and booleans *Alert Flag* and *Out of the way*.

A cluster of times was created in such a way that it is possible to change the frequencies for playing the audio files for each case, as it can be seen in the front panel of this VI in figure 5.30.

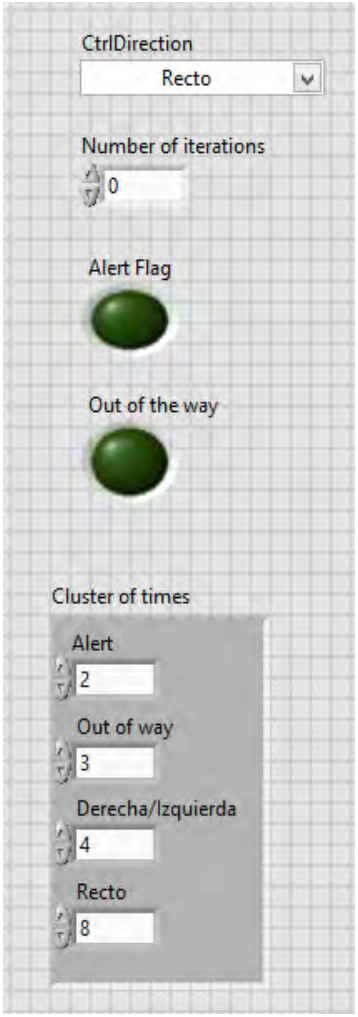


Figure 5.30: Audio Front Panel

The Block diagram of this VI is shown in the following figures 5.31, 5.32, and 5.33. Each image shows a different situation and it is shown how often the voice record is being played in each case:

Out of the way

In this case the alarm sound is played each 3 iterations of the *producer loop*. This sound indicates the user that he has left the path and should ask for help to come back to it. As this can be critical the frequency at which it is played is high. See figure 5.31.

Alert Flag is ON

Here the person should deviate soon otherwise he will leave the path (vanishing point position is out of what is considered safe). The *CtrlDirection* can only have two values: right or left. If we are in this case the waveform files advising the person to deviate left or right are played each 2 iterations. Person should correct his trajectory soon. Block diagram for this case is shown in figure 5.32

Alert Flag is OFF

Here the person is walking in the right direction (vanishing point position is within the allowed error margin). As *CtrlDirection* is straight then the audio file is played each 8 iterations, as shown in figure 5.33. If *CtrlDirection* is left or right then the audio file is played each 4 iterations, so the person can tune his trajectory.

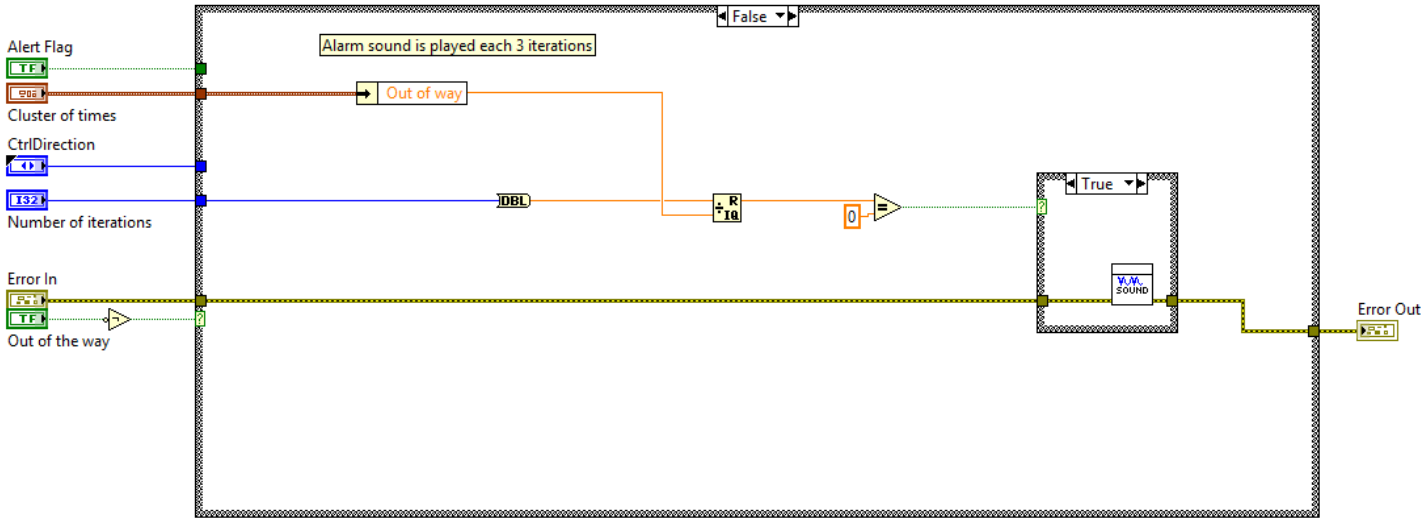


Figure 5.31: Audio VI: *Out of the way* case

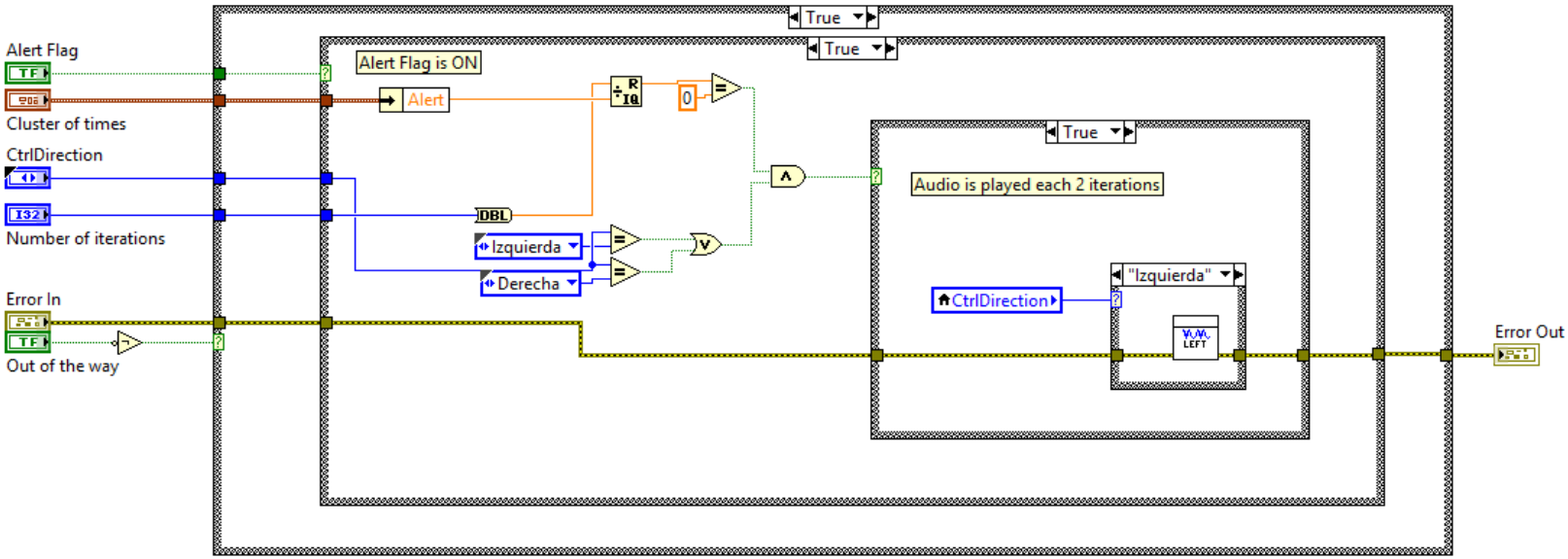


Figure 5.32: Audio VI: *Alert Flag ON* case

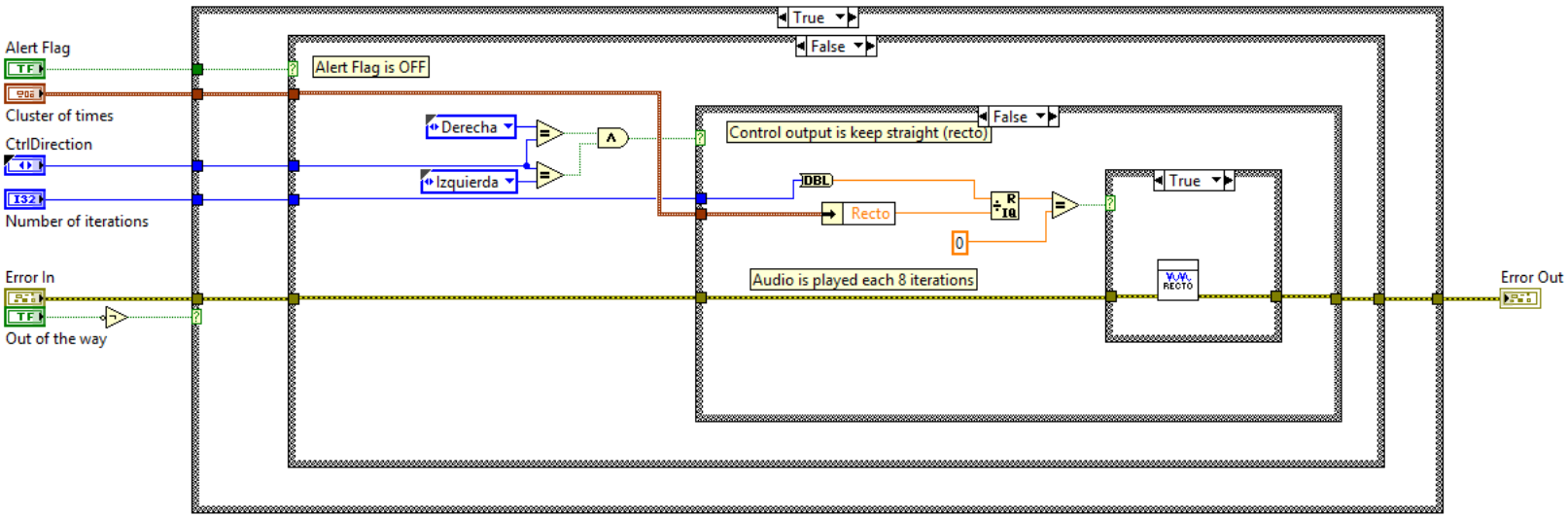


Figure 5.33: Audio VI: *Alert Flag OFF* case (keep straight)

5.6.1 Audio file transfer protocol

Audio files have to be transferred to the myRIO device, and have to be stored within it. For this *FTP (File Transfer Protocol)* was used. Its UI (User Interface) can be seen at figure 5.34.

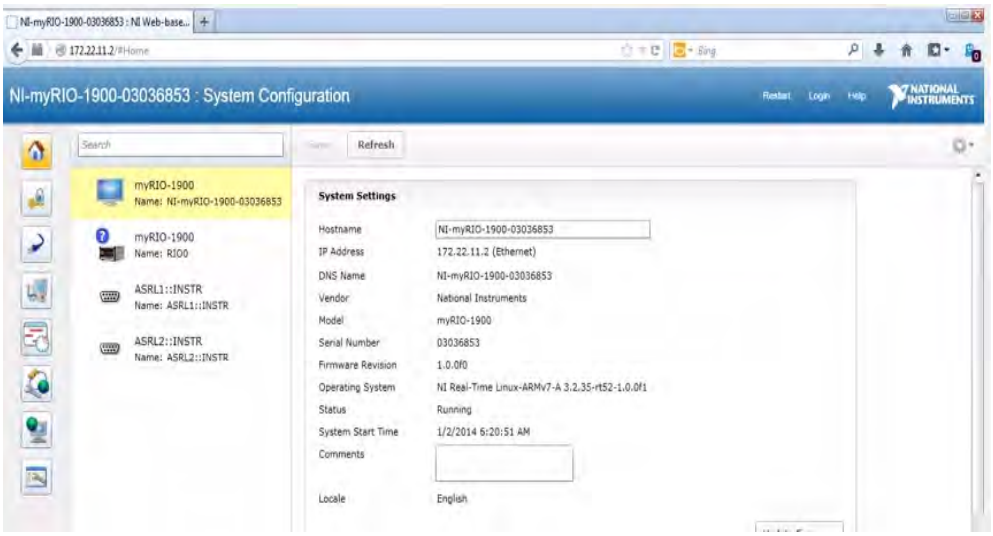


Figure 5.34: Transfer of files to myRIO

Waveform files are copied into one of the folders (sbin folder) contained in myRIO, so the device can find them during execution. The concrete path is */usr/local/sbin*. It is important to copy the files in this exact path, otherwise they won't be found. These files are voice records and an alarm sound. The voice records are three: one telling 'keep straight', another telling 'deviate right' and another telling 'deviate left'. More details of the use of *File Transfer Protocol (FTP)* can be found at [34].

Other method to transfer files to a real-time device is *WebDAV*, a protocol that enables to copy files on the target. It is an extension of HTTP and is supported on Windows and Linux. For using it, it is necessary to have *WebDAV* installed in the target and to configure *WebDAV* in the development computer. More details of this protocol and its use can be found at [35].

The audio files transfer to myRIO is important because this is not done automatically when deploying the code on myRIO. Audio files are resources that have to be present in the device when the code is executed on it.

5.6.2 Audio files processing

It was neccesary to cut the irrelevant parts of the audio waveform files because there is a limitation in the amount of points that admited the memory that send the waveform to the FPGA. The number of samples can be as maximum 10000 (see 'Sample Rate' and 'Values' at [36]). This wasn't trivial and it was hard to get the playback of the files, as it wasn't known that it needed a previous processing.

These waveform files were processed using *Audacity* software [37]. The user interface of this software is shown in figure 5.35. With this software the frequency of the sound wave was decreased to 8kHz and the silence's gaps of the wave were cut. As the waveform files were voice recordings, there were small silence's gaps specially at the beginning and at the end of the record. With *Audacity* software these gaps were easily removed.

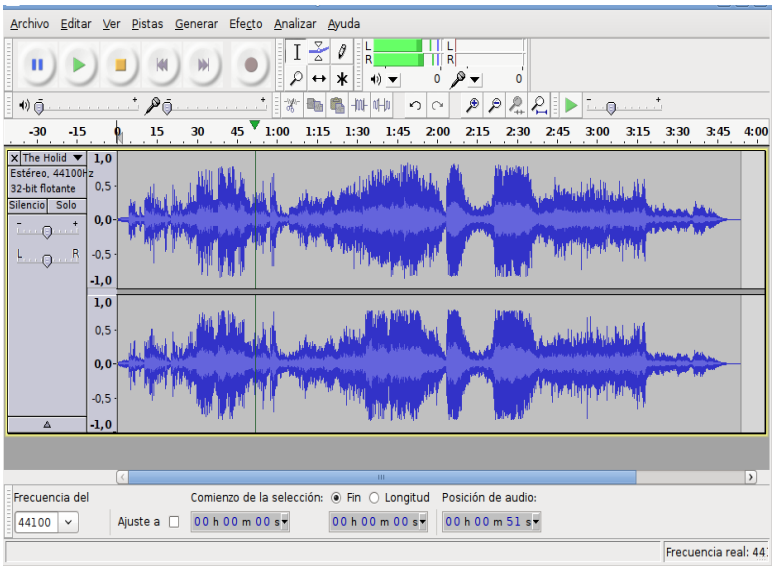


Figure 5.35: Processing Audio Waveforms

FPGA personalities consist of predefined FPGA bitfiles for programming the myRIO. The *LabVIEW myRIO Toolkit* provides the following FPGA personalities: *Default*(for control applications) and *High Throughput*(for audio signals and projects in need of waveform data). As the last personality supports high-speed analog data access, *high-throughput toolkit* was used in this case.

About the output sampling frequency, myRIO with the default defined personality can work with frequency values between 1KHz and 80KHz. If FPGA is programmed instead of using the default personality then output can be used till 345KS/s in all analog output channels (see page 22 of manual [28]).

5.6.3 Playback of audio files

Flocwchart for the playback block is shown in figure 5.36.

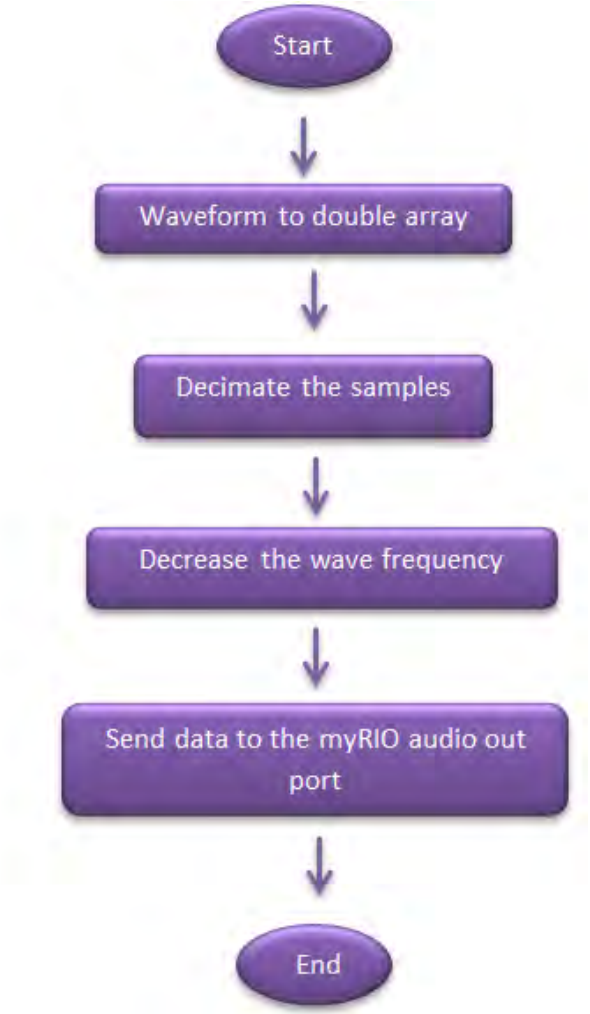


Figure 5.36: Audio Playback Flowchart

- Firstly a *waveform to doubles* function is used in order to modify the values of the wave and its sample rate.
- Secondly, waveform is decimated in order to have less samples. A control is used for selecting which waveform file will be played (*Wave file path*), and another one for selecting the factor at which the waveform will be decimated. The default decimating value is x1. For decimation a *Decimate 1D Array* function is used. This function divides the elements of array into output arrays, placing elements into the outputs successively. In this case

only one of this output arrays is saved and the other is lost. In this way, only half of the samples are used for playback. In this part the sample rate (8 kHz) is divided by 2.

- Thirdly frequency of the wave is checked (it shouldn't be greater than 10kHz). It is just a preventive action taken to avoid errors.
- Finally the result waveform is sent to the audio out port of myRIO.

A block diagram showing the described implementation for audio files playback is shown in figure 5.37.

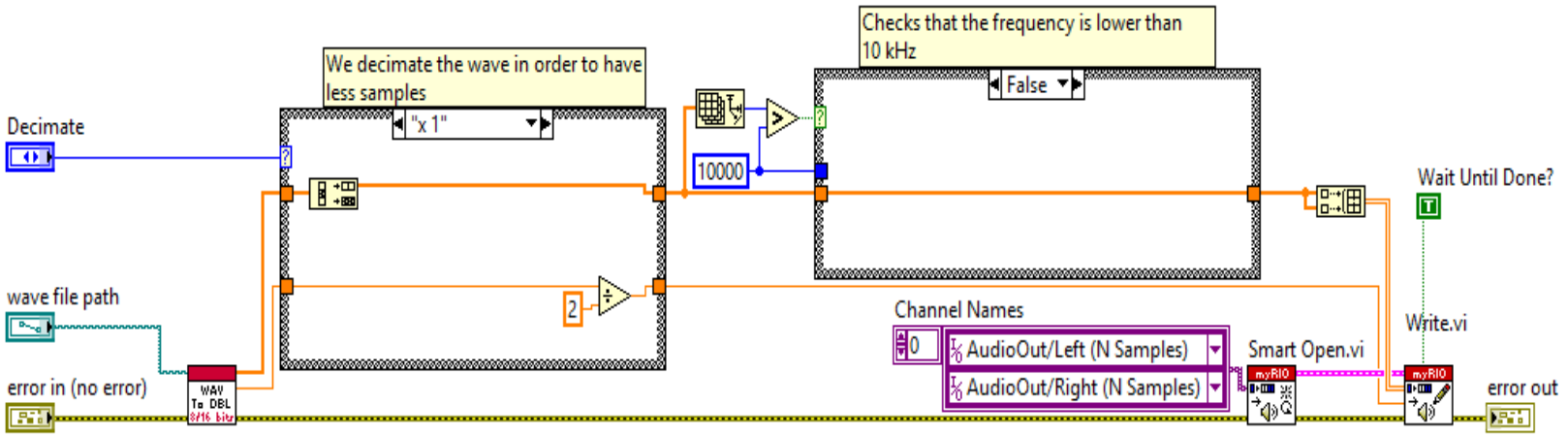


Figure 5.37: Block Diagram for Audio Playback VI

5.7 Complete Block Diagram

In this section an integration of all the blocks previously described will be made. The complete block diagram of the program has been shown at the beginning of the chapter, but in order to make things clearer is shown again in figure 5.38.

In the block diagram it is easier to see which elements run in parallel and which are sequential. Relating to main flowchart (shown in figure 5.1), and as previously mentioned, steps 2 to 5 are sequential and grouped in *Steps 2 to 5 VI*.

After the first iteration, steps 1, 2 to 5, and 6 run in parallel. In figure 5.38 is possible to see that there are three *While loops* in parallel. The first two belong to the previously mentioned *Producer/Consumer Pattern*. The producer loop gets the images being captured by the camera and the consumer loop process them and applies control computations. Iteration rate of second loop (as it contains all the processing and computations) is slowlier. Communication between them is performed using queues and shift registers.

The third loop running in parallel is used for playing the audio files corresponding to each situation. Here the communication, needed in order to get the *Play audio file VI*'s inputs, is performed using local variables. This parallelization gives agility to the program and, as a consequence, better performance.

It is also possible to see that, first (at left) an inizialization of the program takes place. In the central part, data is acquired and processed, and at right we have the closing part. This structure left to right demonstrates one of the main LabVIEW's characteristics: that it is a dataflow programming language. It executes from left to right and allows parallel execution of loops, as it is possible to have inputs for several VIs and functions at the same time.

Another detail worthy to comment is that once the *Stop* button is pressed, *Producer while loop* is stopped, and then the other loops are stopped as well as there is a local varible of the boolean *End Acquisition* connected to their stop terminal. It can also be seen that, if there is an error in whatever part of the program, it is transmitted through the error cluster, and it will eventually stop the execution totally.

Therefore, it can be said that parallelization is an improvement on the system that has been possible thanks to LabVIEW software, and as was mentioned in chapter 3, this was one of the main reasons for choosing the myRIO device.

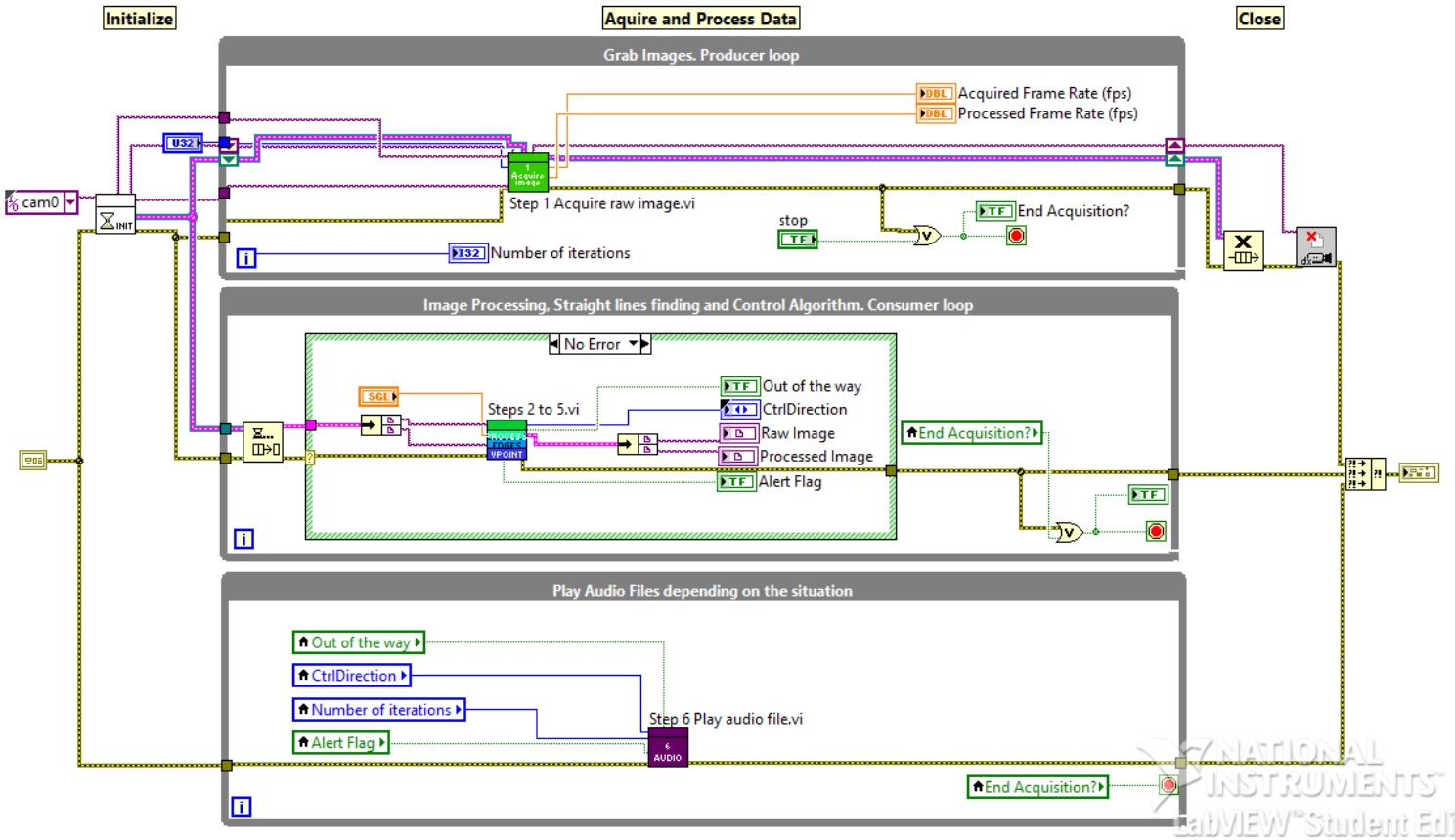


Figure 5.38: Complete Block Diagram (after breakdown)

Chapter 6

Prototype's Tests

In this chapter a report of the different experiments and tests made will be presented. In the first section there will be a historical presentation of the verifications performed with the corresponding extracted conclusions. In the second section it will be possible to see how a person wearing the prototype would look. Finally in the last section, results extracted from the tests are summarized and feedback given from a blind person after using the prototype will be mentioned.

6.1 Performed verifications

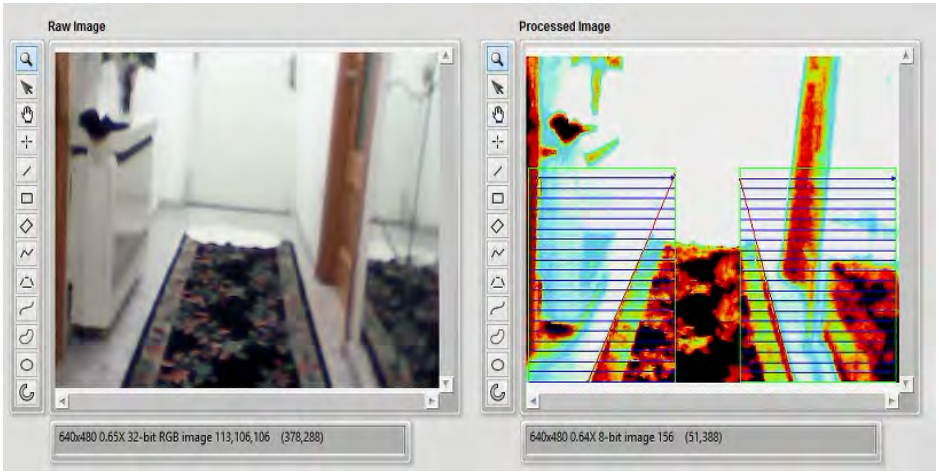


Figure 6.1: Experiment 1

During the development of the prototype it wasn't very clear if the image processing section was needed, specifically the edge detection processes. This arises from seeing that the straight line finding VI was able to find the edges delimiting the path in some cases, as can be seen in figure 6.1.

Some tests were made using the program explained in the previous chapter but without edge detection block, (using just a fast image processing).

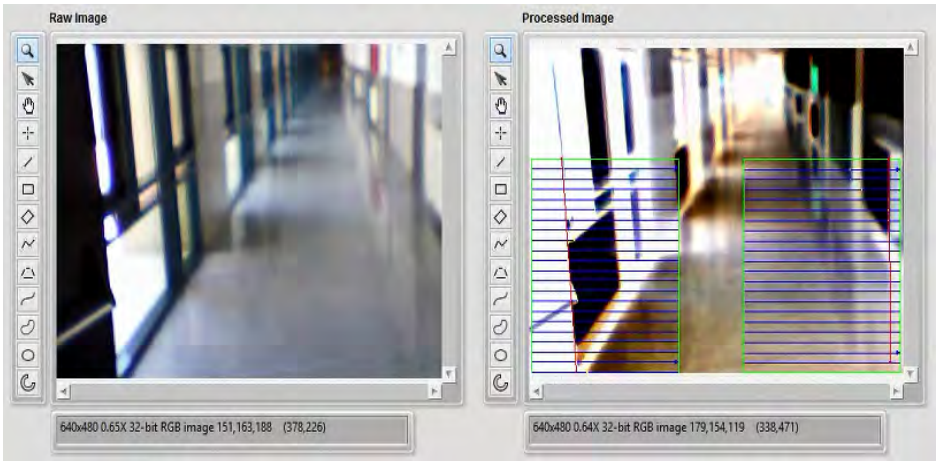


Figure 6.2: Experiment 2

Results showed that even though in certain ocassions it was possible to get path borders, most of the times, the straight lines found were random ones belonging to walls, doors, and other straight lines of the surroundings. This can be seen in figures 6.2 and 6.3.

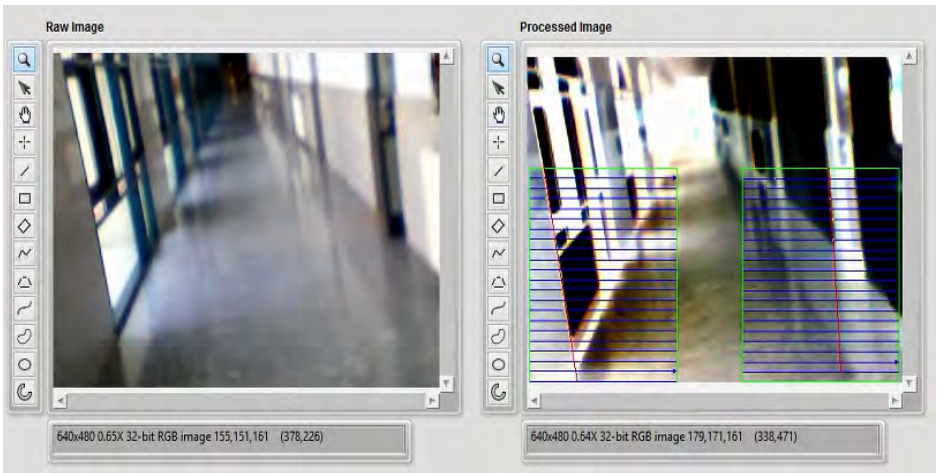


Figure 6.3: Experiment 3

In this way, it was clear that an edge processing that removes noise and leave us just with the edges, (for selecting afterwards just the straight ones in certain regions), was needed.

After these first verifications, it was checked that the edge processing programmed worked properly. For this, a walk with the protoype was done, and it

was evident that the most remarkable edges were detected easily. As an example see figure 6.4.

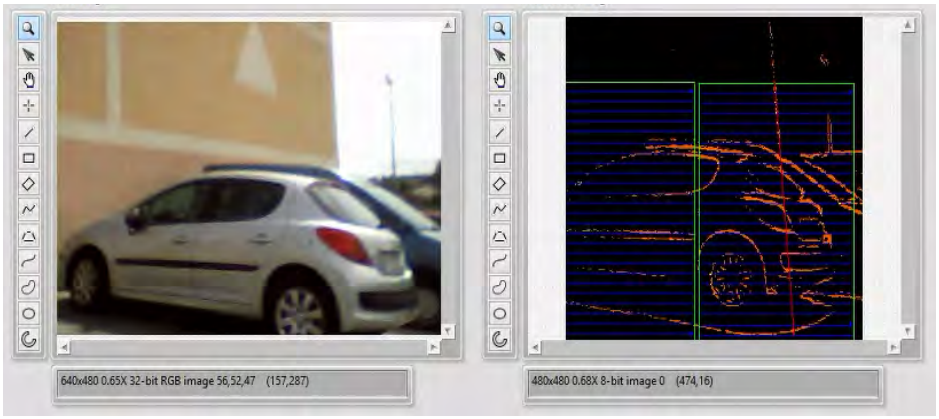


Figure 6.4: Edge Detection Testing

It was seen, inside a mall, that the processing was especially effective where there are **high contrasts** like:

- **Dark shapes with light backgrounds**. Examples can be seen at figures 6.5 and 6.6.

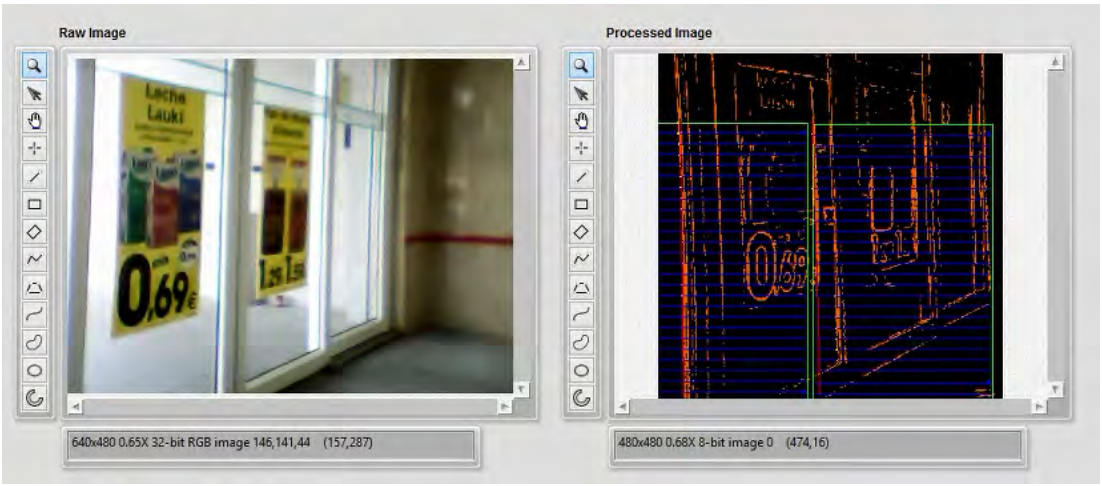


Figure 6.5: Dark shapes with light backgrounds 1

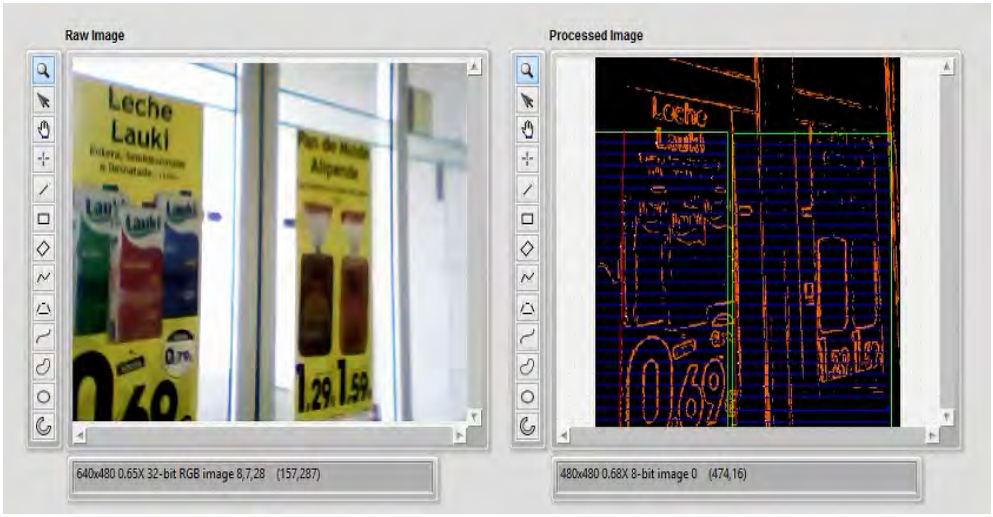


Figure 6.6: Dark shapes with light backgrounds 2

- **Light shapes with dark backgrounds**. Example can be seen at figure 6.7. In this case (light borders with dark background) the test was done on the Automation Laboratory of Carlos III University, where a path is marked. It can be seen that both line borders are detected (fine red lines).

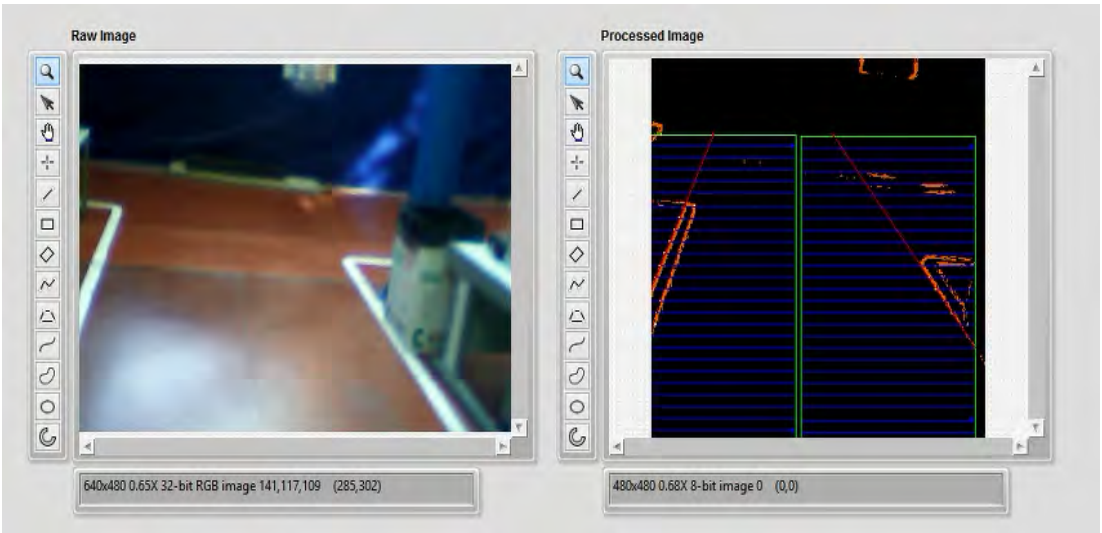


Figure 6.7: Light shape with dark background

Even in a bad situation in which obstacles cover part of the path border, the system works, as seen in figure 6.8.

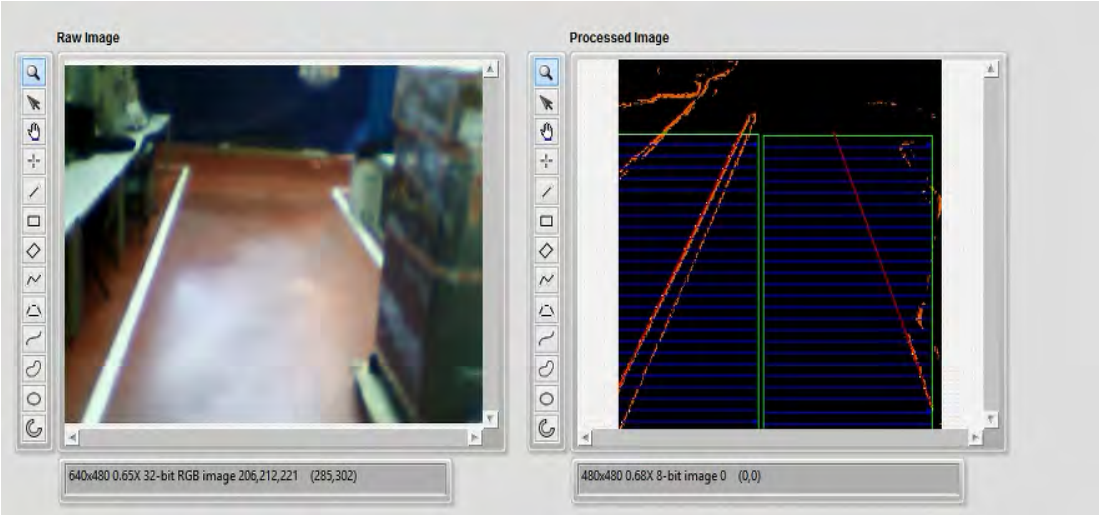


Figure 6.8: System working despite obstacles

Therefore it was found that the developed prototype worked properly only if one of the two previously mentioned conditions (path delimited by light lines and with dark background or the other way round) is fulfilled in the path. This is the case, for example, of jogging tracks, roads, and some paths marked inside airports and hospitals. In these cases, the prototype outputs the right guidance decision in 95 per cent of the times. If none of these two conditions is met, then the percentage of yielding the right guidance decision decreases considerably.

In figure 6.9 it is possible to see that the two path's borders are found, even though we are at the end of the path. It can be noticed that, as our heading is correct, the signals indicating to be *alert* and *out of the way* are both *OFF*. In this moment the prototype's output was 'keep straight' and was received through the earphones.

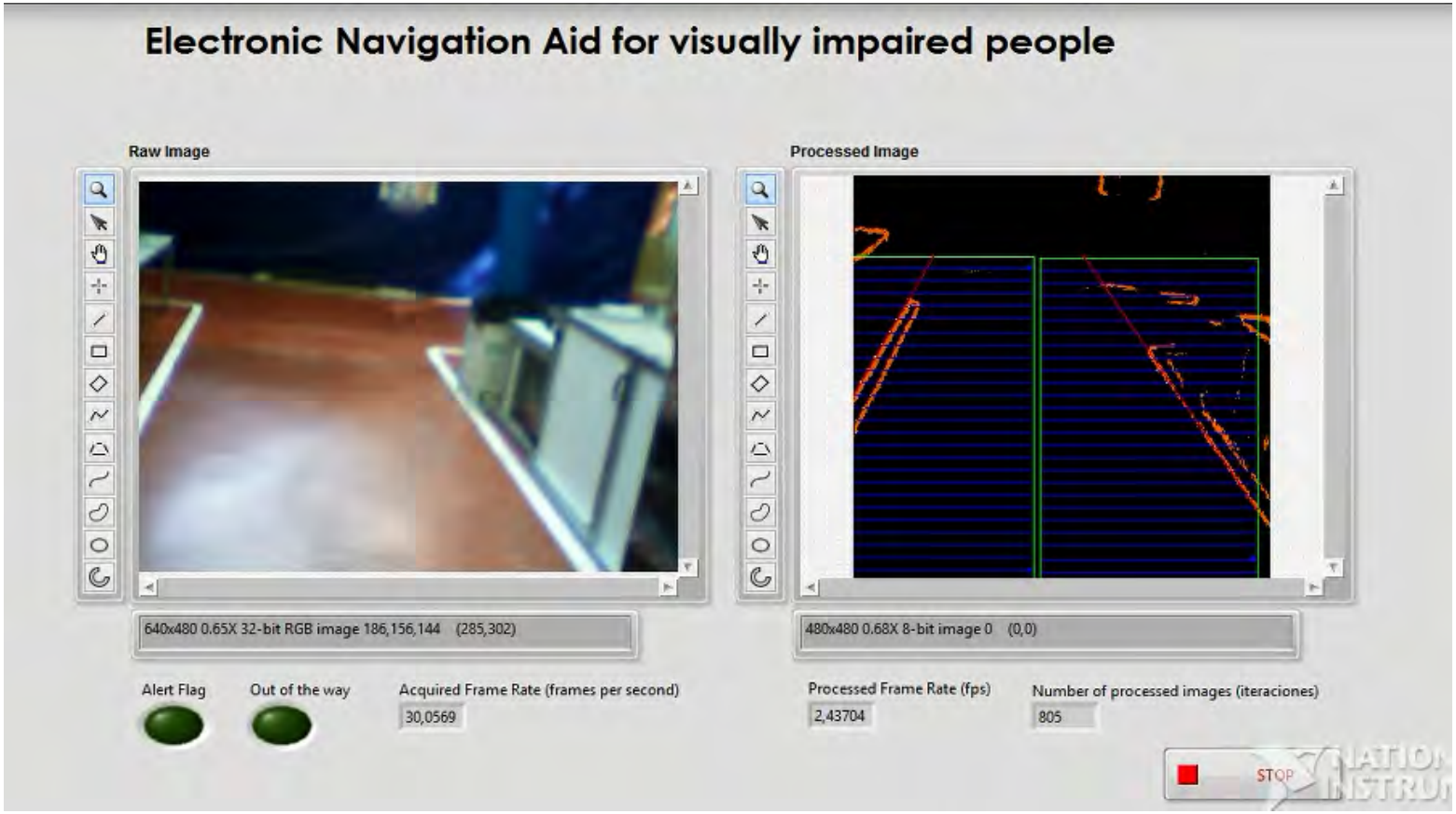


Figure 6.9: Correct prototype's performance

6.2 Prototype's appearance

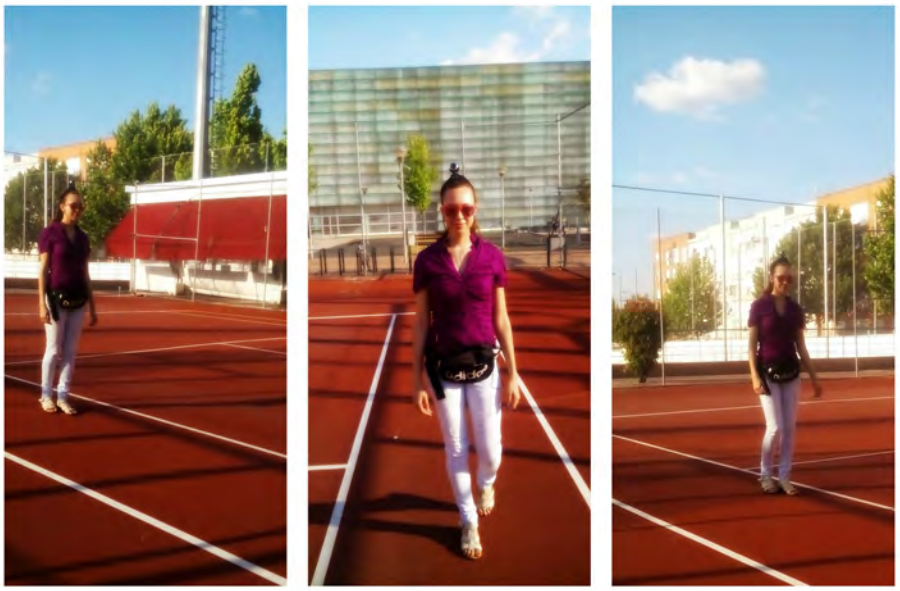


Figure 6.10: Prototype's Appearance

A person wearing this prototype would look as seen in figure 6.10. It can be said that wearing this system is not uncomfortable as it is not heavy, neither bulky. Inside the small bag, that follows the waist, myRIO and its battery are placed. Earphones are connected to the myRIO. Camera has an small hook that can be attached to a diadem in the head, making it stable. Therefore, it can be seen that the objective of making a free-hands prototype has been achieved.

This system does not prevent the person from getting important information from the environment through hearing, neither from having a conversation while walking, as he can wear only one of the earphones.

This system works properly for walking speed. However for jogging speed some troubles related to camera's stability may arise if the person runs fast. This topic will be seen in detail in future improvements section of next chapter.

6.3 Test results

After the above tests and verifications were performed, it was clear that this prototype have several restrictions, and these have to do with the environments in which it can be used. It is necessary to do a good image processing and perform an edge detection method to remove noise and select only the right edges and lines of the frames captured.

Otherwise the guidance decision will be based on lines which are not the ones that delimit the path. If there is not a big contrast difference between the path and its borders then path's borders won't be detected and guidance won't be possible. Consequently this prototype can only be used in paths whose borders are light and whose floor is dark or the other way round. (paths whose borders are dark and whose floor is light). The chosen scenario to do the final tests and verifications were some jogging tracks, and it was found that there the prototype behaves as expected, giving properly guidance according to the marked paths.

There was also found a constraint related to the camera's placement in this prototype. For a walking speed, and even soft jogging, there is not any trouble related with its stability. However, this is not the case for a running speed. In this case the person wouldn't run comfortably as the camera may fall, or even if it doesn't fall it may change its central position with the, consequently, inaccurate outputs.

The best solution for this is shown in figure 6.11. This glasses have the camera in the center so the same control algorithm that assumes that vanishing point should be around the central image pixel, is still valid. In this way, using this webcam integrated in some sunglasses, a blind or visually impaired person would be able to participate in athletic runnings with independence.

Figure 6.11: Glasses with integrated camera

Deeper explanation of some improvements will be made in chapter 8.

It was possible to contact with ONCE Aranjuez in order to perform some tests with a visually impaired person. Therefore, some tests were performed with a blind man named Israel in order to get some feedback of a future user of the prototype, as it can be seen in figure 6.12. Tests were performed in some jogging tracks.

Figure 6.12: Tests with a blind person

Main feedback given from Israel was the following:

- There should be a sound indicating the device has been turned on and turned off.
- He thinks it would be better to have a specific sound for indicating left, another for indicating right, and another for straight. He thinks that short and easily recognizable sounds might be better and faster than speech recording.
- He thinks that better than using earphones it would be to have some small speakers in the sideburns of the sunglasses, and of course, no wires would be a great improvement.

Chapter 7

Work Phases and Budget

Work Phases

Time invested in the design of this project and in its implementation will be exposed. For a better understanding the planification of the project has been divided into phases. A summary is shown at table 7.1.

1. Initial documentation
    - Problem statement: 20 hours.
    - Documentation of the state of the art: 60 hours.
2. Design
    - Searching of different design alternatives and its restrictions: 90 hours.
    - Devices and components searching: 10 hours.
3. Prototype's Implementation
    - Tests and verifications performed: 60 hours.
    - Improvements: 40 hours.
4. Thesis Writing
    - Content's writing: 70 hours.
    - Document structure and script generation: 50 hours.

Therefore, the most important parts in the development of this project have been its design along with its implementation.

Phases	**Hours**
1.Initial documentation	80
2.Design	100
3.Prototype's Implementation	100
4.Thesis Writing	120
Total	400

Table 7.1: Working hours by phases

A *Gantt char*t will be presented in figure 7.1. In this chart it is possible to see the planning of the project. The Gantt chart has been useful for keeping project's development up to date.

Dividing the semester in 24 weeks, is estimated that the first twelve weeks an average of 14 hours per week were spent. During the last twelve weeks an average of 16 hours per week have been invested.

During the prototype's design phase different alternatives were considered. This evaluation of different designs took time and it is reflected in this phase (prototype's design) in the chart. After a decision was taken, the programming of the idea in mind was performed.

Most of the programming took place in the implementation phase. As it was being programmed different tests of system's effectiveness were performed simultaneously, and finally, once all the design was programmed, overall prototype's funcionality was tested again and several improvements were made.

Thesis writing started parallel to prototype's improvements and lasted till the end of the semester.

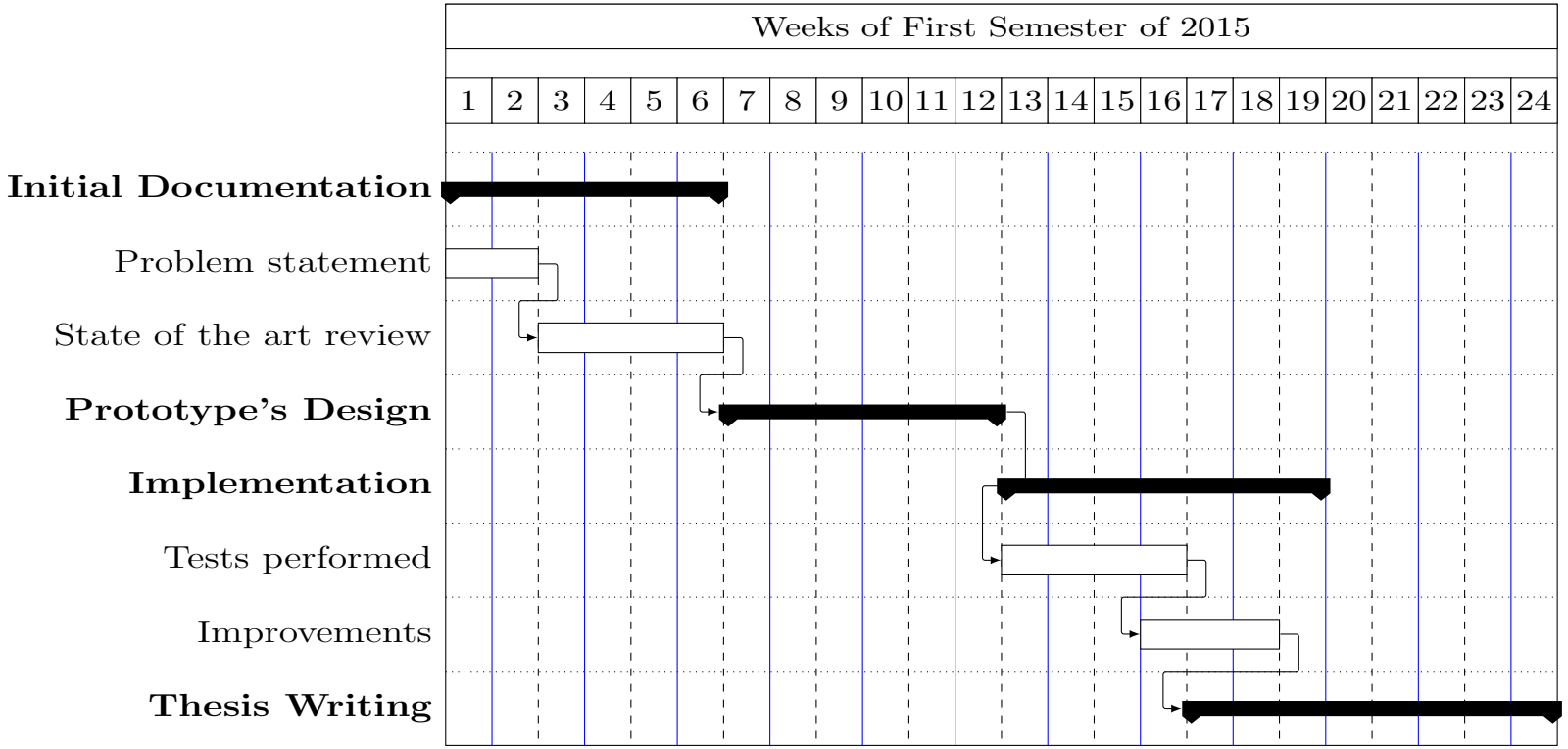


Figure 7.1: Gantt Chart for project's planification

Estimated Budget

A cost estimation will be presented. In the development of this project there are mainly three costs: Devices's cost, software license's cost and engineering costs. In table 7.2 these costs are detailed.

In the estimation non-student but commercial prices have been taken into account. These prices don't take into account any tax (in Spain, IVA is not included). As myRIO is an academic device, which is not sold to the industrial sector, a Single-Board RIO has replaced it. Engineering costs have been estimated taking an average of 200 hours dedicated in the design and implementation of the prototype, and an average of 50 euros per hour that earns a LabVIEW programmer.

Single-Board RIO is useful (as myRIO) for building prototypes or manufacturing pre-series. For manufacturing large series then it is better to choose smaller comercial chips.

Details	Amount(Euros)
HARDWARE COSTS:	
Single-Board RIO	1000
Webcam	25
Battery	l8
Chargers	
DC-DC	12
AC-DC	14
Earphones	6
TOTAL HARDWARE	1075
SOFTWARE COSTS:	
NI LabVIEW Embedded Control and Monitoring Suite	11100
NI Vision Development Module Run-Time License	465
TOTAL SOFTWARE	11565
ENGINEERING COSTS	10000
TOTAL ESTIMATION OF PROJECT'S COST	22640

Table 7.2: Budget Estimation

About the economical environment of electronic aids for guidance and navigation for visually impaired people, it can be said that there are several commercial products in the market with limited functionality and small scientific value. Their cost is relatively high and are not widely used. Examples of this products can be seen at references [38], [39] and [40]. Prices are around 1000 dollars.

This project is just a prototype, and in order to get a commercial product it is necessary to make some of the improvements mentioned in chapter 8, and to do more tests and verifications. Prototype costs are not comparable to commercial products, because the first ones will always be much higher. An entrepreneurship proposal has been carried out (*TFG Emprende*), in which it is shown how a competitive commercial price can be achieved.

Chapter 8

Conclusions and Future Improvements

The created prototype is able to guide a visually impaired person in certain paths. The goals of making it wearable, light and with real-time execution have been achieved. The prototype is *free-hands* and partially *free-ears* as it is not required to use both earphones. The performance obtained in selected paths (those that meet the requirements mentioned in chapter 6) is good enough for a fast walking speed and even for a slow jogging speed.

Using this system a blind or visually impaired person is able to get a higher degree of independency, which adds to his life quality. The development of this kind of prototypes may help a wide range of people. It is always good to use the available technology to improve people's life quality.

The election of the means for the development of this prototype, has been appropriate and rather advantageous. LabVIEW programming language is more intuitive than text-based ones, and in many ocassions, helps to get a better understanding of how our program works. LabVIEW makes possible to have a global perspective of it.

It has been also demonstrated that it is possible to build a working prototype in a semester period using a myRIO device. Both parts, real-time processor and FPGA modules have been used, and thanks to it the system is more efficient than if only one part has been used.

Therefore, as it has been detailed, even though it is not very common, it is possible to create a guidance system using artificial vision. However, as shown in chapter 6, it has several limitations. Coming up next, it will be shown some improvements to be done for achieving higher functionality.

Future Improvements

The first significant improvement, as mentioned in chapter 6, would be using some sunglasses which incorporate a webcam, like the ones shown in figure 6.11. In this way the prototype would be much more unnoticeable and more comfortable. There wouldn't be any trouble for using it while running fast, and of course it wouldn't be necessary to wear a diadem. For extra comfortability it would be ideal that this glasses were wireless and could incorporate some small speakers on its sideburns.

Also, as mentioned by the visually impaired person who tried the prototype, it is possible to replace the speech recording being played by specific and easily recognizable sounds.

The second important improvement has to do with the frames per second rate of processed images. Currently it has an average of 2.6 *fps*. A good way to improve this rate would be trying to implement all the image processing block in the FPGA. Now it is done in the Real-Time processor and FPGA is only used for the audio block. With a higher *fps* rate, the system will be more reliable for fast running.

Another important improvement would be to add another source of guidance information for the user, like using small servo-motors that the person could wear in his clothes and vibrate in the direction in which the person should go. For the implementation of this, some AO (Analog Outputs) of myRIO should be used, and a PID VI could be used for the control of these motors.

For add-on functionality, myRIO Wi-Fi capabilities could be used. With myRIO it is possible to create a wireless network. A laptop can connect to this network to share data. It is possible to create image global variables inside a library and deploy this library in the network, so images could be monitored in real-time.

Additionally a GPS module could be connected to myRIO using some of its AI (Analog Inputs) or one of the UART lines (depending on the module) and then the location information could also be shared using this wireless network. These two funcionalities (image monitoring and location information in real-time) could be useful for the guardian of the visually impaired person.

Another improvement would be to replace the while loops with *Timed loops* in order to control the priority of the loops executing in parallel. These timed loops are very useful for bottleneck analysis.

Finally, for increasing the range of paths where the system can be used, it would be necessary to develop a different image processing algorithm or complement the current one with additional sensors. If the sunglasses had two webcams then it would be possible to have stereo vision, which could be useful for measuring distances to obstacles.

In summary, it can be said that this architecture is created in such a way that is easy to scale it. The scalability is such an important feature, as it allows to have further future development of this prototype.

Glossary

- **AI**: Analog input
- **AO**: Analog output
- **API**: Application programming interface
- **AXI**: Advanced extensible interface
- **DSCED**: Discrete singular convolution edge detector
- **EOA**: Electronic orientation aid
- **ETA**: Electronic travel aid
- **FIFO**: First in, first out
- **FPGA**: Field programmable array
- **FSCS**: Full scale contrast stretching
- **FTP**: File Transfer Protocol
- **GPS**: Global Positioning System
- **IDE**: Integrated Development Environment
- **IMU**: Inertial measurement unit
- **IPNS**: Integrated pedestrian navigation system
- **IRLS**: Iteratively reweighted least squares
- **LabVIEW**: Laboratory Virtual Instrument Engineering Workbench
- **LVQ**: Learning vector quantization
- **MEMS**: Micro-electrical, micro-mechanical sensors
- **NaN**: Not a number

- **NI**: National Instruments
- **ONCE**: Organización Nacional de ciegos españoles
- **OEM**: Original equipment manufacturer
- **PID**: Proportional-integral-derivative controller
- **PLD**: Position locator device
- **RIO**: Reconfigurable input/output
- **SNR**: Signal-to-noise ratio
- **UART**: Universal asynchronus receiver-transmitter
- **UI**: User Interface
- **UVC**: USB Video device class (protocol for image acquisition)
- **VHDL**: Hardware description language
- **VI**: Virtual Instrument
- **VL**: Vanishing line
- **VP**: Vanishing point
- **WHO**: World Health Organization
- **WLAN**: Wireless Local Area Network
- **XML**: Extensible Markup Language

Bibliography

[1] World Health Organization. http://www.who.int/mediacentre/factsheets/fs282/en/. Visual impairment and blindness. May, 2015.

[2] Dimitrios Dakopoulos and Nikolaos G.Bourbakis. Wearable obstacle avoidance electronic travel aids for blind: A survey. *IEE Transactions on systems, man, and cybernetics-Part C: Applications and Reviews,vol 40*, January 2010.

[3] P.B.L. Mejier. An experimental system for auditory image representations. *IEEE Trans. Biomedical Engineering*, February 1992.

[4] A.Hub. J. Diepstraten and T. Ertl. Design and development of an indoor navigation and object identification system for the blind. *Proc. ACM SIGACCESS Accesibility Computing*, January 2004.

[5] L.F.Rodríguez Ramos J.L. González-Mora, A. Rodríguez Hernández and N.Sosa. Development of a new space perception system for blind people, based on the creation of a virtual acoustic space. *www.iac.es/proyect/eavi*, May, 2015.

[6] *Application Software Computer*, chapter Fuzzy image processing scheme for autonomous navigation of human blind. Kota Kinabalu, January 2007.

[7] *Foundations of Orientation and Mobility, Second Edition.* American Foundation for the Blind, 1997.

[8] *Proceedings of a symposium of consumer, user agency, researcher, and commercial experience with talking signs and related technologies*, 1995.

[9] *Transit accessibility improvement through talking signs remote infrared signage: A demonstration and evaluation*, 1995.

[10] R. L. Klatzky J. M.Loomis, R. G.Golledge. Navigation system for the blind: Auditory display modes and guidance. presence: Teleoperators and virtual environments. *MIT Press Journals*, 1998.

[11] Natalie Frietsch Michael Weinmann Gert F. Trommer Christoph Kessler, Christian Ascher. Vision-based attitude estimation for indoor navigation using vanishing points and lines. *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, May 2010.

[12] Staff Writers. Scientists design indoor navigation system for blind. *UPI Space Daily*, June 2012.

[13] AbdulMalik S. Al-Salman Mai A. Al-Ammar King, Hend S. Al-Khalifa King. A proposed indoor navigation system for blind individuals. *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, 2011.

[14] A. Walker R. Fisher, S. Perkins and E. Wolfart. Canny edge detector. june, 2015. Technical report, HI Works, 2003.

[15] A. Walker R. Fisher, S. Perkins and E. Wolfart. Sobel edge detector. june, 2015. Technical report, HI Works, 2003.

[16] Robo realm. Prewitt edge detector. june, 2015. Technical report, Vision for machines, 2015.

[17] A. Walker R. Fisher, S. Perkins and E. Wolfart. Roberts edge detector. june, 2015. Technical report, HI Works, 2003.

[18] Iain Murray Nimali Rajakaruna. Edge detection for local navigation system for vision impaired people using mobile devices. *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), IEEE Ninth International Conference*, 2014.

[19] G. W. W. Z.J. Hou. A new approach to edge detection. *The journal of the pattern recognition society Vol 35*, 2002.

[20] *Method and Means for Recognizing Complex Patterns.* US Patent 3,069,654, 1962.

[21] *Computer Vision: A Modern Approach.* Prentice Hall, 2002.

[22] *Machine Vision.* McGraw-Hill, 1995.

[23] *Robust Estimation.* Toronto, 2004.

[24] Lindsey Pittman Alfonso Croeze and Winnie Reynolds. Non-linear least squares problems with the gauss-newton and levenberg-marquardt methods. june, 2015. Technical report, University of Mississipi, Department of Mathematics, 2012.

[25] W. Zhang J. Kosecka. Video compass. In *Proceedings of the 7th European Conference on Computer Vision*.

[26] *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[27] National Instruments. Ni rio hardware architecture. In *www.ni.com*.

[28] National Instruments. Ni myrio. user guide and specifications. In *http://www.ni.com/pdf/manuals/376047a.pdf*.

[29] Simon Hogg. What is labview? Technical report, National Instruments, June, 2015.

[30] co-founded National Instruments Jeff Kodosky. Is labview a general purpose programming language? In *www.ni.com/white-paper/5313/en/*. June, 2015.

[31] National Instruments. Ni vision for labview user manual. In *http://www.ni.com/pdf/manuals/371007b.pdf*. June, 2015.

[32] A. Walker R. Fisher, S. Perkins and 2015 E. Wolfart. June. Contrast stretching. Technical report, HI Works, 2003.

[33] Umesh Rajashekar. The digital image processing gallery. Technical report, New York University - Computational Vision department, June, 2015.

[34] National Instruments. Transfering files to rt target with ftp. june, 2015. In *http://digital.ni.com/public.nsf/allkb/EF2C5AE37FE3672186256C3D005489A0*.

[35] National Instruments. Using webdav to transfer files to your real-time target. june, 2015. In *http://digital.ni.com/public.nsf/allkb/4EBE45E8A816B19386257B6C0071D025*.

[36] National Instruments. Audio output n samples vis. In *http://zone.ni.com/reference/en-XX/help/373925B-01/myrioreference/*.

[37] Audacity. Digital audio editor. Technical report, Audacity Team, June, 2015.

[38] 2015 Bay Advanced Technologies Ltd. June. K-sonar cane. In *http://www.batforblind.co.nz/index.htm*.

[39] 2015 GDP Research, Australia. June. Miniguide. In *http://www.gdp-research.com.au/*.

[40] 2015 Sound Foresight Ltd. June. Ultracane. In *http://www. batcane.com/*.

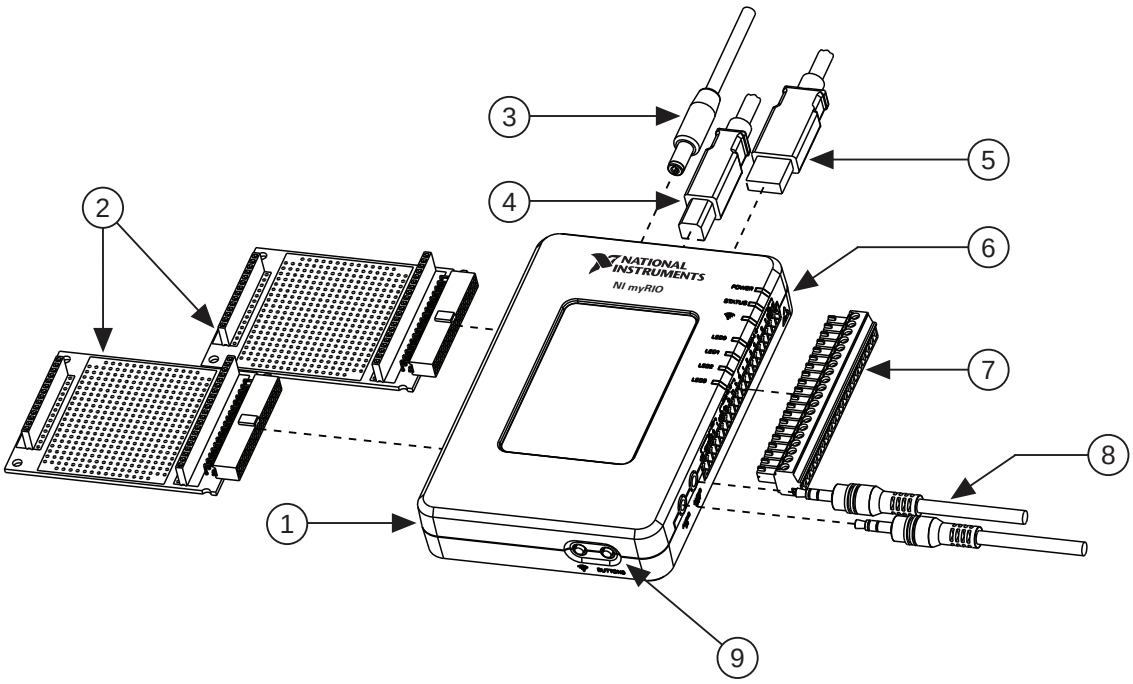
Annex: NI myRIO User Guide

USER GUIDE AND SPECIFICATIONS

NI myRIO-1900

The National Instruments myRIO-1900 is a portable reconfigurable I/O (RIO) device that students can use to design control, robotics, and mechatronics systems. This document contains pinouts, connectivity information, dimensions, mounting instructions, and specifications for the NI myRIO-1900.

Figure 1. NI myRIO-1900



1	NI myRIO-1900	6	LEDs
2	myRIO Expansion Port (MXP) Breakouts (One Included in Kit)	7	Mini System Port (MSP) Screw-Terminal Connector
3	Power Input Cable	8	Audio In/Out Cables (One Included in Kit)
4	USB Device Cable	9	Button0
5	USB Host Cable (Not Included in Kit)		

Connector Pinouts

NI myRIO-1900 Expansion Port (MXP) connectors A and B carry identical sets of signals. The signals are distinguished in software by the connector name, as in `ConnectorA/DIO1` and `ConnectorB/DIO1`. Refer to the software documentation for information about configuring and using signals. The following figure and table show the signals on MXP connectors A and B. Note that some pins carry secondary functions as well as primary functions.

Figure 3. Primary/Secondary Signals on MXP Connectors A and B

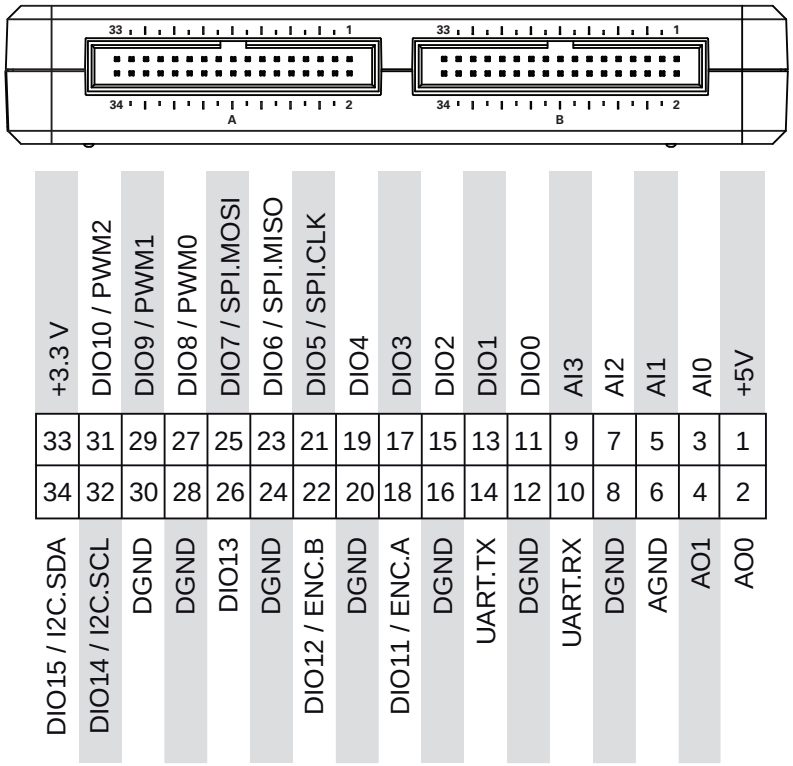


Table 1. Descriptions of Signals on MXP Connectors A and B

Signal Name	Reference	Direction	Description
+5V	DGND	Output	+5 V power output.
AI <0..3>	AGND	Input	0-5 V, referenced, single-ended analog input channels. Refer to the *Analog Input Channels* section for more information.
AO <0..1>	AGND	Output	0-5 V referenced, single-ended analog output. Refer to the *Analog Output Channels* section for more information.
AGND	N/A	N/A	Reference for analog input and output.
+3.3V	DGND	Output	+3.3 V power output.
DIO <0..15>	DGND	Input or Output	General-purpose digital lines with 3.3 V output, 3.3 V/5 V-compatible input. Refer to the *DIO Lines* section for more information.
UART.RX	DGND	Input	UART receive input. UART lines are electrically identical to DIO lines.
UART.TX	DGND	Output	UART transmit output. UART lines are electrically identical to DIO lines.
DGND	N/A	N/A	Reference for digital signals, +5 V, and +3.3 V.

The following figure and table show the signals on Mini System Port (MSP) connector C. Note that some pins carry secondary functions as well as primary functions.

Figure 4. Primary/Secondary Signals on MSP Connector C

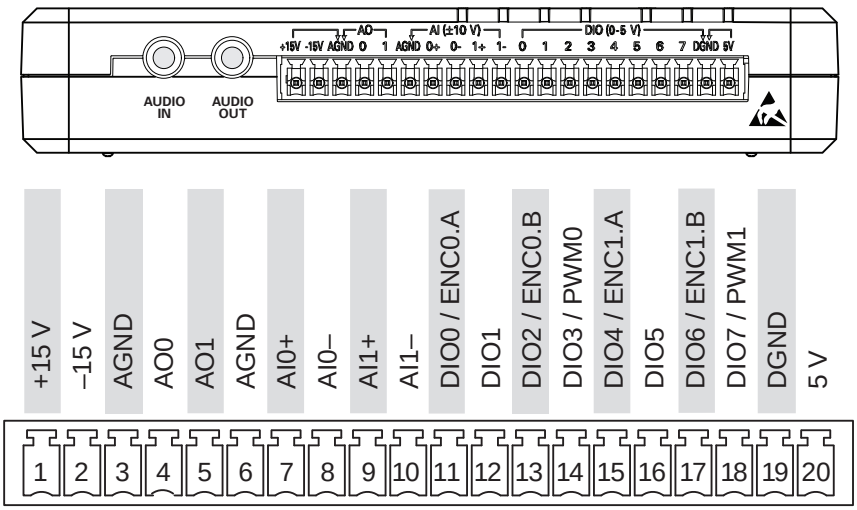


Table 2. Descriptions of Signals on MSP Connector C

Signal Name	Reference	Direction	Description
+15V/-15V	AGND	Output	+15 V/-15 V power output.
AI0+/AI0-; AI1+/AI1-	AGND	Input	±10 V, differential analog input channels. Refer to the *Analog Input Channels* section for more information.
AO <0..1>	AGND	Output	±10 V referenced, single-ended analog output channels. Refer to the *Analog Output Channels* section for more information.
AGND	N/A	N/A	Reference for analog input and output and +15 V/-15 V power output.
+5V	DGND	Output	+5 V power output.
DIO <0..7>	DGND	Input or Output	General-purpose digital lines with 3.3 V output, 3.3 V/5 V-compatible input. Refer to the *DIO Lines* section for more information.
DGND	N/A	N/A	Reference for digital lines and +5 V power output.

Table 3. Descriptions of Signals on Audio Connectors

Signal Name	Reference	Direction	Description
AUDIO IN	N/A	Input	Left and right audio inputs on stereo connector.
AUDIO OUT	N/A	Output	Left and right audio outputs on stereo connector.

Analog Input Channels

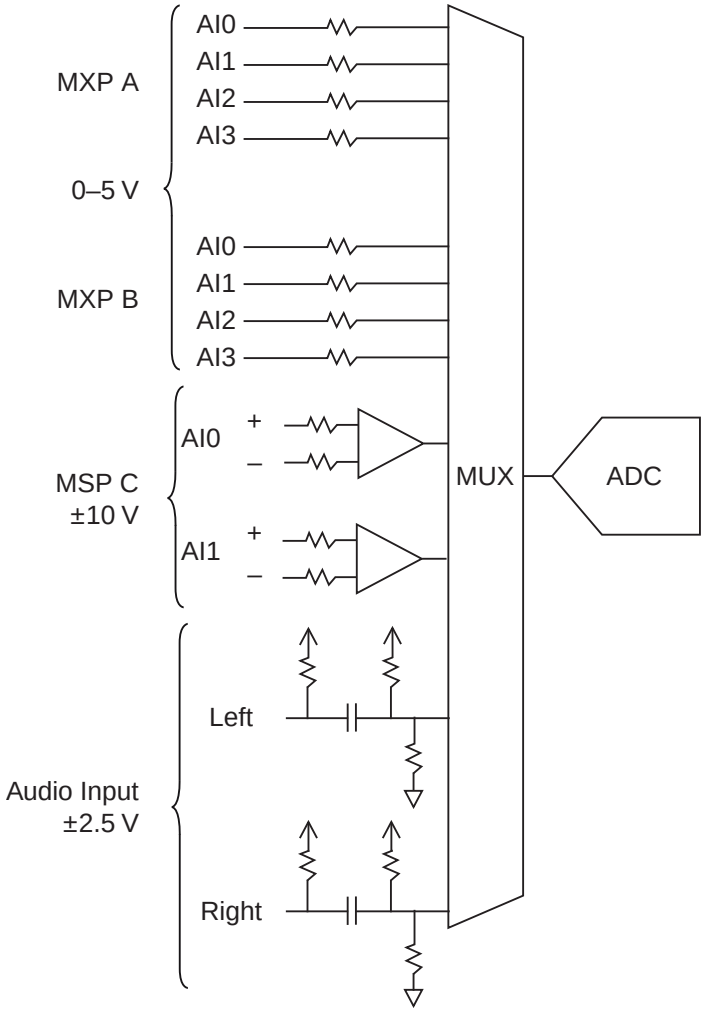
The NI myRIO-1900 has analog input channels on myRIO Expansion Port (MXP) connectors A and B, Mini System Port (MSP) connector C, and a stereo audio input connector. The analog inputs are multiplexed to a single analog-to-digital converter (ADC) that samples all channels.

MXP connectors A and B have four single-ended analog input channels per connector, AI0-AI3, which you can use to measure 0-5 V signals. MSP connector C has two high-impedance, differential analog input channels, AI0 and AI1, which you can use to measure signals up to ±10 V. The audio inputs are left and right stereo line-level inputs with a ±2.5 V full-scale range.

Note For important information about improving measurement accuracy by reducing noise, go to `ni.com/info` and enter the Info Code `analogwiring`.

Figure 5 shows the analog input topology of the NI myRIO-1900.

Figure 5. NI myRIO-1900 Analog Input Circuitry



Analog Output Channels

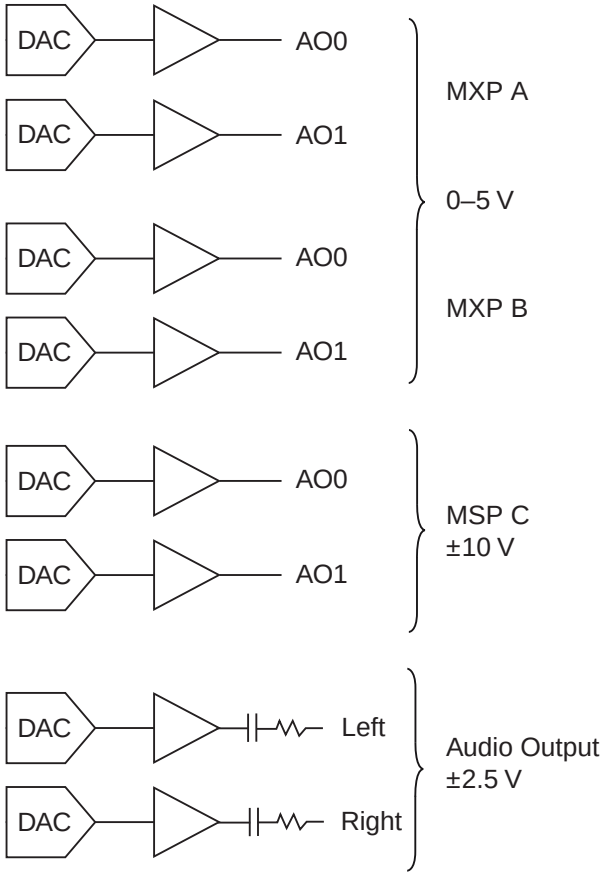
The NI myRIO-1900 has analog output channels on myRIO Expansion Port (MXP) connectors A and B, Mini System Port (MSP) connector C, and a stereo audio output connector. Each analog output channel has a dedicated digital-to-analog converter (DAC), so they can all update simultaneously. The DACs for the analog output channels are controlled by two serial communication buses from the FPGA. MXP connectors A and B share one bus, and MSP connector C and the audio outputs share a second bus. Therefore, the maximum update rate is specified as an aggregate figure in the *Analog Output* section of the *Specifications*.

MXP connectors A and B have two analog output channels per connector, AO0 and AO1, which you can use to generate 0-5 V signals. MSP connector C has two analog output channels, AO0 and AO1, which you can use to generate signals up to ±10 V. The audio outputs are left and right stereo line-level outputs capable of driving headphones.

Caution Before using headphones to listen to the audio output of the NI myRIO-1900, ensure that the audio output is at a safe level. Listening to audio signals at a high volume may result in permanent hearing loss.

Figure 6 shows the analog output topology of the NI myRIO-1900.

Figure 6. NI myRIO-1900 Analog Output Circuitry



Accelerometer

The NI myRIO-1900 contains a three-axis accelerometer. The accelerometer samples each axis continuously and updates a readable register with the result. Refer to the *Accelerometer* section of the *Specifications* for the accelerometer sample rates.

Converting Raw Data Values to Voltage

You can use the following equations to convert raw data values to volts:

$$V = Raw\ Data\ Value * LSB\ Weight$$

$$LSB\ Weight = Nominal\ Range \div 2^{ADC\ Resolution}$$

where *Raw Data Value* is the value returned by the FPGA I/O Node,

LSB Weight is the value in volts of the increment between data values,

Nominal Range is the absolute value in volts of the full, peak-to-peak nominal range of the channel,

and *ADC Resolution* is the resolution of the ADC in bits. (*ADC Resolution* = 12.)

- For AI and AO channels on the MXP connectors,

$$LSB\ Weight = 5\ V \div 2^{12} = 1.221\ mV$$

$$Maximum\ reading = 4095 * 1.221\ mV = 4.999\ V$$

- For AI and AO channels on the MSP connectors,

$$LSB\ Weight = 20\ V \div 2^{12} = 4.883\ mV$$

$$Maximum\ Positive\ Reading = +2047 * 4.883\ mV = 9.995\ V$$

$$Maximum\ Negative\ Reading = -2048 * 4.883\ mV = -10.000\ V$$

- For Audio In/Out,

$$LSB\ Weight = 5\ V \div 2^{12} = 1.221\ mV$$

$$Maximum\ Positive\ Reading = +2047 * 1.221\ mV = 2.499\ V$$

$$Maximum\ Negative\ Reading = -2048 * 1.221\ mV = -2.500\ V$$

- For the accelerometer,

$$LSB\ Weight = 16\ g \div 2^{12} = 3.906\ mg$$

$$Maximum\ Positive\ Reading = +2047 * 3.906\ mg = +7.996\ g$$

$$Maximum\ Negative\ Reading = -2048 * 3.906\ mg = -8.000\ g$$

DIO Lines

The NI myRIO-1900 has 3.3 V general-purpose DIO lines on the MXP and MSP connectors. MXP connectors A and B have 16 DIO lines per connector. On the MXP connectors, each DIO line from 0 to 13 has a 40 kΩ pullup resistor to 3.3 V, and DIO lines 14 and 15 have 2.2 kΩ pullup resistors to 3.3 V. MSP connector C has eight DIO lines. Each MSP DIO line has a 40 kΩ pulldown resistor to ground. DGND is the reference for all the DIO lines. You can program all the lines individually as inputs or outputs. Secondary digital functions include Serial Peripheral

Interface Bus (SPI), I2C, pulse-width modulation (PWM), and quadrature encoder input. Refer to the NI myRIO software documentation for information about configuring the DIO lines.

Figure 7. DIO Lines <13..0> on MXP Connector A or B

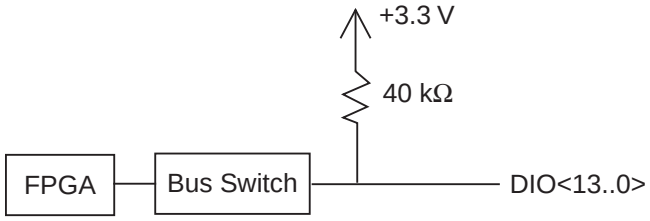


Figure 8. DIO Lines <15..14> on MXP Connector A or B

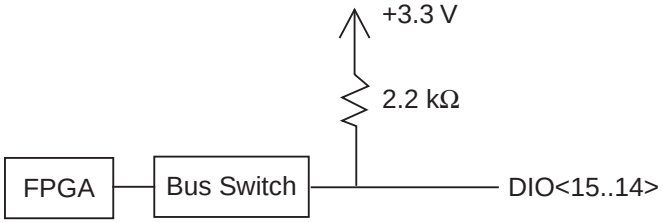
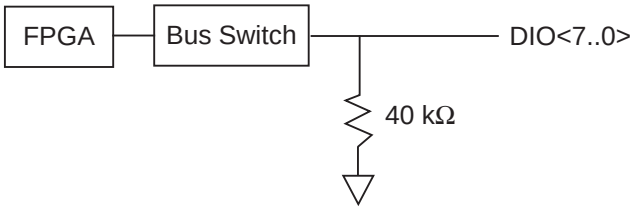


Figure 9. DIO Lines <7..0> on MSP Connector C



When a DIO line is floating, it floats in the direction of the pull resistor. A DIO line may be floating in any of the following conditions:

- when the myRIO device is starting up
- when the line is configured as an input
- when the myRIO device is powering down

You can add a stronger resistor to a DIO line to cause it to float in the opposite direction.

UART Lines

The NI myRIO-1900 has one UART receive input line and one UART transmit ouput line on each MXP connector. The UART lines are electrically identical to DIO lines 0 to 13 on the MXP connectors. Like those lines, UART.RX and UART.TX have 40 kΩ pullup resistors to 3.3 V. Use LabVIEW Real-Time to read and write over the UART lines.

Using the Reset Button

Pressing and releasing the Reset button restarts the processor and the FPGA.

Pressing and holding the Reset button for 5 seconds, then releasing it, restarts the processor and the FPGA and forces the NI myRIO-1900 into safe mode. In safe mode, the NI myRIO-1900 launches only the services necessary for updating configuration and installing software.

When the NI myRIO-1900 is in safe mode, you can communicate with it by using the UART lines on MXP connector A. You need the following items to communicate with the myRIO device over UART:

- USB-to-TTL serial UART converter cable (for example, part number TTL-232RG-VSW3V3-WE from FTD Chip)
- Serial-port terminal program configured with the following settings:
  - 115,200 bits per second
  - Eight data bits
  - No parity
  - One stop bit
  - No flow control

Using the Wireless Button and LED

For information about using the Wireless button, go to `ni.com/info` and enter the Info Code `myriowirelessbutton`.

For information about using the Wireless LED, go to `ni.com/info` and enter the Info Code `myriowirelessled`.

Using Button0

Button0 produces a logic TRUE when depressed and a logic FALSE when not depressed. Button0 is not debounced.

NI myRIO-1900 Physical Dimensions

Figure 10. NI myRIO-1900 Dimensions, Front

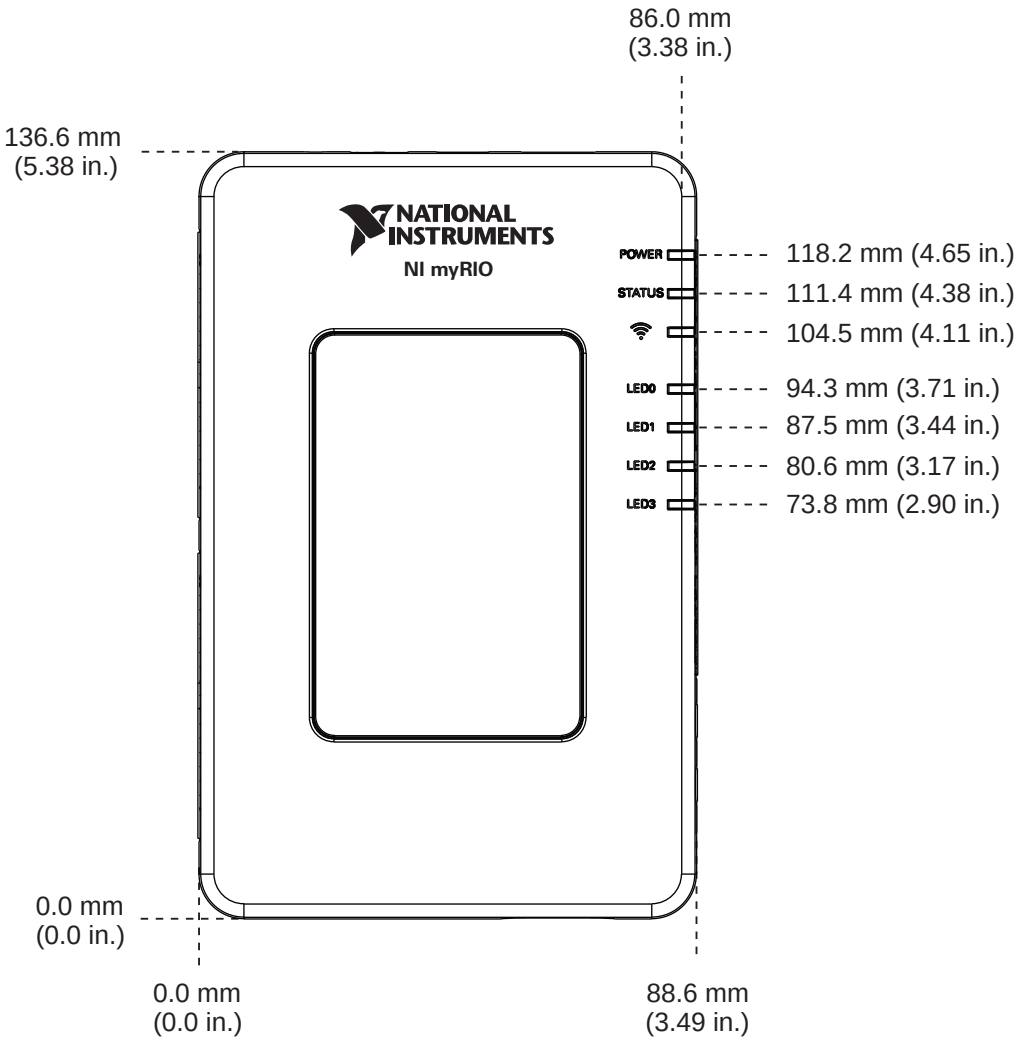


Figure 11. NI myRIO-1900 Dimensions, Back

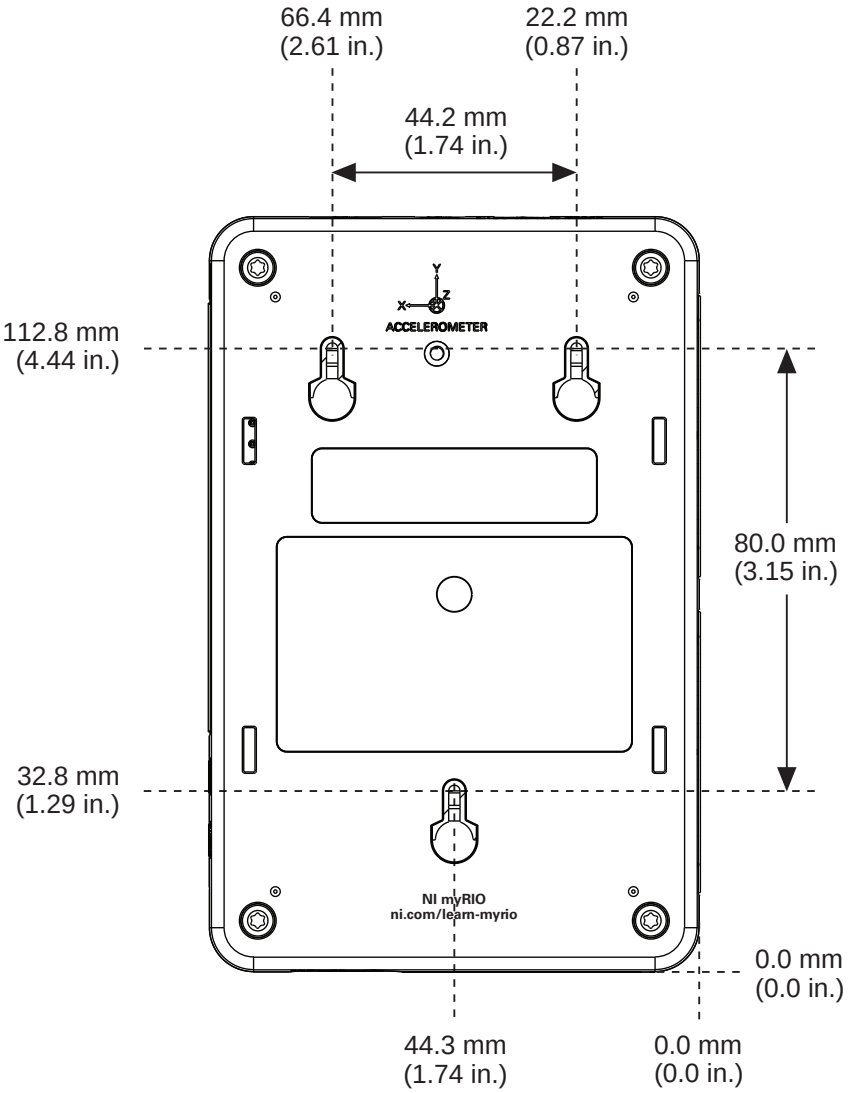
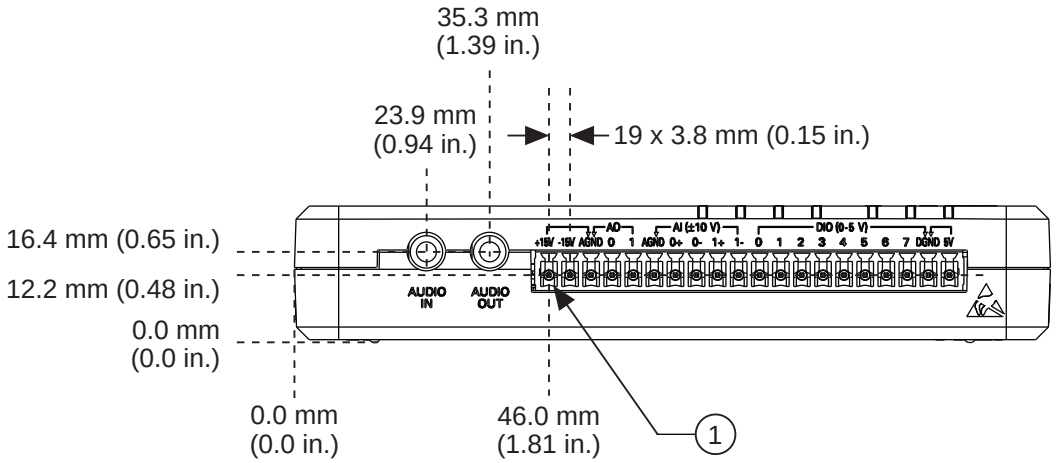
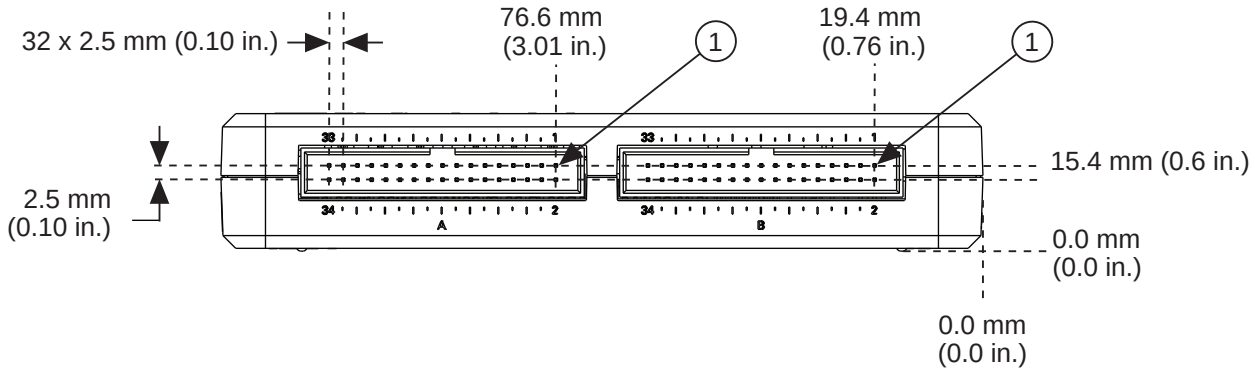


Figure 12. NI myRIO-1900 Dimensions, MSP Side



1	Pin 1

Figure 13. NI myRIO-1900 Dimensions, MXP Side



1 Pin 1

Figure 14. NI myRIO-1900 Dimensions, I/O End

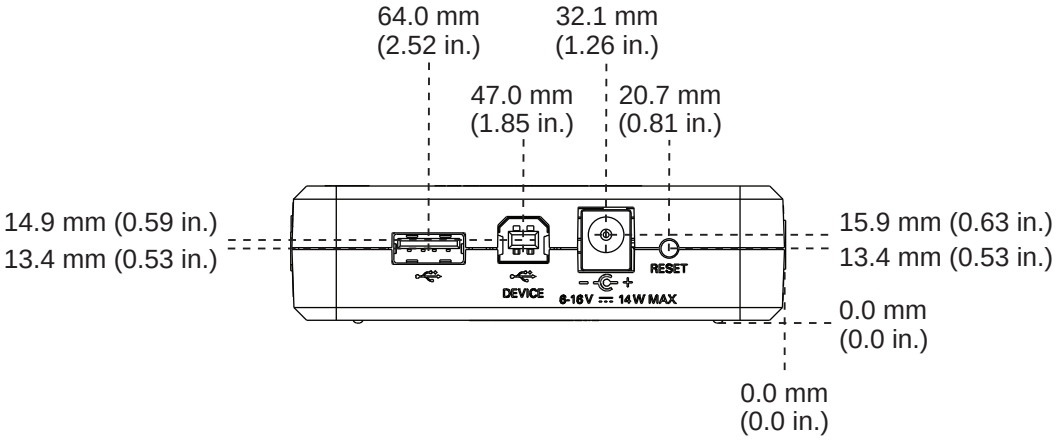
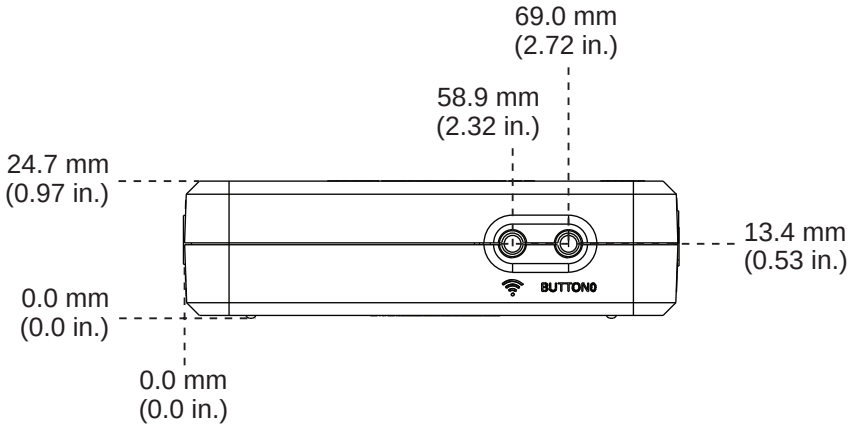


Figure 15. NI myRIO-1900 Dimensions, User End



Mounting the NI myRIO-1900

Mounting the NI myRIO-1900 Using the Key Holes

You can use the provided key holes on NI myRIO-1900 to mount the device on a flat surface. Install the NI myRIO-1900 as shown in Figure 16. Use Unified #4 or ISO M3 screws to mount the NI myRIO-1900 using the key holes. Panhead screws are suitable for use with the NI myRIO-1900 key holes.

Cables and Accessories

Table 5. Accessories Available from NI

Accessory	Description	NI Part Number
Power supply	Power supply for NI myRIO-1900	723403-01
MXP breakouts	Set of five MXP breakout boards for NI myRIO-1900	782696-01
MSP connector	MSP replacement connector plug for NI myRIO-1900	765788-01
Panel mounting kit	Panel mounting kit for NI myRIO-1900	783091-01

Specifications

The following specifications are typical for the 0 to 40 °C operating tempreature range unless otherwise noted.

Processor

Processor type ..... Xilinx Z-7010

Processor speed ..... 667 MHz

Processor cores 2

Memory

Nonvolatile memory ..... 256 MB

DDR3 memory ..... 512 MB

- DDR3 clock frequency ..... 533 MHz
- DDR3 data bus width ..... 16 bits

For information about the lifespan of the nonvolatile memory and about best practices for using nonvolatile memory, go to `ni.com/info` and enter the Info Code `SSDBP`.

FPGA

FPGA type ..... Xilinx Z-7010

Wireless Characteristics

Radio mode ..... IEEE 802.11 b,g,n

Frequency band ..... ISM 2.4 GHz

Channel width ..... 20 MHz

Channels ..... USA 1-11, International 1-13

TX power ..... +10 dBm max (10 mW)

Outdoor range ..... Up to 150 m (line of sight)

Antenna directivity ..... Omnidirectional

Security ..... WPA, WPA2, WPA2-Enterprise

USB Ports

USB host port ..... USB 2.0 Hi-Speed

USB device port ..... USB 2.0 Hi-Speed

Analog Input

Aggregate sample rate ..... 500 kS/s

Resolution ..... 12 bits

Overvoltage protection ..... ±16 V

MXP connectors

- Configuration ..... Four single-ended channels per connector
- Input impedance ..... >500 kΩ acquiring at 500 kS/s
  1 MΩ powered on and idle
  4.7 kΩ powered off
- Recommended source impedance ..... 3 kΩ or less
- Nominal range ..... 0 V to +5 V
- Absolute accuracy ..... ±50 mV
- Bandwidth ..... >300 kHz

MSP connector

- Configuration ..... Two differential channels
- Input impedance ..... Up to 100 nA leakage powered on;
  4.7 kΩ powered off
- Nominal range ..... ±10 V
- Working voltage
  (signal + common mode) ..... ±10 V of AGND
- Absolute accuracy ..... ±200 mV
- Bandwidth ..... 20 kHz minimum, >50 kHz typical

Audio input

- Configuration ..... One stereo input consisting of two AC-coupled, single-ended channels
- Input impedance ..... 10 kΩ at DC
- Nominal range ..... ±2.5 V
- Bandwidth ..... 2 Hz to >20 kHz

Analog Output

Aggregate maximum update rates

- All AO channels on MXP connectors ..... 345 kS/s
- All AO channels on MSP connector and audio output channels ..... 345 kS/s

Resolution ..... 12 bits

Overload protection ..... ±16 V

Startup voltage ..... 0 V after FPGA initialization

MXP connectors

- Configuration ..... Two single-ended channels per connector
- Range ..... 0 V to +5 V
- Absolute accuracy ..... 50 mV
- Current drive ..... 3 mA
- Slew rate ..... 0.3 V/µs

MSP connector

- Configuration ..... Two single-ended channels
- Range ..... ±10 V
- Absolute accuracy ..... ±200 mV
- Current drive ..... 2 mA
- Slew rate ..... 2 V/µs

Audio output

- Configuration ..... One stereo output consisting of two AC-coupled, single-ended channels
- Output impedance ..... 100 Ω in series with 22 µF
- Bandwidth ..... 70 Hz to >50 kHz into 32 Ω load; 2 Hz to >50 kHz into high-impedance load

Digital I/O

Number of lines

- MXP connectors ..... 2 ports of 16 DIO lines (one port per connector); one UART.RX and one UART.TX line per connector
- MSP connector ..... 1 port of 8 DIO lines

Direction control ..... Each DIO line individually programmable as input or output

Logic level ..... 5 V compatible LVTTL input; 3.3 V LVTTL output

Input logic levels

- Input low voltage, V_{IL} ..... 0 V min; 0.8 V max
- Input high voltage, V_{IH} ..... 2.0 V min; 5.25 V max

Output logic levels

- Output high voltage, V_{OH}
  sourcing 4 mA ..... 2.4 V min; 3.465 V max
- Output low voltage, V_{OL}
  sinking 4 mA ..... 0 V min; 0.4 V max

Minimum pulse width..... 20 ns

Maximum frequencies for secondary digital functions

- SPI ..... 4 MHz
- PWM..... 100 kHz
- Quadrature encoder input ..... 100 kHz
- I^2C..... 400 kHz

UART lines

- Maximum baud rate..... 230,400 bps
- Data bits..... 5, 6, 7, 8
- Stop bits 1, 2
- Parity..... Odd, Even, Mark, Space
- Flow control..... XON/XOFF

Accelerometer

Number of axes..... 3

Range ..... ±8 g

Resolution..... 12 bits

Sample rate ..... 800 S/s

Noise..... 3.9 mg_{rms} typical at 25 °C

Power Output

+5 V power output

- Output voltage ..... 4.75 V to 5.25 V
- Maximum current on each connector ..... 100 mA

+3.3 V power output

- Output voltage ..... 3.0 V to 3.6 V
- Maximum current on each connector ..... 150 mA

+15 power output

Output voltage.....+15 V to +16 V

Maximum current .....32 mA (16 mA during startup)

-15 V power output

Output voltage.....-15 V to -16 V

Maximum current .....32 mA (16 mA during startup)

Maximum combined power from +15 V and -15 V power output .....500 mW

Power Requirements

NI myRIO-1900 requires a power supply connected to the power connector.

Caution You must use either the power supply provided in the shipping kit, or another UL Listed ITE power supply marked *LPS*, with the NI myRIO-1900.

Power supply voltage range.....6-16 VDC

Maximum power consumption.....14 W

Typical idle power consumption.....2.6 W

Environmental

To meet these specifications, you must operate the NI myRIO-1900 with the window facing away from the mounting surface and ensure that there is at least 1 in. of clearance in front of the window during use.

Ambient temperature near device (IEC 60068-2-1, IEC 600682-2).....0 to 40 °C

Storage temperature (IEC 60068-2-1, IEC 600682-2).....-20 to 70 °C

Operating humidity (IEC 60068-2-56) .....10 to 90% RH, noncondensing

Storage humidity (IEC 60068-2-56).....10 to 90% RH, noncondensing

Maximum altitude.....2,000 m

Pollution Degree (IEC 60664) 2

Indoor use only.

Physical Characteristics

Weight.....193 g (6.8 oz)