Universidade de Aveiro
2008
Departamento Electrónica, Telecomunicações e Informática

Rui Pedro dos Santos Nunes

Programação de um Sistema de Inspecção Artificial Dinâmica

Universidade de Aveiro 2008 Departamento Electrónica, Telecomunicações e Informática

Rui Pedro dos Santos Nunes

Programação de um Sistema de Inspecção Artificial Dinâmica

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Eng.ª Electrónica e Telecomunicações, realizada sob orientação científica do Prof. Dr. António Ferreira Pereira de Melo, Prof. Catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Prof. Dr. Telmo Reis Cunha, Prof. Auxiliar convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri

presidente

Prof. Dr. Alexandre Manuel Moutela Nunes da Mota
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática
Da Universidade de Aveiro

vogais

Prof. Dr. Carlos Alberto Caridade Monteiro Couto
Professor Catedrático do Departamento de Electrónica Industrial da escola de Engenharia
da Universidade do Minho
(Professor Convidado)

Prof. Dr. António Ferreira Pereira de Melo
Professor Catedrático do Departamento de Electrónica, Telecomunicações e Informática
da Universidade de Aveiro
(Orientador)

Prof. Dr. Telmo Reis Cunha
Professor Auxiliar convidado do Departamento de Electrónica, Telecomunicações e Informática
da Universidade de Aveiro
(Co-orientador)

agradecimentos

Em primeiro, gostaria de agradecer ao Prof. Dr. António Ferreira Pereira de Melo e ao Prof. Dr.Telmo Reis Cunha, já que sem eles não teria sido possível realizar este trabalho.

De seguida, gostaria de agradecer ao Sr. Carlos pela disponibilidade, não só, no esclarecimento de dúvidas mas também do material emprestado, que, mais uma vez, sem ele, não teria sido possível executar este trabalho.

Não me poderia esquecer, claro, de todos os colaboradores da empresa Selmatron, pela forma atenciosa como me receberam, estando sempre disponíveis para me ajudarem.

Por último, um agradecimento especial para a minha família, pais e irmão, pela forma como sempre me apoiaram ao longo deste trabalho.

A todos eles, o meu sincero obrigado.

palavras-chave

Inspecção dinâmica, Reconhecimento de caracteres (OCR), identificação de padrões, Labview.

resumo

Este trabalho tem como objectivo, apresentar dois sistemas capazes de fazer o controlo de qualidade e selecção de objectos. Pretende-se que o trabalho seja a base de um sistema capaz de integrar uma linha industrial de produção e que seja capaz de identificar certos aspectos no produto que está a inspeccionar.

Visando esses mesmos objectivos, planeou-se um sistema capaz de reconhecer a entrada de um objecto numa passadeira rolante e de seguir esse mesmo objecto com o intuito de ser capaz de identificar padrões e de reconhecer caracteres. Já num segundo sistema, o pretendido era que a câmara se deslocasse entre quatro pontos precisos com o propósito de inspeccionar caracteres impressos em mangueiras, estando estas em movimento. Foi, então, necessário construir algoritmos para controlar motores passo-a-passo e ainda para fazer o processamento de imagem.

Por fim, falta apenas referir que toda a programação desenvolvida neste trabalho foi em ambiente Labview

keywords Dynamic inspection, Optical characters recognition (OCR), Pattern identification, Labview

abstract

This work aims to describe two systems intended to realize on line quality control and pattern recognition. The intention is to establish the base of a system, to be introduced in an industrial production line, able to identify certain characteristics of the products passing through.

Having these objectives in mind, we projected a system capable of recognizing the entry of an object in a crosswalk treadmill and follow it, in order to identify patterns and recognize characters. In a second system, it was desired that the camera moves up between four distinct points with in order to recognize the characters printed on a hose in constant motion. So, became necessary to develop algorithms to control stepper motors and to do the image processing.

All the programs were developed using the Labview environment.

Índice

Índice de Figuras

1. INTRODUÇÃO

1.1. ENQUADRAMENTO

Hoje em dia, em qualquer tipo de indústria a qualidade do produto final define a competitividade ou não dessa mesma indústria. Isto porque, para além de outros factores, o preço do produto final também é definido em função dos seus defeitos. Um produto com um número elevado de defeitos faz com que os custos de produção aumentem, visto que os custos de produzir esses mesmos produtos é o mesmo, mas muitas vezes não chegam ao mercado e quando chegam é com um preço reduzido o que diminui a margem de lucro. Pode também acontecer chegarem por lapso, e por consequência diminuírem a credibilidade da própria empresa.

Actualmente, a inspecção final dos produtos ainda é muitas das vezes efectuada por um operador. Como este está sujeito a condições de trabalho monótonas e tediosas, o seu trabalho poderá muitas das vezes degradar-se com o passar das horas de trabalho. Há, portanto, todo o interesse que este tipo de tarefas seja realizado automaticamente e sempre de uma forma rigorosa. É por isso vantajoso instalar um sistema de inspecção automático, onde o operador só terá que se preocupar em definir os vários parâmetros de funcionamento.

Presentemente, já foram desenvolvidos alguns esforços para automatizar a inspecção final através de visão assistida por computador. Muitos destes esforços, há uns tempos atrás, encontravam barreiras tecnológicas difíceis de superar, tais como o tempo de processamento de uma imagem, o que tornava muito lento o processo de inspecção. Com os avanços tecnológicos actuais, como por exemplo, a capacidade e velocidade de processamento, o aparecimento de câmaras de elevada velocidade de aquisição tornou o desenvolvimento de sistemas de inspecção em tempo real possível e com resultados bastante satisfatórios.

Nesta dissertação pretendeu-se desenvolver um sistema automático de reconhecimento de padrões impressos em objectos que se encontram em movimento, cujo objectivo principal é o de reconhecer padrões e/ou caracteres num objecto (OCR).

Para a solução final foi necessário desenvolver algoritmos de controlo e de processamento de imagem, tendo sido estes desenvolvidos em linguagem Labview.

1.1.1. ORGANIZAÇÃO DO DOCUMENTO

Neste primeiro capítulo, apresenta-se o enquadramento do trabalho e o que é pretendido com o trabalho.

Já no segundo capítulo, faz-se a descrição dos sistemas, do problema e ainda dos vários componentes que compõem os sistemas.

No terceiro capítulo, introduz-se alguns conceitos teóricos necessários a compreensão do trabalho.

Seguidamente, no quarto capítulo, descreve-se a construção da solução explicando os vários algoritmos necessários ao funcionamento do sistema final.

No quinto capítulo, apresenta-se os interfaces gráficos que servem para o utilizador interagir com o sistema.

Já no sexto capítulo, descreve-se os resultados obtidos no trabalho final.

Por último, no sétimo capítulo apresenta-se as conclusões resultantes deste trabalho, assim como sugestões de trabalho futuro.

2. APRESENTAÇÃO DOS SISTEMAS

2.1. DESCRIÇÃO DOS SISTEMAS DE TESTE

O desenvolvimento do trabalho aqui descrito assentou no estudo de dois casos práticos – dois sistemas constituintes de processos de produção com topologia de linha de montagem. Descrevem-se em seguida, estes dois sistemas analisados. Salienta-se que a forma modular que foi seguida na concepção permitiu que o código fosse adaptado de uma forma simples a ambos os sistemas de teste.

O primeiro sistema consistia numa passadeira rolante onde se encontram os objectos a inspeccionar. Esta é accionada por um motor AC assíncrono ao qual está acoplado um codificador destinado a fornecer o valor da velocidade a que os objectos se deslocam. Por cima da passadeira está montado um sistema de carris sobre os quais se movimenta uma câmara. Esta, para além de se movimentar paralelamente à direcção do movimento do objecto, também se movimenta transversalmente. Tanto o movimento longitudinal como o transversal são executados através do controlo de dois motores passo-a-passo.

Na entrada da passadeira encontra-se uma fotocélula que informa sobre a entrada de cada objecto. No sistema encontram-se mais 4 sensores indutivos, os quais se destinam a identificar os limites possíveis dos movimentos da câmara. A controlar o movimento da câmara encontra-se um controlador electrónico com entradas e saídas digirais (podendo estas variar entre 0 e 5V).

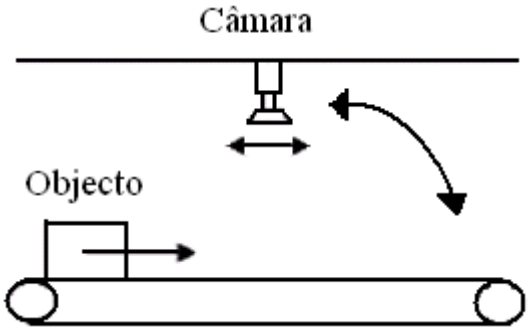


Figura 1 – Esquema do primeiro sistema

Já o segundo sistema consiste, na inspecção de caracteres impressos em mangueiras. Existem quatro tipos de mangueiras com diâmetro diferente que estão dispostas paralelamente umas as outras. Por cima destas encontra-se a calha onde se desloca a câmara sendo o movimento desta transversal ao das mangueiras. A câmara vai-se deslocando entre os quatro tipos de mangueiras e com estas em movimento (sendo usado novamente um codificador para medir o deslocamento) vai tirando fotos de modo a confirmar que são impressos na mangueira os caracteres correctos.

Figura 2 – Sistema utilizado para o reconhecimento de caracteres nas mangueiras

2.2. DESCRIÇÃO DO PROBLEMA

Mais uma vez, nesta parte, descrever-se-á os problemas relativos aos dois sistemas de teste, que são análogos.

Nesta dissertação, o pretendido é programar um sistema de inspecção artificial dinâmico aplicável à indústria. Concretamente, para o primeiro sistema, o que se terá que fazer é adquirir duas imagens de um objecto, uma da face superior e outra da face lateral. Para tal, terá que haver um sincronismo entre a posição da câmara, uma vez que esta não se encontra fixa, e o momento de aquisição das imagens. Pode-se então dividir o problema em duas partes, que são o controlo do deslocamento da câmara e o sincronismo descrito anteriormente para o posterior processamento da imagem.

Relativamente ao primeiro problema, o que se terá que fazer é o seguinte: após detectada a entrada de um objecto, independentemente da posição da câmara, esta terá que se colocar sobre o objecto e depois seguir paralelamente a este à mesma velocidade.

Quanto ao processamento de imagem, o pretendido é reconhecer padrões e caracteres.

Já para o segundo sistema terá que se tirar um número determinado de fotos dependendo do número de caracteres que se pretende imprimir na mangueira. De novo, aqui terá que haver um sincronismo entre o deslocamento da câmara e o momento da aquisição das fotos. A diferença é que aqui o movimento da câmara é menos complexo do que no sistema anterior, ou seja, aqui a câmara só se terá que deslocar de um ponto para o outro e esperar a aquisição das imagens. Esta é coordenada através da contagem de impulsos do codificador, controlando assim o deslocamento das mangueiras.

Quanto ao processamento de imagem, neste segundo caso só se pretende o reconhecimento de caracteres.

2.3. DESCRIÇÃO DOS COMPONENTES DO SISTEMA

Os componentes utilizados na montagem do sistema (neste caso do segundo sistema que foi o único a ser completamente montado) foram os seguintes:

- 1 motor passo-a-passo;

- 1 câmara;

- 2 sensores indutivos;

- 1 codificador;

- 1 motor AC.

- 1 controlador NI crio-9002

Figura 3 – Hardware utilizado

Em seguida apresenta-se uma descrição dos elementos mais importantes do sistema, assim como do seu funcionamento.

2.3.1. CÂMARA

A câmara utilizada foi a *imaging DFK 21BF04-Z.H* da Sony. Em seguida apresenta-se as suas características técnicas mais importantes:

- Dimensões: H: 50.6 mm, W: 50.6 mm, L: 130 mm;

- tensão de alimentação: 8~30VDC;

- iluminação própria;

-massa: 380g;

- zoom;

- formatos de vídeo @ Frame rate:

  640 x 480 UYVY @ 30, 15, 7.5, 3.75 fps

  640 x 480 BY8 @ 60, 30, 15, 7.5, 3.75 fps

- controlo automático/manual da *Íris* e *Foccus;*

Figura 4 – câmara imaging DFK 21BF04-Z.H

- controlo automático/manual de *Shutter*, *Gain*, *Offset* e *White balance*;

- interface de comunicação: *Firewire*.

Figura 5 – Interface Firewire da câmara

2.3.2. CONTROLADOR NI CRIO-9002

O cRIO (Compact Real Input Output) possui uma arquitectura modular sendo constituído por uma FPGA, um processador de tempo real e módulos isolados de I/O.

Através da FPGA é feito o acesso de baixo nível, controlando-se as saídas e entradas e ainda permite fazer controlo de algumas tarefas mais prioritárias. Cada módulo I/O contém electrónica de conexão aos sensores e electrónica de acondicionamento de sinal e circuitos de conversão (DAC ou ADC).

2.3.3. MÓDULO DE PROCESSAMENTO DE TEMPO-REAL

Figura 6 - Módulo de processamento cRIO

O NI cRIO-9002 da National Instruments é um controlador de tempo-real "embedded". Possui um processador Pentium de 195 MHz, 32 MB de memória DRAM e 64 MB CompactFlash não volátil para armazenamento de ficheiros. Contém ainda, para comunicação, uma porta Ethernet 10/100 Mb/s e uma porta RS-232. Tem já incorporado um servidor HTTP e um servidor FTP. As tensões de alimentação aceites variam entre 6 e 35 VDC.

2.3.3.1. CHASSI RECONFIGURÁVEL

O chassi é uma das partes vitais do cRio já que contém uma FPGA reconfigurável. A FPGA tem uma conexão individual para cada um dos módulos I/O. É programada muito facilmente com funções de ler e escrever informação de cada módulo. Pode fazer processamento de sinal e tomar decisões e passar directamente de um módulo para o outro. A FPGA está também ligada com o controlador de tempo real CompactRIO através de um barramento PCI. Este pode dar informação de qualquer controlo ou indicador da FPGA. Pode, ainda, gerar interrupções de modo a ficar sincronizada com o módulo de tempo real.

Figura 5 – Chassi do CRIO

2.3.3.2. CARTAS DE AQUISIÇÂO DE SINAL

Figura 6 – Carta de aquisição de sinal

A carta de aquisição NI 9401 possui 8 canais digitais bidireccionais. Quer isto dizer que elas podem ser programadas para serem entradas ou saídas. É neste componente que se encontra toda a electrónica de acondicionamento de sinal e circuitos de conversão (DAC ou ADC). Os níveis de tensão aceites por esta carta específica são os 0 e 5 V.

2.3.4. MOTOR PASSO-A-PASSO

Figura 7- Motor passo-a-passo e respectivo driver

O motor utilizado no sistema foi o PK599-AE da ORIENTAL MOTORS. Este é um motor passo-a-passo de 5 fases, cujo deslocamento por passo é de 0,72°. É construído de maneira a produzir um elevado binário, a ter baixa vibração e serem o mais silencioso possível. As cinco fases do motor conectam-se a um driver ao qual, para se controlarem os motores, apenas se tem que enviar dois sinais digitais. Um deles, dependendo do nível, controla a direcção de rotação e o outro sinal controla a velocidade. Este último sinal é um PWM que dependendo da duração do período do PWM aumenta ou diminui a velocidade. Para que o material se comporte de maneira devida, é necessário respeitar alguns limites de tempo, que estão representados no diagrama temporal da figura seguinte:

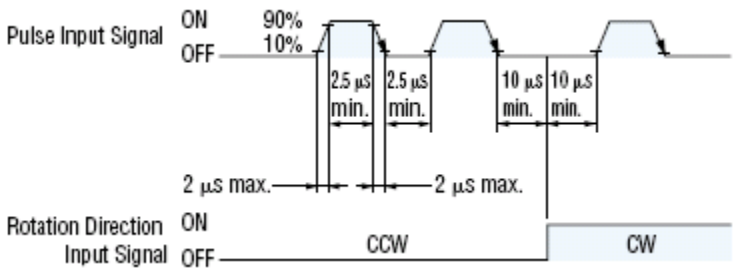


Figura 8 - Diagrama temporal dos sinais enviados para o driver

2.4. INTERLIGAÇÃO DOS COMPONENTES DO SISTEMA

O seguinte diagrama de blocos descreve a comunicação entre os vários elementos do sistema:

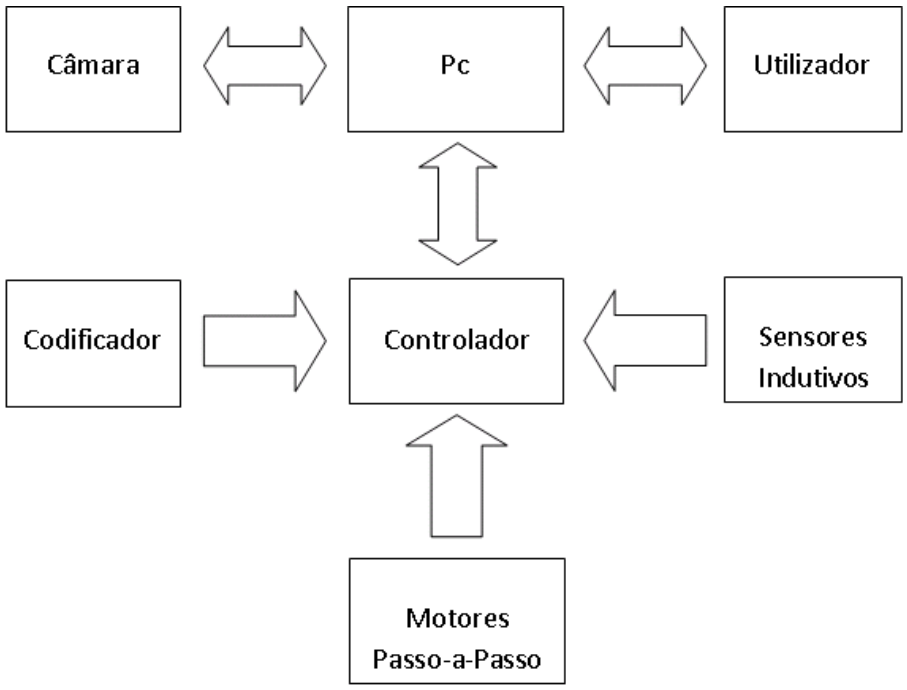


Figura 9 - Diagrama com interligação dos vários blocos constituintes do sistema

Essencialmente, há dois elementos principais, são eles o PC e o controlador. São estes dois componentes os responsáveis pela coordenação e sincronismo dos restantes elementos.

A câmara, como já foi referido, é ligada ao PC através de Firewire. As imagens adquiridas pela câmara são enviadas para o PC, sendo este o encarregado pelo seu processamento. É, igualmente, no PC que o utilizador controla todo o sistema através de interfaces gráficos.

Já o controlador é o responsável pelos movimentos da câmara (sendo estes limitados através da leitura dos sensores indutivos que se encontram nos limites possíveis

do mesmo), tendo para o primeiro caso, que garantir que esta vá ao encontro do objecto e posteriormente o siga à mesma velocidade. Para isso, o controlador quantifica o valor da velocidade do objecto através da leitura do número de impulsos por unidade de tempo dados pelo codificador e aplica os sinais de controlo no driver do motor. Então, este encarrega-se de fazer a rotação do motor passo-a-passo.

Já no segundo sistema, o controlador terá, mais uma vez, que controlar o movimento da câmara deslocando-a de um ponto para o outro e ainda sincronizar o momento da aquisição das fotos, contando os impulsos do codificador, com o deslocamento das mangueiras.

2.5. SOFTWARE UTILIZADO

2.5.1. LABVIEW

O Labview (Laboratory Virtual Instrument Engineering Workbench) é uma linguagem de programação gráfica (sendo designada por "G") pertencente a National Instruments, que usa ícones em vez de linhas de texto para criar aplicações.

A primeira versão surgiu em 1986 para o Macintosh. Hoje existem ambientes de desenvolvimento também para Windows, Linux e Solaris. Está vocacionada essencialmente para efectuar medições e para automação.

Os programas feitos em Labview são chamados de VIs (virtual instruments), devido a sua aparência, muito similar a instrumentos, tais como, osciloscópios, multímetros e outros. Cada VI é composto por três elementos:

- Front Painel – serve como interface do utilizador;

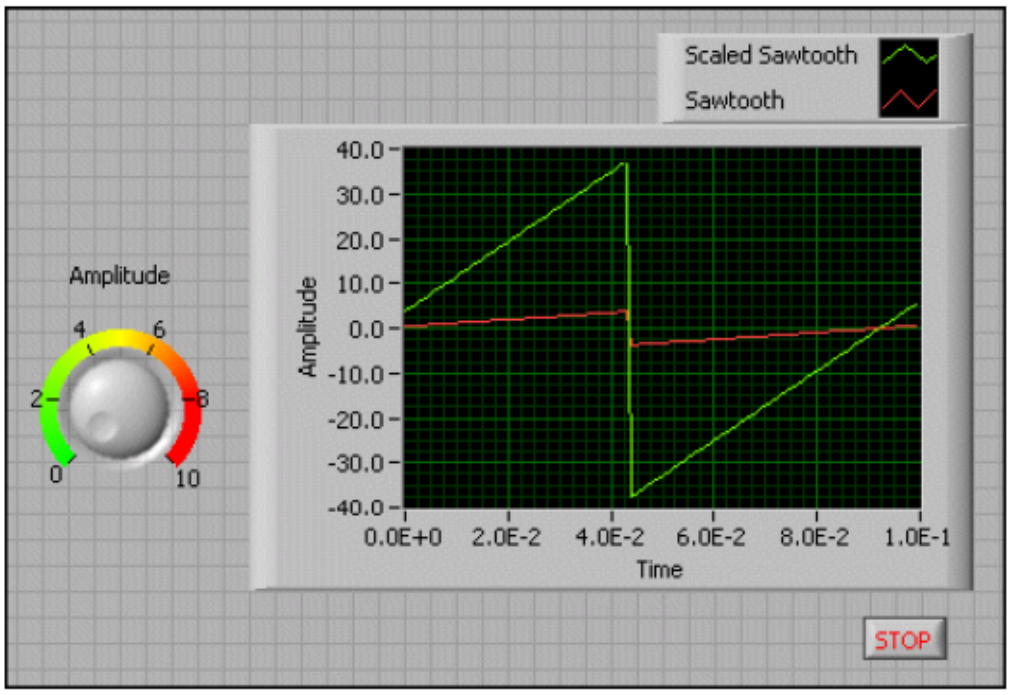


Figura 10 - Exemplo de Front Painel

- Block diagram – é onde o código gráfico é desenvolvido;

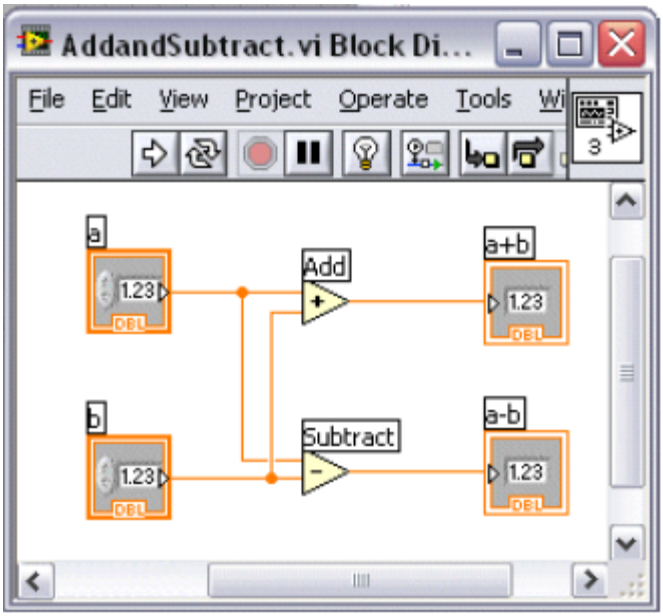


Figura 11 - Exemplo de block diagram

- Icon – identifica o VI e permite que este seja usado em outros VIs (é o equivalente as sub rotinas nas linguagens de linhas de texto).

Figura 12 - Ícone que identifica o VI

MÉTODO DE PROGRAMAÇÂO

Ao contrário das linguagens de texto onde a sequência de instruções dita a ordem pela qual é executado o programa, o Labview usa fluxo de dados (*data flow*), o que significa que a chegada de dados a um determinado nó determina a ordem de execução, ou seja, o programador na construção da sua aplicação vai conectando diferentes VIs, definindo assim, a ordem pela qual o programa vai ser executado. Se um VI possui várias entradas somente inicia a sua execução, quando todas as entradas já estão disponíveis. Uma vantagem deste método, é que pode haver vários processos a serem executados em paralelo.

Um outro aspecto interessante, é o facto de muitos VIs e funções serem polimórficas, isto é, são capazes de aceitar vários tipos de dados a entrada, o que evidencia a grande versatilidade inerente a programação em Labview. Por exemplo, o VI "*Build Array*" pode ser usado para a construção de um *array* de *strings* ou mesmo de inteiros, sem qualquer tipo de problema.

Relativamente a construção do interface gráfico, é muito simples através do *Front Painel* que disponibiliza um conjunto de controlos (tais como botões interruptores…) e indicadores (tabelas, gráficos, leds…) e mais uma vez sem escrever qualquer linha de código, aumentando assim a facilidade de leitura dos dados.

Por último falta dizer, que para além do Labview, propriamente dito, existem outros pacotes de software específicos para determinadas tarefas que facilitam em muito o desenvolvimento do código. Especificamente nesta dissertação utilizaram-se os seguintes:

FPGA module – permite criar código para uma FPGA no ambiente gráfico do Labview sem ter qualquer tipo de conhecimento das tradicionais HDL (Hardware Description Language), como VHDL.

Real-time module – com este módulo, cria-se código que tem que ser executado em tempo real e de forma determinística. Está vocacionado para fazer aquisição de dados em tempo-real assim como controlo de sistemas.

Vision development module – é uma colecção de funções de processamento de imagem e de visão, que facilita a criação de código, de entre muitas outras coisas, para o reconhecimento de padrões e de caracteres (OCR).

3. CONCEITOS TEÓRICOS

Nesta secção serão abordados alguns conceitos necessários para a compreensão da restante dissertação.

3.1. PROCESSAMENTO DE IMAGEM

Processar uma imagem significa retirar a informação pretendida, que nela está contida, sendo por vezes necessário modificá-la para facilitar a tarefa. Por isso, o processamento de imagens aborda temas como realce, filtragem, restauração, análise e reconstrução. Todas estas técnicas visam diminuir alguns defeitos no momento da aquisição da imagem, como por exemplo, iluminação deficiente ou captação de algum ruído ou então, evidenciar certas características da imagem.

De todas as técnicas mencionadas atrás apenas serão descritas, as utilizadas nesta dissertação.

3.1.1. REALCE DO CONTRASTE

Uma das principais dificuldades em visão artificial é a capacidade de adaptação automática às mudanças de iluminação. A aptidão de um sistema para compensar efeitos, tais como, sombras ou pontos muito brilhantes “hot-spots” pode determinar o sucesso dos algoritmos subsequentes. Tais efeitos, conjuntamente com a não homogeneidade da iluminação e ainda a abertura incorrecta do diafragma da câmara durante a aquisição da imagem afecta o contraste desta. Torna-se, por isso, evidente a importância da utilização desta técnica no processamento de imagem.

Dos vários métodos existentes para resolução deste problema, o utilizado nesta dissertação foi o da equalização do histograma, o qual vai ser descrito em seguida.

3.1.1.1. EQUALIZAÇÂO DO HISTOGRAMA

O histograma de uma imagem representa para cada nível de intensidade, o número de pixéis com aquele nível. Matematicamente, é descrito pela seguinte expressão:

$$H(r_k) = n_k$$

onde r_k é o k-ésimo nível de intensidade luminosa possível de uma faixa que varia de [0, L-1] e n_k é o número de pixéis contendo o nível de intensidade luminosa igual a r_k na imagem.

Pode ser igualmente descrito pela sua forma normalizada:

$$P(r_k) = n_k/n$$

onde n é o numero total de pixéis da imagem.

Deste pode-se retirar, a distribuição dos diferentes níveis de intensidade numa imagem e aferir se esta é escura ou clara e até sobre o seu contraste. Num histograma onde haja uma concentração em um determinado intervalo de níveis de intensidade significa que a imagem de onde se retirou esse histograma tem pouco contraste. Se essa concentração se encontrar no inicio da escala de intensidade a imagem é escura, se pelo contrário se encontrar no final da escala a imagem é clara, como se pode observar nos exemplos a seguir:

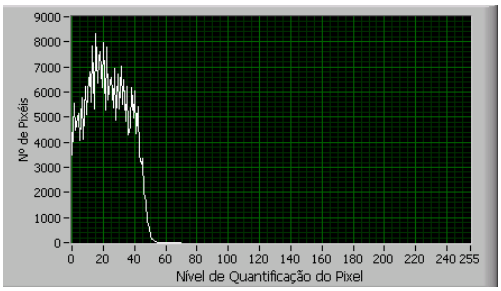


Figura 13 - Exemplo de um histograma de uma imagem escura

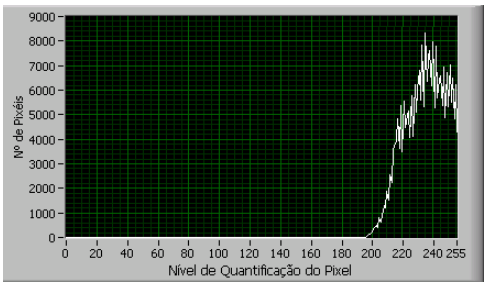


Figura 14 - Exemplo de um histograma de uma imagem clara

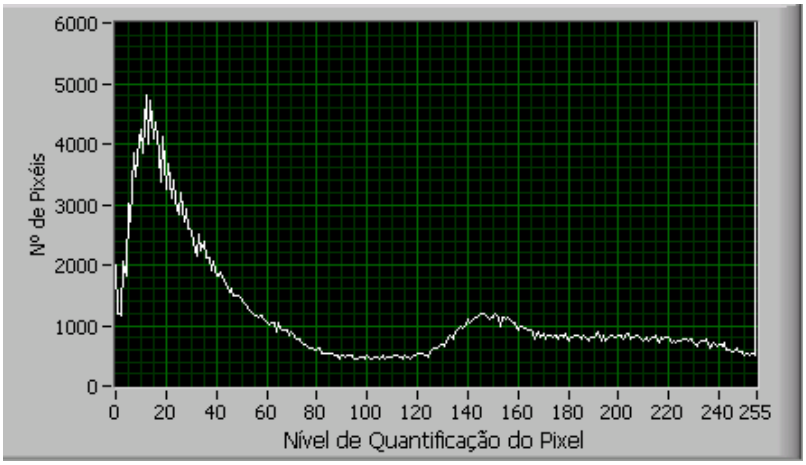


Figura 15 - Exemplo de um histograma de uma imagem com grande contraste

O que se faz, então, na equalização do histograma é uma redistribuição dos níveis de intensidade luminosa dos seus pontos, de forma a atingir uma distribuição mais uniforme de uma faixa ou de toda a faixa de intensidades luminosas. Sendo assim, imagens escuras de níveis de intensidade luminosa predominantemente baixos teriam estes valores de intensidade redistribuídos, gerando um histograma mais uniforme, ocupando maior parte da vasta faixa de níveis de intensidade luminosa. Isto resultaria numa imagem mais clara e nítida, com maior contraste entre os objectos e o fundo da imagem. O mesmo princípio pode ser aplicado às imagens muito claras, procurando realçar as características mais importantes, para permitir futuros processamentos.

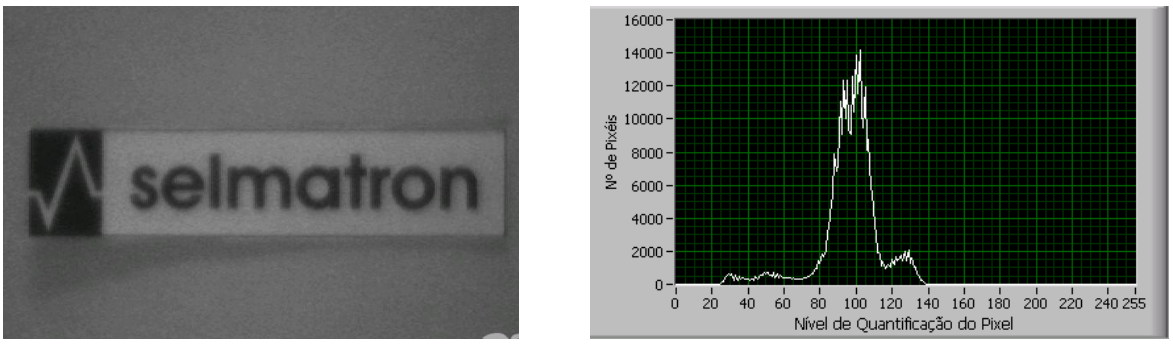


Figura 16 - Imagem antes de se aplicar a equalização do histograma

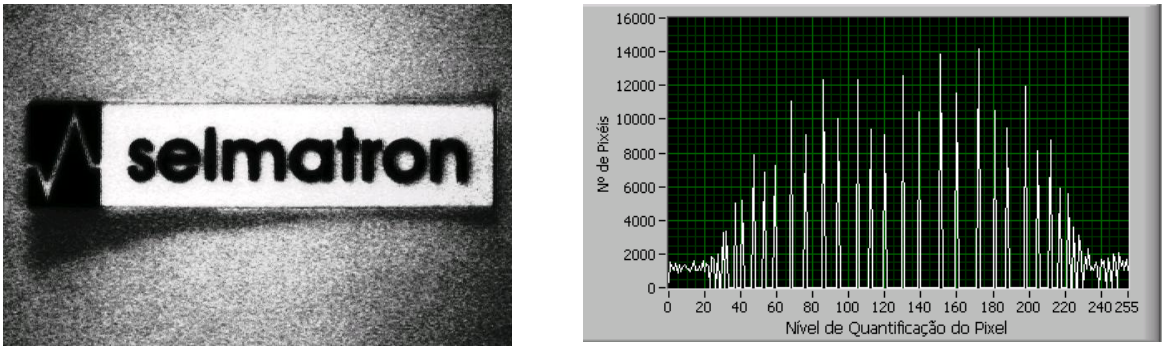


Figura 17 - Imagem após a aplicação da equalização do histograma

Como se pode observar, a equalização do histograma é um bom método para um automático realce do contraste uma vez que é baseada numa função de transformação que depende do histograma da imagem. Contudo é um método limitado, uma vez que, a sua única função é linearizar o histograma. Processo este, que só é aplicável quando o histograma possui uma determinada forma.

3.1.2. SEGMENTAÇÂO

A segmentação é o processo de dividir uma imagem nos elementos distintos que a constituem. É uma das etapas mais importantes do processamento de imagem, visto que, é nesta parte que os objectos da imagem são extraídos para consequente reconhecimento e análise. Em geral, os algoritmos de segmentação baseiam-se em dois

princípios básicos que são: descontinuidade e similaridade. Um dos algoritmos de segmentação é a utilização de um limiar de threshold, que vai ser explicada em seguida.

3.1.2.1. THRESHOLD

A utilização de um limiar de treshold refere-se à definição de um valor ou de uma função, que permita separar elementos de interesse na imagem, do fundo. A operação deste método consiste em definir um valor para uma determinada gama de valores dos pixéis da imagem e um outro valor para a restante gama, como se exemplifica nas equações seguinte:

$$I(m,n) = \begin{cases} 0, & O(m,n)<L \\ & \text{com } m,n = 1,2,3\ldots K \\ 255, & O(m,n)>L \end{cases}$$

Onde, L é um limiar definido para o qual o pixel da imagem O(m,n) toma o valor 0 ou o valor 255 (isto para uma imagem cujo quantificação seja de 8 bits).

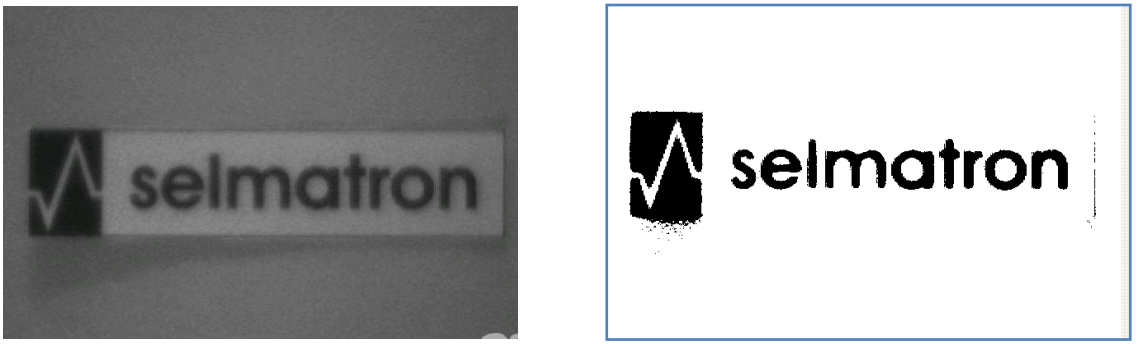


Figura 18 - Exemplo da aplicação do limiar de threshold

Pode-se, então transformar a imagem original numa imagem binária. Este método pode ser considerado como um dos mais simples para a segmentação de imagens.

3.1.3. DETECÇÃO POR CORRESPONDÊNCIA DE PADRAO (*TEMPLATE MATCHING)*

Uma das técnicas mais utilizadas para detectar um objecto numa imagem é a correspondência por padrão (*template matching*), na qual uma réplica do objecto (padrão) que se quer detectar é comparada com todos os objectos desconhecidos da imagem. Se houver uma correspondência entre o padrão e o objecto satisfatória, isto é, dentro de uns certos limites previamente estabelecidos, o objecto pretendido é identificado.

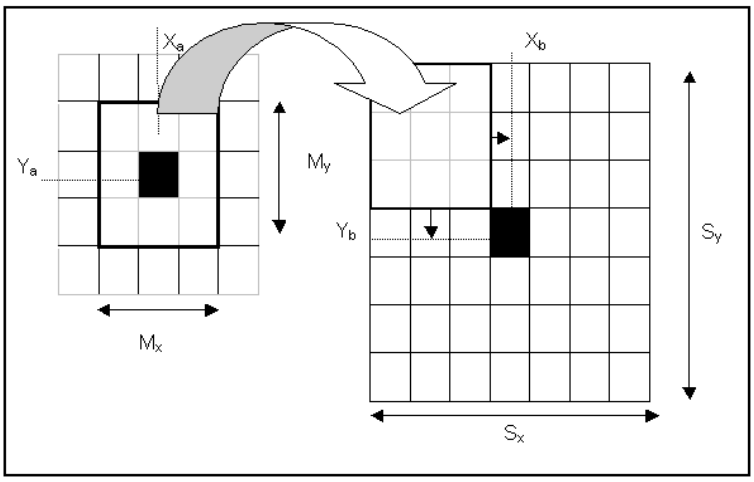


Figura 19 - Processo da correspondência por padrão

No processo de detecção o padrão é sequencialmente comparado, em termos de similaridade, com porções da imagem onde se pretende encontrar o objecto pretendido como se exemplifica na imagem anterior.

Uma correspondência por padrão raramente é exacta devido ao ruído e até aos efeitos de quantificação. Deste modo, um procedimento comum é obter uma medida de diferença D(m,n) (onde –M<m<M e –N<n<N sendo M e N os limites da porção da imagem que está a ser comparada com o padrão) de todos os pontos do padrão e o campo da imagem que está a ser comparado. Há uma correspondência quando a diferença for menor que um determinado limiar L. Normalmente a medida de diferença é o erro quadrático médio definido por:

$$D(m,n) = \sum_j \sum_k [F(j,K) - T(j-m,k-n)]^2 \qquad \text{(eq.1)}$$

onde, F(m,n) representa o campo onde se efectua a pesquisa e T(j,k) é o padrão.

Escrevendo a eq.1 de outra maneira tem-se:

$$D(m,n) = D_1(m,n) - 2D_2 + D_3(m,n)$$

onde,

$$D_1 = \sum_j \sum_k [F(j,k)]^2$$

$$D_2 = \sum_j \sum_k I(j,k)T(j-m,k-n)$$

$$D_3 = \sum_j \sum_k [T(j-m,k-n)]^2$$

O termo D_3(m,n) representa a energia do padrão e tem valor constante independentemente das coordenadas (m,n). Já D_1(m,n) é o valor da energia da porção da imagem onde se está a realizar a pesquisa, e geralmente, vai variando suavemente pela imagem. O segundo termo deve ser reconhecido como a correlação cruzada entre o padrão e o campo da imagem. Numa posição onde exista correspondência de padrão, a correlação cruzada deverá conduzir a uma pequena diferença (D(m,n)). Contudo a magnitude da correlação cruzada nem sempre é uma medida adequada isto porque por vezes, mesmo em uma condição de não correspondência, pode haver partes do campo de imagem que está a ser comparado muito semelhante ao padrão, o que faz, logo, aumentar o valor da correlação cruzada. Esta dificuldade é superada através da correlação cruzada normalizada.

$$\tilde{F}_{IT}(m,n) = \frac{D_2(m,n)}{D_1(m,n)} = \frac{\sum_j \sum_k I(j,k)T(j-m,k-n)}{\sum_j \sum_k [I(j,k)]^2}$$

Logo, existe uma correspondência quando $\tilde{F}_{IT}(m,n)$ é maior que um determinado limiar L pré-definido.

Por fim, falta frisar a desvantagem do método, que é o grande número de comparações que se têm que efectuar para ter em conta a rotação e a mudança de escala dos objectos que se pretende identificar. Por esta razão a correspondência por padrão é, geralmente, usada em imagens pequenas onde exista, pouca possibilidade de rotação ou de mudança de escala do objecto.

3.2. MOTORES PASSO-A-PASSO

Apesar de este trabalho não incidir directamente sobre algum tipo de estudo sobre motores passo-a-passo, estes são um elemento importante na construção do sistema e por isso, pretende-se aqui fazer uma breve descrição sobre alguns aspectos dos mesmos.

O motor passo-a-passo é um dispositivo electromecânico que converte impulsos eléctricos em movimentos angulares bem definidos. Esses movimentos assim como a direcção para a qual roda o eixo, dependem de como são aplicados os impulsos eléctricos. É por isso, uma boa opção, quando se trata de sistemas onde é necessário controlar o ângulo de rotação, velocidade ou mesmo posição.

-PRINCÍPIO DE FUNCIONAMENTO

Há vários tipos de motor passo-a-passo, em seguida vai-se ilustrar o funcionamento de um dos de mais fácil compreensão.

Um motor passo-a-passo opera de uma forma diferente em relação a um motor convencional DC, que roda quando uma diferença de potencial é aplicada aos seus

terminais. Ao contrário, os motores passo-a-passo possuem no estator vários electroímanes “dentados” (como se ilustra nas figuras abaixo) que rodeiam uma peça de ferro igualmente “dentada”.

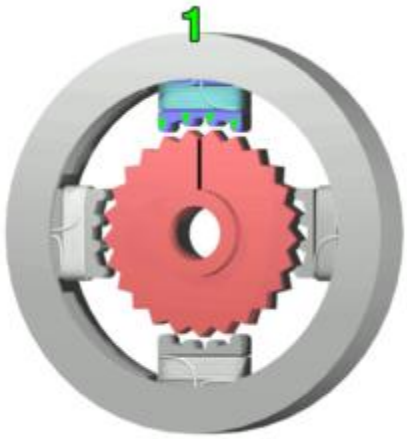


Figura 20 – Primeiro passo na rotação do motor

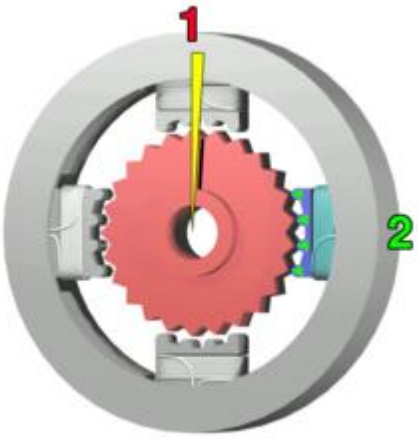


Figura 21 – Segundo passo na rotação do motor

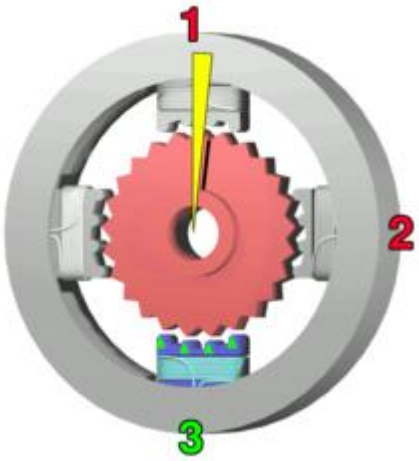


Figura 22 – Terceiro passo na rotação do motor

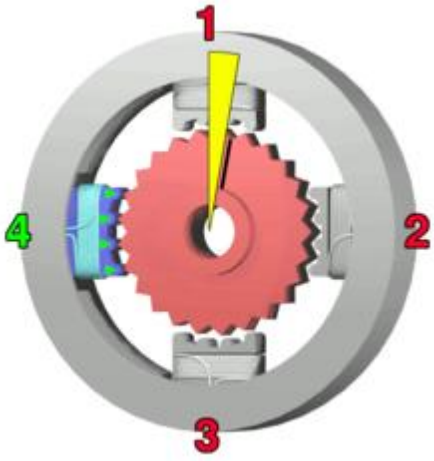


Figura 23 – Quarto passo na rotação do motor

Quando é para fazer o motor rodar, é ligado um electroíman (de referir que o controlo e a ordem de ligar os electroímanes é feito através de um circuito de controlo exterior ao motor) o que faz com que os dentes desse electroíman exerçam uma força de atracção no rotor, e este roda ate ficar devidamente alinhado com o electroíman que está ligado, mas ligeiramente desalinhado com um outro electroíman, que quando for ligado

repete todo o processo. E assim, se faz rodar um motor passo-a-passo com deslocamentos angulares bem definidos (que são chamados de “steps”).

Após esta explicação, pode-se enumerar algumas das vantagens do uso dos motores passo-a-passo:

- é preciso no posicionamento e tem uma boa repetibilidade, sendo o erro de um passo para o outro não cumulativo;

- tem uma boa resposta em relação a iniciar, parar e inverter o movimento;

- uma vez, que não existem contactos entre o rotor e o estator, confere um carácter mais duradouro ao motor;

- já que é controlado através de impulsos, faz com que seja possível efectuar o controlo em malha aberta.

3.3. FPGAs

As FPGAs foram introduzidas há já bastante tempo e actualmente constituem componentes típicos de muitos produtos comerciais. São os dispositivos lógicos programáveis de maior capacidade disponível hoje em dia. O número de portas de sistema em FPGAs comerciais tem crescido bastante rapidamente desde a sua introdução nos meados dos anos 80, e atinge actualmente 10M em dispositivos da Xilinx. A arquitectura típica de uma FPGA está representada na fig. 24.

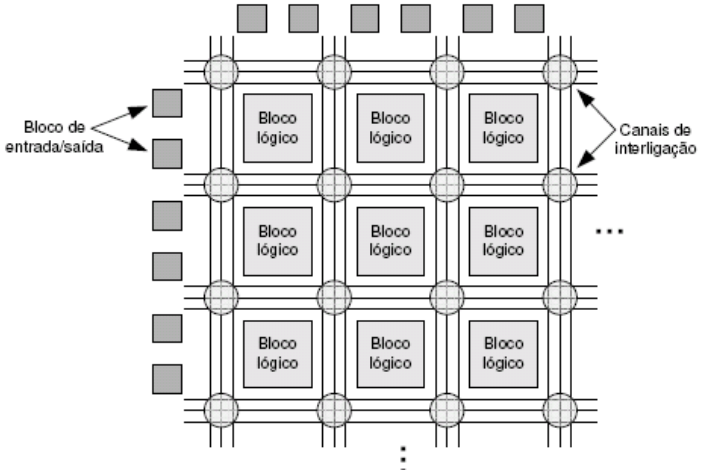


Figura 24 – Arquitectura interna de uma FPGA

Uma FPGA inclui um *array* de blocos lógicos interligados por recursos de encaminhamento e cercado por um conjunto de blocos de entrada/saída, sendo todos estes componentes programáveis pelo utilizador. Os blocos lógicos contêm elementos combinatórios e sequenciais possibilitando a implementação de funções lógicas bem como de circuitos sequenciais. Os recursos de encaminhamento incluem segmentos de pistas de ligação pré-fabricadas e interruptores programáveis. Um circuito lógico é implementado em FPGA ao distribuir a lógica entre os blocos individuais e interligá-los posteriormente com os interruptores programáveis. É de notar que os atrasos resultantes são fortemente influenciados pela distribuição da lógica e pela estrutura de encaminhamento. Portanto, o desempenho de circuitos mapeados em FPGA depende bastante da eficácia das ferramentas CAD (*Computer Aided Design*) utilizadas para a implementação.

Uma das maiores vantagens das FPGAs é a sua arquitectura flexível que serve muito bem para uma ampla gama de aplicações. Estas aplicações incluem implementação de controladores de dispositivos, circuitos de codificação, lógica arbitrária, prototipagem e emulação de sistemas, etc. As FPGAs recentes passaram a incorporar várias estruturas heterogéneas tais como blocos de memória, o que possibilita a implementação de sistemas completos num único encapsulamento.

4. DESENVOLVIMENTO DA SOLUÇÃO

4.1. ESTRUTURA DA SOLUÇÃO

Como já foi mencionado anteriormente, a solução passa por decompor o problema em dois menores, que são eles: o controlo do movimento da câmara e o processamento da imagem. Foi então necessário criar algoritmos para a resolução destes dois problemas menores. Após se resolver estes dois problemas resta apenas sincronizar a aquisição das imagens com o movimento da câmara. De referir, mais uma vez, que os algoritmos de processamento de imagem são executados no computador enquanto os algoritmos de controlo são executados no controlador. Dentro da solução foi ainda necessário desenvolver, algum código para a FPGA com o objectivo de se controlar as saídas e entradas digitais e processar alguns cálculos que necessitavam de uma maior rapidez.

Esta estrutura, serve para ambos os sistemas mencionados no início deste documento.

4.2. PROCESSAMENTO DE IMAGEM

O pretendido, relativamente, ao processamento de imagem era, como já foi mencionado, o reconhecimento de padrões e/ou de caracteres. Foram, por isso, desenvolvidos dois algoritmos a fim de executar as duas tarefas que vão ser descritas em seguida.

4.2.1. RECONHECIMENTO DE PADRÕES

Reconhecimento de padrões é entendido como a caracterização de dados de entrada em classes identificáveis através de extracção de características ou atributos fundamentais.

O reconhecimento de padrões envolve dois níveis de processamento: extracção de características e classificação. A extracção de características consiste da análise dos dados de entrada a fim de extrair e derivar informações úteis para o processo de reconhecimento. O estágio final do reconhecimento de padrões é a classificação, onde através da análise das características da entrada de dados o objecto em análise é declarado como pertencente a uma determinada categoria.

Em seguida, é mostrado o código referente ao reconhecimento de padrões:

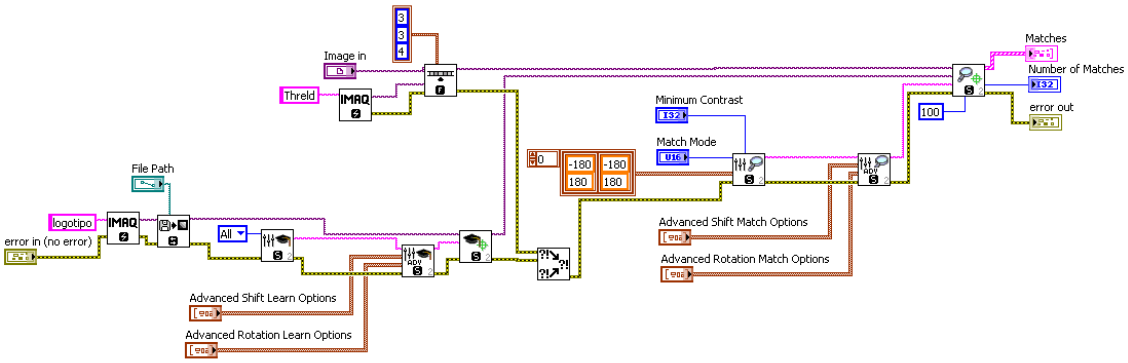


Figura 25 - Código do algoritmo de reconhecimento de padrão

Esta fracção de código faz o seguinte: inicia o processo com a abertura de uma sessão de reconhecimento de padrão. Logo a seguir é carregado para o programa o padrão que se pretende identificar na imagem. Depois a imagem onde é procurado o padrão, que é um parâmetro de entrada, é processada, sendo-lhe aplicado um filtro passa-alto com o objectivo de realçar os contornos. Após estes procedimentos, é feita a pesquisa do padrão na imagem pelo método da correspondência de padrão que já foi anteriormente explicado. No final, é devolvido por este algoritmo, o número de padrões identificados e um array de estruturas referente a cada um dos padrões reconhecidos, com os seguintes campos: a sua localização dentro da imagem (sendo devolvido as

coordenadas do centro do padrão que foi reconhecido e as coordenadas do pixel mais a esquerda), a escala que neste caso é sempre um, o que significa que o programa só reconhece padrões do mesmo tamanho. É ainda devolvido um valor que nos dá uma noção da semelhança entre o padrão encontrado na imagem e aquele que foi carregado inicialmente no programa.

4.2.2. RECONHECIMENTO DE CARACTERES

O reconhecimento óptico de caracteres (OCR) é um caso particular do reconhecimento de padrões que visa, somente, reconhecer caracteres.

O algoritmo desenvolvido é muito semelhante ao de reconhecer padrões.

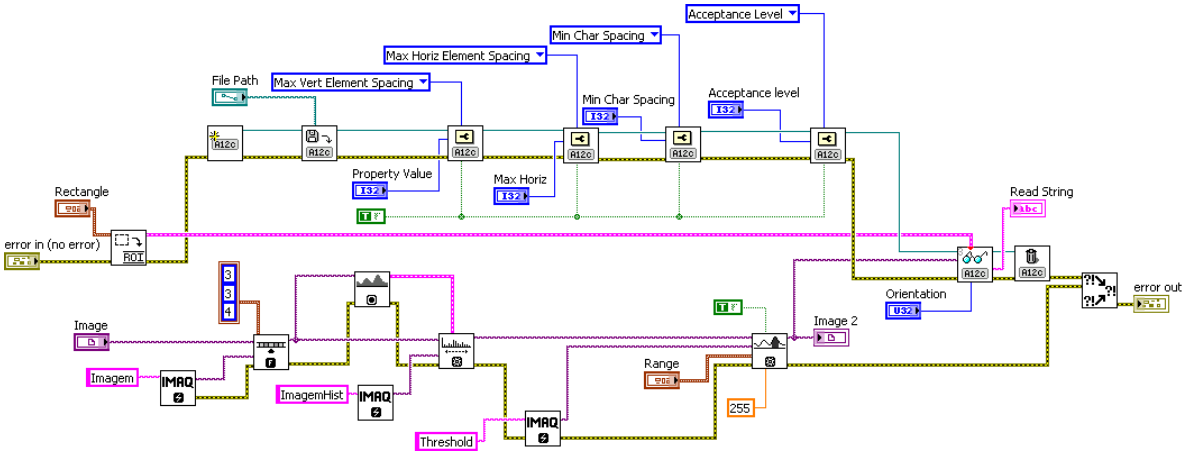


Figura 26 - Código do algoritmo de reconhecimento de caracteres

No início é aberta uma sessão OCR e são estabelecidas algumas propriedades referentes ao algoritmo, tais como, o nível de aceitação (“Acceptance level”) que um objecto é considerado um caracter, o mínimo espaçamento entre cada caracter (“Min. Char Spacing”) e o espaçamento entre os elementos constituintes do caracter tanto horizontal como vertical. Depois, é carregado para a sessão um ficheiro que contém os caracteres a serem reconhecidos. Paralelamente aos procedimentos anteriores, a

imagem, na qual se quer fazer o reconhecimento de caracteres, é processada de modo a se extrair mais facilmente a informação, sendo-lhe aplicado um filtro passa-alto, a fim de realçar os contornos. Posteriormente, faz-se a equalização do histograma e, por fim, aplica-se um limiar de threshold tornando a imagem a cores numa a preto e branco. Após o processamento é levada em conta uma região de interesse (ROI "*region of interest*"), que é a zona onde se encontram os caracteres a serem reconhecidos e que foi previamente definida pelo utilizador. Nessa mesma zona são isolados os diferentes caracteres e posteriormente, são comparados com os caracteres existentes no ficheiro e assim reconhecidos. No fim, destas tarefas todas são devolvidos os caracteres que foram identificados ou o carácter '?' associado aos elementos que não foram identificados.

4.3. CONTROLO DO MOVIMENTO DA CÂMARA

Para se efectuar o controlo do movimento da câmara foi necessário elaborar dois algoritmos, um para ler e contabilizar o número de impulsos vindos do codificador (*encoder*) e outro para o controlo dos motores. Em seguida, apresentar-se-á uma descrição dos mesmos.

4.3.1. LEITURA DO CODIFICADOR

O algoritmo de leitura do codificador é muito simples, é somente a leitura das entradas digitais, correspondentes as fases A e B do codificador, em intervalos de tempo definidos e verificar se a amostra anterior é diferente da actual. Se for diferente, significa que houve uma transição, logo é incrementado o contador de impulsos. É ainda, contabilizado o intervalo de tempo entre duas transições com o intuito de calcular a velocidade do objecto no tapete rolante. De referir, que houve a necessidade de se impor

mais uma condição, que é de só se contabilizar uma transição por cada fase alternadamente, isto é, só após de uma transição da fase A é que se contabiliza uma transição da fase B e vice-versa. Isto, porque se verificou que na eventualidade de se parar o codificador na zona de transição de uma das fases o seu estado ficava indefinido, ou seja, transitava entre o '1' e o '0' lógico permanentemente.

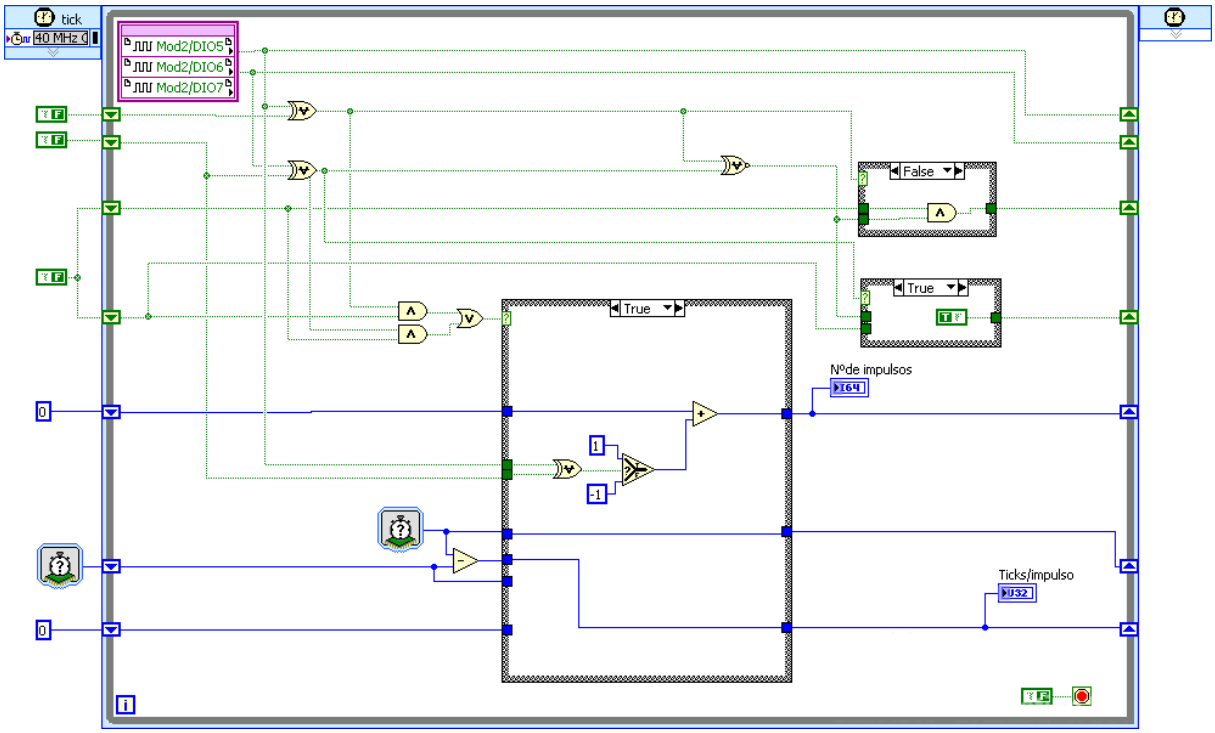


Figura 27 - Código do algoritmo de leitura do codificador

4.3.2. CONTROLO DO MOTOR

O controlo do motor já foi uma tarefa mais complexa uma vez que, exigia a interacção e coordenação entre a FPGA e o controlador. O algoritmo tem como parâmetros de entrada a aceleração, desaceleração, velocidade final, o modo de referência e a posição final pretendidos. Após estes parâmetros estarem definidos o processo começa por dividir o percurso em zonas de aceleração, desaceleração e zonas

onde é para manter a velocidade constante, sendo estes cálculos efectuados no controlador e posteriormente, enviados para a FPGA.

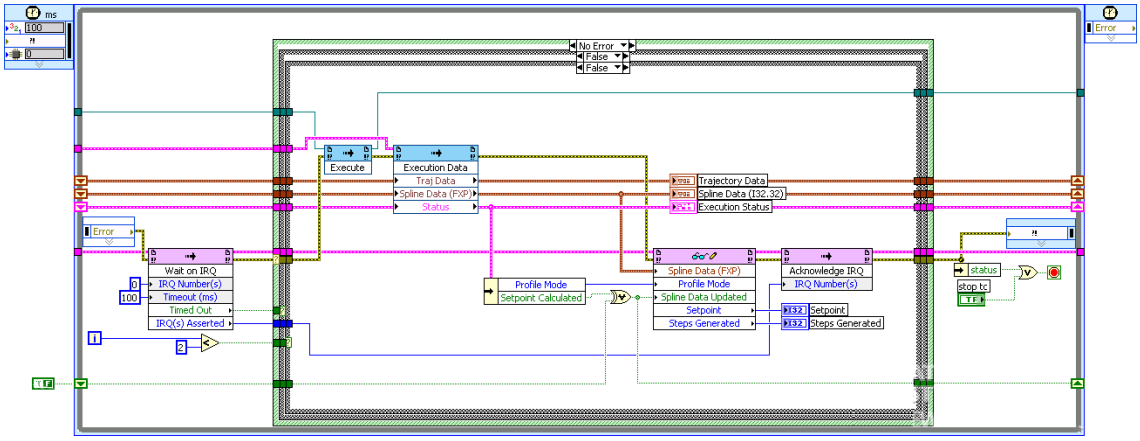


Figura 28 - Código presente no controlador referente ao controlo dos motores

Na FPGA, com base nesses dados é calculada a duração do PWM e o nível do sinal que indica a direcção sendo, depois, enviados estes dois sinais para o driver do motor.

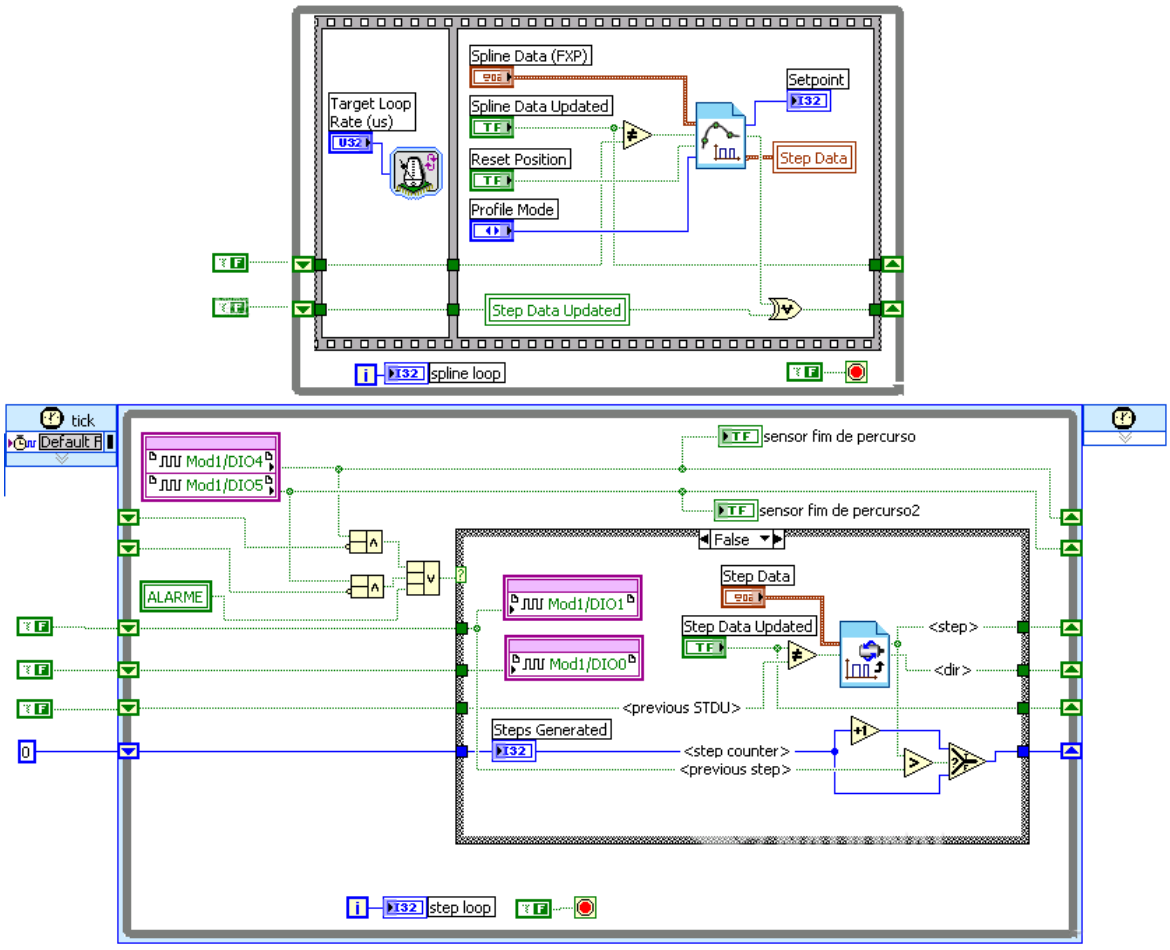


Figura 29 - Código referente ao controlo do motor presente na FPGA

Por fim, falta referir que a FPGA se sincroniza com o controlador gerando uma interrupção a este último. Com este método a FPGA informa que processou os dados referentes a uma dada parte do percurso e assim o controlador pode enviar novos dados referentes a uma nova parte do mesmo.

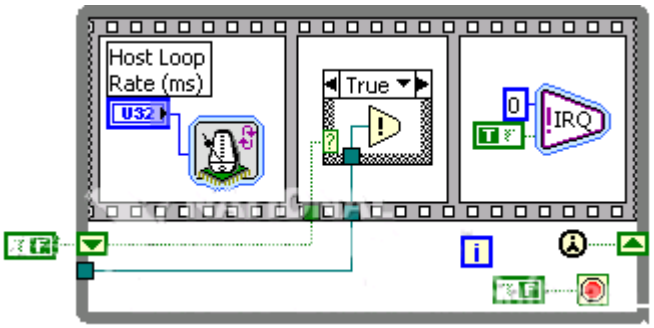


Figura 30 - Código responsável pela sincronização

4.4. PROGRAMA FINAL

Após se ter explicado os algoritmos que servem de base ao programa final, ir-se-á explicar o mesmo para cada um dos sistemas. Somente neste ponto a programação difere entre os dois.

Como se pode, observar no diagrama de blocos abaixo (referente ao primeiro sistema), o primeiro passo é inicializar o sistema, que consiste em colocar a câmara em movimento no sentido da entrada dos objectos sem posição final definida. Esta será parada quando for detectada pelo sensor indutivo de fim de percurso. É então estabelecida a origem/referência para os restantes movimentos da câmara. Em seguida, o sistema fica a espera da entrada de um objecto. Assim que é detectada a entrada do objecto, através da fotocélula, é iniciado o cálculo (cálculo este que é explicado mais a frente) da posição de encontro. Calculada a posição de encontro é enviada para o algoritmo de controlo dos motores como posição final e é então que é iniciado o movimento da câmara. Espera-se que a câmara atinja a posição, ou melhor, que a

velocidade de deslocamento da câmara seja igual a zero para se inverter o sentido do deslocamento passando agora a câmara a acompanhar o objecto.

Depois espera-se, mais uma vez, que a velocidade da câmara seja igual ao do objecto e é então que se faz a primeira aquisição de imagem, que posteriormente é processada. Dada a primeira aquisição faz-se com que o segundo motor desloque a câmara transversalmente ao sentido do movimento do objecto para se realizar a segunda aquisição de imagem. Concretizada a segunda aquisição, é dada a ordem de recuo da câmara. De referir que se durante o recuo da câmara, entrar um novo objecto, a câmara pára imediatamente e inicia-se o processo de cálculo da posição de encontro.

Ficou ainda por mencionar que toda a sincronização entre o movimento da câmara e os dois momentos de aquisição de imagem é efectuada através de uma livraria de variáveis globais que se encontra no controlador cRio mas que o computador tem acesso a ela através da ligação Ethernet.

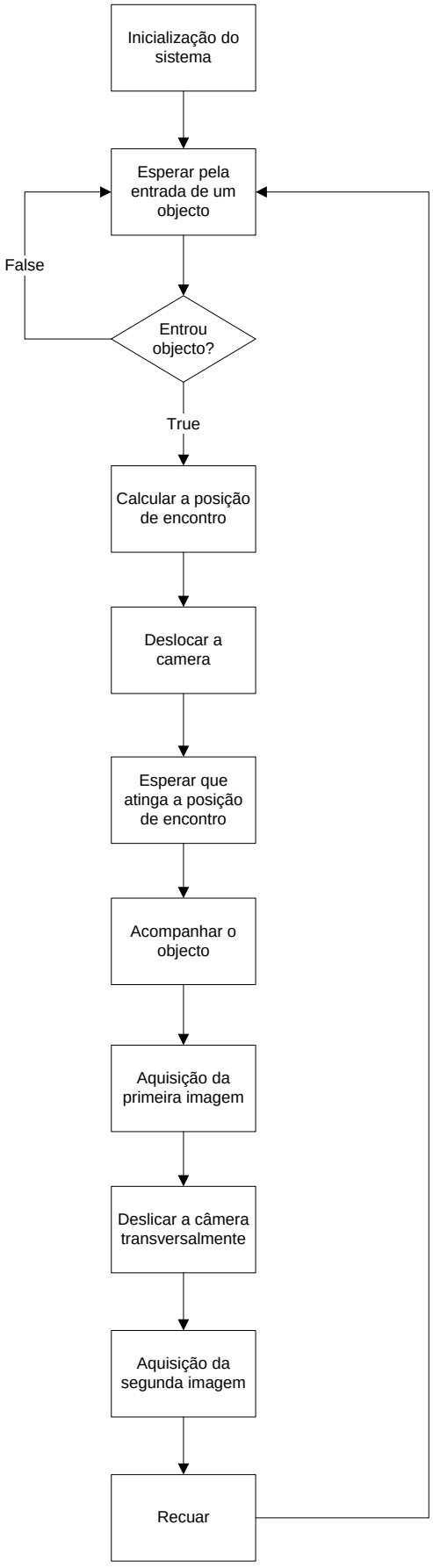


Figura 31 - Diagrama de blocos do algoritmo final do primeiro sistema

4.4.1. CÁLCULO DA POSIÇÂO DE ENCONTRO

O cálculo da posição de encontro entre a câmara e o objecto que entra no tapete rolante, resumiu-se a um problema de física no qual é necessário calcular a posição de encontro de dois objectos que seguem em sentidos opostos, sabendo a partida as posições dos dois objectos, assim como o valor da velocidade de ambos. No caso do motor considerou-se que este, arranca e pára instantaneamente, de modo a eliminar as acelerações das equações do algoritmo.

Tendo em conta a equação da velocidade assim como a figura:

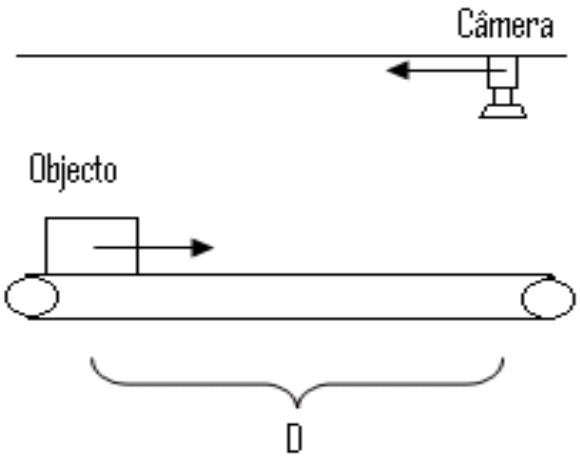


Figura 32 - Esquema do sistema

$$velocidade = \frac{Deslocamento}{Tempo}$$

Podem-se escrever o seguinte sistema de equações:

$$\begin{cases} v_c = \frac{d_c}{t} \\ v_o = \frac{d_o}{t} \\ D = d_c + d_o \end{cases}$$

Onde,

v_c - velocidade da câmara

d_c – deslocamento da câmara da posição inicial até a posição de encontro (é a variável que interessa calcular)

v_o – velocidade do objecto

d_o – deslocamento do objecto da posição inicial até a posição de encontro

D – distância entre o objecto e a câmara (esta variável é conhecida já que se sabe a posição do motor por este ser passo-a-passo, e a do objecto através do codificador)

Eliminando a variável tempo, e resolvendo o sistema em ordem a d_c obtém-se:

$$d_c = \frac{v_c * D}{v_o + v_c}$$

que era o pretendido.

ALGORITMO FINAL DO SEGUNDO SISTEMA

Neste segundo sistema construiu-se uma máquina de estados que faz o seguinte:

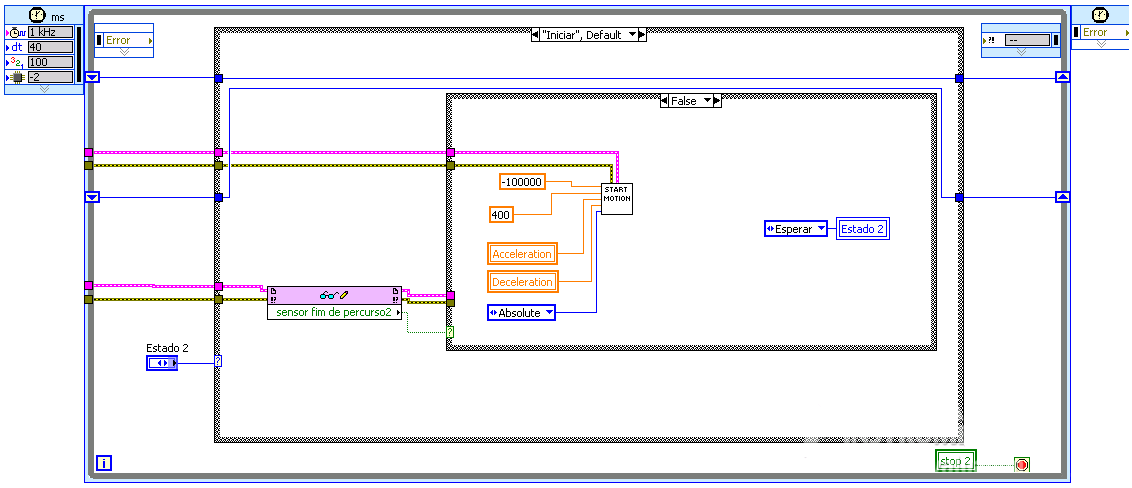


Figura 33 - Código do estado "iniciar"

Inicializa o sistema, ou seja, faz deslocar a câmara para um dos lados com o objectivo de determinar um ponto que seja a referência para os posteriores posicionamentos da câmara. Ponto esse, que é determinado quando o sensor indutivo, colocado no limite da calha, transita do estado lógico '0' para o estado lógico'1'.

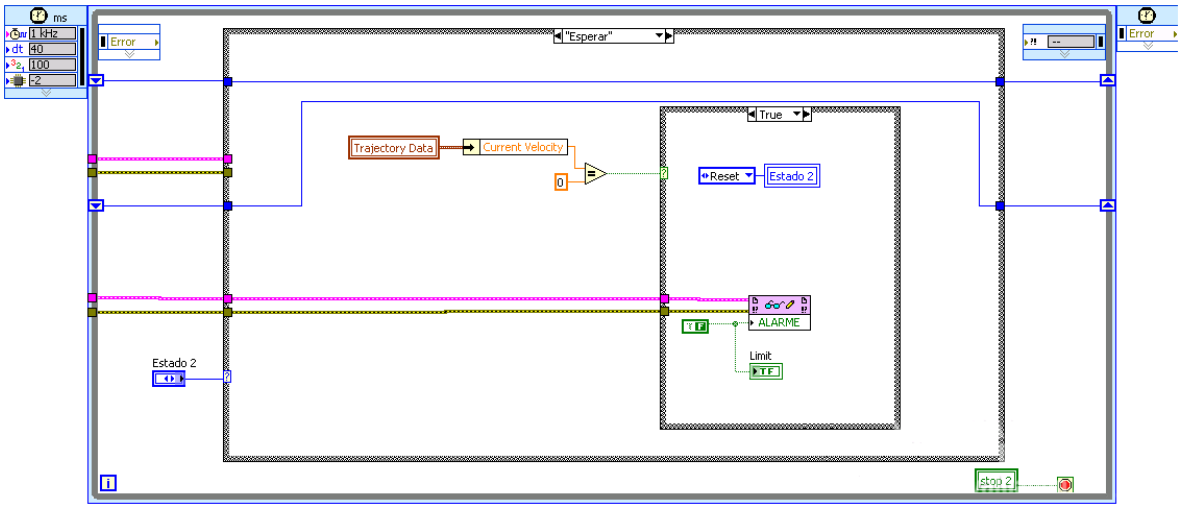


Figura 34 - Código do estado de "espera"

A posição onde a câmara pára, fica a ser a posição zero.

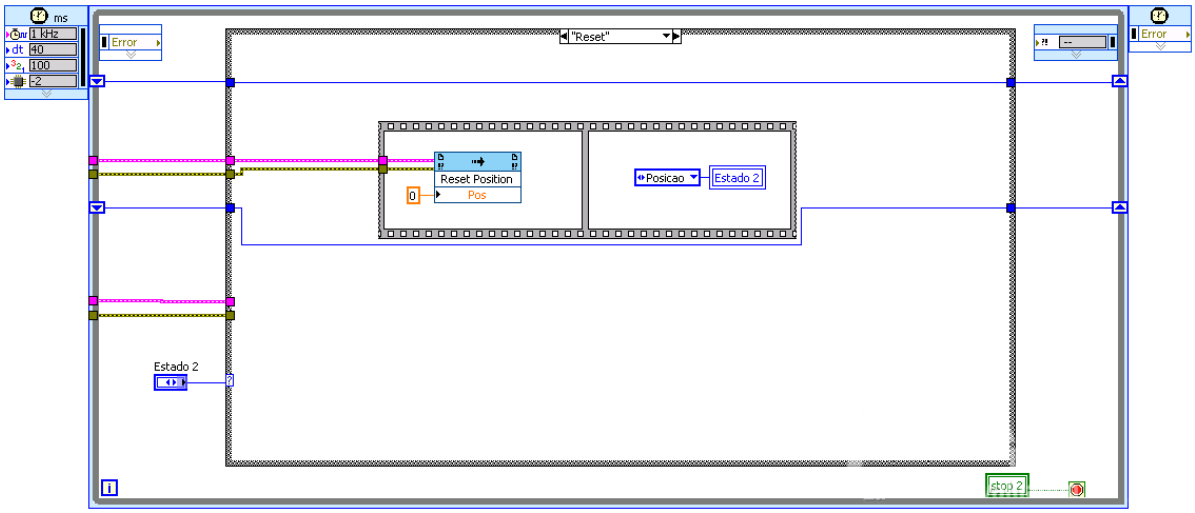


Figura 35 - Código do estado "Reset Posição"

Após este procedimento inicia-se o deslocamento até ao primeiro ponto de aquisição de imagens, onde se inicia a sua captura.

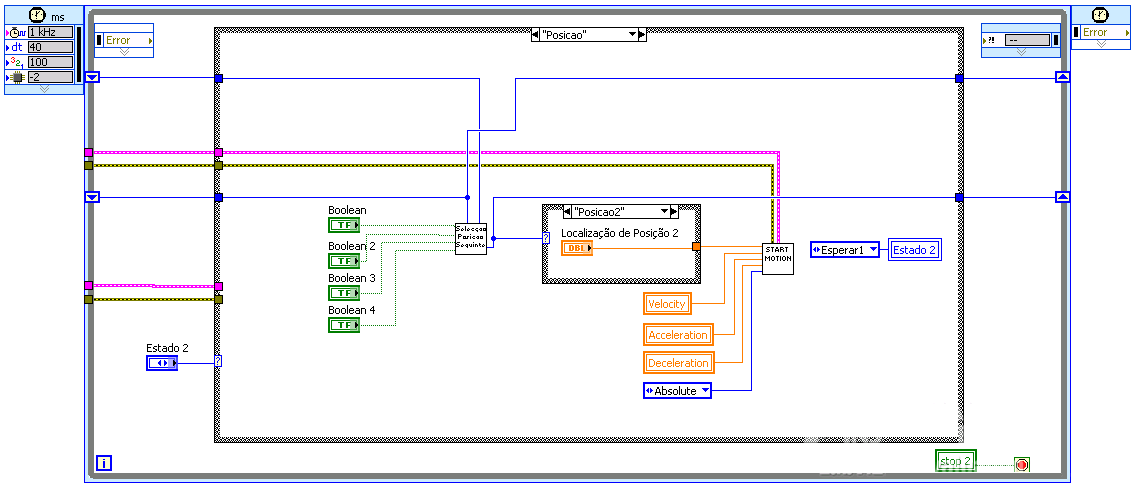


Figura 36 - -Código onde é calculado a posição seguinte e iniciado o movimento

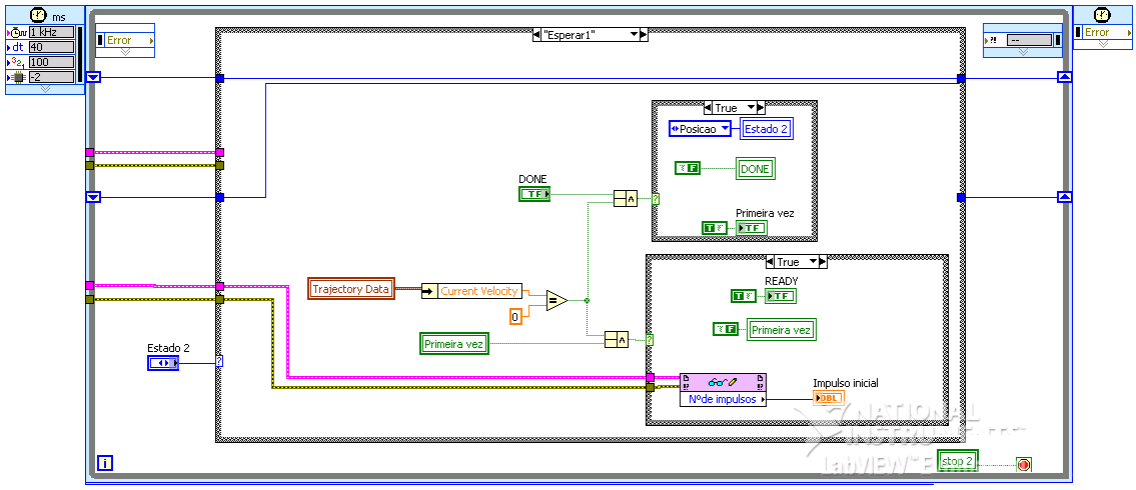


Figura 37 - Código referente ao estado de aquisição das fotografias

Quando se encontra no estado de captura das imagens, os vários momentos de aquisição das mesmas, são controlados através da contagem dos impulsos do codificador.

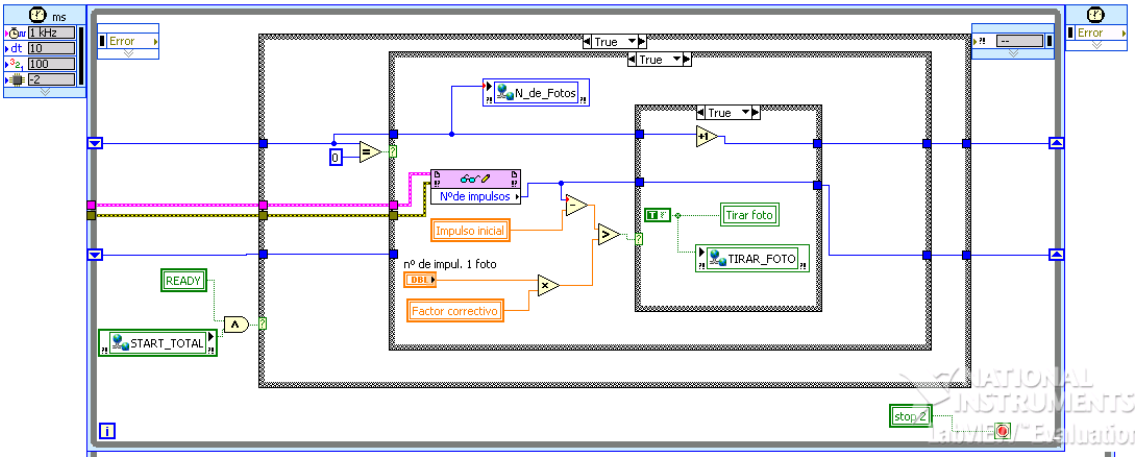


Figura 38 - Código de determinação dos momentos de aquisição das imagens

Durante a captura da sucessão de imagens estas vão sendo processadas, isto é, reconhecidos os caracteres existentes nas várias imagens. Ao extrair os caracteres, estes vão sendo concatenados aos das imagens anteriores. No final da última imagem a frase construída através do reconhecimento é comparada a frase introduzida pelo utilizador.

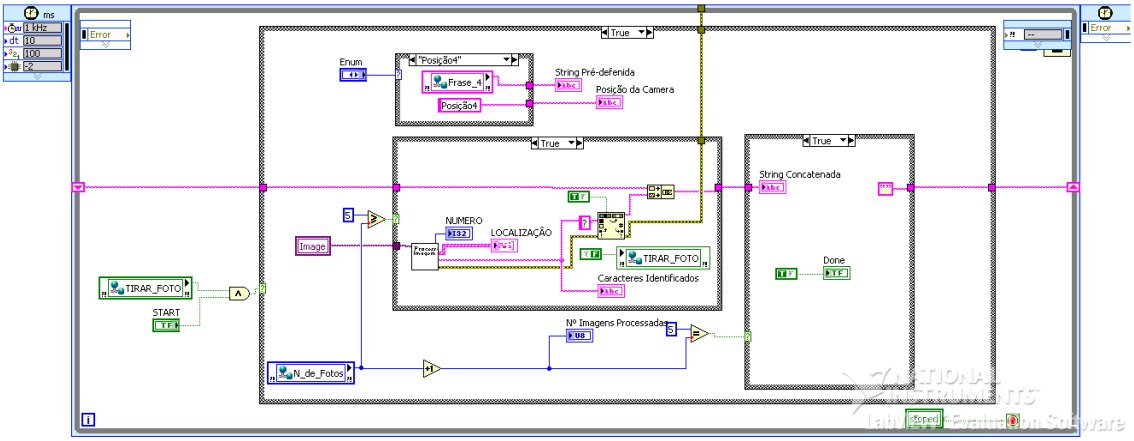


Figura 39 - Código referente ao processamento de imagem e concatenação dos caracteres

Em seguida, o PC informa o controlador que a câmara pode ser deslocada para o próximo ponto e assim sucessivamente.

5. INTERFACES GRÁFICOS DESENVOLVIDOS

Apresentados os algoritmos de todo o sistema apresentar-se-á, uma das partes mais visíveis do trabalho, que são os interfaces gráficos que dão acesso a todas as funcionalidades do sistema. Neste trabalho foram desenvolvidos essencialmente dois interfaces, um que transmite os objectivos principais do trabalho, isto é, os resultados do processamento de imagem e um outro que serve mais como uma ferramenta de monitorização do controlo dos motores. Em seguida apresenta-se uma descrição das funcionalidades de ambos os interfaces.

5.1. INTERFACE GRÁFICO DE PROCESSAMENTO DE IMAGEM

O interface construído para o processamento de imagem foi o seguinte:

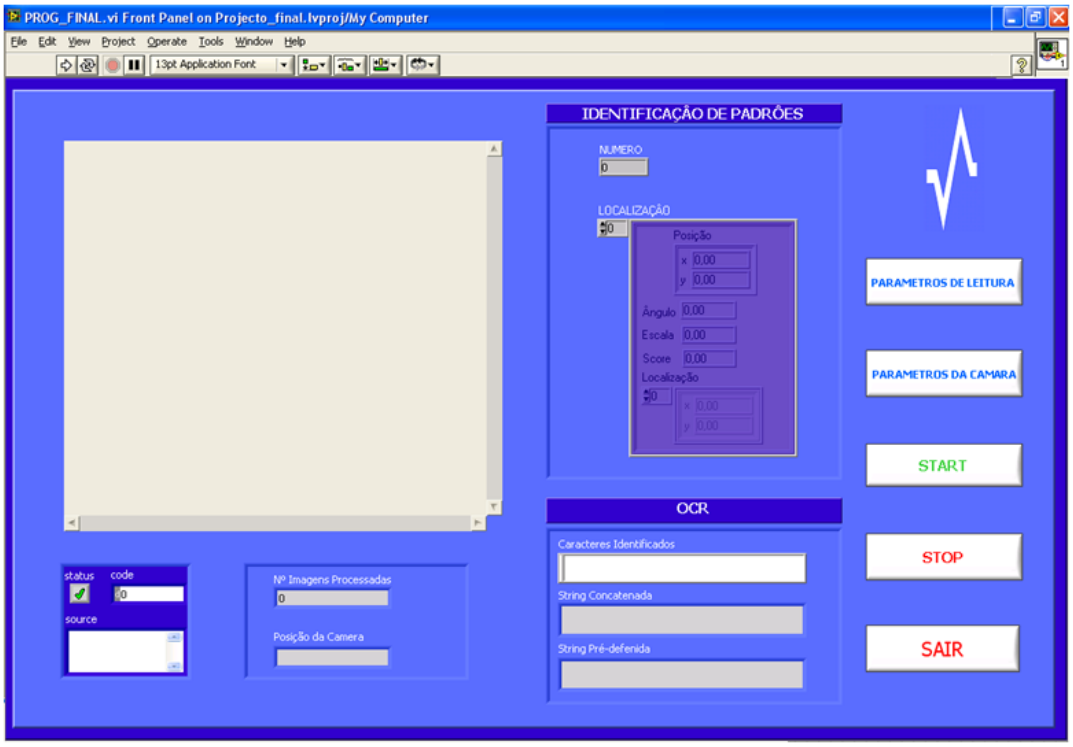


Figura 40 - Interface da aplicação de processamento de imagem

Como se pode observar na figura anterior, o interface é constituído por uma janela onde se visualiza as imagens captadas pela câmara, uma parte onde o utilizador é

informado dos erros que acontecem internos a aplicação, uma outra parte onde se encontra os dados do processamento de imagem que indica, na zona de identificação de padrões, o número de padrões reconhecidos na imagem, as suas posições na imagem e ainda os graus de rotação relativamente ao padrão predefinido. Apesar de lá estar o campo "escala", este tem sempre o valor 1, o que significa que só reconhece padrões do mesmo tamanho. Depois na zona de OCR, é onde o programa imprime os caracteres reconhecidos.

Por último, falta apenas fazer referência a zona de opções disponibilizadas pelo interface. Os botões "START" e "STOP" servem para iniciar e parar, respectivamente, todo o processo a que se destina o sistema, isto é, fazer o reconhecimento de caracteres nas mangueiras. Já o botão "Parâmetros da câmara" faz com que apareça a seguinte janela flutuante:

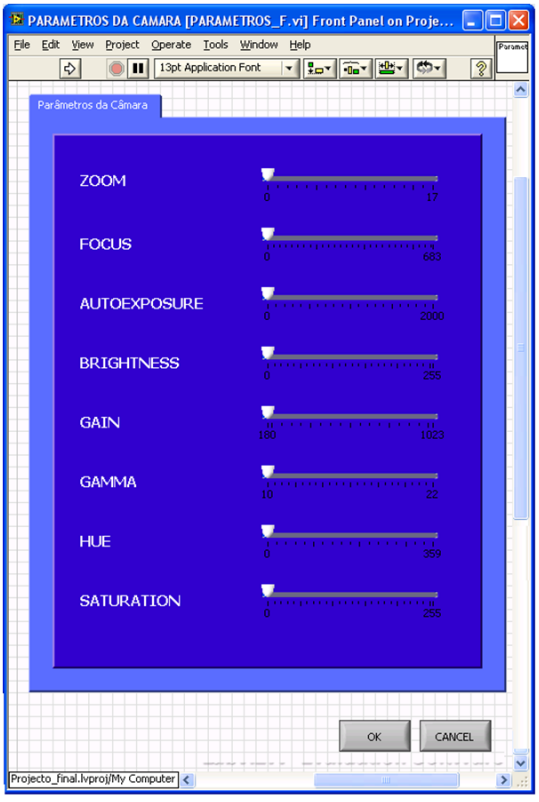


Figura 41 - Janela de configuração dos parâmetros da câmara

a qual permite definir as propriedades da câmara. Carregando no botão "OK" o programa guarda as alterações efectuadas, pelo contrário, descarta as modificações feitas se o botão "CANCEL" for premido.

Quanto ao botão “Parâmetros de Leitura” abre a janela flutuante da figura 42 na qual é permitido definir alguns parâmetros relacionados com o reconhecimento de padrões e com o OCR.

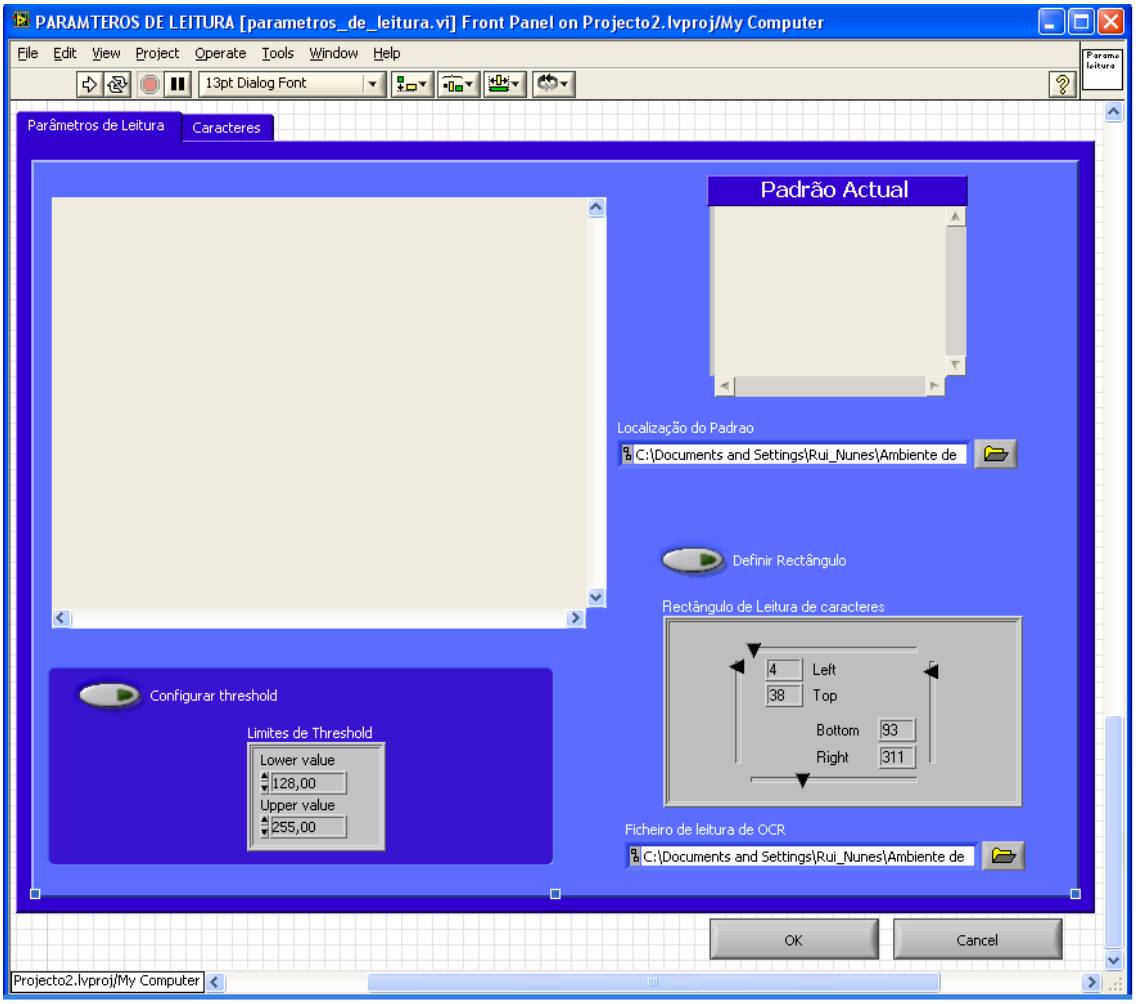


Figura 42 - Janela de configuração dos parâmetros de leitura

Nesta janela é possível definir o padrão que pretende reconhecer depois no processamento de imagem. Este é seleccionado na área de imagem, com o auxílio do rato, rodeando o padrão pretendido com um rectângulo. Este procedimento vai fazer com que a área seleccionada com o rato apareça na janela do “Padrão Actual” podendo assim o utilizador verificar se está a seleccionar correctamente o padrão.

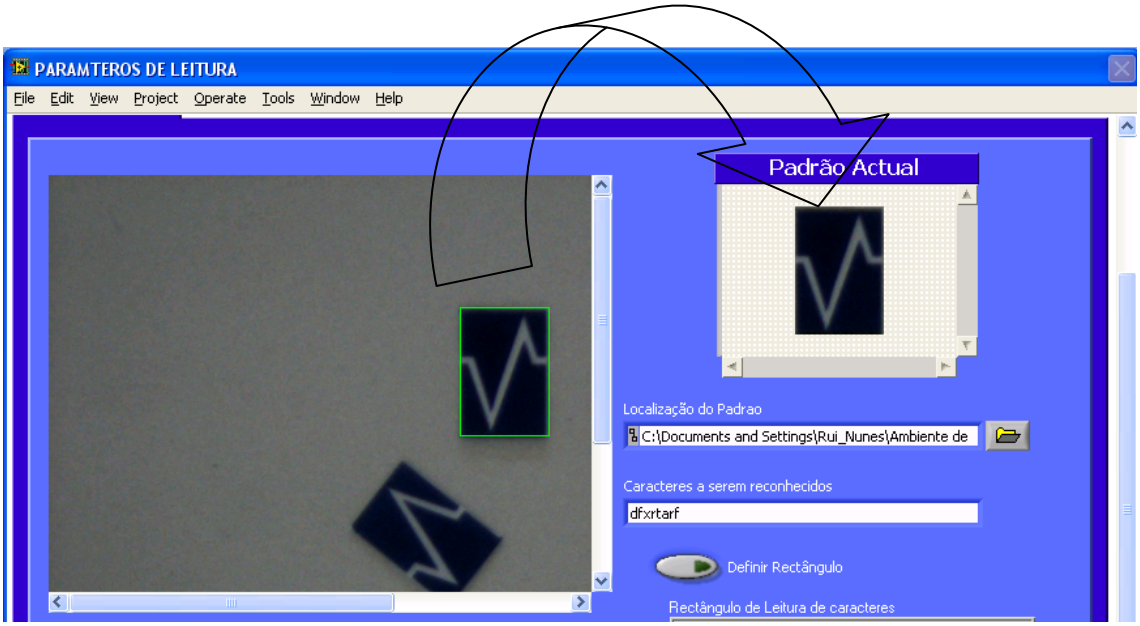


Figura 43 - Escolha do padrão pretendido

Após ter sido seleccionado correctamente, o utilizador tem a possibilidade de escolher a localização onde pretende guardar o padrão.

Relativamente as opções de "OCR" o utilizador terá que colocar, a localização do ficheiro que contém os caracteres a serem reconhecidos e ainda definir o rectângulo dentro do qual o programa procurará os caracteres a serem reconhecidos. Para definir o rectângulo o utilizador deverá carregar no botão "Definir Rectângulo" ficando este activo (como mostra a figura 44) e depois fazer o mesmo processo de escolha do padrão.

Figura 44 - Botão de "definir rectângulo" activo

Há ainda a possibilidade do utilizador definir o limiar de threshold. Basta para isso, activar o botão "Configurar threshold" como mostra a figura 45. A imagem mudará para aquela que é, efectivamente, processada pelo algoritmo de reconhecimento de caracteres.

Figura 45 - Botão "Configurar threshold" activo

Por último, falta fazer referência ao separador “Caracteres”, que ao carregar nesse mesmo separador aparece os campos para se introduzirem os caracteres a serem reconhecidos em cada uma das posições de leitura.

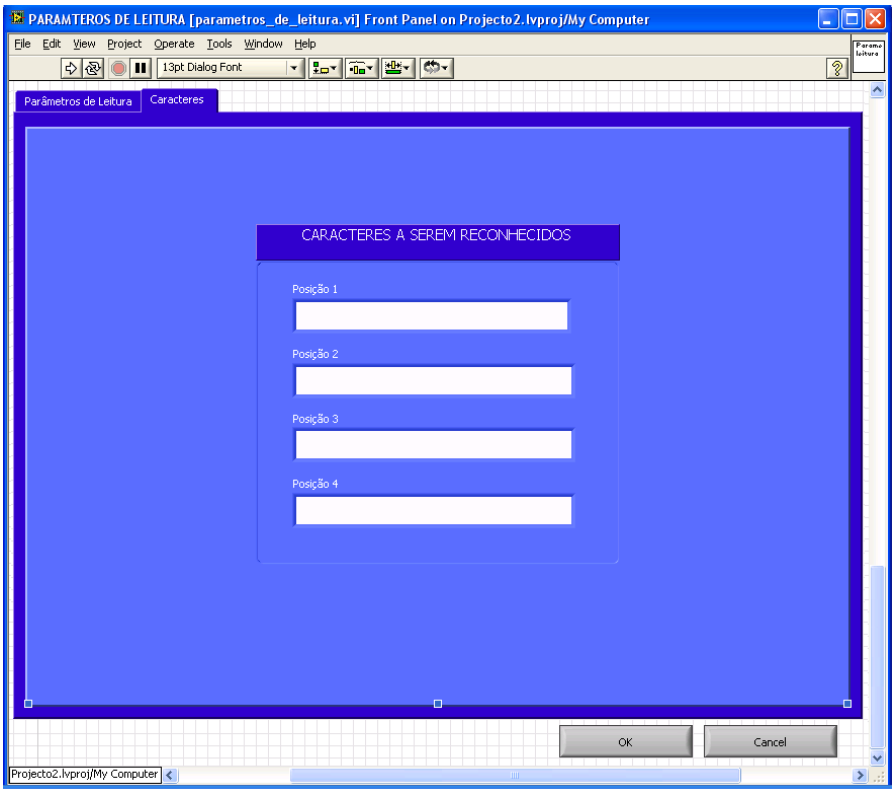


Figura 46 - Janela onde se inserem os caracteres a serem reconhecidos

5.2. INTERFACE DE CONTROLO DO MOTOR

Este interface permite ao utilizador monitorizar e definir todos os parâmetros relativos ao funcionamento do motor. Como se pode observar, na figura abaixo, apresenta campos para a introdução dos valores da velocidade, aceleração e desaceleração dos deslocamentos da câmara. Permite ainda a definição das quatro posições de inspecção das mangueiras, assim como, em quais se deve fazer a respectiva inspecção. Através dos campos "nº de impulsos" são inseridos o número de impulsos necessários para se fazer a aquisição da primeira e das restantes imagens. O factor correctivo tem como função fazer a correspondência entre o deslocamento das mangueiras verificado durante um impulso do codificador. O interface possibilita ainda um controlo manual do motor, inserindo um valor no campo "Target Position" e premindo o botão "Start".

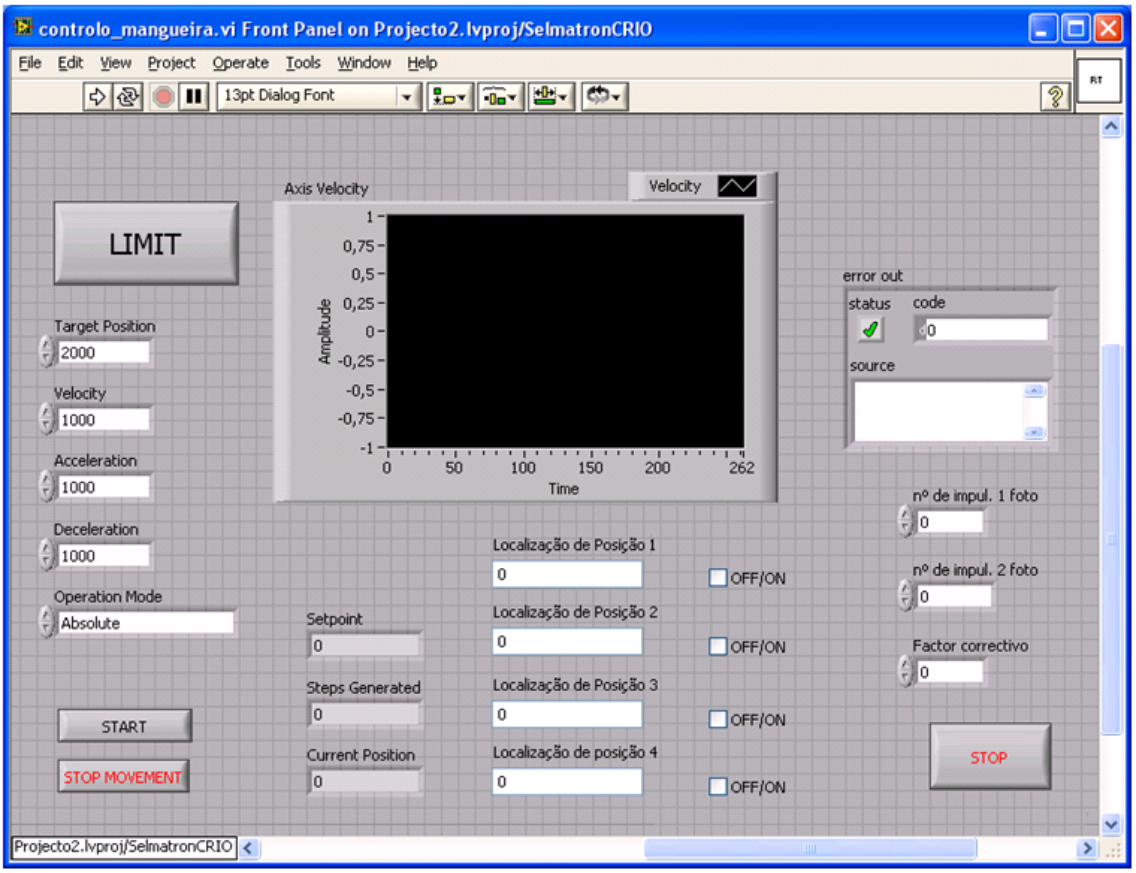


Figura 47 - Interface de controlo dos motores

Todos os restantes elementos que este interface possui, são para monitorização do comportamento dos motores. Apresenta a posição actual em relação a um referencial, o número de “steps” total que foram produzidos para movimentar o motor, assim como um gráfico que é actualizado em tempo-real com a velocidade do motor.

6. RESULTADOS

Neste capítulo pretende-se expor a evolução e os principais resultados obtidos neste trabalho. Manter-se-á a estrutura seguida até aqui, apresentando primeiro os resultados do processamento de imagem e depois o controlo do movimento da câmara.

6.1. PROCESSAMENTO DE IMAGEM

Os resultados relativamente ao processamento de imagem foram bastante satisfatórios no final, embora tenham sido obtidos com dificuldade diferente, tendo sido mais fácil obter os resultados do reconhecimento de padrão comparativamente com o de OCR. Neste só se obteve os resultados desejados após se ter utilizado todas as técnicas de processamento de imagem descritas atrás.

Para a demonstração dos resultados relativamente ao reconhecimento de padrão ir-se-á utilizar o padrão da figura abaixo:

Figura 48 - Padrão exemplo

Em seguida, apresenta-se várias imagens com os respectivos resultados do programa:

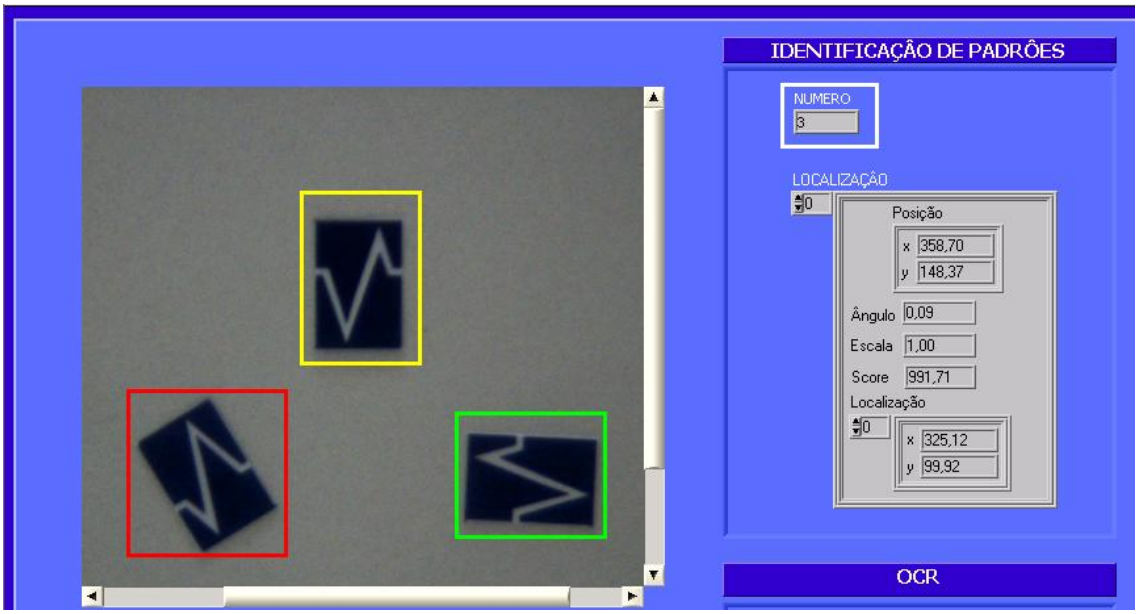


Figura 49 - Resultados relativa a identificação de padrões.

Como se pode observar na imagem, a aplicação foi capaz de identificar o número de padrões presente na imagem e como a seguir, se pode comprovar devolveu, igualmente, os restantes dados tal como a localização, ângulo de rotação, o score e a escala (que como já foi referido anteriormente é sempre igual a 1):

Figura 50 - Dados relativos á identificação de padrões. A primeira tabela é referente ao padrão realçado a amarelo, a segunda ao padrão realçado a vermelho e a última ao padrão com realce verde

Como já foi referido, relativamente ao OCR o resultado final pretendido já foi mais difícil de obter, isto porque o algoritmo de identificação de caracteres tem que separar primeiro cada um dos caracteres para posterior identificação. O que acontecia era que quando um caracter não estava separado do seguinte, o algoritmo entendia esses dois

caracteres como sendo um só, então se a imagem de entrada do algoritmo de reconhecimento de caracteres fosse a da figura abaixo, isto é, sem qualquer tipo de processamento o resultado do algoritmo era muitas das vezes '*selmatmn*':

Figura 51 - Imagem sem processamento

Este problema ficou resolvido quando se aplicou as técnicas de processamento de imagem. Através destas, ficou possível controlar a distância entre os vários caracteres e assim fazer uma leitura correcta dos mesmos.

6.1.1. RESULTADOS FINAIS

Em seguida apresenta-se uma sequência de fotos retiradas já na leitura de caracteres numa mangueira:

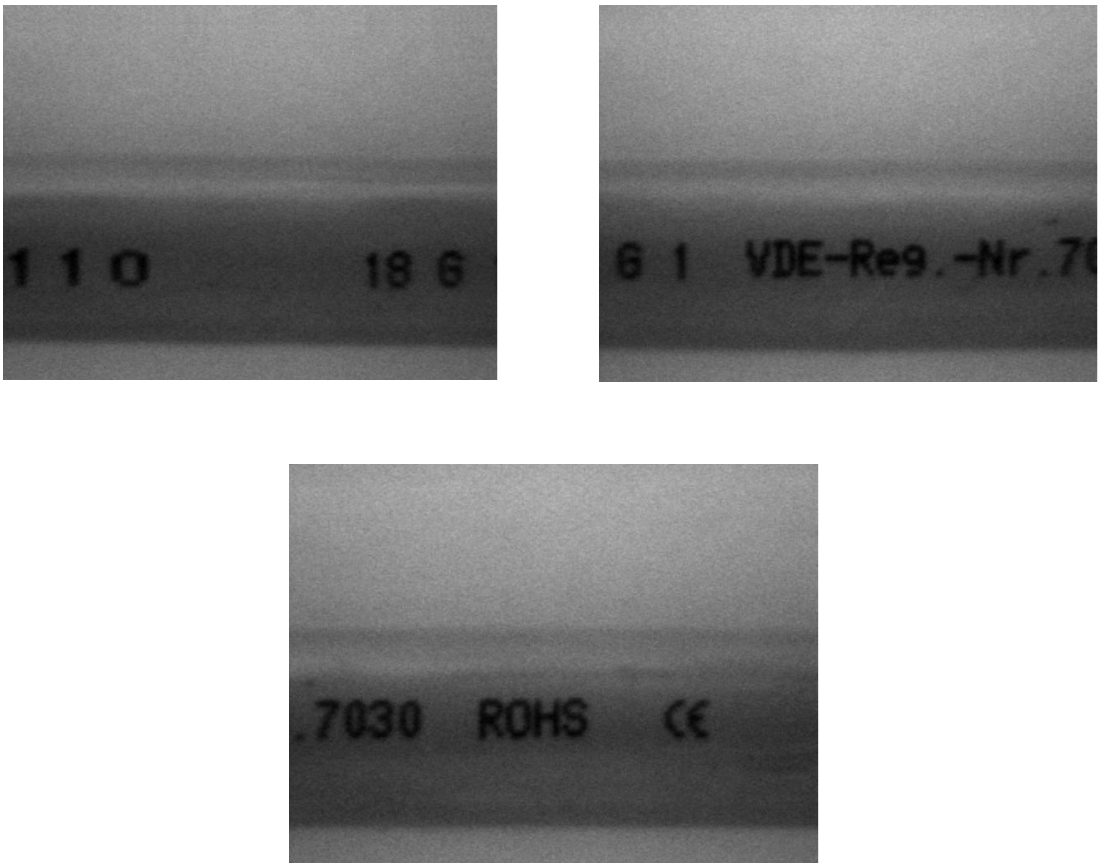


Figura 52 - Sequência de imagens resultantes do sistema em funcionamento

Como se pode observar existem alguns caracteres repetidos de umas fotos para as outras. Este aspecto é devido ao facto de não se querer correr riscos em relação à perda de caracteres. Então optou-se por fazer a aquisição da imagem contendo a parte final da imagem anterior. Para não haver problema de repetição de caracteres reduziu-se a região de interesse, ou seja, a zona da imagem onde é para ler os caracteres. O problema que surgiu neste ponto do trabalho foi em relação aqueles caracteres que ficam no limite dessa zona (figura 53).

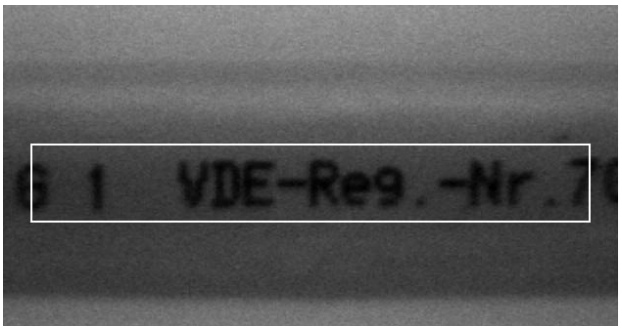


Figura 53 - Zona de interesse indicada pelo rectângulo branco

Aí, se a parte do caracter que fica dentro da zona de interesse não for reconhecida como carácter então, não há problema, visto que associada a esse termo o algoritmo de reconhecimento de caracteres devolve '?' e, estes são removidos antes de se proceder a concatenação. Já quando é interpretado como sendo um outro carácter introduz um notório erro de leitura que depois vai influenciar o resultado final. Este último problema por falta de tempo não ficou resolvido.

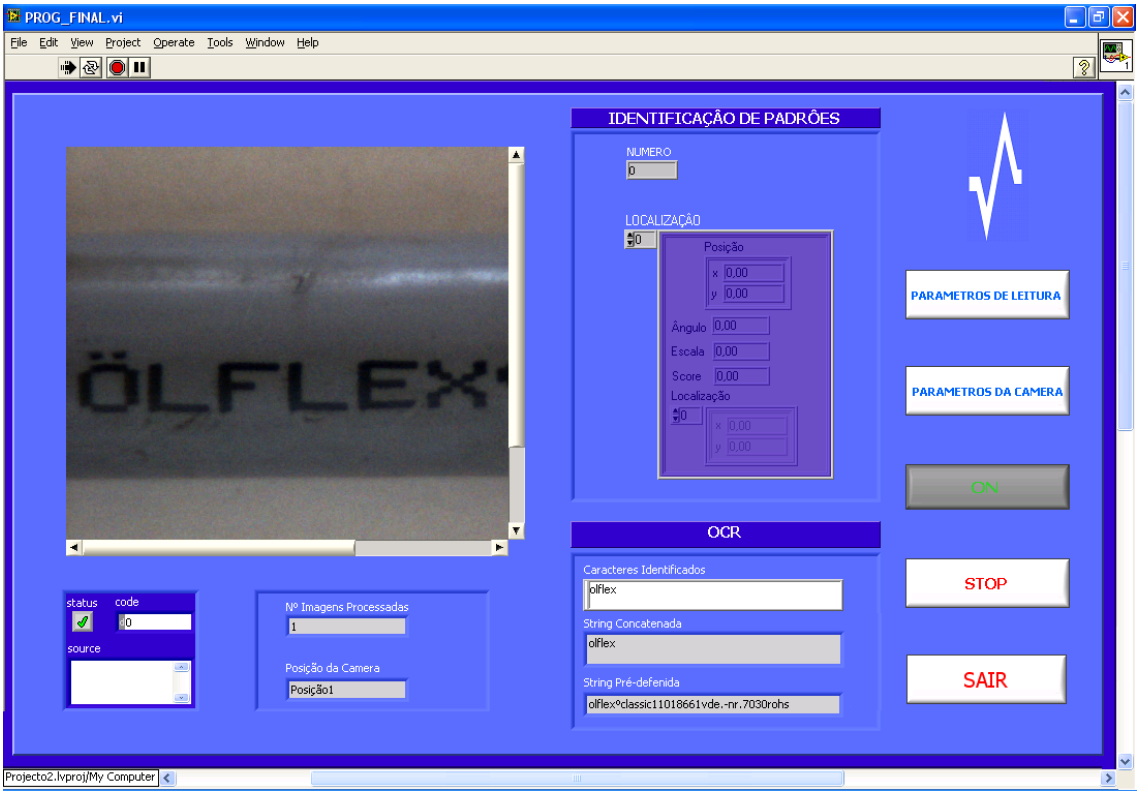


Figura 54 - Interface de processamento de imagem em funcionamento

Por último, falta somente fazer uma referência ao excelente comportamento do motor passo-a-passo no posicionamento da câmara. Ficando comprovado, todas as vantagens deste, tal como, a repetibilidade do processo que se não era perfeita, era muito perto disso.

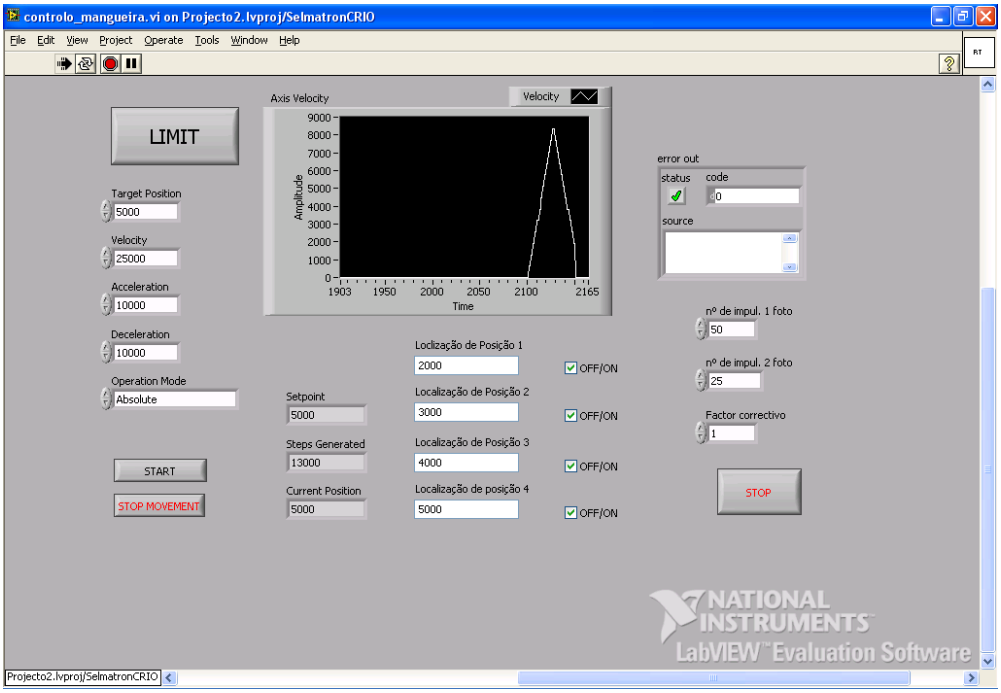


Figura 55 - Interface de controlo do motor em funcionamento

7. CONCLUSÃO E TRABALHO FUTURO

Na conclusão deste trabalho pode-se verificar, o quanto este tipo de sistemas são úteis no auxílio à inspecção de material e que são capazes de substituir o ser humano em algumas destas tarefas rotineiras e aborrecidas.

E apesar de não ter ficado a funcionar a cem por cento, o sistema é capaz de desempenhar a sua função uma vez que o objectivo na instalação deste sistema em específico era o de não deixar passar metros de mangueira sem que tivessem sido impressos os caracteres, e com o auxílio do reconhecimento de padrões consegue-se atingir esse mesmo objectivo, colocando como padrão a reconhecer um qualquer carácter.

Uma outra, conclusão que se pode retirar deste trabalho, é o extraordinário desempenho e comportamento de um motor passo-a-passo, que permite um controlo de posicionamento muito preciso da câmara, vezes sem conta.

Fica também provado a grande versatilidade que estes sistemas têm, já que com pequenas alterações de código se pode adaptar as funcionalidades desenvolvidas a situações diferentes.

Por último, falta fazer uma referência ao Labview. É uma ferramenta extremamente poderosa e versátil. Após adaptação do utilizador, este é capaz de fazer trabalhos de algum nível de complexidade num curto espaço de tempo, que é o pretendido na indústria. Em parceria com o hardware da National Instruments, formam ainda um conjunto mais poderoso para aquilo que se pretende, obter medidas de sensores e fazer automação.

TRABALHO FUTURO

Em primeiro lugar, em caso de haver uma continuação do trabalho seria necessário, inevitavelmente, melhorar o processo de concatenação dos caracteres, visto que ficou algo imperfeito. Após este melhoramento, o caminho a seguir seria o de dotar a câmara de tantas funções quantas possíveis, tais como, o reconhecimento de cor, leitura

de código de barras e até utilizar a câmara para localizar pontos de referência, por exemplo.

Após implementar as funcionalidades atrás referidas adaptar estas a tantos sistemas quanto possível, ficando por isso a sugestão de se fazer um estudo prévio das necessidades da indústria para além do tipo de sistema construído neste trabalho.

8. REFERÊNCIAS

[1] – NATIONAL INSTRUMENTS, “LabVIEW Fundamentals”, 2007

[1] – NATIONAL INSTRUMENTS, “Getting Started with the NI SoftMotion Development Module for LabView”, 2007

[3] – NATIONAL INSTRUMENTS, “Real-Time Module User Manual”, 2004

[4] – NATIONAL INSTRUMENTS, “Getting Results with CompactRio and LabVIEW”, 2006

[5] – NATIONAL INSTRUMENTS, “NI-MOTION User Manual”, 2003

[6] – NATIONAL INSTRUMENTS, “FPGA Module User Manual”, 2004

[7] – NATIONAL INSTRUMENTS, “FPGA Interface User Guide”, 2004

[8] – NATIONAL INSTRUMENTS, “NI Vision”, 2005

[9] – K. Fu, R. Gonzalez, C. Lee; “Robotics: Control, Sensing, Vision, and Intelligence”; McGraw-Hill, 1987

[10] – William K. Pratt, “DIGITAL IMAGE PROCESSING”, Wiley-Interscience, 2º edição, 1991.

[11] – Oriental Motors – PH599 – AE, Datasheet

[12] – Lika, “Incremental encoders – series I40-I41, Datasheet

[13] – Ifm electronic, “IG5938”, Datasheet

9. ANEXOS

9.1. APRESENTAÇÃO DO CÓDIGO DESENVOLVIDO

No presente anexo, pretende-se explicar o código que foi desenvolvido, para além daquele que já foi anteriormente apresentado. Por isso, vai-se explicar parte do código dos ficheiros "Prog.vi" e "CRIO.vi" e ainda todo o código dos ficheiros "parâmetros_de_leitura.vi", "Parametros_Camara.vi", "processar_imagem.vi", "seleccao_posicao_seguinte.vi" e "Start_Motion.vi". Todos os outros ficheiros apresentados na imagem abaixo já foram devidamente explicados.

A organização dos ficheiros e o local onde cada um é executado está ilustrado na seguinte figura:

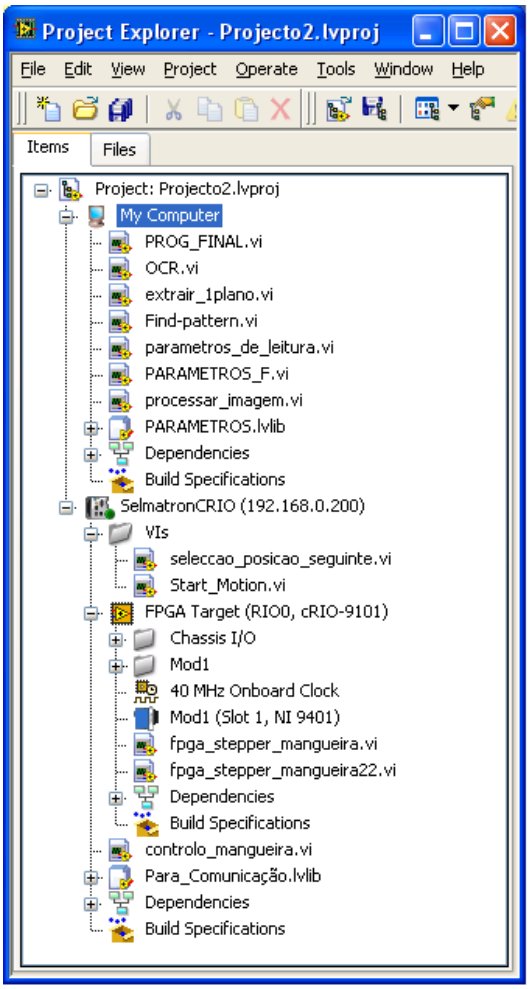


Figura 56 - Estrutura do projecto

A descrição do código de cada ficheiro será pela ordem que aparece na imagem acima.

9.1.1. PROG.VI

Este VI pode-se dividir em três partes, que correspondem às três funcionalidades presente no VI, das quais só serão apresentadas e explicadas duas, uma vez, que a terceira, que tem como função a concatenação dos caracteres reconhecidos, já foi explicada anteriormente neste documento.

Na seguinte imagem apresenta-se a porção de código referente a primeira funcionalidade, com todos os seus casos:

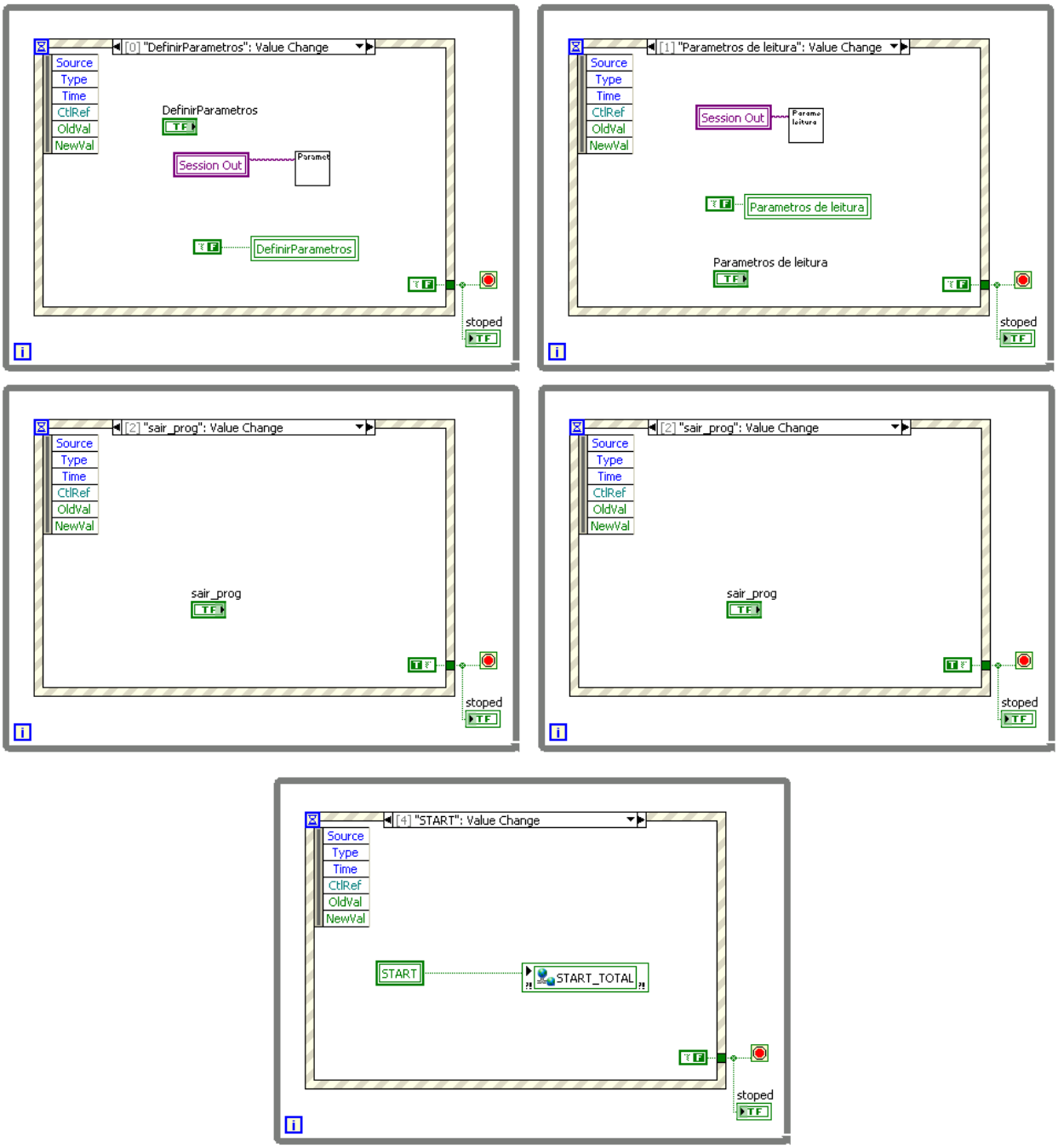


Figura 57 – Código presente no ficheiro "PROG.vi"

Esta tinha a função de executar as funcionalidades associadas aos controlos do interface gráfico de processamento de imagem, por exemplo, quando se carrega no botão “Parâmetros de leitura” este tem uma acção associada que é a de executar o VI “Parametros_de_leitura” o que faz então aparecer a janela flutuante descrita anteriormente.

Em seguida apresenta-se a segunda porção de código do VI ”PROG.vi”:

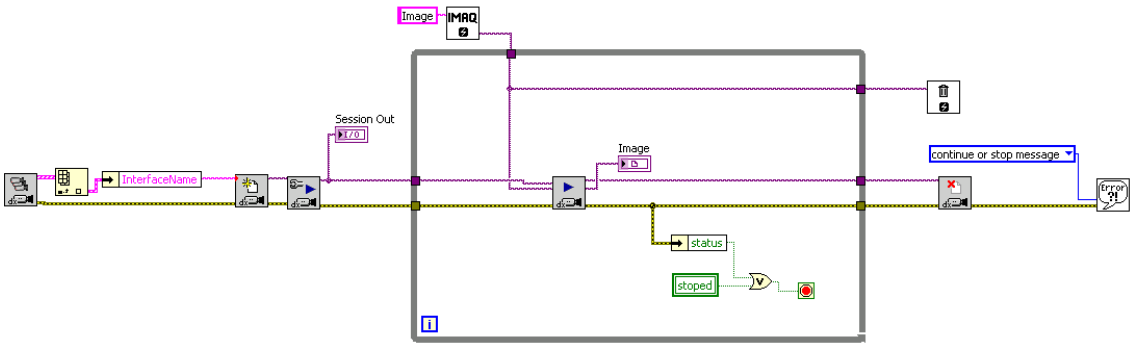
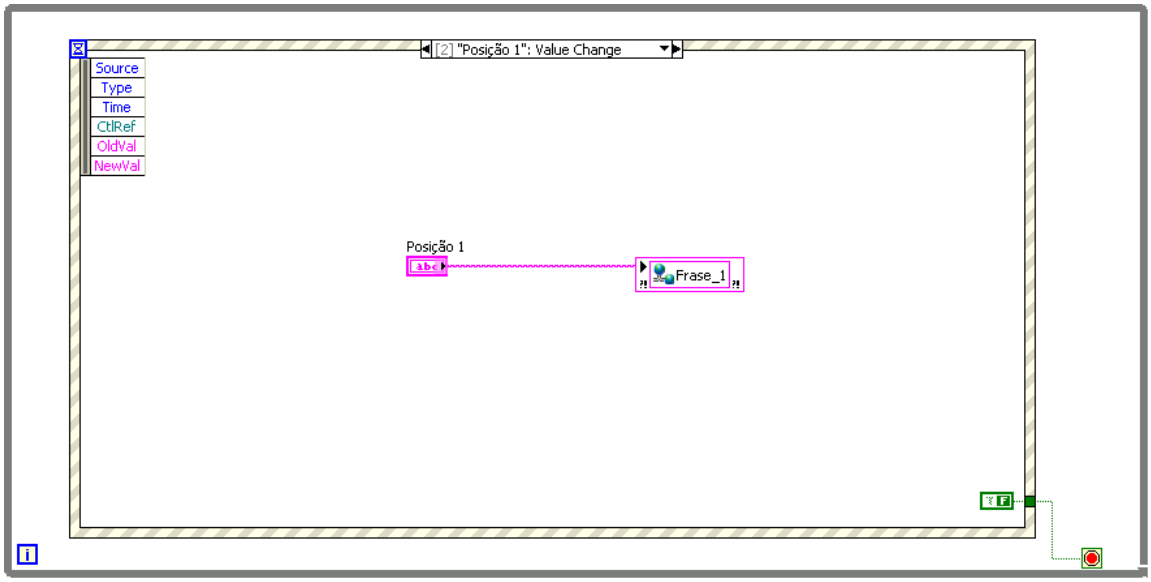


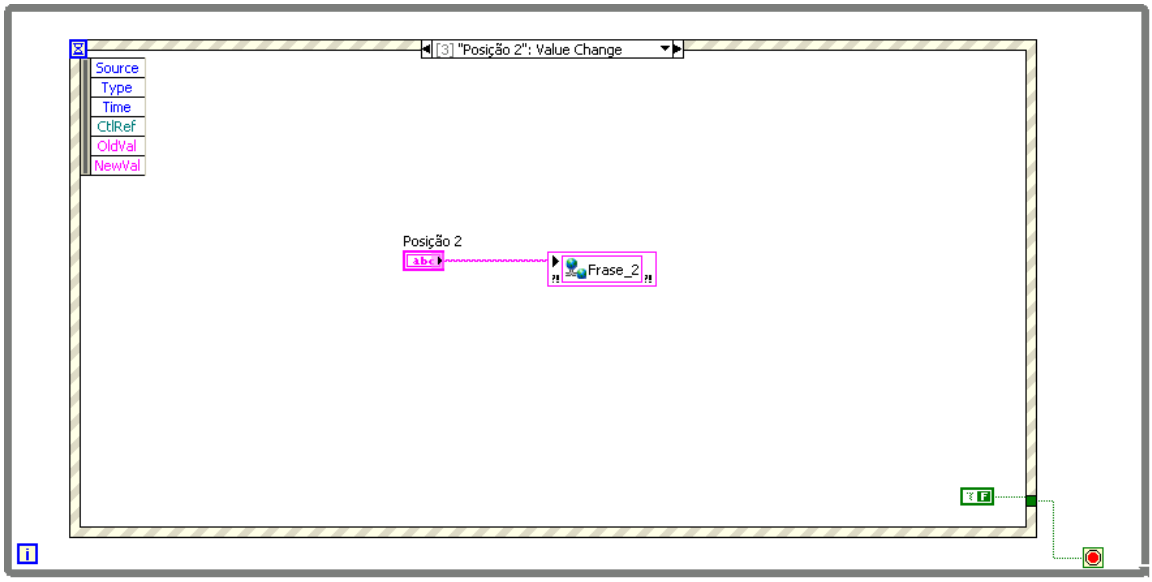
Figura 58 - Código para a aquisição de imagem

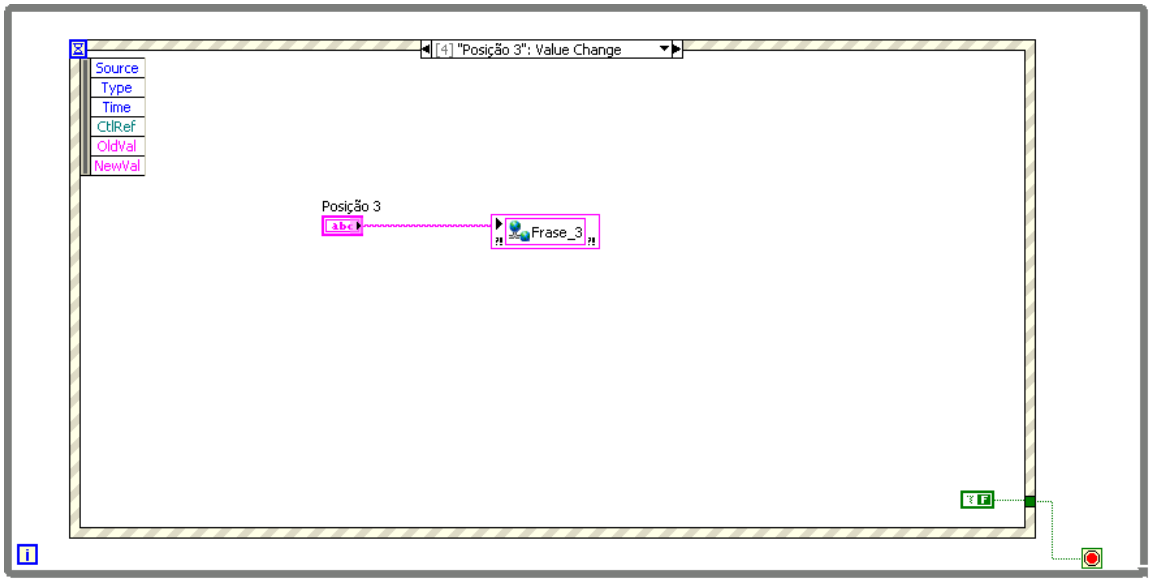
Que tinha o intuito de abrir uma sessão com a câmara e em seguida fazer a aquisição de imagem e quando fosse o pretendido fechar essa mesma sessão.

9.1.2. PARAMETROS_DE_LEITURA.VI

Tal como o anterior, este Vi também se pode dividir em duas partes. Uma delas é, igualmente, para executar as funcionalidades dos controlos do interface gráfico, como se pode verificar na figura seguinte:


[2] "Posição 1": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
Posição 1
Frase_1


[3] "Posição 2": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
Posição 2
Frase_2


[4] "Posição 3": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
Posição 3
Frase_3

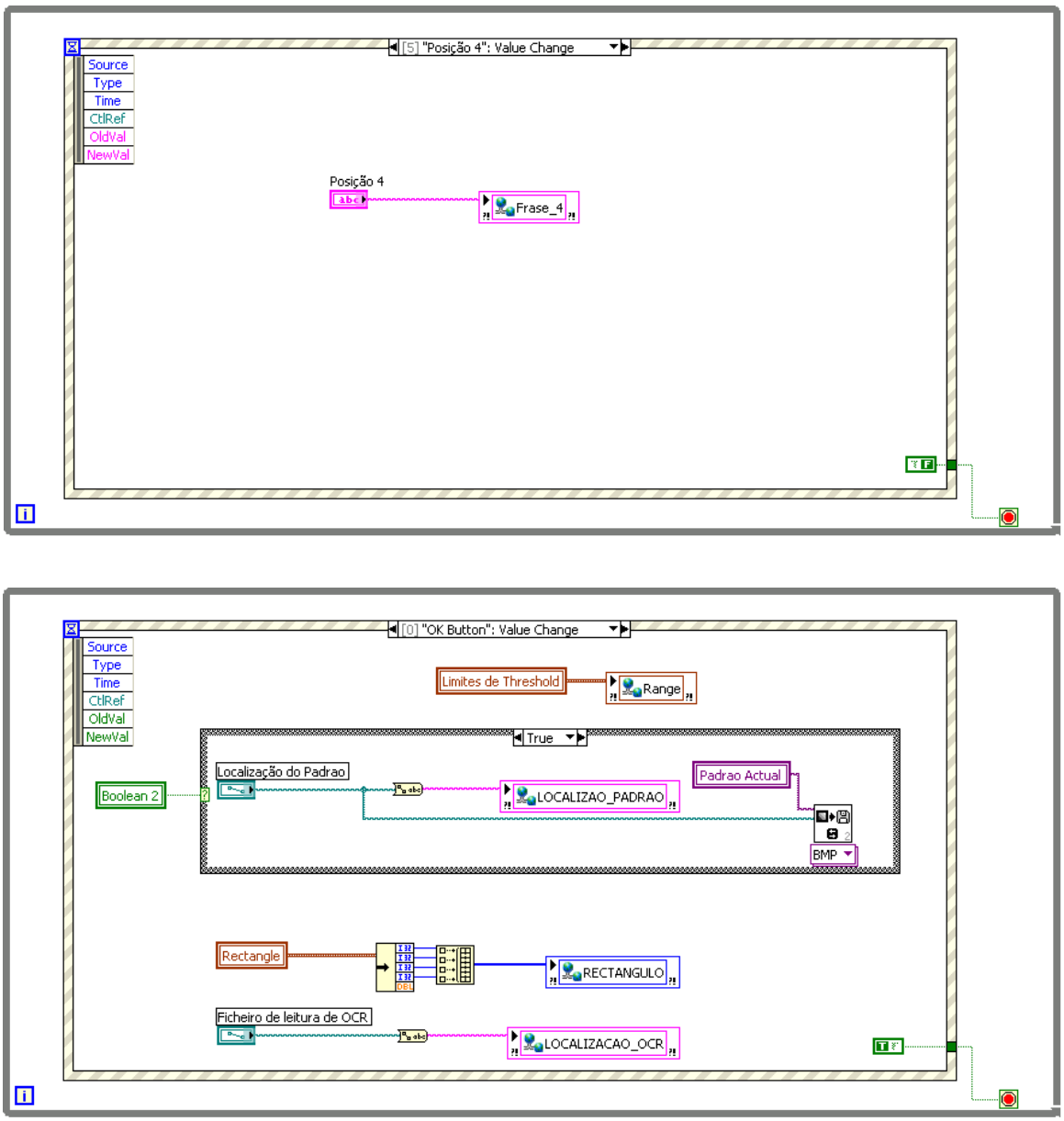


Figura 59 – Código do ficheiro "Parametros_de_leitura.vi"

A outra parte do código pertencente a este VI é o seguinte:

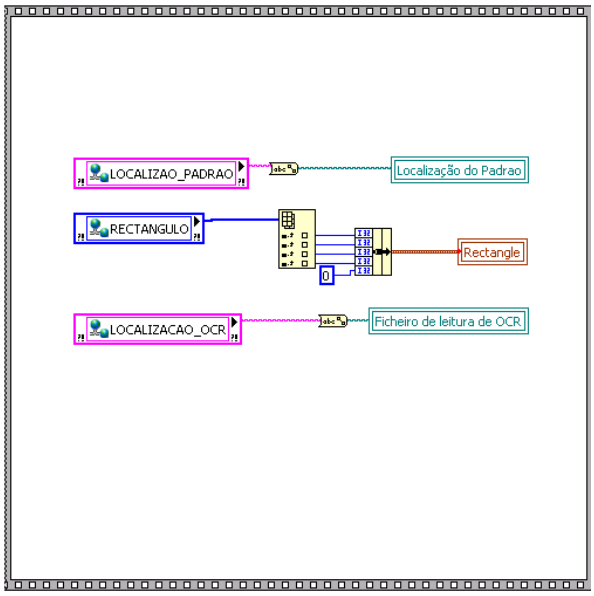


Figura 60 – Código de inicialização das variáveis

Onde são iniciadas todas as variáveis do VI, e ainda:

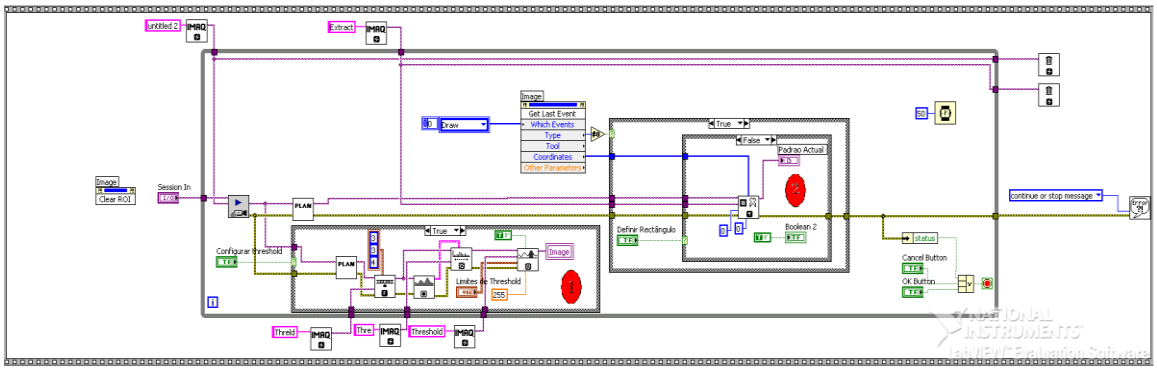


Figura 61 – Código pertencente ao ficheiro "Parametros_de_leitura.vi"

O código assinalado com o número 1, é o responsável pela funcionalidade de mostrar a imagem no interface gráfico de definição dos parâmetros de leitura, tal como ela é processada pelo algoritmo de OCR. Já o assinalado com o número 2 é o que permite a definição da ROI e da selecção do padrão na imagem.

9.1.3. PARAMETROS_CAMARA.VI

Neste Vi como o pretendido é somente reconhecer dados dos controlos do interface gráfico. Portanto o seu código resume-se a uma inicialização das várias variáveis e, posteriormente, a recolha dos seus novos valores através dos controlos do interface gráfico.

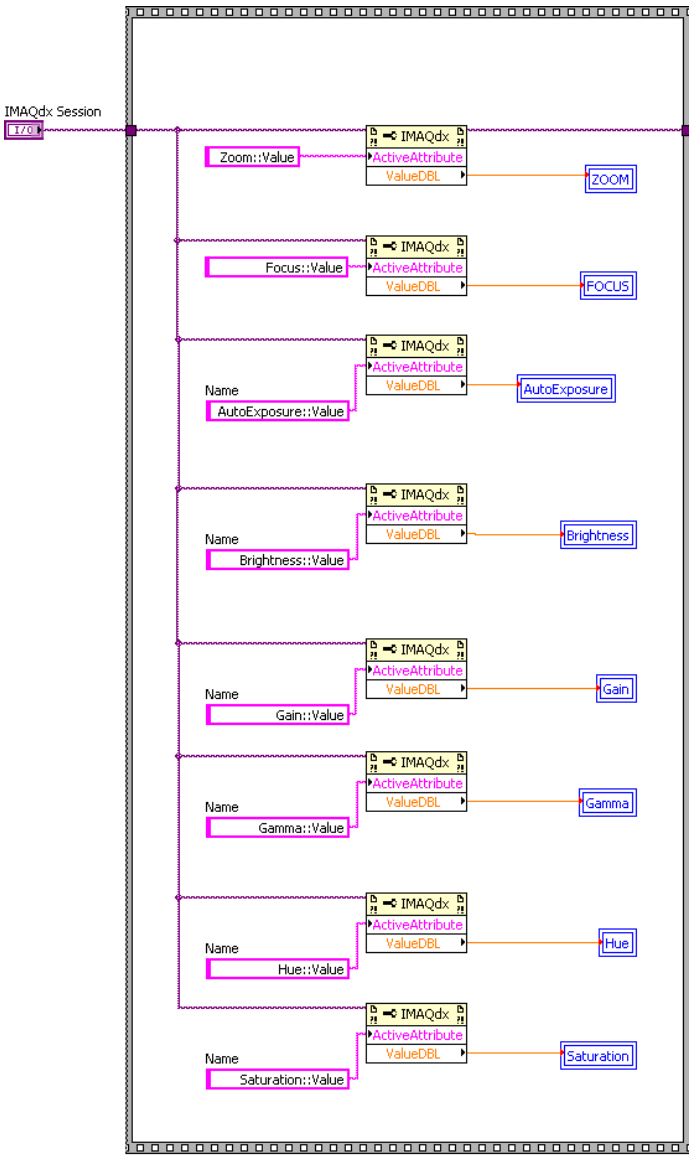
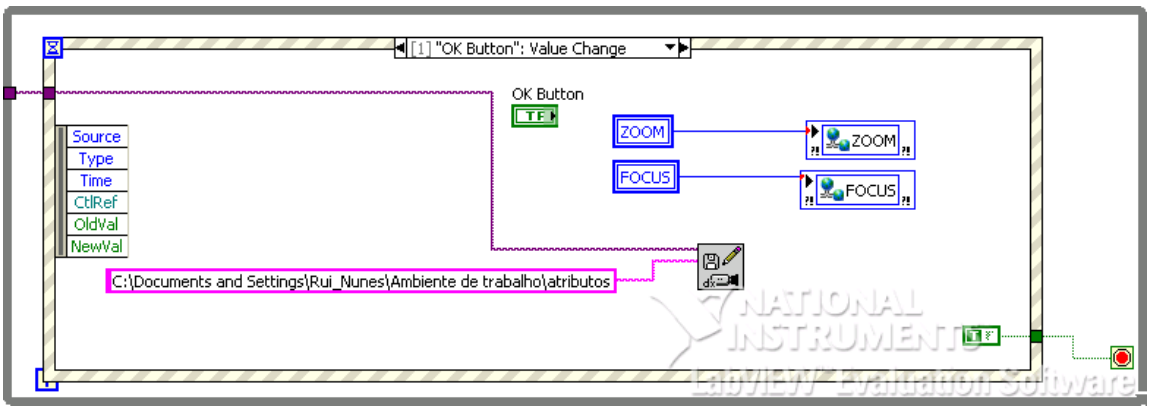
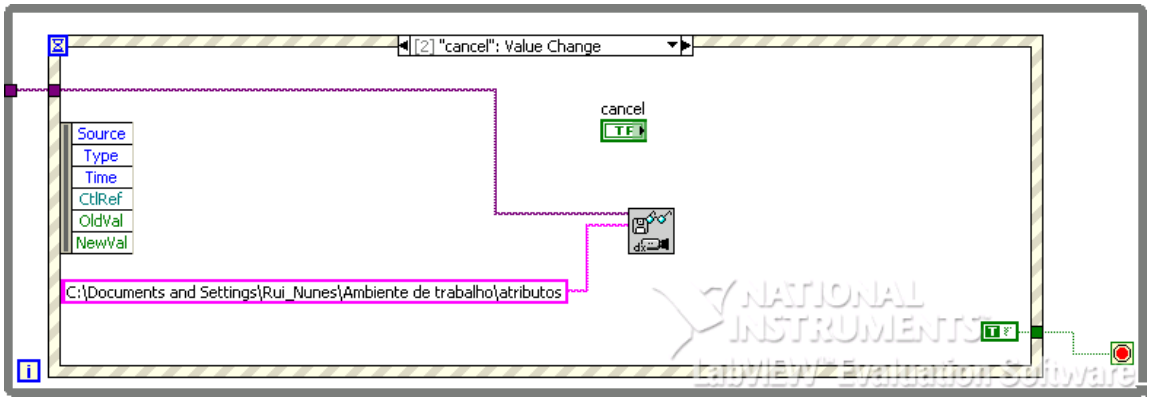
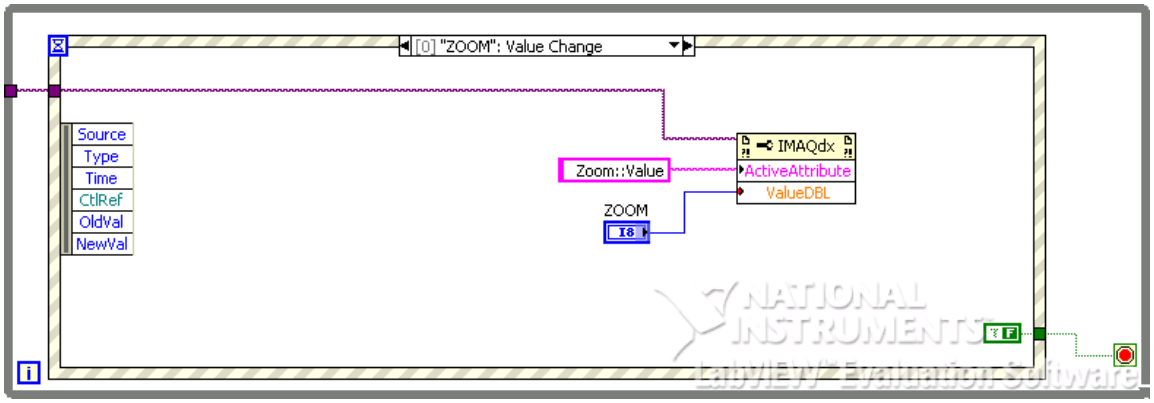
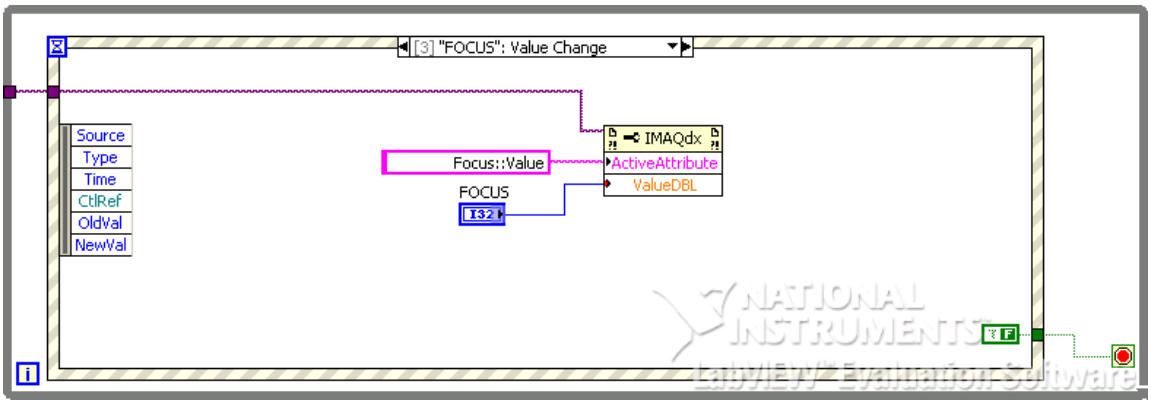


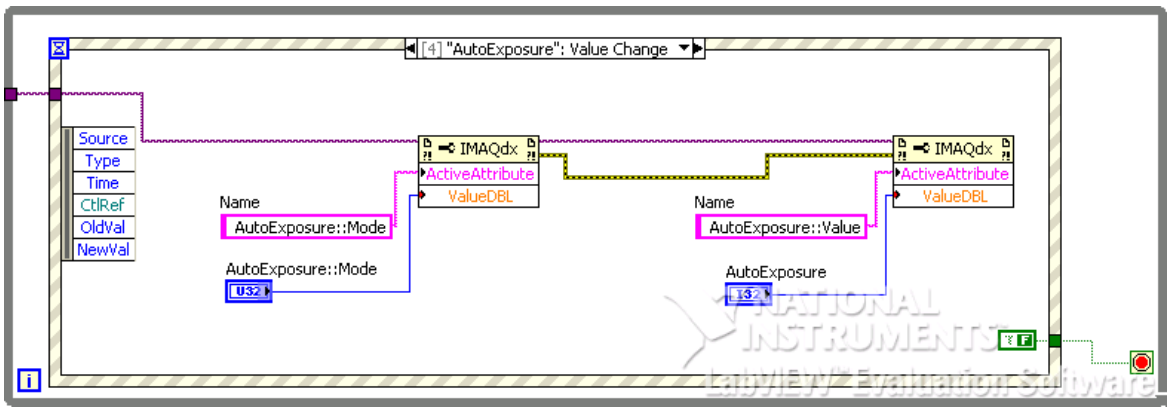
Figura 62 - Código de inicialização do ficheiro "Parametros_camara.vi"

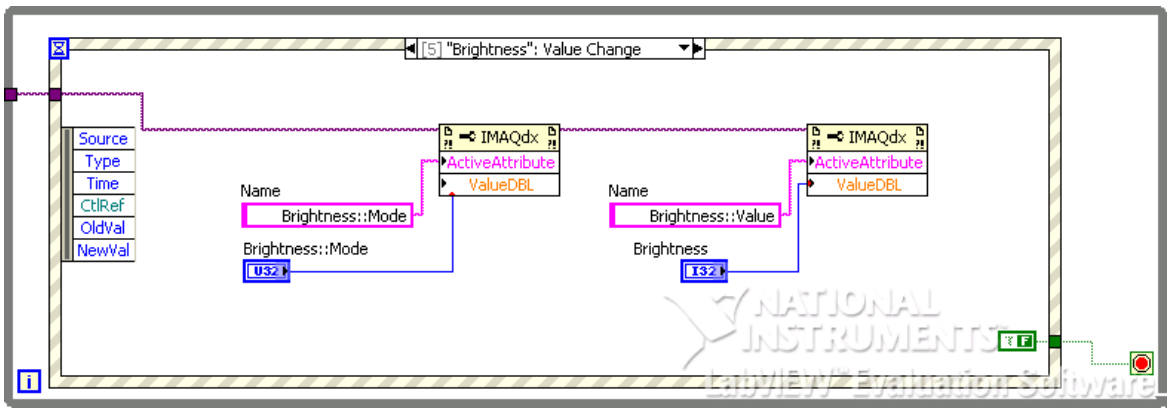

[1] "OK Button": Value Change
OK Button
TF
Source
Type
Time
CtlRef
OldVal
NewVal
ZOOM
FOCUS
ZOOM
FOCUS
C:\Documents and Settings\Rui_Nunes\Ambiente de trabalho\atributos
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software

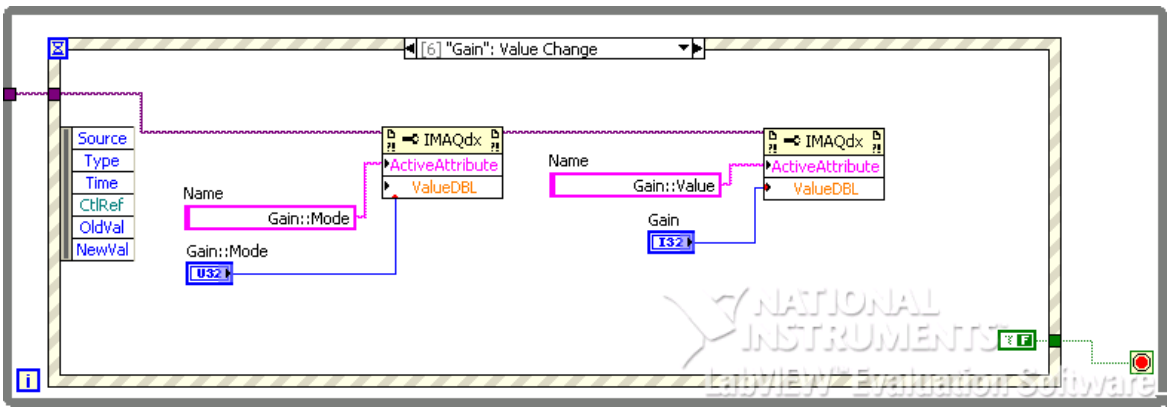

[2] "cancel": Value Change
cancel
TF
Source
Type
Time
CtlRef
OldVal
NewVal
C:\Documents and Settings\Rui_Nunes\Ambiente de trabalho\atributos
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software

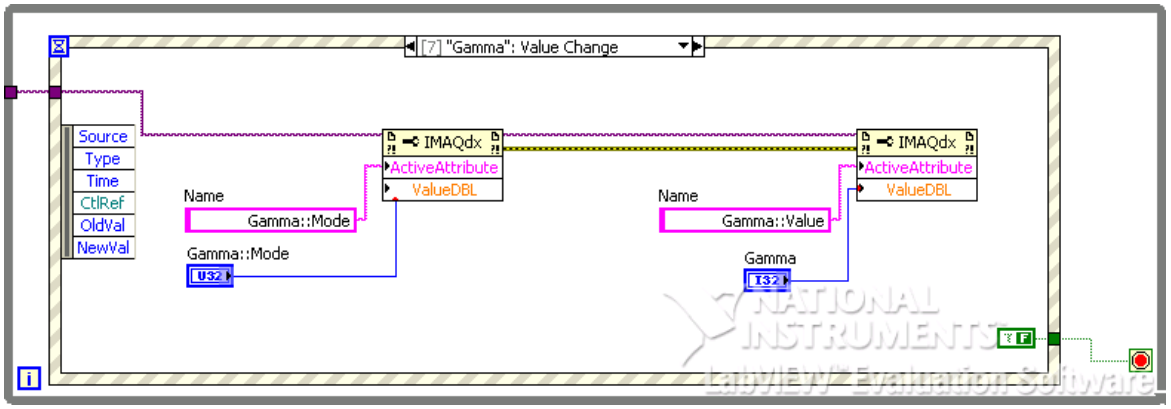

[0] "ZOOM": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
Zoom::Value
IMAQdx
ActiveAttribute
ValueDBL
ZOOM
I8
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software


[3] "FOCUS": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
Focus::Value
IMAQdx
ActiveAttribute
ValueDBL
FOCUS
I32
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software


[4] "AutoExposure": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
IMAQdx
ActiveAttribute
ValueDBL
Name
AutoExposure::Mode
AutoExposure::Mode
U32
Name
AutoExposure::Value
AutoExposure
I32
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software


[5] "Brightness": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
IMAQdx
ActiveAttribute
ValueDBL
Name
Brightness::Mode
Brightness::Mode
U32
Name
Brightness::Value
Brightness
I32
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software


[6] "Gain": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
IMAQdx
ActiveAttribute
ValueDBL
Name
Gain::Mode
Gain::Mode
U32
Name
Gain::Value
Gain
I32
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software


[7] "Gamma": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
IMAQdx
ActiveAttribute
ValueDBL
Name
Gamma::Mode
Gamma::Mode
U32
Name
Gamma::Value
Gamma
I32
NATIONAL INSTRUMENTS
LabVIEW Evaluation Software

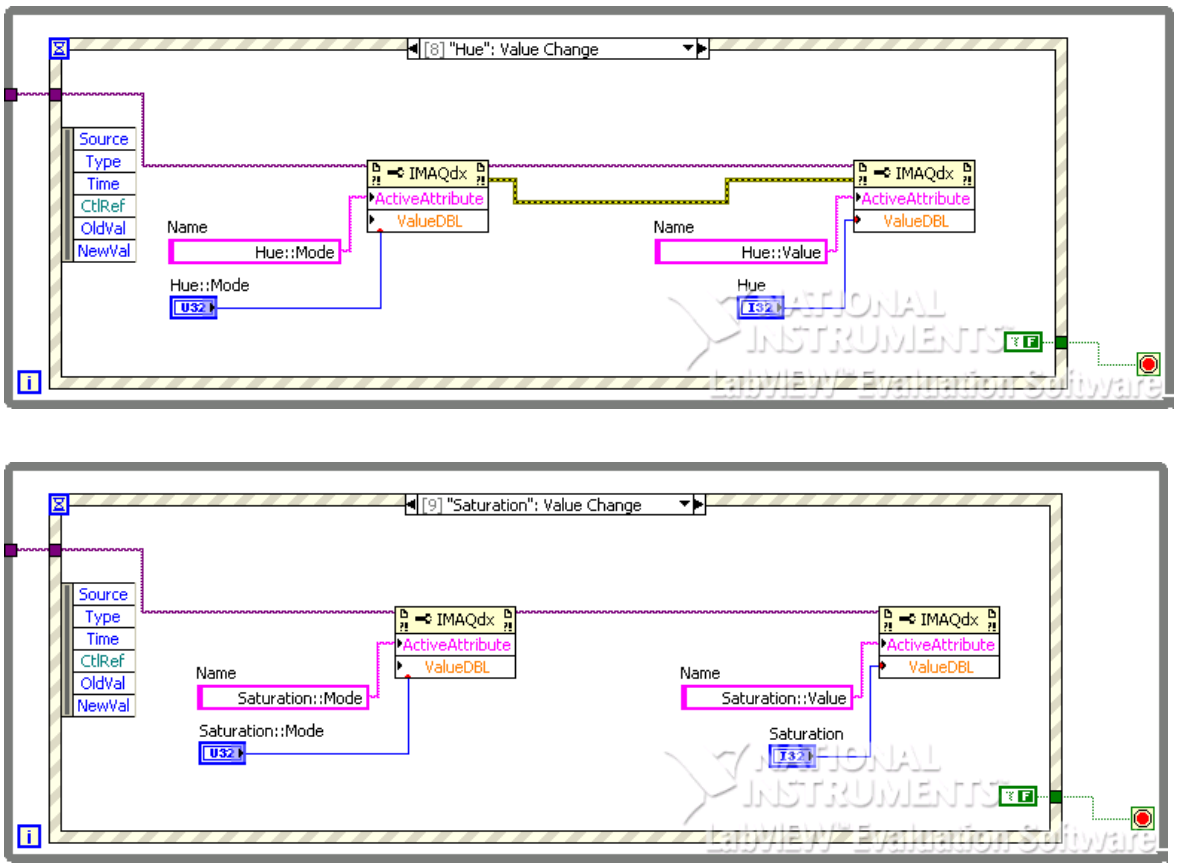


Figura 63 – Código de definição dos parâmetros da câmara

9.1.4. PROCESSAR_IMAGEM.VI

Este Vi serve, como o próprio nome indica para processar a imagem. É o responsável pela invocação do VI de localização de padrão e do VI de identificação de caracteres.

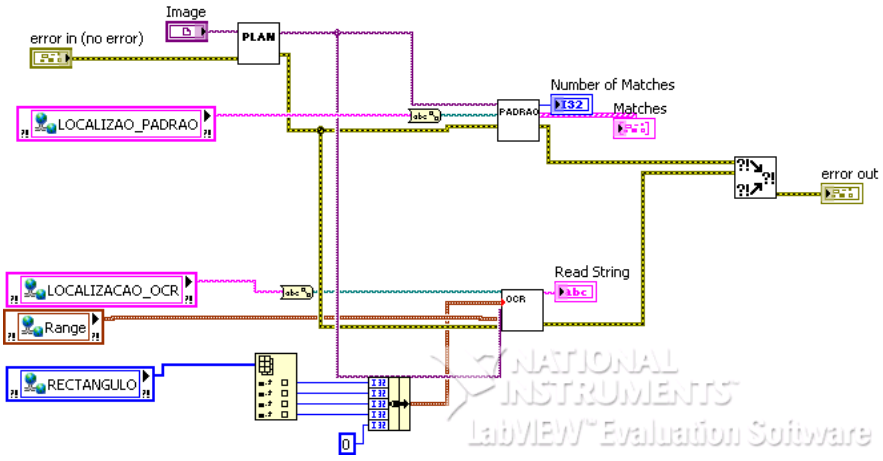


Figura 64 – Código do ficheiro "Processar_imagem.vi"

9.1.5. CRIO.VI

Relativamente, a este VI falta somente descrever duas parte do código. São elas a inicialização das variáveis da FPGA, como se mostra na seguinte figura:

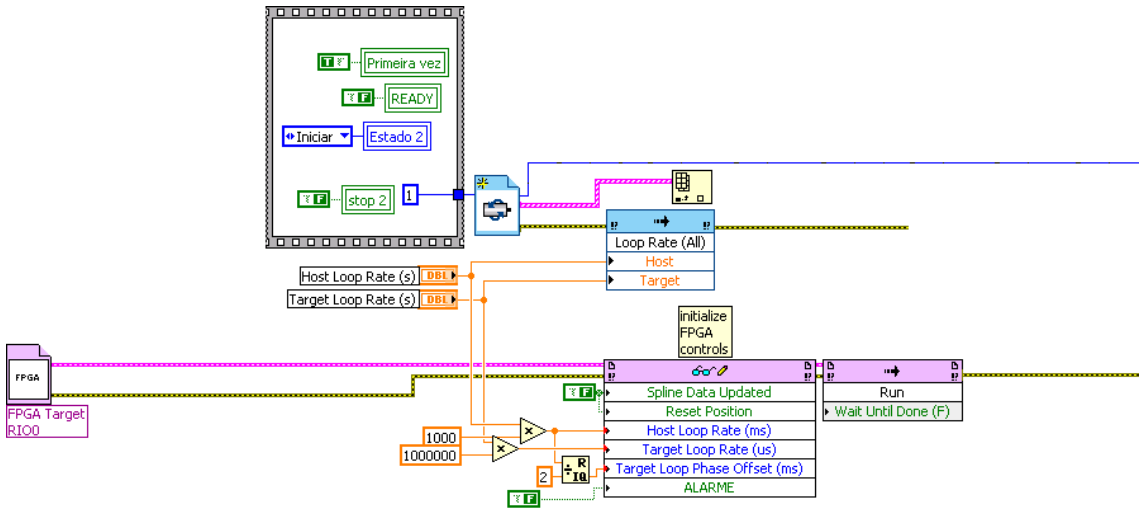


Figura 65 – Código de inicialização da FPGA

E o trecho de código que permite o control manual do posicionamento da câmara:

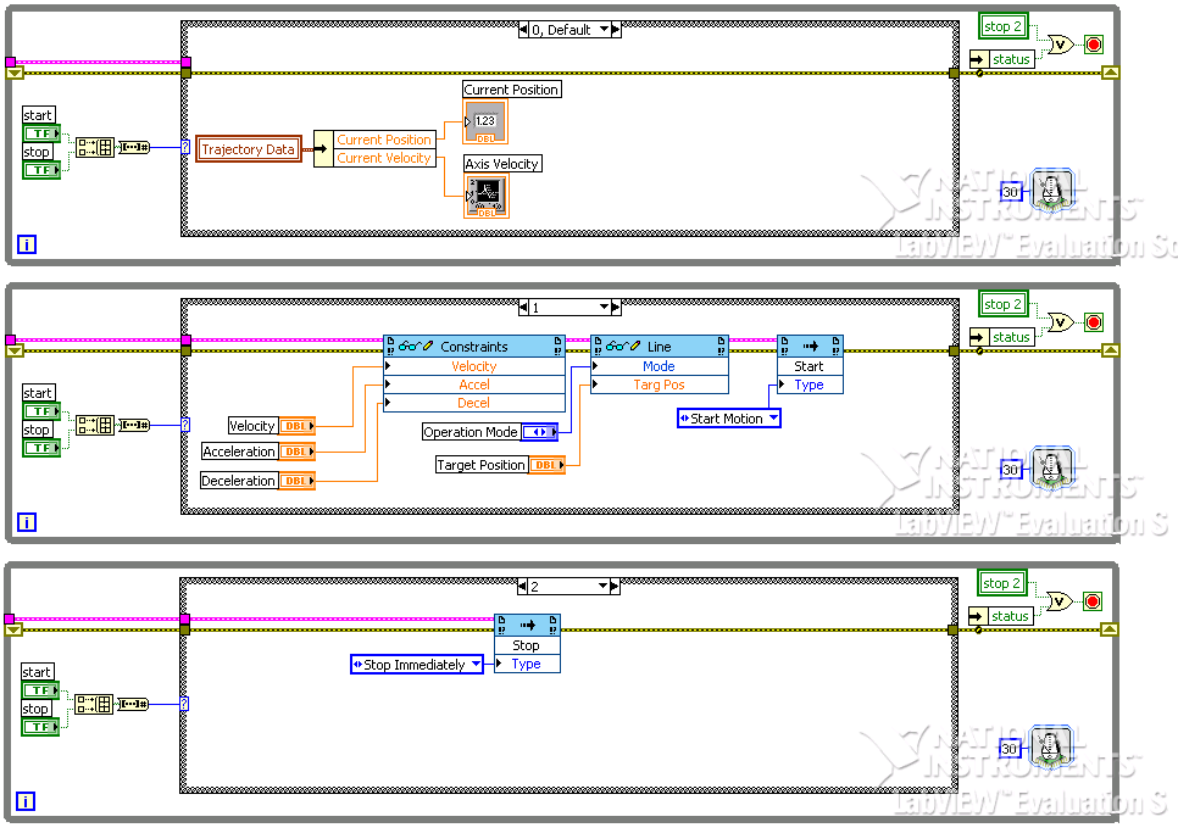
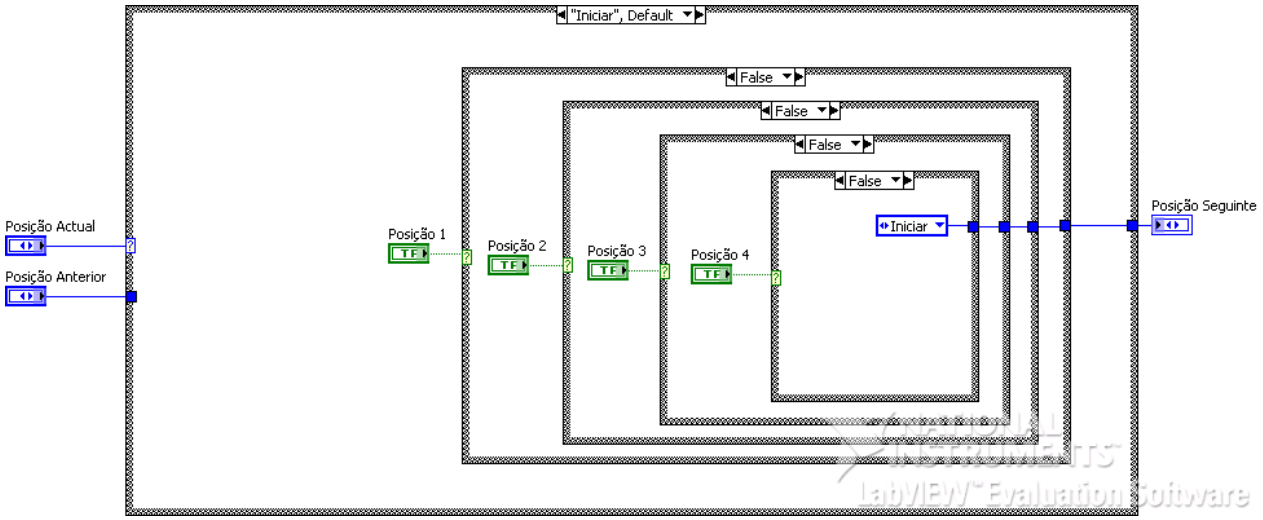
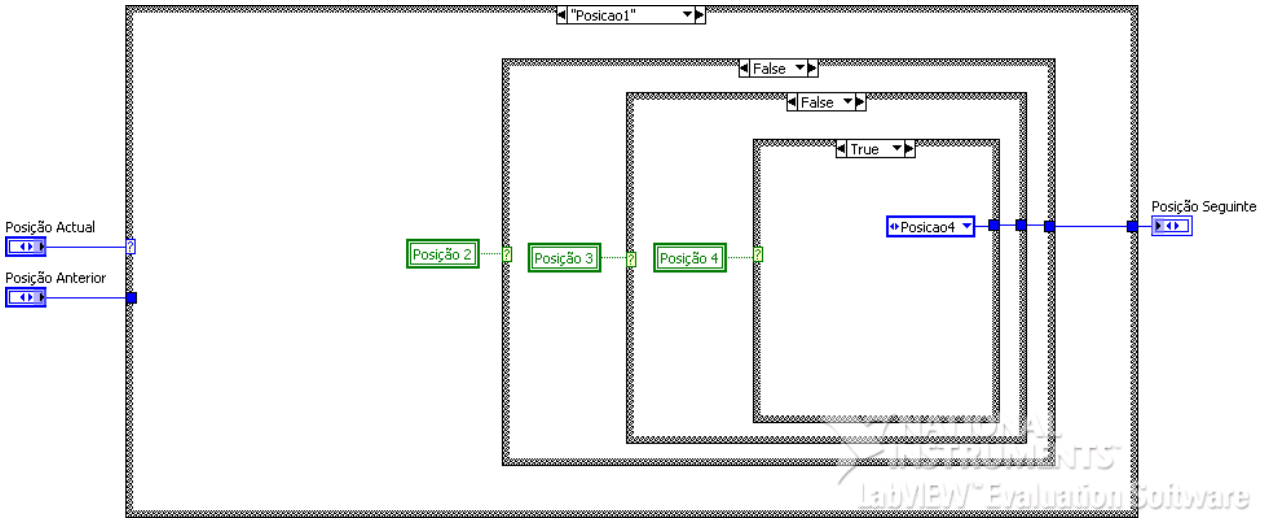


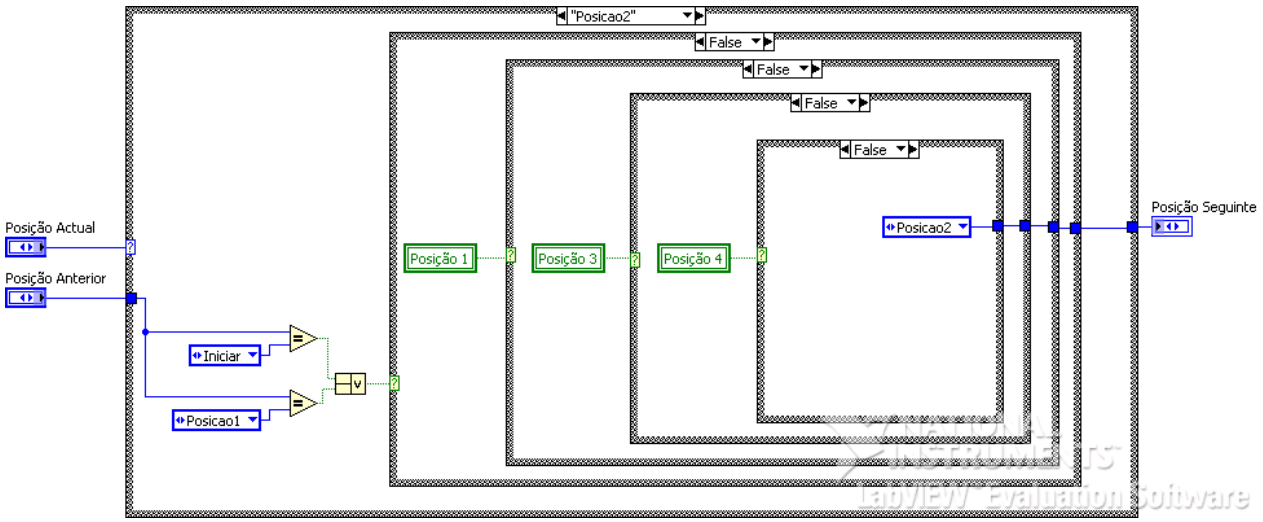
Figura 66 – Código que permite o controlo manual do motor

9.1.6. SELECCAO_POSICAO_SEGUINTE.VI

Este VI, é somente, uma sucessão de "if's", que tendo em conta a posição da câmara actual e as posições de inspecção decide a posição seguinte para a qual a câmara se deve deslocar. De seguida apresenta-se o código, mostrando todos os casos iniciais, detalhando-se posteriormente o caso em que a posição actual é a dois. O código para os restantes casos é igual, variando só as posições da câmara dentro dos "if´s".


"Iniciar", Default
False
False
False
False
Iniciar
Posição Actual
Posição Anterior
Posição 1
Posição 2
Posição 3
Posição 4
Posição Seguinte
LabVIEW Evaluation Software


"Posicao1"
False
False
True
Posicao4
Posição Actual
Posição Anterior
Posição 2
Posição 3
Posição 4
Posição Seguinte
LabVIEW Evaluation Software


"Posicao2"
False
False
False
False
Posicao2
Posição Actual
Posição Anterior
Iniciar
Posicao1
Posição 1
Posição 3
Posição 4
Posição Seguinte
LabVIEW Evaluation Software

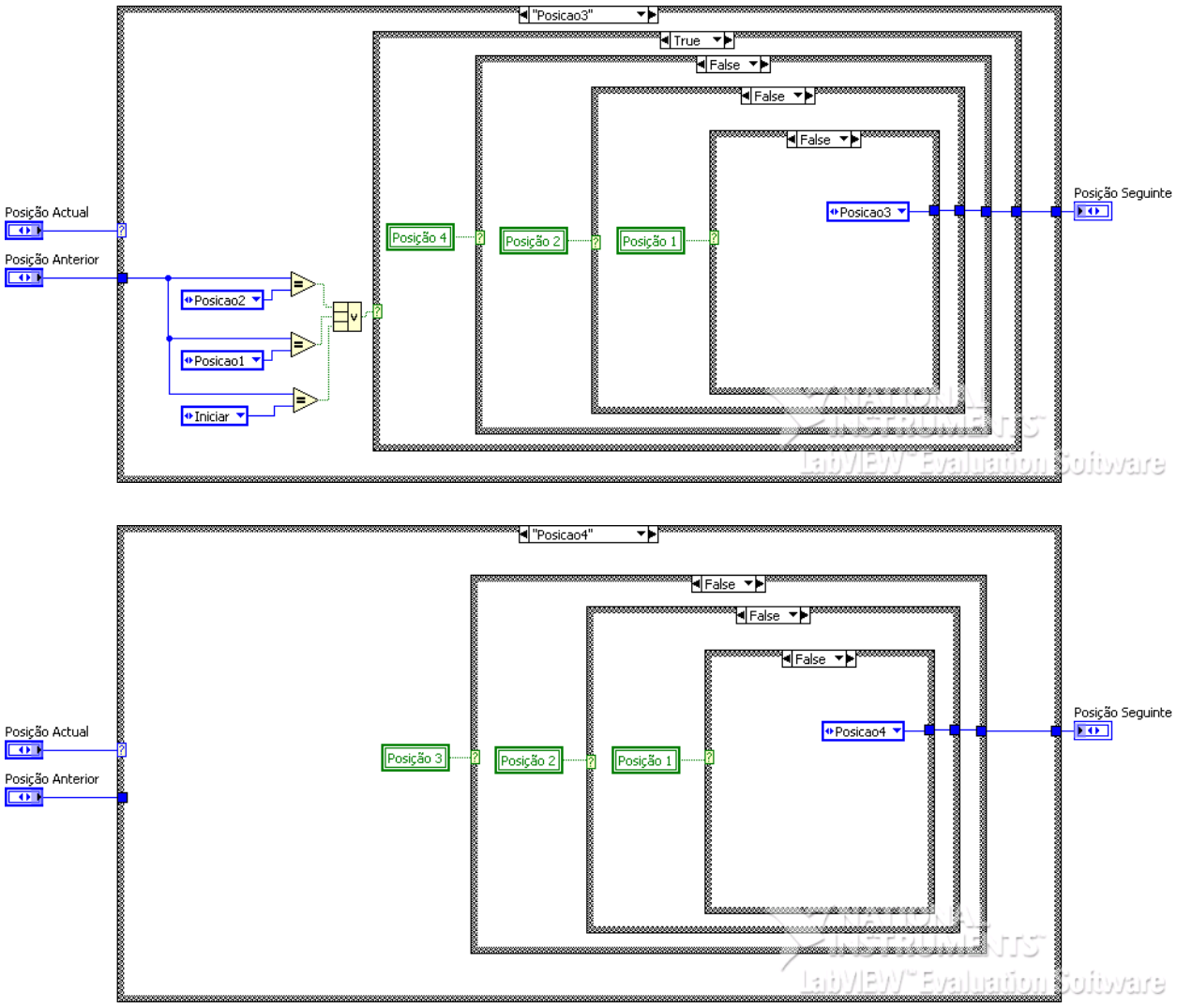
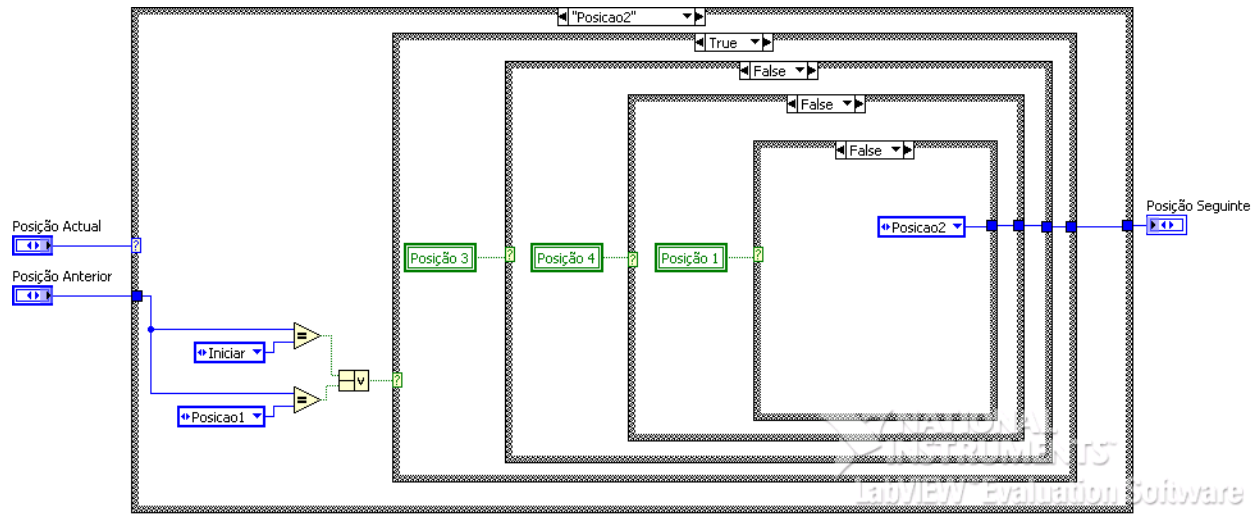


Figura 67 - Código do ficheiro "Seleccionar_posicao_seguinte.vi"



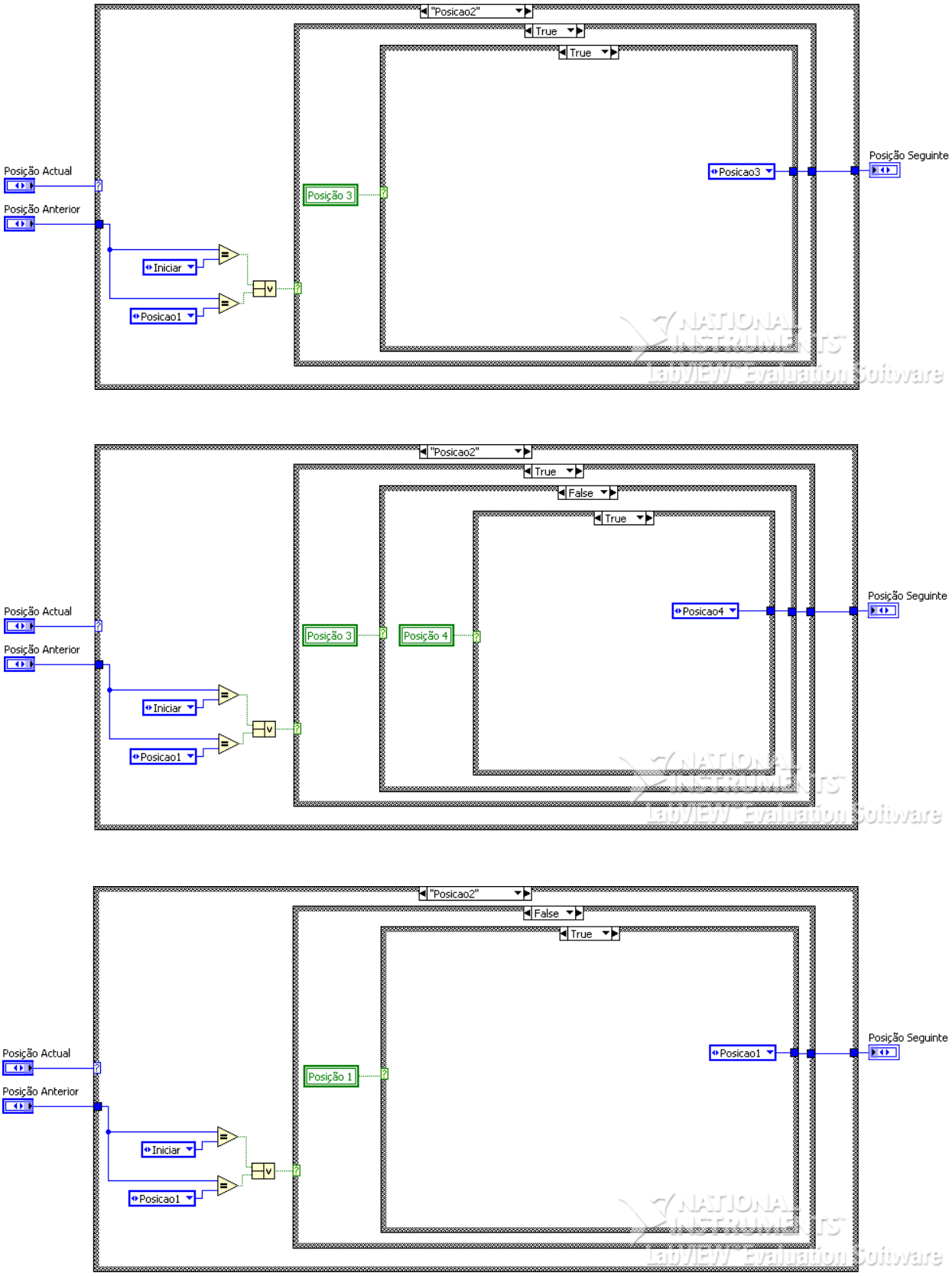


Figura 68 – Código para o caso específico da posição actual dois

9.1.7. START_MOTION.VI

Como o nome indica, este VI tem como função iniciar o deslocamento da câmara assim que todos os parâmetros estejam definidos.

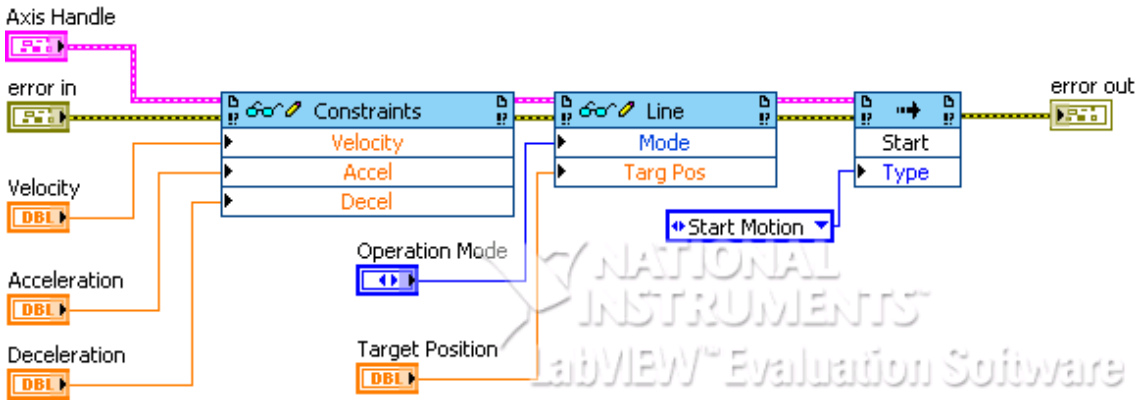


Figura 69 - Código do ficheiro "Start_Motion.vi"

LISTA DE ACRÓNIMOS

cRIO Compact Real Input Output

FPGA Field Programmable Gate Array

HDL Hardware Description Language

I/O Input/Output

NI National Instruments

OCR Optical Character Recognition

VI Virtual Instrument