PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA / ENGENHARIA ELÉTRICA
Trabalho Final de Curso II

Juan Manuel Morales Avalos

SISTEMA AUTOMÁTICO PARA CONTROLE DE VELOCIDADE AUTOMOTIVA – MÁXIMA PERMITIDA

Trabalho Final de Curso como parte dos requisitos para obtenção do título de bacharel em Engenharia Elétrica, apresentado à Pontifícia Universidade Católica de Goiás.

BANCA EXAMINADORA:

Prof. Dr. Carlos Augusto Guimarães Medeiros – Orientador. PUC Goiás.
Prof. Me. Carlos Alberto Vasconcelos Bezerra – PUC Goiás.
Prof. Me. Clebes André da Silva – PUC Goiás.

Goiânia, 22 de junho de 2022.

SISTEMA AUTOMÁTICO PARA CONTROLE DE VELOCIDADE AUTOMOTIVA-MÁXIMA PERMITIDA

Juan Manuel Morales Avalos, orientando, Engenharia Elétrica, PUC Goiás

***Resumo* — Este trabalho tem como objetivo desenvolver um subsistema de controle de velocidade para automóveis. Foi realizado de forma experimental, em bancada, composto pelo Arduino Uno R3, motor de indução trifásico e inversor de frequência. Além disso, apresenta uma proposta de implementação do Código de Barra 128 em placas de fiscalização de limite de velocidade. Após experimentos laboratoriais, obteve-se resultados de controle de velocidade do motor em bancada, de acordo com limites estabelecidos por placas de trânsito virtuais. Assim, obtém-se melhor assistência ao motorista.**

***Palavras-chave* — Fiscalização Eletrônica, Limite de Velocidade, Automação, Código de Barras 128.**

I. INTRODUÇÃO

O radar de velocidade fixo ou portátil, instrumento de fiscalização eletrônica de trânsito, é utilizado para controlar a velocidade de circulação definida nas vias urbanas e rurais com o intuito de manter o funcionamento adequado do trânsito. A legislação que regulamenta o uso dos radares é a Resolução 798/20 do Conselho Nacional de Trânsito (CONTRAN) [1]. Ela especifica que a fiscalização eletrônica precisa ter a presença de placas R-19 de Limite de Velocidade para alertar o condutor da presença de um radar fixo ou portátil.

No Brasil existem 3 tipos de infrações que podem levar a multa e até mesmo suspenção da CNH (Carteira Nacional de Habilitação), quando é ultrapassado o limite de velocidade em trechos de fiscalização eletrônica. A primeira infração é considerada média quando a velocidade for superior a 20% da máxima permitida na via, com multa de R$130,16 e 4 pontos na CNH. A segunda infração é considerada grave quando a velocidade estiver superior de 20% a 50% da máxima permitida com multa de R$ 195,23 e 5 pontos na CNH. A terceira infração é considerada gravíssima quando ultrapassa 50% da máxima permitida, com multa de R$880,41 e 7 pontos na CNH, podendo ter sua suspensão imediata e também o direito de dirigir.

Adaptação Inteligente de Velocidade (ISA) consistem de sistemas implementados em veículos, que garantem que a velocidade estabelecida da via não seja ultrapassada pelo condutor. Assim, em caso de excesso, o condutor é alertado e a velocidade é reduzida automaticamente para a máxima permitida da via [2]. Os Sistemas ISA mais conhecidos no mercado automobilístico mundial são: O Controle de Cruzeiro Moderno (*Modern Cruise Control – MCC*) e o Controle de Cruzeiro Adaptativo (*Adaptive Cruise Control – ACC*).

Mesmo com o Sistema de Fiscalização Eletrônica e penalidades de multas e outras, as infrações por excesso de velocidade são frequentes. Este trabalho visa propor um tipo de subsistema evitando que essas infrações nos trechos onde há fiscalização eletrônica de limite de velocidade sejam cometidas.

Assim, com o subsistema acoplado em um sistema ISA já existente no automóvel, não importando se o motor é de combustão ou elétrico, a imagem de uma placa de Fiscalização Eletrônica ao ser captada, terá sua velocidade ajustada automaticamente, mostrando no painel do veículo informações como o tempo e o distanciamento do percurso entre a placa e o radar como descrito na Resolução Nº 798.

Para o subsistema ter resultados mais concretos, sugere-se modificar a Placa de Limite de Velocidade adicionando um código de barras logístico unidimensional denominado Código 128. Assim é necessário a utilização de uma Câmera de Leitor de códigos de barras por imagem para captar a imagem da placa, decodificando o código nela inserido.

II. PROPOSTA

Como já mencionado este trabalho tem como objetivo desenvolver um subsistema capaz de capturar e reconhecer imagens das placas de

fiscalização eletrônica de limite máximo de velocidade que se encontram nas vias urbanas, rodovias e estradas, reduzindo as infrações por excesso de velocidade. Este subsistema atende tanto a captura de imagens sem código quanto a leitura com código 1D ou 2D. Quando uma imagem com Código de Barras 128 que é unidimensional (1D) é capturada e reconhecida em sua memória, são enviadas as informações armazenadas do tipo da placa do trecho, da velocidade máxima permitida, do distanciamento, e o tempo de travamento para os sistemas ISA. Essas informações, por sua vez, variam e ajustam a velocidade do automóvel de acordo com o Código de Trânsito Brasileiro (CTB), mostrando no painel do veículo o distanciamento do percurso da placa e radar, e quanto tempo a velocidade permanecerá constante através de um cronômetro. A Fig. 1 mostra um fluxograma do funcionamento da proposta.

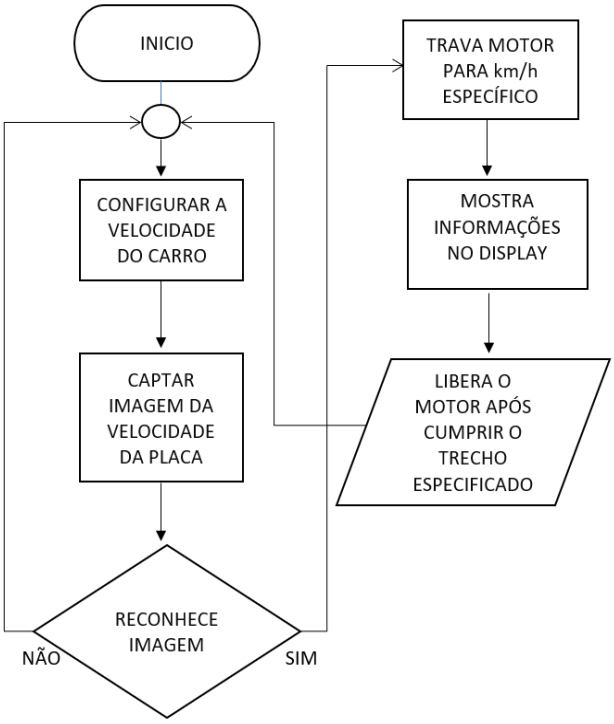

Fig. 1. Fluxograma do Subsistema Integrado ao Sistema.

Atualmente nos trechos urbanos, rodovias e estradas, câmeras compostas com tecnologia de leitor de reconhecimento óptico de caracteres, que capturam a imagem da placa do veículo, registram a velocidade medida em km/h e estando acima da máxima permitida, o condutor do veículo é penalizado de acordo com o Código Brasileiro de Trânsito.

Então, por esta proposta, a câmera de imagem digital captará a imagem da placa normal enquanto o automóvel estiver em movimento. Esta câmera ficará localizada junto ao retrovisor interno do veículo. A imagem capturada será enviada para o registrador de memória do Arduino UNO R3 que verificará a existência desta imagem no seu arquivo de dados e, se reconhecida, o subsistema enviará um sinal de comando para os Sistemas ISA do veículo, seja ele MCC ou ACC, após receber o sinal do subsistema, certos comandos serão executados. No sistema MCC por exemplo, uma mensagem no painel do veículo será automaticamente exibida, alertando o condutor para ajustar a velocidade do veículo. Após o ajuste da velocidade pelo condutor, um cronômetro será ativado e exibido no painel e mostrará em segundos o tempo em que o veículo deverá permanecer nesta velocidade até chegar ao ponto do medidor de velocidade. O tempo de travamento depende do distanciamento da placa de fiscalização até o medidor de velocidade, dividido pela máxima permitida pela legislação de trânsito para o trecho, ver a Eq. (1).

$$t = \frac{distanciamento\ km}{velocidade\ km/h} \quad (1)$$

Já no sistema ACC, a velocidade do veículo automaticamente será travada sem aparecer a mensagem de redução de velocidade no painel e será destravado da mesma maneira que no sistema MCC.

Um esquema do sistema em blocos está descrito na figura 2.

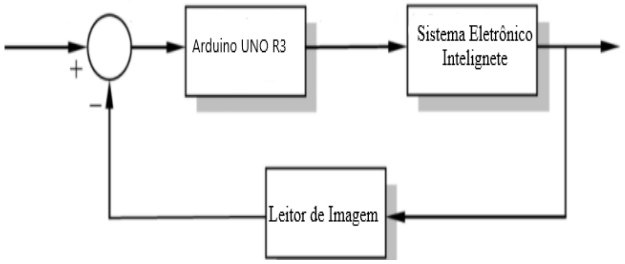

Fig. 2. Sistema de função do produto proposto.

É necessário a implementação do Código de Barras 128 nas placas de fiscalização eletrônica de limite máximo de velocidade para melhorar o subsistema proposto.

A câmera de imagem digital do subsistema pode ser substituída por um leitor de código de barras por imagem que, por sua vez, poderá inovar os sistemas já existentes no mercado, como também facilitar a captura de imagens e extrair com menos complicações as informações codificadas no

código de barras 128 para executar a variação de velocidade no veículo.

Na Fig. 3 e Fig. 4 tem-se dois exemplos de placas de fiscalização eletrônica de velocidade máxima. Uma sem o Código de Barras 128 e outra com o código.

Fig. 3. Placa de Fiscalização Eletrônica De Velocidade Máxima Atual.

Fig. 4. Placa de Fiscalização Eletrônica de Velocidade Máxima com o Código de Barras 128.

III. FUNDAMENTO TEORICA

A. Sistema de Adaptação de Velocidade Inteligente

Como já mencionado na introdução deste artigo, Adaptação Inteligente de Velocidade (ISA) são sistemas que garantem que o veículo não ultrapasse a velocidade estabelecida pelo condutor. O sistema de Controle de Cruzeiro Moderno – MCC encontrado em alguns tipos de automóveis podem ter dois funcionamentos. O primeiro funcionamento é o Limitador de Velocidade - "*LIMIT*": esta função impede que o veículo ultrapasse a velocidade programada pelo condutor. O segundo funcionamento é o Regulador de Velocidade - "*CRUIZE*": regula automaticamente a velocidade do carro ao valor programado pelo condutor sem que o motorista tenha que manter o pé no acelerador [3]. Para ativar qualquer uma das funções, é preciso selecionar o modo de "LIMIT" ou "CRUISE" por meio de um botão, acelerar ou desacelerar o veículo até a velocidade desejada e pressionar a tecla de ativação do limitador ou regulador. A desativação de qualquer uma das funções é feita por meio da tecla desativação do limitador ou regulador ou pisando no freio.

Com o progresso da tecnologia, os fabricantes de automóveis têm sido capazes de desenvolver Sistemas ISA mais sofisticados. Um destes sistemas é conhecido no mercado automotivo como Controle de Cruzeiro Adaptativo (*Adaptive Cruise Control – ACC*) que pode variar a velocidade de um carro de acordo com a movimentação do tráfego [4]. O ACC fornece melhor assistência ao motorista do que o sistema MCC já que o sistema ACC vem equipado com sensores de radar ou *laser*, para manter o intervalo de distanciamento entre automóveis [5]. A ativação da maioria dos sistemas ACC se encontra no volante do automóvel onde, simultaneamente, o condutor regula a velocidade desejada e a distância em relação ao veículo da frente. Definida a velocidade e o distanciamento de seguimento, o sistema mantém o veículo nessa velocidade, desde que não haja nada à sua frente. Se, por exemplo, um veículo estiver na frente com uma velocidade menor, o sistema automaticamente desacelera e reduz a velocidade para manter a distância que foi programada. Uma vez que o veículo não esteja mais a frente, o sistema acelera e retoma para a velocidade programada.

B. Código de Barras 128

O Código de Barra 128 é um código de barras logístico unidimensional (1-D) de alta densidade e mais eficaz que outros códigos de barras unidimensionais. Composto por uma coleção de espaços em branco e barras pretas, que representam caracteres alfabéticos, números e sinais de pontuação, pode codificar todos os 128 caracteres da Tabela ASCII (*American Standard Code for Information Interchange*). Os dados codificados em um código de barras unidimensional, após decodificados, são utilizados para acessar informações guardadas em registros de memória como as de um computador. Quando o Código 128 é identificado, a informação guardada no registro é decodificada [6]. O comprimento de um Código 128 como outros códigos de barras unidimensionais depende da finalidade, da dimensão mínima da largura da barra e da quantidade de dados codificados [7]. Um Código de Barras 128 é composto das seguintes seções:

1. Símbolo de zona quieta
2. Símbolo de início
3. Símbolo de dados codificados
4. Símbolo de verificação
5. Símbolo de parada
6. Símbolo de zona quieta

Cada símbolo é composto por um total de 11 unidades de largura, 3 barras e 3 espaços, onde cada barra ou espaço pode ser composto por 1, 2, 3 ou 4 unidades de largura [8].

A Fig. 5 mostra a composição de um Código de Barra 128.

Fig. 5. Seções do Código de Barra 128.

C. Exemplo de Codificação

A *string* é VELOCIDADE MAXIMA 40km/h.

	Posição	Valor	Total	Barras/Espaços
Start B		104	104	11010010000
V	1	54	54	11101011000
E	2	37	74	10001101000
L	3	44	132	10001101110
O	4	47	188	10001110110
C	5	35	175	10001000110
I	6	41	246	11000100010
D	7	36	252	10110001000
A	8	33	264	10100011000
D	9	36	324	10110001000
E	10	37	370	10001101000
ESPAÇO	11	0	0	11011001100
M	12	45	540	10111011000
A	13	33	429	10100011000
X	14	56	784	11100010110
I	15	41	615	11000100010
M	16	45	720	10111011000
A	17	33	561	10100011000
ESPAÇO	8	0	0	11011001100
4	19	20	380	11001001110
0	20	16	320	10011101100
k	21	43	903	10110001110
m	22	77	1694	11110111010
/	23	15	345	10111001100
h	24	72	1728	10011000010

A seguir, na Fig. 6, mostra-se o exemplo do código que foi obtido da *string* VELOCIDADE MAXIMA 40km/h de um gerador de Código de Barras 128.

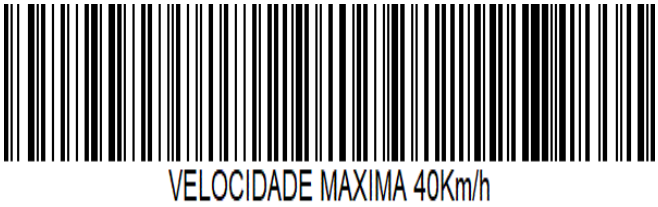

Fig. 6. Velocidade Máxima 40 km/h.

D. Decodificador

O Código 128 pode ser reconhecido por um *hardware* denominado leitor de código de barras. Este leitor é um dispositivo com luzes, lentes e um sensor que decodifica e captura as informações contidas nos códigos de barras. O leitor usa um raio *laser* como fonte de luz e, geralmente, empregam espelhos oscilantes que refletem a luz do raio *laser* para frente e para trás por todo o código de barras unidimensional. Um fotodiodo mede a luz refletida das barras do código no que resulta na geração de um sinal analógico, o sinal analógico é então convertido em um sinal digital.

Além disso, o hardware como leitor de código de barras por imagem que usa um sensor de matriz de área, semelhante aos encontrados nas câmeras digitais, para obter uma imagem dos códigos de barras 1D e 2D, está altamente sendo utilizado nas indústrias e substituindo os leitores de códigos de barras a lazer. Leitores de código de barras por imagens utilizam um microprocessador, executando um *software* especial de processamento de imagens que localiza e decodifica o código antes de distribuir os dados resultantes em uma rede [9].

E. Arduino UNO R3

O Arduino UNO R3, ver Fig. 7, é uma placa micro controladora programável de código aberto, fácil de usar, flexível e de baixo custo que pode ser integrada a uma variedade de projetos eletrônicos. Possui microcontrolador AVR Atmega328, 6 pinos de entrada analógica e 14 pinos de entrada/saída digitais, dos quais 6 são usados como saída PWM. Esta placa contém uma interface USB usada para conectá-la a um computador e ao *software* Arduino IDE (Ambiente de Desenvolvimento Integrado). O *software* é usado para programar a placa. Mais informações sobre o Arduino Uno R3 podem ser obtidas de seu manual [10].

Fig. 7. Placa de Arduino UNO R3.

F. Reconhecimento e Processamento de Imagem

O reconhecimento de imagem é a capacidade de uma câmera alimentada por computador de identificar e detectar objetos ou recursos de uma imagem. O reconhecimento de imagem é alcançado pelo uso de técnicas de algoritmos como a de redes neurais convolucionais para filtrar

imagens por meio de uma série de camadas de neurônios artificiais que utilizam a combinação de técnicas como *pooling* máximo, configuração de passada e preenchimento. O processo do algoritmo, é receber uma imagem como entrada e gerar como saída o que a imagem contém [11]. Para um algoritmo saber o que uma imagem contém, ele precisa ser treinado para aprender as diferenças entre classes. É um processo bem complexo.

IV. METODOLOGIA

Foi desenvolvido em laboratório a simulação de um veículo elétrico (EV) utilizado para variar as velocidades, integrando nele a simulação de um subsistema como proposto neste estudo. A Fig. 8 mostra os montagem do veículo elétrico [12].

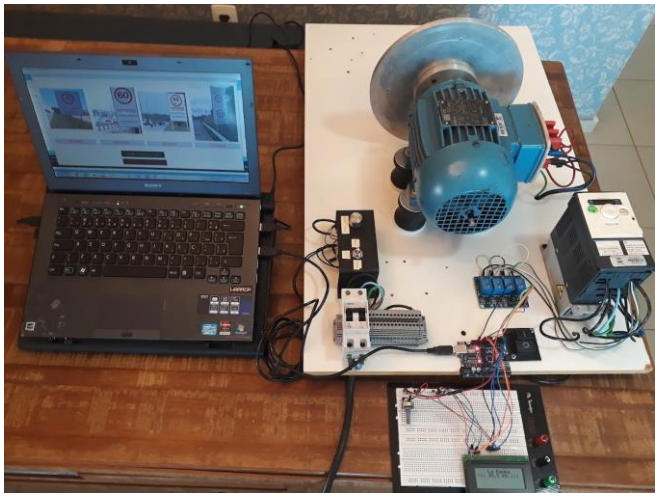

Fig. 8. Simulação de um veículo elétrico.

V. DESENVOLVIMENTO

Os componentes utilizados neste experimento foram um inversor de frequência da Schneider Eletric ATV312H075M2 com tensão nominal de entrada e saída de 200...240, e frequência nominal de 50...60 hertz, interligado a um motor Weg. Motor de indução – gaiola de 4 polos, com conexão triângulo para obter uma entrada elétrica de 220 volts. Nas pinagens +10V, AI1 e COM do inversor foi implementado um potenciômetro para controlar a variação de frequência do ATV312H075M2 que na comparação ao automóvel elétrico neste projeto ele é equivalente ao acelerador. Um disco de ferro galvanizado de aproximadamente 2 kg, e diâmetro de 25cm, foi instalado no eixo do motor de forma equivalente a uma roda de automóvel. Uma conexão monofásica de 220 volts diretamente da rede elétrica com saída de 60 hertz foi conectada para alimentar o inversor. No menu do inversor ATV312H075M2 utilizado, se encontra uma memória reservada para preestabelecer até 16 frequências para usos aleatórios. As frequências escolhidas e preestabelecidas foram para fazer uma comparação às velocidades preestabelecidas em um veículo real com um sistema ISA instalado. Foi definido empregar o sistema ACC com a finalidade de poder ajustar e variar a velocidade do automóvel quando uma imagem de placa de limite de velocidade sem o com um código de barra 128 fosse detectado pelo subsistema.

Foi utilizado um LCD 16x4, display que possui 16 colunas e 4 linhas para escrita, interligado com o controlador Arduino UNO R3 para mostrar a velocidade da placa de Fiscalização Eletrônica em km/h, o tempo e o percurso, após captura de uma imagem de placa de limite de velocidade. Este display foi usado para simular o painel do automóvel. A Eq. (2) mostra como a velocidade do veículo mostrada no display foi calculada.

$$v = r \times \omega \quad (2)$$

Onde:

v = velocidade linear
r = raio da roda
ω = velocidade angular = RPM_{roda}

$$RPM_{roda} = \frac{120 * f}{p} \quad (3)$$

f = frequência
p = números de polos do motor.

Para medir a velocidade angular descrita na Eq. (3), foi utilizado um tacômetro digital a laser para aferir o RPM a cada variação de frequência preestabelecidas na memória do ATV312H075M2.

A Tab. 1 mostra algumas conversões de RPM para velocidade em km/h.

Tabela 1
RPM para Velocidade

	Frequência preestabelecidas (Hz)	**RMP_{real}**	**Velocidade (km/h)**
1	28.3	849	40
2	42.4	1273	60
3	56.6	1698	80
4	63.6	1910	90
5	77.8	2334	110

Além da visualização da velocidade no LCD 16x4, um cronômetro foi programado para fazer a contagem decrescente do tempo como descrita na equação (1), como também a contagem decrescente do distanciamento que precisa ser percorrido,

enquanto a velocidade do automóvel se mantém constante do ponto da captura da imagem da placa de Fiscalização Eletrônica de Limite de Velocidade Máxima, até o medidor de velocidade.

O reconhecimento de imagem é a capacidade de um sistema ou software de identificar objetos, pessoas ou lugares em forma de imagens**.** Ele usa tecnologias de visão de máquina com inteligência artificial e algoritmos treinados para reconhecer imagens por meio de um sistema de câmeras digitais.

Para demonstrar a simulação de captura de imagem de placas de Fiscalização Eletrônica, e do Código de Barras 128, foi usado o *software Matlab* 2017b, (*Image Processing Toolbox*), que permite gravar arquivos de imagens em formatos como BMP, GIF, JPEG, PNG e TIFF, e extraí-las quando solicitadas. Esse software foi a opção mais assertiva tanto para câmera digital, quanto para o Leitor de Código de Barras por Imagem integrado ao veículo elétrico.

Os sistemas compostos e testados, se integrados a outros, são capazes de criar seu próprio sistema. A figura 9 mostra como estes subsistemas foram interligados experimentalmente.

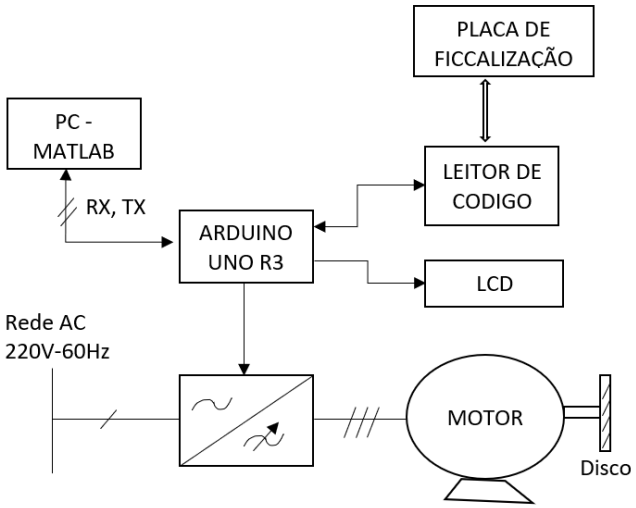

Fig. 9. Diagrama de blocos experimental.

No primeiro subsistema foi utilizado o Arduino UNO R3 como controlador, interligado a 4 de seus pinos digitais se encontram 4 relés de 5v de entrada para 24v de saída, utilizados para extrair as frequências preestabelecidas na memória do inversor ATV312H075M2 após os reconhecimentos das imagens. Outros 6 pinos digitais também foram conectados ao Arduino para utilização do display LCD 16x4.

Também uma conexão USB Arduino-computador foi necessária para ligar o Arduino UNO R3, que por sua vez, faz a transferência de dados de comandos via programação de *softwares* Matlab-Arduino. Os códigos programados no software Arduino IDE (Ambiente de Desenvolvimento Integrado) igual aos códigos de comando da utilização da ferramenta *Image Processing Toolbox* podem ser vistos na parte do apêndice deste trabalho.

O segundo subsistema ao invés da utilização de reconhecimento de imagens via a ferramenta *Image Processing Toolbox* do *software Matlab* 2017b, foi utilizado *software* Arduino IDE e acoplado ao Arduino UNO R3 uma placa USB *Host Shield* 2.0. Foi conectado o leitor de código de barra unidimensional *Elgin ccd flash* na entrada da placa USB *Host Shield* 2.0, ele foi usado como comparação ao leitor de código de barras por imagem para fazer leitura e extrair as informações codificadas dentro do código de barras 128. Os códigos programados no software Arduino IDE para o funcionamento deste subsistema se encontram na parte do apêndice deste trabalho.

VI. Resultados

Ficou comprovado experimentalmente que o primeiro subsistema composto pela combinação de uma câmera de Imagem digital, simulada através da ferramenta *Image Processing Toolbox*, em conjunto com o controlador Arduino UNO R3, as frequências preestabelecidas na memória do inversor ATV312H075M2 foram extraídas com sucesso após reconhecimento de imagens, a cada imagem reconhecida, o motor variou sua velocidade. Também informações como a velocidade em km/h, o tempo em segundos em que o motor ficou em velocidade constante e o percurso em metros entre a placa de Fiscalização Eletrônica até o radar foram visualizadas a cada reconhecimento de imagem, no LCD 16X4.

Já no segundo subsistema onde foi utilizado o leitor de código de barras Elge como simulador de um leitor de código de barras por imagem, em conjunto com Arduino UNO e a placa USB HOST SHIELD 2.0 não foram obtidos os resultados esperados como no subsistema 1, por falta de tempo estabelecido para esse trabalho, sendo assim não foi possível fazer outros experimentos e comprovar que a simples leitura de código de barras 128 pode mudar um problema que até o momento não teve solução concreta.

VII. DISCUSSÃO

Após os estudos e experimentos no laboratório com os subsistemas foi discutido que o reconhecimento entre a captura de uma placa com Código de Barras 128 para uma placa sem o Código de Barras 128, é a complexidade de treinamento de imagens entre ambas, uma placa sem o Código de Barras 128 precisa ser treinada por algoritmos complexos como o algoritmo de rede neural convolucional, enquanto em uma placa com Código de Barras 128 é necessário apenas a leitura do mesmo para extrair as informações nele contidas. Na discussão sobre o custo entre ambos, o subsistema por leitura de código de barras foi o mais econômico, além do menos complexo para fabricação.

VIII. CONCLUSÃO

Após a realização dos experimentos no laboratório, os resultados obtidos pelo primeiro subsistema foram altamente positivos. Assim, o Sistema Automático para Controle de Velocidade Automática – Velocidade Máxima proposto, pode ser sugerido como complemento de integração aos Sistemas ISA, reduzindo as infrações por excesso de velocidade no território nacional brasileiro e trazendo melhorias tecnológicas inovadoras. O sistema proposto para o uso de controle de velocidade no território nacional em comparação com os sistemas já proposto existentes, é que esse é mais eficaz pelo fato dele utilizar as regras descritas na RESOLUCAO Nº780, como também mostrar para o usuário no painel do automóvel as informações do tempo em segundos e o distanciamento em metros, garantindo que não seja excedida a velocidade máxima permitida na via, assim evitando que o condutor não cometa infrações por excesso de velocidade.

Como sugestão para futuros estudos, sugere-se utilizar outros tipos de placas com diferentes microcontroladores instalados, para obter resultados não conquistados neste estudo. Também, sugere-se que a utilização de uma câmera digital e um leitor de código por imagem sejam adaptados nesses futuros estudos para ter resultados da variação de velocidade do motor após a captura de imagem em tempo real.

IX. AGRADECIMENTOS

Agradeço a Dom Emmanuel Gomes de Oliveira que em 1948, sendo Arcebispo de Goiana, lança a ideia de criar a primeira Universidade do Centro-Oeste. Agradecesse como igual, a Dom Fernando Gomes dos Santos, Arcebispo de Goiânia, que organizou a Universidade de Goiás. Também, agradeço o cardeal polonês Zenon Grocholewski, que reconheceu a Universidade Católica de Goiás (UCG) como Pontifícia Universidade Católica de Goiás (PUC Goiás). Pela visão, esforço e expansão destes educadores da palavra do bem, orgulhosamente posso disser, que a boa educação recebida nesta instituição me torno como num homem prudente que construiu a sua casa sobre a rocha.

REFERÊNCIAS

[1] Brasil. Resolução N°798, de 2 de setembro de 2020. Diário Oficial da União, Brasília, DF,9, setembro de 2020. Seção 1, p.43.

[2] Lehtonen, Esko, Neha Malhotra, Nicola J. Starkey, and Samuel G. Charlton. "Speedometer monitoring when driving with a speed warning system." *European transport research review* 12, no. 1 (2020): 1-12.

[3] Shaout, A., and M. A. Jarrah. "Cruise control technology review." *Computers & electrical engineering* 23, no. 4 (1997): 259-271.

[4] KHAIR, OSAMA. ADAPTIVE CRUISE CONTROL BASIC DESIGN. Diss. NEAR EAST UNIVERSITY, 2019.

[5] Marsden, Greg, Mike McDonald, and Mark Brackstone. "Towards an understanding of adaptive cruise control." *Transportation Research Part C: Emerging Technologies* 9, no. 1 (2001): 33-51

[6] Johnston, Robert B., and Alvin Khin Choy Yap. "Two-dimensional bar code as a medium for electronic data interchange." *International Journal of Electronic Commerce* 3, no. 1 (1998): 86-100.

[7] Fales, J. F., and R. S. Vincent. *Datamatrix and PDF417 data integrity test*. No. ORNL/SUB-93-SL161. Oak Ridge National Lab., TN (United States); Ohio Univ., Athens, OH (United States). Center for Automatic Identification Education and Research, 1993.

[8] Sawant, Ms Sayali Kapdule1 Mr Shreyas, and Mr Prathamesh Sukale3 Mr Vinayak Malavade. "Application of Barcode Technology as Improved Examination System."

[9] Cognex. Introdução à Leitura Industrial de Códigos de Barras (Compreensão do funcionamento interno dos códigos 1D e 2D, métodos de impressão e marcação e tipos de leitores de código de barras,pp.12.

[10] Arduino UNO R3, “Product Reference Manual,” SKU: A000066, Modified: 23/05/2022.

[11] Uchida, Seiichi. "Image processing and recognition for biological images." *Development, growth & differentiation* 55.4 (2013): 523-549.

[12] Cheng, Ka Wai Eric. "Recent development on electric vehicles." In *2009 3rd International Conference on Power Electronics Systems and Applications (PESA)*, pp. 1-5. IEEE, 2.

APÊNDICE A – PROGRAMA ARDUINO – MATLAB

(1)

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2);

char x; //cria variavel x tipo char
int rele1 = 10;
int rele2 = 9;
int rele3 = 8

float distancia;

void desativar_pinos(){
   digitalWrite(rele1, LOW);
   digitalWrite(rele2, LOW);
   digitalWrite(rele3, LOW);
}

void setup(){
  lcd. begin(16,4);
  lcd. setCursor(0,0);
  lcd. print("    La Banba    ");
  lcd. setCursor(0,1);
  lcd. print("((( 95.5 FM )))");

  pinMode(rele1, OUTPUT);
  pinMode(rele2, OUTPUT);
  pinMode(rele3, OUTPUT);

  Serial.begin(9600);
  digitalWrite(rele1, HIGH);
  digitalWrite(rele2, HIGH);
  digitalWrite(rele3, HIGH);

}
void loop(){
  if (Serial.available())
  {
    x = Serial.read();
   if (x == '1') {
       desativar_pinos();
       digitalWrite(rele1, HIGH);
       distancia = 0;

       lcd.clear();
       lcd.setCursor(0,0);
       lcd.print("Speed 40km/h");
       lcd.setCursor(0,1);
       lcd.print("time = ");
       lcd.print(0);
       lcd.print("s");
       lcd.setCursor(0,2);
       lcd.print("dist = ");
       lcd.print((int)round(distancia));
       lcd.print("m");

       delay(1000);

       for (int i = 1; i <= 45; i++){
          distancia += 11.11;

          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("Speed 40km/h");
          lcd.setCursor(0,1);
          lcd.print("time = ");
          lcd.print(i);
          lcd.print("s");
          lcd.setCursor(0,2);
          lcd.print("dist = ");
          lcd.print((int)round(distancia));
          lcd.print("m");
```

(2)

```
   lcd.print((int)round(distancia));
   lcd.print("m");

   delay(1000);
       }
 lcd.clear();
    }
    if (x == '2'){
       desativar_pinos();
       digitalWrite(rele2, HIGH);

       distancia = 0;

       lcd.clear();

       lcd.setCursor(0,0);
       lcd.print("Speed 60km/h");
       lcd.setCursor(0,1);
       lcd.print("time = ");
       lcd.print(0);
       lcd.print("s");
       lcd.setCursor(0,2);
       lcd.print("dist = ");
       lcd.print((int)round(distancia));
       lcd.print("m");

       delay(1000);

       for (int i = 1 ; i <= 30; i++){

          distancia += 16.66;

          lcd.clear();

          lcd.setCursor(0,0);
          lcd.print("Speed 60km/h");
          lcd.setCursor(0,1);
          lcd.print("time = ");
          lcd.print(i);
          lcd.print("s");
          lcd.setCursor(0,2);
          lcd.print("dist = ");
          lcd.print((int)round(distancia));
          lcd.print("m");

          delay(1000);
       }

       lcd.clear();
    }
 if (x == '3') {
       desativar_pinos();
       digitalWrite(rele1, HIGH);
       digitalWrite(rele2, HIGH);

       distancia = 0;

       lcd.clear();

       lcd.setCursor(0,0);
       lcd.print("Speed 80km/h");
       lcd.setCursor(0,1);
       lcd.print("time = ");
       lcd.print(0);
```

(3)

```
       lcd.print("s");
       lcd.setCursor(0,2);
       lcd.print("dist = ");
       lcd.print((int)round(distancia));
       lcd.print("m");

          delay(1000);
       }

       lcd.clear();
    }
    if (x == '4'){
       desativar_pinos();
       digitalWrite(rele3, HIGH);

       distancia = 0;

 lcd.clear();

       lcd.setCursor(0,0);
       lcd.print("Speed 110km/h");
       lcd.setCursor(0,1);
       lcd.print("time = ");
       lcd.print(0);
       lcd.print("s");
       lcd.setCursor(0,2);
       lcd.print("dist = ");
       lcd.print((int)round(distancia));
       lcd.print("m");

       delay(1000);

       for (int i = 1; i <= 82; i++){

          distancia += 30.55;

          lcd.clear();

          lcd.setCursor(0,0);
          lcd.print("Speed 80km/h");
          lcd.setCursor(0,1);
          lcd.print("time = ");
          lcd.print(i);
          lcd.print("s");
          lcd.setCursor(0,2);
          lcd.print("dist = ");
          lcd.print((int)round(distancia));
          lcd.print("m");

          delay(1000);
       }

       lcd.clear();

    }
    //Se x for igual a zero '0' vai desligar todos os
reles:
    if (x == '0'){
       desativar_pinos();
       digitalWrite(rele1, HIGH);
       digitalWrite(rele2, HIGH);
       digitalWrite(rele3, HIGH);

       lcd. setCursor(0,0);
       lcd. print("    La Banba    ");
       lcd. setCursor(0,1);
       lcd. print("((( 95.5 FM )))");

    }
  }
}
```

APÊNDICE B – PROGRAMA MATLAB - ARDUINO COM A FUNÇÃO GUIDE

Feito no MATLAB 2017b

```
function varargout = velocidade_var(varargin)
% VELOCIDADE_VAR MATLAB code for velocidade_var.fig
%      VELOCIDADE_VAR, by itself, creates a new VELOCIDADE_VAR or raises the existing
%      singleton*.
%      H = VELOCIDADE_VAR returns the handle to a new VELOCIDADE_VAR or the handle to
%      the existing singleton*.
%      VELOCIDADE_VAR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in VELOCIDADE_VAR.M with the given input arguments.
%
%      VELOCIDADE_VAR('Property','Value',...) creates a new VELOCIDADE_VAR or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before velocidade_var_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to velocidade_var_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help velocidade_var
% Last Modified by GUIDE v2.5 08-Jun-2022 08:28:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @velocidade_var_OpeningFcn, ...
                   'gui_OutputFcn',  @velocidade_var_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before velocidade_var is made visible.
function velocidade_var_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to velocidade_var (see VARARGIN)

% Choose default command line output for velocidade_var
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes velocidade_var wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = velocidade_var_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function botaoiniciar_Callback(hObject, eventdata, handles)
% hObject    handle to botaoiniciar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
global Porta

delete(instrfind(('Port'),('COM6'))); % Deleta objeto tipo Port, se existir, na serial COM6.
Porta = serial('COM6', 'BaudRate', 9600); %Cria o objeto Porta p/ comunicacao serial com Arduino.
fopen(Porta) %Abre a porta Serial para uso com Arduino.
pause(2)

% --- Executes on button press in botaoterminar.
function botaoterminar_Callback(hObject, eventdata, handles)
% hObject    handle to botaoterminar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Porta
%Fechando tudo:
fclose(Porta) %Fecha Porta.
delete(Porta) %Deleta a Porta.
clear Porta %Limpa a variavel Porta.
%close all %Fecha todas as figuras.

% --- Executes on button press in botao40.
function botao40_Callback(hObject, eventdata, handles)
% hObject    handle to botao40 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Porta

t = imread('C:\Users\Juan\Desktop\photos de placas de Limite de Velocidade\40-km(1).jpeg');
axes(handles.axes1);
imshow(t);

d = '1';
 fprintf(Porta, '%c',d);%Escreve na Porta o valor de d igual a '1': envia '1' para Arduino.
d = '0'; % Para desligar - Todos os reles de uma vez.
fprintf(Porta,'%c',d);

% --- Executes on button press in botao60.
function botao60_Callback(hObject, eventdata, handles)
% hObject    handle to botao60 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Porta
 t1 = imread('C:\Users\Juan\Desktop\photos de placas de Limite de Velocidade\60-km-code.jpg');
axes(handles.axes2);
imshow(t1);

 % Ligando e desligando o rele 2:
d = "2"; % Variavel tipo string, para ligar e rele 2 somente
 fprintf(Porta, '%c',d);%Escreve na Porta o valor de d igual a '2': envia '2' para Arduino.
d = '0'; % Para desligar - Todos os reles de uma vez.
fprintf(Porta,'%c',d);

% --- Executes on button press in botao80.
function botao80_Callback(hObject, eventdata, handles)
% hObject    handle to botao80 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Porta
t2 = imread('C:\Users\Juan\Desktop\photos de placas de Limite de Velocidade\80-km-code.jpg');
axes(handles.axes3);
imshow(t2);

d = "3"; % Variavel tipo string, para ligar e rele 1 e 2 somente
fprintf(Porta, '%c',d);%Escreve na Porta o valor de d igual a '3': envia '3' para Arduino.
d = '0'; % Para desligar - Todos os reles de uma vez.
fprintf(Porta,'%c',d);

% --- Executes on button press in botao110.
function botao110_Callback(hObject, eventdata, handles)
% hObject    handle to botao110 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Porta
 t3 = imread('C:\Users\Juan\Desktop\photos de placas de Limite de Velocidade\110-km(1).jpg');
axes(handles.axes4);
imshow(t3);
d = '4'; % Variavel tipo string, para ligar e rele 3 somente
fprintf(Porta, '%c',d);%Escreve na Porta o valor de d igual a '1': envia '1' para Arduino.
d = '0'; % Para desligar - Todos os reles de uma vez.
fprintf(Porta,'%c',d)
```

APÊNDICE C – PROGRAMA ARDUINO PARA LEITOR DE CODIGO DE BARRAS

(1)

```
#include <usbhid.h>
#include <usbhub.h>
#include <hiduniversal.h>
#include <hidboot.h>
#include <SPI.h>

class MyParser : public HIDReportParser {
  public:
    MyParser();
    void Parse(USBHID *hid, bool is_rpt_id, uint8_t len, uint8_t *buf);
  protected:
    uint8_t KeyToAscii(bool upper, uint8_t mod, uint8_t key);
    virtual void OnKeyScanned(bool upper, uint8_t mod, uint8_t key);
    virtual void OnScanFinished();
};

MyParser::MyParser() {}

void MyParser::Parse(USBHID *hid, bool is_rpt_id, uint8_t len, uint8_t *buf) {
  // If error or empty, return
  if (buf[2] == 1 || buf[2] == 0) return;

  for (uint8_t i = 7; i >= 2; i--) {
    // If empty, skip
    if (buf[i] == 0) continue;

    // If enter signal emitted, scan finished
    if (buf[i] == UHS_HID_BOOT_KEY_ENTER) {
      OnScanFinished();
    }

    // If not, continue normally
    else {
      // If bit position not in 2, it's uppercase words
      OnKeyScanned(i > 2, buf, buf[i]);
    }
    return;
  }
}
uint8_t MyParser::KeyToAscii(bool upper, uint8_t mod, uint8_t key) {
  // Letters
  if (VALUE_WITHIN(key, 0x04, 0x1d)) {
    if (upper) return (key - 4 + 'A');
    else return (key - 4 + 'a');
  }
  // Numbers
  else if (VALUE_WITHIN(key, 0x1e, 0x27)) {
    return ((key == UHS_HID_BOOT_KEY_ZERO) ? '0' : key - 0x1e + '1');
  }
  return 0;
}
void MyParser::OnKeyScanned(bool upper, uint8_t mod, uint8_t key) {
  uint8_t ascii = KeyToAscii(upper, mod, key);
  Serial.print((char)ascii);
}
```

(2)

```
void MyParser::OnScanFinished() {
  Serial.println();
}

USB          Usb;
USBHub       Hub(&Usb);
HIDUniversal Hid(&Usb);
MyParser     Parser;

void setup() {
  Serial.begin( 115200 );
  Serial.println("Start");

  if (Usb.Init() == -1) {
    Serial.println("OSC did not start.");
  }

  delay( 200 );

  Hid.SetReportParser(0, &Parser);
}

void loop() {
  Usb.Task();
}
```

APÊNDICE D – PROJETO NO LABORATÓRIO

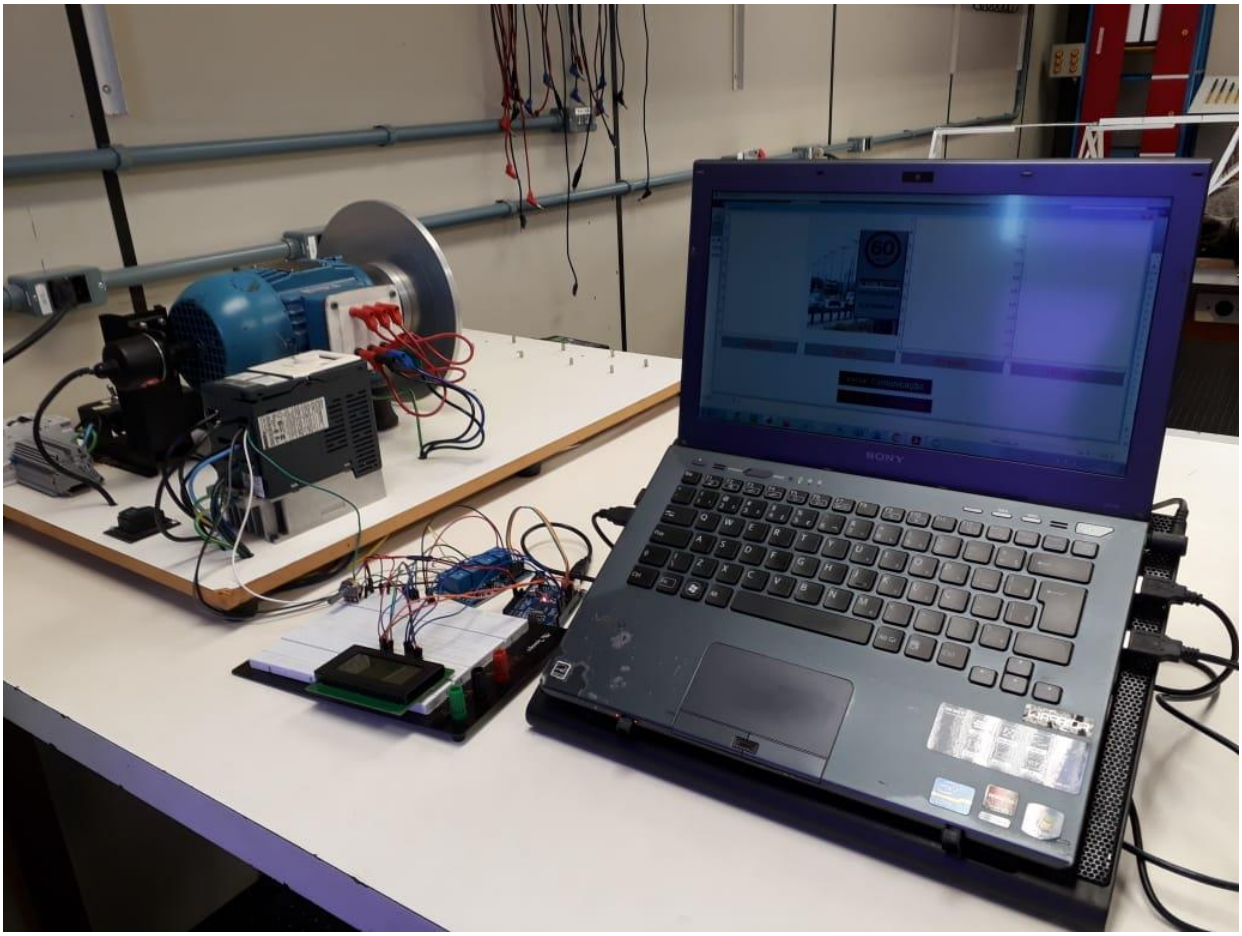

Fig. D.1. Bancada de experimentos
Fonte: Angela de Araújo, Juan Morales.