TESTES AUTOMATIZADOS EM SISTEMAS DE INFOENTRETENIMENTO UTILIZANDO ROBÔ CARTESIANO

Andrei José Orben[1]

Resumo: Este artigo descreve a idealização e implementação de um robô cartesiano, que busca mitigar erros em testes automatizados de software que envolvem toques em telas touch de sistemas de infoentretenimento automotivo. Testes realizados somente por software possuem limitações que acabam gerando resultados inválidos e irrelevantes para execuções em larga escala. Para tal demanda, foi desenvolvido um robô cartesiano capaz de interagir fisicamente com a tela através de um sistema de controle que o interliga com as rotinas de teste automatizado em software, visando proporcionar maior confiabilidade e robustez nos testes com toque.

Palavras-Chave: Robô cartesiano. Testes automatizados. Touchscreen. Infoentretenimento Automotivo.

APPLICATION OF CARTESIAN ROBOT FOR PHYSICAL TEST AUTOMATION IN AUTOMOTIVE INFOTAINMENT SYSTEMS

Abstract: This article describes the conception and implementation of a Cartesian robot that seeks to mitigate errors in automated software testing that involves touch screens in automotive infotainment systems. Tests performed only by software have limitations that end up generating invalid and irrelevant results for large-scale executions. To meet this demand, a Cartesian robot capable of physically interacting with the screen through a control system that interconnects it with automated software testing routines was developed, in order to provide greater reliability and robustness in tests with touch.

Keywords: Cartesian robot. Automated tests. Touchscreen. Automotive infotainment.

1. INTRODUÇÃO

Na última década houve um rápido aumento no uso de telas sensíveis ao toque (touch) em veículos, particularmente para funções de informação e entretenimento. E por sua vez, a tela touch se tornou uma grande atração para a indústria automobilística. Como resultado, hoje a maioria dos fabricantes de automóveis as utilizam em vez de botões estáticos (PATEL et al., 2022).

Com o aumento da demanda de qualidade nos produtos que contém telas touch, seja a nível físico ou de integração com o software, ferramentas como a automação de testes surgem como uma boa opção para esta finalidade. E no que diz respeito a testes automatizados: “Para se colocar um software em produção com garantia da qualidade é necessário a realização de testes para encontrar o máximo de erros possível e corrigi-los antes que possam causar uma má experiência para o usuário final.” (SANTOS, 2019). De forma complementar, Santos (2019) ainda descreve que testes de software, quando realizados manualmente, podem ser custosos e estão muito sujeitos a erros por distração humana, por frequentemente se tratar de

1 Acadêmico do curso Bacharel em Engenharia Elétrica do Instituto Federal de Santa Catarina. andrei.j07@aluno.ifsc.edu.br.

tarefas longas e repetitivas. E para este contexto em específico, ainda é viável citar que: “A automatização de testes garante maior consistência e confiabilidade uma vez que a rotina de testes, se bem implementada, é executada de forma aderente ao planejado.”(Santos, 2019).

Ao olhar para o que as empresas estão fazendo sobre este tema, nota-se a relevância deste tipo de projeto, por exemplo ao observar o conceito de Robotic Test Automation (RTA) apresentado pela Cognizant (2023) – que quer dizer testes automatizados com robôs – e se baseia na ideia de que o mundo digital e físico estão diretamente interligados no cotidiano das pessoas, e que na soma destes mundos na qual eles denominam “phygital” – physical + digital, ou físico mais digital – os testes, em outras palavras, necessitam ser mais robustos mesmo nas ações mais simples como o toque na tela; e assim apresenta uma solução com robôs colaborativos utilizados em ciclos de testes automatizados que permitem entregas de qualidade e de forma ágil para as empresas que os utilizarem, que de certa forma também serve de inspiração para este trabalho.

Desta forma os objetivos principais contidos nesse artigo incluem não somente aumentar a confiabilidade nos resultados de testes automatizados envolvendo telas sensíveis ao toque, mas também oferecer uma alternativa a mais aos casos de teste manual ou onde a emulação de toques virtuais é impossível ou inviável por limitações técnicas, através da integração de um robô cartesiano ao ciclo de teste. Para tal, foi realizada a implementação de uma estrutura mecânica do robô, responsável pelas suas ações físicas propriamente ditas, na qual tem seus movimentos gerenciados por um sistema de controle baseado na junção de uma Raspberry Pi com um Arduino, que por sua vez é comandado por um software baseado em Python, desenvolvido pelo autor, tendo então como objeto alvo uma tela touch cujo conteúdo (software) presente na tela e a lógica de retorno da execução do touch foram também desenvolvidas pelo autor.

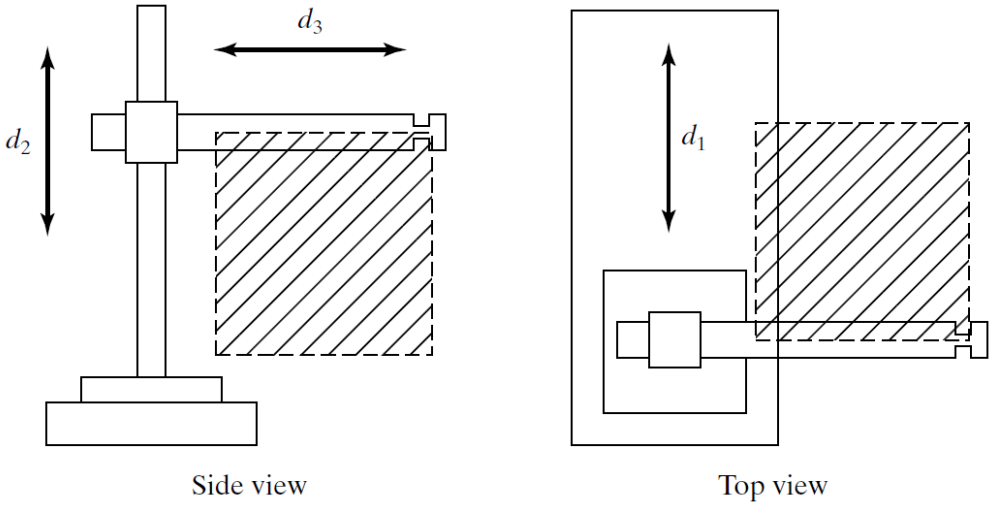
2. DESENVOLVIMENTO

Este projeto consiste em alguns elementos que se complementam num sistema único e dividem-se entre a estrutura física, a parte de software e o sistema de controle que os interliga, por fim levando a tela sensível ao toque, que é o objeto alvo do atuador do robô. E para melhor compreender o que foi desenvolvido, torna-se relevante revisar alguns conceitos fundamentais relacionados ao projeto.

2.1. Robôs cartesianos

Um robô cartesiano tem uma das configurações mais simples dentre as estruturas mecânicas de movimentação. Como mostrado na Figura 1, as juntas 1 a 3 são prismáticas (que permitem deslocamentos em linha reta), mutuamente ortogonais e correspondem às direções cartesianas X, Y e Z. Esta configuração produz robôs com estruturas muito rígidas. Outras vantagens dos manipuladores cartesianos decorrem do fato de que as três primeiras juntas são desacopladas. Isso os torna mais simples de projetar. (CRAIG, 2014).

Figura 1 – Representação da estrutura de um robô cartesiano



Fonte: Craig (2014)

2.2. G-code e GRBL

O código G é uma linguagem de programação originalmente usada para controlar máquinas CNC (Controladoras Numéricas Computadorizadas) e consiste em uma série de comandos que instruem a máquina sobre movimentos, velocidades, profundidades de corte e outras operações necessárias para executar um design específico (RPTECHNOLOGIES, 2024). Da mesma maneira que tais comandos são úteis nas máquinas CNC, estes também servem como ferramenta para o controle de posicionamento do robô cartesiano.

Alguns comandos de maior interesse a este trabalho, exibidos na Tabela 1, foram obtidos através de tabelas como a disponibilizada por (Haas Automation, 2024) e experimentações do autor.

Tabela 1 – Exemplos de Código G

Comando	**Descrição**	**Exemplo**
X, Y, Z	Eixo de movimento + número do tamanho do passo em mm	G0 X8 Y8 Z8
F	Taxa de alimentação em mm/min	G01 X10 Y10 Z10 F1500
G0	Posicionamento de Movimento Rápido	G0 X10 Y20 Z30
G01	Movimento de Interpolação Linear (mais preciso e controlados)	G01 X10 Y10 Z10 F1500
G10	Define offsets	G10 P0 L20 X0 Y0 Z0
G20	Selecionar polegadas	-
G21	Selecionar métrico (mm)	-
G28	Regressar ao Ponto Zero da Máquina	-
G29	Regressar do Ponto de Referência	-
G90	Posicionamento absoluto (coordenadas relativas ao zero)	-
G91	Posicionamento relativo (coordenadas relativas ao ponto atual)	-

Fonte: O autor (2024).

“GRBL é um software gratuito, de código aberto e de alto desempenho para controlar o movimento de máquinas”(JEON, 2024). Conforme descrito por seus próprios criadores, este software disponibilizado na plataforma GitHub é capaz de controlar uma série de máquinas, incluindo as do tipo CNC, que de certa forma possuem o mesmo tipo de movimentação desejado ao robô cartesiano. O GRBL é

capaz de interpretar o código G, é de fácil utilização e eficiente.

2.3. Controladores

O Raspberry Pi é um computador de placa única (single-board computer) do tamanho de um cartão de crédito, projetado para ser acessível, versátil e de baixo custo. Ela foi desenvolvida pela Raspberry Pi Foundation, com o objetivo de promover o ensino da ciência da computação e de eletrônica de forma acessível ao redor do mundo e é amplamente usada por entusiastas de tecnologia, estudantes e profissionais em uma ampla gama de aplicações, como automação residencial, robótica, Internet das Coisas (IoT) e outros. (GUSE, 2024). Dentre suas características principais, as mais relevantes a este projeto são o seu tamanho compacto, a possibilidade de conexão remota, a disponibilidade de portas USB, e a presença de conectores GPIO auxiliares; e sua principal função é centralizar as conexões do sistema de controle e os códigos em software, de forma que toda a estrutura necessária para o funcionamento do robô se concentre num só lugar e possa ser acessada remotamente, aumentando a produtividade, organização e comodidade ao usuário.

Segundo o site oficial (Arduino, 2018), Arduino é uma plataforma eletrônica de código aberto baseada em hardware e software de fácil utilização, capaz de receber instruções para um microcontrolador na placa através da linguagem de programação Arduino e a IDE Arduino Software. Sua natureza simples e acessível o torna uma ótima ferramenta para construção de protótipos de baixo custo, dentre outras aplicações.

Entretanto, com base em Quadros (2021), na maioria dos casos, a interação da placa Arduino com “o mundo externo” é feita conectando-se elementos através dos conectores de expansão, ou seja, uso de fios e matriz de contatos; o que eventualmente resulta em um resultado frágil em termo de conexões. E a partir disso surgem os shields (escudo em inglês). Os shields são placas que se encaixam ao Arduino para acrescentar funcionalidades de uma forma simples e confiável. (QUADROS, 2021). Dentre os vários tipos de shields existentes, encontra-se o CNC Shield V3, que segundo MakerHero (2024) reúne em um só módulo toda a eletrônica necessária para controlar os motores de passo e demais componentes do projeto. Dentre suas características principais estão a entrada para alimentação externa, pinos para controle de sensores e soquetes para 4 drivers.

E no que diz respeito aos drivers aqui se destaca o modelo A4988 que foi especialmente desenvolvido para controle de motores de passo bipolares, com capacidade de pequenos passos (microstepping) para maior suavidade e precisão na movimentação dos motores. (MAKERHERO, 2024). Algumas outras características descritas por MakerHero (2024) e também visíveis em documentações técnicas do fabricante (datasheets) são: a tensão de operação lógica de 3 V a 5,5 V; a capacidade de controlar motores de até 35 V e 2 A por bobina e microstepping com precisão de até 1/16 passos.

2.4. Touch screen

As telas touch são atualmente um dos dispositivos de interações homem-máquina mais comuns, sendo os dois principais tipos o touch screen resistivo e capacitivo. (HU, 2024).

Telas touch resistivas consistem em um painel de vidro ou acrílico que é revestido com camadas eletricamente condutoras e resistivas feitas com óxido de índio e estanho (ITO) cuja as camadas finas são separadas por espaçadores. As

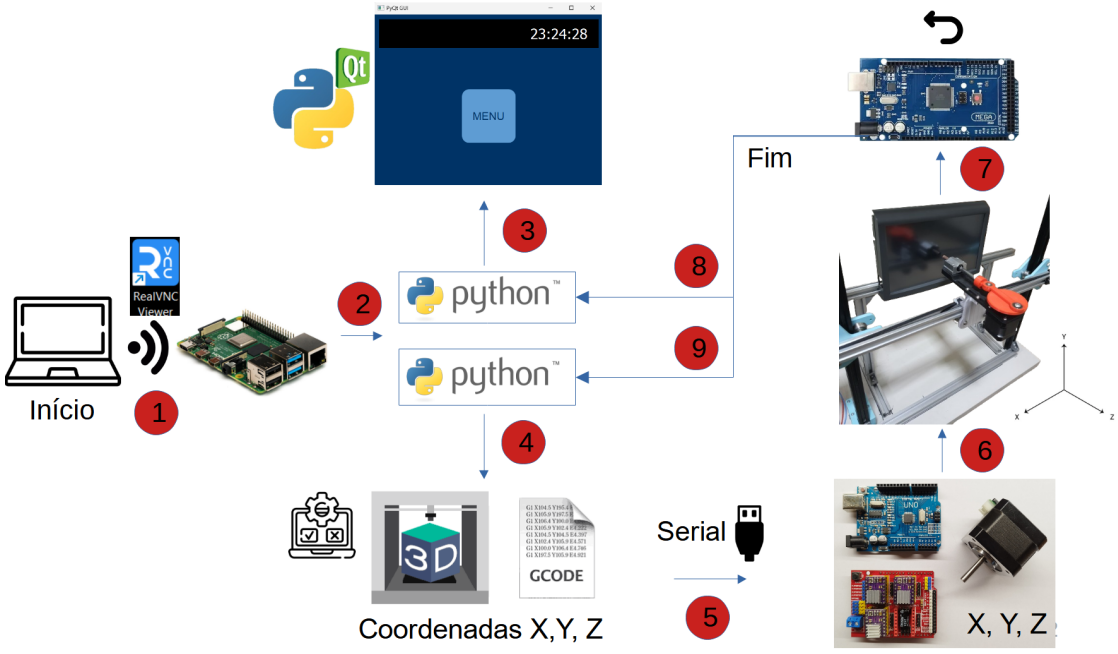
telas resistivas são geralmente o tipo mais acessível de tela touch e têm boa durabilidade, o que as tornam bastante utilizadas em aplicações de alto uso, apesar de sua menor precisão em relação a outros tipos de tela touch. As duas arquiteturas resistivas mais populares usam configurações de 4 fios ou 5 fios. Os circuitos resistivos determinam a localização do toque em duas dimensões de pares de coordenadas. (DOWNS, 2005). Hu (2024) ainda destaca que a tela resistiva pode responder ao toque de qualquer objeto, o que se torna sua maior vantagem; mas também que o ITO é muito fino e com o tempo de uso pode apresentar rachaduras que resultam numa lenta e progressiva perda da capacidade de touch, até chegar na completa falha da tela.

Já as telas touch capacitivas se resumem a distribuição de um campo elétrico gerado por uma corrente induzida num eletrodo, abaixo de uma camada condutora. Enquanto não há toque, essa distribuição é uniforme. Do contrário, a variação do campo (diretamente ligada a capacitância total gerada pelo contato com o corpo humano ou caneta touch), é medida e o posicionamento do toque identificado. (HU, 2024). São muito mais sensíveis ao toque e precisas, porém possuem custo mais elevado.

3. METODOLOGIA

Como já mencionado o robô cartesiano é o resultado da integração de alguns elementos (Figura 2). E seu desenvolvimento se dá por uma sequência de etapas que de forma geral se descreve como: (1) conexão remota entre o computador do usuário com a Raspberry Pi, onde nela são executados dois scripts Python (2). Um destes é referente a GUI exibida na tela (3) e outro se relaciona ao teste automatizado e controle do robô em si (4). Isto se dá por comandos enviados por comunicação serial (5) que chegam num controlador Arduino, que efetua o movimento do robô nos eixos através dos motores de passo (6). As interações com a tela são capturadas por outro Arduino e enviadas para os scripts de GUI e de teste automatizado (8) e (9). Nesta seção, tais elementos e etapas serão detalhados.

Figura 2 – Visão geral do projeto



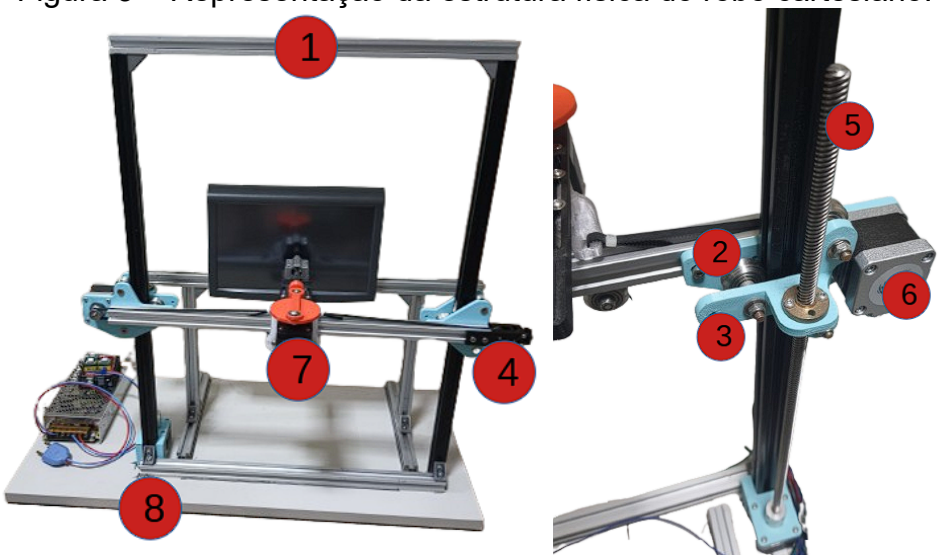
Fonte: O autor (2024)

3.1. Estrutura física

Esta é composta por partes fixas e móveis, como: perfis de alumínio do tipo v-slot (1), roldanas de metal (2) e suportes para roldanas (3), parafusos, porcas e perfis em L (cantoneiras), correia e tensor (4), fuso e castanhas (5), motores de passo (6), atuador (7) e base de madeira (8), conforme a Figura 3.

Os perfis de alumínio com v-slot foram selecionados por sua disponibilidade e por serem os mais adequados para as roldanas, também já disponíveis para uso no projeto. Para que os eixos possam efetuar movimentações verticais e horizontais, peças específicas são necessárias; e em virtude da acessibilidade ao recurso de impressão 3D, tais peças personalizadas foram implementadas para estes fins, servindo como suporte para as roldanas e se posicionando em torno dos perfis de alumínio. Uma outra extremidade destas peças se conecta ao meio de movimentação de cada eixo. Por exemplo, para o eixo X a movimentação ocorre através de uma correia com tensor, que é então fixada na peça de base do atuador do eixo Z que é peça principal onde ocorre o movimento de toque através de um mecanismo de ação direta ligado ao seu motor de passo (Figura 4). Enquanto no eixo Y o processo é feito com um fuso cuja uma “castanha” promove a fixação e conversão do movimento radial do motor de passo, em uma trajetória linear do respectivo eixo.

Figura 3 – Representação da estrutura física do robô cartesiano.



Fonte: O autor (2024)

Figura 4 – Representação mecânica dos eixos X e Z.

Fonte: O autor (2024)

3.2. Touchscreen e GUI

Fixada a um suporte ajustável posicionado logo a frente do atuador, como que configurando uma estrutura única sobre a base de madeira, encontra-se a tela sensível ao toque (Figura 5). Trata-se de um touch screen resistivo a 4 fios de dez polegadas com entrada VGA, que será o objeto principal de interação do robô.

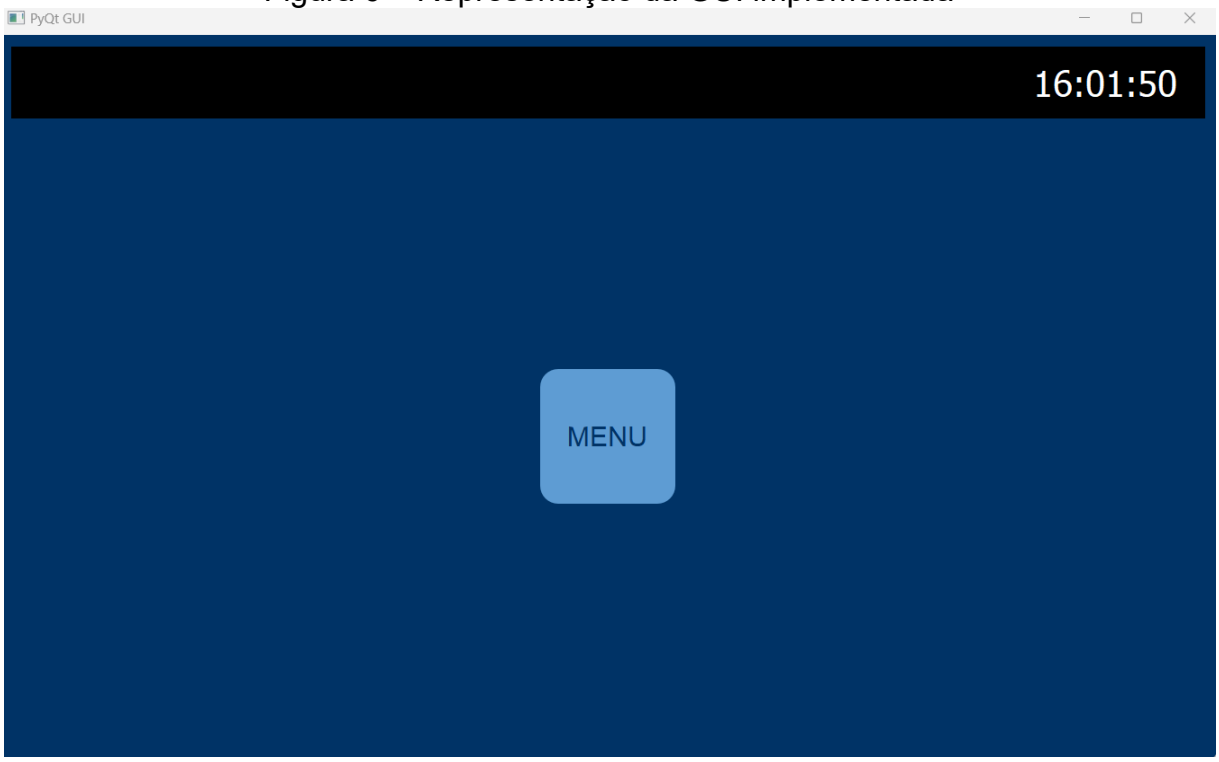
Figura 5 – Tela touch utilizada.

Fonte: O autor (2024).

A escolha de tal modelo se deu devido a sua disponibilidade e simplicidade de uso. Há duas conexões importantes relacionadas à tela: a conexão de vídeo feita com um microcomputador Raspberry Pi 3, via adaptador VGA-HDMI, para exibição de uma GUI (Figura 6) desenvolvida em Python utilizando a biblioteca PyQt5, pelo autor, como amostra de teste; e a conexão com um microcontrolador Arduino MEGA para se obter as coordenadas de toque na tela e assim ser possível a retroalimentação na lógica de testes automatizados e de atualização do conteúdo da

interface gráfica.

Figura 6 – Representação da GUI implementada



Fonte: O autor (2024).

3.3. Software de testes automatizados

Uma parte fundamental deste projeto é sua aplicação, pois não se trata apenas do robô cartesiano, mas sim onde ele será utilizado. Para tal, foi implementada uma rotina de testes na linguagem Python a partir dos comandos em código G e comunicação serial. Dentre as principais funções efetuadas pelo código, destacam-se: a configuração e uso das portas seriais, o comando para escrever uma mensagem a ser enviada via serial e o comando para ler as respostas. Estes dois últimos foram agrupados em uma única função chamada 'send_gcode(command)' (Apêndice A – linha 19), que significa 'enviar código G' e recebe um comando como parâmetro de entrada; sendo posteriormente utilizada dentro de outra função criada – perform_touch(x, y, z) (Apêndice A – linha 25) – que significa 'efetuar toque' e recebe as coordenadas desejadas como parâmetro de entrada. Ou seja, através deste programa é possível inicializar uma rotina automatizada de teste, composta também por comandos que envolvem o touch na tela, que antes seriam feitos por emulação e agora fazem uso do toque físico realizado pelo atuador do robô. Este código especificamente realiza um cenário de teste criado, onde são realizados toques sequenciais num mesmo ponto da tela. Os parâmetros de quantidade de toques na tela (Apêndice A – linha 68), e as coordenadas de movimentação dos três eixos (Apêndice A – linha 70) podem ser redefinidos no código mediante novos cenários de teste criados. Todos os códigos desenvolvidos estão disponíveis em um repositório do autor no GitHub (nebroierdna/robotic-test-automation).

3.4. Sistema de controle do robô cartesiano

Para interligar os comandos gerados via software ao posicionamento e movimentação efetivamente aplicados fisicamente, foi desenvolvido um sistema de controle de baixa complexidade, baseado no uso de um Raspberry Pi 3 B V1.2, um Arduino Uno e um CNC shield V3 com três drivers A4988 regulados para os motores de passo de 1 A. O Arduino foi gravado e configurado com a biblioteca GRBL, sendo esta utilizada aqui como uma ferramenta já disponível, sem a necessidade de qualquer intervenção no código original. Desta forma, os comandos em código G provenientes do código Python presente na Raspberry e recebidos pela porta serial do Arduino, são então traduzidos para passos/mm e enviados para os motores de passo através dos drivers presentes na placa do CNC shield V3, cuja alimentação provém de uma fonte DC de 12 V. Ressalta-se a importância de garantir a tensão de alimentação especificada na placa do CNC shield, bem como a regulagem do V_{REF} dos drivers, para evitar mal funcionamentos em relação a atuação dos motores de passo, conforme observados em testes durante a implementação.

4. RESULTADOS E DISCUSSÃO

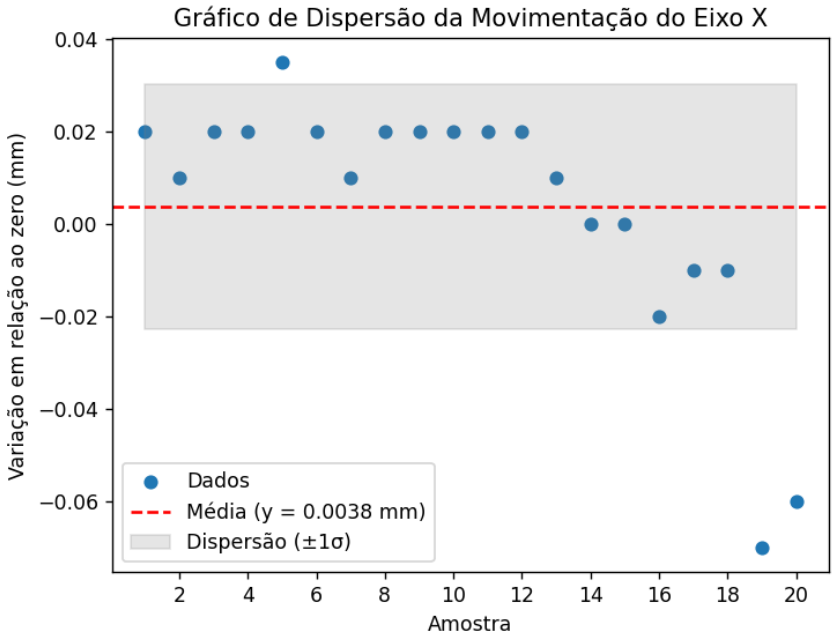
Para analisar o desempenho do robô cartesiano foram considerados ao menos dois cenários: teste de repetibilidade de movimentação, onde são observadas as variações de movimento em relação a uma referência, a medida que o robô realiza movimentos nos eixos X e Y; e teste de repetibilidade de toque, onde são efetuados múltiplos toques na tela num mesmo ponto.

4.1. Análise de repetibilidade de movimentação

Para este teste, foi realizado um ensaio com auxílio de um relógio comparador, que consiste em definir uma posição inicial arbitrária, zerar o instrumento de medição (referência) e fazer o robô se deslocar por uma determinada distância num mesmo eixo e retornar à posição inicial. A cada movimentação, o relógio fornece um valor que demonstra o quanto o robô variou em seu posicionamento em relação à referência. Este processo foi realizado para os eixos X e Y, onde 20 amostras foram coletadas de cada um.

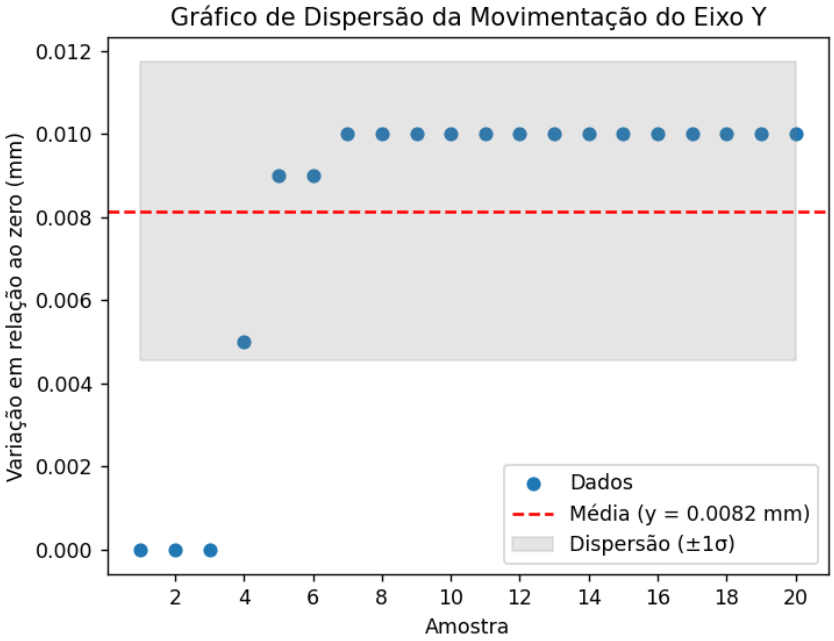
Para analisar o resultado, foram gerados gráficos de dispersão para cada eixo (Figura 7 e Figura 8). Inicialmente, destaca-se que os valores de erro máximo, ou seja, os “piores casos” dentre as amostras, foram consideravelmente baixos, sendo 0,07 mm para o eixo X e 0,01 mm para o eixo Y. Ainda assim, nota-se que para ambos os casos, os valores apresentaram constância ao longo dos testes, o que somado ao fato anterior, indica que para esta aplicação, os erros de movimentação são ínfimos e demonstram que o robô é capaz de realizar as movimentações necessárias sem comprometer os testes automatizados. Porém ressalta-se que a interpretação de tais resultados pode variar mediante a aplicação, pois para certos sistemas mais exigentes ou sensíveis os valores citados podem ser considerados não aceitáveis, o que não é o caso desta aplicação.

Figura 7 – Gráfico de dispersão (eixo X)



Fonte: O autor (2024).

Figura 8 – Gráfico de dispersão (eixo Y)



Fonte: O autor (2024).

4.2. Análise de repetibilidade de toque

Para esta validação foi utilizado o código em Python presente no Apêndice A, com 3 diferentes amostras: 10, 20 e 50 touchs consecutivos, cujos resultados são apresentados na Tabela 2. O foco deste teste é monitorar a consistência e assertividade da tela resistiva, a medida que o robô cartesiano realiza toques de forma consecutiva e em maior quantidade.

Tabela 2 – Resultados de testes de repetibilidade

Qtd. Alvo de Touchs	Tempo de execução	Precisão (tela)
10	00:00:35	100%
20	00:01:05	80%
50	00:02:30	86%

Fonte: O autor (2024).

Os destaques positivos se dão pela realização de 80 toques na tela em cerca de 4 minutos de forma automatizada, o que em comparação com testes manuais, e extrapolando para números maiores e cenários mais abrangentes de desenvolvimento de produtos, poderia se tornar repetitivo, cansativo e mais passível de falhas caso feito por um ser humano. Ressalta-se também que a quantidade de amostras não está limitada aos números apresentados, mas pode ser ajustada para testes de longa duração; o que foi considerado como não necessário para demonstração do desempenho neste quesito.

Outro ponto relevante é o fato de o robô ter sido capaz de identificar durante os testes, falhas no touch da tela utilizada (precisão inferior a 100%), sendo este um dos principais objetivos dos testes automatizados, no que diz respeito a qualidade do produto, reforçando a relevância dos resultados atingidos pelo robô cartesiano nessa aplicação.

5. CONSIDERAÇÕES FINAIS

Neste artigo foi apresentado o desenvolvimento de um robô cartesiano para aplicação em testes automatizados de telas touch (infoentretenimento), baseado em uma estrutura versátil, controlada remotamente através de elementos como a Raspberry Pi e o Arduino, e comandada por software com rotinas de código em Python. O que permitiu revisar e analisar conceitos de robótica, teste e validação de produtos, controle e programação, conhecer e utilizar ferramentas úteis (GRBL e G-code) bem como a avaliação do desempenho do robô em termos de aplicabilidade para teste, tempo economizado com automação e precisão, em dois cenários de repetibilidade (movimentação e toque). Os resultados apontaram que o robô cartesiano é capaz de se movimentar de forma a não afetar os testes que executa e ainda encontrar defeitos na tela dentro de seus ciclos de execução de toques que pode variar conforme a necessidade do usuário.

Os estudos e implementações apresentados servem como base para melhorias futuras, que podem incluir mas não somente: aprimoramento dos materiais utilizados e métodos de construção física para resolver problemas de amostras inválidas de toque; refinamento dos códigos de programação para otimizar a captura dos dados na porta serial e mitigar possíveis ruídos nas análises e uso das coordenadas touch; análise e implementação da compactação do protótipo no que poderia se tornar uma versão evoluída e mais próxima de uma versão comercial deste projeto, como solução para testes de infoentretenimento em larga escala.

Por fim entende-se que os objetivos principais deste projeto foram atingidos, e há margens factíveis para melhorias, que podem trazer ainda mais relevância para projetos futuros neste seguimento, que possui fortes tendências de crescimento no mercado, principalmente o automotivo.

REFERÊNCIAS

SANTOS, Luciano Vargas dos et al. **Robô para Automação de Testes em Máquinas de Cartão de Crédito.** In: 14º SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, 2019, Ouro Preto, doi: 10.17648/sbai-2019-111244.

COGNIZANT. **Robotic Test Automation | Cognizant**. 2023. 2 min. Disponível em: https://www.youtube.com/watch?v=Hqi1gqZaRdQ&ab_channel=Cognizant. Acesso em: 29 abr. 2024.

PATEL, S., Liu, Y., Zhao, R., Liu, X., Li, Y. (2022). **Inspection of In-Vehicle Touchscreen Infotainment Display for Different Screen Locations, Menu Types, and Positions**. In: Krömker, H. (eds) HCI in Mobility, Transport, and Automotive Systems. HCII 2022. Lecture Notes in Computer Science, vol 13335. Springer, Cham. https://doi.org/10.1007/978-3-031-04987-3_18

RPTECHNOLOGIES. **G-Codes in CNC Machining**. 2024. Disponível em: https://rptechnologies.co.uk/khub/g-code-in-cnc-machining/. Acesso em: 15 nov. 2024.

JEON, Sungeun K.. **Github Wiki grbl**. 2024. Disponível em: https://github.com/grbl/grbl/wikihttps://github.com/grbl/grbl/wiki. Acesso em: 15 nov. 2024.

HAAS AUTOMATION. **Lathe – G-Codes.** 2024. Disponível em: https://www.haascnc.com/pt/service/service-content/guide-procedures/lathe---g-codes.html. Acesso em: 21 out. 2024.

CRAIG, John J.. **Introduction to Robotics: Mechanics and Control**. 3. ed.: Pearson Education Limited, 2014. 373 p.

GUSE, Rosana. **O que é Raspberry Pi?** 2024. Disponível em: https://www.makerhero.com/guia/raspberry-pi/o-que-e-raspberry-pi/. Acesso em: 15 nov. 2024.

ARDUINO. **What is Arduino?** 2018. Disponível em: https://www.arduino.cc/en/Guide/Introduction. Acesso em: 03 nov. 2024.

QUADROS, Daniel. **O que são shields para Arduino?** 2021. Disponível em: https://www.makerhero.com/blog/o-que-sao-shields-para-arduino/#comments. Acesso em: 03 nov. 2024.

MAKERHERO. **CNC Shield V3 para Arduino.** 2024. Disponível em: https://www.makerhero.com/produto/cnc-shield-v3-para-arduino-impressora-3d/?form=MG0AV3. Acesso em: 03 nov. 2024.

MAKERHERO. **Driver Motor de Passo A4988.** 2024. Disponível em: https://www.makerhero.com/produto/driver-motor-de-passo-a4988/. Acesso em: 03 nov. 2024.

DOWNS, Rick. **Using resistive touch screens for human/machine interface**. Analog Applications Journal Q, v. 3, p. 5-10, 2005.

HU, Ruijie. (2024). **The advantages and comparison of resistive touch screens and capacitive touch screens.** Applied and Computational Engineering. 30. 94-103. 10.54254/2755-2721/30/20230080.

APÊNDICE A – Código de teste automatizado em Python

```
import serial
import time

# Configurações da porta serial
port_grbl = 'COM4'  # Substitua pela porta correta para GRBL
port_arduino = 'COM9'  # Porta para o Arduino ligado à tela touch
baudrate_grbl = 115200  # Taxa de baud padrão para GRBL
baudrate_arduino = 9600  # Taxa de baud padrão para Arduino

# Inicializa a conexão serial
ser_grbl = serial.Serial(port_grbl, baudrate_grbl)
ser_arduino = serial.Serial(port_arduino, baudrate_arduino)
time.sleep(2)  # Aguarda a inicialização da conexão

# Variáveis para cálculo de precisão
total_touches = 0
registered_touches = 0

def send_gcode(command):
    ser_grbl.write(f'{command}\n'.encode())
    response = ser_grbl.readline().decode().strip()
    print("Resposta da máquina:", response)
    return response

def perform_touch(x, y, z):
    send_gcode(f"G1 X{x} F2000")
    send_gcode(f"G1 Y{y} F2000")
    send_gcode(f"G1 Z{z} F1500")

# Função para ler as coordenadas do toque do Arduino
def read_touch_coordinates(discard_first=False):
    global registered_touches
    if ser_arduino.in_waiting:
        line = ser_arduino.readline().decode('utf-8').strip()
        if "X = " in line and "\tY = " in line:
            try:
                x_str = line.split("X = ")[1].split("\t")[0]
                y_str = line.split("\tY = ")[1]
                x, y = int(x_str), int(y_str)
                print(f"Coordenadas do toque: X = {x}, Y = {y}")

                if not discard_first:
                    registered_touches += 1

                return x, y
            except ValueError:
                print(f"Valor inválido recebido: {line}")
    return None
```

```
49
50      # Função para mostrar a precisão dos toques
51      def show_accuracy():
52          accuracy = (registered_touches / total_touches) * 100 if total_touches > 0
else 0
53          print(f"Precisão de toque: {accuracy:.2f}%")
54
55      # PRECONDITION
56      # Set zero
57      send_gcode("G10 P0 L20 X0 Y0 Z0")
58
59      # Define as unidades de medida para mm e pos. relativa a posição anterior
60      send_gcode("G21 G91")
61      time.sleep(2)
62
63      # Desconsiderar a primeira leitura para descartar qualquer ruído
64      print("Descartando primeira leitura...")
65      read_touch_coordinates(discard_first=True)
66
67      # ACTION
68      number_of_touch_executions = 50
69      for touch in range(number_of_touch_executions):
70          perform_touch(0, 0, 8)
71          total_touches += 1
72          time.sleep(3)
73          coordinates = read_touch_coordinates()
74          if coordinates:
75              x, y = coordinates
76              print(f"Toque registrado nas coordenadas: X = {x}, Y = {y}")
77          else:
78              print("Nenhum toque registrado.")
79
80      # POSTCONDITION
81      # Mover de volta pro zero
82      send_gcode("G90 G1 X0 Y0 F1000")
83
84      # Exibir a precisão dos toques
85      show_accuracy()
86
87      # Fecha a conexão serial
88      ser_grbl.close()
89      ser_arduino.close()
```

ANDREI JOSÉ ORBEN

TESTES AUTOMATIZADOS EM SISTEMAS DE INFOENTRETENIMENTO UTILIZANDO ROBÔ CARTESIANO

Este trabalho foi julgado adequado para obtenção do título de Engenheiro Eletricista, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

IFSC Joinville, 19, Dezembro de 2024.

Prof. Michael Klug, Dr.
Orientador
Instituto Federal de Santa Catarina

Prof. Jeferson Luiz Curzel, Dr.
Instituto Federal de Santa Catarina

Prof. Janderson Duarte, Dr.
Instituto Federal de Santa Catarina