REVISTA BRASILEIRA DE MECATRÔNICA
FACULDADE SENAI DE TECNOLOGIA MECATRÔNICA

RETROFITTING DO HARDWARE DE CONTROLE DE ROBÔ MANIPULADOR PARA FINS DIDÁTICOS

RETROFITTING OF MANIPULATING ROBOT CONTROL HARDWARE FOR EDUCATIONAL PURPOSES

Eduardo Henrique Gomes[1], [i]
Daniel Barbuto Rossato[2], [ii]
André Luis dos Santos[3], [iii]
Paulo Andre dos Santos[4],[iv]

Data de submissão: (22/11/2022) Data de aprovação: (26/04/2023)

RESUMO

O projeto tem como objetivo apresentar uma proposta de melhoria no circuito eletrônico de controle de um robô manipulador intitulado *Future Education Robot Open System* (FEROS), que vem sendo desenvolvido desde 2009 por diferentes grupos de alunos. Atualmente, algumas tecnologias tornaram-se defasadas ou obsoletas. No presente trabalho serão apontadas soluções a fim de atualizar as tecnologias empregadas no sistema e possibilitar o aumento da quantidade de servomotores controlados para um braço mecânico com seis graus de liberdade. Dessa forma será ampliada a gama de aplicações onde o FEROS pode ser utilizado. Para isso, será detalhado o projeto de desenvolvimento de um novo sistema de controle e comunicação dos servomotores com maior capacidade de processamento de dados. Assim, busca-se garantir um melhor desempenho com tecnologias atualizadas mantendo o cerne do projeto.

Palavras-chave: Robótica, Sistema Didático, Raspberry, Servodrive

ABSTRACT

The project aims to present a proposal to improve the electronic control circuit of a manipulator robot called Future Education Robot Open System (FEROS), which has been developed since 2009 by different groups of students. Currently, some technologies have become outdated or obsolete. In the present work, solutions will be pointed out in order to update the technologies used in the system and make it possible to increase the number of

[1] Pós-graduado em Automação e Controle na Faculdade SENAI São Paulo. E-mail: eduardo.gomes@sp.senai.br
[2] Docente na Faculdade SENAI São Paulo – Campus Mariano Ferraz. E-mail: daniel.rossato@sp.senai.br
[3] Docente na Faculdade SENAI São Paulo – Campus Mariano Ferraz. E-mail: andre.lsantos@sp.senai.br
[4] Docente na Faculdade SENAI São Paulo – Campus Mariano Ferraz. E-mail: paulo.andre@sp.senai.br

controlled servomotors for a mechanical arm with six degrees of freedom. In this way, the range of applications where FEROS can be used will be expanded. For this purpose, the development project of a new control and communication system for servomotors with greater data processing capacity will be detailed. Thus, we seek to ensure better performance with updated technologies while maintaining the project core.

Keywords: Robotics, Didactic Systems, Raspberry, Servodrive

1 INTRODUÇÃO

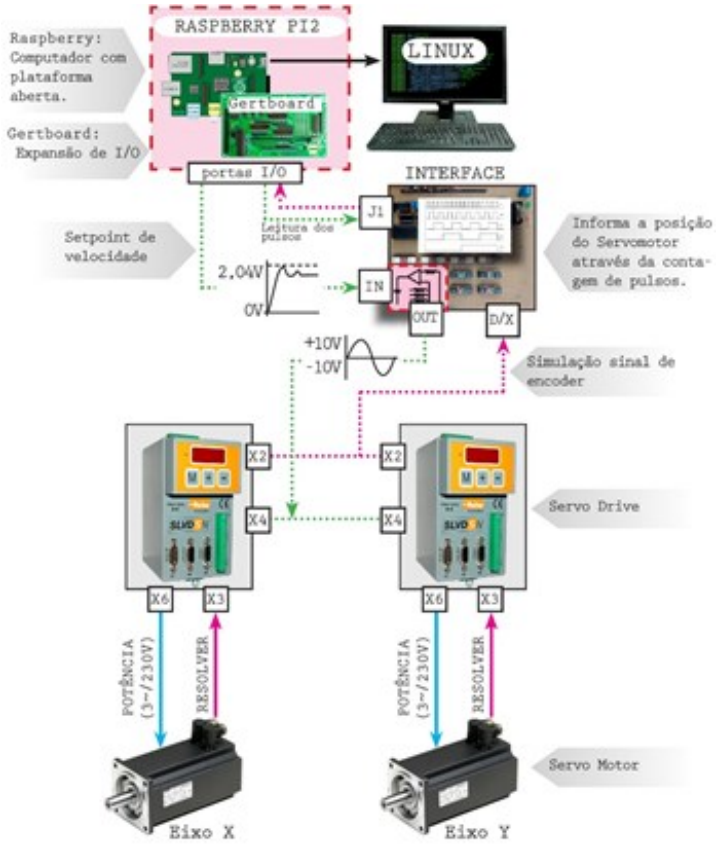
A robótica é uma ciência que abrange tecnologias de diversas áreas da engenharia. O projeto de um robô manipulador agrega conhecimentos de mecânica, ciência dos materiais, física, computação, eletroeletrônica e controle, por exemplo. Isso o torna um objeto de estudo propício para o ensino e pesquisa dessas áreas em instituições formadoras de profissionais da indústria.

Os fabricantes limitam o acesso às informações, que como pode-se notar englobam aspectos muito variados. Com base nessa condição limitante no que tange o ensino de robótica, foi proposto por Rossato (2009), o desenvolvimento de um sistema aberto para ensino de robôs manipuladores. O nome adotado no projeto foi o acrônimo FEROS, que se origina de *Future Education Robot Open System.*

Rossato (2009) aborda diferentes aspectos justificando a aplicação das tecnologias que poderiam ser implementadas viabilizando o projeto. MELO et al. (2011) iniciaram a elaboração de uma interface eletrônica para controle do robô manipulador por meio do computador utilizando um sistema operacional de tempo real. SANTOS et al. (2012) realizaram testes de viabilidade para utilização de uma interface USB comercial com computador rodando Linux em tempo real. Manso e Barbosa (2016) desenvolveram o hardware atual, adaptando o sistema de controle do robô para *Raspberry Pi* 2. Até então, o projeto contava com a possibilidade de controle de dois servomotores.

O presente trabalho de *retrofitting* parte deste ponto, visando atualizar o circuito eletrônico de controle implementado em 2016, representado na figura 1.

Figura 1 –Placa eletrônica atual



Fonte: MANSO; BARBOSA (2016)

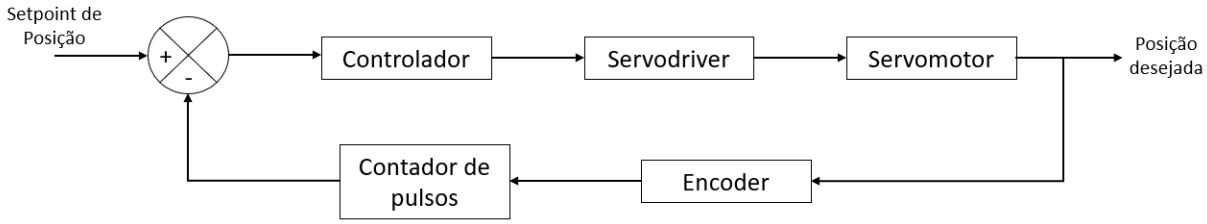
Foram propostas modificações em pontos críticos do projeto, por exemplo, atualização do computador (*Raspberry Pi*) utilizado, aumento da velocidade de comunicação e possibilidade de controle de até seis servomotores.

As melhorias visam obter um controle de posicionamento mais preciso em uma estrutura mecânica. Neste sentido, as placas desenvolvidas propõem-se a alcançar uma contagem de pulsos dos *encoders* com valor mais próximo ao real e gerar uma tensão saída, usada no controle de movimentação dos servomotores, com maior resolução. Além disso, será incluído o terceiro servomotor para o movimento de rotação da base. O novo sistema também possibilita que futuramente sejam implementados mais três servomotores para a movimentação de uma ferramenta.

2 ATUALIZAÇÕES DO HARDWARE DE CONTROLE

No projeto FEROS, o sistema de controle de posicionamento do manipulador é um sistema de malha fechada. O controlador, um computador, recebe a posição atual dos servomotores, que é determinada de acordo com a contagem de pulsos, e a compara com a posição desejada. Após a realização dos cálculos do compensador, de acordo com a função transferência do braço robótico, um sinal analógico que varia de -10 V a +10 V é enviado ao *servodrive* para rotacionar o servomotor na velocidade e direção ideais a fim de alcançar a posição desejada no braço robótico. Esse diagrama de blocos está representado na figura 2.

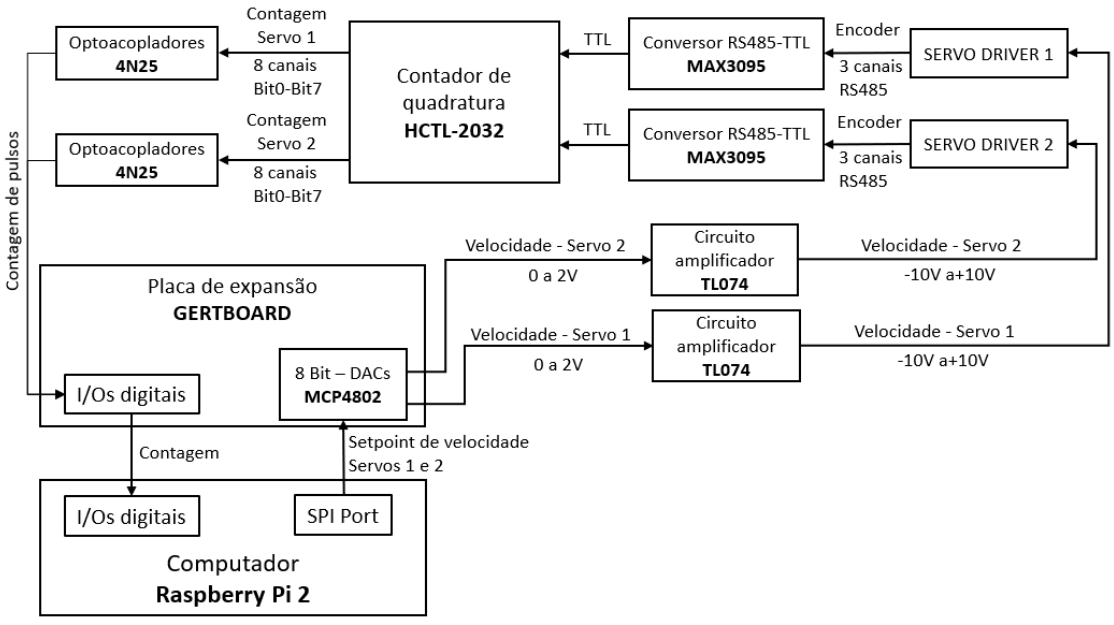
Figura 2 – Diagrama de blocos do sistema de controle dos servomotores



Fonte: Elaborado pelo autor

O diagrama mostrado na figura 3 representa a arquitetura anterior ao início deste projeto:

Figura 3 – Arquitetura antiga do projeto



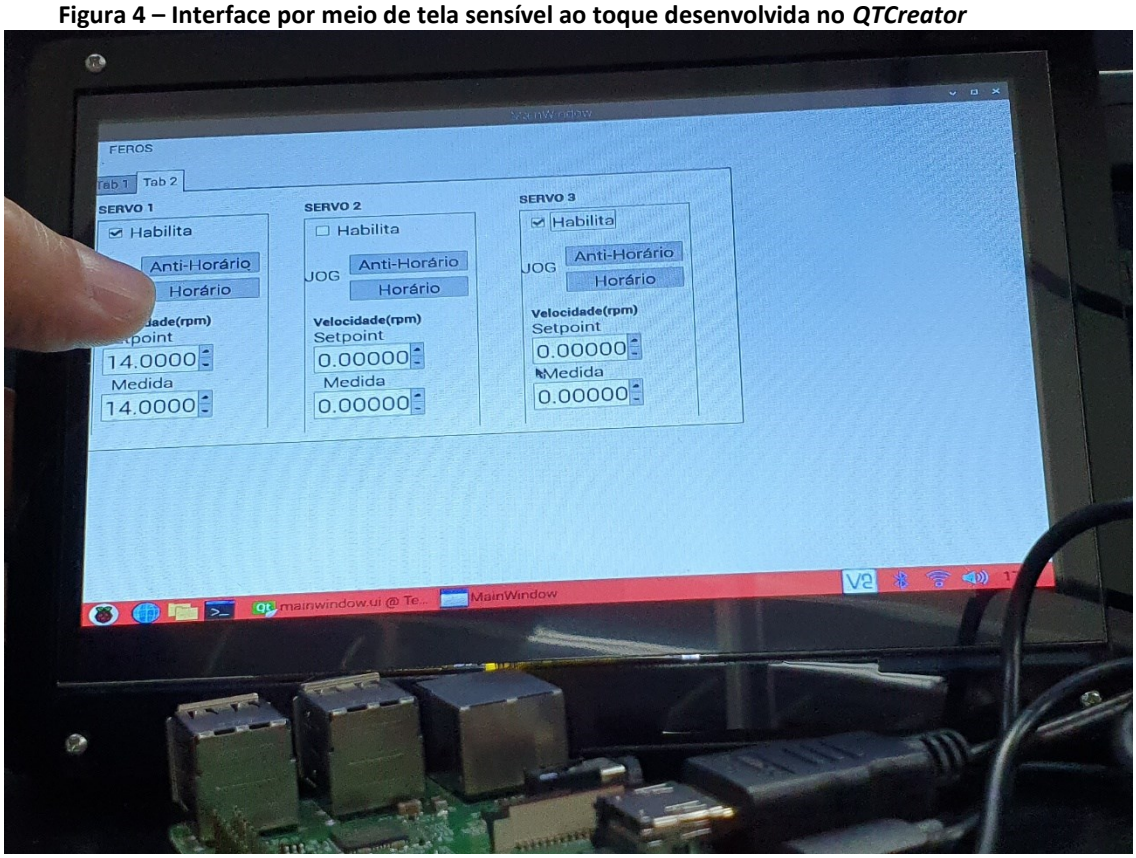
Fonte: Elaborado pelo autor

O computador de placa única *Raspberry Pi* 2 (que possui processador *quad-core* ARM *Cortex* A7 rodando a 900 MHz e 1 GB de memória RAM) era utilizado como controlador. Ele era responsável pelo processamento dos pulsos advindos do *servodrive*, através de uma

interface que realizava o tratamento dos sinais. O computador foi substituído, pois já se encontra no mercado o *Raspberry Pi* 4, que conta com maior poder de processamento (processador *quad-core* ARM Cortex-A72 rodando a 1,5GHz e opções com 2GB, 4GB e 8GB de memória RAM).

Conforme é mostrado na figura 4, a interface de usuário dar-se-á por meio de tela sensível ao toque, que se conecta ao *Raspberry Pi* 4 por meio de porta USB, para uso da função de toque, e porta HDMI, para transmissão de imagem. O software foi desenvolvido na plataforma *QTCreator*, que permite a criação de interface gráfica utilizando linguagem similar a *Wiring* (que por sua vez, baseia-se em C++). Isso possibilitará que o docente realize alterações no sistema com os alunos de maneira mais fácil, já que eles possuem maior familiaridade com a linguagem, vista em outras unidades curriculares.

Figura 4 – Interface por meio de tela sensível ao toque desenvolvida no *QTCreator*



Fonte: Elaborado pelo autor

O projeto anterior contava com uma *Gertboard* como interface entre o *Raspberry Pi* e os *servodrives*. Ela é uma placa de expansão de entradas e saídas para o *Raspberry Pi*. Nela há uma variedade de componentes como doze portas de entrada ou saída, botões, LEDs, conversores digital-analógico de dois canais, conversor analógico-digital de dois canais e um microcontrolador Atmel AVR. (FEN LOGIC LTD., 2012).

Este trabalho visa expandir a capacidade de servomotores controlados de dois para seis. Cada servomotor necessita de um sinal analógico de referência para controle de velocidade. Como a *Gertboard* possui apenas dois canais e se conecta diretamente aos 40 pinos do *Raspberry Pi* de uma vez, impossibilita a conexão de mais de uma placa de expansão.

Além disso, as portas de entrada e saída digitais suportam um nível de tensão de no máximo 7V. Os *drives* de controle dos servomotores operam com sinais digitais de 24V. Logo,

o projeto antigo contava com um circuito que convertia, por meio de opto acopladores 4N25, o nível dos sinais de 24V para 3,3V.

Tendo em vista as características da *Gertboard* citadas anteriormente, constatou-se a necessidade de projetar circuitos capazes de realizar a interface dos sinais digitais de controle entre *Raspberry Pi* e *servodrives*, geração do sinal analógico de controle dos *servodrives* e contagem dos pulsos advindos do *encoder*.

A fim de facilitar o entendimento dos alunos que utilizarão o sistema, optou-se por dividir o circuito em três placas diferentes. Isso possibilita a análise de cada etapa do funcionamento do circuito de controle separadamente e permite que cada placa seja programada em momentos e até mesmo unidades curriculares diferentes. Por exemplo, a placa de geração de contagem de pulsos pode ser utilizada em programação de microcontroladores para explicar o funcionamento e configuração das interrupções externas e a placa de geração de sinal analógico pode ser utilizada em eletrônica analógica para exemplificar uma aplicação de amplificadores operacionais. O funcionamento de cada uma das placas será detalhado nos próximos tópicos.

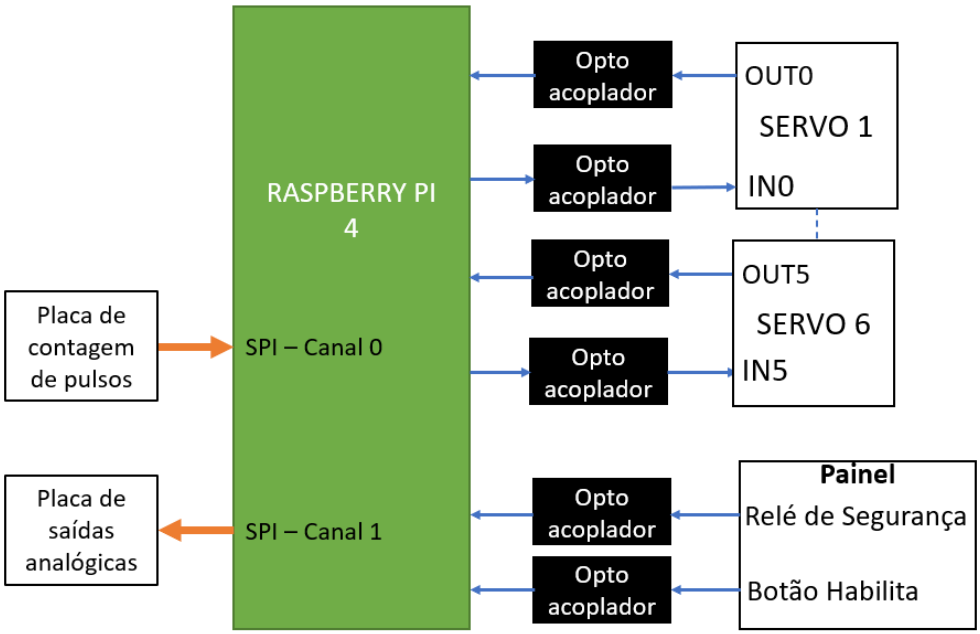
3 PLACA DE INTERFACE DIGITAL

Na versão anterior, eram utilizados opto acopladores 4N25, circuito integrado (CI) de 6 pinos, para realizar a conversão de nível de tensão e proteção entre as portas digitais do *Raspberry Pi* (3,3V) e as portas digitais do *servodrive* (24V). Na versão atual, foram utilizados opto acopladores PC817, que são menores que os 4N25 e possuem velocidade de resposta e níveis de corrente e tensão compatíveis. Dessa forma, foi possível diminuir consideravelmente as dimensões da placa.

É importante destacar que cada *servodrive* conta com uma entrada para habilitar seu funcionamento e uma saída para informar possível falha do mesmo. Portanto, a placa conta com um par de opto acopladores para cada servomotor. Além disso, ainda há as portas do *Raspberry Pi* que receberão os sinais 24V do relé de segurança e do botão habilitador do painel, totalizando quatorze CIs PC817. A velocidade de comunicação entre os servomotores e o computador é um ponto crítico do projeto. O posicionamento e a velocidade do motor são calculados com base nas contagens de pulsos dos *encoders*, que é feita pela placa dedicada a isso. Considerando os possíveis protocolos de comunicação serial disponível no *Raspberry Pi* 4: UART, I2C e SPI, optou-se por utilizar este último, já que possui velocidade de transmissão de dados superior aos demais.

Conforme as necessidades de projeto citadas e os pinos disponíveis no *Raspberry Pi*, a placa de interface digital foi projetada de acordo com o diagrama funcional mostrado na figura 5. Já a alocação de pinos está detalhada na tabela 1.

Figura 5 – Diagrama funcional da placa de interface digital



Fonte: Elaborado pelo autor

Tabela 1 – Alocação de pinos do *Raspberry Pi*

Pino	GPIO	Circuito Eletrônico
1	3,3V	
3	GPIO2	
5	GPIO3	Falha Servo 6
7	GPIO4	Botão Habilita o Painel
9	GND	
11	GPIO17	SPI1(CS1) - Placa de contagem de pulsos
13	GPIO27	Relé de Segurança
15	GPIO22	Habilita Servo 1
17	3,3V	
19	GPIO10	SPI0(MOSI) - Placa de saídas analógicas
21	GPIO9	SPI0(MISO) - Placa de saídas analógicas
23	GPIO11	SPI0(SCLK) - Placa de saídas analógicas
25	GND	
27	ID_SD	
29	GPIO5	Habilita Servo 2
31	GPIO6	Habilita Servo 3
33	GPIO13	Falha Servo 1
35	GPIO19	SPI1(MISO) - Placa de contagem de pulsos
37	GPIO26	Habilita Servo 4
39	GND	

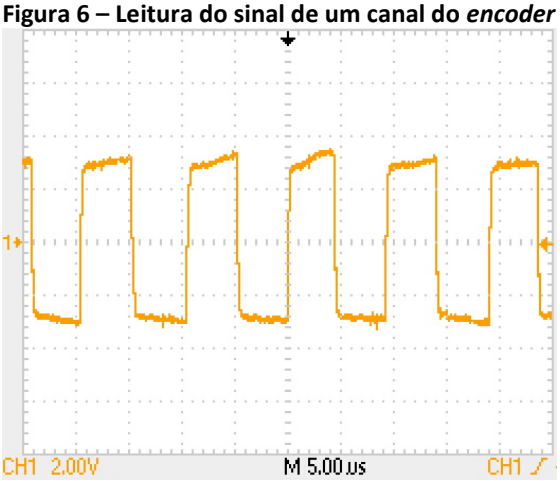
Pino	GPIO	Circuito Eletrônico
2	5V	
4	5V	
6	GND	
8	GPIO14	Habilita Servo 5
10	GPIO15	Habilita Servo 6
12	GPIO18	SPI1(CS0) - Placa de contagem de pulsos
14	GND	
16	GPIO23	Falha Servo 2
18	GPIO24	Falha Servo 3
20	GND	
22	GPIO25	Falha Servo 4
24	GPIO8	SPI0(CS0) - Placa de saídas analógicas
26	GPIO7	SPI0(CS1) - Placa de saídas analógicas
28	ID_SC	
30	GND	
32	GPIO12	Falha Servo 5
34	GND	
36	GPIO16	SPI1(CS2) - Placa de contagem de pulsos
38	GPIO20	SPI1(MOSI) - Placa de contagem de pulsos
40	GPIO21	SPI1(SCLK) - Placa de contagem de pulsos

Fonte: Elaborado pelo autor

4 PLACA DE CONTAGEM DE PULSOS

De acordo com PARKER (2008), o *servodrive* (modelo SLVD5N) fornece, através do conector X2, um sinal de *encoder* simulado, A transmissão dos pulsos é uma tensão diferencial no padrão RS-485. É possível configurá-lo para que a saída apresente de 4 a 65000 pulsos por volta. Os testes realizados com osciloscópio mostrados na figura 6, quando a velocidade máxima do motor, 6000rpm (rotações por minuto), é atingida, a frequência dos pulsos do *encoder* estabiliza em torno de 100KHz e a tensão de pico-a-pico (Vpp) é de aproximadamente 6,5V.

Figura 6 – Leitura do sinal de um canal do *encoder* realizado com osciloscópio



Fonte: Elaborado pelo autor

O padrão RS485 fornece na saída uma tensão diferencial entre os dois cabos, positivo e negativo, utilizados para a transmissão de dados. Devido a essa característica, consegue transmitir dados de maneira confiável mesmo sob condições severas de deterioração de sinal ao longo dos cabos e conectores. Essa robustez é a principal razão pela qual o padrão RS485 é adequado para transmissão a longas distâncias em ambientes com ruídos elétricos. (KUGELSTADT, 2021).

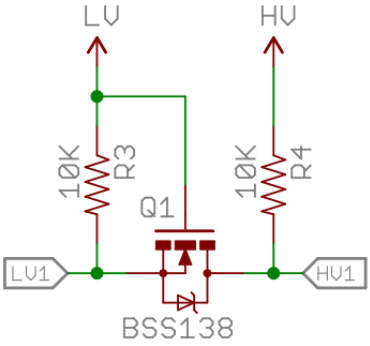
Portanto, para que o contador de quadratura que era utilizado, HCTL-2032, pudesse realizar a contagem dos pulsos, os sinais dos *drives* eram convertidos do padrão RS485 para TTL através do CI MAX3095 (MAXIM INTEGRATED PRODUCTS, 2019). O contador anteriormente empregado é um circuito integrado difícil de ser encontrado no mercado. Outra característica indesejada é que a contagem de cada servomotor era transmitida ao computador por meio de uma interface paralela de 8 canais, ocupando muitos pinos do *Raspberry Pi*.

Na placa atual, o sinal no formato RS485 advindo do *encoder* do servomotor continua sendo convertido para TTL através do CI MAX3095, que possui taxa máxima de transmissão de dados estimada pelo fabricante de 10Mbps (bits por segundo). Porém, o encapsulamento escolhido foi o SMD(componente de montagem em superfície) ao invés de PTH (pino através de furo) anteriormente empregado. Dessa forma houve uma melhor otimização do espaço da placa.

Como o nível de tensão de saída do MAX3095 é 5V, é necessário convertê-lo para 3,3V, que é a tensão suportada pelas portas da ESP32. Para esta tarefa é utilizado um conversor de nível lógico. Conforme observa-se no circuito da figura 7, o sinal de 5V é recebido pelo pino

HV1 e o sinal de nível baixo, 3,3V é enviado a partir do pino LV1. É necessário alimentar o circuito com 5V e 3,3V, nos pinos HV e LV respectivamente.

Figura 7 – Circuito do conversor de nível de tensão



Fonte: RIK, 2016

O MOSFET utilizado, BSS138, possui tempo de atraso na ligação e no desligamento de 20ns, conforme visto na tabela 3.

Tabela 3 – Características de chaveamento do BSS138

Tempo de atraso na ligação (*Turn-On Delay Time*)	td(on)	20ns
Tempo de atraso no desligamento (*Turn-Off Delay Time*)	td(off)	20ns

Fonte: Diodes Incorporated, 2021

Utilizando os dados da tabela 3 na equação 1, é possível estimar a frequência máxima de chaveamento do transistor, 25MHz. Como a frequência do *encoder* é de apenas 100KHz, este transistor pode ser utilizado no circuito de conversão de tensão.

$$fmax = \frac{1}{t_{don}+t_{doff}} = \frac{1}{20.10^{-9}+20.10^{-9}} = \frac{1}{4.10^{-9}} = 25MHz \quad (1)$$

Após testes com outras plataformas, a placa de desenvolvimento ESP32 mostrou-se adequada para substituir o contador HCTL-2032. Além de poder ser facilmente programada no IDE do Arduino, que é muito difundido e utilizado em escolas, ela possui poder de processamento (microprocessador dual-core de 240MHz) e periféricos (como duas portas SPI acessíveis) que possibilitaram seu uso no projeto. Além disso, possui um periférico de hardware para contagem de quadratura. Por possuir dois núcleos, optou-se por utilizar cada um deles para realizar a contagem de um servomotor.

Segundo PARKER (2008), o *encoder* absoluto pode ser configurado para que uma volta completa do motor seja equivalente a 4096 pulsos, a contagem de 0 a 4095 é mostrada no display do *servodrive*. Então, foi criado um programa (figura 8) que permite realizar a contagem de pulsos utilizando o periférico de hardware de contagem de quadratura da ESP32 utilizando a biblioteca ESP32Encoder. De acordo com HARRINGTON (2022), a biblioteca utiliza interrupções apenas quando o buffer de 16 bits do contador estoura e o periférico de hardware nos permite controlar até oito *encoders*.

Figura 8 – Trecho do programa "Contador de quadratura"

```
#include <ESP32Encoder.h>//Inclui a biblioteca de contagem de quadratura

ESP32Encoder encoder;//Cria um objeto da classe ESP32Encoder

//Intervalo de tempo entre exibições da contagem no monitor serial
#define TIME_INTERVAL 100
//Instante, em ms, que a ultima contagem foi mostrada
uint32_t displayInterval = 0;

void setup() {
  Serial.begin(115200);
  //Configura os pinos 34 e 35 da ESP32 para efetuar a contagem de quadratura
  encoder.attachFullQuad(34, 35);
  //Zera a contagem do encoder
  encoder.clearCount();
   //Botão auxiliar usado para zerar a contagem de pulsos
  pinMode(13, INPUT);
}

void loop() {
  //Mostra a contagem de pulsos dentro do intervalo determinado
  if ((millis() - displayInterval) > TIME_INTERVAL) {
    //Obtém a contagem de pulsos e a mostra no monitor serial
    Serial.println(encoder.getCount());
   displayInterval = millis();
  }
  if (digitalRead(13)) {//Lê o estado do bot do pino 13
    encoder.setCount(0);//Zera a contagem do encoder
  }
}
```

Fonte: Elaborado pelo autor

Como os valores de contagem mostrados no visor do *servodrive* são absolutos, é necessário converter a quantidade de pulsos medida para posição absoluta do eixo do motor. Sendo assim, é necessário utilizar a equação 2 para calcular a quantidade de voltas do eixo e então a equação 3 para estimar a posição atual do motor.

$$Numero\ de\ Voltas = \frac{Contagem\ da\ ESP32}{4096} \quad (2)$$

$$Posição\ calculada = [(n°\ de\ voltas) - (n°\ de\ voltas\ truncado)] \cdot 4096 \quad (3)$$

Foram realizadas diversas contagens e de acordo com os resultados obtidos na tabela 4, nota-se que o circuito contador de pulsos é capaz de acompanhar com exatidão a contagem dos pulsos do *servodrive.*

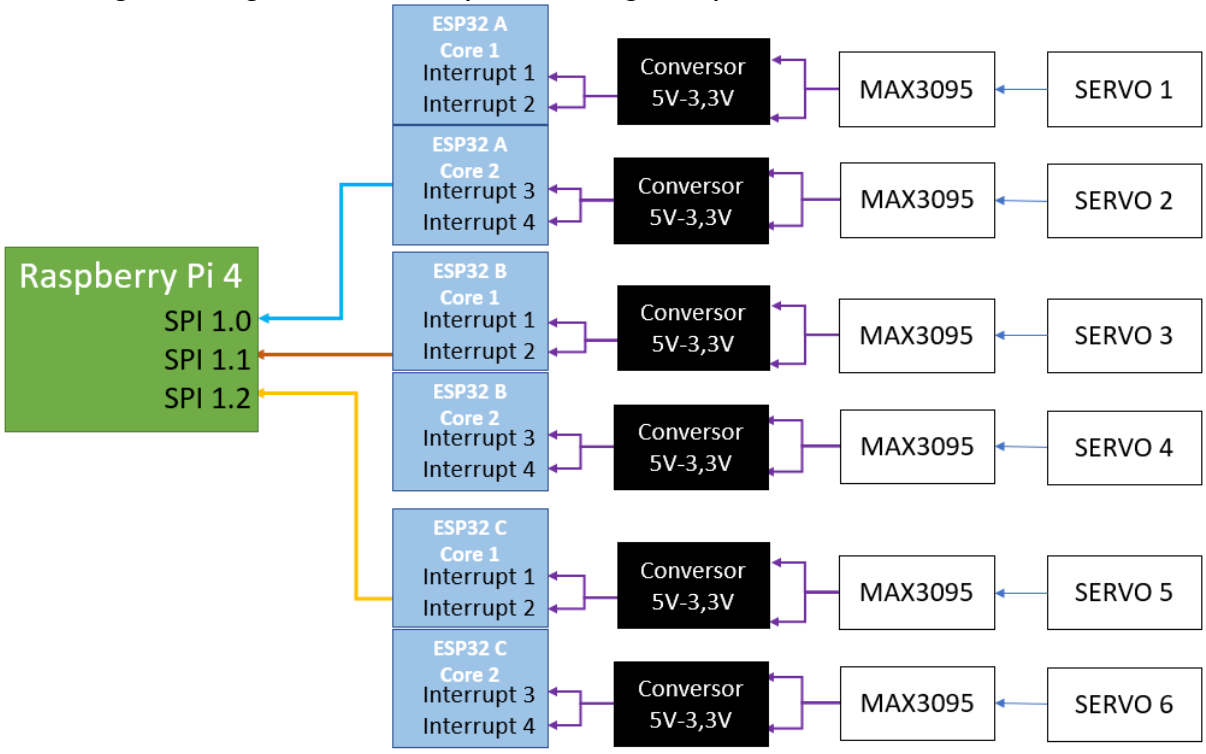
Tabela 4 – Comparação entre a contagem da placa e do *servodrive*

Velocidade (rpm)	Contagem (ESP32)	Posição calculada	Posição real do motor
1000	7645	3549	3549
1000	12563	275	275
1000	29635	963	963
1000	308523	1323	1323
1000	456321	1665	1665
2000	9854	1662	1662
2000	25634	1058	1058
2000	265320	3176	3176
2000	323659	75	75
2000	360548	100	100
3000	5654	1558	1558
3000	96545	2337	2337
3000	125897	3017	3017
3000	263214	1070	1070
3000	311954	658	658

Fonte: Elaborado pelo autor

O projeto visa possibilitar o controle de um manipulador com seis graus de liberdade, então a placa projetada conta com a possibilidade de contar os pulsos de seis servo motores. O resultado dessa arquitetura é representado pela figura 9.

Figura 9 – Diagrama funcional da placa de contagem de pulsos



Fonte: Elaborado pelo autor

5 PLACA DE SAÍDAS ANALÓGICAS

A faixa de tensão para o controle dos servomotores é de -10V a +10V. O conversor digital analógico (DAC) presente na *Gertboard*, placa de expansão para *Raspberry Pi* 2, é o MCP4802. Ele possui resolução de apenas 8 bits e somente dois canais, possibilitando o controle de apenas dois servos. Na nova arquitetura, é utilizado o DAC MCP4921, este conversor possui 12 bits de resolução e assim como o MCP4802 utiliza protocolo SPI para comunicação. A placa foi projetada para ser montada com até seis conversores, e assim controlar seis servomotores. Como o *Raspberry Pi* 4 não possuiria portas SPI suficientes, duas ESP32 realizam o recebimento dos dados advindos do *Raspberry Pi* 4 e os redirecionam para os conversores, também via SPI.

Os trechos de programa da figura 10 são responsáveis pela operação de recebimento e envio de dados via SPI. O programa completo está disponível em um repositório na nuvem no seguinte endereço: https://drive.google.com/drive/folders/1T7tlfSO2z9c0ILYHtYE6I3sh08D1Z9li?usp=sharing.

Figura 10 – Trecho do programa de envio e recebimento de dados via SPI

```
//Realiza o recebimento de dados pelo VSPI (SPI2)
void slaveSPI(){
  uint8_t byte1; uint8_t byte2;

  //Verifica se existem dados a serem lidos na porta SPI
  if(slave.getInputStream()->length() && digitalRead(_SS)==HIGH)
  {
    //Recebimento do primeiro byte
    byte1 = slave.getInputStream()->getBuffer()[1];
    //Recebimento do segundo byte
    byte2 = slave.getInputStream()->getBuffer()[0];
    //Chama a função masterSPI para enviar os dados recebidos para o DAC
    masterSPI(byte1, byte2);
  }
}

//Realiza o envio de dados pelo HSPI(SPI3)
void masterSPI(byte byte1, byte byte2){
  //Garante que o primeiro nibble seja 0011 (Datasheet DAC-MCP4921)
  byte sendByte = byte1 | 0x30;
  byte sendByte2 = byte2;
  //Inicia a comunicação via SPI
  hspi->beginTransaction(SPISettings(10000000, MSBFIRST, SPI_MODE0));
  //Envia nível baixo pelo pino ChipSelect do DAC, habilitando
  //seu recebimento de dados
  digitalWrite(MASTER_SPI_SS_DAC, LOW);
  //Finaliza a comunicação via SPI por parte da ESP32
  hspi->endTransaction();
}
```

Fonte: Elaborado pelo autor

Após a conversão do sinal digital em analógico, um amplificador operacional de instrumentação (AOP), INA126, é utilizado para obter a tensão na faixa de -10V a +10V. Um dos requisitos do projeto é que o AOP, mantivesse nível de tensão zero quando fosse

necessário a parada do servomotor. Em testes realizados, o INA126 mostrou-se adequado por manter um nível de tensão de saída constante e estável, conforme visto no teste da figura 11.

Figura 11 – Teste da saída do DAC em 0V

Fonte: Elaborado pelo autor

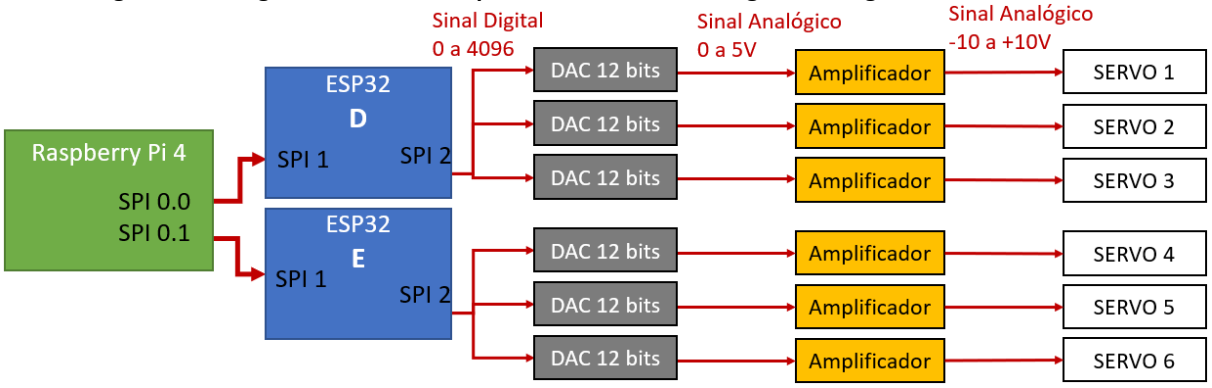
O circuito adotado atualmente fornece uma resolução de 4,88mV, conforme demonstrado na equação 4. Anteriormente, a resolução era de 78,43mV, como pode ser visto na equação 5.

$$V\text{min} = \frac{\Delta V}{2^n - 1} = \frac{10V - (-10V)}{2^{12} - 1} = \frac{20}{4095} = 4{,}88mV \quad (4)$$

$$V\text{min} = \frac{\Delta V}{2^n - 1} = \frac{10V - (-10V)}{2^{8} - 1} = \frac{20V}{255} = 78{,}43mV \quad (5)$$

A figura 12 mostra o diagrama funcional da placa de conversores digital-analógico. Assim como foi feito na placa de contagem de pulsos, o projeto da placa dos conversores digital-analógico também possibilita o controle de até seis servo motores.

Figura 12 – Diagrama funcional da placa dos conversores digital-analógico.

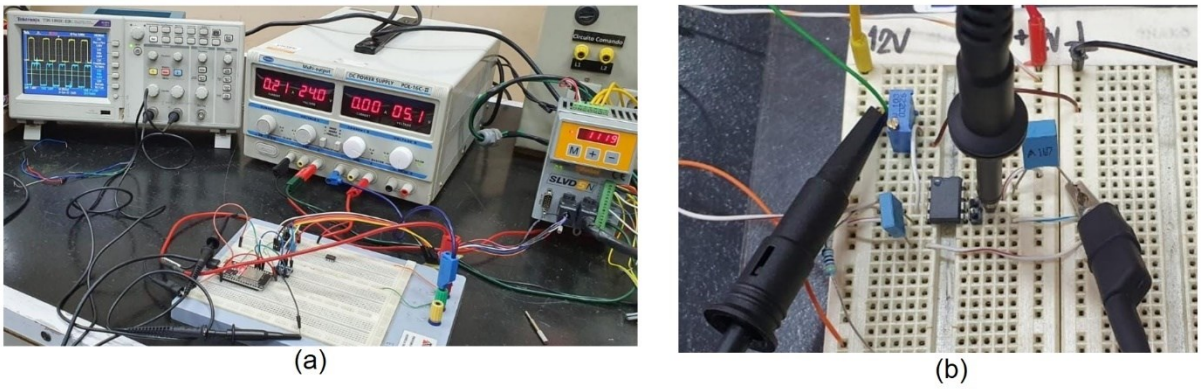


Fonte: Elaborado pelo autor

6 RESULTADOS

Os testes iniciais, para o projeto dos circuitos das placas deram-se em protoboard, de forma isolada. Cada uma das etapas do tratamento do sinal foi montada individualmente, conforme é exemplificado na figura 13: com os testes da contagem de pulsos com a ESP32 e testes com o amplificador operacional de instrumentação.

Figura 13 – Parte dos testes realizados: a. Teste de contagem de pulsos do *encoder*; b. Testes com o amplificador operacional de instrumentação

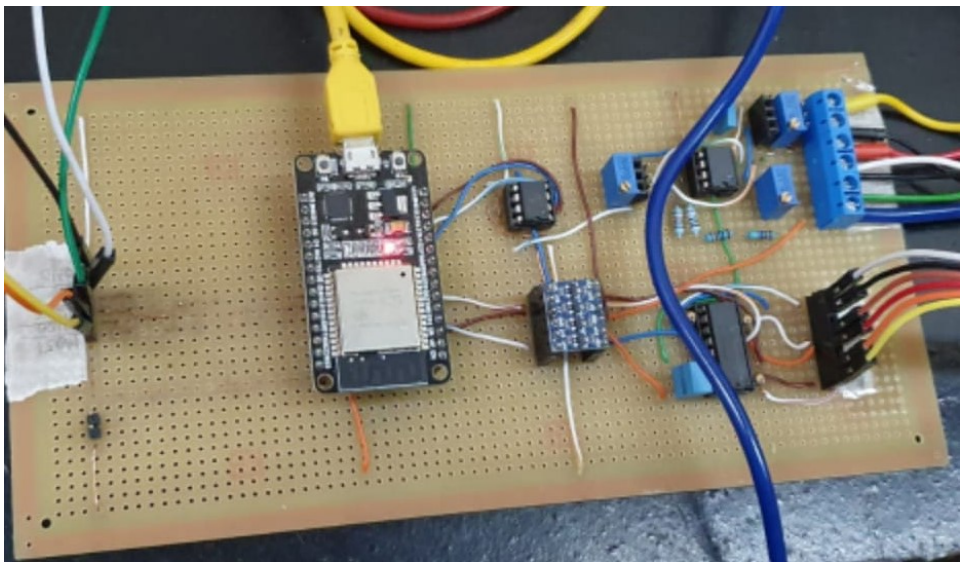


(a) (b)

Fonte: Elaborado pelo autor

Assim que os testes iniciais foram feitos, foi elaborada uma placa protótipo para validar o funcionamento dos circuitos em conjunto, recebimento dos pulsos do *encoder* e envio da tensão de controle de um *servodrive*. Conforme pode ser visto na figura 14, foi utilizada uma placa perfurada padrão.

Figura 14 – Placa protótipo para um servo motor

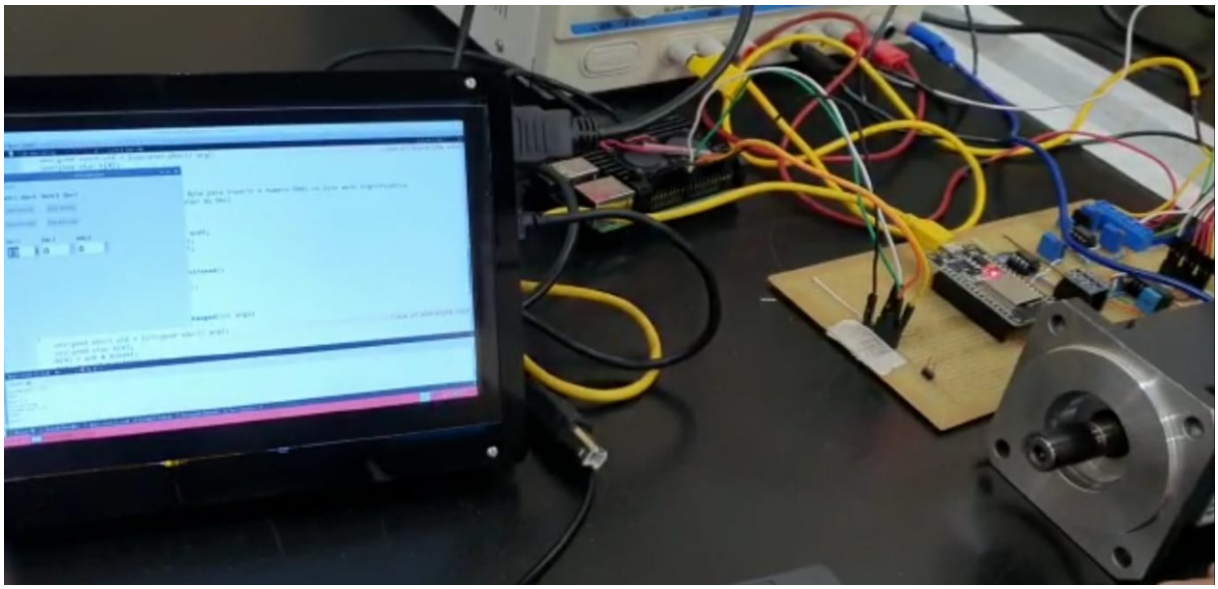


Fonte: Elaborado pelo autor

A partir desse ponto, iniciou-se o esboço da interface de controle no *Raspberry Pi*. Na figura 15 é mostrado um teste onde a velocidade do servomotor é controlada por meio da

tela sensível ao toque. Nesse caso, já está sendo usada interface criada com o *QTCreator* no *Raspberry Pi*.

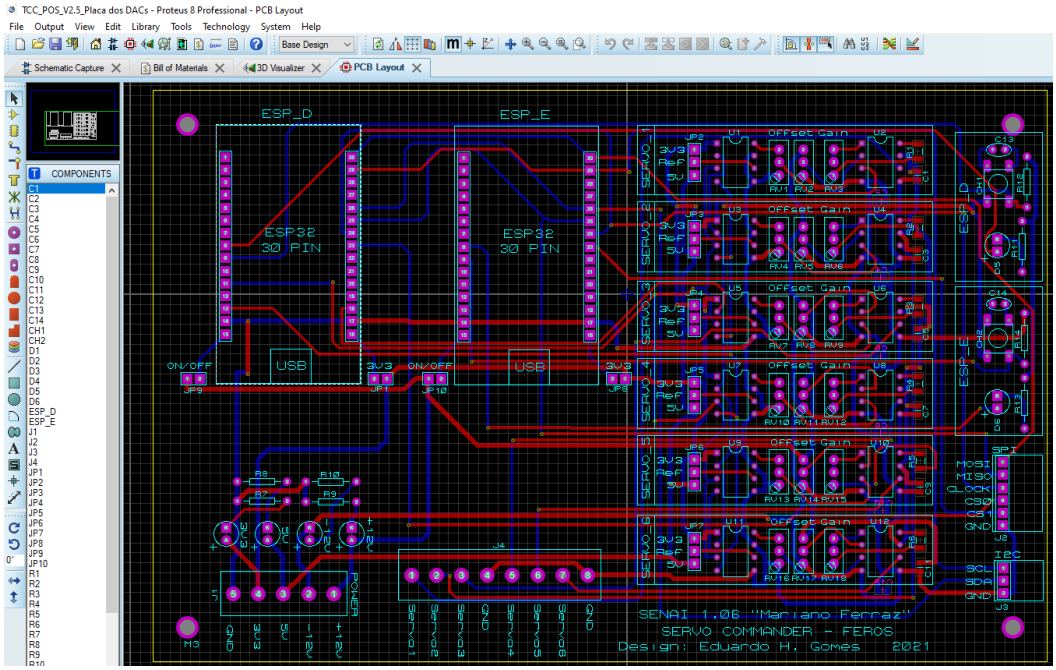
Figura 15 – Teste da interface de controle com tela sensível ao toque



Fonte: Elaborado pelo autor

Com o circuito validado, deu-se início ao projeto do leiaute das placas de circuito impresso (PCI). Esse processo foi realizado no software Proteus, conforme é mostrado na figura 16.

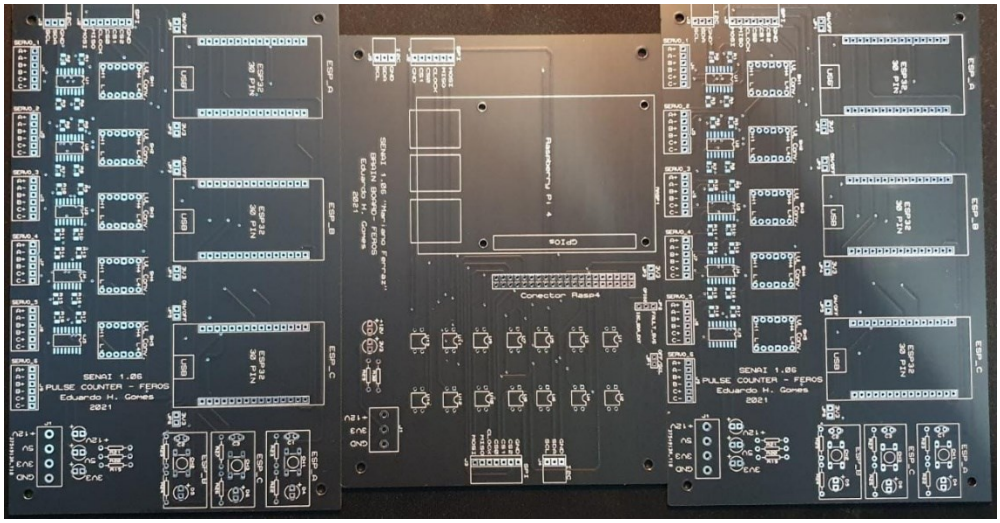
Figura 16 – Leiaute da placa de saídas analógicas feito no Proteus



Fonte: Elaborado pelo autor

Para reduzir o custo, as placas foram confeccionadas na China, onde o pedido mínimo é de cinco placas para cada leiaute. O custo total para se ter as placas em mãos foi de R$330,42. Ou seja, o custo individual de cada uma das quinze placas foi de R$ 22,02. No Brasil, cada uma delas sairia por mais de R$ 600,00. Os componentes para montar todas as placas, incluindo o *Raspberry Pi*, custam R$ 1393,00 no total. As placas podem ser vistas antes da montagem na figura 17.

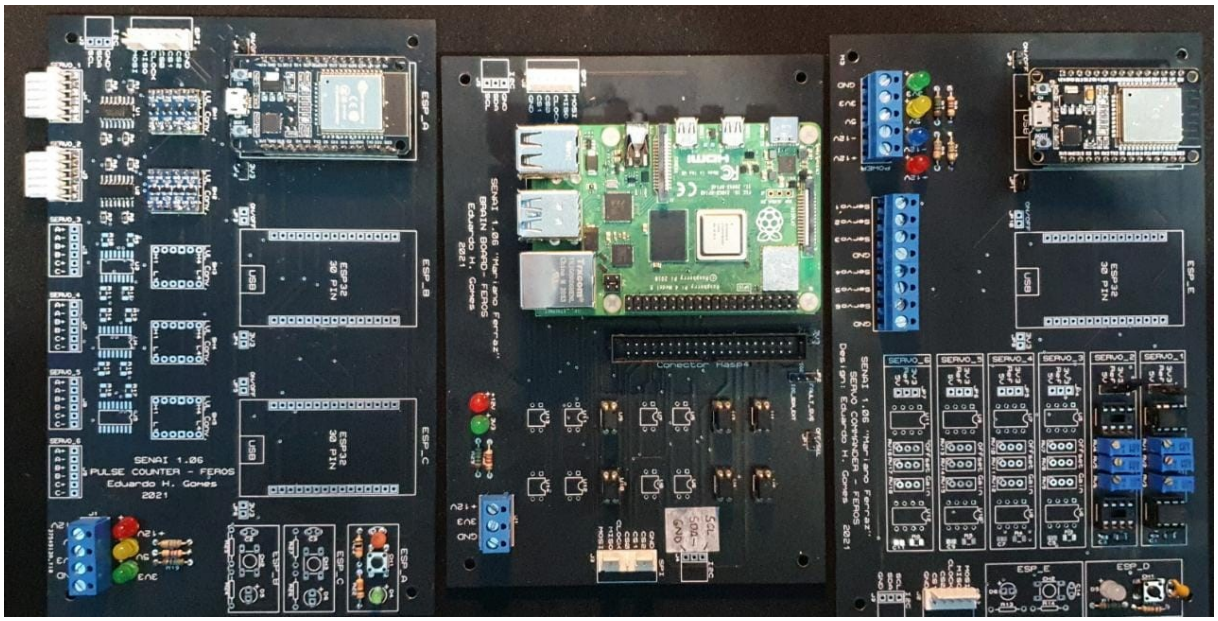
Figura 17 – Placas de circuito impresso confeccionadas



Fonte: Elaborado pelo autor

Para validar o leiaute, as placas foram montadas com a possibilidade de controlar apenas um servomotor. A figura 18 mostra como as placas encontram-se atualmente.

Figura 18 – Placas parcialmente montadas



Fonte: Elaborado pelo autor

O resultado do teste foi satisfatório, uma vez que é possível controlar a velocidade e o sentido de giro do servomotor conforme a necessidade do projeto. Além disso, comprovou-se que o circuito contador de pulsos é capaz de acompanhar a contagem de pulsos do *encoder*.

Outra melhoria a ser implementada em novas versões das placas é o acréscimo de pontos de medição de sinais em pontos específicos, como na saída do amplificador operacional de instrumentação e na saída dos DACs, por exemplo. Dessa forma, os testes práticos em aula podem ser realizados de maneira mais fácil, sem depender de colocar as pontas de prova do equipamento de medição diretamente nos terminais dos componentes.

Para que futuros estudantes possam dar continuidade ao projeto, toda a documentação, como esquemáticos, programas, leiautes, datasheets, planilhas e anotações estão disponibilizadas no repositório a seguir https://drive.google.com/drive/folders/1T7tlfSO2z9c0ILYHtYE6I3sh08D1Z9li?usp=sharing.

7 CONCLUSÃO

Diante dos testes feitos nos protótipos projetados e confeccionados e das informações obtidas, o *retrofitting* proposto neste artigo terá impacto nos seguintes tópicos: possibilitar um melhor controle com uma eletrônica atualizada e mais acessível aos estudantes; permitir aos docentes o estudo de diferentes unidades curriculares como eletrônica analógica, eletrônica digital e programação, além de robótica; e a adição de terceiro servomotor no manipulador deixando-o com três graus de liberdade.

Com isso, considera-se que o *retrofitting* proposto atenderá a condição de tornar o FEROS um equipamento de aprendizagem voltado para os cursos na área de automação industrial. Assim, conclui-se que as mudanças feitas no projeto oferecem condições de tornar o FEROS um equipamento que pode ser incluído nos planos de aula dos cursos das áreas de automação industrial e eletroeletrônica.

REFERÊNCIAS

DIODES INCORPORATED. **BSS138 N-Channel Enhancement Mode MOSFET**. 2021. Disponível em https://www.diodes.com/assets/Datasheets/BSS138.pdf. Acesso em 9 dez. 2021.

FEN LOGIC LTD. **Gertboard User Manual**. 2012. Disponível em https://www.farnell.com/datasheets/1683444.pdf. Acesso em 8 dez. 2021.

HARRINGTON, Kevin. **ESP32Encoder**. Disponível em https://github.com/madhephaestus/ESP32Encoder/. Acesso em 14 fev. 2022.

KUGELSTADT, Thomas. **The RS-485 Design Guide.** Texas Instruments. Disponível em https://www.ti.com/lit/pdf/slla272. Acesso em 9 dez. 2021.

MANSO, Adriano; BARBOSA, Luiz. **Adaptação do sistema de controle do robô FEROS para Raspberry Pi 2**. Trabalho de Conclusão de Curso (Graduação em Tecnologia de Automação Industrial) – Faculdade de Tecnologia SENAI Mariano Ferraz. São Paulo. 2016.

MAXIM INTEGRATED PRODUCTS. **MAX3095/MAX3096**. 2018. Disponível em https://datasheets.maximintegrated.com/en/ds/MAX3095-MAX3096.pdf. Acesso em: 9 dez. 2021.

MELO, Luis; ARAUJO, Reinaldo; MUNHOS, Ricardo; LOPES, Rodolfo; LOPES, Silas. **Desenvolvimento de um robô tipo five-bar linkage: eletrônica e software**. Trabalho de Conclusão de Curso (Graduação em Tecnologia de Automação Industrial) – Faculdade de Tecnologia SENAI Mariano Ferraz. São Paulo. 2011.

PARKER. **SLVD-N User Manual.** 2008. Disponível em: https://www.parker.com/content/dam/Parker-com/Literature/Electromechanical-Europe/User-Guides/192_141101_SLVD-N_manual.pdf . Acesso em: 9 dez. 2021.

RIK. **LC-04 4 Channel logic converter 3.3V – 5.0V - Eletronics-lab**. 2016. Disponível em: <https://www.electronics-lab.com/lc-04-4-channel-logic-converter-3-3v-5-0v/>. Acesso em: 9 dez. 2021.

ROSSATO, Daniel B. **Desenvolvimento de um sistema aberto para ensino de robôs manipuladores**. Dissertação (Mestrado em Engenharia) – Escola Politécnica da Universidade de São Paulo. São Paulo. 2009.

SANTOS, Lucas; SANTOS, Ricardo; SANTOS, Thiago. **Avaliação de desempenho do módulo de aquisição de dados Advantech USB-4716 em sistema Linux Debian 4**. Trabalho de Conclusão de Curso (Graduação em Tecnologia de Automação Industrial) – Faculdade de Tecnologia SENAI Mariano Ferraz. São Paulo. 2012.

SOBRE O(S) AUTOR(ES)

[i] **EDUARDO HENRIQUE GOMES**

Possui pós-graduação em Automação e Controle pela Faculdade SENAI Mariano Ferraz (2021) e graduação em Engenharia de Controle e Automação pelo Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (2012). Tem experiência na área de Engenharia Eletrônica, em projetos de circuitos eletrônicos e programação de microcontroladores. Atualmente, é instrutor na Escola SENAI Mariano Ferraz. https://orcid.org/0009-0003-8476-2191

[ii] **DANIEL BARBUTO ROSSATO**

Possui doutorado em Engenharia Eletrônica e Computação pelo ITA (2022), mestrado em Engenharia Elétrica – Sistemas pela EPUSP (2009) e bacharelado em Engenharia Elétrica – Automação e Controle pela EPUSP (2002). Atualmente é docente na Faculdade SENAI São Paulo – Campus Mariano Ferraz, atuando nas áreas de Sistemas e Controle, Internet das Coisas, Redes Neurais Artificiais, Robótica e Segurança. Tem experiência em manutenção em sistemas de automação industrial. https://orcid.org/0000-0003-1654-3424

[iii] **ANDRÉ LUIS DOS SANTOS**

Possui mestrado em Engenharia Mecânica pela Universidade de São Paulo (2016) e graduação em Engenharia Mecatrônica pela Universidade Paulista (2001). Atualmente é docente na Faculdade de Tecnologia SENAI São Paulo – Campus Mariano Ferraz. Tem experiência em automação industrial e desenvolvimento de software, atuando nos temas: processamento de sinais, tomografia por impedância elétrica, redes de comunicação e microcontroladores.
https://orcid.org/0000-0001-6627-3886

[iv] **PAULO ANDRÉ DOS SANTOS**

Possui mestrado em Engenharia Elétrica pela Universidade Federal do ABC (2014) e graduação em Mecatrônica Industrial pelo SENAI - Departamento Regional de São Paulo (2005). Atualmente é docente na Faculdade SENAI São Paulo – Campus Mariano Ferraz. Tem experiência na área de Engenharia Elétrica, com ênfase em Energia e Automação, atuando principalmente nos seguintes temas: fator de potência, eficiência energética e FPGA. https://orcid.org/0000-0002-1177-1949