CONTROLE DE BRAÇO ROBÓTICO INDUSTRIAL ATRAVÉS DE MINIATURA

José Gomes*, João Marcelo Teixeira†, Gutenberg Barros‡, Veronica Teichrieb§

Emails: jgsn@cin.ufpe.br, jmxnt@cin.ufpe.br, gutenbar@gmail.com, vt@cin.ufpe.br

Abstract— The main goal of the project is to develop an integrated hardware and software control solution for a specific robotic arm model, smart NS 16 1-65, so that its use is facilitated. This solution makes use of the Arduino Platform, which is connected to the interface of the robot through TCP / IP communication, establishing a client-server architecture. The project was validated on COMAU's robotic arm NS 16 - 1.65, assigned to federal university of pernambuco(UFPE).

Keywords— Control, Robotic arm, miniature.

Resumo— O objetivo principal do projeto é desenvolver uma solução integrada de hardware e software de controle para um modelo de braço robótico específico, smart NS 16 1-65, de forma que seu uso seja facilitado. Essa solução faz uso da Plataforma Arduino, que é conectada à interface do robô através de comunicação TCP/IP, estabelecendo uma arquitetura cliente-servidor. O projeto foi validado no braço robótico smart NS 16-1.65 da COMAU, cedido à Universidade Federal de Pernambuco.

Palavras-chave— Controle, Braço robótico, Miniatura.

1 Introdução

Esse trabalho propõe uma solução de automação e controle voltada à robótica industrial, especificamente focando o braço robótico Smart NS 16 1-65 de 6 eixos articulados. A solução encontrada para o projeto, baseada em uma integração *hardware-software*, utiliza a plataforma Arduino como ponte entre o controle realizado pelo usuário e o braço robótico propriamente dito. Através da porta serial, o Arduino se conecta a um computador e este repassa as informações capturadas para o sistema do robô via comunicação TCP/IP. Esse tipo de comunicação foi escolhida pois permite uma maior confiabilidade na transmissão dos dados, visto que, além de enviar a informação na ordem correta, o que é necessário para o bom funcionamento do projeto, ela também dificulta que a informação seja corrompida, além de ser nativamente suportada pelo sistema do robô.

Já na parte de *hardware* o projeto é composto de uma miniatura customizada do braço robótico utilizado no trabalho, elaborada através de modelagem e posterior impressão 3D, que simula todos os movimentos do robô usando como sensores 6 potenciômetros, cada um deles correspondendo a um eixo, que são responsáveis por enviar para o Arduino a informação de deslocamento angular de cada junta.

Este artigo está organizado da seguinte maneira: a Seção 2 apresenta uma descrição de trabalhos relacionados; a Seção 3 apresenta uma descrição do braço robótico utilizado no projeto, assim como uma apresentação da linguagem PDL2 empregada no sistema do robô; a Seção 4 trata da metodologia adotada neste trabalho; a Seção 5 apresenta a validação da solução; a seção 6 introduz os resultados por fim, a Seção 7 apresenta a conclusão do trabalho assim como direções de trabalhos futuros.

2 Trabalhos relacionados

O trabalho descrito em (Achari et al., 2018) tem por objetivo desenvolver um sistema de interação homem computador (IHC) que permita reconhecer certos gestos de mão de humanos para transformar essa informação em comandos específicos. O controle do braço robótico é feito via gestos de mão através de algoritimo de reconhecimento de objetos. A imagem é capturada através de WebCam com interface ligada a uma rasberry pi, que por sua vez enviava via *wireless* a informação pra o arduino uno. Este processava a informação e controlava servomotores que atuavam no braço robótico. Por fim, foi possível concluir que a técnica de processamento de imagem é possível de ser usado para controlar um braço robótico ou outro tipo de sistema.

O trabalho descrito em (Alvarenga Andrade et al., 2016) apresenta o desenvolvimento de uma aplicação que visa conectar duas plataformas distintas, Android e LEGO Mindstorms NXT, para o controle de um robô móvel. Essa conexão possibilitou descrever trajetórias circulares além de permitir ao usuário ter controle manual sobre o robô para que descreva a trajetória desejada, fazendo uso de uma arquitetura cliente-servidor, na qual a plataforma Android tem o papel de cliente e o robô representa o servidor. Para a integração entre as plataformas foi utilizada uma conexão *bluetooth*.

Por sua vez o trabalho descrito em (Chae et al., 2018) propõe um método para calcular probabilidades Bayesian e estimar movimentos usando EMG e dados orientados para dispositivos Myo. O propósito do método consiste no processamento, treino e estágios de reconhecimentos. Orientação e os dados de EMG são obtidos através dos dispositivos Myo. Os dados orientados são transmitidos para o calculo da probabilidade Bayesian, passo do estágio de treino. Para o cal-

culo dessa probabilidade é usado um algorítimo genético que obtém pesos, que são então transferidos para a etapa de estimativa de movimento na etapa de reconhecimento. Por ultimo, por possível chegar a conclusão que mostramos que os movimentos do braço podem ser estimados aprendendo a probabilidade Bayesiana com apenas um dispositivo Myo de braço inferior após aprender usando dois dispositivos Myo: um em cada parte superior e inferior de um braço. Apesar dos trabalhos citados serem voltados para a área de controle, o projeto proposto se diferencia deles na característica de trabalhar com uma versão em miniatura do equipamento real, como forma de controle do movimento, além de integrar três interfaces de programação distintas (C/C++ para Arduino, C/C++ para Windows Desktop e PDL2 para o braço robótico da COMAU).

3 Descrição do robô

O braço robótico utilizado no projeto corresponde ao modelo Smart NS 16-1.65 da COMAU, ilustrado na Figura 1. O mesmo constitui um braço articulado de 6 eixos, no qual os limites dos graus de liberdade podem ser observados na Figura 2 e na Figura 3. Este robô foi especificamente projetado para aplicações em que a precisão e a rapidez são imprescindíveis como em solda em arco, corte de plasma e manuseio de embalagem. Suportando um máximo de carga de 16kg, é parte fundamental para uma ampla gama de montagens.

Figura 1: Braço robótico Smart NS 16-1.65 (Comau, 2003aa).

Por padrão, o braço robótico possui duas formas básicas de controle. Na primeira forma o usuário pode manualmente mover cada um dos 6 eixos, através do TP (*Teach Pendant* ou Terminal de Programação), ilustrado na Figura 4, para a posição desejada, respeitando suas limitações físicas. Na segunda forma, o usuário pode endereçar os valores com a posição para a qual o braço deve ir, estando o software do braço responsável pelo cálculo de cinemática necessário para que o movimento seja realizado.

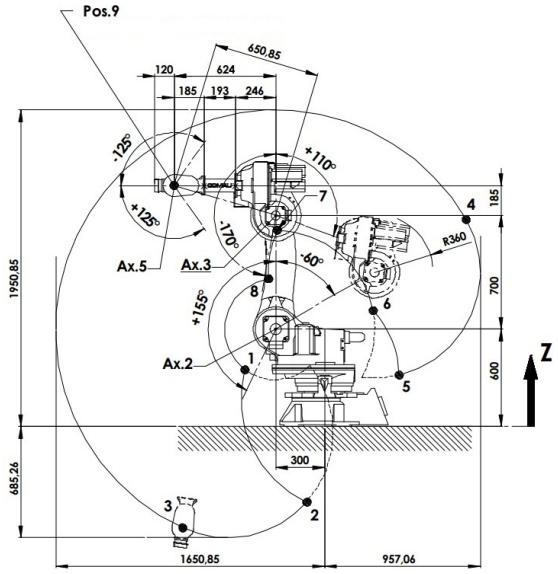

Figura 2: Limites angulares do movimento dos eixos 2, 3 e 5 (Comau, 2003ac).

Com o objetivo de programar o braço robótico nativamente, tem-se a linguagem de programação PDL2. Tal linguagem baseia-se na linguagem Pascal, sendo usada para escrever programas e aplicações com o propósito de: mover braços robóticos, seguir um fluxo no programa (if, while, repeat, ...), receber e enviar informações de arquivos, entre outras. Os benefícios do uso dessa linguagem se dão principalmente ao fato dela incluir mais de 200 rotinas ou seja, funções pré-definidas a atuarem em algum processo. Com o uso de PDL2 é possível executar mais de 250 programas simultaneamente, tendo controle sobre todos os aspectos da aplicação. Além disso, fornece um ambiente de edição integrado, incluindo checagem de sintaxe e execução de instruções que estão disponíveis para o controle do robô (SPA et al., 2015).

4 Controlando os movimentos do braço robótico COMAU com sua miniatura

4.1 Modelagem do problema

O objetivo principal do projeto é replicar o controle dos movimentos do braço robótico real através de movimentos realizados em uma versão miniaturizada do mesmo. A solução proposta fará o envio do valor individual de cada ângulo das juntas para o robô, não sendo necessário nenhum cálculo de cinemática para se chegar à posição destino. A Figura 5 ilustra o modelo em miniatura desenvolvido através de uma perspectiva explodida, sendo possível visualizar todas as partes que o compõem.

Paralelo à escolha da forma de controle, determinou-se que potenciômetros seriam a forma mais adequada para um primeiro protótipo, de baixo custo, tendo em vista a facilidade de ob-

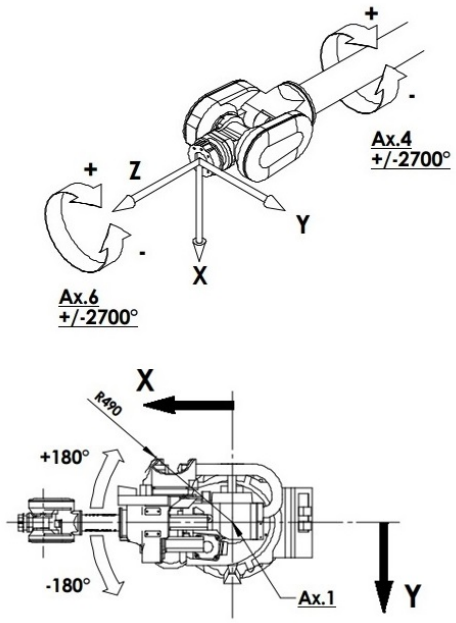

Figura 3: Limites angulares do movimento dos eixos 1, 4 e 6 (Comau, 2003ac).

Figura 4: Controle atual (TP) do braço robótico (Comau, 2003ab).

tenção desses dispositivos no mercado. Os potenciômetros usados possuem intervalo de medição de 0 a 300 graus.

O próximo passo consistiu em determinar como a parte de software seria contruída de forma a se integrar com o hardware utilizado, determinando-se assim a utilização de uma arquitetura cliente/servidor, usando o protocolo TCP/IP como parte principal da comunicação.

4.2 Componentes de hardware do braço robótico COMAU em miniatura

4.2.1 Braço robótico em miniatura: projeto e impressão 3D

O modelo construído para atuar como interface de controle do braço Comau Smart NS 16 – 1.65 foi concebido em escala aproximada de um décimo

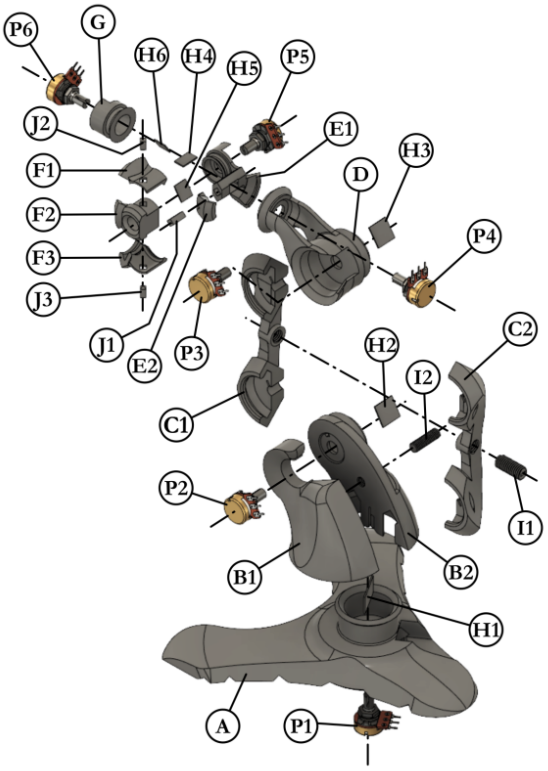

Figura 5: Perspectiva explodida do modelo interativo mostrando os elementos internos, externos e os potenciômetros (elementos P1 a P6).

do maquinário real. Uma vez importado em um software de modelagem paramétrica e escalonado para 160 mm de altura, o modelo digital original do braço robótico, obtido no site da COMAU (Comau, 2003b), trouxe a posição exata dos seis eixos de giro e dos seus planos de apoio para esses movimentos. O modelo para controle foi então idealizado ao redor dessa informação, preservando a proporção da distância entre os eixos.

Como os potenciômetros usados para captar a informação de rotação dada pelo usuário controlador foram posicionados nos pontos de rotação das peças, as estruturas do modelo foram construídas ao redor deles, não só para abrigá-los e manter os corpos fixados como para não interferir na captação das medidas. Desta maneira, o giro do eixo do potenciômetro é realizado através de uma pequena ligação plana na peça giratória (de cada junção), a qual passa por dentro da fenda do eixo do potenciômetro, travando-o. Para evitar que novas peças necessitem ser refabricadas por quebra dessa estreita ligação, ela foi substituída por pequenos paralelepípedos inseridos em fendas nas peças giratórias (Figura 5, elementos H1 a H6). Dessa maneira, em caso de quebra, apenas essa pequena peça precisa ser substituída, tornando a manutenção barata e rápida.

Duas versões do modelo reduzido para controle foram construídas. A primeira era consti-

tuída de quatro elementos, chamados de Base (Figura 6, elemento A), Corpo (Figura 6, elemento B), Braço (Figura 6, elemento C) e Mão (Figura 6, elemento D), além de três travas de giro para os eixos dos potenciômetros a serem posicionadas nas junções das peças (Figura 5, elementos H1, H2 e H3). Desta maneira, cada elemento abrigava um potenciômetro, exceto o Braço, visto que nas duas pontas de junção os dispositivos eletrônicos estavam nas outras peças conectadas, o Corpo e a Mão. O braço também foi dividido em duas partes (Figura 5, elementos C1 e C2) para que pudesse ser encaixado e travado, através de um parafuso transversal ao seu comprimento (Figura 5, elemento I1), ao redor de trilhos existentes no Corpo e na Mão. Fendas existentes nas laterais das partes do Braço expõem o trilho e permitem aumentar o atrito para enrijecer o giro deste elemento, já que ele suporta quase todas as peças do modelo em possíveis posições de balanço.

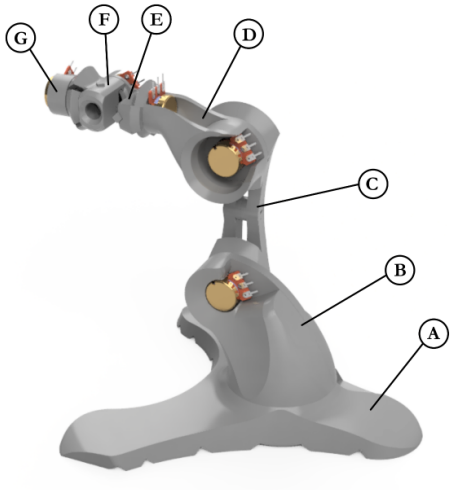

Figura 6: Perspectiva da segunda versão do modelo interativo. A: Base; B: Corpo; C: Braço; D: Mão; E: Falange; F: Falanginha; G: Falangeta.

Através desta primeira versão, verificou-se a fragilidade do encaixe Base-Corpo, então finas hastes. Também houve dificuldade em extrair a estrutura de suporte da Mão e este ato comprometeu a estrutura. A superfície externa do Braço, por erro de produção, posicionada sobre estruturas de suporte, ficou áspera, mas isso não afetou o desempenho.

Também percebeu-se que o alcance de giro dos potenciômetros não coincidia com a medida verificada no braço robótico real, requerendo que eles fossem reorientados. Dessa maneira, a produção das peças restantes foi suspensa e uma nova versão do modelo começou a ser produzida.

Na segunda versão do modelo, o Corpo foi dividido em duas partes (Figura 5, elementos B1 e B2), exatamente como acontece com o Braço desde a versão 1, para que a base passasse a ter um trilho circular completo, mais resistente do que as hastes usadas no primeiro teste. O local destinado aos potenciômetros foi reorientado para alinhar a bissetriz do alcance de giro do eixo de cada um com a bissetriz do giro do relativo eixo do braço da COMAU. Como os potenciômetros possuem um alcance maior do que os eixos do Smart NS 16 1-65, todos os ângulos de rotação necessários estariam compreendidos.

Além disso, os elementos restantes foram finalizados já com a orientação necessária aos potenciômetros. Foram nomeados como Falange (com duas partes, mostradas na Figura 5, elementos E1 e E2), Falanginha (dividida em três peças, apresentadas na Figura 5, elementos F1, F2 e F3) e Falangeta (Figura 5, elemento G) e, juntamente com a conexão com a Mão, abrigam três potenciômetros (Figura 5, elementos P4 a P6), consequentemente três eixos de rotação e três travas de rotação (Figura 5, elementos H4 a H6), em um pequeno espaço. A Falange e a Falanginha foram divididas em duas e três partes, respectivamente, para que elas pudessem ser encaixadas de modo a envolver os trilhos circulares que permitem os giros dos elementos. Posteriormente foram travadas com cunhas encaixadas sob pressão (Figura 5, elementos J1 a J3). Dessa maneira, a Falange permite o giro no eixo longitudinal, a Falanginha, em um eixo perpendicular a ele, e a Falangeta, no mesmo eixo longitudinal, que então poderá ter sido rotarionado pela Falanginha.

As peças foram exportadas como arquivos no formato STL e abertas no software de fatiamento para impressão Cura, versão 3.2, da Ultimaker (Ultimaker, 2018), em grupos, de acordo com a semelhança de forma e de características necessárias para a impressão. A Tabela 1 relaciona estes grupos de peças. A Tabela 2 apresenta as configurações de impressão comuns a todos os grupos. Por fim, na Tabela 3 é possível ver as configurações que sofreram variação de parâmetros de acordo com os grupos.

Os grupos foram exportados em formato G-Code e construídos na tecnologia de manufatura aditiva Fused Filament Fabrication de uma impressora de mesa quadrada de 200 mm de lado.

4.2.2 Potenciômetros : controle do movimento

Paralelo à escolha da forma do controle, determinou-se que potênciometros seriam a forma mais adequada para um primeiro modelo de baixo custo. Os 6 potenciômetros usados são conectados às portas analógicas de um Arduino Mega e os valores lidos são representados com 10 bits, representando um intervalo entre 0 a 1023, onde os 300 graus de variação estão compreendidos. Devido à alta variabilidade dos potenciômetros de baixo custo utilizados, fez-se necessário um estudo para

Tabela 1: Agrupamento dos objetos para impressão, segundo a nomenclatura mostrada na Figura 5.

Grupo	Partes
1	as duas partes do Braço (C1 e C2)
2	as duas partes do Corpo (B1 e B2)
3	a parte maior da Falange (E1)
4	Falangeta e parte central da Falanginha (G e F2)
5	as partes superior e inferior da Falanginha e a parte menor da Falange (F1, F3 e E2)
6	Mão (D)
7	Parafusos de união do Corpo e do Braço (I1 e I2)
8	Base (A)
9	travas de giro dos potenciômetros e cunhas de travamento da Falanginha e da Falange (H1 a H6 e J1 a J3)

Tabela 2: Parâmetros de impressão comuns.

Material	PETG
Temperatura	215°
Temperatura da mesa	70°
Altura da camada	0,3 mm
Espessura da camada da base e do teto	0,6 mm
Padrão de construção da parede	linhas
Velocidade de impressão	60 mm/s
Velocidade de impressão da parede	40 mm/s
Ângulo mínimo para suporte	50°
Limite das camadas adaptativas	350

poder associar cada valor de ângulo com um valor compreendido no intervalo de saída lido do potênciometro.

Esse estudo foi realizado medindo-se os ângulos de rotação utilizando um círculo trigonométrico e a saída do potênciometro sendo lida em um código implementado na plataforma Arduino, permitindo assim obter a relação variação ângular / valor de saída do potenciômetro, de extrema importância para a validação do projeto. Todos os 6 potenciômetros foram analisados, e as informações coletadas armazenadas em tabelas.

4.3 Componentes de software do braço robótico COMAU em miniatura

4.3.1 Arduino: leitura dos potenciômetros

Das três partes de software que compõem este projeto, a primeira delas foi implementada na plataforma Arduino e é responsável por receber os valores enviados pelos 6 potenciômetros e criar um pacote com essa informação. É mostrado na Listagem 1 o código feito no Arduino, o qual é importante ressaltando o papel da função analogRead responsável por ler os valores do potenciômetro correspondente e setar esses valores em val. Já as variáveis inteiras analogpin são usadas com o propósito de se ligar cada uma a um potenciômetro, o que ocorre ao se determinar um valor a elas correspondente ao pino analógico do Arduino ao qual o potenciômetro foi ligado. Programado isso, o programa escrito na linguagem de programação C está conectado à porta serial do Arduino recebendo o pacote de informações vindo dos potenciômetros. A placa Arduino utilizada foi a Mega 2560 R3 por oferecer uma maior quantidade de portas analógicas e digitais.

Listagem 1: Código implementado na plataforma Arduino para coleta dos valores dos potenciômetros.

```
int val[6];

void setup() {
  Serial.begin(9600);
}

void loop() {
  for(int i=0; i<6; i++){
      analogValues[i]=analogRead(i);
    }
    Serial.write((unsigned char *)
        analogValues, 12);
    delay(20);
}
```

4.3.2 Computador: ponte entre plataformas

A implementação em C foi necessária no lado do computador, ficando responsável por realizar uma função de ponte, conectando-se ao Arduino via serial e ao robô via ethernet (TCP/IP). Além de servir de ponte de comunicação, a aplicação desktop, antes de enviar os dados para o robô, realiza a conversão dos valores lidos de cada potenciômetro para os ângulos correspondentes, obtidos e armazenados nas tabelas mencionadas anteriormente. A Listagem 2 ilustra um trecho da implementação que representa a conexão com a porta serial do Arduino (neste exemplo representada pela COM2) e exibe também a parte na qual o programa recebe a informação e utiliza esta para realizar a tradução valor/ângulo do potenciômetro.

O trecho de código ilustrado na Listagem 3 é também responsável pelo envio periódico das mensagens através do laço criado utilizando uma estrutura *while*. O envio através do socket TCP é realizado pela função *send*.

Tabela 3: Configurações variáveis pelos grupos de impressão.

Grupo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Espessura da parede (mm)	0,8	0,8	0,8	0,8	0,8	0,8	0,8	1,2	0,8
Densidade do preenchimento interno	12%	12%	12%	12%	12%	12%	12%	13%	12%
Camadas de preenchimento interno gradual	0	0	0	0	0	0	0	4	0
Altura da camada de preenchimento gradual	-	-	-	-	-	-	-	3	-
Padrão do suporte	Grade	Zig Zag	Grade	Grade	Grade	Grade	Grade	Grade	Grade
Densidade do suporte	20%	12%	25%	25%	5%	20%	25%	20%	25%
Distância X/Y do suporte (mm)	2	2	1	1	0,7	1,5	1	2	1
Camadas de suporte gradual	5	5	5	5	0	5	5	5	5
Altura da camada do suporte gradual (mm)	1	1	2	2	-	2	2	1	2
Tipo de adesão à mesa de impressão	Skirt	Skirt	Brim	Skirt	Brim	Brim	Brim	Skirt	Skirt
Variação das camadas adaptativas	0,3	0,3	0,3	0,3	0,3	0,3	0,3	-	-
Altura das camadas adaptativas	0,1	0,1	0,1	0,1	0,1	0,1	0,1	-	-

Listagem 2: Código implementado na plataforma Arduino para coleta dos valores dos potenciômetros.

```
int main() {
  SerialPort jgn;
  int error=jgn.conncet("COM2");
  unsigned char *bjs;
  bjs=(unsigned char*)malloc(sizeof(
      short) * 6);
  short *bjs2 = (short *)bjs;
  int i, k, t, a, excolha, j, port
      [6][1024], save[6];
  char filename[50];
  memset(save, 0, sizeof(int) * 6);

  for (i = 0; i < 6; i++) {
    sprintf(filename, "pot%d.txt", i);
    FILE *file = fopen(filename, "r");
    a = 0;
    if (fopen(filename, "r") == NULL)
        {
      printf("impossivel␣abrir␣o␣
          arquivo\n");
      a = 1;}
    if (a == 0) {
      for (k = 0; k < 1024; k++) {
        fscanf(file,"%d",&pot[i][k]);
      }
    fclose(file);
    }
  }
(...)
```

Listagem 3: Código implementado na plataforma Arduino para coleta dos valores dos potenciômetros.

```
while(1) {
  int r = jgn.getArray(bjs, 12);
    if(r != 12) continue;
    for(i = 0; i < 6; i++){
      if(bjs2[i] >= 0) save[i] = pot[i
          ][bjs2[i]];
      printf("%d", bjs2[i]);
      }
      printf("\n");
      for(i=0; i < 6; i++){
        if (bjs2[i] >= 0) save[i] =
            pot[i][bjs2[i]];
        printf("%d" save[i]);
          }
      printf("\n\n");
          send(sockfd, (const char *)(
              save), sizeof(int) *6,
              0);
        }
}
closesocket(sockfd);
printf("cliente␣encerrado");
}
```

4.3.3 Braço robótico: execução do movimento

A implementação no lado do robô, realizada em PDL2, é responsável por receber a informação que vem do computador via TCP/IP e enviar para

cada eixo o deslocamento angular que eles devem realizar. A primeira observação a ser feita é na classificação do programa como *HOLD*, ou seja, esse programa pode conter comandos que movimentem o braço.

A Listagem 4 expõe um trecho do código responsável por atribuir os valores de deslocamento angular nos eixos específicos. Através da função *READ lun_tcp_rs* os valores são lidos do socket TCP diretamente para a variável temporária, e depois passados à variável *ZERO*. A função *WRITE LUN_CRT* escreve os valores recebidos no console do TP, para fins de *debugging*. Por fim, a função *MOVE* desloca as juntas para as posições angulares informadas.

Listagem 4: Código implementado na plataforma Arduino para coleta dos valores dos potenciômetros.

```
VAR
luc_tcp_rs : INTEGER
comm_success : BOOLEAN
shutdown : BOOLEAN EXPORTED FROM
    ROS_COMAU_trajectory
tmp_msg : ROS_comau_robot_status_msg
is_moving : BOOLEAN EXPORTED FROM
    ROS_COMAU_motion
ROUTINE status_decode2 : BOOLEAN
    EXPORTED from funciona
ROUTINE tcp_accpet(port, vi_netlun,
    vi_scrlun, verbose : INTEGER)
    EXPORTED FROM ROS_COMAU_tcp_utils
ROUTINE status_decode2 : BOOLEAN
var
ZERO : JOINTPOS
i : INTEGER
int_field : INTEGER
                ANG : ARRAY[6] OF
                    INTEGER
header : ROS_message_header
arm1 : JOINTPOS          --POSITION
            BEGIN
            for i :=1 TO 6 DO
                READ lun_tcp_rs(ANG[i)
                    ::4)
            ENDFOR
                WRITE LUN_CRT (NLZ, '␣
                    valores␣recebidos:
                    ', ANG[1],ANG[2],
                    ANG[3],ANG[4],ANG
                    [5],ANG[6], NL)
            FOR i := 1 to 6 do
            zero[i] :ANG[i]
            ENDFOR
MOVE TO ZERO
RETURN (TRUE)
END status_decode2
```

5 Validação

Os testes foram propostos com o intuito de validar a miniatura desenvolvida como uma alternativa para o controle do braço robótico real. Inicialmente a validação do protótipo desenvolvido utilizou o software de simulação 3D do braço (RoboSim, simulador que faz parte da ferramenta WinC5G) , que é capaz de simular todas as ações do braço. Nele, através do uso da miniatura foram realizados uma série de movimentos nos eixos, inclusive movimentos simultâneos dos eixos, com o intuito de que o braço robótico no simulador repetisse os movimentos de maneira coerente. Após a constatação desse fato foram realizados testes no braço robótico real, onde foi possível controlar o movimento do equipamento real utilizando a miniatura como forma de controle . Alguns quadros comparando o movimento realizado na miniatura e o resultado realizado pelo simulador podem ser vistos na Figura 7 e um vídeo demostrativo da validação pode ser visto em https://bit.ly/2KsY8A6. E em https://bit.ly/2KFmmGx pode ser visto o vídeo que demonstra um dos testes feitos no braço robótico real.

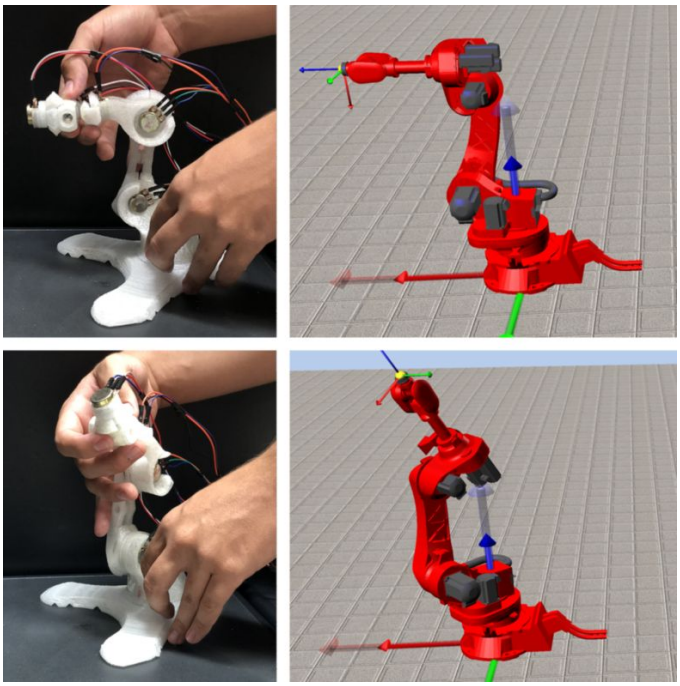

Figura 7: Código em C responsável pelo envio de informação ao sistema do robô.

6 Resultados

Após realizar os testes de validação foi possível notar que o sistema proposto apresentou o comportamento desejado, proporcionando um controle em tempo real do equipamento robótico e de forma mais natural, uma vez que os movimentos realizados na miniatura eram mimetizados pelo braço robótico simulado. Além disso, verificou-se que houve êxito na implementação das diversas partes que compõem o trabalho.

Todavia, notou-se a partir dos testes uma deficiência na precisão do modelo de potenciômetro utilizado. O programa que executa no computador e é responsável por realizar a ponte entre a entrada através dos potenciômetros e o robô é bastante dependente da calibração valor/ângulo, realizada previamente. Com a utilização do protótipo, há desgaste dos potenciômetros e os valores de calibração não correspondem mais aos valores iniciais adquiridos, pois a relação angular é modificada. Esse problema fez com que o mapeamento do movimento ficasse menos preciso, porém ainda funcional.

7 Conclusão

Esse projeto propôs uma alternativa de controle de um braço robótico industrial utilizando a plataforma de desenvolvimento Arduino, um programa escrito na linguagem de programação C e o software do braço em questão. A arquitetura cliente/servidor foi responsável pelo esquema de comunicação entre miniatura e braço robótico real através do protocolo TCP/IP.

Foi possível concluir com a miniatura desenvolvida que a alternativa de controle proposta é viável, assim como os benefícios por ela proposta como tornar o processo de controle mais dinâmico e aumentar a mobilidade de movimento.

Para o futuro almeja-se melhorar a qualidade dos resultados que foram alcançados com os testes no braço robótico real, para assim poder expandir para diferentes equipamentos este tipo de controle através de miniatura. Além disso, foi considerada a implementação do protocolo UDP de comunicação ao invés do TCP usado, porém acredita-se que não haverá impacto significativo, visto a frequência de transmissões realizadas entre robô e miniatura ser baixa. Sabe-se que o protocolo UDP possui um overhead significativamente menor, mas dada a quantidade de dados a ser transmitida por vez e o tempo para realizar o movimento, a diferença entre eles é de pelo menos duas ordens de grandeza. Há também o interesse em utilizar uma forma mais confiável de se medir o deslocamento angular no controle em miniatura de forma a não depender de uma relação definida anteriormente, como por exemplo fazendo uso de *encoders ópticos*.

Estão sendo realizados testes com usuários reais a fim de ser possível mensurar o quão facilitado é o controle do braço robótico real através da miniatura proposta, em comparação ao controle padrão realizado usando o TP.

Também serão estudadas alternativas e melhorias no processo de modelagem e impressão 3D da miniatura. Além disso, está em processo de testes uma solução que utiliza um *shield ethernet* que conecta diretamente a plataforma Arduino ao robô, para que não seja necessário o intermédio do PC na comunicação.

Agradecimentos

Os autores gostariam de agradecer à COMAU, em nome de William Teotônio dos Anjos, pela disponibilidade, suporte e valiosos ensinamentos a respeito do controle do braço robótico Smart NS 16-1.65.

Referências

Achari, S. M., Mirji, S. G., Desai, C. P., Hulasogi, M. S. and Awari, S. P. (2018). Gesture based wireless control of robotic hand using image processing, *GESTURE* **5**(05).

Alvarenga Andrade, G., Milanes, A. and Argolo Jesus, T. (2016). Integração das plataformas android e lego mindstorms nxt para controle de um robô móvel, *Anais do XXI Congresso Brasileiro de Automática - CBA2016*, Sociedade Brasileira de Automática, pp. 2007–2012.

Chae, J., Jin, Y., Sung, Y. and Cho, K. (2018). Genetic algorithm-based motion estimation method using orientations and emgs for robot controls, *Sensors* **18**(1): 183.

Comau (2003aa). Braço robótico NS 16 1-65, `http://www.comau.com`. [acessado em outubro de 2017].

Comau (2003ab). Controle atual (TP) do braço robótico, `www.comau.com`. [acessado em outubro de 2017].

Comau (2003ac). NS 16 - 1.65 Working Areas, `http://www.comau.com/Download/robot/ns16/Comau_NS16Arc_workingareas.pdf`. [acessado em outubro de 2017].

Comau (2003b). NS 16 - 1.65 CAD Files - STL version, `http://www.comau.com/Download/robot/ns16/cad-files/EN_arc3dstep_ns16.zip`. [acessado em 29 de setembro de 2017].

SPA, C., TEKNIKER, F., EV, A. F., SIRKETI, T. T. O. F. A., TEKNIKER, L. and PILZ, C. (2015). Expert cooperative robots for highly skilled operations for the factory of the future.

Ultimaker (2018). Ultimaker Cura software, `https://ultimaker.com/en/products/ultimaker-cura-software`. [acessado em 6 de fevereiro de 2018].