BRAÇO COLABORATIVO DE FÁCIL PROGRAMAÇÃO

ETEC Rosa Perrone Scavone

Itatiba - SP

Giovanna de Morais Nunes
Julia Oliveira
Laura Prado Bezerra

Orientador: Prof. Ms. Anderson Wilker Sanfins
Coorientador: Prof. Wellington Fernandes Barbosa

Período de Desenvolvimento do Projeto: Fevereiro à Outubro de 2024

Artigo Científico

SUMÁRIO

RESUMO DO PROJETO

Este artigo descreve o desenvolvimento de um braço robótico na ETEC Rosa Perrone Scavone – Itatiba SP, inspirado em modelos industriais, com o propósito de simplificar atividades diárias. Integrando tecnologias avançadas e uma interface amigável, o sistema permite uma programação intuitiva e acessível mesmo para usuários sem experiência em robótica. A capacidade de alternar entre modos manual e automático, juntamente com a funcionalidade de gravação e execução de movimentos, é um diferencial para esse projeto educacional. Priorizando segurança e interação homem-máquina, o braço robótico é versátil e adaptável, podendo ser utilizado em uma variedade de ambientes. Suas aplicações incluem tarefas como manipulação de objetos e assistência em montagem, tanto em ambientes domésticos, industriais e educacionais, quanto para cursos técnicos e superiores na área de tecnologia. Utilizando os conhecimentos do curso técnico em Automação Industrial, com a projeção das peças, impressão, montagem e programação, tem-se por objetivo a construção desse robô de fácil utilização.

Palavras-chave: Robótica, Programação e Interação

ABSTRACT

This article describes the development of a robotic arm at ETEC Rosa Perrone Scavone – Itatiba SP, inspired by industrial models, with the purpose of simplifying daily activities. Integrating advanced technologies and a user-friendly interface, the system allows intuitive and accessible programming even for users with no experience in robotics. The ability to switch between manual and automatic modes, along with the recording and execution functionality of movements, is a differentiator for this educational project. Prioritizing safety and human-machine interaction, the robotic arm is versatile and adaptable and can be used in a variety of environments. Its applications include tasks such as object manipulation and assembly assistance, both in domestic, industrial and educational environments, as well as for technical and higher education courses in the area of technology. Using knowledge from the technical course in Industrial Automation, with the design of parts, printing, assembly and programming, the aim is to build this easy-to-use robot.

Keywords: Robotics, Programming and Interaction

Artigo Científico

INTRODUÇÃO

Atualmente, já existem diversos braços robóticos utilizados na produção industrial, medicina e engenharia, além das mais diversas áreas de fabricação. Segundo o site Engenharia Híbrida (2022), os primeiros braços robóticos surgiram na década de 60 e foram desenvolvidos principalmente para aplicações industriais, como por exemplo, os braços Unimate (1961), Stanford (1969), PUMA (1978) e SCARA (Seleção de Articulações Robóticas para Aplicações de Montagem, 1978).

Unimate (1961): O Unimate (Figura 1), criado por George Devol e Joseph Engelberger, é amplamente considerado o primeiro robô industrial. Era um braço robótico utilizado na indústria automobilística para tarefas de soldagem. Este foi um avanço significativo, pois marcou o início da automação industrial.

Figura 1 – Unimate (1961) – Primeiro braço robótico do mundo

Braço Stanford (1969): O Braço Stanford (Figura 2), desenvolvido na Universidade Stanford, foi um dos primeiros braços robóticos controlados por computador e serviu como uma plataforma de pesquisa importante para o desenvolvimento de técnicas de controle e programação de robôs.

Figura 2 – Stanford (1969)

Braço PUMA (1978): O braço PUMA (Programmable Universal Machine for Assembly) foi desenvolvido pela empresa Unimation e se tornou um dos primeiros braços robóticos amplamente comercializados. Ele foi projetado para ser mais versátil e foi utilizado em diversas aplicações, desde montagem até manipulação de materiais.

Braços SCARA (Seleção de Articulações Robóticas para Aplicações de Montagem, 1978) (Figura 3): A categoria de robôs SCARA foi desenvolvida para aplicações de montagem de alta precisão e é caracterizada por seus movimentos rápidos e repetíveis. Esses robôs se tornaram populares em indústrias de eletrônicos e montagem.

Figura 3 – Braço SCARA (1978)

O desenvolvimento contínuo na área de robótica tem levado à criação de braços robóticos mais sofisticados, com sensores avançados, maior capacidade de carga, precisão e flexibilidade. Além disso, a integração de inteligência artificial e aprendizado de máquina tem permitido que esses robôs sejam mais autônomos e capazes de realizar tarefas complexas com maior eficiência (Chaves, 2022).

Artigo Científico

Robôs colaborativos, também conhecidos como cobots, são uma categoria de robôs projetados para trabalhar em colaboração direta com seres humanos em ambientes de trabalho. Eles desempenham um papel importante na automação industrial e em várias outras aplicações em que a interação homem-máquina é fundamental. Diferentemente dos robôs industriais tradicionais, que normalmente operam em áreas isoladas ou com barreiras de segurança, os cobots são projetados para compartilhar espaços de trabalho com seres humanos de forma segura. Eles incorporam sensores avançados e tecnologia de visão que lhes permite detectar a presença de pessoas e parar imediatamente se houver uma interação potencialmente perigosa.

Apesar de todos os benefícios encontrados, a implementação bem-sucedida de robôs colaborativos pode apresentar desafios, como a integração com sistemas existentes, a formação de pessoal e a garantia de que os processos sejam seguros e eficazes. A tendência é que a adoção de robôs colaborativos continue a crescer à medida que a tecnologia avança. Espera-se que eles desempenhem um papel cada vez mais importante na indústria 4.0 e na automação de processos em diversos setores.

Em resumo, os robôs colaborativos, bem como os braços robóticos que se enquadram nessa classe, representam uma evolução significativa na automação industrial e em muitas outras áreas, proporcionando uma colaboração segura e eficiente entre humanos e máquinas para melhorar a produtividade e a qualidade do trabalho. Estas automações proporcionam um avanço muito relevante. Em um processo de trabalho compartilhado, eles reduzem o desgaste físico e auxiliam o operador humano. (Vido, 2018). Diante do exposto, e da crescente evolução tecnológica é cada vez mais comum se deparar com atividades automatizadas por robôs.

OBJETIVOS E RELEVÂNCIA DO TRABALHO

Para a construir esse trabalho, inicialmente foi levado em consideração o grande uso de braços robóticos nas indústrias em geral, sendo estes utilizados para exercer diversas funções, desde robôs programados para tarefas simples, como erguer objetos massivos, até aqueles que são usados para tarefas complexas e precisas, tal qual cirurgias oculares. Também foi analisado o uso em áreas de risco ou inflamáveis que colocam em risco o operador.

No entanto, estas máquinas possuem um sistema complexo de programação, permitindo apenas a utilização por pessoas devidamente capacitadas para esses fins. Dessa forma, cogita-se a criação de um braço robótico de fácil programação, para que assim qualquer pessoa possa usá-lo de forma simples e ágil, sem a necessidade de um conhecimento grande na área, otimizando tarefas do dia-a-dia.

Esse cenário despertou um interesse significativo e motivador no desenvolvimento deste projeto, que tem como principal objetivo facilitar e tornar mais acessível a utilização de robôs em pequenas indústrias, estabelecimentos comerciais e educacionais, e automatizar tarefas repetitivas.

Para alcançar esses objetivos, tem-se as seguintes metas a cumprir:

- Projetar as peças em 3D através software Autodesk;
- Imprimir as peças 3D;
- Programar em C++ para Arduíno;
- Montar o protótipo;
- Testar o funcionamento e integração dos sistemas

DESENVOLVIMENTO DO PROJETO

Diante do exposto e da problemática encontrada, a proposta central foi criar um braço robótico de fácil programação, permitindo que os usuários automatizem uma variedade de tarefas repetitivas de forma simples e intuitiva. Inicialmente foram feitas algumas pesquisas sobre modelos e formas de programações para estes braços, e chegou-se à conclusão de criar 2 braços: o principal, que chamaremos de Braço 1, maior e que realizará efetivamente a tarefa para o qual foi programado, e um secundário, chamado então de Braço 2, que será semelhante ao Braço 1, porém em proporções menores, onde este segundo será usado para fazer a programação de movimentos. Assim sendo, com o Braço 2, o usuário gera o movimento e esse movimento é gravado para que o braço principal execute da mesma maneira a qual foi programado. Toda a carcaça de ambos braços foi feita em uma impressora 3D da ETEC Rosa Perrone Scavone e, para a parte de programação, o hardware utilizado fora o Arduino, usando a linguagem C++.

Somando o tempo de impressão de peças, montagem do protótipo e programação, foram utilizadas aproximadamente 6 meses.

Materiais Utilizados:

- Arduino UNO: Um microcontrolador que serve como a unidade de processamento central. (R$ 80,00)
- Adafruit PWM Servo Driver: Um controlador de servomotores que se comunica com o Arduino via I2C, permitindo o controle simultâneo de até 16 servomotores. (R$ 75,00)
- Potenciômetros: Utilizados para fornecer entradas analógicas que permitem o controle manual dos servomotores.
- Botões de Pressão: Utilizados para alternar entre modos de operação e para salvar estados específicos dos servomotores.
- LEDs: Utilizados como indicadores visuais do estado do sistema.
- Servomotores: Responsáveis pelo movimento do braço robótico. (R$ 220,00)
- Componentes de Conexão: Incluindo fios, protoboard, e resistores pull-up

internos para os botões.

Para confecção do projeto estima-se um custo total de R$ 700,00. Porém, para a confecção desse projeto, os itens de maior valor foram doados (Arduino, servo drive e servomotores) e os componentes de menor valor (botões, leds, fios, etc.) foram disponibilizados pela escola técnica.

RESULTADOS DO PROJETO

Após todas as pesquisas e análises do projeto, foi definido a ordem de impressão em 3D e início da montagem em uma base de MDF branco:

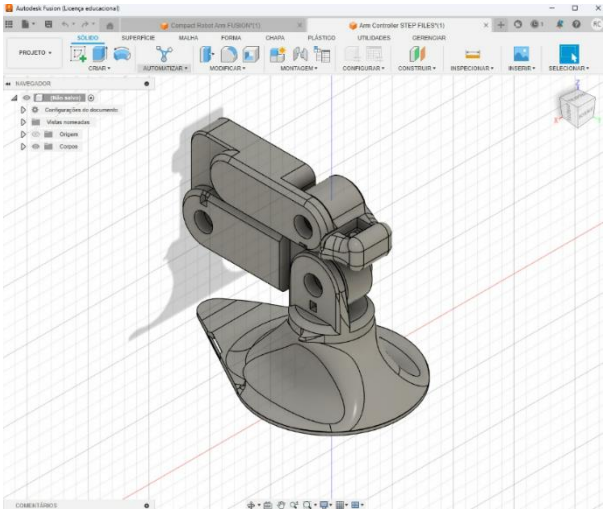

Figura 4 - Modelos 3D para impressão (fotografada pela autora) – março/2024

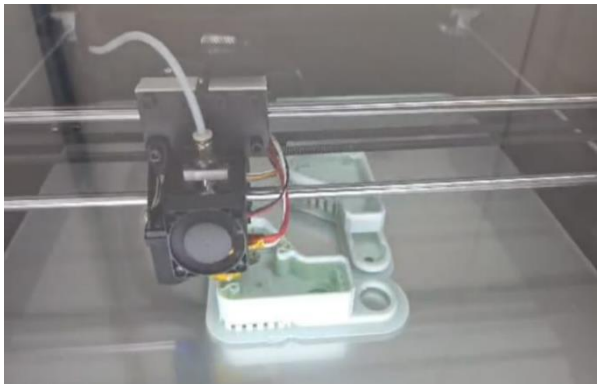

Figura 5 – Impressão em 3D na ETEC Rosa Perrone Scavone (fotografada pela autora) – abril/2024

Figura 6 – Peças impressas (fotografada pela autora) – abril/2024

Figura 7 – Procedimento de Montagem na ETEC (fotografada pela autora) – maio/2024

Programação e aplicação

Os potenciômetros são conectados aos pinos analógicos do Arduino para leitura dos valores de posição. Botões de pressão são conectados aos pinos digitais configurados com resistores pull-up internos para detectar pressões. O Adafruit PWM Servo Driver é conectado ao Arduino via I2C para controlar os servomotores.

O software é implementado em C++ e estruturado de forma modular para facilitar a compreensão e a manutenção. A seguir, uma visão geral das principais seções do código:

Artigo Científico

Definições de Parâmetros:

```
4   // Definições de intervalo de pulso para os servo motores
5   #define MIN_PULSE_WIDTH 650
6   #define MAX_PULSE_WIDTH 2350
7   #define FREQUENCY 50
```

Parâmetros como largura mínima e máxima de pulso e frequência de PWM são definidos para garantir movimentos precisos e dentro dos limites operacionais dos servos.

Inicialização do Driver PWM:

```
9   // Cria um objeto Adafruit_PWMServoDriver para controlar os servos
10  Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
```

Um objeto do Adafruit_PWMServoDriver é criado para gerenciar o controle dos servomotores via PWM.

Definição dos Pinos:

```
12  // Define os pinos dos potenciômetros, motores e botões
13  int potWrist = A3;
14  int potElbow = A2;
15  int potShoulder = A1;
16  int potBase = A0;
17  int potHandButton = 13;
18
19  int hand = 11;
20  int wrist = 12;
21  int elbow = 13;
22  int shoulder = 14;
23  int base = 15;
```

Os pinos do Arduino utilizados para ler os valores dos potenciômetros e controlar os servomotores são especificados.

Variáveis de Controle de Modo e Estado dos Botões:

```
25  // Variáveis para controle de modo e estado dos botões
26  boolean modoManual = true;
27  boolean modoBotaoEstadoAnterior = HIGH;
28  boolean salvarBotaoEstadoAnterior = HIGH;
```

Variáveis booleanas são usadas para alternar entre os modos manual e automático e para detectar mudanças de estado nos botões.

Artigo Científico

Definição dos Pinos do LED e Botões:

```
30  // Pino do LED
31  const int ledPin = 2;
32
33  // Pinos dos botões
34  const int modoBotaoPin = 12;
35  const int salvarBotaoPin = 11;
```

Os pinos utilizados para o LED indicador e os botões de modo e salvar são definidos.

Definição do Número Máximo de Passos e Matriz para Armazenar os Passos:

```
37  // Definição do número máximo de passos
38  #define MAX_PASSOS 100
39
40  // Matriz para armazenar os passos: 5 colunas (4 para os potenciômetros e 1 para o botão do hand)
41  int passos[MAX_PASSOS][5];
42  int contadorPassos = 0;
43  unsigned long tempoUltimoPasso = 0;
```

Um limite para o número de passos que podem ser armazenados é estabelecido, e uma matriz é definida para guardar esses passos.

Função setup:

```
45  void setup() {
46    // Inicialização do Arduino
47    delay(5000); // Atraso inicial de 5 segundos para permitir qualquer preparação manual
48    pwm.begin(); // Inicializa o driver do servo motor
49    pwm.setPWMFreq(FREQUENCY); // Define a frequência de PWM
50    pwm.setPWM(11, 0, 90); // Posição inicial do servo motor "hand"
51
52    // Configura os pinos dos botões e LEDs
53    pinMode(13, INPUT_PULLUP); // Configura o pino do botão do hand como entrada com pull-up
54    pinMode(ledPin, OUTPUT); // Configura o pino do LED como saída
55    pinMode(modoBotaoPin, INPUT_PULLUP); // Configura o pino do botão de modo como entrada com pull-up
56    pinMode(salvarBotaoPin, INPUT_PULLUP); // Configura o pino do botão de salvar como entrada com pull-up
57
58    Serial.begin(9600); // Inicializa a comunicação serial para monitoramento
59  }
```

Inicializa o driver PWM, configura os pinos e inicia a comunicação serial. Esta função garante que todos os componentes estejam prontos para uso quando o sistema é ligado.

Loop Principal:

O "Loop Principal" do código desempenha um papel fundamental no funcionamento do sistema de controle do braço robótico, coordenando as principais funcionalidades do programa de forma contínua enquanto o sistema está em operação. Este loop é responsável por gerenciar as entradas do usuário,

alternar entre os modos de operação (manual e automático), salvar e executar sequências de movimentos, e fornecer feedback visual sobre o estado do sistema.

```
61  void loop() {
62    // Leitura dos estados dos botões
63    int modoBotaoEstado = digitalRead(modoBotaoPin);
64    int salvarBotaoEstado = digitalRead(salvarBotaoPin);
65    int potHandButtonState = digitalRead(potHandButton);
```

Em primeiro lugar, são lidos os estados dos botões de modo e salvar, bem como o estado do botão do hand. Esta etapa é crucial para detectar as interações do usuário e determinar as ações a serem executadas pelo sistema.

```
67    // Verifica se ambos os botões de modo e salvar são pressionados simultaneamente
68    if (modoBotaoEstado == LOW && salvarBotaoEstado == LOW) {
69      resetPassos(); // Reseta a memória de passos
70      delay(500); // Pequeno atraso para evitar leituras múltiplas dos botões
71    }
72
73    // Verifica se houve uma mudança de estado no botão de modo
74    if (modoBotaoEstado == LOW && modoBotaoEstadoAnterior == HIGH) {
75      modoManual = !modoManual; // Alterna entre os modos manual e automático
76      Serial.print("Modo de operacao: ");
77      Serial.println(modoManual ? "Manual" : "Auto"); // Imprime o modo atual
78      delay(500); // Pequeno atraso para evitar leituras múltiplas do botão
79    }
80    modoBotaoEstadoAnterior = modoBotaoEstado;
```

Após a leitura dos estados dos botões, o sistema verifica se houve uma mudança de estado no botão de modo. Se o botão foi pressionado e o estado anterior era HIGH, o modo de operação é alternado entre manual e automático. Esta funcionalidade proporciona ao usuário flexibilidade para escolher o modo de controle mais adequado para a situação.

```
82    // Verifica se o botão de salvar foi pressionado
83    if (salvarBotaoEstado == LOW && salvarBotaoEstadoAnterior == HIGH) {
84      salvarPasso(); // Salva o passo atual
85      delay(500); // Pequeno atraso para evitar leituras múltiplas do botão
86    }
87    salvarBotaoEstadoAnterior = salvarBotaoEstado;
```

Em seguida, é verificado se o botão de salvar foi pressionado. Se ambas as condições de pressionamento do botão de salvar e estado anterior HIGH forem atendidas, o sistema chama a função salvarPasso() para salvar o estado atual dos potenciômetros e do botão do hand como um passo na matriz de passos. Este recurso permite ao usuário gravar sequências de movimentos personalizadas para execução posterior.

```
// Lógica para operação manual ou automática
if (modoManual) {
  // Modo manual
  moveMotor(potWrist, wrist);
  moveMotor(potElbow, elbow);
  moveMotor(potShoulder, shoulder);
  moveMotor(potBase, base);

  // Controla o hand com base no estado do botão
  int pushButton = digitalRead(13);
  if (pushButton == HIGH) {
    pwm.setPWM(hand, 0, 180); // Gira o hand para agarrar
  } else {
    pwm.setPWM(hand, 0, 90); // Gira o hand para soltar
  }
  digitalWrite(ledPin, HIGH); // Acende o LED
} else {
  // Modo automático
  static int passoAtual = 0; // Armazena o passo atual a ser executado
  if (contadorPassos > 0 && millis() - tempoUltimoPasso >= 2000) {
    // Move os motores para as posições do passo atual
    moveMotorAuto(passos[passoAtual][0], wrist);
    moveMotorAuto(passos[passoAtual][1], elbow);
    moveMotorAuto(passos[passoAtual][2], shoulder);
    moveMotorAuto(passos[passoAtual][3], base);

    // Controla o hand de acordo com o estado salvo no passo
    if (passos[passoAtual][4] == HIGH) {
      pwm.setPWM(hand, 0, 180); // Gira o hand para agarrar
    } else {
      pwm.setPWM(hand, 0, 90); // Gira o hand para soltar
    }

    // Imprime os valores do passo atual no monitor serial
    Serial.println("Passo executado:");
    Serial.println("Passo " + String(passoAtual + 1) + ":");
    for (int i = 0; i < 5; i++) {
      Serial.print("Potenciometro ");
      Serial.print(i);
      Serial.print(": ");
      Serial.println(passos[passoAtual][i]);
    }
```

```

    passoAtual++;
    tempoUltimoPasso = millis();

    if (passoAtual >= contadorPassos) {
      // Reinicia a execução dos passos
      passoAtual = 0;
    }
  }
```

Se o sistema estiver no modo automático e houver passos gravados na matriz, a execução automática dos movimentos é iniciada. A cada iteração do loop, o sistema verifica se o tempo decorrido desde o último passo é maior ou igual a 2 segundos. Se for, os servomotores são movidos para as posições

especificadas no próximo passo da matriz, e o estado do botão do hand é ajustado conforme gravado. Além disso, as informações do passo são impressas na serial para fornecer feedback ao usuário sobre os movimentos em execução.

```
140     piscarLED(500); // Faz o LED piscar a cada 500 ms
```

Por fim, a função piscarLED() é chamada para controlar o LED indicador. O LED pisca a cada 500 ms, fornecendo feedback visual sobre o estado do sistema. Este recurso é valioso para fornecer ao usuário uma indicação clara do funcionamento do sistema, mesmo em ambientes com pouca visibilidade.

Em resumo, o "Loop Principal" garante que o sistema responda de forma adequada às entradas do usuário, execute sequências de movimentos gravadas e forneça feedback visual enquanto estiver em operação. Sua estrutura modular e eficiente é crucial para o funcionamento correto e eficaz do sistema de controle do braço robótico.

Funções Auxiliares:

As funções auxiliares desempenham papéis cruciais na operação do sistema, facilitando o controle dos motores e a gestão dos passos gravados.

moveMotor(int controlIn, int motorOut):

```
144 // Função para mover um motor com base no valor do potenciômetro
145 void moveMotor(int controlIn, int motorOut) {
146   int pulse_wide, pulse_width, potVal;
147   potVal = analogRead(controlIn); // Lê o valor do potenciômetro
148   pulse_wide = map(potVal, 800, 240, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH); // Mapeia o valor do potenciômetro para o intervalo de pulso necessário
149   pulse_width = int(float(pulse_wide) / 1000000 * FREQUENCY * 4096); // Calcula o comprimento do pulso
150   pwm.setPWM(motorOut, 0, pulse_width); // Define o pulso para o motor
151 }
```

Esta função movimenta um servomotor com base no valor lido de um potenciômetro. O valor analógico lido do potenciômetro é mapeado para a largura do pulso que o servo deve receber.

Leitura do Potenciômetro: A função analogRead(controlIn) lê o valor do potenciômetro conectado ao pino especificado.

Mapeamento: O valor lido é mapeado para o intervalo de pulsos necessário usando a função map(potVal, 800, 240, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH).

Cálculo da Largura do Pulso: A largura do pulso é calculada e convertida para um valor que o driver PWM possa usar.

Controle do Servo: A função pwm.setPWM(motorOut, 0, pulse_width) é utilizada para definir o pulso no servo motor.

moveMotorAuto(int controlIn, int motorOut):

```
// Função para mover um motor automaticamente sem ler do potenciômetro
void moveMotorAuto(int controlIn, int motorOut) {
  int pulse_wide, pulse_width, potVal;
  potVal = (controlIn); // Usa diretamente o valor passado como parâmetro
  pulse_wide = map(potVal, 800, 240, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH); // Mapeia o valor para o intervalo de pulso necessário
  pulse_width = int(float(pulse_wide) / 1000000 * FREQUENCY * 4096); // Calcula o comprimento do pulso
  pwm.setPWM(motorOut, 0, pulse_width); // Define o pulso para o motor
}
```

Semelhante à função moveMotor, mas usa um valor predefinido em vez de ler um valor de um potenciômetro. Isto é útil para a execução automática de passos gravados.

Uso do Valor Predefinido: O valor passado como controlIn é mapeado para o intervalo de pulsos necessário.

Cálculo e Controle: A largura do pulso é calculada e o servo é controlado da mesma forma que na função moveMotor.

salvarPasso():

```
// Função para salvar o estado atual dos potenciômetros como um passo
void salvarPasso() {
  if (contadorPassos < MAX_PASSOS) {
    // Lê os valores dos potenciômetros
    passos[contadorPassos][0] = analogRead(potWrist);
    passos[contadorPassos][1] = analogRead(potElbow);
    passos[contadorPassos][2] = analogRead(potShoulder);
    passos[contadorPassos][3] = analogRead(potBase);
    passos[contadorPassos][4] = digitalRead(potHandButton); // Salva o estado do botão do hand
    contadorPassos++;

    // Imprime os valores salvos do passo atual no monitor serial
    Serial.println("Passo salvo com sucesso:");
    Serial.println("Passo " + String(contadorPassos) + ":");
    for (int i = 0; i < 5; i++) { // Loop para imprimir os valores salvos em cada passo
      Serial.print("Potenciometro ");
      Serial.print(i);
      Serial.print(": ");
      Serial.println(passos[contadorPassos - 1][i]); // Imprime o valor do potenciômetro na coluna i do passo atual
    }
  } else {
    Serial.println("Memoria de passos cheia. Nao foi possivel salvar o passo."); // Mensagem de erro se a memória de passos estiver cheia
  }
}
```

Esta função salva o estado atual dos potenciômetros e do botão do hand como um passo na matriz de passos. É crucial para a gravação de uma sequência de movimentos.

Leitura dos Potenciômetros: Os valores atuais dos potenciômetros são lidos e armazenados na matriz passos.

Estado do Botão: O estado do botão do hand é lido e armazenado.

Incremento do Contador: O contador de passos é incrementado.

Verificação da Capacidade: Se o número máximo de passos é atingido, uma mensagem de erro é exibida na serial.

resetPassos():

```
// Função para resetar a memória de passos
void resetPassos() {
  contadorPassos = 0; // Reinicia o contador de passos
  Serial.println("Memoria de passos resetada."); // Mensagem de confirmação
}
```

Esta função reseta a memória de passos quando os botões de modo e salvar são pressionados simultaneamente.

Reset do Contador: O contador de passos é resetado para 0.

Feedback Serial: Uma mensagem é enviada para a serial indicando que os passos foram resetados.

piscarLED(int intervalo):

```
// Função para fazer o LED piscar em um intervalo específico
void piscarLED(int intervalo) {
  static unsigned long previousMillis = 0;
  static boolean estadoLED = LOW;
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= intervalo) {
    estadoLED = !estadoLED;
    digitalWrite(ledPin, estadoLED);
    previousMillis = currentMillis;
  }
}
```

Esta função faz o LED piscar a um intervalo específico, fornecendo feedback visual sobre o estado do sistema.

Controle de Tempo: Utiliza a função millis() para medir o tempo decorrido e alternar o estado do LED.

Alternância do Estado do LED: O estado do LED é alternado entre ligado e desligado com base no intervalo especificado.

O sistema implementado permite o controle manual preciso do braço robótico através dos potenciômetros, bem como a gravação de uma sequência de movimentos que podem ser executados automaticamente. A memória de passos pode ser resetada pressionando simultaneamente os botões de modo e salvar, e o LED indica visualmente o estado do sistema.

Durante os testes, observou-se que o sistema responde de forma precisa aos comandos manuais e executa a sequência de passos gravados conforme o esperado. A implementação da memória de passos dinâmica garantiu que a capacidade de armazenamento não fosse limitada a um número fixo, permitindo maior flexibilidade. Além disso, a adição da funcionalidade de reset aumentou a usabilidade do sistema, facilitando a reconfiguração rápida dos movimentos gravados.

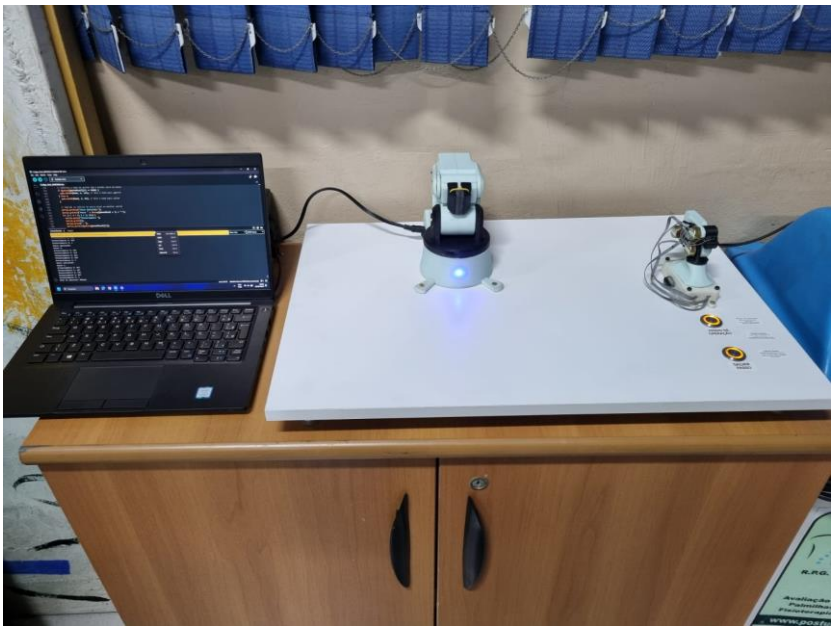

Figura 5 – Funcionamento do Braço colaborativo (fotografada pela autora) – junho/2024

Após o funcionamento em modo automático e manual, busca-se a realização de testes para determinar a real capacidade de carga, força, velocidade e outros parâmetros para melhorar sua performance robótica.

Espera-se como melhoria para o projeto, a implementação modular do código para facilitar futuras expansões e adaptações, permitindo a integração com sensores adicionais e interfaces de usuário mais sofisticadas.

CONCLUSÕES

O projeto demonstrou uma implementação bem-sucedida de controle de um braço robótico utilizando Arduino e o Adafruit PWM Servo Driver. A capacidade de alternar entre modos manual e automático, juntamente com a funcionalidade de gravação e execução de movimentos, torna o sistema versátil e aplicável em diversas situações. A implementação modular do código facilita futuras expansões e adaptações, permitindo a integração com sensores adicionais e interfaces de usuário mais sofisticadas.

Após a finalização desta produção, com todo o tempo utilizado, sendo aproximadamente 10 meses ao todo, desde pesquisas, entendimentos, problemáticas levantadas, decisões a serem tomadas e efetivamente, a produção e confecção do robô, conclui-se que, de fato, a indústria 4.0 tem muito a ganhar com a inclusão de Cobots, tal qual o que foi apresentado neste artigo. Apesar de ser um robô consideravelmente pequeno, se colocado em larga escala, poderá levantar mais do que o dobro do peso de seu programador, além de permitir uma amplitude de movimento a qual um ser humano, cuja estatura média, não permitiria e com mais segurança.

REFERÊNCIAS BIBLIOGRÁFICAS

ADAFRUIT, Industrie. PCA9685 PWM Servo Driver Library. Disponível em:https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library> acessado em fevereiro/2024.

BOECHAT, A. DA C. et al. Braço robótico manipulado através de movimentos humanos. Revista Interdisciplinar Pensamento Científico, v. 5, n. 4, 2019.

CHAVES, F. Desenvolvimento de um braço robótico acionado por sensor capacitivo fortaleza 2022. [s.l: s.n.]. Disponível em: https://repositorio.ufc.br/bitstream/riufc/65836/3/2022_tcc_fcxavier.pdf. Acessado em fevereiro/2024.

CHAVES, L. E. Robótica: O que é, histórico, tipos e aplicações. Disponível em: https://www.engenhariahibrida.com.br/post/robotica-o-que-e-historico-tipos-aplicacoes#:~:text=A%20s%C3%A9rie%20Unimate%201900%20se. Acessado em fevereiro/2024.

GOMES, L. S. Desenvolvimento e Implementação de um Sistema de Controle para Braço Robótico Manipulador. 2022. 86 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) - Universidade Federal de Minas Gerais, Belo Horizonte, 2022

MARTINS, P. C.; ALMEIDA, R. S. Robótica Industrial: Princípios e Aplicações. 3. ed. São Paulo: Editora ABC, 2023. 350 p.

NUNES, G. F.; SOUZA, M. A. Dispositivo de Fixação para Braços Robóticos. BR Patent 10203004000, 20 jan. 2023.

OLIVEIRA, J. M.; PEREIRA, F. R.; SANTOS, A. L. Um Estudo sobre Sensores de Posicionamento para Aplicações em Braços Robóticos. In: Congresso Brasileiro de Engenharia Mecânica, 25., 2023, Rio de Janeiro. Anais... Rio de Janeiro: ABCM, 2023. p. 120-130

SILVA, A. B.; SANTOS, C. D.; OLIVEIRA, E. F. Desenvolvimento de um Braço Robótico para Aplicações Industriais. Revista Brasileira de Engenharia Mecânica, São Paulo, v. 10, n. 2, p. 45-58, abr./jun. 2023.

SIMPLÍCIO, P. V. G.; LIMA, B. R. Manipuladores Robóticos Industriais. Caderno de Graduação - Ciências Exatas e Tecnológicas - UNIT - SERGIPE, v. 3, n. 3, p.

85–85, 24 out. 2016.

SANTANA, T. A.; ALMEIDA, H. M. Introdução à Programação de Braços Robóticos. In: Portal de Robótica Aplicada

SIQUEIRA-BATISTA, R. et al. Robotic surgery: bioethical aspects. Abcd. Arquivos Brasileiros de Cirurgia Digestiva (São Paulo), v. 29, n. 4, p. 287–290, dez. 2016.

SOARES, R.; LUCATO, A. V. R. Robótica colaborativa na indústria 4.0, sua importância e desafio. Revista Interface Tecnológica, v. 18, n. 2, p. 747–759, 20 dez. 2021.

VIDO, M.; LUCATO, W.; MARTENS, M. O robô colaborativo na indústria 4.0: conceitos para a interação humano-robô em um posto de trabalho. [s.l: s.n.]. Disponível em: https://abepro.org.br/biblioteca/TN_STO_290_1634_37074.pdf. Acessado em fevereiro/2024.

VÍDEO:

Build Some Stuff. How I Built a 3D Printed Robot Arm from Scratch (Arduino Based). YouTube, 23 de abr. de 2023. Disponível em: < https://www.youtube.com/watch?v=5toNqaGsGYs&ab_channel=BuildSomeStuf f>. Acesso em: janeiro de 2024.