UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA
CÂMPUS DE ILHA SOLTEIRA

Gabriel Shiguematsu Ogawa

APLICATIVO NA LINGUAGEM PYTHON PARA CONTROLAR UM MANIPULADOR ROBÓTICO POR SMARTPHONE

Ilha Solteira
2025

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

Gabriel Shiguematsu Ogawa

APLICATIVO NA LINGUAGEM PYTHON PARA CONTROLAR UM MANIPULADOR ROBÓTICO POR SMARTPHONE

Trabalho de Graduação apresentado à Faculdade de Engenharia de Ilha Solteira – Unesp como parte dos requisitos para obtenção do título de **Engenheiro Eletricista**

Nome do orientador
Profa Dra. Suely Cunha Amaro Mantovani

Ilha Solteira
2025

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

O34a

Ogawa, Gabriel Shiguematsu.
Aplicativo na linguagem Python para controlar um manipulador robótico por smartphone / Gabriel Shiguematsu Ogawa. -- Ilha Solteira: [s.n.], 2025
71 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, Ilha Solteira, 2025

Orientador: Suely Cunha Amaro Mantovani

Inclui bibliografia

1. Manipulador robótico. 2. Graus de liberdade. 3. Linguagem Python. 4. Cinemática direta. 5. Linguagem Flutter. 6. Aplicativo Android.

Sandra Montibeller - CRB-8/060

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos vinte quatro dias do mês de junho do ano de dois mil e vinte e cinco, o discente **Gabriel Shiguematsu Ogawa**, matriculado sob o nº **181052271**, tendo como banca examinadora a sua orientadora, *a* Profa. Dra. Suely Cunha Amaro Mantovani, o Prof. Dr. Marcelo Augusto Assunção Sanches e a Dra. Stephany de Souza Lyra, apresentou o Trabalho de Graduação intitulado "**APLICATIVO NA LINGUAGEM PYTHON PARA CONTROLAR UM MANIPULADOR ROBÓTICO POR SMARTPHONE**", obtendo a nota 9,0 (nove) e conceito Aprovado.

Profa. Dra. Suely Cunha Amaro Mantovani
- Orientadora -

Gabriel Shiguematsu Ogawa
- Discente -

Prof. Dr. Marcelo Augusto Assunção Sanches
- Membro da Banca -

Dra. Stephany de Souza Lyra
- Membro da Banca -

Faculdade de Engenharia – Departamento de Engenharia Elétrica
Avenida Professor José Carlos Rossi, 1370 - CEP 15.385-000, Ilha Solteira - SP - Brasil
Tel (18) 37431150 - Fax (18) 3742-2735 - www.feis.unesp.br

DEDICATÓRIA

Gostaria de dedicar este trabalho aos amigos, familiares, professores e a minha orientadora pela paciência comigo em momentos difíceis, e por estarem comigo em toda a minha graduação.

AGRADECIMENTOS

Gostaria de agradecer de coração a todos que participaram desta jornada que tive durante a graduação. Agradeço aos amigos feitos enquanto que tive o prazer de conhecer durante os anos iniciais de faculdade. Agradeço a vocês por me ensinarem sobre o mestrado e doutorado de vocês, por compartilharem de momentos bons e ruins que tivemos e por sempre abrirem um tempo na agenda de vocês para conversarem comigo quando me encontravam no prédio central.

Também agradeço de coração e alma a minha família, pois sempre me ajudaram em momentos de minha depressão e ansiedade generalizada. Sem vocês com toda certeza não teria chegado até aqui e não me tornaria a pessoa que sou hoje.

Agradeço também aos meus amigos da república em que morei, por sempre estarem comigo em momentos bons, ruins e também por sempre me chamarem para sair quando eu não queria. Vocês me ensinaram que apenas uma conversa na cozinha era o suficiente para melhorar o meu dia. Agradeço as dicas, pelas risadas, pelos momentos que tivemos. Sem vocês, com toda certeza, este trabalho de graduação não teria acontecido.

E finalmente, agradeço a minha orientadora Profa. Suely Cunha Amaro Mantovani pelas risadas, pelo carinho, pelas broncas e também pela paciência. Todo ensinamento dado, foi com toda certeza, uma motivação que tive para continuar. Obrigado por me aceitar como orientado e também por ter me ajudado durante a graduação.

“Quanto mais você sua no treinamento, menos sangra no campo de batalha”. George S. Patton

RESUMO

Tem-se conhecimento da importância dos robôs para o mundo moderno que têm sido utilizado nas diversas áreas, incluindo as indústrias, apresentando todos os recursos necessários de sensores, armazenamento, automação para se conectar com outros robôs e máquinas, de forma remota. Assim, neste trabalho foi realizado um aplicativo na linguagem Python para controlar um manipulador robótico por smartphone remotamente. Para isso é utilizada a plataforma de desenvolvimento Arduino UNO R3 , a conexão wireless via *Bluetooth*, tecnologia de transmissão de dados sem fio, destinada a pequenas distâncias e que permite a troca de dados entre dispositivos por radiofrequência e o manipulador robótico disponível no laboratório de sistemas digitais que possui cinco graus de liberdade, dados por sete servomotores. O aplicativo foi desenvolvido utilizando a linguagem de programação Python no software VS Code, a extensão Flutter e o software Android Studio para a emulação do aplicativo no computador. A versão do aplicativo criado apresenta uma tela de página inicial, informações e tarefas para o manipulador; composto de 4 botões e uma aba lateral para navegação no aplicativo. A programação para dar movimento ao manipulador robótico foi feita utilizado o software Arduino IDE e a biblioteca *VarSpeedServo* para o controle da velocidade dos servomotores. Foram feitos testes de posicionamento no manipulador para vários ângulos, sob o comando dos botões do Aplicativo e nos resultados do comando pelo aplicativo do manipulador, são demonstrados boa repetibilidade e precisão, mas pouca acurácia que é atribuída a estrutura mecânica imprecisa do manipulador.

Palavras-chave: Manipulador robótico; Graus de liberdade; Linguagem Python; Cinemática direta; Linguagem Flutter; Aplicativo Android; Comunicação sem fio.

ABSTRACT

We know the importance of robots for the modern world and they have been used in various areas, including industries, presenting all the necessary resources of sensors, storage, automation to connect with other robots and machines, remotely. Thus, in this work an application was created in the Python language to control a robotic manipulator via smartphone remotely. For this purpose, the Arduino UNO R3 development platform and the wireless connection via *Bluetooth* are used, a wireless data transmission technology designed for short distances and which allows data exchange between devices via radio frequency and the robotic manipulator available in the digital systems laboratory has five degrees of freedom, given by seven servomotors. The application was developed using the Python programming language in the VS Code software, the Flutter extension and the Android Studio software to emulate the application on the computer. The application version created presents a home screen, information and tasks for the manipulator; composed of 4 buttons and a side tab for navigation in the application. The programming to give movement to the robotic manipulator was done using the Arduino IDE software and the VarSpeedServo library to control the speed of the servomotors. Positioning tests were performed on the manipulator for various angles, under the command of the Application buttons and in the command results of by manipulator's application , a good repeatability and precision are demonstrated, but little accuracy which is attributed to the imprecise mechanical structure of the manipulator.

Keywords: Robotic manipulator; Python language; Degrees of freedom; Forward kinematics; Flutter language; Android application; Wireless communication.

LISTA DE FIGURAS

LISTA DE QUADROS E TABELAS

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, vem sendo observada a evolução dos robôs devido ao grande avanço tecnológico, com o surgimento de plataformas de controle, sensores miniaturizados e atuadores que levam à redução dos preços de um robô e uma significativa melhoria em seu desempenho. Por isso, os robôs têm sido aplicados nos mais diversos setores, como medicina, em serviços em geral, fábricas de carros e dispositivos, além do entretenimento e em segurança, entre outros.

Também é crescente o grande interesse dos educadores do ensino básico, médio e cursos superiores, pelo domínio da tecnologia sobre os robôs, programação, automação e tudo que permeia o tema, usando a robótica como motivação para o desenvolvimento de seus estudantes.

O interesse das indústrias nos manipuladores robóticos são em função de características como o aumento na produção, padronização de produtos, possuírem força mecânica e precisão, apresentarem boa repetitividade e menos suscetíveis às falhas, se instalados e programados corretamente A maioria dos robôs utilizados, atualmente, nas indústrias é dito de segunda geração, apresentam inteligência para se conectar com outros robôs e máquinas, para armazenar programas e se comunicar com outros sistemas computacionais (CARRARA, 2015). Agregando aos conhecimentos da robótica têm-se outras tecnologias como o controle feito pelos smartphones e os seus aplicativos, usando a voz, importantes para a automação dos processos (GREGOLETO, 2019).

Um robô é definido como um dispositivo eletromecânico, podendo ser controlado de forma automática ou manual, e programável. São colocados para trabalhar em qualquer atividade que representem riscos ao ser humano (CRAIG, 2012).

Na ficção cientifica e cinematográfica, os robôs são representados por figuras quase humanas ou com semelhanças a um humano. Na realidade contemporânea os robôs são apenas um conjunto de peças e circuitos eletrônicos que fazem uma tarefa repetitiva ou pré-programada, vistos nas indústrias os braços robóticos (ou manipuladores), os robôs aspiradores em residências, etc.

A robótica abrange as mais diversas áreas da engenharia, incluindo a mecânica, elétrica, matemática e programação. Estas áreas fornecem o

equacionamento necessário para a construção e orientação do movimento de cada robô e como o robô deve operar, utilizando as teorias de cinemática direta e inversa de manipuladores (CRAIG, 2012).

Consistem de uma sequência de elos que podem ser rígidos ou flexíveis, interconectados por meio de articulações chamadas de juntas; são compostos por um braço que assegura mobilidade, um pulso e um efetuador (*end-effector)*, ou garra, que pode ser acoplada a uma determinada ferramenta para realizar a tarefa desejada pelo robô. Cada junta apresentada na estrutura mecânica do manipulador contribui, normalmente, com um (1) grau de liberdade (GDL), ou em inglês, *Degrees of Freedom* (DOF) (NIKU, 2010).

Trata-se este projeto do controle de um manipulador robótico comandado por um aplicativo desenvolvido para smartphone na plataforma Android e usando a tecnologia *Bluetooth* do *smatphone*. O manipulador tem sete servomotores e conta com cinco graus de liberdade montado por Nunes (2016), sendo possível pegar objetos e transportá-los para outra posição no seu volume de trabalho.

1.1 Objetivos

O objetivo do trabalho é o desenvolvimento de um aplicativo para *smartphone* Android visando comandar um manipulador robótico de cinco graus de liberdade, que por transmissão sem fio, se comunica com um módulo *Bluetooth* conectado ao Arduino, que por sua vez comanda os servomotores do manipulador robótico. A linguagem Python é usada para a criação do aplicativo, com o auxílio dos *softwares* VS Code, Flutter e o software Android Studio.

1.2 Organização do texto

Depois desta introdução, tem-se no capítulo 2 a revisão de literatura que fornece os conhecimentos necessários para o desenvolvimento deste trabalho e a escolha dos seus componentes .

No Capítulo 3 são abordadas as principais características de um robô manipulador também chamado braço robótico, tais como, os componentes, a construção e os conceitos envolvidos entre esses, elos, juntas, os tipos de juntas, grau de liberdade, efetuador e volume de trabalho. Também a cinemática direta que

estuda o posicionamento e a movimentação de corpos, sem ponderar as forças e os momentos presentes.

Apresentam-se no capítulo 4 o hardware e o software utilizados na pesquisa, entre esses uma breve introdução sobre o Arduino e a sua IDE , o módulo *Bluetooth* e servomotores, utilizados no protótipo e também a linguagem Python e o software VS Code.

São abordados no capítulo 5 o desenvolvimento do aplicativo e os testes com o aplicativo e o manipulador.

No capítulo 6 apesentam-se os resultados e discussões. Tem-se a conclusão no capítulo 7 em seguida vem as referências bibliográficas, os apêndices e anexos.

2 REVISÃO DE LITERATURA

Nesta revisão de literatura descrevem-se alguns dos trabalhos que proporcionaram os conhecimentos necessários, tais como, os componentes de hardware e software para o desenvolvimento desta pesquisa.

Ferreira e Alves (2013) descrevem em seu trabalho de conclusão de curso, o projeto de um braço robótico controlado remotamente, via smartphone com comandos seriais enviados por interface *Bluetooth*. O braço tem sua estrutura mecânica feita em alumínio com peças elaboradas no AutoCAD, acopladas a seis servomotores que possibilitam o seu movimento. Foi usado no projeto, além dos servomotores o kit de desenvolvimento *LaunchPad* MSP430, uma plataforma controladora. O braço robótico é comandado por meio de um aplicativo para dispositivos na plataforma Android que foi criado para enviar, via Bluetooth, um pacote de números de 0 a 180^0 referentes ao ângulo de rotação de cada servo. Estes números são controlados por meio de um slide bar que permite ao usuário um controle proporcional de movimento. O microcontrolador MSP430 recebe os dados do aplicativo, traduzindo em um sinal PWM (*Pulse Width Modulation* – do inglês: Modulação por largura de pulso) que aciona cada servo, posicionando seus eixos no ângulo desejado. Para suprir o consumo de corrente dos servomotores foi utilizada uma fonte de alimentação externa. Concluem que os objetivos foram alcançados com o funcionamento do Aplicativo de forma satisfatória.

É descrito no trabalho de conclusão de curso de Barbosa (2016), o projeto de controle de um braço articulado comandado por um smartphone da plataforma Android via *Bluetooth*, aplicativo desenvolvido usando o programa *app inventor2*. A estrutura do braço é feita em MDF e composta por 4 servomotores que fazem sua movimentação. O aplicativo foi elaborado para transmitir usando o Bluetooth e o Arduino UNO, o sinal com pacotes de números que correspondem ao ângulo de rotação de cada servo, todo esse movimento devidamente controlado e monitorado. Obteve resultados satisfatórios quanto à movimentação do braço e quanto à comunicação entre dispositivo e *Bluetooth*. Com este projeto o autor tinha como objetivo auxiliar na orientação dos alunos de engenharia mecânica nas aulas de mecatrônica, automação industrial e popularizar a robótica dentro da engenharia.

Em De Almeida et al. (2022), descrevem a construção de um braço articulado com dois graus de liberdade, cujos movimentos são controlados remotamente via *Bluetooth*, por meio de um smartphone Android. Para isso, utilizam componentes eletrônicos como, Arduino UNO R3, servomotores Tower Pro SG90s 9 G e o módulo *Bluetooth*. HC-05, de baixo custo e comuns em projetos acadêmicos. No trabalho os autores relatam que conseguiram estabelecer a comunicação serial entre o smartphone e a placa Arduino, que por sua vez está conectada ao braço robótico portanto, pelo smartphone são enviados os comandos de movimento aos motores, remotamente, por meio de um aplicativo Android desenvolvido por eles na plataforma *MIT App Inventor*. O aplicativo possui uma interface gráfica característica, apresentando quatro botões para as tarefas, onde cada um dos motores é interpretado por duas barras deslizantes que se movimentam horizontalmente, em um intervalo de 0 a 180°, limite de rotação do servomotor. Ao final, foram realizados testes de bancada no protótipo, em que todas as funcionalidades foram atendidas.

Com essa revisão de literatura pôde-se entender melhor todo o processo de comando do braço robótico e definir os materiais para uso neste projeto de pesquisa, principalmente o hardware.

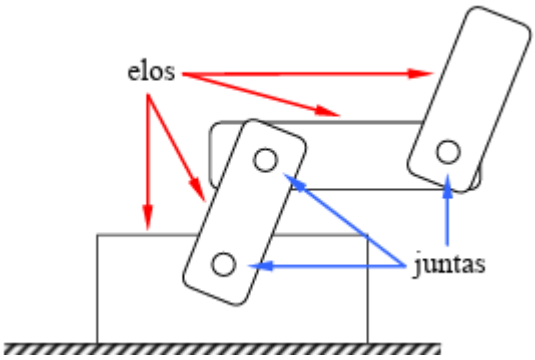
3 MANIPULADORES ROBÓTICOS

Neste capitulo são abordadas as principais características de um robô manipulador, e seus detalhes construtivos, componentes, arquitetura e os conceitos envolvidos, entre esses, elos, juntas, os tipos de juntas, grau de liberdade, efetuador e volume de trabalho. Em seguida descreve-se brevemente a cinemática direta de forma a dar os subsídios necessários para o entendimento do manipulador e a sua aplicação no projeto.

3.1 Anatomia dos manipuladores

Os manipuladores robóticos possuem uma anatomia que se assemelha a um braço humano, deste modo consiste de elementos denominados **elos** unidos por **juntas** de movimento relativo, onde são acoplados os acionadores para realizar estes movimentos individualmente, conforme mostrado na Figura 1.

Figura 1 – Ilustração dos elos e juntas de um manipulador.



Fonte: Carrara (2015).

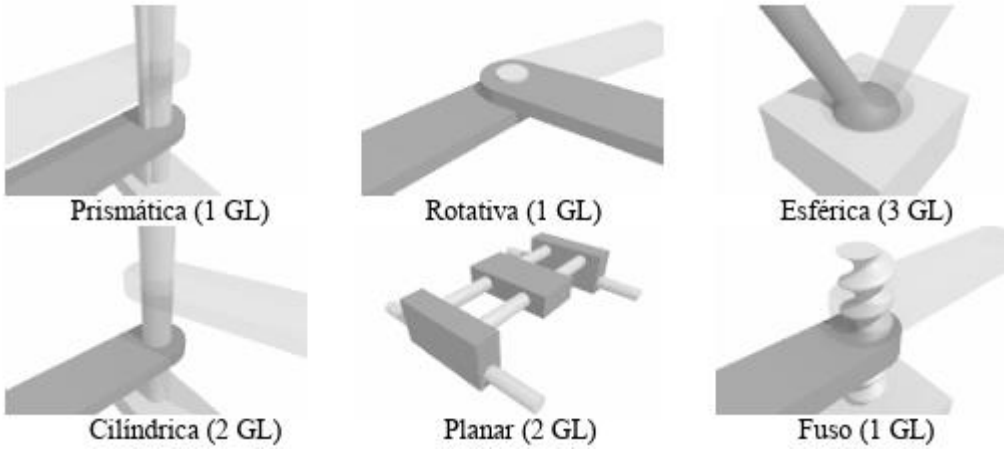
As **juntas** permitem calcular os graus de liberdade de um manipulador, podendo ser do tipo prismática ou linear, rotativa, esférica, cilíndrica, planar e fuso (CARRARA, 2015):

- A **prismática** ou **linear** move-se em linha reta, são compostas de duas hastes que deslizam entre si;
- A junta **rotacional** gira em torno de uma linha imaginária estacionária chamada de eixo de rotação (gira como uma cadeira giratória e abrindo e fechando como uma dobradiça) define um grau de liberdade ;

- A junta **esférica** funciona com a combinação de três juntas de rotação, e permite rotações em torno de três eixos distintos, por isso tem três graus de liberdade (dificuldade de acionamento e construção) ;
- A junta **cilíndrica** é composta por duas juntas, uma rotacional e uma prismática, define dois graus de liberdade;
- A junta **planar** é composta por duas juntas prismáticas, e realiza movimentos em duas direções, definindo dois graus de liberdade;
- A junta **parafuso** (ou tipo **fuso)** é constituída de um parafuso e uma rosca que executa um movimento semelhante ao da junta prismática, porém, com movimento de rotação no eixo central (movimento do parafuso), definindo um grau de liberdade.

Na Figura 2 tem-se uma ilustração dos tipos de juntas encontradas normalmente em manipuladores robóticos.

Figura 2 – Tipos de juntas encontradas em manipuladores.



Fonte: Carrara (2015).

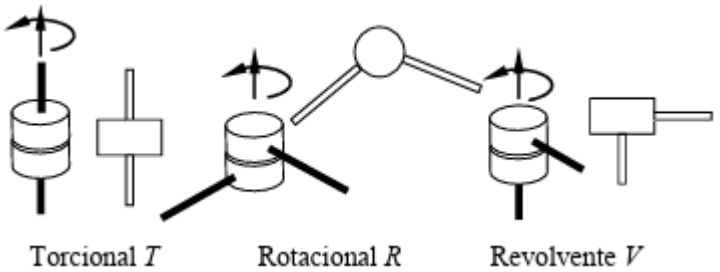
As juntas do tipo **Rotativas** são caracterizadas por permitirem que os elos girem em torno de um eixo, entre essas têm-se:

- **Torção ou torcional** (T), os elos de entrada e de saída tem a mesma direção do eixo de rotação da junta;
- **Rotacional** (R), os elos de entrada e de saída são perpendiculares ao eixo de rotação da junta;

- **Revolvente (V),** o elo de entrada possue a mesma direção do eixo de rotação, porém o elo de saída é perpendicular a este. As juntas rotativas acrescentam um grau de liberdade ao manipulador.

As juntas do tipo rotativas são mostradas na Figura 3.

Figura 3 – Tipos de juntas rotativas.

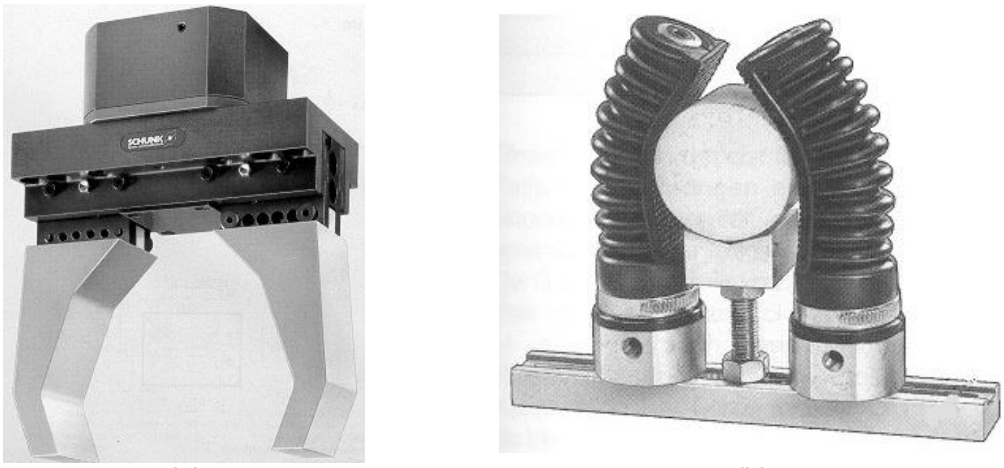


Fonte: Carrara (2015).

Os **graus de liberdade (GDL)**, ou graus de movimentação de um robô determinam os movimentos do manipulador no espaço bidimensional ou tridimensional. Cada junta define um,dois ou mais graus de liberdade, e, assim, o número de graus de liberdade do robô é igual à somatória dos graus de liberdade de suas juntas. Quanto maior o grau de liberdade de um manipulador, mais complexo será a matemática da cinemática direta e o controle do manipulador, porém mais precisa será a sua movimentação (CARRARA, 2015). O manipulador utilizado neste projeto de pesquisa possui cinco graus de liberdade, sendo um destes graus de liberdade, o do pulso, giratório.

Outro conceito importante é o **efetuador,** elemento que liga o robô ao meio externo, ou seja, a ferramenta que está posicionada na extremidade livre do manipulador e é responsável diretamente, pela realização das atividades ou tarefas. Este pode ser uma garra, uma pistola de solda, ventosa, eletroímã, entre outros. Uma garra semelhante a mão humana é a mais versátil como órgão terminal, podendo segurar objetos das mais diversas formas e tamanhos, com a quantidade mínima de força (CRAIG, 2012). Apesar dessa vantagem, o uso dessa garra aumenta a complexidade do controle e da construção, na Figura 4 são mostrados os tipos de garra mais comuns.

Figura 4 – Garras de dois dedos (a) Dedos rígidos. (b) Dedos flexíveis.

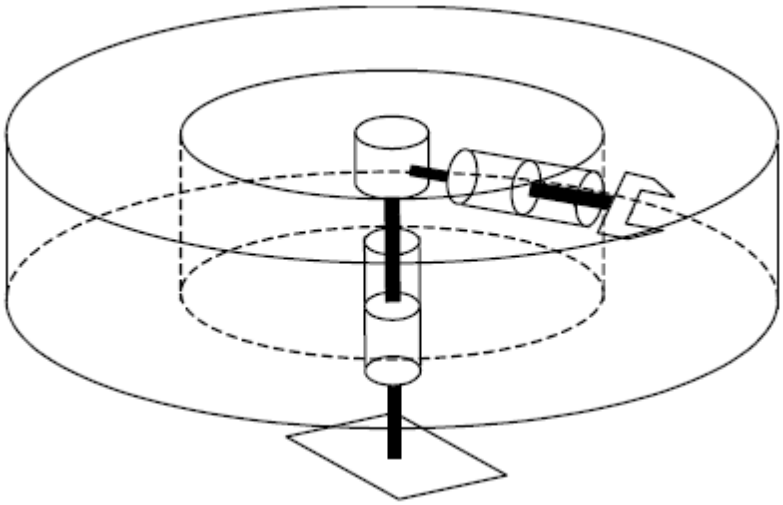


(a) (b)

Fonte: Nunes (2016).

Quando se estuda um manipulador robótico, outro conceito importante é o seu **volume de trabalho ,** pois defini a área de trabalho do manipulador ou conjunto de todos os pontos do espaço que o punho de um robô pode alcançar, sendo seu limite determinado pelo projeto estrutural do braço, pelo limite de movimento das juntas e o tamanho dos componentes do corpo, braço e punho. Convenciona-se utilizar o punho para que não haja interferência dos diferentes tamanhos de efetuadores/garras que possam ser acopladas ao punho do robô. Mostra-se na Figura 5, um exemplo do volume de trabalho de um braço robótico articulado verticalmente, onde se observa que seu volume real é diferente de seu volume de trabalho teórico (CARRARA, 2015).

Figura 5 – Volume de trabalho.



Fonte: Carrara (2015).

3.2 Cinemática dos Manipuladores

De acordo com Craig (2012), a cinemática é um ramo da mecânica que estuda o posicionamento e a movimentação de corpos, sem ponderar as forças e os momentos presentes. Na robótica é possível, a partir do estudo cinemático e das características geométricas de cada elo do manipulador, relacionar o sistema de referência da ferramenta do manipulador com o sistema de referência fixo do espaço em que este está inserido.

Existem duas formas de tratar a cinemática de manipuladores, pela cinemática direta ou cinemática inversa. Conhecendo as variáveis de cada junta e os parâmetros dos elos, a partir da cinemática direta é possível determinar a posição da ferramenta em um sistema de coordenadas cartesianas. Enquanto que, na cinemática inversa, sabendo-se a posição da ferramenta e os parâmetros dos elos é possível determinar uma configuração das variáveis das juntas, que satisfaça o correto posicionamento do manipulador. Neste trabalho estuda-se somente a cinemática direta e o método geométrico para a análise da cinemática direta.

3.2.1 Representação de pontos e vetores no espaço

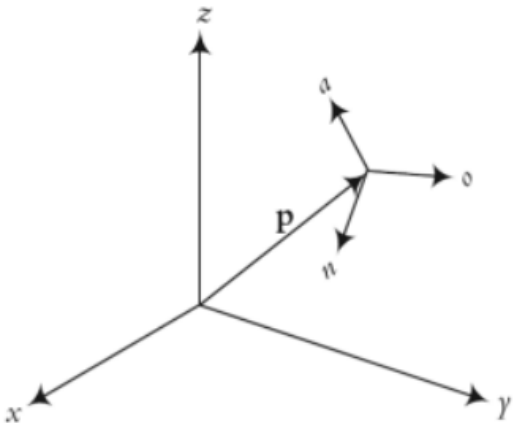
Antes de abordar os conceitos da cinemática direta é necessário o embasamento sobre o sistema de posicionamento e transformações elementares.

Com base na origem de um sistema de referência é possível localizar a posição de um ponto P no espaço tridimensional, mediante a um vetor de três elementos, referentes as coordenadas x, y e z, dado na forma matricial pela Matriz (1),

$$\mathrm{P} = \begin{bmatrix} \mathrm{P_x} \\ \mathrm{P_y} \\ \mathrm{P_z} \end{bmatrix} \tag{1}$$

Com base na Figura 6, segundo Niku (2013), a orientação de um referencial no espaço é representada pela relação de coeficientes angulares (n, o, a) sendo n (normal), o (orientação) e a (abordagem), os três elementos de coordenadas cartesianas de um referencial qualquer (fixo ou não), dado pela matriz R.

Figura 6 – Ilustração para um referencial fixo.

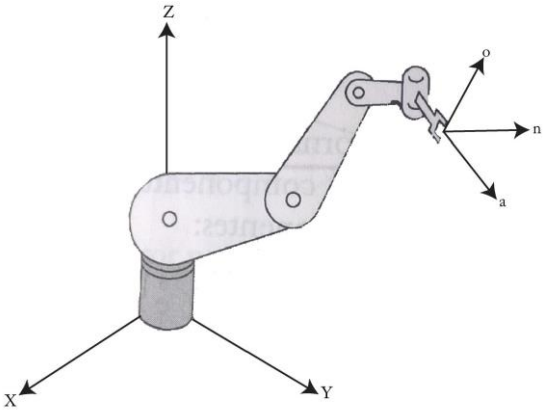


Fonte: Niku (2013).

$$R = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \quad (2)$$

Sendo que a **abordagem** é realizada ao longo do eixo z da pinça, a **orientação** se dá em torno do eixo y e a **normal** entre estes, está relacionada ao eixo x, conforme o manipulador apresentado na Figura 7.

Figura 7 – Eixos, normal, orientação e abordagem (para a garra do manipulador).



Fonte: Niku (2013).

Para representar um referencial em relação a outro de forma completa, devem ser especificados a localização da sua origem e as direções de seus eixos. Isto é possível ao se adicionar o vetor de posição representado pela Matriz (1), aos elementos da Matriz (2). Uma quarta linha com quatros elementos que representam fatores de perspectiva e escala é adicionada, completando a matriz homogênea 4x4,

F. Na matriz F, representada em (3), o referencial pode ser expresso por três vetores que descrevem seus vetores direcionais unitários e o quarto vetor descreve a sua direção. O elemento ‘1’, da última posição, indica que o vetor de posição está com fator de escala unitário, uma vez que seu comprimento é importante (NIKU, 2010).

$$F = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

3.2.2 Análise Geométrica para a Cinemática direta dos manipuladores

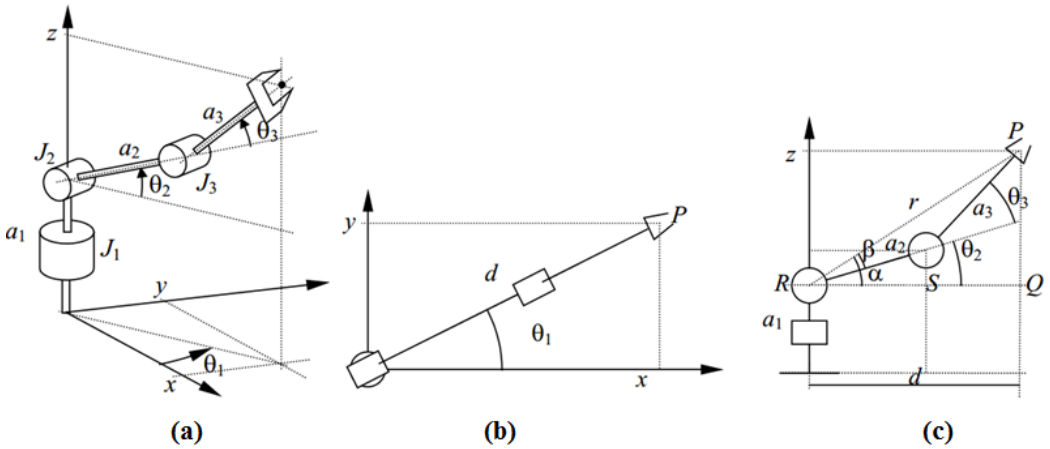
Na cinemática direta deve-se calcular a posição cartesiana no espaço, bem como a orientação do punho, com base no conhecimento dos ângulos das juntas, (CARRARA, 2015). Requer também o conhecimento do comprimento dos elos com precisão adequada. A resolução de problemas que tratam da cinemática direta de manipuladores é considerada simples se comparada ao da forma inversa. Existem diversas maneiras de realizar essa análise, entre elas estão: o modelamento geométrico, vetores locais e a sistemática de Denavit–Hartenberg (ou parâmetros de D-H) principalmente usado para mais de 4GDL (ROSÁRIO, 2005).

Na modelagem de um robô por meio de D-H obtém-se a posição e a orientação finais do elemento terminal. Por outro lado, utilizando vetores locais obtêm-se as posições e as orientações de diversos pontos de interesse que podem ser utilizados na construção gráfica do robô por meio de elementos primitivos.

3.2.3 Método geométrico para calcular a cinemática direta

Apresenta-se neste item, o método da análise geométrica para determinar a cinemática direta de um manipulador robótico de 3 GDL como exemplo, conforme configuração com todos os seus parâmetros (NUNES, 2016), dado na Figura 8.

Figura 8 – Manipulador com 3 GDL, vistas: (a) Espacial.(b) Superior. (c) Lateral.



Fonte: Nunes (2016).

A junta J1 gira em torno do eixo z na Figura, enquanto as juntas J2 e J3 giram em torno do plano x, y; a distância do ponto *P* em termos do plano x, y é representada por *d* na Figura 7(b), que pode ser determinada por meio da vista lateral, Figura 7 (c), e conforme a Equação (4),

$$d = a_2 \cdot \cos(\theta_2) + a_3 \cdot \cos(\theta_2 + \theta_3) \tag{4}$$

Com a equação de *d* , pode-se obter por trigonometria, as coordenadas do ponto *P* que indicam a posição da ferramenta terminal do manipulador robótico, para isso usa-se as Figuras 7 (a) e (b), obtendo as Equações (5), (6) e (7).

$$x = d \cdot \cos(\theta_1) = (a_2 \cdot \cos(\theta_2) + a_3 \cdot \cos(\theta_2 + \theta_3)) \cdot \cos(\theta_1) \tag{5}$$

$$y = d \cdot \text{sen}(\theta_1) = (a_2 \cdot \cos(\theta_2) + a_3 \cdot \cos(\theta_2 + \theta_3)) \cdot \text{sen}(\theta_1) \tag{6}$$

$$z = a_1 + a_2 \cdot sen(\theta_2) + a_3 \cdot sen(\theta_2 + \theta_3) \tag{7}$$

Para manipuladores com mais de 3 GDL a cinemática direta obtida geometricamente se torna mais complexa, nesses casos, normalmente é recomendado o uso da solução por meio da sistemática de D-H que é uma ferramenta utilizada como um padrão de representação de manipuladores robóticos.

Para resolver a cinemática direta de manipuladores robóticos de n graus de liberdade na sistemática de D-H, as equações de movimentos são construídas a

partir dos dados dos elos e juntas, independentemente da complexidade da configuração do robô (NIKU, 2013).

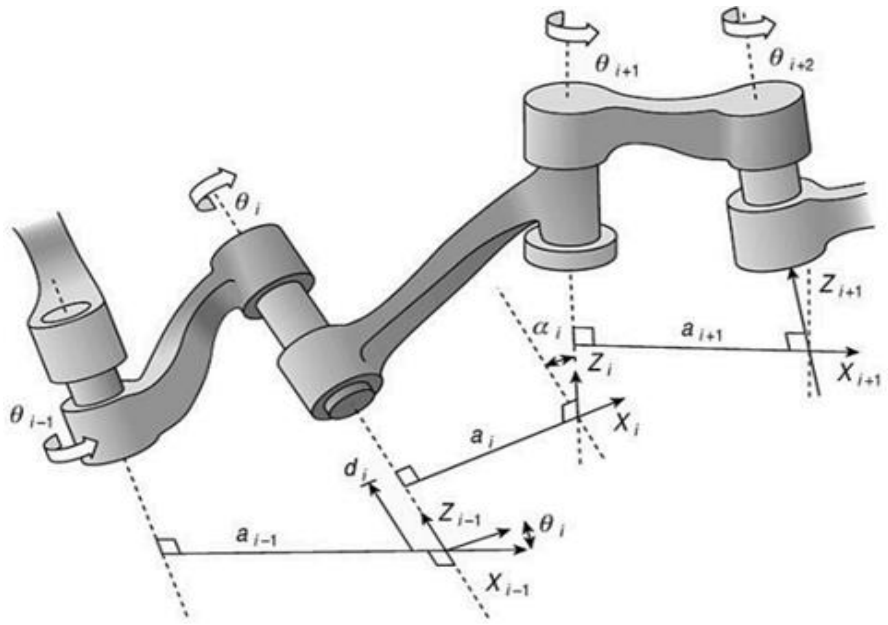
Na sistemática de D-H são atribuídos sistemas de referências de coordenadas cartesianas para cada junta, conforme mostrado na Figura 9. Se a junta é do tipo rotacional, o eixo Z_{i-1} é alocado ao longo do movimento da junta i em conformidade com a regra da mão direita, para rotações. Se a junta for do tipo prismática, este eixo é alocado ao longo do movimento linear. O eixo X_i é a normal comum entre os eixos Z_{i-1} e Z_i e apontado para fora de Z_{i-1}. O eixo Y_i completa o sistema conforme a regra da mão direita, porém, esse último não é utilizado na representação de D-H, (ROSÁRIO, 2005). No Quadro 1 são apresentadas as variáveis para a notação de D-H.

Quadro 1 - Variáveis para a notação de D-H.

Variável	**Definição**
a_i	Comprimento do elo
d_i	Distância entre eixos de duas articulações consecutivas
α_i	Ângulo entre eixos de juntas consecutivas.
θ_i	Rotação em torno do eixo de articulação.

Fonte: Niku (2013.).

Figura 9 – Exemplo de parâmetros para notação de Denavit-Hartenberg.



Fonte: Niku (2013).

Os estudos feitos neste capítulo foram necessários para o embasamento teórico do projeto e do manipulador robótico utilizado. Ressalta-se que neste trabalho não foi feito nenhum tipo de abordagem de cinemática direta ou sistemática de D-H para o protótipo.

4 HARDWARE E SOFTWARE UTILIZADOS

São abordados neste capítulo o hardware e o software utilizados na pesquisa, entre esses uma breve introdução sobre o Arduino e o módulo *Bluetooth*, servomotores e também sobre a linguagem Python e os *softwares* VS Code e Flutter.

4.1 Placas Programáveis

As placas programáveis, também chamadas de placas de prototipagem rápida[1], são ferramentas utilizadas para facilitar a solução de diversos problemas que podem ser encontrados no dia a dia do desenvolvimento – cria versões preliminares de uma ideia para testá-la e refiná-la antes do lançamento. São sistemas automatizados que facilitam o trabalho dos desenvolvedores de hardware, principalmente. Um ótimo exemplo de placa de prototipagem é a plataforma Arduino, que iniciou uma grande revolução tecnológica no mundo para o desenvolvimento de sistemas automatizados. Outras placas surgiram no mercado de componentes depois da linha Arduino, como exemplo tem-se a linha do *Raspberry*, *ESP* e outras (CARDOSO et al, 2021).

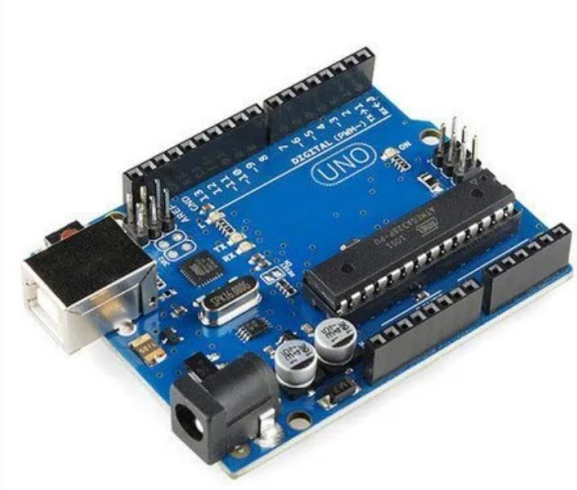
Estas placas são capazes de fornecer prototipação de problemas com circuitos, sendo a programação feita em softwares próprios, que possibilitam a criação de programas para projetos eletrônicos simples ou complexos. Para este projeto de pesquisa a placa escolhida foi um Arduino UNO R3.

4.1.1 Arduino

O Arduino é uma placa de prototipagem rápida[1] programável e utilizada para facilitar o desenvolvimento de problemas encontrados no dia a dia na engenharia eletrônica, com muitas vantagens no seu uso, entre esses, a facilidade de programar, versatilidade, utiliza linguagem própria para programar e há outros modelos para qualquer projeto mais complexo. Na Figura 10 mostra-se um Arduino UNO R3 com versão do ATmega 328 em DIP (CARDOSO et al, 2021).

1 Prototipagem rápida – ação de gerar um protótipo. Criação rápida e iterativa de protótipos, utilizando métodos e ferramentas que permitem a avaliação visual e funcional de um produto (ou conceito) de maneira ágil e econômica, comparada aos métodos tradicionais.

Figura 10 – Arduino UNO R3.



Fonte: Própria do autor.

O Arduino Uno é baseado no microcontrolador ATmega328, possuindo 14 portas digitais, das quais 6 podem ser utilizadas como saídas PWM e 6 portas analógicas; uma porta UART no padrão TTL de 5V, e outras duas portas para interrupção, tem suporte ao protocolo de comunicação SPI e I2C e ADC com resolução de 10 bits. É conectado a um computador com um cabo USB para a sua alimentação, mas pode ser alimentado externamente por uma fonte de 5V. Suas especificações técnicas são apresentadas no Quadro 2.

Quadro 2 – Especificações técnicas do Arduino UNO R3.

Microcontrolador:	**ATmega328 (8-bit AVR RISC); Clock: 16MHz**
Tensão de entrada:	7V à 9V
Tensão de funcionamento:	5V
Saídas Digitais:	14 Pinos (dos quais 6 podem ser saída PWM).
Saídas analógicas:	6 saídas com tensão em 3.3V.
Memory ATmega328:	Flash 32KB (dos quais 0,5KB são utilizados pelo carregador de inicialização); EEPROM: 1KB; SRAM: 2KB.

Fonte: Própria do autor.

A programação do Arduino (para todos os modelos) é feita em softwares próprios, chamado de Integrated Development Environment (IDE) um aplicativo de computador que possui um compilador integrado, simples de se utilizar. A linguagem de programação do Arduino é baseada na linguagem Wiring, Processing e outras de código aberto, sendo que a maioria delas tem origem nas linguagens C/C++. O

Arduino UNO R3 é muito utilizado para desenvolver projetos eletrônicos e robóticos, sistemas de automação doméstica, entre outros.

O software Arduino IDE é open source utiliza a programação em linguagem C++ com algumas modificações, sendo possível escrever, compilar e carregar códigos nas placas do Arduino que disponibiliza diversas bibliotecas feitas pelos próprios usuários do software, que facilitam o desenvolvimento de projetos.

4.2. Módulo *Bluetooth*

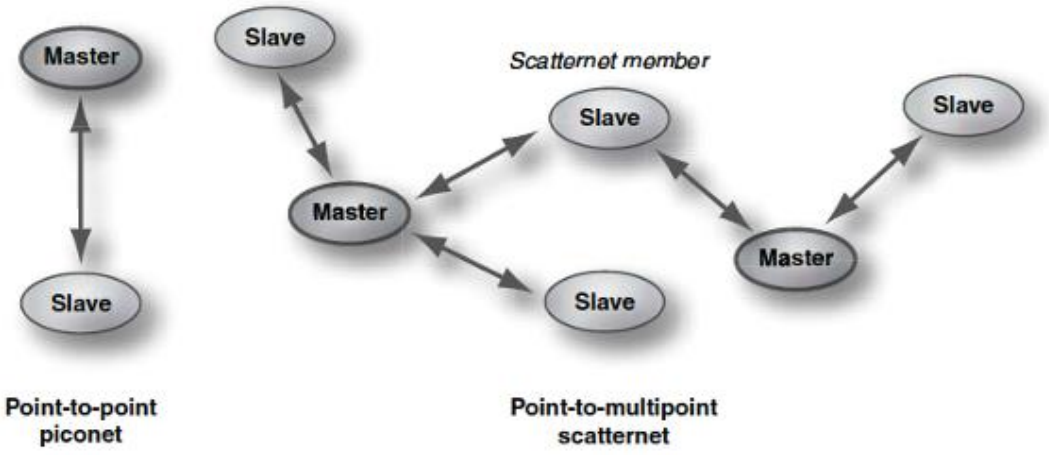
Um módulo *Bluetooth* é um dispositivo que permite a transmissão e a recepção de dados no modo *wireless* (sem fio). O conceito do *Bluetooth* tem suas origens em meados de 1994 pela empresa Ericsson, quando pesquisava a substituição dos cabos conectores para celulares e computadores com a conexão *wireless*. A Ericsson rapidamente notou o grande potencial de mercado para o *Bluetooth* e teve o auxílio e o interesse das empresas Nokia, IBM, Intel e Toshiba (MORROW, 2002).

Quando dois ou mais dispositivos se interconectam de modo sem fio pelo *Bluetooth,* um destes dispositivos é designado de *master* (do inglês: mestre) e os outros são designados como *slaves* (do inglês: escravos). O mestre é a unidade que inicia a conexão de comunicação, dessa maneira, todo dispositivo *Bluetooth* pode ser mestre ou escravo (MORROW, 2002).

A rede que comunica com os dispositivos é denominada de *piconet* (pequena rede). Quando a conexão tem apenas um escravo, a conexão é chamada de *point-to-point*. Quando a conexão suporta mais que um escravo (sendo no máximo até sete escravos), é chamada de configuração *point-to-multipoint.* Ressalta-se que os escravos se comunicam diretamente apenas com o mestre e nunca com os outros escravos (MORROW,2002).

A comunicação entre as *piconets* pode ser realizada se o dispositivo *Bluetooth* pode ser escravo em mais de uma *piconet* ou um mestre em uma *piconet* e escravo em outra. As *piconets* configuradas desse modo são chamadas de *scatternets* (MORROW, 2002). Na Figura 11 são apresentadas as configurações possíveis para as conexões *Bluetooth*.

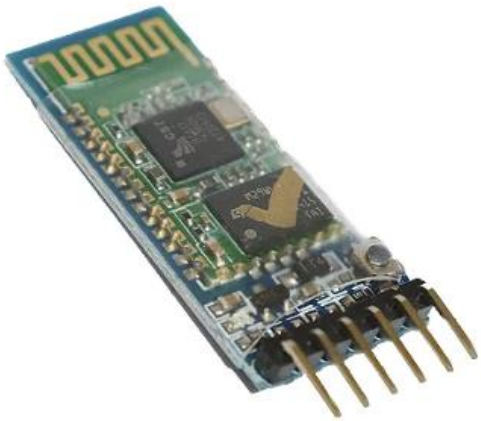
Figura 11 – Configuração para a conexão *Bluetooth.*



Fonte: Morrow (2002)

O módulo *Bluetooth* utilizado nesta pesquisa é um módulo HC-05, mostrado na Figura 12. Este módulo suporta uma tensão mínima de 3,3 a 5V e consegue se manter conectado com qualquer dispositivo por uma distância de até 10 metros.

Figura 12 – Módulo *Bluetooth* HC-05.



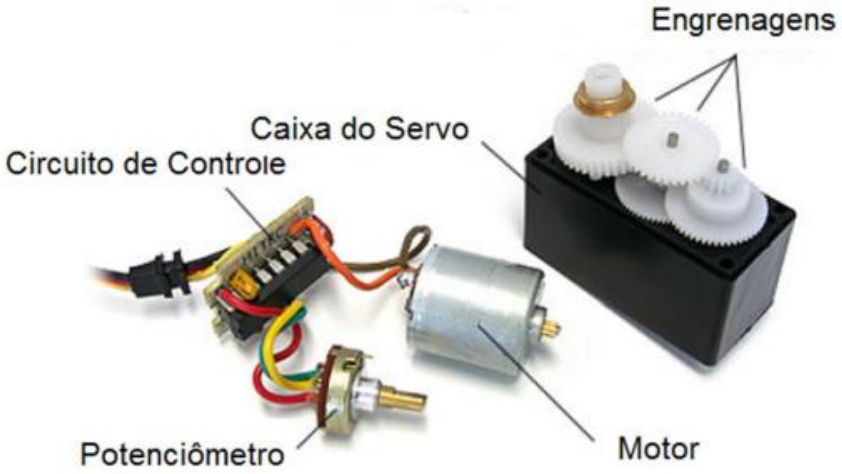
Fonte: Própria do autor.

4.3 Servomotores

O servomotor é um atuador elétrico projetado para aplicações onde é necessário fazer o controle de movimento com posicionamento de alta precisão, reversão rápida e de alto desempenho. É composto por um motor de corrente continua (DC) e um conjunto de engrenagens que funciona como redutor de

velocidade, junto com um sensor de posição (potenciométrico) e um circuito de controle realimentado, são pequenos, com ampla variação de torques. Mostra-se na Figura 13 seus componentes internos (CARRARA, 2015).

Figura 13 – Componentes internos de um servomotor.



Fonte: Própria do autor.

Um motor DC funciona com base na Lei de Lorentz, isto é, uma espira imersa em um campo elétrico, energizada, e com o aparecimento da corrente nessa espira, surgem as forças de sentidos opostos nas extremidades da bobina fazendo com que a espira gire. Este é o princípio no qual se baseia um motor DC, um motor real apresenta várias espiras, tal que as forças em cada uma se somam produzindo um torque alto, e também escovas de carbono para que seja feita a inversão do sentido de corrente nas espiras, permitindo que o eixo gire uma volta completa.

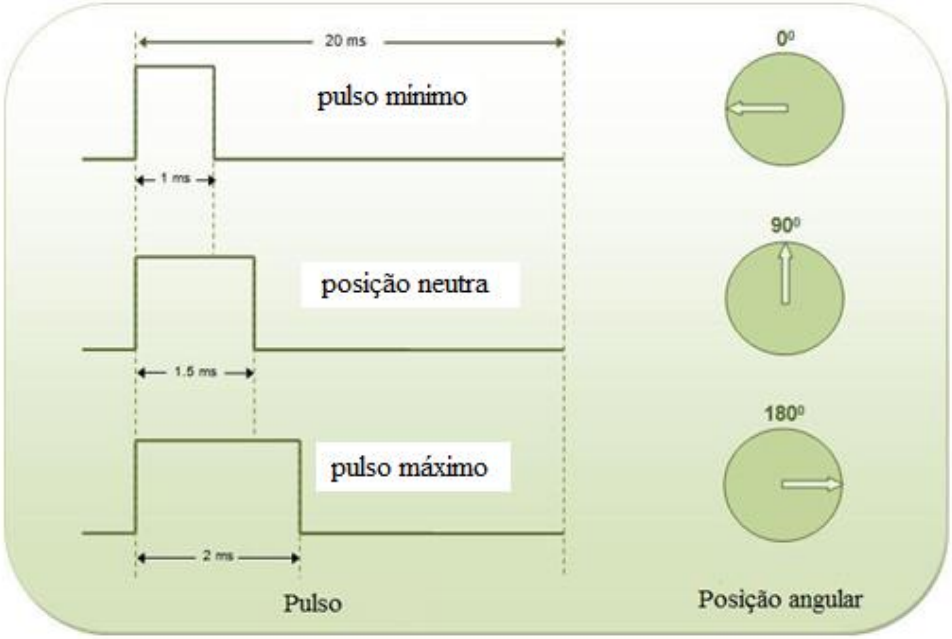
Os servomotores são pequenos e leves, devido a sua construção, mas apresentam altos torques. O sistema de controle utiliza o potenciômetro para identificar a posição do eixo e ajustar a posição em relação ao sinal de comando em sua entrada. Desta forma, estes motores podem mover o eixo e manter a posição desejada, enquanto houver o sinal de comando (CARRARA, 2015).

Os servomotores apresentam três fios para conexão, de diferentes cores: vermelho - alimentação; preto (ou marrom, dependendo do fabricante) deve ser conectado à terra (referência) e o terceiro fio, que pode ser amarelo, branco ou laranja, é o de sinal de controle, sendo padrão para seus conectores

A alimentação DC do servomotor está entre 4.8V e 6V, dependendo do motor. O sinal de controle ou comando é do tipo PWM este é interpretado pelo circuito de controle para mover o eixo para a posição desejada. Pulsos de 1ms a 2ms são

enviados para o motor, cerca de cinquenta vezes por segundo (50 Hz), para definir a posição angular do eixo, sendo que a largura do pulso (entre 1ms e 2ms) carrega a informação sobre o posicionamento. Na Figura 14 mostra-se a relação entre a largura do pulso e a posição angular do eixo.

Figura 14– Relação entre largura de pulso e posição angular.



Fonte: Adaptado de Mota (2017).

Ao receber o sinal de controle, o circuito interno do servo converte-o em informação de posição. Caso a posição atual do servo seja diferente da obtida pelo sinal de comando, o motor gira até alcançar a nova posição. Um sinal com largura de pulso de 1ms corresponde a posição do servo todo a esquerda ou 0 graus e assim por diante para as outras duas posições. Ao se tentar alterar a posição do servo motor, verifica-se uma resistência devida ao motor, esta resistência é chamada de Torque. O Torque medido em kgf.cm (kilograma força por centímetro) é uma das principais características do servomotor.

4.4 Python e suas extensões - software.

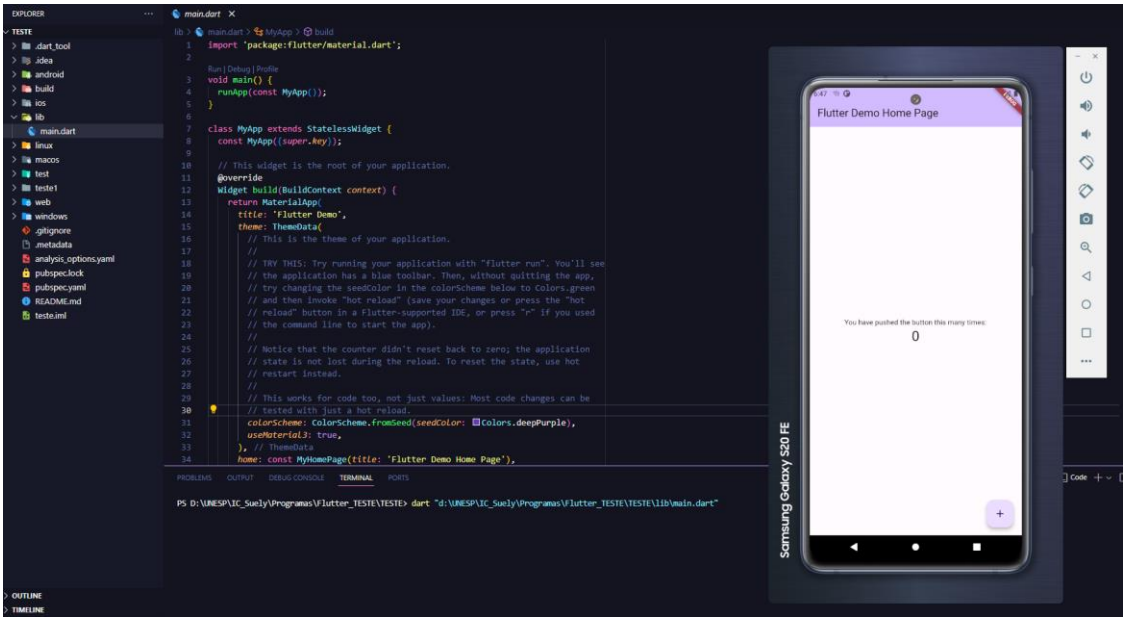
A linguagem Python foi idealizada e desenvolvida por Guido Van Rossum, matemático holandês, no início dos anos 90. Tinha como objetivo otimizar a leitura de códigos e estimular a produtividade de um programador ou qualquer outro profissional (KRIEGER, 2022).

Por ser uma linguagem de sintaxe simples e de fácil compreensão, popularizou-se entre profissionais da indústria tecnológica que não são especificamente programadores, como, matemáticos, cientistas de dados, pesquisadores e outros (KRIEGER, 2022).

Por ser uma linguagem amplamente utilizada, atualmente, nota-se que há um alto número de interpretadores da linguagem, tais como: PyCharm, Jupyter, Spider, VS Code, etc. O VS Code é um dos mais usados atualmente por ser fácil de programar e com layout amigável. O interpretador Jupyter funciona com programação por blocos contudo, não suporta algumas extensões como o Flutter, que permite a programação e criação de aplicativos em Python.

O Flutter é um *framework* para desenvolvimento de interfaces multiplataforma, geralmente utilizado com a linguagem Dart. No entanto, pode ser integrado ao VS Code, onde também é possível trabalhar com Python em paralelo. O Flutter, por ser um programa de extensão, pode ser programado e utilizado pelo VS Code sem nenhum problema de visualização, porém ainda é necessário programar para que o aplicativo fique com a interface desejada, colocar botões e telas de mudança para que o aplicativo funcione perfeitamente. Na Figura 15 mostra-se a tela de inicialização do VS Code com o Flutter, quando se inicia a programação com a versão do Android Samsung S20 Fe.

Figura 15 – Tela de iniciação do VS Code com o Flutter.



Fonte: Própria do autor.

4.5 Android Studio - software.

O Android Studio é um Ambiente de Desenvolvimento Integrado (IDE), um programa de computador que reúne as características e ferramentas de apoio para a criação de aplicativos para dispositivos móveis Android, permitindo a criação de aplicativos e emulação de celulares em computadores e *notebooks*. A função de emular o celular em um computador serve para que o desenvolvedor verifique a criação do aplicativo em tempo real e também como o aplicativo ficará em um celular Android (HARADA, 2019).

É uma ferramenta oficial da Google para projetos em aplicativos Android apresentado em 2013 e lançado em dezembro do ano seguinte. O software contém diversas funções exclusivas que permitem desenvolvedores criarem de forma rápida e eficaz aplicativos para plataformas Android, e tem como base a linguagem de programação Kotlin, Java e suportes para C/C++.

O Android Studio permite criar maquinas virtuais de celulares devido o AVD Manager (do inglês: Android Virtual Device Manager – Gerenciamento de Dispositivos Virtuais Android), contém diversos SDKs (do inglês: Software Developer Kit – Kit de Desenvolvimento de Software) que permitem criar aplicativos para quaisquer dispositivos Android e também contém um sistema de automação de compilação baseado em Gradle. Contém grande variedade de recursos, por isso sua importância para a criação e desenvolvimento de aplicativos para dispositivos Android (CALANDRO, 2023).

4.6 Bibliotecas do Arduino

O Arduino tem várias bibliotecas algumas foram utilizadas neste projeto, por exemplo a *VarSpeedServo*. Esta foi criada para a manipulação da velocidade dos servomotores e pode ser encontrada na página do GitHub do usuário pvanallen. A biblioteca conta com diversas funções para o controle de servomotor, descritas a seguir:

- slowmove (ângulo, velocidade) - utilizada para movimentar o servomotor até um ângulo configurado pelo software e também a velocidade com que se movimenta. A velocidade é dada de 0 até 255 (resolução de 8

bits) para o PWM , sendo 0 a velocidade mínima de movimentação e 255 é a velocidade mais máxima.;

- attach(Pino) - conecta o servomotor em um pino de entrada;
- sequencePlay(Sequência, Posição da Sequência) - inicia sequência de movimentos em *loop* começando na posição 0;
- sequencePlay(Sequência, Posição da Sequência, *Loop*, Posição de Início) - inicia uma sequência de movimentos com o número da posição na sequência com valor booleano de *loop.* Se o valor booleano for verdadeiro inicia o movimento na posição de início;
- sequenceStop() - pára a sequência na posição de movimento.

Neste capítulo foram descritos os principais componentes de hardware e software escolhidos entre tantos, para ser utilizado no desenvolvimento da pesquisa .

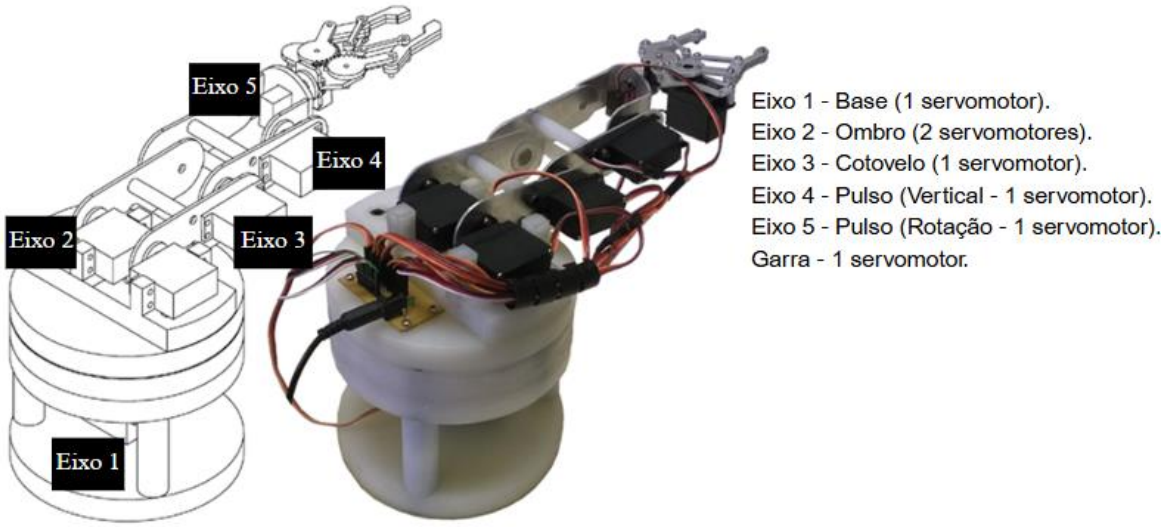
5 DESENVOLVIMENTO DO PROJETO

Neste tópico é apresentado o protótipo do manipulador robótico usado no trabalho e o desenvolvimento do aplicativo.

5.1 O Protótipo

O manipulador robótico utilizado no projeto do aplicativo foi projetado e montado por Nunes (2016), cujo projeto e o modelo real são mostrados na Figura 16, confeccionado nos laboratórios do DEE-FEIS-UNESP e no IFSP Câmpus Presidente Epitácio.

Figura 16 – Arquitetura do manipulador robótico- modelo e protótipo.



Fonte: Adaptado de Nunes (2016)

Trata-se de um manipulador articulado (RRR) de 5 GDL, sendo três graus de liberdades necessários para o movimento do “braço” e dois para o “pulso”. Os elos são de alumínio e no conjunto da base foi usado o nylon que proporciona ao manipulador sustentação e confiabilidade na realização de suas tarefas.

Os graus de liberdade são dados por sete servomotores que permitem a sua movimentação, sendo seis para o movimento do braço, antebraço e punho do manipulador e um para garra e suas características construtivas são apresentadas no Quadro 3.

Quadro 3 – Características construtivas do protótipo.

Número de eixos	5	
Alimentação	5 VCC/3A	
Alcance máximo	Com a garra	Sem a garra
	466 mm	310 mm
Peso	2641 g	
Base (Diâmetro)	150 mm	
Capacidade de carga	300g (na garra)	
Movimento dos eixos	**Eixo**	**Faixa de trabalho**
	1	180°
	2	90°
	3	45°
	4	90°
	5	180°

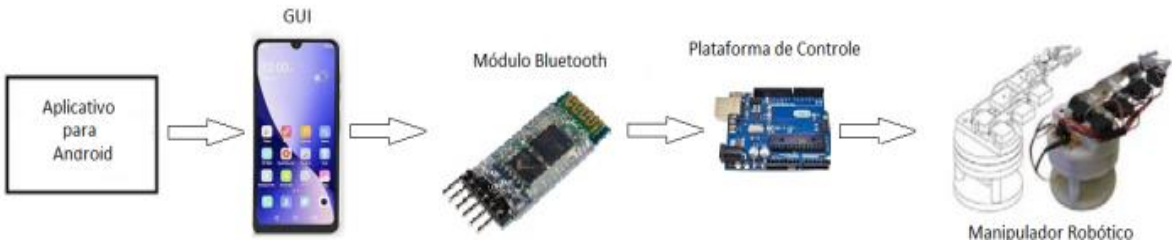
Fonte: Nunes (2016).

As especificações dos servomotores e a barra de ligação com a pinagem dos servomotores são encontrados nos **ANEXOS A** e **B.** Este protótipo é capaz de executar funções de movimentação e levantar objetos de pequeno porte e leve.

5.2 Diagrama ilustrativo do projeto

O diagrama ilustrativo do funcionamento do sistema proposto é apresentado na Figura 17. Neste, um aplicativo no smartphone Android comanda via *Bluetooth* o manipulador que está conectado fisicamente ao Arduino e ao *Bluetooth*, uma vez que o Arduino não possui este recurso. O Arduino é alimentado por conexão USB com o computador e o manipulador é alimentado por uma fonte de alimentação de bancada.

Figura 17 – Diagrama ilustrativo do projeto.



Fonte: Própria do autor.

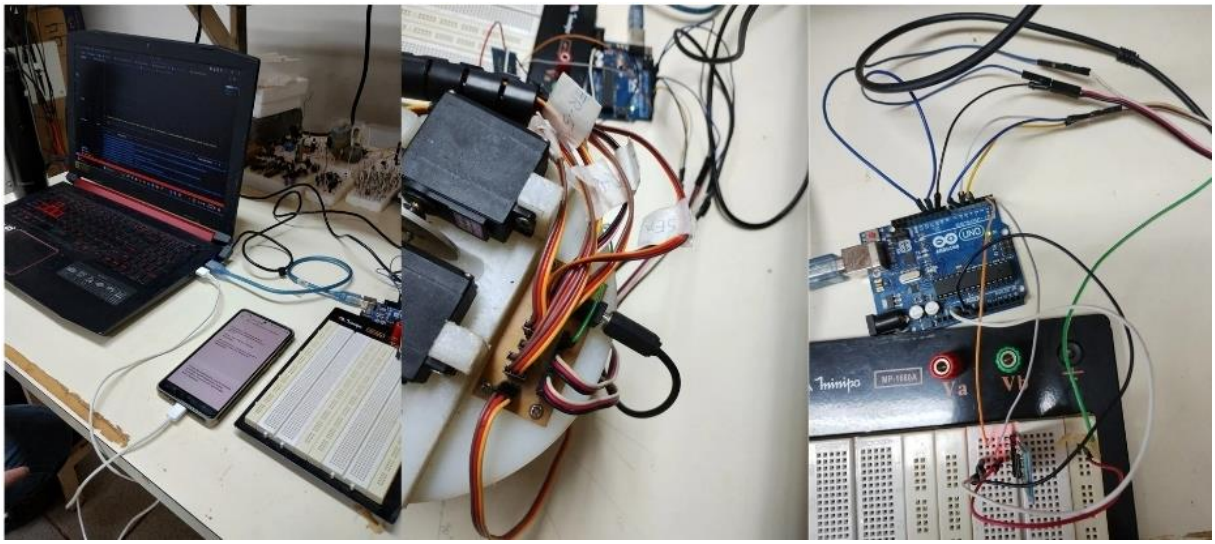
O Arduino UNO R3 foi escolhido devido a facilidade na sua programação e sua versatilidade quando utilizado com o Python e também com o módulo *Bluetooth* HC-05 , compatível com a alimentação dada pelo Arduino .

5.3 Sistema completo para o hardware

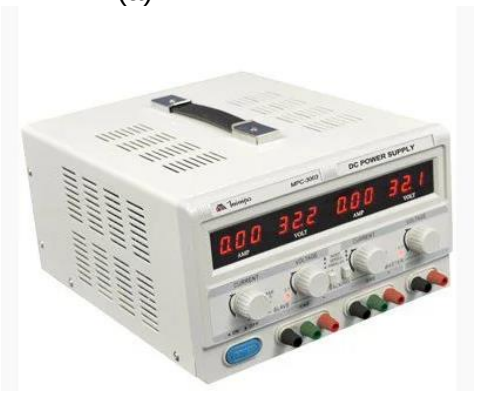
O sistema completo para o hardware implica em ter o Aplicativo emulado no smartphone, parelhado com o módulo *Bluetooth* HC-05 que está com o conectado ao Arduino, que por sua vez está conectado por fios ao manipulador robótico que é alimentado por uma fonte de alimentação de bancada DC dupla 30V/3A - Minipa MPC-3003.

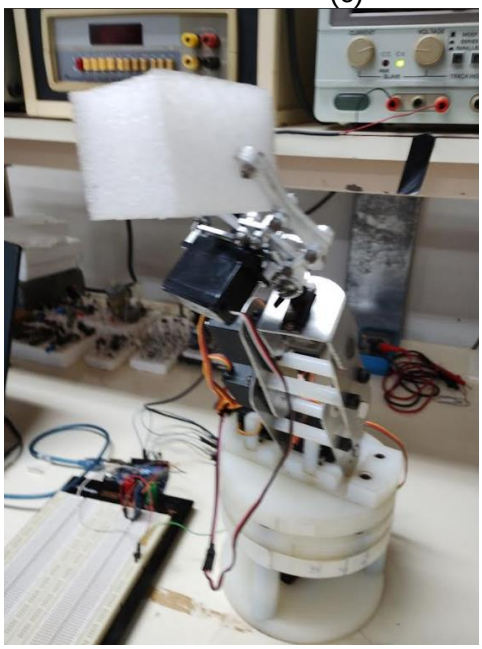
As conexões aos dispositivos são mostrados na Figura 18. As conexões do Arduino UNO ao notebook estão na Figura 18(a); as conexões dos servomotores à entrada da fonte de alimentação.são mostradas na Figura 18(b) e na Figura 18 (c) têm-se as conexões do módulo Bluetooth ao Arduino e aos servomotores.d) Manipulador alimentado por fonte de alimentação de bancada.

Figura 18 – Hardeware. a) Conexões Arduino x notebook.b) Conexões servomotores e a alimentação. c) Conexões do *Bluetooth* , Arduino e servomotores Fonte de alimentação de bancada - Minipa MPC-3003.



(a) (b) (c)





Fonte: Própria do autor.

Na Figura 18(a) observa-se o smartphone ligado ao notebook, em um processo de carregamento do aplicativo no smartphone, habilitado no modo desenvolvedor, ou seja, para quando não é criado a extensão **.**apk.

5.4 Programação

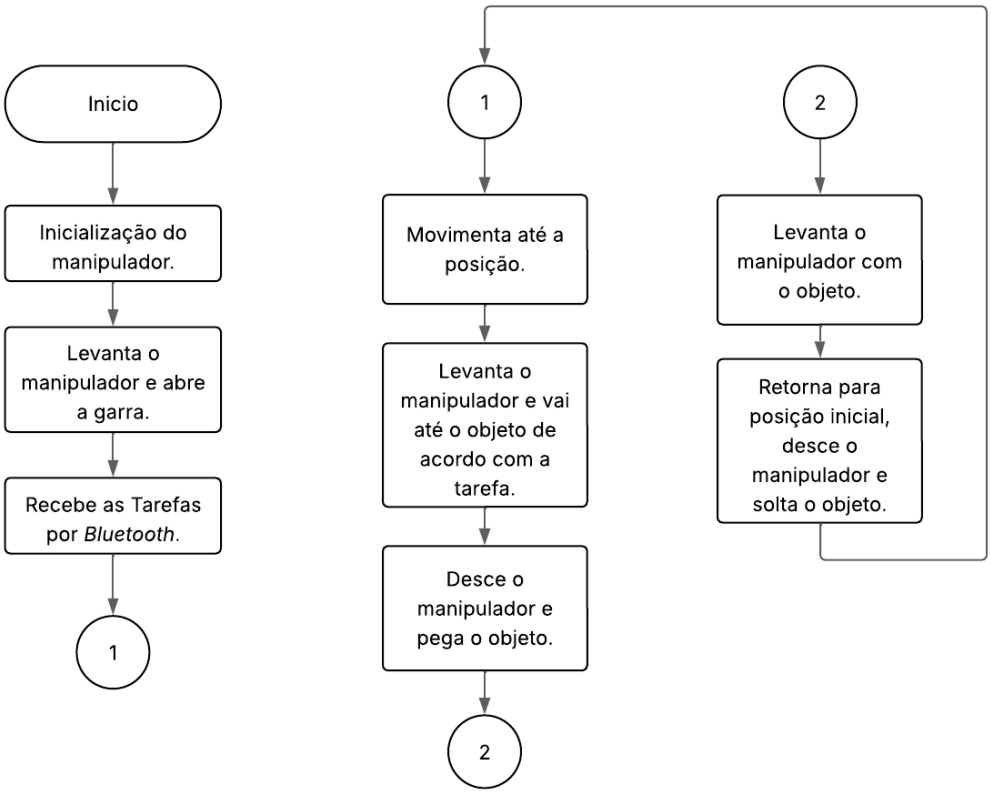
Para desenvolver a proposta deste trabalho foram feitos dois programas um para o Arduino UNO para movimentar o manipulador robótico e outro para criar o Aplicativo (App.) .

5.4.1 Programa para movimentar o manipulador robótico

O Fluxograma do programa para movimentar o braço é mostrado na Figura 19, baseado na sequência listada a seguir:

- Inicializa o manipulador;
- Levanta o manipulador e abre a garra;
- Espera tarefa (posicionamento em 45^0, 60^0 e 90^0);
- Movimenta até a posição ;
- Desce o manipulador e pega o objeto ;
- Levanta o manipulador com o objeto;
- Retorna à posição inicial ;
- Desce o manipulador e solta o objeto.

Figura 19 – Fluxograma para movimentação do manipulador robótico.



Fonte: Própria do autor.

5.4.2 Programa para criar o Aplicativo

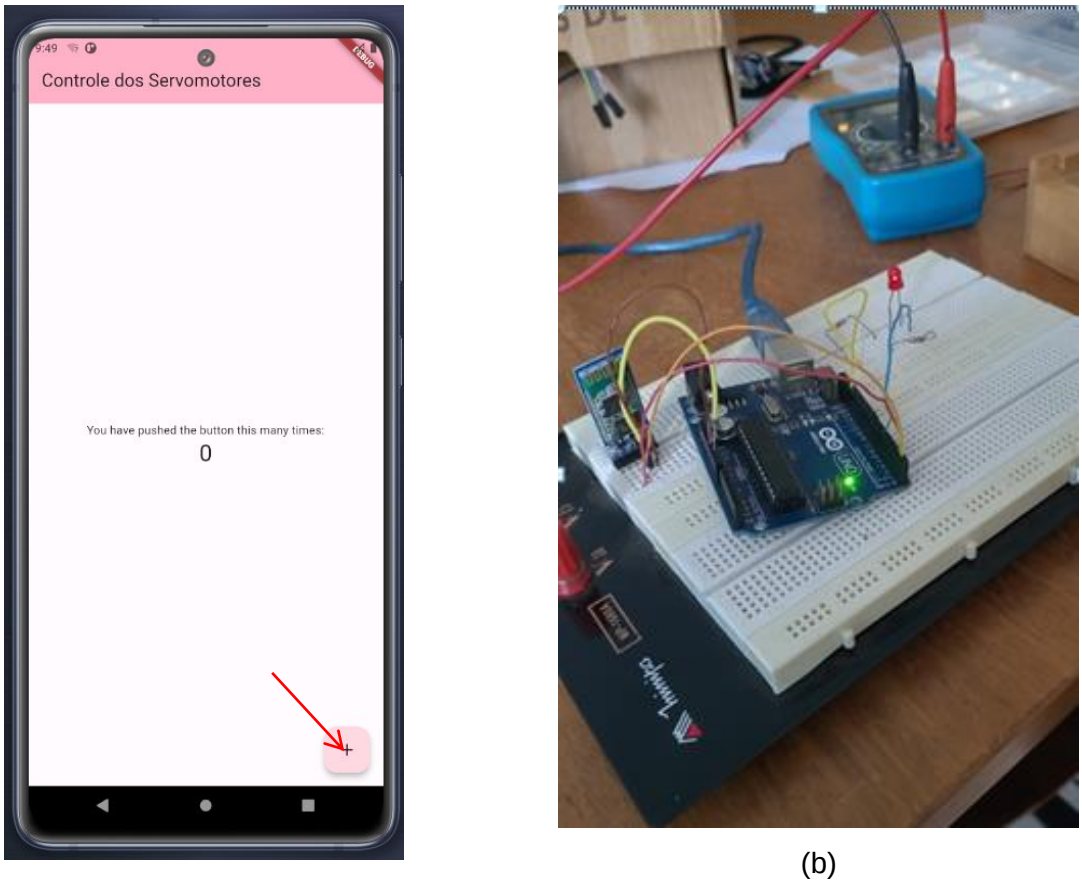
Para programar o aplicativo foi utilizado o *framework* Flutter que aceita a linguagem de programação em Python, convertendo essa programação em um aplicativo para Android no sistema de desenvolvimento VS Code.

Com o VS Code e o Flutter pode-se emular o aplicativo no próprio computador e observar a interface da criação do aplicativo, em tempo real.

Para verificar se a programação do aplicativo estava correta, foi feito um teste, onde um botão na interface projetada do aplicativo usando o Python deveria acender um LED de maneira remota, por meio do Arduino e o *Bluetooth*. Este teste é mostrado na Figura 20 com o funcionamento do botão no aplicativo e o LED aceso, indicando que a programação está correta. Ao final, para o Aplicativo foram criados quatro botões que realizam tarefas distintas para o manipulador robótico. Quando o

Aplicativo está conectado ao módulo HC-05, o *Bluetooth* pára de piscar rapidamente e no Aplicativo aparece que o dispositivo fez a conexão.

Figura 20 – a) Tela inicial emulada para o Aplicativo.b) Acende um LED comandado (seta no botão) pelo Aplicativo.



(a) (b)

Fonte: Própria do autor.

Deste modo, o fluxograma do programa contendo a movimentação do manipulador robótico embarcado no Arduino UNO R3 e a especificação de todas tarefas é apresentado no **APÊNDICE D**.

A programação completa incluindo a programação do aplicativo, encontra-se no **APÊNDICE A** e o programa que movimenta os servomotores, embarcado no Arduino no **APÊNDICE B**.

6 RESULTADOS E DISCUSSÕES

Neste capítulo apresenta-se os resultados obtidos com o Aplicativo e os testes realizados como o manipulador pelo comando do Aplicativo criado.

6.1 Apresentação do Aplicativo desenvolvido

O Aplicativo criado para comandar o manipulador é constituído por quatro botões onde cada botão realiza uma tarefa programada, descritas a seguir :

Tarefa A (cor verde): Deslocamento em 90 graus saindo da posição 0 graus, indo até a posição 90 graus, pega um objeto e retorna para a posição inicial;

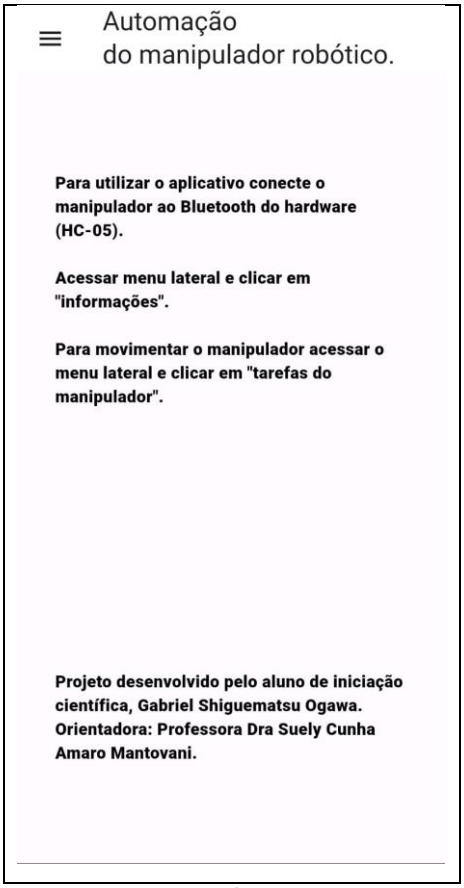
Tarefa B (azul): Deslocamento em 60 graus com roteiro igual a Tarefa A;

Tarefa C(azul magenta): Deslocamento em 30 graus com roteiro igual a Tarefa A;

Tarefa D (vermelho): Parar para manutenção **-** termina a última tarefa executada e volta para a posição inicial, corrigindo o posicionamento de todos os servomotores.

Desse modo, tem-se na Figura 21 e 22 a sequência de telas do aplicativo

Figura 21 – Tela 1 do aplicativo.



Fonte: Própria do autor.

Figura 22 – Tela 2 e 3 do aplicativo.

Fonte: Própria do autor.

Na primeira tela do aplicativo foi adicionada as informações iniciais de como utilizar o aplicativo. Na segunda tela foi criada a página de informações sobre o manipulador, como: grau de liberdade e número de servomotores e a explicação das tarefas e o roteiro. Na última tela têm-se a indicação de *Bluetooth* ligado (ou não) e os botões com as tarefas para movimentar o manipulador robótico. Também foi adicionada uma aba lateral em cada tela que inclui acesso a “Informações” e “Tarefas do manipulador”, com os quatro botões para a movimentação do manipulador robótico.

6.2 Resultados dos testes

Para verificar o funcionamento do manipulador robótico foram feitos vários testes, dentro do volume de trabalho do manipulador., e observados o comportamento e as medidas .

A resposta do manipulador deve ser a mesma para uma tarefa repetida várias vezes , ou seja, ter repetibilidade e a precisão requer que o manipulador saia da posição inicial e retorne para o mesmo ponto, sem grandes diferenças nas medidas.

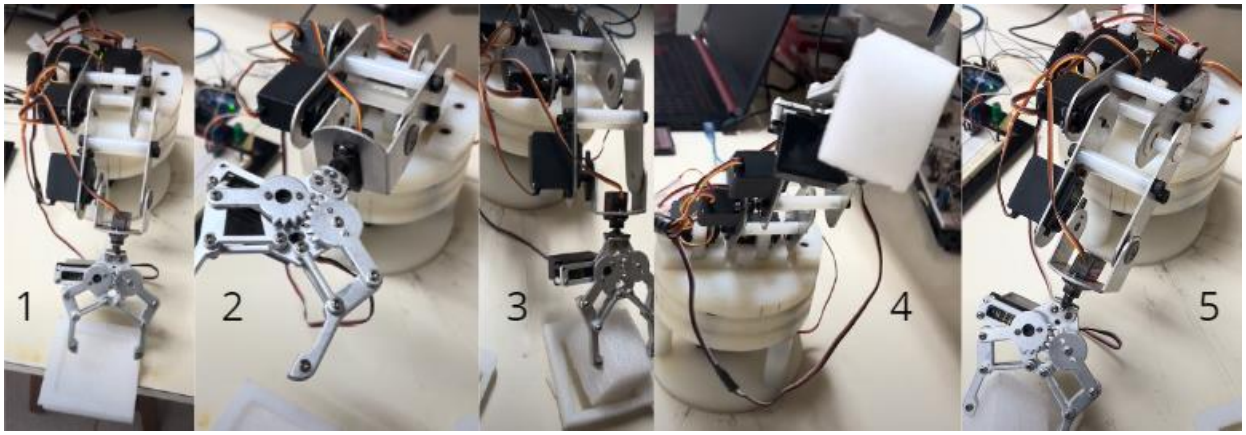
Os resultados com a Tarefa A repetida seis vezes visando verificar o desempenho do Aplicativo criado são apresentados na Tabela 1, que contém a posição programada ,a posição real alcançada e o erro. A Tarefa A do Aplicativo é ilustrada na Figura 23..

Tabela 1 – Posição programada e posição real para a Tarefa A.

Posição no programa (°).	Posição real (°).	Erro percentual de posicão (%)
90	105	16,7
90	105	16,7
90	105	16,7
90	105	16,7
90	105	16,7
90	105	16,7

Fonte: Própria do autor.

Figura 23 – Movimentação para a Tarefa A.



Fonte: Própria do autor.

Para a Tarefa B foram feitos testes semelhantes ao da Tarefa A, para verificar a repetibilidade e a precisão, obtendo-se seis medidas da posição real do manipulador na bancada, conforme apresentado na Tabela 2 que contém a posição programada ,a posição real alcançada e o erro. A Tarefa B do Aplicativo é ilustrada na Figura 24.

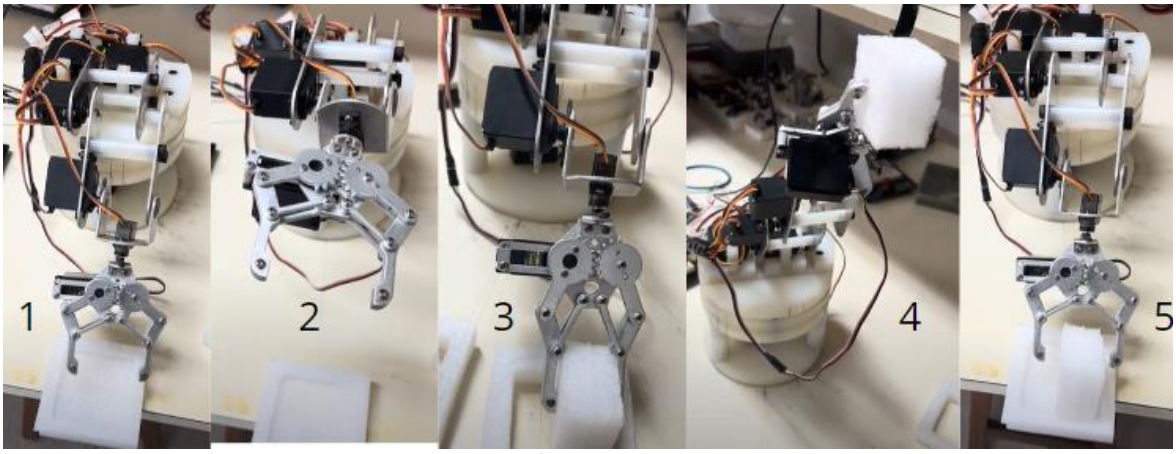
.

Tabela 2 – Posição programada e posição real para a Tarefa B.

Posição no programa (°).	Posição real (°).	Erro percentual de posicão (%)
45	65	44,4
45	65	44,4
45	65	44,4
45	65	44,4
45	65	44,4
45	65	44,4

Fonte: Própria do autor.

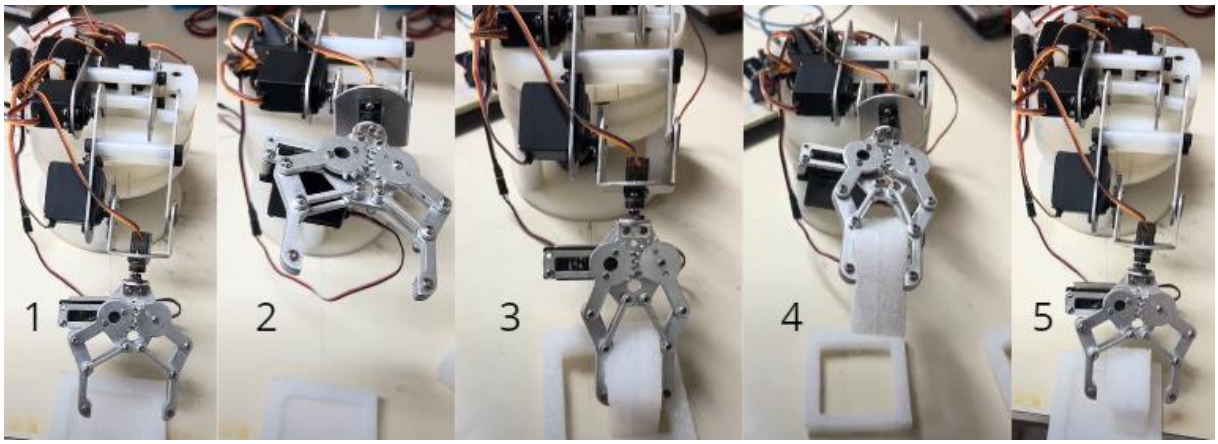
Figura 24 – Movimentação para a Tarefa B.



Fonte: Própria do autor.

Analogamente foram feitos seis testes para a Tarefa C e os resultados da posição programada foi de 30° e a posição real alcançada, 47.5° e o erro relativo é de 58,3. A Tarefa C do Aplicativo é ilustrada na Figura 25.

Figura 25 – Movimentação para a Tarefa C.

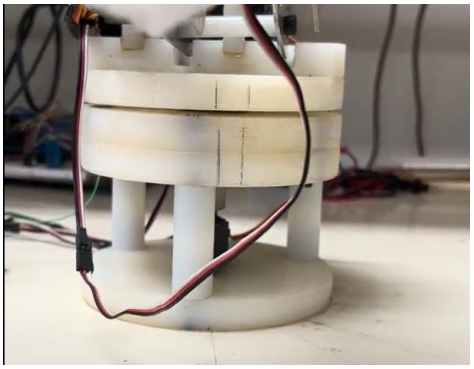


Fonte: Própria do autor.

O Aplicativo foi realizado com êxito e a ideia inicial de 4 botões e 4tarefas, que pode ser progrtamado para ter mais tarefas e funções.

Quanto aos resultados dos testes de posicionamento do manipulador sob o comando dos botões do Aplicativo criado, obtidos para vários ângulos no seu volume de trabalho, demonstram boa repetibilidade e precisão, mas pouca acurácia que é atribuída a estrutura mecânica imprecisa do manipulador. que tem uma diferença angular entre o referencial da garra (posição em que está fixada) e a referência inicial (0°) da base, vista na Figura 26.

Figura 26 – Diferenças angulares nas referências do servomotor da base e a garra.



Fonte: Própria do autor.

7. CONCLUSÃO

Teve-se como tema de pesquisa a realização de um Aplicativo para smartphone Android utilizando a plataforma Arduino UNO, um módulo *Bluetooth*, visando comandar um manipulador robótico com sete servomotores e cinco graus de liberdade, para executar o movimento e posicionamento visando pegar pequenos objetos, respeitando a sua construção e o seu volume de trabalho..

O aplicativo foi desenvolvido na linguagem Python usando o *software* VS Code, interpretador da linguagem Python, e a extensão Flutter, que é uma extensão com bibliotecas que cria aplicativos para smartphone Android. Também foi utilizado o *software* Android Studio que permite a emulação de um smartphone na tela do computador, para verificação no computador .

A utilização da plataforma Arduino UNO permitiu que o trabalho fosse facilitado pelas grandes quantidades de referências e bibliotecas disponíveis para o usuário . Na programação e o acionamento dos servomotores por PWM foi utilizada a IDE do Arduino em conjunto com a biblioteca *varspeedservo*, que incluía o controle de velocidade dos servomotores visando o seu posicionamento e movimentos mais suaves.

Dessa forma, pode-se concluir que o projeto teve êxito, pois o manipulador robótico foi comandado pele Aplicativo criado para smartphone Android, baseado nos resultados das tarefas. E quanto aos resultados do posicionamento do manipulador em seu volume de trabalho oberva-se boa repetibilidade e precisão, mas pouca acurácia, que é justificada pelo mecanismo do manipulador. É importante ressaltar que esta pesquisa foi publicada no CIC –UNESP – 2024, cujo resumo consta do **APÊNDICE C**.

Em projetos futuros recomenda-se usar este aplicativo para comandar outros manipuladores robóticos e a criação de novas tarefas.

REFERÊNCIAS BIBLIOGRÁFICAS

BARBOSA W. H. A. **Braço articulado comandado via Bluetooth por um aplicativo desenvolvido na plataforma Android**. Trabalho de Conclusão de Curso em Engenharia Mecânica- Universidade de Rio Verde, UNIRV, GO. 2016. Pág 1-17.

BRANDÃO L. O.. **Introdução ao uso de funções**. Disponível em: https://www.ime.usp.br/~leo/mac2166/2017-1/introducao_funcoes.html. Acesso em: 02/06/2025.

BOULOS, Paulo; CAMARGO, Ivan de. São Paulo. **Geometria Analítica: Um tratamento vetorial**. Makron. Pág: 1-385.

CALANDRO, Fernanda.. Disponível em: https://grumft.com/android-studio/?translate=pt. Acesso em: 13 jun 2025.

CARDOSO et al. **Placas de Desenvolvimento e Prototipagem com a plataforma Arduino**, 2021. Disponível em: https://pep2.ifsp.edu.br/mct/wpcontent/uploads/2021/10/00081_01_Poster-Placas-de-Desenvolvimento-e-Prototipagem-com-a-plataforma-Arduino.pdf. Acesso em:19 jul 2023.

CARRARA, Valdemir. **Introdução à robótico industrial**.2015. Instituto Nacional de Pesquisas Espaciais. São José dos Campus. Pág: 1-91.

COMPUTAÇÃO BÁSICA.. Disponível em: https://sae.unb.br/cae/conteudo/unbfga/cb/new_subprograma.html. Acesso em: 02 jun 2025.

CRAIG, John J.**Robótica**. 2012. São Paulo. Pearson. Pág: 1-379.

COODESH. Disponível em: https://coodesh.com/blog/dicionario/o-que-e-biblioteca/. Acesso em: 02 jun 2025.

DATACAMP Disponível em: https://www.datacamp.com/pt/blog/what-is-an-algorithm Acesso em: 02 jun 2025.

DE ALMEIDA, A. L.; DE ASSIS, C. V. S. R.; DE ALMEIDA G. A. Q.; GAIA FILHO, H.M.; GUARIENTI, G. S. S.; DE CARVALHO, F. B.. **Braço robótico controlado via smartphone Android**. Tecnologia da Informação e Comunicação: Pesquisas em Inovações Tecnológicas. 2022 - VOLUME 2 .DOI. 10.37885/220408525.

DE ANDRADE, Ana P.. **O que é Flutter?**. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-flutter?utm_source=google&utm_medium=pmax&utm_campaign=home-A-iniciantes&utm_source=&utm_medium=&utm_campaign=&utm_content=&gad_source=1&gad_campaignid=20282427526&gbraid=0AAAAAC_zUlt7iBd7lxkKMLYwgccp_g-

&gclid=Cj0KCQjw0qTCBhCmARIsAAj8C4aj9eptPG7Jz0mupaQLRMmyrFJQdPjgVoxJ5TdIixLyAmPyF8DzvDQaAsDkEALw_wcB. Acesso em: 11 jun 2025.

ELETROGATE**.** Disponível em: https://www.eletrogate.com/modulo-bluetooth-rs232-hc-05. Acesso em: 15 mai 2025.

FERREIRA, E. D. P.; ALVES, N. D. L. A. **Braço articulado com controle proporcional de movimentos comandado via Bluetooth por um aplicativo desenvolvido para plataformas Android**. Universidade do Vale do Paraíba. São José dos Campos. 2013.

GOOGLE CLOUD. Disponível em: https://cloud.google.com/looker/docs/studio/case-simple?hl=pt-br. Acesso em 02 jun 2025.

HANASHIRO, AKIRA. **VS Code – O que é e por que você deve usar?**. Disponível em: https://www.treinaweb.com.br/blog/vs-code-o-que-e-e-por-que-voce-deve-usar. Acesso em 11 jun 2025..

HARADA, EDUARDO Y. **O que é o Android Studio, ferramenta criada para desenvolver apps mobile.** Disponível em: https://www.tecmundo.com.br/software/146361-o-android-studio-ferramenta-criada-desenvolver-apps-mobile.htm?ab=true&. Acesso em: 28 ago 2024.

KRIGER, Daniel. **O que é python, para que serve e por que aprender?** Disponível em: https://kenzie.com.br/blog/o-que-e-python/. Acesso em: 19 jul 2023.

MICROSOFT LEARN. Disponível em: https://learn.microsoft.com/pt-br/cpp/cpp/break-statement-cpp?view=msvc-170. Acesso em 02 jun 2025.

MORROW, Robert. **Bluetooth operation and use**. 2002. Nova Iorque. McGraw-Hill. Pág 1-567.

MOTA, Allan. **O que é Servomotor? | Controlando um Servo com Arduino**. Disponível em: https://portal.vidadesilicio.com.br/o-que-e-servomotor/. Acesso em 29/06/2025.

NIKU, SAEED B. **Introdução à Robótica – Análise, Controle, Aplicações**. 2013. Pág 1-404.

NUNES, R.F .**Mapeamento da cinemática inversa de um manipulador robótico utilizando redes neurais artificiais configuradas em paralelo**. 2016. 100 f. Dissertação (Mestrado em Engenharia Elétrica) – Faculdade de Engenharia, Universidade Estadual Paulista, Ilha Solteira, 2016. Disponível em: https://repositorio.unesp.br/bitstream/handle/11449/138302/nunes_rf_me_ilha.pdf?sequence=4. Acesso em: 5 nov. 2022

PATTON, George S.. Quanto mais você sua no treinamento, menos sangra no campo de batalha. Disponível em: https://www.pensador.com/frase/NjM0NjU5/. Acesso em: 11 jun 2025.

PENTIADO, Thiago Rodrigues. **Abordagem Cinemática para o controle de um Manipulador Robótico de 3GL usando Raspberry Pi**. 2015. Ilha Solteira. UNESP. Pág: 1-78.

PINHEIRO, Fagner. Flutter: O que são widgets e qual sua importância.. Disponível em: https://www.treinaweb.com.br/blog/flutter-o-que-sao-widgets-e-qual-sua-importancia. Acesso em: 11 jun 2025.

ROCKCONTENT Disponível em: https://rockcontent.com/br/blog/algoritmo/ . Acesso em 02 jun 2025.

ROSÁRIO, J. M. **Princípios de Mecatrônica**. São Paulo: Pearson Prentice Hall, 2005.

APÊNDICE A – CÓDIGO DO APLICATIVO SIMULADO NO SMARTPHONE

Programa principal:

```
import 'package:flutter/material.dart';
import 'package:flutter_application/abas/tarefas_manipulador.dart';
import 'package:flutter_application/abas/pagina_inicial.dart';
import 'package:flutter_application/abas/informacoes.dart';
import 'package:flutter/services.dart';
import 'package:flutter_application/widgets/button_widget.dart';
import 'package:flutter_application/widgets/navigationbar.dart';

void main() async {
  runApp(const MainApp());
}
class MainApp extends StatelessWidget {
  const MainApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(useMaterial3: true),
      debugShowCheckedModeBanner: false,
      home: const PaginaInicial(),
    );
  }
}
```

Aplicativo:

```
        import 'dart:convert';
        import 'dart:ui';
        import 'package:flutter_application/widgets/action_button.dart';
        import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
        import 'package:flutter/material.dart';
        import 'package:permission_handler/permission_handler.dart';
        import 'package:flutter_application/widgets/navigationbar.dart';

        class TarefasManipulador extends StatefulWidget {
          const TarefasManipulador({super.key});

          @override

          State<TarefasManipulador> createState() => _MainPageState();
        }

        class _MainPageState extends State<TarefasManipulador> {
          final _bluetooth = FlutterBluetoothSerial.instance;
          bool _bluetoothState = false;
          bool _isConnecting = false;
          BluetoothConnection? _connection;
```

```
List<BluetoothDevice> _devices = [];
BluetoothDevice? _deviceConnected;
int times = 0;


void _getDevices() async {
  var res = await _bluetooth.getBondedDevices();
  setState(() => _devices = res);
}


void _receiveData() {
  _connection?.input?.listen((event) {
    if (String.fromCharCodes(event) == "p") {
      setState(() => times = times + 1);
    }

  });

}

void _sendData(String data) {

  if (_connection?.isConnected ?? false) {

    _connection?.output.add(ascii.encode(data));

  }

}


void _requestPermission() async {
  await Permission.location.request();
  await Permission.bluetooth.request();
  await Permission.bluetoothScan.request();
  await Permission.bluetoothConnect.request();
}

@override
void initState() {
  super.initState();
  _requestPermission();
  _bluetooth.state.then((state) {

    setState(() => _bluetoothState = state.isEnabled);
```

```
  });
  _bluetooth.onStateChanged().listen((state) {
    switch (state) {
      case BluetoothState.STATE_OFF:
        setState(() => _bluetoothState = false);
        break;
      case BluetoothState.STATE_ON:
        setState(() => _bluetoothState = true);
        break;
      // case BluetoothState.STATE_TURNING_OFF:
      //   break;
      // case BluetoothState.STATE_TURNING_ON:
      //   break;
    }
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    drawer: NavBar(),
    appBar: AppBar(
      centerTitle: true,
      title: const Text('Tarefas do Manipulador.'),
    ),
    body: Column(
      children: [
        _controlBT(),
        _infoDevice(),
        Expanded(child: _listDevices()),
        _buttons(),
      ],
    ),
  );
}

Widget _controlBT() {
  return SwitchListTile(
    value: _bluetoothState,
    onChanged: (bool value) async {
      if (value) {
        await _bluetooth.requestEnable();
      } else {
        await _bluetooth.requestDisable();
      }
    },
    tileColor: Colors.black26,
    title: Text(
      _bluetoothState ? "Bluetooth ligado." : "Bluetooth desligado.",
    ),
```

```
  );
}

Widget _infoDevice() {
  return ListTile(
    tileColor: Colors.black12,
    title: Text("Conectado com: ${_deviceConnected?.name ?? "nenhum."}"),
    trailing: _connection?.isConnected ?? false
        ? TextButton(
            onPressed: () async {
              await _connection?.finish();
              setState(() => _deviceConnected = null);
            },
            child: const Text("Desconectar"),
          )
        : TextButton(
            onPressed: _getDevices,
            child: const Text("Ver dispositivos"),
          ),
  );
}

Widget _listDevices() {
  return _isConnecting
      ? const Center(child: CircularProgressIndicator())
      : SingleChildScrollView(
          child: Container(
            color: Colors.grey.shade100,
            child: Column(
              children: [
                ...[
                  for (final device in _devices)
                    ListTile(
                      title: Text(device.name ?? device.address),
                      trailing: TextButton(
                        child: const Text('Conectar'),
                        onPressed: () async {
                          setState(() => _isConnecting = true);

                          _connection = await BluetoothConnection.toAddress(
                              device.address);
                          _deviceConnected = device;
                          _devices = [];
                          _isConnecting = false;

                          _receiveData();

                          setState(() {});
                        },
                      ),
```

```
                )
              ]
            ],
          ),
        ),
      );
}

Widget _buttons() {
  return Container(
    padding: const EdgeInsets.symmetric(vertical: 24.0, horizontal: 8.0),
    color: Colors.black12,
    child: Column(
      children: [
        const Text('Movimentos do Manipulador.',
            style: TextStyle(fontSize: 18.0)),
        const SizedBox(height: 8.0),
        Row(
          children: [
            Expanded(
              child: ActionButton(
                text: "Tarefa A",
                color: Colors.green,
                onTap: () => _sendData("0"),
              ),
            ),
            const SizedBox(width: 8.0),
            Expanded(
              child: ActionButton(
                color: Colors.blue,
                text: "Tarefa B",
                onTap: () => _sendData("1"),
              ),
            ),
            const SizedBox(width: 8.0)
          ],
        ),
        Row(
          children: [
            const SizedBox(width: 8.0),
            Expanded(
              child: ActionButton(
                color: Colors.cyan,
                text: "Tarefa C",
                onTap: () => _sendData("2"),
              ),
            ),
            const SizedBox(width: 8.0),
            Expanded(
              child: ActionButton(
```

```
                    color: Colors.red,
                    text: "Tarefa D",
                    onTap: () => _sendData("3"),
                  ),
                ),
              ],
            ),
          ],
        ),
      );
    }
  }
```

Configuração do botão:

```
import 'package:flutter/material.dart';
class ActionButton extends StatelessWidget {
  const ActionButton({
    super.key,
    required this.color,
    required this.text,
    this.onTap,
  });
  final Color color;
  final String text;
  final VoidCallback? onTap;
  @override
  Widget build(BuildContext context) {
    return Material(
      color: color,
      borderRadius: BorderRadius.circular(24.0),
      clipBehavior: Clip.antiAlias,
      child: InkWell(
        onTap: onTap,
        child: SizedBox(
          height: 150.0,
          child: Center(
            child: Text(
              text,
              style: const TextStyle(fontSize: 18.0, color: Colors.white),
            ),
          ),
        ),
      ),
    );
  }
}
```

Configuração do botão 2:

```
import 'package:flutter/material.dart';
class ButtonWidget extends StatelessWidget {
  final IconData icon;
  final String text;
```

```
  final VoidCallback onClicked;

  const ButtonWidget({
    Key? key,
    required this.icon,
    required this.text,
    required this.onClicked,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) => ElevatedButton(
        style: ElevatedButton.styleFrom(
          minimumSize: Size.fromHeight(50),
        ),
        child: buildContent(),
        onPressed: onClicked,
      );

  Widget buildContent() => Row(
        mainAxisSize: MainAxisSize.min,
        children: [
          Icon(icon, size: 28),
          SizedBox(width: 16),
          Text(
            text,
            style: TextStyle(fontSize: 22, color: Colors.white),
          ),
        ],
      );
}
```

Configuração da aba lateral:

```
import 'dart:ui';
import 'package:flutter/material.dart';
import 'package:flutter_application/abas/informacoes.dart';
import 'package:flutter_application/abas/pagina_inicial.dart';
import 'package:flutter_application/abas/tarefas_manipulador.dart';
import 'package:flutter_application/main_page.dart';

class NavBar extends StatelessWidget {
  const NavBar({super.key});

  @override
  Widget build(BuildContext context) {
    return Drawer(
        child: ListView(children: [
      ListTile(
          leading: const Icon(Icons.home),
          title: const Text('Página Inicial.'),
          onTap: () => selectedItem(context, 0)),
      ListTile(
```

```
          leading: const Icon(Icons.info),
          title: const Text('Informações.'),
          onTap: () => selectedItem(context, 1)),
      ListTile(
          leading: const Icon(Icons.work),
          title: const Text('Tarefas do Manipulador.'),
          onTap: () => selectedItem(context, 2)),
    ]));
  }
  void selectedItem(BuildContext context, int index) {
    Navigator.of(context).pop();
    switch (index) {
      case 0:
        Navigator.of(context).push(MaterialPageRoute(
          builder: (context) => PaginaInicial(),
        ));
        break;
      case 1:
        Navigator.of(context).push(MaterialPageRoute(
          builder: (context) => Informacoes(),
        ));
        break;
      case 2:
        Navigator.of(context).push(MaterialPageRoute(
          builder: (context) => TarefasManipulador(),
        ));
        break;
    }
  }
}
```

APÊNDICE B – CÓDIGO EMBARCADO NO ARDUINO

Programação do Arduino e movimentação dos servomotores:

```
#include <VarSpeedServo.h>

// Pinagem dos Servomotores

//uint8_t servo1 = 3;
//uint8_t servo2 = 5;
//uint8_t servo3 = 6;
//uint8_t servo4 = 9;
//uint8_t servo5 = 10;
//uint8_t servo6 = 11;
uint8_t led = 5;
char data;
//int pos = 0;
//int pos1 = 90;
//int pos2 = 120;
//int pos3 = 70;
//int pos4 = 70;
//int pos5 = 100;
//int pos6 = 180;

// ServoMotores

VarSpeedServo servo1;
VarSpeedServo servo2;
VarSpeedServo servo3;
VarSpeedServo servo4;
VarSpeedServo servo5;
VarSpeedServo servo6;

// Configuração de entrada e saída.

void setup() {
  // put your setup code here, to run once:
    servo1.attach(3);
    servo2.attach(5);
    servo3.attach(6);
    servo4.attach(9);
    servo5.attach(10);
    servo6.attach(11);
    pinMode(led, OUTPUT);
    Serial.begin(9600);
    servo1.slowmove(0,15);
    delay(4000);
    servo6.slowmove(90,15);
    delay(4000);
    servo2.write(140,15);
    delay(4000);
```

```
}
// Execução dos movimentos.

void loop()
{
   if (Serial.available() > 0)
   {
      data = Serial.read();
   }

   switch (data)
   {
     case '0': //Movimentação A -  Pega o objeto em 90 graus e transporta o objeto para 0 graus.

       servo1.slowmove(0,15);
       delay(4000);
       servo6.slowmove(90,15);
       delay(4000);
       servo2.slowmove(110,15);
       delay(4000);
       servo1.slowmove(90,15);
       delay(4000);
       servo2.slowmove(150,15);
       delay(4000);
       servo3.slowmove(140,15);
       delay(4000);
       servo4.slowmove(90,15);
       delay(4000);
       servo5.slowmove(70,15);
       delay(4000);
       servo6.slowmove(10,15);
       delay(4000);
       servo6.slowmove(10,15);
       delay(4000);
       servo2.slowmove(110,15);
       delay(4000);
       servo3.slowmove(80,15);
       delay(4000);
       servo1.slowmove(0,15);
       delay(4000);
       servo2.slowmove(150,15);
       delay(4000);
       servo3.slowmove(140,15);
       delay(4000);
       servo6.slowmove(90,15);
       delay(4000);

       break;
     case '1': //Movimentação B - Pega o objeto em 60 graus e transporta até 0 graus.
```

```
servo1.slowmove(0,15);
delay(4000);
servo6.slowmove(90,15);
delay(4000);
servo2.slowmove(110,15);
delay(4000);
servo1.slowmove(45,15);
delay(4000);
servo2.slowmove(150,15);
delay(4000);
servo3.slowmove(140,15);
delay(4000);
servo4.slowmove(90,15);
delay(4000);
servo5.slowmove(70,15);
delay(4000);
servo6.slowmove(10,15);
delay(4000);
servo6.slowmove(10,15);
delay(4000);
servo2.slowmove(110,15);
delay(4000);
servo3.slowmove(80,15);
delay(4000);
servo1.slowmove(0,15);
delay(4000);
servo2.slowmove(150,15);
delay(4000);
servo3.slowmove(140,15);
delay(4000);
servo6.slowmove(90,15);
delay(4000);

  break;

case '2': // Movimentação C - Pega em 45 graus e transporta para 180 graus.

servo1.slowmove(0,15);
delay(4000);
servo6.slowmove(90,15);
delay(4000);
servo2.slowmove(110,15);
delay(4000);
servo1.slowmove(30,15);
delay(4000);
servo2.slowmove(150,15);
delay(4000);
servo3.slowmove(140,15);
delay(4000);
```

```
        servo4.slowmove(90,15);
        delay(4000);
        servo5.slowmove(70,15);
        delay(4000);
        servo6.slowmove(10,15);
        delay(4000);
        servo6.slowmove(10,15);
        delay(4000);
        servo2.slowmove(110,15);
        delay(4000);
        servo3.slowmove(80,15);
        delay(4000);
        servo1.slowmove(0,15);
        delay(4000);
        servo2.slowmove(150,15);
        delay(4000);
        servo3.slowmove(140,15);
        delay(4000);
        servo6.slowmove(90,15);
        delay(4000);

          break;

        case '3': // Para o manipulador.

        servo1.slowmove(0,15);
        delay(4000);
        servo6.slowmove(90,15);
        delay(4000);
        servo2.slowmove(110,15);
        delay(4000);

        break;

    default:
        break;
    }

    delay(1000);
}
```

APÊNDICE C – RESUMO DO XXXVI CONGRESSO DE INICIAÇÃO CIENTÍFICA - 2024 – UNESP

MANIPULADOR ROBÓTICO COMANDADO POR SMARTHPHONE VIA BLUETOOTH UTILIZANDO UM APLICATIVO CRIADO NA LINGUAGEM PYTHON.

GABRIEL SHIGUEMATSU OGAWA, SUELY CUNHA AMARO MANTOVANI, Faculdade de Engenharia – FEIS, Campus de Ilha Solteira, Engenharia Elétrica, gs.ogawa@unesp.br.

CIC 2024- *"Ciência em tempos de crise climática e social"*

INTRODUÇÃO: Nos últimos anos, tem-se observado a evolução dos robôs devido ao grande avanço tecnológico, com o surgimento de plataformas de controle, sensores miniaturizados e atuadores que levam à redução dos preços de um robô e uma significativa melhoria em seu desempenho. Um robô é definido como um dispositivo eletromecânico, podendo ser controlado de forma automática ou manual[1]. Agregando aos conhecimentos da robótica, têm-se outras tecnologias para o controle, como o uso dos smartphones e os seus aplicativos. Nesta pesquisa tem-se como objetivo, o desenvolvimento de um aplicativo Android visando comandar um manipulador robótico via *Bluetooth*, utilizando um Arduino e a linguagem de programação em Python.

MATERIAL E MÉTODOS: O manipulador robótico[2] usado possui 5 graus de liberdade, dado por sete servomotores em suas juntas. Para o controle e acionamento do manipulador robótico foi usado o Arduino UNO, um módulo *Bluetooth*, HC-05 e uma fonte de tensão de alimentação regulada em 5V para a alimentação dos servomotores. Para criar o aplicativo utiliza-se o software VS Code, para a programação em Python, as extensões Flutter e Android Studio, e a biblioteca *Bluetooth* Serial. Usa-se a IDE (Integrated Development Environment) do Arduino para programar os movimentos do manipulador na linguagem de programação C. Na Figura 1 mostra-se o diagrama esquemático para o controle do manipulador, o *Bluetooth* conectado ao Arduino (alimentado por conexão USB com o computador).

Figura 1. Diagrama de blocos do controle do manipulador.

RESULTADOS E DISCUSSÃO: O circuito montado para os testes é apresentado na Figura 2, sendo mostrada a conexão com o laptop ao Arduino (a), que por sua vez aciona os servomotores,(b).

. **Figura 2.** Circuito e conexões feitas para o manipulador

O aplicativo criado, Figura 3, tem uma tela inicial (a), com as informações para a utilização e o autor, a segunda tela, (b), como usar as tarefas e a terceira com as tarefas que controlam o manipulador, (c). Nesta última tela tem-se o status da conexão Bluetooth, e os botões para movimentar o manipulador, em 90°(A), 60°(B) e 30°(C), e parar (D). Um dos testes, apertando o botão A (verde) de controle, é mostrado na Figura 4, sendo que o manipulador sai da sua posição inicial, agarra um objeto em 90° e retorna com o objeto a posição inicial , 0°.

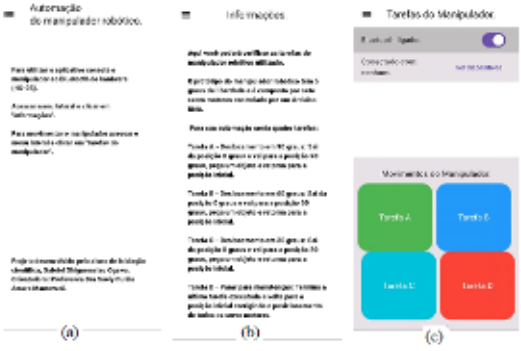


Figura 3. Telas do aplicativo.

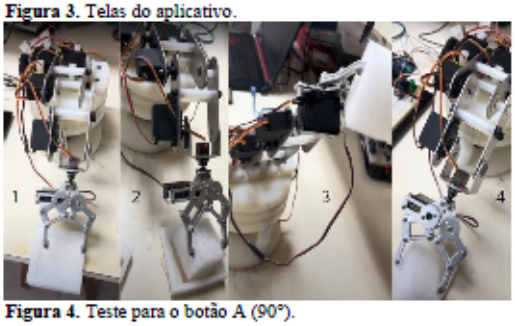
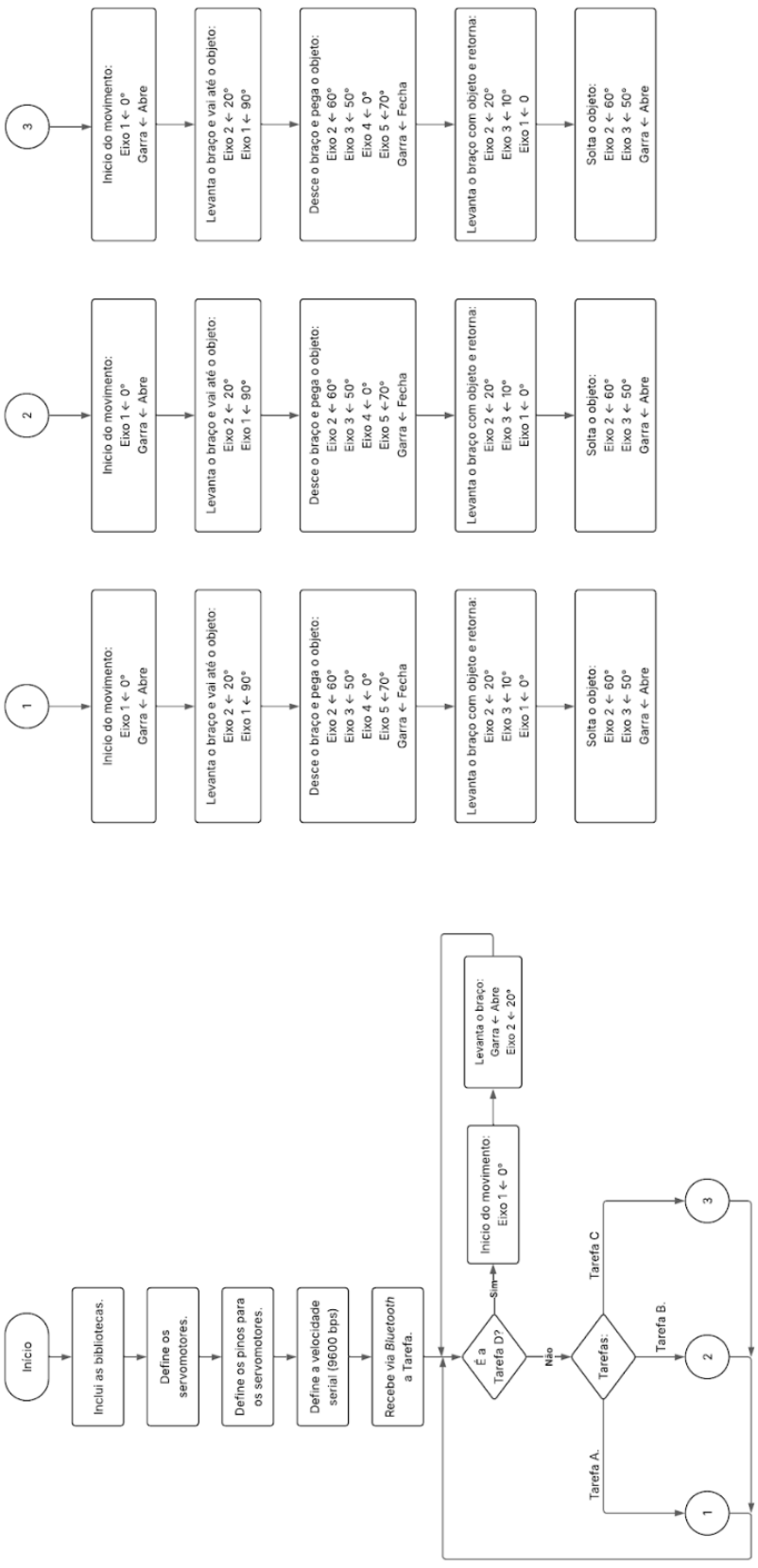


Figura 4. Teste para o botão A (90°).

CONCLUSÕES: O controle de um manipulador robótico por um aplicativo em um smartphone, o Arduino e um módulo *Bluetooth* foi realizado de forma satisfatória. Para os primeiros testes foram criadas 3 tarefas para o posicionamento em seu volume de trabalho, com boa repetibilidade, embora seja necessário reduzir o erro relativo de posicionamento final, que está em torno de 17%, visando aplicação em uma indústria.

REFERÊNCIAS:
1 ROBÓTICA. CRAIG, John J. 2012. São Paulo. Pearson. Pág: 1-379.
2 NUNES, R. F. **Mapeamento da cinemática inversa de um manipulador robótico utilizando redes neurais artificiais configuradas em paralelo.** 2016. 100 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, UNESP. 2016.

APÊNDICE D – FLUXOGRAMA DO PROGRAMA COM AS TAREFAS



Fonte: Própria do autor.

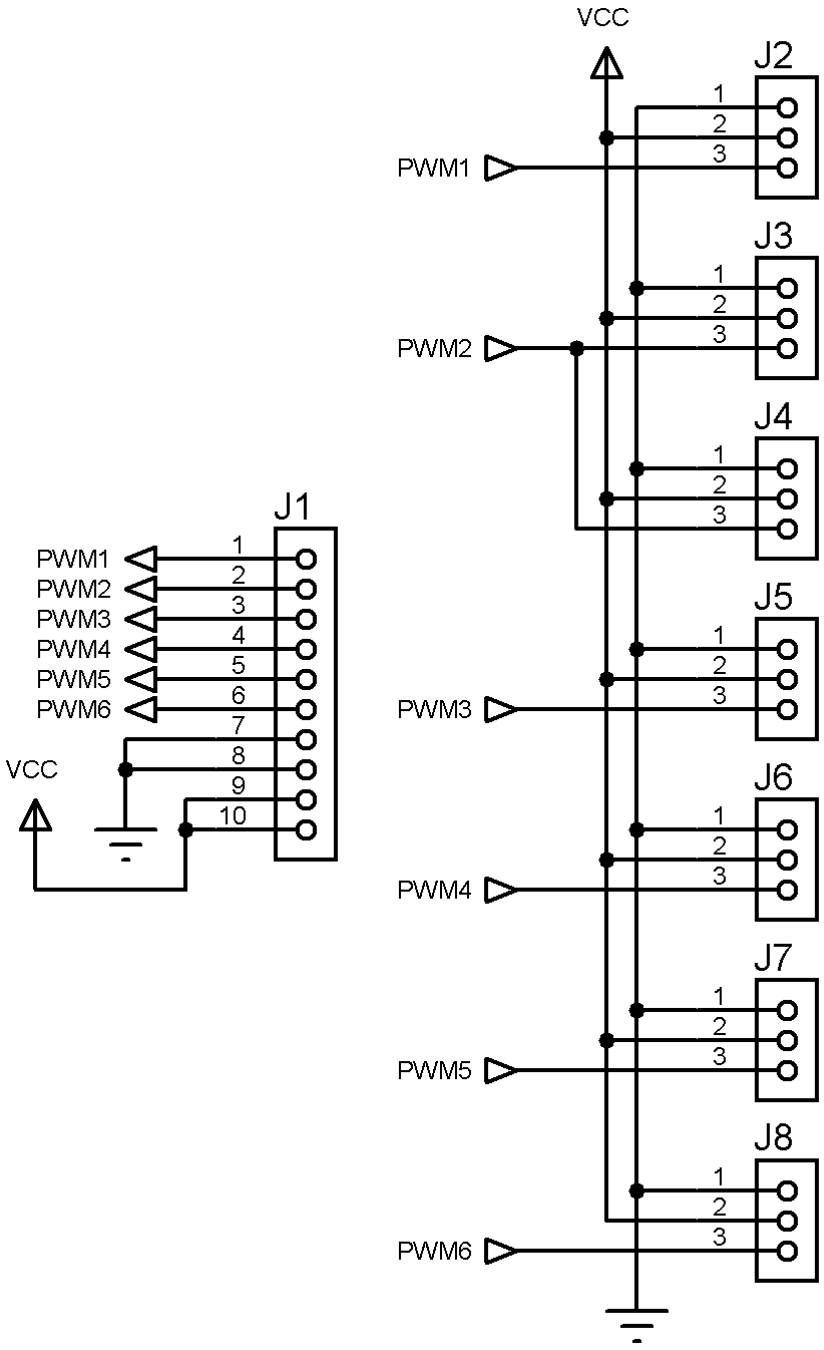
ANEXO A – TABELA COM AS ESPECIFICAÇÕES DOS SERVOMOTORES

Quadro 4 - Dados dos servos utilizados.

Servomotor	**Alocação**	**Torque a 4.8V** (kgf·cm)	**Peso (g)**
MG995 (3) Futaba	Base, antebraço, punho(vertical)	9,4	55
MG996R (2) TowerPRO	Braço	9,4	55
MG90S (1) TOwerPRO	Punho (giro)	1,8	13,4
S3003 (1) TowerPRO	Garra	3,2	37,2

Fonte: Nunes (2016)

ANEXO B – PLACA DE INTERFACE DOS TERMINAIS DOS SERVOMOTORES



Fonte: Nunes (2016)