Desenvolvimento de um Braço Robótico com Controle Multimodal Usando o Controlador ESP8266 NodeMCU

Joedson da Silva Souza[1], Magnum S. de A. P. Ferreira[2], Daniel dos Anjos Costa[3]

[1] Instituto Federal da Bahia (IFBA) – Euclides da Cunha – BA – Brasil

[2] Instituto Federal da Bahia (IFBA) – Euclides da Cunha – BA – Brasil

[3] Informática – Instituto Federal da Bahia (IFBA) – Euclides da Cunha – BA – Brasil

joedsonsouza.ifba.br@gmail.com, magnumsapf@gmail.com, daniel.anjos@ifba.edu.br

Abstract. *This work deals with the implementation of a robotic arm with three different control mechanisms designed to be used in parallel. We developed the following control mechanisms: a local control by buttons on the board, a remote control by mobile app and web interface, and a voice control. We used the ESP8266 NodeMCU to process the input signals coming from each of the control mechanisms. Due to the limited number of digital inputs on the processing board, we used a 74HC148 coding integrated circuit. In the development of the remote control mechanism by application and by voice we used the Blynk IoT platform and the Google Assistant and IFTTT technologies.*

Resumo. *Este trabalho aborda a implementação de um braço robótico com três diferentes mecanismos de controle projetados para serem utilizados de forma paralela. Desenvolvemos os seguintes mecanismos de controle: um controle local por botões na placa, um controle remoto por aplicativo mobile e interface web, e um controle por voz. Foi usado o ESP8266 NodeMCU para processar os sinais de entrada provenientes de cada um dos mecanismos de controle. Em decorrência da limitação da quantidade de entradas digitais da placa de processamento, foi usado um circuito integrado de codificação 74HC148. No desenvolvimento do mecanismo de controle remoto por aplicativo e por voz utilizamos a plataforma Blynk IoT e as tecnologias Google Assistente e IFTTT.*

1. INTRODUÇÃO

Com os avanços tecnológicos que vêm ocorrendo de maneira exponencial ao longo dos anos, tornou-se cada vez mais viável a implementação de projetos envolvendo robótica. Por definição, a robótica é o desenvolvimento de sistemas programáveis capazes de interagir com objetos no mundo físico e executar tarefas complexas de maneira autônoma [Matarić 2014]. Em razão desses avanços, o desenvolvimento de braços robóticos capazes de executar tarefas em diferentes áreas do cotidiano humano tornou-se viável, trazendo benefícios como a facilidade e segurança para realizar atividades que antes expunham pessoas a situações de risco. Com a utilização desses dispositivos robóticos, é possível substituir a intervenção humana em tarefas perigosas ou desgastantes, garantindo maior eficiência e proteção em diversos setores.

O avanço do conceito de Internet das Coisas (IoT), que não se limita apenas à interconexão de dispositivos cotidianos com a internet, mas também busca torná-los inteligentes, capazes de coletar e processar dados do ambiente ou das redes às quais estão conectados [de Oliveira 2017], foi fundamental para o desenvolvimento deste projeto. Através da IoT, torna-se possível a criação de dispositivos mais eficientes e automatizados, que contribuem para a otimização de processos e melhoria da qualidade de vida das pessoas por meio da internet. Deste modo, a IoT nos permitiu interagir e controlar nosso braço robótico através da voz por intermédio de um *smartphone*, dispositivo muito presente no cotidiano da sociedade atual.

Neste projeto, explorou-se a interação de máquinas utilizando comandos de voz e aplicativos *web* ou *mobile*, com base nas possibilidades disponíveis por meio da IoT. O projeto consiste em um braço robótico conectado a um ESP8266 NodeMCU e três diferentes mecanismos de controle projetados para serem utilizados de forma paralela. Explorando as diversas funcionalidades dos dispositivos que estavam disponíveis, foram desenvolvidos os seguintes mecanismos de controle: um controle local com botões na placa de prototipação, um controle remoto por aplicativo *mobile* e interface *web*, e um controle por voz usando o Google Assistente. Em decorrência da limitação da quantidade de entradas digitais da placa de processamento e a grande demanda de sinais elétricos de comunicação, foi utilizado o circuito integrado de codificação 74HC148 para reduzir a demanda de sinais de entrada ligados diretamente ao ESP8266.

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção descreve as tecnologias de *hardware* e *software* utilizados no projeto. Para o desenvolvimento dos mecanismos de controle e do braço robótico, foram utilizados os materiais e tecnologias mencionados na Tabela 1.

Tabela 1. Itens e Tecnologias usadas para construção do projeto

Item	Definição
ESP8266 NodeMCU	É uma plataforma de prototipação com um firmware de código aberto baseado na linguagem de programação Lua [de Oliveira 2017]. Possui um módulo de conexão Wi-Fi integrado, que lhe confere a capacidade de se comunicar com outros dispositivos por meio de uma rede sem fio [Parihar 2019].
Codificador 74HC148	É um circuito integrado baseado em portas lógicas que converte sinais de entrada em códigos binários [Guimarães, 2019]. Esse codificador converte oito entradas de linha para três pinos de saída de dados binários [Datasheet do 74HC148 1992].
Google Assistente	É um assistente virtual inteligente desenvolvido pela Google LLC, disponível para *Smartphones*, que permite que seus usuários tenham a possibilidade de acessar apps e dispositivos inteligentes remotamente por meio de comandos de voz [Google 2023].
IFTTT	É um serviço que permite conectar diferentes aplicações *web* mediante a definição de condições que desencadeiam determinadas ações [Salinas Castillo 2022].
Webhooks	É um serviço de notificações que permite a comunicação entre duas aplicações quase que em tempo real. A troca de informações entre as aplicações acontece quando um evento específico ocorre em algum dos elos da conexão [Calado 2022].
Blynk IoT	Blynk IoT é um conjunto de softwares utilizados para prototipação de projetos, o qual permite controlar remotamente microcontroladores, e analisar informações em tempo real dos sensores e dispositivos conectados à Internet das Coisas por meio de sua plataforma [Blynk 2023].

2.1. Trabalhos Relacionados

Na literatura, foram identificados diversos projetos similares ao que foi desenvolvido, no entanto, nenhum deles incorporou todas as tecnologias mencionadas neste projeto. Daremos destaque a apenas dois deles.

O primeiro projeto, conduzido por Almeida (2022), envolveu a criação de um protótipo de braço robótico articulado. Esse braço foi controlado remotamente por meio de um *smartphone* Android, utilizando uma conexão *Bluetooth* como interface. O trabalho teve como foco a interação humano-computador, com ênfase nos seguintes componentes eletrônicos: Arduino UNO R3, Servo Motores Tower Pro SG90s 9G e o módulo *Bluetooth* HC-05. Esse trabalho demonstrou a viabilidade do controle remoto de um braço robótico utilizando um *smartphone* Android e uma conexão *Bluetooth*. A integração dos componentes eletrônicos e a implementação do aplicativo proporcionaram um produto fácil de se usar. A utilização de componentes eletrônicos acessíveis no mercado de robótica, como o Arduino UNO e os servomotores, torna a construção desse estilo de sistema mais motivador e permite que outros entusiastas desenvolvam projetos semelhantes.

No segundo trabalho, apresentado por Fornarolo e Bacarin (2020), explorou-se a criação de uma Garra Robótica Controlada por Voz. O principal objetivo desse estudo foi desenvolver um sistema que permitisse a execução de ações específicas pela garra robótica em resposta a comandos de voz predefinidos. Essa abordagem foi concebida para simplificar as tarefas diárias de indivíduos com limitações físicas e motoras. A comunicação vocal é estabelecida por meio do módulo de reconhecimento de voz ELECHOUSE V3, que permite a detecção e interpretação dos comandos vocais. Esse módulo é capaz de processar uma variedade de comandos seriais para controlar a garra robótica. A utilização do Arduino e do módulo de reconhecimento de voz demonstra o potencial dessas tecnologias para criar soluções inovadoras que atendam às necessidades dessas pessoas, promovendo a inclusão e a independência.

3. DESENVOLVIMENTO

Nesta seção apresenta-se com detalhes todo o projeto físico do braço desenvolvido. Na Figura 1 é apresentada uma ilustração de alto nível que apresenta a organização esquemática dos controles que atuam no braço robótico. Observa-se que os controles via aplicativo e comandos de voz, utilizando o Google Assistente, podem ser manuseados a distância pela internet, enquanto que o controle local só pode ser manuseado na protoboard, onde estão os circuitos eletrônicos do braço.

Para a movimentação das partes mecânicas do braço quatro servomotores foram utilizados. Esses atuadores tem alta precisão nos seus movimentos e permite controlar o ângulo de seu giro com precisão. Por essa razão, optou-se por utilizar servomotores no projeto em vez de motores convencionais.

A modelagem das peças do braço mecânico utilizado no trabalho é de domínio público e foi retirada do blog Ultimaker Thingiverse [Gray 2014]. Decidiu-se utilizar esse modelo devido à sua ampla utilização em outros projetos descritos na literatura, além de relatos positivos sobre seu design. As peças do braço foram fabricadas em MDF de 3mm, um material composto por placas de fibra de madeira prensadas, utilizando corte a laser.

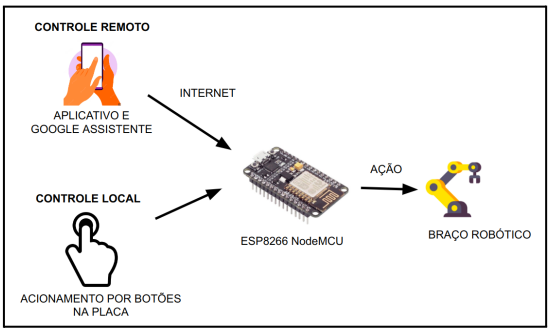


Figura 1. Organização esquemática dos controles

3.1. Controle Local

O controle local de um dispositivo robótico proporciona o total controle do robô no próprio ambiente de atuação. Ele foi implementado usando botões como dispositivos de entrada de dados. A interação do usuário com os botões provoca a reação dos motores, e, por consequência, a movimentação do robô. Neste projeto, empregou-se 7 botões para o controle do braço. Três servomotores são controlados por pares de botões que indicam, quando pressionados, o lado que o eixo do servomotor deve girar. O último botão controla o quarto servomotor que serve para abertura e fechamento da garra do braço.

Devido a limitação da quantidade de entradas digitais do ESP8266 NodeMCU, foi utilizado o codificador 74HC148 (no módulo de codificação) que media os sinais digitais vindo dos botões para as portas de entrada do ESP8266. Utilizando-o, tivemos uma redução de três portas de entrada digital no dispositivo de processamento. Na Figura 2 ilustramos a distribuição dos sinais emitidos pelos módulos do sistema.

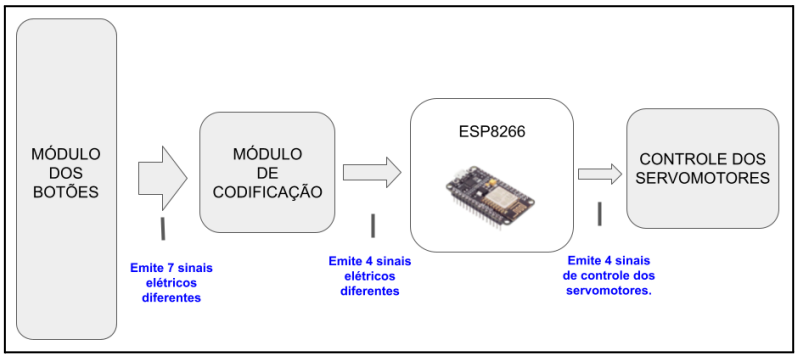


Figura 2. Distribuição dos sinais emitidos pelos módulos do sistema

Na Tabela 2 pode-se observar a combinação das possíveis entradas do módulo de codificação e a saída que será emitida. As colunas nomeadas com E0, E1, E2, E3, E4, E5, E6 e E7 representam os sinais de entrada recebidos pelos botões. E as colunas S0, S1, S2 e GS representam a combinação de saída emitida pelo codificador. A saída GS, segundo o *datasheet* do codificador, é um sinal de controle que quando está em nível lógico 0, informa que os valores emitidos nas saídas S0, S1 e S2 são válidos.

Na linha número 1, apenas a entrada E0 recebe sinal 0 e as demais entradas recebem sinal 1. Essa combinação de entrada resulta na combinação de saída S0=1, S1=1 e S2=1. Seguindo a mesma lógica, na linha 2 apenas E1 recebe sinal lógico 0 e as saídas posteriores recebem sinal lógico 1. Essa segunda combinação resulta na saída S0=1, S1=1 e S2=0. Cada combinação é o identificador de um determinado botão quando pressionado.

Ainda na linha 2, podemos perceber que a entrada E0 recebe o valor x. Isso significa que ela pode receber nível lógico 1 ou 0 que não vai alterar o valor da combinação de saída enquanto E1 estiver pressionado. Isso acontece porque o codificador possui uma ordem de prioridade que privilegia as entradas de maior numeração. Esse procedimento resolve conflitos quando duas ou mais entradas são acionadas ao mesmo tempo.

Na Figura 3 é apresentado o circuito eletrônico do projeto. A figura ilustra a ligação dos botões utilizados no Controle Local, do módulo de codificação, do ESP8266 NodeMCU, dos quatro servomotores e de uma fonte de alimentação externa, que serve para alimentar os servomotores.

Tabela 2. Combinações de entradas e saídas do módulo de codificação.

	ENTRADAS								SAÍDAS			
Nº	**E0**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**S0**	**S1**	**S2**	**GS**
1	**0**	1	1	1	1	1	1	1	**1**	**1**	**1**	0
2	x	**0**	1	1	1	1	1	1	**1**	**1**	**0**	0
3	x	x	**0**	1	1	1	1	1	**1**	**0**	**1**	0
4	x	x	x	**0**	1	1	1	1	**1**	**0**	**0**	0
5	x	x	x	x	**0**	1	1	1	**0**	**1**	**1**	0
6	x	x	x	x	x	**0**	1	1	**0**	**1**	**0**	0
7	x	x	x	x	x	x	**0**	1	**0**	**0**	**1**	0
8	x	x	x	x	x	x	x	**0**	**0**	**0**	**0**	0

É possível observar na Figura 3 que o codificador recebe 3.3V em sua entrada VCC e 0V na entrada GND, que são sinais de tensão provenientes do ESP8266 NodeMCU. Suas saídas nomeadas com A0, A1, A2 e GS foram interligadas nas portas de entrada digital D0, D1, D2 e D6, respectivamente.

Na montagem dos quatro servomotores, foi utilizada uma fonte externa de 5V para alimentação dos motores, enquanto que os sinais de controle são enviados pelo ESP8266. Observa-se na Figura 3 que os fios vermelhos (5V) e o fio preto (0V), saem dos terminais da fonte e são interligados nos dois pinos mais a direita dos servomotores. Os fios que levam o sinal de controle para os servomotores M0, M1, M2 e M3 estão em

destaque na cor lilás, e saem das portas de saída digital D3, D4, D7 e D8 do ESP8266 respectivamente.

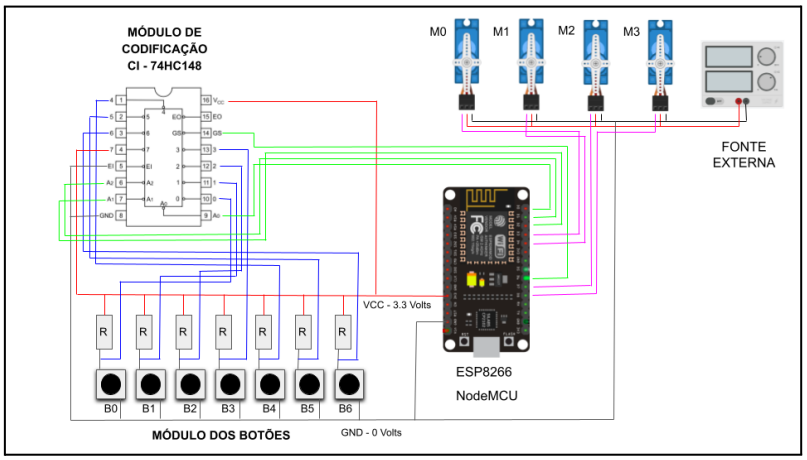


Figura 3. Circuito eletrônico do braço robótico

Diante do circuito eletrônico apresentado na Figura 3, implementou-se o código do controle local que foi carregado no ESP8266 NodeMCU. A Figura 4 contém a primeira parte do pseudocódigo do controle local. Dentro da função "*setup*( )" atribuiu-se aos pinos D0, D1, D2 e D6 o status de pinos de entrada (*INPUT*) nas linhas 9 a 12. Esses pinos estão ligados às saídas A0, A1, A2 e GS do codificador 74HC148, conforme apresentado na Figura 3. Ainda na função *setup*( ) foi atribuído aos pinos D3, D4, D7 e D8 o status de pinos de saída (*OUTPUT*) nas linhas 14, 16, 18 e 20, respectivamente. A esses mesmos pinos vinculou-se os servomotores: "M0", "M1", "M2" e "M3", respectivamente, nas linhas 15, 17, 19 e 21.

```
1
2  #include <Servo.h>
3
4  Servo M0, M1, M2, M3; // Criação dos objetos dos servomotores
5  int anguloM0, anguloM1, anguloM2, anguloM3; // Armazenam os ângulos dos motores
6
7  void setup()
8  {
9    pinMode(D0, INPUT);  //pino de entrada que recebe o sinal A0 do Codificador
10   pinMode(D1, INPUT);  //pino de entrada que recebe o sinal A1 do Codificador
11   pinMode(D2, INPUT);  //pino de entrada que recebe o sinal A2 do Codificador
12   pinMode(D6, INPUT);  //pino de entrada que recebe o sinal GS do Codificador
13
14   pinMode(D3, OUTPUT); //pino de saída que vai atuar no servo M0
15   M0.attach(D3, 500, 2500);
16   pinMode(D4, OUTPUT); //pino de saída que vai atuar no servo M1
17   M1.attach(D4, 500, 2500);
18   pinMode(D7, OUTPUT); //pino de saída que vai atuar no servo M2
19   M2.attach(D7, 500, 2500);
20   pinMode(D8, OUTPUT); //pino de saída que vai atuar no servo M3
21   M3.attach(D8, 500, 2500);
22 }
23
24 void loop() {
25   controleLocal();
26 }
```

Figura 4. Primeira parte do pseudocódigo do Controle Local.

A função "*loop*( )" é responsável por executar o programa principal, executando-o infinitamente enquanto o dispositivo permanecer ligado. Nessa função, ocorre um chamado para a função "controleLocal( )" que comanda toda a

movimentação dos servomotores pelo acionamento dos botões do sistema. O pseudocódigo desta função é apresentado na Figura 5.

```
29  void controleLocal()
30  {
31    int A0 = digitalRead(D0);  //leitura do sinal digital A0 do Codificador
32    int A1 = digitalRead(D1);  //leitura do sinal digital A1 do Codificador
33    int A2 = digitalRead(D2);  //leitura do sinal digital A2 do Codificador
34    int GS = digitalRead(D6);  //leitura do sinal digital GS do Codificador
35
36    if(GS == 0)
37    {
38      if(A2 == 1 && A1 == 1 && A0 == 1) { /*Movimenta o servo M0*/ }
39      else if(A2 == 1 && A1 == 1 && A0 == 0) { /*Movimenta o servo M1 para a DIREITA*/ }
40      else if(A2 == 1 && A1 == 0 && A0 == 1) { /*Movimenta o servo M1 para a ESQUERDA*/ }
41      else if(A2 == 1 && A1 == 0 && A0 == 0) { /*Movimenta o servo M2 para a DIREITA*/ }
42      else if(A2 == 0 && A1 == 1 && A0 == 1) { /*Movimenta o servo M2 para a ESQUERDA*/ }
43      else if(A2 == 0 && A1 == 1 && A0 == 0) { /*Movimenta o servo M3 para a DIREITA*/ }
44      else if(A2 == 0 && A1 == 0 && A0 == 1) { /*Movimenta o servo M3 para a ESQUERDA*/ }
45      else  {  /*Não faça nada*/  }
46    }
47  }
```

Figura 5. Segunda parte do pseudocódigo do Controle Local.

Dentro da função “controleLocal( )” nas linhas 31 a 34, foram criadas variáveis locais do tipo inteiro, A0, A1, A2 e GS para armazenar o valor digital da leitura dos pinos D0, D1, D2 e D6. Em seguida, uma sequência de condições para associar as combinações das entradas A0, A1, A2 e GS aos movimentos dos servomotores do projeto foi utilizada.

Na linha 36 da Figura 5 há a primeira condição que verifica se o valor de GS é igual a zero. Segundo o fabricante, quando o sinal GS emitido pelo codificador tem valor lógico “0” informa que são válidos os valores emitidos por A0, A1 e A2. E quando esse sinal tem valor “1” os valores de A0, A1 e A2 são inválidos. Portanto, a primeira condição (linha 36) garante que os motores só serão acionados por sinais válidos.

Em seguida, entre as linhas 38 e 45, implementou-se oito condições em cascata que fazem a verificação dos sinais recebidos em A0, A1 e A2. Na linha 38 a condição a ser atendida para que o trecho de código dentro do “*if*” seja executado é a combinação “A2 == 1 && A1 == 1 && A0 == 1”. Quando essa combinação for verdadeira, o bloco de código que está dentro dessa condição fica responsável por movimentar o servomotor M0 e salvar a ângulo de giro na variável “anguloM0”. O mesmo procedimento se repete nas estruturas de decisões das linhas 39 a 44 que controlam os demais servomotores, para que assim, a cada vez que um botão for pressionado, haja a execução de um determinado bloco de códigos.

3.2. Controle Remoto por Aplicativo e Interface Web

Nesta seção, é abordado de forma detalhada o controle remoto do braço. Para isso, foi utilizada a mesma estrutura física do circuito ilustrado nas seções anteriores, mas com o acréscimo de linhas de código, que permitem ao ESP8266 receber comandos de uma plataforma externa, seja ela *Web* ou *Mobile*.

De forma resumida, os comandos para controlar o braço robótico são enviados por meio de um aplicativo disponibilizado pela plataforma do Blynk IoT. Esses comandos passam por esta plataforma através da internet e, em seguida, chegam ao

ESP8266 NodeMCU, que é responsável por acionar os motores de acordo com as instruções recebidas.

Dentro da plataforma Blynk IoT foi necessário configurar alguns *Datastreams* (Pinos Virtuais), que são, variáveis que nos permitem armazenar os valores enviados do aplicativo, que serão repassados para o ESP8266 NodeMCU, e consequentemente, para os motores do braço robótico.

Na Figura 6 implementou-se o pseudocódigo com os comandos que vinculam o aplicativo a plataforma Blynk IoT. Nas linhas 2, 3 e 4 estão os comandos para incluir as bibliotecas necessárias para as funcionalidades adicionadas. Nas linhas 6, 7 e 8 foram colocados os *tokens* de validação disponibilizados pelo Blynk IoT. Nas linhas 12 a 14 foram adicionadas variáveis globais para armazenar o SSID da rede Wi-Fi que o ESP8266 NodeMCU ficará conectado, e a senha de autenticação. Na linha 17 utilizamos o comando "*Blynk.begin(auth, ssid, pass)*" que tem a função de conectar o dispositivo a rede Wi-Fi e ao Blynk IoT dentro da função "*setup( )*".

Dentro da função "*loop( )*" destaca-se o comando "*Blynk.run( )*", na linha 20, que mantém a comunicação com a plataforma Blynk IoT ativa e aguardando a interação do usuário por meio das interfaces gráficas anteriormente criadas. As funções "*BLYNK_WRITE( )*" são fornecidas pelas bibliotecas do Blynk e quando incluídas no código, podem ser sobrescritas para executar as ações desejadas nos pinos virtuais que são passados como parâmetros.

```
2  #include <ESP8266WiFi.h>
3  #include <BlynkSimpleEsp8266.h>
4  #include <Servo.h>
5
6  #define BLYNK_TEMPLATE_ID "ID_DO_TEMPLATE" //Configuração do Blynk IoT
7  #define BLYNK_TEMPLATE_NAME "NOME_APLICACAO"
8  #define BLINK_AUTH_TOKEN "CHAVE_INFORMADA_PELO_BLYNK"
9  #define BLYNK_PRINT Serial
10
11 //Criação de variáveis globais para conexão na rede Wi-Fi
12 char auth[] = BLYNK_AUTH_TOKEN;
13 char ssid[] = "NOME_DA_REDE";
14 char pass[] = "SENHA_DA_REDE";
15
16 setup() { /*Demais comandos da aplicação da seção de Controle local (OMITIDO)*/
17   Blynk.begin(auth, ssid, pass); } //Conectar ao Wi-Fi e a Plataforma
18
19 loop() { /*Demais comandos da aplicação da seção de Controle local (OMITIDO)*/
20   Blynk.run(); } //Mantém a comunicação com o Blynk
21
22 BLYNK_WRITE(V0) {    /* Pino virtual V0 está vínculado ao servo0 */ }
23 BLYNK_WRITE(V1) {    /* Pino virtual V1 está vínculado ao servo1 */ }
24 BLYNK_WRITE(V2) {    /* Pino virtual V2 está vínculado ao servo2 */ }
25 BLYNK_WRITE(V3) {    /* Pino virtual V3 está vínculado ao servo3 */ }
```

Figura 6. Comandos para vincular a aplicação a plataforma Blynk IoT.

3.3. Controle Remoto por Voz

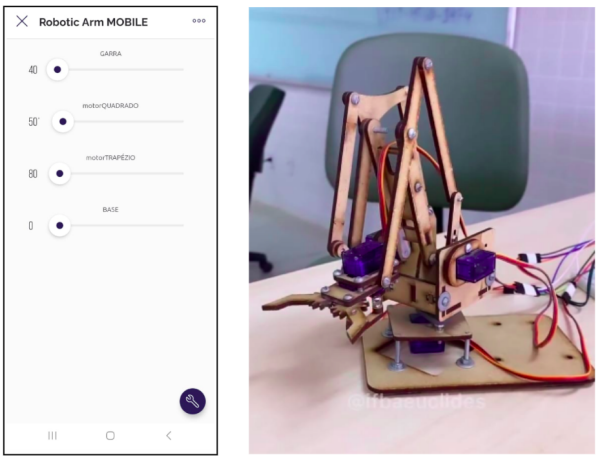
Para a implementação desta nova funcionalidade, foi utilizado o mesmo circuito físico apresentado nas seções anteriores. Além disso, foi realizada a configuração das plataformas *IFTTT*, incluindo o recurso do *Webhooks*, e *Blynk IoT* para reconhecerem os comandos de voz e realizarem as ações correspondentes.

É importante destacar que, para cada comando de voz adicionado à aplicação, um novo pino virtual foi criado na plataforma do *Blynk*. A variação do estado desse

novo pino é usada para disparar eventos no ESP8266. No código embarcado do ESP8266 acrescentou-se mais quatro funções *BLYNK_WRITE()* associadas aos quatro pinos virtuais criados para cada comando de voz. Para um melhor entendimento da utilização das tecnologias mencionadas e as ações desencadeadas por cada uma delas, segue o esquema que apresenta os passos que são executados após um comando de voz ser emitido até a conclusão no ESP8266:

1. O usuário utiliza o Google Assistente para passar um comando de voz;
2. O Assistente Virtual reconhece que o usuário emitiu um comando de voz e envia uma notificação para o IFTTT informando a frase dita.
3. O IFTTT concentra os condicionais de verificação dos comandos de voz. Se a frase for reconhecida em uma das condições pré-estabelecidas, é lançada uma URL com parâmetros para modificar os valores dos novos pinos virtuais na plataforma Blynk, por meio da tecnologia Webhooks.
4. A tecnologia Webhooks permite a comunicação entre duas aplicações em tempo real, garantindo que a ação enviada pelo IFTTT seja executada.
5. A plataforma Blynk recebe os parâmetros via URL, os identifica e altera os valores do pino virtual associado;
6. Os novos valores dos pinos virtuais são enviados para o ESP8266 que os converte em movimento de acordo com a programação pré-estabelecida.

Na Figura 7a é mostrada uma captura de tela do aplicativo desenvolvido para controlar o braço robótico via *smartphone*, e na Figura 7b é apresentada uma fotografia da versão final do braço desenvolvido.



(a) (b)

Figura 7. (a) Aplicativo de controle desenvolvido. (b) Foto do braço.

4. CONCLUSÃO

Este trabalho apresentou o desenvolvimento de um projeto de um braço robótico que conta com três diferentes mecanismos de controle, sendo eles: o controle local, o controle remoto e o controle por voz. Para sua implementação, foi utilizado um

ESP8266 NodeMCU em conjunto com um módulo codificador, que intermediou a troca de sinais entre o controle local e o NodeMCU.

O projeto apresentou excelentes resultados, apesar das diversas dificuldades enfrentadas durante sua implementação. Todas as premissas estabelecidas no início do projeto foram não só cumpridas, como também ampliadas e aprimoradas. Com isso, identificou-se um grande potencial de aperfeiçoamento, e para continuar a evolução do projeto, alguns objetivos podem ser elencados, tais como: 1) Aprimoramento do mecanismo de controle por voz, visando sua maior eficiência e precisão; 2) Integração de novas funcionalidades ao projeto, como a capacidade de realizar movimentos mais complexos por meio de um mecanismo de controle utilizando gestos; 3) Desenvolvimento de um mecanismo de segurança, a fim de garantir a integridade física do usuário em caso de falhas ou acidentes.

Dessa forma, o projeto apresenta grande potencial para contribuir com o avanço da robótica e da automação, possibilitando a criação de dispositivos mais eficientes e seguros para as mais diversas áreas.

REFERÊNCIAS BIBLIOGRÁFICAS

Almeida, A. L. D. (2022). Robotic ARM-braço robótico controlado remotamente via smartphone android.

Blynk. (2023). Documentation. https://docs.blynk.io/en/. [Online: acesso em 4-Janeiro-2023].

Calado, A. (2022). Entenda o que é um webhook e como ele funciona. https://rockcontent.com/br/blog/o-que-e-um-webhook/. [Online: acesso em 11-Janeiro-2023].

Datasheet do 74HC148. (1992). https://pdf1.alldatasheet.com/datasheet-pdf/view/23041/STMICROELECTRONICS/74HC148.html. [Online: acesso em 25-Abril-2023].

de Oliveira, S. (2017). Internet das coisas com ESP8266, Arduino e Raspberry PI. Novatec Editora.

Fornarolo, G. C., e Bacarin, V. F. (2020). Garra Robótica Controlada por Comando de Voz (Doctoral dissertation, [sn]).

Google. (2023). Google Assistant. https://assistant.google.com/intl/pt_br/platforms/phones/#get-answers. [Online: acesso em 9-Janeiro-2023].

Gray, B. (2014). MeArm V0.4 - Pocket Szied Robot Arm. https://www.thingiverse.com/thing:360108/files. [Online: acesso em 10-Maio-2023].

Guimarães, F. (2019). Codificador e decodificador – ED. https://mundoprojetado.com.br/codificador-e-decodificador-aula-7-ed/. [Online: acesso em 25-Abril-2023].

Matarić, M. J. (2014). Introdução à robótica. Editora Blucher.

Parihar, Y. S. (2019). Internet of things and nodemcu. journal of emerging technologies and innovative research, p. 1085-1088.

Salinas Castillo, M. E. (2022). Implementación de aplicaciones IoT usando el ESP32 y la plataforma NodeRED con los servicios IFTTT, Webhook Relay y Google Home.