

INSTITUTO FEDERAL DE EDUCAÇÃO,  
CIÊNCIA E TECNOLOGIA DA PARAÍBA

COORDENAÇÃO DO CURSO SUPERIOR DE  
TECNOLOGIA EM AUTOMAÇÃO INDUSTRIAL



RODRIGO TOMAZ PEDROSA

**INTEGRANDO ASSISTENTE PESSOAL ALEXA E APLICATIVO DE  
CELULAR BLYNK PARA O CONTROLE DO ESP8266 NODEMCU EM  
APLICAÇÕES DE AUTOMAÇÃO EM GERAL**

CAJAZEIRAS  
2021

RODRIGO TOMAZ PEDROSA

**INTEGRANDO ASSISTENTE PESSOAL ALEXA E APLICATIVO DE CELULAR  
BLYNK PARA O CONTROLE DO ESP8266 NODEMCU EM APLICAÇÕES DE  
AUTOMAÇÃO EM GERAL**

Trabalho de Conclusão de Curso submetido  
à Coordenação do Curso de Tecnologia em  
Automação Industrial do Instituto Federal de  
Educação, Ciência e Tecnologia da Paraíba,  
como parte dos requisitos para a obtenção do  
grau de Tecnólogo em Automação Industrial

Orientador: Prof. Dr. Raphael Maciel De  
Sousa

Cajazeiras  
2021

IFPB /Campus Cajazeiras  
Coordenação de Biblioteca  
Catalogação na fonte: Daniel Andrade CRB-15/593

P372i

Pedrosa, Rodrigo Tomaz

Integrando assistente pessoal Alexa e aplicativo de celular Blynk para o controle do Esp8266 Nodemcu em aplicações de automação em geral / Rodrigo Tomaz Pedrosa; orientador Raphaell Maciel de Sousa.- 2021.

54 f. : il.

Orientador: Raphaell Maciel de Sousa.

TCC (Tecnólogo em Automação Industrial) – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2021.

1. Automação residencial 2. Amazon Alexa 3. Comando de voz 4. VoiceFlow I. Título

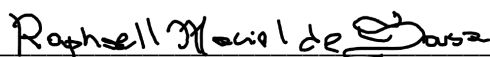
CDU 681.5(0.067)

RODRIGO TOMAZ PEDROSA

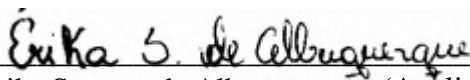
**INTEGRANDO ASSISTENTE PESSOAL ALEXA E APLICATIVO DE CELULAR  
BLYNK PARA O CONTROLE DO ESP8266 NODEMCU EM APLICAÇÕES DE  
AUTOMAÇÃO EM GERAL**

Trabalho de Conclusão de Curso submetido  
à Coordenação do Curso de Tecnologia em  
Automação Industrial do Instituto Federal de  
Educação, Ciência e Tecnologia da Paraíba,  
como parte dos requisitos para a obtenção do  
grau de Tecnólogo em Automação Industrial

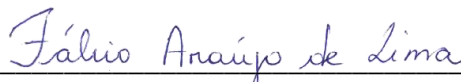
**BANCA EXAMINADORA**



Prof. Dr. Raphael Maciel De Sousa - (Orientador)  
Unidade Acadêmica de Indústria do IFPB



Profa. Me. Érika Spencer de Albuquerque - (Avaliador externo)  
Unidade Acadêmica de Indústria do IFPB.



Prof. Dr. Fábio Araújo de Lima - (Avaliador interno)  
Unidade Acadêmica de Indústria do IFPB.

Cajazeiras, 07 de outubro de 2021

*“Nada temos a temer quanto ao futuro, a menos que nos esqueçamos como Deus tem nos conduzido no passado.”*

*(Ellen G. White)*

## **DEDICATÓRIA**

Dedico este trabalho em especial aos meus pais Marconio Pedrosa e Luzenira Tomaz de Aquino Pedrosa pela dedicação e apoio em todos os momentos.

Também, a minha querida e amada esposa Bianca de Moraes Cruz Tomaz, por todo o companheirismo, dedicação e amor.

## **AGRADECIMENTOS**

Agradeço à Deus por conduzir os meus passos e me conceder a coragem e saúde necessárias para a conclusão do curso. Sei que Ele esteve ao meu lado, me concedendo sabedoria, paciência e todos os recursos dos quais precisei para minha formação.

Agradeço à minha esposa Bianca Tomaz e aos meus pais Marconio Pedrosa e Luzenira Tomaz, por todo o apoio concedido a cada dia.

Agradeço às minhas tias Luzinete, Luciana e Elza, ao Rogério, aos amigos e demais familiares pelo acolhimento e carinho.

Ao Instituto Federal da Paraíba, IFPB, pela oportunidade de realização de trabalhos em minha área de atuação e pelos diversos momentos de aprendizado.

Agradeço ao meu Orientador, Dr. Raphael Maciel De Sousa por se colocar à disposição e me auxiliar na construção deste trabalho.

## RESUMO

A automação residencial vem se desenvolvendo a cada dia, de modo que é possível encontrar dispositivos já prontos para aplicações de domótica. Entretanto, alguns desses dispositivos presentes no mercado possuem padronização e pouca flexibilidade para projetos mais complexos e personalizados.

Neste trabalho, é proposto o desenvolvimento da integração de dispositivos eletrônicos para aplicação em projetos de automação em geral, dispositivos como a assistente pessoal *Echo Dot* Alexa, ESP8266 NodeMCU, sensores de temperatura, umidade e chuva e a carga que será representada por um *LED (Light emitting diode)*. Estes elementos estarão conectados à internet por meio do servidor do aplicativo *Blynk*, que possibilitará o acompanhamento em tempo real dos dados referentes ao projeto na tela de um smartphone. Além disso, os parâmetros do projeto, temperatura, umidade, chuva e a carga podem ser acessados por meio de comandos de voz pela assistente pessoal da *Amazon* para a verificação dos valores dos sensores e para o acionamento e o desligamento da carga.

**Palavras-Chave:** Automação residencial. *Amazon* Alexa. Comando de voz. *VoiceFlow*.



## ABSTRACT

Home automation is developing every day, so it is possible to find ready-made devices for home automation applications. However, some of these devices on the market have standardization and little flexibility for more complex and customized projects.

In this work, it is proposed the development of the integration of electronic devices for application in automation projects in general, devices such as the personal assistant Echo Dot Alexa, ESP8266 NodeMCU, temperature, humidity and rain sensors and the load that will be represented by an LED (Light emitting diode). These elements will be connected to the internet through the Blynk application server, which will enable real-time monitoring of project data on a smartphone screen. In addition, the design parameters, temperature, humidity, rain and load can be accessed via voice commands by Amazon's personal assistant to verify sensor values and to turn the load on and off.

**Keywords:** Home automation. *Amazon Alexa*. Voice command. *VoiceFlow*.

## LISTA DE ILUSTRAÇÕES

Figura 1: Estrutura do <i>Blynk</i>	14
Figura 2: <i>Echo Dot</i> 3ª Geração	16
Figura 3: <i>Voiceflow</i>	19
Figura 4: ESP8266 NodeMCU	20
Figura 5: Composição do NodeMCU	21
Figura 6: Estrutura do projeto	24
Figura 7: Seção de bibliotecas	25
Figura 8: Seção de definições	26
Figura 9: Tela de início do monitor serial	26
Figura 10: Comandos iniciais complementares	27
Figura 11: Configurações de servidor e rede	27
Figura 12: Função void Chuva ( )	28
Figura 13: Função void Temp ( )	29
Figura 14: Função BLYNK WRITE (V1)	29
Figura 15: Função void setup ( )	30
Figura 16: Função void loop ( )	30
Figura 17: Tela inicial do <i>Voiceflow</i>	31
Figura 18: Grupos de blocos do <i>Voiceflow</i>	32
Figura 19: Estrutura da Skill desenvolvida no <i>Voiceflow</i>	33
Figura 20: Bloco de Opções (3)	34
Figura 21: Configurações do bloco de API	36
Figura 22: Condição If (sensor de chuva)	38
Figura 23: Configurando o aplicativo <i>Blynk</i>	40
Figura 24: Configurando as ferramentas do projeto <i>Blynk</i>	41
Figura 25: Esquema de ligação elétrica do projeto	42
Figura 26: Projeto integrado em funcionamento	44
Figura 27: Teste prático do projeto em funcionamento	45
Figura 28: Vídeo apresentando o projeto em funcionamento	51

## **LISTA DE ABREVIATURAS**

*API - Application Programming Interfaces*  
*IDE - Integrated Development Environment*  
*IFTTT - If This and This Then That*  
*iOS - iPhone Operating System*  
*IOT - Internet of Things*  
*IP - Internet Protocol*  
*LED - Light Emitting Diode*  
*URL - Uniform Resource Locator*  
*USB - Universal Serial Bus*

## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>12</b>
<b>2 REVISÃO DE LITERATURA.....</b>	<b>13</b>
<b>2.1 BLYNK.....</b>	<b>13</b>
<b>2.1.1 Aplicativo <i>Blynk</i>.....</b>	<b>14</b>
<b>2.1.2 Servidor <i>Blynk</i> .....</b>	<b>14</b>
<b>2.1.3 Bibliotecas <i>Blynk</i>.....</b>	<b>15</b>
<b>2.2 AMAZON ALEXA .....</b>	<b>15</b>
<b>2.2.1 <i>Echo</i> .....</b>	<b>16</b>
<b>2.2.2 <i>Alexa Cloud</i> (Nuvem) .....</b>	<b>17</b>
<b>2.2.3 <i>Amazon Skill Servers</i> (Servidores de habilidades) .....</b>	<b>17</b>
<b>2.2.4 <i>Third-Party Skill Servers</i> (Servidores de habilidades terceirizados) .....</b>	<b>18</b>
<b>2.2.5 <i>Echo Application</i> (Aplicativo) .....</b>	<b>18</b>
<b>2.3 VOICEFLOW .....</b>	<b>18</b>
<b>2.4 ESP8266 NODEMCU.....</b>	<b>19</b>
<b>2.5 TRABALHOS CORRELATOS .....</b>	<b>22</b>
<b>3 MÉTODOS E TÉCNICAS DE DESENVOLVIMENTO .....</b>	<b>24</b>
<b>4 ANÁLISE DOS RESULTADOS .....</b>	<b>44</b>
<b>CONCLUSÃO.....</b>	<b>48</b>
<b>REFERÊNCIAS .....</b>	<b>49</b>
<b>APÊNDICE A – ATUAÇÃO DA ALEXA (VÍDEO).....</b>	<b>51</b>
<b>APÊNDICE B – CÓDIGO FONTE DO PROJETO .....</b>	<b>52</b>

## 1 INTRODUÇÃO

Muitos dispositivos eletrônicos de hoje em dia já vêm programados e configurados com um certo grau de “inteligência”, ou seja, são capazes de realizar algumas ações de forma autônoma. Segundo Paixão et al (2019), a IoT (*Internet of Things* ou Internet das Coisas) está evoluindo rapidamente, como também, a rede de periféricos que contém tecnologia de comunicação embarcada. Dessa forma, está cada vez mais comum encontrarmos dispositivos conectados entre si.

O próximo passo dessa evolução são os dispositivos que permitem interação por comandos de voz. As assistentes virtuais já são uma realidade em quase todos os *smartphones* dos grandes fabricantes do mercado (PAIXÃO et al, 2019).

E não somente os *smartphones*, mas agora já é possível ter acesso a dispositivos eletrônicos que possuem a tecnologia do comando de voz. Conforme o site Homesales (2017) destacou, até alguns anos atrás, os dispositivos domésticos eram limitados ao controle por toque ou controle remoto. Desde o lançamento de dispositivos de automação residencial, como *Amazon Echo* e *Google Home*, a tecnologia de reconhecimento de voz progrediu significativamente e mudou a maneira como interagimos com os aparelhos comuns em casa.

Assim sendo, com essa revolução tecnológica é cada vez mais recorrente o desejo pelo conceito de casas automatizadas ou domótica. Domótica é definida como implementação de tecnologias capazes de prover o gerenciamento dos diversos dispositivos presentes em um ambiente residencial (SGARBI, 2006).

O artigo da Homesales (2017) também apontou que, da iluminação a sistemas de segurança, aquecimento e refrigeração, entretenimento doméstico e entrega de conteúdo; a tecnologia inteligente e a automação residencial oferecem um nível de controle e interatividade nunca visto antes.

O objetivo deste trabalho é desenvolver uma aplicação integrada de sensores, e uma carga genérica, de modo que seja possível acionar ou desativar a carga e acessar os dados gerados pelos referidos sensores por meio de um aplicativo. Além disso, será realizada a integração do protótipo desenvolvido com a assistente da Amazon, Alexa, de modo que seja possível realizar o controle dos dispositivos por meio de comandos de voz.

## 2 REVISÃO DE LITERATURA

Este trabalho envolve dispositivos que comumente são utilizados em projetos de automação residencial, projetos de escolas, faculdades, dentre outras aplicações. Os componentes são: sensores, *LED*, microcontrolador ESP8266 NodeMCU, assistente pessoal, etc. No momento em que os referidos dispositivos são integrados ao aplicativo *Blynk*, as possibilidades para novas aplicações aumentam, pois, o aplicativo tem a capacidade de atender a diversos tipos de projetos com seus recursos disponíveis. Os projetos se tornam mais interessantes quando se pode interagir com eles por meio de um smartphone ou por comandos de voz, por exemplo.

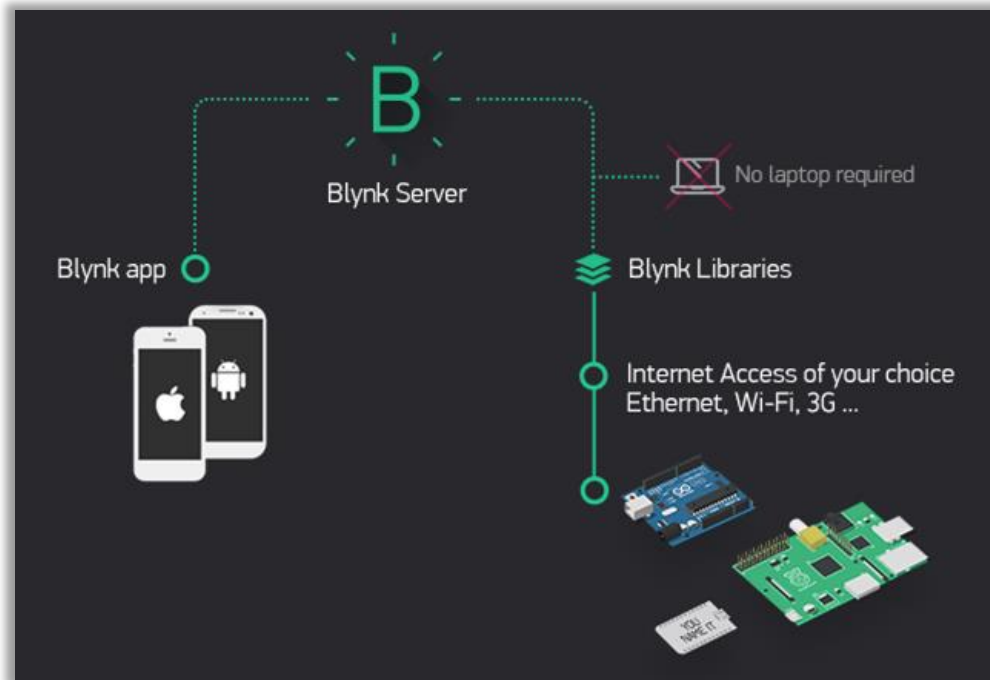
A seguir, estão destacados os conceitos dos elementos que foram explorados no projeto, a partir dos mesmos foi possível compreender melhor o funcionamento dos sistemas empregados no trabalho, como também desenvolver os programas, as instruções e a estrutura do protótipo.

A plataforma *Blynk* foi utilizada no projeto para realizar a comunicação via Wi-Fi do NodeMCU com o aplicativo na tela do *smartphone*, e executar os comandos e instruções atribuídas ao trabalho.

### 2.1 *Blynk*

*Blynk* é um serviço baseado em um aplicativo personalizável que permite controlar remotamente um *hardware* programável, bem como reportar dados do hardware ao aplicativo. Desta forma, é possível construir interfaces gráficas de controle de forma rápida e intuitiva e que interagem com mais de 400 placas de desenvolvimento, em sua maioria baseadas em Arduino (SERRANO, 2018).

O serviço da plataforma *blynk* é composto por três partes principais, o aplicativo, o servidor e suas bibliotecas, que formam sua estrutura e possibilitam seu funcionamento, conforme a Figura 1 a seguir.

Figura 1: Estrutura do *Blynk*

Fonte: [www.embarcados.com.br/wp-content/uploads/2018/05/image3.png](http://www.embarcados.com.br/wp-content/uploads/2018/05/image3.png)

O *Blynk* foi desenvolvido para ser utilizado em projetos IoT que é um termo utilizado para descrever a forma como objetos do mundo real permanecem conectados à rede e podem ser acessados através da internet (OLIVEIRA, 2017).

### 2.1.1 Aplicativo *Blynk*

O aplicativo *Blynk* é um *software* disponível para Android e iOS (sistema operacional móvel da Apple) que permite ao usuário criar aplicações que interagem com o hardware. Por meio de um espaço próprio para cada projeto, o usuário pode inserir *Widgets* (ferramentas) que implementam funções de controle (como botões, controles deslizantes e chaves), notificação e leitura de dados do *hardware* (exibindo em *displays*, gráficos e mapas) (SERRANO, 2018).

O aplicativo também possibilita ao usuário criar interfaces de controle de forma simples, onde é necessário apenas arrastar os *widgets* e em poucos passos fazer a configuração (OLIVEIRA, 2017).

### 2.1.2 Servidor *Blynk*

O *Blynk Server*, onde toda comunicação entre o aplicativo e o *hardware* do usuário é realizada por meio do *cloud* (*nuvem*) *Blynk*, o servidor é responsável por transmitir os dados ao *hardware*, armazenar estados do aplicativo e do *hardware* e também armazenar dados de

sensores lidos pelo dispositivo físico mesmo se o aplicativo estiver fechado. Vale ressaltar que os dados armazenados no servidor *Blynk* podem ser acessados externamente por meio de uma *API HTTP*, o que abre a possibilidade de utilizar o servidor para armazenar dados gerados periodicamente, tais como parâmetros da leitura realizada por sensores de temperatura ou umidade, por exemplo (SERRANO, 2018).

O servidor *Blynk* é responsável por todas as comunicações entre o dispositivo móvel e a plataforma. Você pode usar o *Blynk Cloud* ou executar um servidor *Blynk* em sua máquina local. Esse servidor pode trabalhar com diversos dispositivos, inclusive pode rodar em um Raspberry Pi (OLIVEIRA, 2017).

### 2.1.3 Bibliotecas *Blynk*

*Blynk Libraries*, finalmente, ao lado do *hardware* estão as bibliotecas *Blynk* para diversas plataformas de desenvolvimento. Essas bibliotecas são responsáveis por gerir toda a conexão do *hardware* com o servidor da plataforma e gerir as requisições de entrada e saída de dados e comandos. A forma mais fácil e rápida é utilizá-la como bibliotecas Arduino, no entanto, é possível obter versões da biblioteca para *Linux*, *Raspberry Pi*, *Python*, *Lua*, entre outras (SERRANO, 2018).

Há bibliotecas para todas as plataformas mais populares e compatíveis com o *Blynk*, permitindo assim, a comunicação com o servidor na nuvem ou local, processando todos os comandos de entrada e saída (OLIVEIRA, 2017).

Outro dispositivo utilizado no projeto foi o *Echo Dot* 3ª geração da *Amazon*, um equipamento eletrônico que utiliza a assistente pessoal Alexa para interação com o usuário por meio da fala. Sua estrutura está apresentada a seguir.

## 2.2 Amazon Alexa

Popularmente conhecido como ‘Alexa’, o *Echo Dot* é o *smart speaker* (alto-falante inteligente) de maior sucesso da *Amazon*, o aparelho é controlado por voz com ajuda de uma assistente virtual chamada Alexa, e pode ser utilizado em qualquer ambiente (CAMIOTTO, 2021).

A Alexa, da *Amazon*, é comercializada como uma assistente pessoal digital inteligente, os usuários deste serviço interagem com ele usando os dispositivos *Amazon Echo* ou *Amazon Echo Dot*, esses dispositivos recebem os comandos de voz dos usuários e enviam para o servidor



da *Amazon*. Usando esses dispositivos, um usuário pode realizar uma variedade de tarefas (HAACK, 2017).

O usuário pode pedir músicas, notícias, informações e muito mais, além disso, pode ligar para amigos e familiares e controlar dispositivos compatíveis com funcionalidade de casa inteligente (CAMIOTTO, 2021).

### 2.2.1 *Echo*

O *Echo Dot* é o *smart speaker* da *Amazon* capaz de controlar os dispositivos inteligentes da casa, realizar ligações, chamar um Uber, pedir pizza, entre outras opções. Tudo isso graças à Alexa, que recebe o comando de voz do dono e está programada para entender o contexto (CARDOSO, 2018).

O dispositivo *Echo* (ou *Echo Dot*) Figura 2 abaixo, fornece uma interface baseada em fala para o usuário, os comandos falados, as perguntas, o *Echo* responde com a fala. O dispositivo está sempre ouvindo, e apenas quando detecta a palavra de ativação ("Alexa", por padrão), ele transmite o áudio subsequente (presumivelmente o comando ou pergunta do usuário) para o serviço de nuvem Alexa. Em seguida, aguarda para receber uma resposta baseada em texto da nuvem, para então, renderizar e reproduzir o áudio em forma de uma resposta falada (HAACK, 2017).

Figura 2: Echo Dot 3ª Geração



Fonte: [www.amazon.com.br](http://www.amazon.com.br)

Na parte de cima do aparelho, estão disponíveis quatro botões de controle, sendo dois de volume, um botão de ação e um último para desligar o microfone. Ao redor da parte superior,

existe um círculo de luz de *LED* que indica quando a Alexa entendeu o comando de voz e está respondendo ao usuário (CARDOSO, 2018).

Apesar de pequeno, o *Amazon Echo Dot* traz de fábrica as mesmas funções básicas de qualquer dispositivo da família, como o *Echo* e o *Echo Plus*. Esse assistente consegue, por exemplo, controlar os dispositivos inteligentes da casa, quando disponíveis. Assim, o usuário pode ligar ou desligar a luz do ambiente apenas com a voz, fechar as persianas e controlar a temperatura, por exemplo. O assistente tem os microfones espalhados que captam sons de todas as direções (CARDOSO, 2018).

### **2.2.2 Alexa Cloud (Nuvem)**

O serviço de nuvem Alexa recebe os áudios de dispositivos *Echo* e é responsável por lidar com a fala, reconhecimento e mapeamento de comandos de voz para intenções e dados associados. Com base em determinada intenção, o serviço de nuvem Alexa delega o comando a um servidor de habilidades, operado pela *Amazon* para suas habilidades integradas ou para habilidades de aplicativos de terceiros. Quando o serviço de nuvem Alexa recebe uma resposta do servidor de habilidade, ele transmite isso de volta para o dispositivo *Echo* (HAACK, 2017).

### **2.2.3 Amazon Skill Servers (Servidores de habilidades)**

Habilidades são essencialmente tarefas que o serviço Alexa pode realizar para um usuário como, verificar o tempo ou pedir uma pizza. Algumas dessas habilidades são oferecidas pela *Amazon* e vêm como uma funcionalidade integrada para todos os usuários dos dispositivos da fabricante, e esses são gerenciados pelos próprios servidores de habilidade da *Amazon*, as solicitações podem ser feitas para esses servidores pelo serviço de nuvem, e esses servidores, em seguida, fornecem respostas, em ambos os formatos predeterminados (HAACK, 2017).

Essas habilidades são chamadas de *skills*. Uma *skill* é uma tarefa personalizada para cada aplicação. Após efetuado um comando de voz, ele é interpretado e é executado por servidores da própria *Amazon*, onde são executados conforme programados (NETO, 2018).

A Alexa está sempre aprendendo e adicionando novas *Skills*, como jogos, notícias e muito mais. Com 4 microfones de longo alcance, a Alexa ouve até do outro lado do cômodo (CAMIOTTO, 2021).

#### **2.2.4 Third-Party Skill Servers (Servidores de habilidades terceirizados)**

Outras habilidades são oferecidas por desenvolvedores terceirizados, como aplicativos de pedido de comida ou compartilhamento de carona. Essas habilidades são administradas pelos próprios desenvolvedores terceirizados, e o processo de solicitação / resposta é quase idêntico ao dos servidores de habilidades da *Amazon* (HAACK, 2017).

Aqui também entram as diversas aplicações em projetos de automação, seja para controle de lâmpadas, equipamentos ou eletrodomésticos. Serviços como o *Voiceflow* ou *IFTTT* (*If This and This Then That*) são boas opções para o desenvolvimento dessas habilidades para a Alexa.

#### **2.2.5 Echo Application (Aplicativo)**

A *Amazon* oferece um aplicativo móvel que é usado para gerenciar o serviço Alexa, com este aplicativo, os usuários podem ajustar as configurações, ver todas as interações anteriores com a Alexa e fornecer feedback sobre a qualidade dessas interações. Quando um usuário interage com a assistente, o serviço de nuvem Alexa empurra cartões para o aplicativo móvel, que contêm as informações relevantes sobre a interação a ser exibida no aplicativo. Observe que essas informações incluem uma gravação da pergunta do usuário ou comandos para seu dispositivo *Echo* (HAACK, 2017).

Outra ferramenta utilizada no desenvolvimento do trabalho foi o *Voiceflow*, um site que possibilita a elaboração de habilidades personalizáveis, capazes de interagir com os dispositivos da *Amazon*, por meio de sua plataforma. Sua estrutura está apresentada a seguir.

### **2.3 VoiceFlow**

O *Voiceflow* é uma ferramenta online lançada em 2018 que permite desenvolver *skills* para Alexa e *actions* para o *Google* Assistente sem que a pessoa tenha qualquer conhecimento de programação. O *Voiceflow* hospeda mais de 200 milhões de interações por ano, com cerca de sete mil aplicativos de voz desenvolvidos (NEWVOICE, 2021).

A plataforma possibilita que de maneira visual e sem precisar saber programar, que você desenhe, prototipe e publique *skills* ou *Actions* (SANTOS, 2020).

Figura 3: *Voiceflow*

Fonte: [cdn.dribbble.com/users/1260346/screenshots/6583164/voiceflow\\_01\\_4x.png](https://cdn.dribbble.com/users/1260346/screenshots/6583164/voiceflow_01_4x.png)

O *Voiceflow* funciona com uma estrutura de blocos, que incluem funções como: Falar, com esse bloco você pode definir o que será respondido pelo assistente; Escolha, nesse bloco, é possível estabelecer as opções de condições que podem ser selecionadas pelo usuário, também é possível adicionar blocos lógicos, dentre outras funções.

As coleções de ferramentas do *Voiceflow* permitem que os desenvolvedores projetem e criem aplicativos de voz para qualquer sistema compatível. Os *designs* podem ser prototipados e preparados para publicação em várias plataformas, atuando como assistentes de voz autônomos, capazes de se adaptar ao ambiente (SCHWARTZ, 2021).

O *Voiceflow* reuniu muitas de suas atualizações em uma reformulação abrangente de sua plataforma chamada *Voiceflow V2*, com foco em permitir que os criadores implantem aplicativos de voz construídos para uma plataforma para publicá-los em outras (SCHWARTZ, 2021).

Entre os objetivos da nova versão está a de testar aplicativos para qualquer assistente de voz, além da Alexa e do *Google Assistente*. De acordo com o CEO da *Voiceflow*, Braden Ream, existirá uma colaboração em tempo real entre toda a equipe que trabalha em um mesmo projeto (NEWVOICE, 2021).

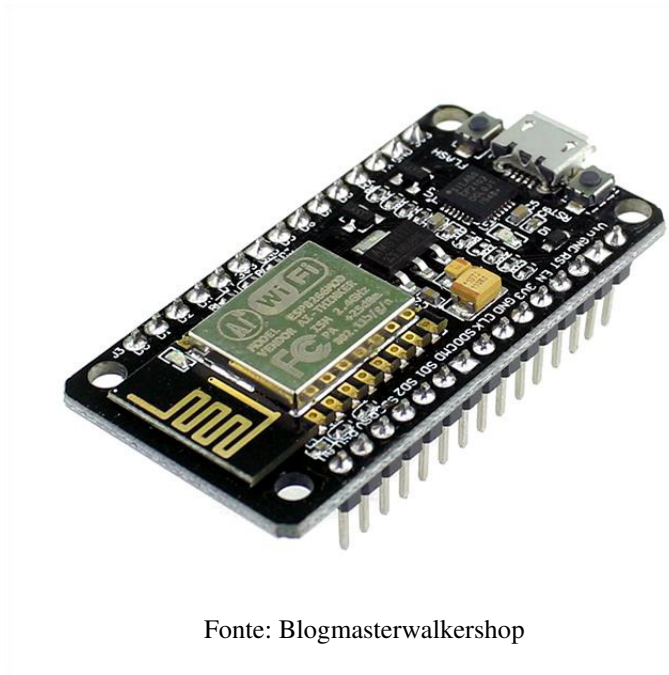
## 2.4 ESP8266 NodeMCU

Segundo Oliveira (2016), hoje, para falar de qualquer plataforma embarcada / microcontrolada é necessário mencionar o Arduino antes de qualquer coisa. Isso porque desde seu lançamento em 2005, ele é a plataforma microcontrolada favorita dos *makers* e *hobbistas*. Mas o que algumas pessoas iniciantes no mundo IoT não sabem, é que o Arduino tem um concorrente bem a altura e com algumas características singulares, que fazem desta plataforma uma opção atrativa para certos projetos; o ESP8266 NodeMCU, outra plataforma de

desenvolvimento muito empregada em muitos projetos assim como o Arduino (OLIVEIRA, 2016).

ESP8266 é o nome de um microcontrolador projetado pela *Espressif Systems*. Ele anuncia a si mesmo como uma solução de rede Wi-Fi independente, oferecendo-se como uma opção de microcontrolador com conexão à internet, também com capacidade de executar aplicações independentes (KOLBAN, 2016).

Figura 4: ESP8266 NodeMCU



Fonte: Blogmasterwalkershop

O NodeMCU, Figura 4, é uma plataforma *open source* da família ESP8266 criado para ser utilizado no desenvolvimento de projetos IoT. Esta placa foi iniciada em 2014 e é bastante interessante, pois, ao contrário de alguns módulos desta família que necessitam de um conversor *USB (Universal Serial Bus)* serial externo para que haja troca de informações entre computador e o módulo, o NodeMCU já vem com este conversor integrado (OLIVEIRA, 2016).

Esta plataforma é composta basicamente por um chip controlador (ESP8266 ESP-12E), um porta micro *USB* para alimentação e programação, conversor *USB* serial integrado e já possui Wi-Fi nativo (OLIVEIRA, 2016).

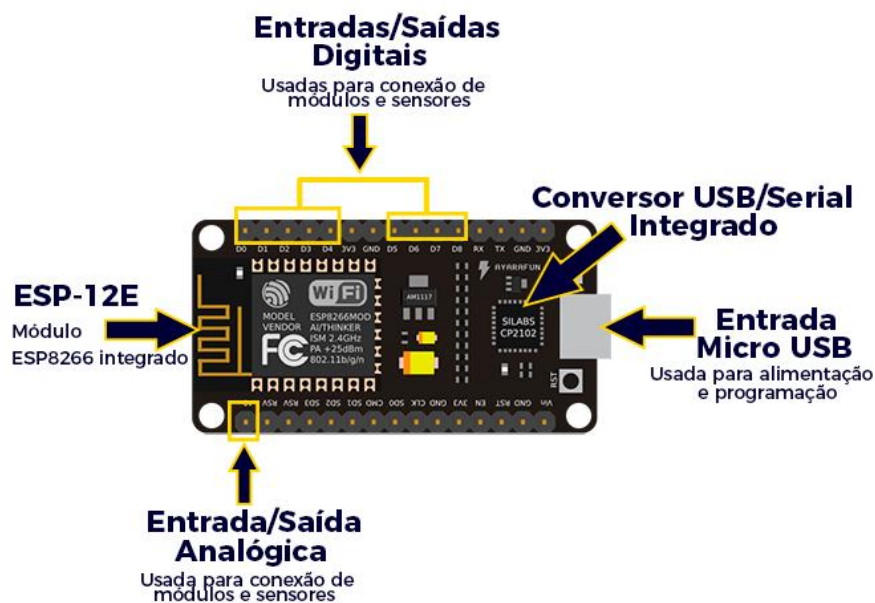
Conforme Oliveira (2016), abaixo, você pode ver as principais características do NodeMCU:

- Processador ESP8266-12E
- Arquitetura RISC de 32 bits
- Processador pode operar em 80MHz / 160MHz
- 4Mb de memória flash

- 64Kb para instruções
- 96Kb para dados
- Wi-Fi nativo padrão 802.11b/g/n
- Opera em modo AP, *Station* ou AP + *Station*
- Alimentação - 5VDC através do conector micro *USB*– Possui 11 pinos digitais
- Possui 1 pino analógico com resolução de 10 bits
- Pinos digitais, exceto o D0 possuem interrupção, PWM, I2C e *one wire*
- Pinos operam em nível lógico de 3.3V
- Pinos não tolerantes a 5V
- Possui conversor *USB Serial* integrado
- Programável via *USB* ou Wi-Fi (OTA)
- Compatível com a *IDE* do Arduino
- Compatível com módulos e sensores utilizados no Arduino

Na Figura 5 a seguir, você pode acompanhar uma breve descrição da composição da placa:

Figura 5: Composição do NodeMCU



Fonte: Blogmasterwalkershop

Conforme dito anteriormente, o NodeMCU possui características singulares que o fazem se destacar, como por exemplo seu baixo custo, suporte integrado a redes Wi-Fi, tamanho reduzido e baixo consumo de energia. Portanto, se você está desenvolvendo um projeto que necessite de comunicação entre dispositivos através de uma rede Wi-Fi, com certeza o NodeMCU te atenderá bem e você gastará bem menos do que gastaria ao utilizar o Arduino (OLIVEIRA, 2016).

Uma das grandes vantagens em utilizar plataformas baseadas no ESP8266, é a possibilidade de se programar utilizando a *IDE (Integrated Development Environment)* do Arduino. Assim como em outras placas da família ESP8266, o NodeMCU também é compatível com esse ambiente de desenvolvimento (OLIVEIRA, 2016).

## 2.5 Trabalhos correlatos

Nesta seção, estão apresentados trabalhos com propostas semelhantes ao deste trabalho atual, o qual incluem microcontroladores, assistente pessoal e até o aplicativo *Blynk* em projetos de automação residencial.

O trabalho: Automação Residencial de Baixo Custo, desenvolvido por Moura Júnior (2020), utiliza o aplicativo *Blynk* e o *Google Assistant* em seu projeto de IoT. Sua proposta consiste em desenvolver um dispositivo que, conectado à internet por meio da rede Wi-Fi, seja capaz de monitorar a umidade e temperatura do ambiente e controlar, via *smartphone*, três lâmpadas, por meio de toques na tela do celular e por comandos de voz (MOURA JUNIOR, 2020).

O projeto teve o funcionamento conforme o esperado, tornando possível acender e apagar todas as lâmpadas, abrir e fechar o portão da garagem e ligar e desligar a ventilação, além de monitorar a temperatura e umidade do ambiente, tudo isso feito através do *smartphone* com alguns toques na tela ou por comando de voz feitos ao *Google Assistant*. Com a finalização do projeto, foram obtidos resultados bastantes satisfatórios com relação ao uso o aplicativo *Blynk*, e a automação desenvolvida funcionou bem (MOURA JUNIOR, 2020).

O segundo trabalho: Protótipo de Automação Residencial Utilizando uma Assistente de Voz, desenvolvido por Neto (2018), tem como escopo criar um protótipo de automação residencial por comandos de voz utilizando a assistente pessoal da *Amazon*, Alexa. O projeto utiliza o microcontrolador Arduino Ameba para se conectar na Internet por meio de uma rede sem fio, controlar as luzes de uma casa, abrir uma porta por meio de um eletroímã e enviar dados de temperatura e umidade do ambiente para um servidor HTTP (NETO, 2018).

A assistente pessoal é clara ao falar e muito assertiva em identificar o comando do usuário. Além do mais, o tempo de resposta ao efetuar o comando para o aplicativo por voz é em torno de quatro segundos, o que significa um excelente tempo de resposta (NETO, 2018).

O comando efetuado por voz traz uma comodidade bem maior ao usuário, pois, não se faz necessário ter em mãos algum dispositivo para a aplicação tomar uma ação. Este tipo de funcionalidade pode abrir margem para inovações que vão além da domótica. Podemos tomar como

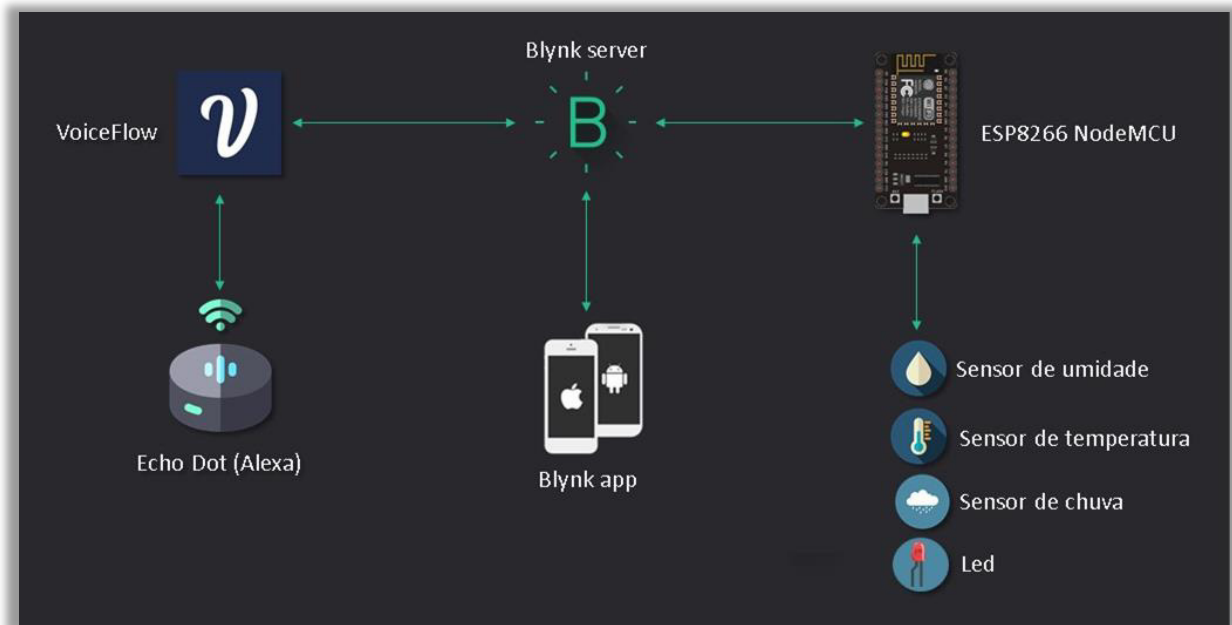
exemplo o ambiente industrial, onde o contato físico com algum dispositivo pode trazer algum perigo ao usuário. Logo, este pode tornar o trabalho mais prático e seguro (NETO, 2018).



### 3 MÉTODOS E TÉCNICAS DE DESENVOLVIMENTO

Este trabalho se propõe a criar a integração entre o dispositivo *Echo Dot* Alexa, o aplicativo *Blynk*, um ESP8266 NodeMCU e um conjunto de sensores e um led conforme a Figura 6, de modo que sirva como protótipo inicial e exemplar para aplicações de automação.

Figura 6: Estrutura do projeto



Fonte: Elaborado pelo autor

A ideia é integrar todos esses dispositivos, e a partir disso, trocar informações com o sistema tanto pelo app, quanto por comandos de voz. Os dados dos sensores poderão ser consultados por meio do aplicativo na tela do *smartphone* ou pelos comandos do *Echo Dot* desenvolvidos neste trabalho. Da mesma forma, também será possível ligar e desligar o led, que representa um dispositivo de saída. O conjunto de instruções e comandos são realizados utilizando o servidor *Blynk* para o aplicativo, e a plataforma *VoiceFlow* para o *Echo*.

O aplicativo *Blynk* permite que o usuário crie seu próprio “aplicativo” personalizável com uma variedade de funções para o controle da entrada e saída de dados, são botões, *displays*, medidores, blocos de instruções, eventos programáveis, *timers*, dentre outras possibilidades virtuais. Por isso, foi a opção ideal para o projeto, por possibilitar uma aplicação que seja distinta, em relação a outras opções.

O *VoiceFlow* por sua vez, permite ao usuário a elaboração de projetos que envolvam comandos de voz e dispositivos como a assistente pessoal da *Amazon*, por exemplo. Por meio dele, é possível criar programas de instruções também personalizáveis, para a interação com

outras plataformas, entre elas o *Blynk*. Seja para o envio ou consulta de dados em tempo real, com o *VoiceFlow*, é possível definir as respostas que serão apresentadas para os questionamentos feitos ao *Echo Dot*, também é possível acessar os servidores *Blynk* para o acionamento das saídas de microcontroladores e para a coleta dos dados de sensores.

Neste trabalho foi utilizado um *LED* como representação de carga, portanto, a mesma saída também poderia ser utilizada para o acionamento de lâmpadas, persianas automatizadas, portões eletrônicos, toldos, enfim, o que cada aplicação exigir. O projeto ainda inclui sensores de chuva, temperatura e umidade para a entrada de dados, e o microcontrolador utilizado é o ESP8266 NodeMCU.

Em seguida estão definidas as etapas de desenvolvimento para a elaboração do projeto:

1. Desenvolvimento do código com a programação para o ESP8266 baseado na plataforma *Blynk*;
2. Realização da integração do código com a assistente pessoal e elaboração de um ambiente de interação personalizado para os comandos por voz na plataforma *Voiceflow*.

O *software* utilizado para o desenvolvimento da programação é o *Arduino IDE*, que é um ambiente integrado de desenvolvimento, por meio dele é possível realizar a comunicação com o ESP8266 NodeMCU utilizando a linguagem de programação C/C++ por meio de seus comandos e instruções.

O primeiro passo para o desenvolvimento do código é a adição das bibliotecas que irão executar conjuntos de instruções específicas para a aplicação desejada, seja a conexão com a rede, leitura dos sensores, etc. Neste projeto foram utilizadas as seguintes bibliotecas: *ESP8266WiFi.h*, *BlynkSimpleEsp8266.h*, *SimpleTimer.h* e *DHT.h*. Essas bibliotecas executam funções de conexão com a rede Wi-Fi, onde é possível adicionar a rede local e senha, conexão com o servidor *Blynk* para que o projeto esteja conectado ao serviço de nuvem da plataforma, tempo para interação em milésimos de segundo e leitura do sensor de temperatura e umidade *Dht11*, respectivamente, conforme a Figura 7 a seguir.

Figura 7: Seção de bibliotecas

```
// Bibliotecas do Projeto
#include <ESP8266WiFi.h>           // Configurações de conexão com a rede Wi-fi
#include <BlynkSimpleEsp8266.h>   // Configurações de conexão com o servidor blynk
#include <SimpleTimer.h>          // Funções de tempo e milésimos de segundo
#include <DHT.h>                  // Configurações dos sensores de temperatura e umidade
```

Fonte: Elaborado pelo autor

Definidas as bibliotecas, agora serão introduzidos os primeiros comandos de configurações para o projeto, são comandos para exibição do status do projeto, definição dos pinos do ESP8266 que serão utilizados e tipo de sensor, esses comandos estão demonstrados na Figura 8.

Figura 8: Seção de definições

```
#define BLYNK_PRINT Serial // Função p/ exibição de status do projeto
#define LED_Red 4 // Definindo Pino(D2) como saída p/ Led Vermelho
#define pinSensorD 12 // Definindo Pino(D6) p/ sensor Digital de chuva
#define DHTPIN 14 // Definindo Pino(D5) do ESP8266 p/ o sensor DHT
#define DHTTYPE DHT11 // Definindo tipo de sensor: DHT 11
```

Fonte: Elaborado pelo autor

O comando BLYNK\_PRINT Serial envia para o monitor serial do Arduino *IDE* o *status* do projeto em relação a conexão com a rede Wi-Fi e informações como o IP resultante dessa conexão, o *status* de conexão com o servidor *Blynk*, exibindo também a porta de acesso. Tudo isso conforme exibido na Figura 9 a seguir.

Figura 9: Tela do monitor serial iniciando

```
OIt,kŕ$`zŕxŕYIIY□ŕ0ŕŕAŕ[82] Connecting to brisa-1121540
[6312] Connected to WiFi
[6312] IP: 192.168.1.8
[6312]

  / _ ) / / _ _ _ _ _ / / _
 / _ / / / / / _ \ / ' _ /
 / _ _ / _ \ \ , / _ / / _ \ \
      / _ / v1.0.0 on ESP8266

[6387] Connecting to blynk-cloud.com:80
[6835] Ready (ping: 88ms).
```

Fonte: Elaborado pelo autor

Já os demais “#define” da Figura 8, são para nomeações referentes aos pinos do ESP8266, o “LED\_Red” foi definido para o pino 4 do ESP8266, que corresponde ao pino D2 na placa NodeMCU, que será utilizado para acender e apagar um Led vermelho do projeto. O “pinSensorD” foi definido para o pino 12, que corresponde ao pino D6 do NodeMCU, este será utilizado para o sensor de chuva que passa informações digitais, ou seja, 0 ou 1. O “DHTPIN” foi definido para o pino 14, correspondente ao pino D5 da placa, este será utilizado para a leitura do sensor de temperatura e umidade que informa valores referentes a seus parâmetros de medição. Por fim, o “DHTTYPE DHT11” está informando para a sua biblioteca o modelo ou

tipo de sensor que está sendo empregado, isso garante que a leitura seja feita de maneira apropriada, evitando distorções no valor final obtido.

Seguindo no código, estão apresentadas outras configurações iniciais para o projeto.

Figura 10: Comandos iniciais complementares

```
WidgetLED Led1(V8);           // Led Virtual na tela do Blynk (indicador de Chuva)
DHT dht(DHTPIN, DHTTYPE);    // Setando configurações junto a biblioteca
BlynkTimer timer;           // Atribuindo o tempo ao 'timer'
```

Fonte: Elaborado pelo autor

O comando “WidgetLED” apresentado na Figura 10, atribui o nome Led1 para o pino Virtual V8, um pino virtual serve como saída na tela do aplicativo *Blynk*, dessa vez, ao invés de um Led físico junto ao ESP8266, apenas será ligado um *LED* virtual na tela do aplicativo, esse *LED* servirá para informar a leitura do sensor de chuva, ou seja, *LED* virtual ligado sinal de chuva, desligado, não há chuva.

O próximo comando, “DHT” atribui o nome dht ao sensor de temperatura e umidade informando para a biblioteca o pino de leitura e o modelo do sensor.

O comando “BlynkTimer” atribui o nome timer para sua função, sendo assim, “timer” passa a exercer as instruções adicionadas a sua biblioteca. O “timer” funcionará como uma espécie de relógio do projeto, podendo ser utilizado para temporizar ações dentro do código.

Em seguida, estão o *token* do projeto que é uma chave de acesso único ao projeto junto ao *Blynk* e os comandos para conexão com a rede Wi-Fi, conforme a Figura 11.

Figura 11: Configurações de servidor e rede

```
// Configurações de servidor e rede
char auth[] = "1717f5115e2846da9b9384a7530dd7c3"; // Token do projeto Blynk
char ssid[] = "brisa-1121540"; // Rede Wi-Fi Local
char pass[] = "t5zpzodb"; // Senha da rede
```

Fonte: Elaborado pelo autor

O char “auth[]” recebe a chave de acesso *Blynk*, também chamada de *token* que é gerado no aplicativo ao iniciar um novo projeto. Já o “char ssid[]” recebe o nome da rede Wi-Fi ao qual o projeto estará conectado. O “char pass[]” recebe a senha da rede Wi-Fi. Esta é a etapa responsável por conectar o NodeMCU à internet, se a conexão for bem sucedida, o monitor serial apresentará uma mensagem de confirmação conforme exibido anteriormente na Figura 9.

Observação: É importante conferir se os caracteres estão escritos de forma correta para evitar erros de conexão.

Agora, iniciará a seção das funções desenvolvidas para a leitura dos sensores de chuva, temperatura e umidade e para o acionamento de um *LED*.

A primeira função será a void `Chuva()` exibida na Figura 12, dentro da função está inserida uma condição “if” para a verificação do sensor, se a leitura “digitalRead” do `pinSensorD` for um valor alto (positivo), isso é um sinal de que o sensor está enxuto, não há água, portanto não está chovendo. Quando isso acontece, o comando “`Led1.off()`” é ativado, esse comando desativa o *LED* virtual na tela do aplicativo *blynk*. Caso contrário, ou seja, se o valor lido for diferente de positivo, isso indica que o sensor foi ativado por água em sua superfície, sendo assim, é ativado o comando “`Led1.on()`”, esse comando aciona o *LED* virtual na tela do app, indicando que está chovendo.

Figura 12: Função void `Chuva()`

```
void Chuva(){ // Função para verificação do sensor de Chuva
  if (digitalRead(pinSensorD) == HIGH )
  {
    Led1.off(); // LED Virtual no Blynk OFF
  }
  else
  {
    Led1.on(); // LED Virtual no Blynk ON
  }
}
```

Fonte: Elaborado pelo autor

A próxima função apresentada será a void `Temp()`, conforme Figura 13. Nesta função serão realizadas as leituras do sensor DHT, ele retornará os valores da temperatura e umidade, para isso, deve-se começar definindo “h” como variável do tipo *float* para receber o valor de leitura da umidade, por meio do comando “`dht.readHumidity()`”, o valor retornado é atribuído a “h”, o mesmo se segue para “t” que receberá os valores referentes a temperatura, por meio do comando “`dht.readTemperature()`”. Em seguida, é adicionado a condição “if” para informar caso o sensor não esteja conseguindo realizar a leitura. O comando “`isnan()`” verifica se está chegando algum valor não numérico, caso seja verdadeiro, ele retorna *true*, do contrário, *false*, dessa forma, se retornar *true* será apresentada no monitor serial a seguinte mensagem “Falha ao ler o sensor DHT!”. Por fim, o comando “`Blynk.virtualWrite(V6, h)`” atribui o valor da umidade ao pino Virtual “V6” para que possa ser consultado na tela do aplicativo, o mesmo é

feito no comando “`Blynk.virtualWrite(V5, t)`”, onde o valor da temperatura é enviado para o pino virtual “V5” no aplicativo.

Figura 13: Função void Temp ( )

```
void Temp() // Função para monitoramento de Temperatura e Umidade
{
  float h = dht.readHumidity(); // Definindo h p/ receber valor de umidade
  float t = dht.readTemperature(); // Definindo t p/ receber valor de temperatura

  if (isnan(h) || isnan(t)) {
    Serial.println("Falha ao ler o sensor DHT!");
    return; }

  Blynk.virtualWrite(V6, h); // Enviando h p/ o pino virtual V6 do Blynk
  Blynk.virtualWrite(V5, t); // Enviando t p/ o pino virtual V5 do Blynk
}
```

Fonte: Elaborado pelo autor

A função a seguir, apresentada na Figura 14, será utilizada para acender e apagar o *LED* do projeto, o qual serve como exemplo para o acionamento de outras saídas do ESP8266, por meio do aplicativo ou pelo assistente pessoal.

Figura 14: Função BLYNK WRITE (V1)

```
BLYNK_WRITE(V1)
{
  int pinValue = param.asInt(); // Requisitando valor binário do APP Blynk
  digitalWrite(LED_Red, pinValue);
}
```

Fonte: Elaborado pelo autor

Essa função é própria da biblioteca *Blynk*, e será utilizada para se conectar ao pino virtual “V1” no app. Dentro dela, está definido “pinValue” como variável do tipo int (inteiro). O comando “`param.asInt( )`” é responsável por coletar as informações introduzidas ao V1 por meio do aplicativo, neste caso, o V1 foi configurado para enviar apenas 0 ou 1 por meio de um botão virtual, portanto, o comando “`digitalWrite( )`” irá ativar ou desativar a saída “LED\_Red” de acordo com o valor de “pinValue”, sendo 0 para apagar o *LED*, e 1 para acendê-lo.

Em seguida, está a função “`void setup( )`”, apresentada na Figura 15. Ela é própria do Arduino *IDE*, e serve para definir parâmetros em relação a entrada e saída de dados, e para executar as demais instruções de inicialização das bibliotecas e outros comandos inseridos no programa.

Figura 15: Função void setup ( )

```

void setup()
{
  Serial.begin(9600);           // iniciando comunicação serial
  pinMode(pinSensorD, INPUT);  // Entrada para sensor de Chuva
  pinMode(LED_Red, OUTPUT);    // Saída para ligação do Led
  digitalWrite(LED_Red, LOW);  // Mantendo Led inicialmente OFF
  Blynk.begin(auth, ssid, pass); // Iniciando funções do Blynk
  dht.begin();                 // Iniciando funções do sensor DHT

  timer.setInterval(1000L, Temp); // chamando a função Temp a cada segundo
  timer.setInterval(1000L, Chuva); // chamando a função Chuva a cada segundo
}

```

Fonte: Elaborado pelo autor

Dentro da função de *setup*, está o comando “`serial.begin(9600)`”, ele inicia as instruções de comunicação serial e define o valor de bits por segundo para tal. Nesta aplicação, se utiliza (9600) bits por segundo para a comunicação serial. Em seguida, utiliza-se o comando “`pinMode(pinSensorD, INPUT);`” para definir “`pinSensorD`” como entrada de sinal que será empregado para armazenar os dados do sensor de chuva. O mesmo comando é atribuído ao “`LED_Red`”, mas, para torná-lo uma saída, (*OUTPUT*), é por meio dele que vamos ligar o *LED* como demonstração de carga do projeto. O próximo comando, “`digitalWrite(LED_Red, LOW)`” está desligando a saída do *LED*, para que o mesmo inicie desativado. Em seguida, estão os comandos “`Blynk.begin( )`” e “`dht.begin( )`” que são utilizados para inicializar as instruções de suas respectivas funções para que seus comandos ao longo do código possam funcionar corretamente. Concluindo o *setup*, o comando “`timer.setInterval( )`”, é aplicado para chamar as funções “`void Temp`” e “`void Chuva`” a cada (1000L) ou mil milissegundos, para que os dados dos sensores possam ser enviados constantemente ao aplicativo.

A última seção do código é a que corresponde ao “`void loop( )`” conforme a Figura 16 a seguir.

Figura 16: Função void loop ( )

```

void loop()
{
  Blynk.run(); // Mantém o código ativo (rodando)
  timer.run(); // Ativa instruções de temporização
}

```

Fonte: Elaborado pelo autor



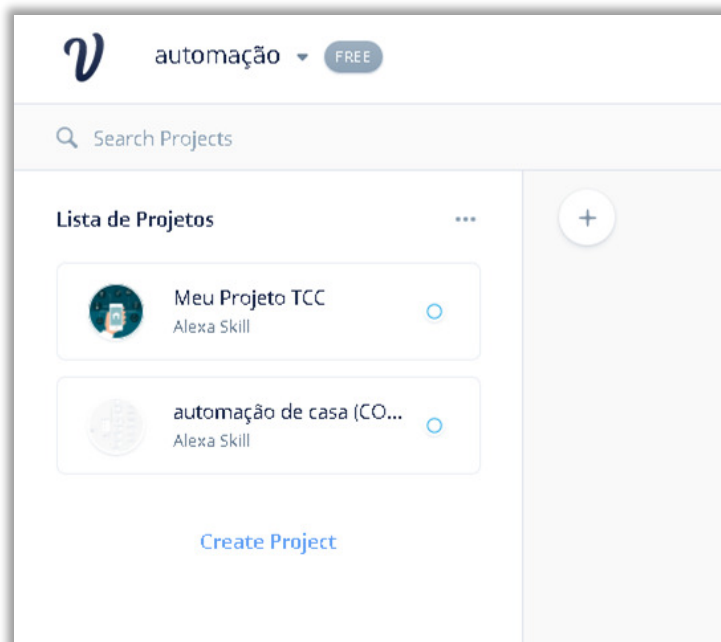
Dentro do *loop* existem dois comandos, o primeiro “Blynk.run( );” é responsável por executar as rotinas de instruções para manter o programa sempre conectado ao seu servidor, possibilitando o envio e recebimento de dados. O segundo comando, “timer.run( );” é quem executa as instruções referentes aos temporizadores definidos no código, assegurando que os intervalos de tempo sejam executados de maneira correta.

O código apresentado acima, está disponibilizado em APÊNDICE B – CÓDIGO FONTE DO PROJETO, ao final do trabalho.

Agora com a programação já concluída, pode-se avançar para a próxima seção de execução do projeto, que é realizar a integração do código com o dispositivo Alexa da *Amazon*. Para isso, será utilizada a plataforma de desenvolvimento *Voiceflow*.

Começar a criar projetos no *Voiceflow* é uma tarefa relativamente simples, mas, que requer dedicação e prática. Após realizar um cadastro utilizando o e-mail, o usuário é direcionado a página inicial do site.

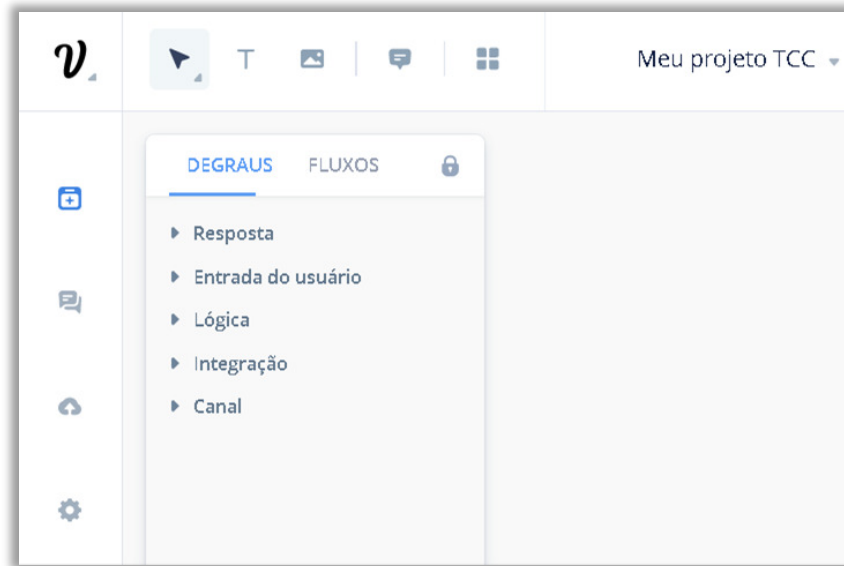
Figura 17: Tela inicial do *Voiceflow*



Fonte: Elaborado pelo autor

Conforme a Figura 17, é possível criar um projeto no *Voiceflow*, ao clicar em “Create Project”. Dentro do novo projeto, ao lado esquerdo da tela temos os *STEPS* (Degraus) que são os grupos onde estão os blocos de funções. Esses grupos são divididos nas seguintes categorias: resposta, entrada do usuário, lógica, integração e canal, conforme na Figura 18.

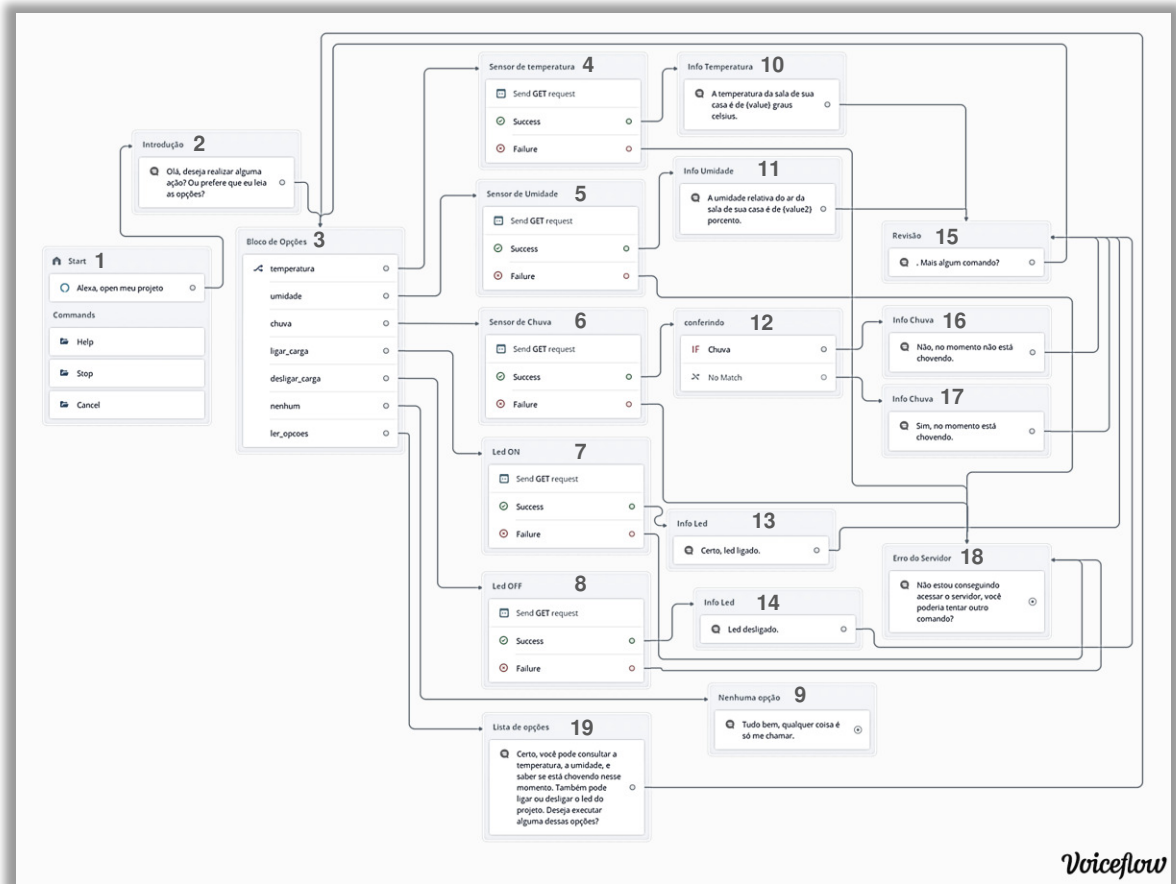


Figura 18: Grupos de blocos do *Voiceflow*

Fonte: Elaborado pelo autor

Cada grupo de blocos tem suas funções específicas. Algumas dessas funções serão utilizadas na execução do projeto, conforme será mostrado nos parágrafos a seguir.

Como já mencionado, o *Voiceflow* tem sua estrutura baseada em blocos de construção, em que cada um deles executa comandos distintos. Dentro de um novo projeto, sempre o primeiro bloco será o de definição do comando de acesso da *skill* (habilidade) específica, conforme o bloco 1 (*Start*), da Figura 19. Funciona da seguinte forma: Primeiro se define o nome da *skill* a ser criada, para este trabalho foi definido como, “meu projeto”. Escolhido o nome, e com a habilidade carregada nos servidores da *Amazon*, para acessar essa *skill*, é dito ao dispositivo de voz, “Alexa, inicie meu projeto”. Pronto, a partir disso, o *Echo* irá responder ou executar aquilo que está dentro da *skill* “meu projeto”.

Figura 19: Estrutura da *Skill* desenvolvida no *Voiceflow*

Fonte: Elaborado pelo autor

Após a etapa inicial, pode-se incluir os blocos subsequentes que irão compor a habilidade desenvolvida para o *Echo dot*.

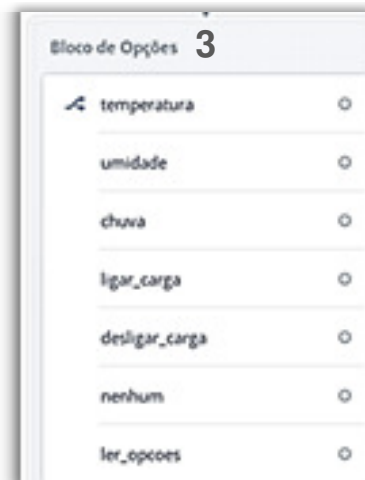
A ideia é fazer com que a assistente pessoal Alexa, possa apresentar os valores referentes aos sensores, e também que possa realizar o acionamento e o desligamento da saída do NodeMCU onde está o *LED*, isso sendo feito por meio de comandos de voz direcionados ao *Echo*.

Sendo assim, foi desenvolvida a sequência de blocos apresentada na Figura 19 para a estrutura de projeto, a fim de atender as demandas da proposta inicial.

Quando a *skill* “meu projeto” é chamada, a Alexa responde conforme o que está inserido no bloco 2 (Introdução), da Figura 19, (Olá, deseja realizar alguma ação? Ou prefere que eu leia as opções?). Esse é um bloco de fala, que faz parte do grupo de resposta, ele é um comando definido para servir de introdução, para que em seguida, o usuário possa informar qual a ação desejada.

O bloco 3 (Bloco de Opções), inserido logo após a introdução, é um bloco de escolha, que faz parte do grupo de entrada do usuário. É nesse bloco que estão incluídas as funções disponíveis para o usuário consultar / executar, que são: temperatura, umidade, chuva, ligar\_carga, desligar\_carga, nenhum e ler opções, conforme apresentado na Figura 20 a seguir.

Figura 20: Bloco de Opções (3)



Fonte: Elaborado pelo autor

Caso o usuário não conheça quais são as opções disponíveis no projeto, é possível solicitar à Alexa “leia as opções”, dessa forma, a assistente pessoal irá direcionar o fluxo para o bloco 19, e responderá ao usuário: “Certo, você pode consultar a temperatura, a umidade, e saber se está chovendo nesse momento. Também pode ligar ou desligar o led do projeto. Deseja executar alguma dessas opções?”.

Para que o usuário possa então escolher alguma das funções disponíveis, dentro delas estão listados os comandos para o acesso a cada uma das opções. Os comandos são definidos pelo usuário, conseqüentemente é possível personalizá-los de acordo com a aplicação desejada. Portanto, o usuário pode falar qualquer uma dessas frases estabelecidas em cada função do bloco de escolha para que a assistente pessoal acesse o servidor do *Blynk* e consulte / execute junto ao microcontrolador a ação solicitada pelo usuário.

Para acessar a função temperatura, estão definidos os seguintes comandos:

- 1. Qual a temperatura da sala?
- 2. Qual a temperatura da casa?
- 3. Temperatura!
- 4. Me informe a temperatura da sala!
- 5. Qual a temperatura?

No exemplo dos comandos acima, todas as opções de fala têm a palavra temperatura, sendo assim, é importante lembrar que o *Voiceflow* também pode identificar comandos que sejam semelhantes a alguns desses que envolvam a palavra temperatura, mesmo que não sejam idênticos aos exemplos definidos.

Seguindo adiante, caso o usuário fale para a Alexa algum desses comandos e consequentemente acione a função temperatura, o fluxo dos blocos irá seguir, direcionando para o bloco 4 (sensor de temperatura), conforme apresentado na Figura 19. Dentro desse bloco, estão definidas as instruções para que o *Voiceflow* consiga coletar os dados solicitados no servidor do *blynk*, e então armazene o valor em uma variável que será chamada de “value”, e assim possa retornar ao usuário a resposta por meio do bloco 10 (Info Temperatura), esse é um bloco de fala que será utilizado para apresentar o valor da temperatura por meio da seguinte resposta, “A temperatura da sala de sua casa é de {value} graus celsius” onde “value” é a variável do *Voiceflow* para o armazenamento do valor de temperatura coletado no servidor do *blynk*. Dessa maneira, sempre que a temperatura for solicitada à Alexa, ela irá informar o valor lido pelo sensor em tempo real.

Posteriormente, o fluxo será direcionado para o bloco 15 (Revisão), ele também é um bloco de fala, e será utilizado para perguntar ao usuário se deseja realizar mais alguma ação, por meio da frase, “Mais algum comando?”. Em seguida, o fluxo é direcionado novamente para o bloco 3 (Bloco de Opções), e a resposta do usuário então recebida é comparada aos comandos das funções do bloco, para que seja realizada uma nova ação, conforme desejado.

Sobre o bloco 4 (Sensor de temperatura), esse é um bloco de API (*Application Programming Interfaces*) ou Interfaces de programação de aplicativos, do grupo de integração. Os blocos de API são capazes de acessar outros sites, serviços ou servidores para executar as ações planejadas no projeto, nesse caso, a API será utilizada para acessar o servidor do *Blynk* e consultar o valor lido pelo sensor de temperatura, que está vinculado ao pino virtual (V5) definido na programação do ESP8266.

Conforme pode ser observado na Figura 21, para que o bloco de API tenha acesso ao servidor do *Blynk*, na opção *Request URL (Uniform Resource Locator)*, faz-se necessário a utilização do *link* de acesso formatado com a seguinte estrutura: **http://Blynk\_IP\_Address/auth\_token/get/pin\_number**.

O **/Blynk\_IP\_Address/** é o endereço IP (*Internet Protocol*) do servidor da região, que pode variar de acordo com a sua localização global. Para saber qual o IP de sua região, basta inserir o comando: “ping blynk-cloud.com” no prompt de comando.

O **/auth\_token/** é o código de acesso do seu projeto *Blynk*. Ele é enviado ao *email* do usuário sempre no início de um novo projeto *Blynk*, geralmente formado por uma combinação aleatória de letras e números.

O comando **/get/** para essa aplicação de consulta de valores, é fixo, portanto, para esse bloco não é preciso alterá-lo.

E por fim, **/pin\_number/** é o pino virtual do *Blynk*, no qual é realizada a leitura do sensor e enviada ao aplicativo, que neste caso, corresponde ao pino (V5).

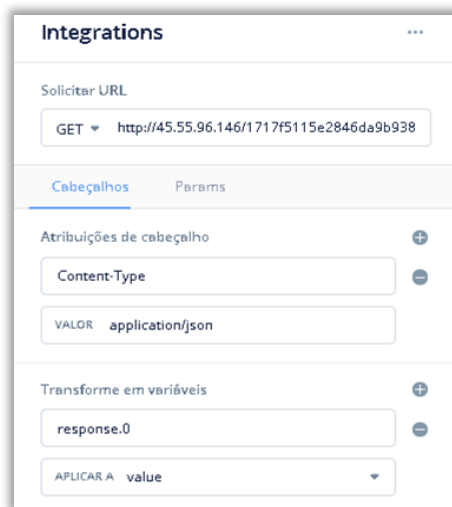
Com essa estrutura definida, para realizar a consulta de outros sensores, é necessário apenas alterar os pinos virtuais de acordo com a configuração de cada sensor no *Blynk*.

Observação: Caso os blocos de API não consigam se conectar ao servidor por meio da URL de requisição para realizar a troca de informações, eles retornarão a função “failure” (falha). As funções de falha do projeto estão direcionadas ao bloco 18 (Erro do Servidor) que apresenta a seguinte mensagem: “Não estou conseguindo acessar o servidor, você poderia tentar outro comando?”, para que o usuário seja informado do erro, e assim, possa reiniciar o processo de solicitação.

Em seguida, dentro da API é definida uma variável para receber os valores coletados dos sensores, também é definido o tipo de variável que será utilizada, neste caso corresponde ao tipo “Content-Type”. O formato do valor de requisição empregado em aplicações desse tipo é “application/json”.

Concluindo a estrutura da API, atribui-se os comandos “response.0” para a obtenção do valor bruto da leitura de acordo com o formato definido anteriormente “json”. Por último, é

Figura 21: Configurações do bloco de API



Fonte: Elaborado pelo autor

definido o nome da variável “value”, que irá armazenar os valores recebidos do servidor *Blynk*. Conforme demonstrado na Figura 21.

Feito isto, o *Voiceflow* será capaz de acessar o servidor *Blynk*, solicitar os dados do sensor de temperatura, armazenar o valor na variável estabelecida e informar ao usuário, por meio do bloco de fala inserido na sequência do bloco de API.

Semelhante à temperatura, para consultar a umidade relativa do ar, estão configurados os comandos:

- 1. Me informe a umidade da sala.
- 2. Me informe a umidade relativa do ar da sala.
- 3. Umidade!
- 4. Qual a umidade relativa do ar da sala?
- 5. Qual a umidade da sala?

Assim como para a requisição da temperatura, o processo se repete na API do bloco 5 (Sensor de umidade). Para o sensor de umidade, altera-se o pino virtual (pin\_number) na URL de requisição para o pino correspondente à umidade, que é o pino virtual (V6), e cria-se uma nova variável de armazenamento para os dados do referido sensor, “value2”. Desse modo, é possível questionar ao *Echo* qual a umidade relativa do ar de onde o sensor estiver instalado.

Portanto, quando solicitada a informação respectiva à umidade, a resposta apresentada pela assistente pessoal será a seguinte: “A umidade relativa do ar da sala de sua casa é de {value2} por cento”, em que “value2” corresponde ao valor numérico obtido no servidor do *Blynk*, no momento da requisição.

Logo em seguida, assim como a seção anterior da temperatura, o fluxo segue para o bloco 15 (Revisão), para questionar ao usuário se há mais alguma instrução a ser realizada. A resposta do usuário é então direcionada novamente ao bloco 3 (Bloco de Opções), e comparada aos comandos existentes.

Para o sensor de chuva, foram utilizados os comandos:

- 1. Está chovendo?
- 2. Está chovendo lá fora?
- 3. Me informe se está chovendo.

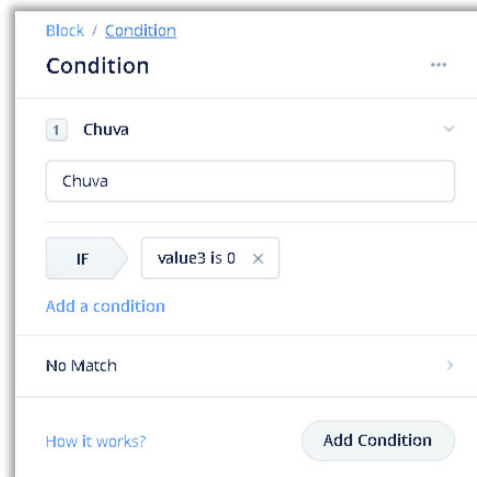
Sobre a chuva, por se tratar de um sensor digital, a Alexa irá informar se está chovendo ou não, de acordo com a solicitação feita por meio dos padrões definidos acima.

O processo de requisição da API do bloco 6 (Sensor de chuva) em relação aos dados do sensor é o mesmo que foi empregado para temperatura e umidade. Mais uma vez, as alterações

são: a mudança do pino virtual (pin\_number) na URL de requisição, que para o sensor de chuva foi utilizado (V8), e a variável de armazenamento alterada para “value3”.

Em sequência está o bloco 12 (conferindo), um bloco de condição do grupo de lógica. Esse bloco será responsável por verificar se o valor recebido e armazenado em “value3” é igual a 0, conforme apresentado na Figura 22 a seguir.

Figura 22: Condição If (sensor de chuva)



Fonte: Elaborado pelo autor

Caso a condição seja verdadeira, e “value3” seja igual a 0, é sinal que o sensor não está acionado, portanto, não há chuva no momento, essa resposta será apresentada pelo bloco 16 (Info Chuva), da seguinte maneira: “Não, no momento não está chovendo”. Caso a condição seja falsa, ou seja, “value3” seja diferente de 0, isso indica que o sensor está ativado e que, portanto, está chovendo, nesse caso, a resposta apresentada será: “Sim, no momento está chovendo”, conforme o bloco 17 (Info Chuva). Posteriormente, o fluxo é então direcionado para o bloco 15 (Revisão), e em seguida para o bloco 3 (Bloco de Opções), conforme já abordado.

Para a função `ligar_carga`, foram estabelecidas as seguintes opções de comandos de voz:

- 1. Acenda o *LED*.
- 2. Ligue o *LED*.
- 3. Ligue o *LED* vermelho.

Desse modo, quando deparada com alguma dessas solicitações, a Alexa irá acender o *LED* que se encontra conectado à saída do NodeMCU.

Para executar essa ação e ligar o *LED*, se utiliza a API do bloco 7 (Led ON). Na sua estrutura foram aplicados os comandos “Content-Type” e “application/json” novamente.

Entretanto, a URL de requisição possui formato diferente do utilizado antes: **http://Blynk\_IP\_Address/auth\_token/update/pin\_number?/value=1**

Diferente da URL utilizada anteriormente, nessa, há a alteração do comando */get/* por */update/*, e também se encontra a adição de um ponto de interrogação (?), após */pin\_number/*. Por fim, é introduzido o comando */value=1*, que realiza o envio do valor numérico “1” para “pinValue” que é a variável de entrada na programação do NodeMCU, para o acionamento do *LED*. O pino virtual utilizado para o *LED* foi o pino (V1).

Dessa maneira, o servidor atualizará a variável “pinValue”, ao receber o valor 1 enviado pelo *Voiceflow*, a porta de saída D2 irá para o estado “HIGH” (alto) e o *LED* será acesso.

Em seguida, o fluxo de blocos segue para o bloco 13 (Info Led), que informará ao usuário, “Certo, led ligado”. Após isso, o fluxo segue para os blocos 15 (Revisão) e 3 (Bloco de Opções), respectivamente, atuando conforme descrito anteriormente.

Para realizar o desligamento do *LED*, dentro da função *desligar\_carga* foram definidos os comandos a seguir:

- 1. Apague o *LED*.
- 2. Desligue o *LED*.
- 3. Desligue o *LED* vermelho.

Com isso, a assistente poderá desligar o *LED*, quando solicitado pelo usuário. Para executar essa ação, usa-se a mesma URL utilizada no acionamento, apenas alterando o valor de envio para 0, “value=0”, com isso, o valor de “pinValue” será modificado para “LOW” (baixo), o *LED* será desligado e será apresentada ao usuário a seguinte mensagem: “Led desligado”, conforme o bloco 14 (Info Led). Após isso, o fluxo segue para os blocos 15 (Revisão) e 3 (Bloco de Opções), para verificar se há algum novo comando a ser introduzido.

Por fim, a função *nenhum*, foi definida para funcionar como válvula de escape para quando o usuário desistir de realizar alguma ação ou quando não quiser mais executar um novo comando em sequência. Nessa função estão configurados os comandos de voz:

- 1. Nada.
- 2. Nenhum.
- 3. Não.
- 4. No momento não.

Portanto, ao se deparar com as falas da assistente pessoal, “Olá, o que você deseja realizar?” ou “Mais algum comando?”, o usuário poderá desistir de efetuar algum comando ou questionamento à Alexa, sendo necessário apenas pronunciar alguma dessas opções



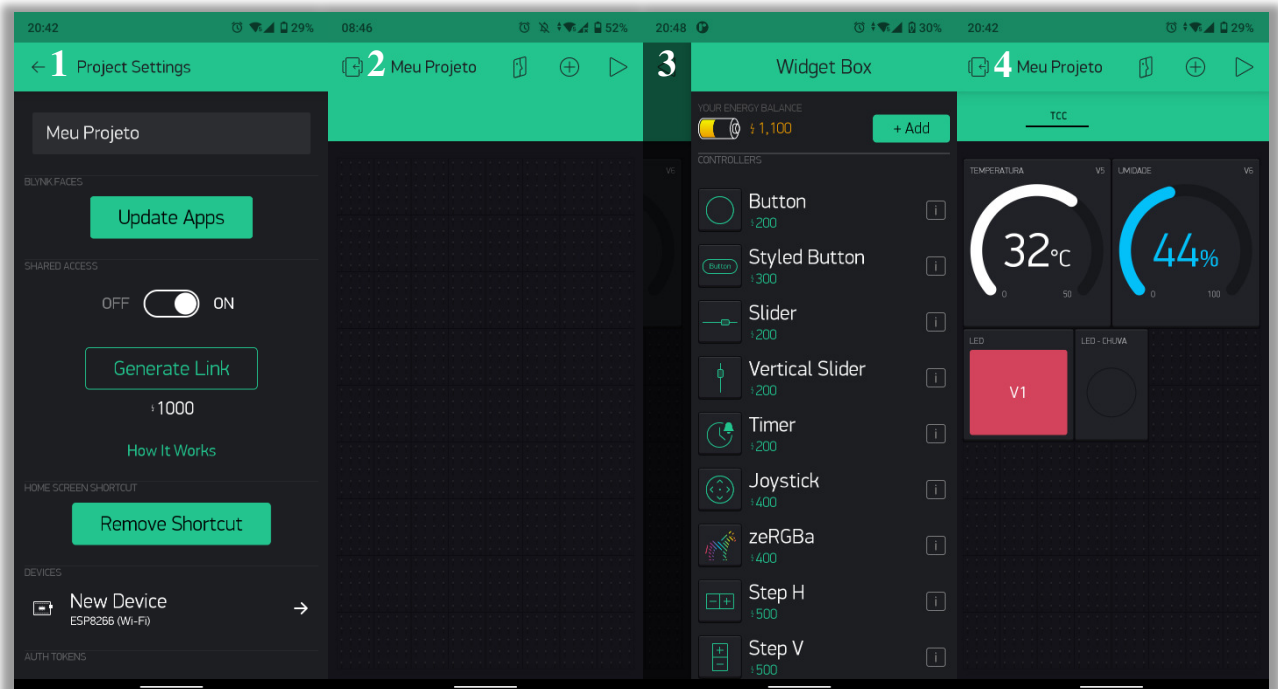
demonstradas, que a Alexa responderá conforme o bloco 9 (Nenhuma opção), “Tudo bem, qualquer coisa é só me chamar”.

Com a estrutura do *Voiceflow* concluída, o upload do projeto para a Alexa pode ser executado, após o carregamento o projeto estará disponível para uso.

Desenvolvido o código do NodeMCU, as funções do *Voiceflow* criadas e suas variáveis já definidas, agora é possível configurar o aplicativo para a interação com o projeto.

A Figura 23 a seguir, apresenta em sequência as telas (itens 1, 2, 3 e 4) para o desenvolvimento do aplicativo que será utilizado no projeto. No item 1, está exibida a tela de definição das configurações iniciais, nome do projeto e escolha do dispositivo ESP8266 (Wi-Fi). O item 2 apresenta a tela do projeto, esse é o campo aberto onde podem ser inseridos os elementos que serão adicionados por meio do botão (+) no canto superior direito da tela. O item 3 apresenta alguns dos elementos que podem ser adicionados ao projeto, entre eles estão botões, controles deslizantes, temporizadores, medidores, dentre outros. Por fim, o item 4 apresenta o projeto com os elementos já adicionados, um medidor para temperatura, um medidor para umidade, um botão para o acionamento do *LED* e um *LED* virtual para indicação de chuva.

Figura 23: Configurando o aplicativo *Blynk*

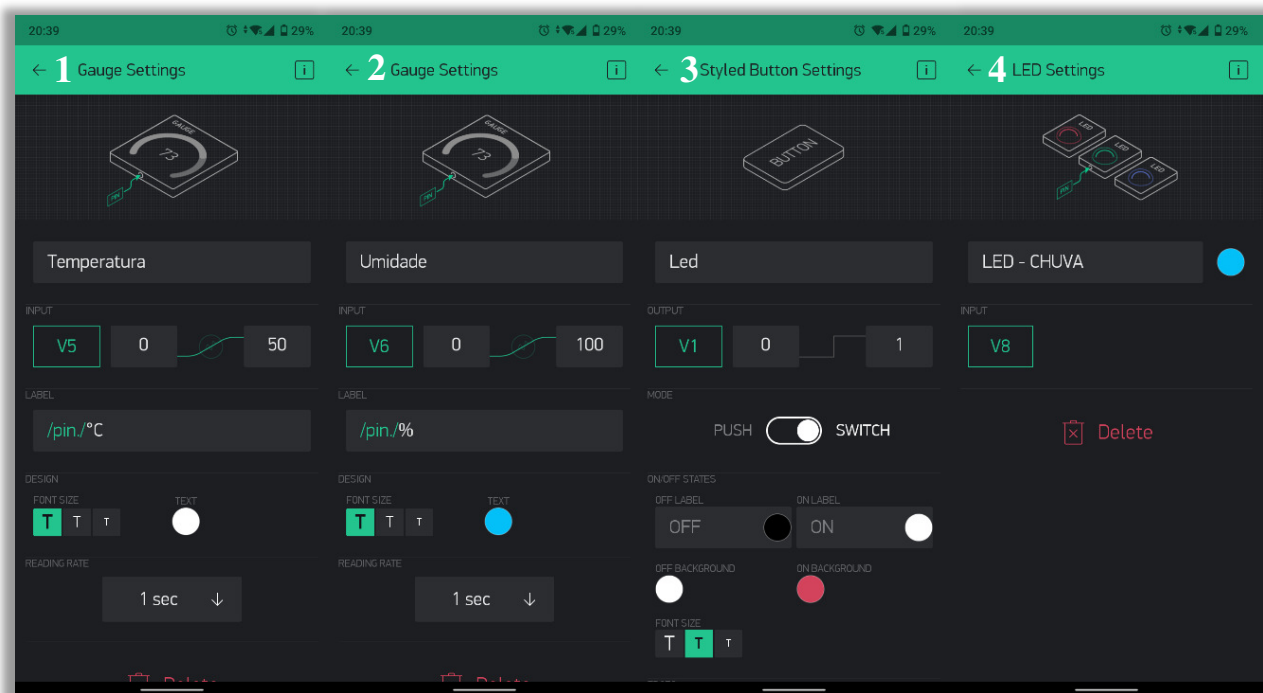


Fonte: Elaborado pelo autor

Em seguida, é necessário configurar cada elemento adicionado, atribuindo as variáveis correspondentes aos sensores e ao botão.

A Figura 24, apresenta as telas de configuração para os seguintes elementos: medidor de temperatura (item 1), medidor de umidade (item 2), botão de acionamento do *LED* (item 3), e *LED* virtual indicador de chuva (item 4).

Figura 24: Configurando as ferramentas do projeto *Blynk*



Fonte: Elaborado pelo autor

No item 1, é atribuído o nome temperatura para o medidor, em seguida se adiciona o pino virtual correspondente à temperatura (V5) e é definido o *range* de atuação do medidor. Neste caso, foi estabelecida a variação de 0 à 50 graus celsius e adicionado o símbolo indicador de temperatura nessa escala (°C). Por último, ao escolher a cor de apresentação do medidor como branco e o tempo de envio de dados para 1 sec (1 segundo), ou seja, o valor da temperatura será atualizado na tela do aplicativo a cada segundo.

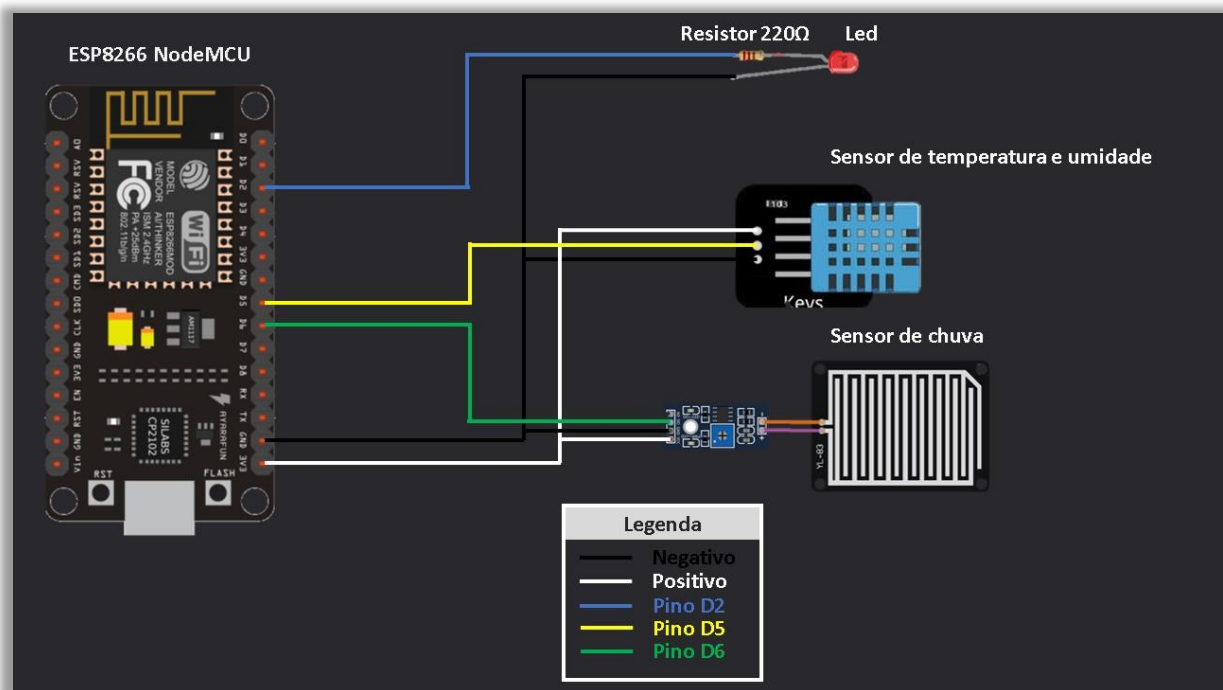
Semelhante ao item 1, no item 2 foi adicionado o nome umidade para o medidor. O pino virtual correspondente é o pino virtual (V6), seu *range* de atuação varia de 0 à 100 por cento, que é o nível de umidade relativa do ar, adiciona-se o símbolo (%) para indicação dos valores. Em seguida é escolhida a cor azul para o medidor, e também definido o tempo de atualização para um segundo.

No item 3, é estabelecido o nome *LED* para o botão, e adicionado o pino virtual correspondente ao *LED* do projeto (V1), em seguida, estão definidos os valores que serão enviados para o desligamento ou acionamento do led, que são, 0 e 1 respectivamente. Também é possível optar pela função “*switch*” do botão, que serve para manter o botão retido naquele estado até o próximo clique. Por fim, atribuem-se as cores para cada estado do botão, o botão irá permanecer branco e com indicação *OFF* quando o *LED* estiver desligado e vermelho com indicação *ON* quando o *LED* estiver ligado.

No item 4, a configuração para o *LED* virtual indicador de chuva é realizada definindo-se o pino virtual que corresponde ao sensor de chuva (V8) e determinando a cor do *LED* como azul, para ser apresentado na tela do aplicativo.

Com o aplicativo configurado, a etapa seguinte consiste na montagem e ligação elétrica dos componentes, os mesmos foram conectados conforme apresentado na Figura 25.

Figura 25: Esquema de ligação elétrica do projeto



Fonte: Elaborado pelo autor

Os componentes utilizados no projeto foram:

- ESP8266 NodeMCU;
- Módulo sensor de temperatura e umidade (DHT11);
- Módulo sensor de chuva;
- *LED* vermelho;

- Resistor ( $220\Omega$ );
- Jumpers.

Em síntese, ao realizar a montagem e carregar o código com a programação para o ESP8266 NodeMCU, ele já estará disponível para enviar os dados do projeto para o servidor do *Blynk*, de modo que possam ser acessados pelo aplicativo ou pela assistente pessoal.

## 4 ANÁLISE DOS RESULTADOS

Os resultados obtidos foram satisfatórios, com o projeto correspondendo bem as expectativas. A seguir, na Figura 26 é possível observar os dispositivos integrados utilizados neste trabalho em operação. A assistente pessoal está aguardando um comando, o *LED* vermelho está acionado e os valores de temperatura e umidade são apresentados na tela do smartphone.

Figura 26: Projeto integrado em funcionamento



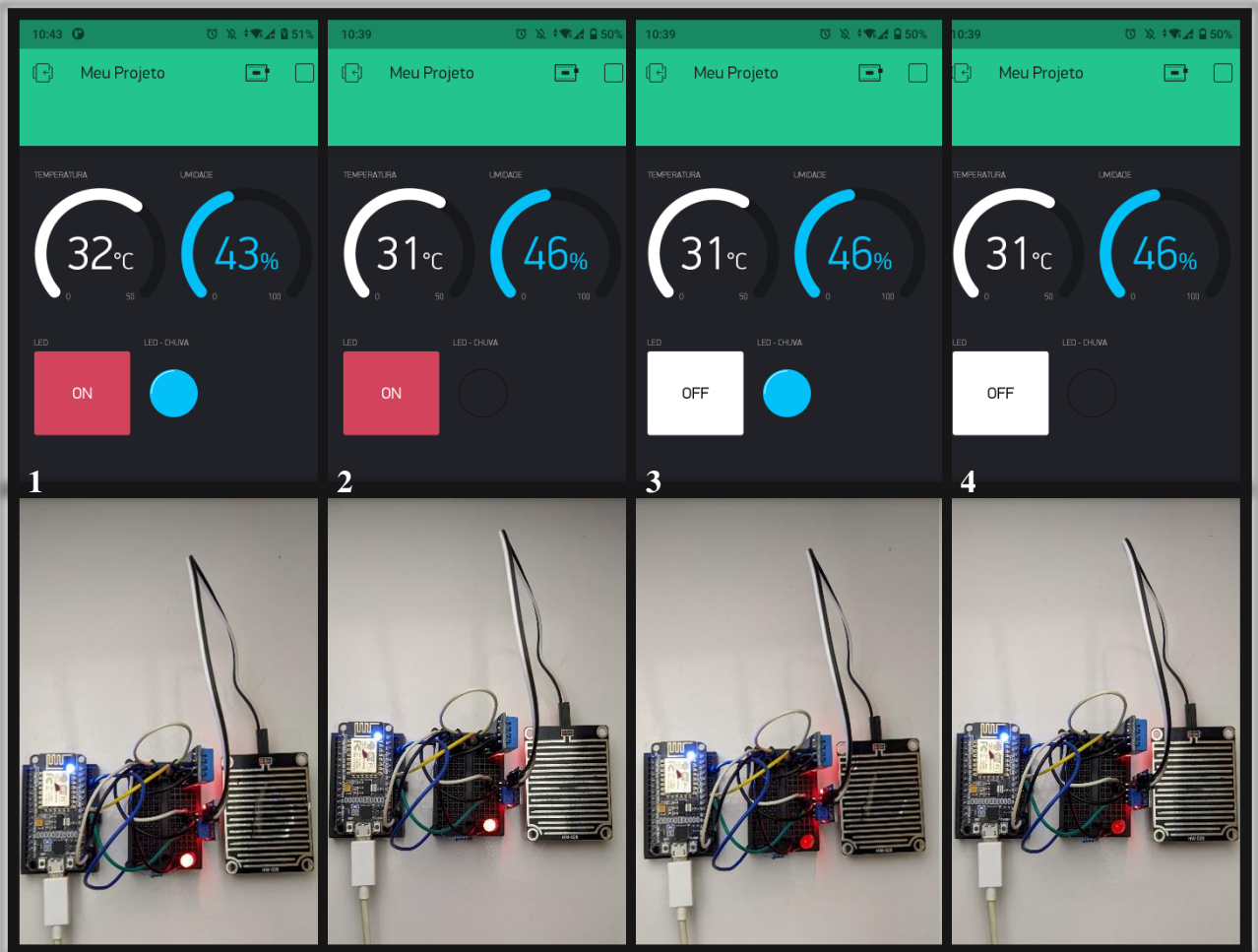
Fonte: Elaborado pelo autor

O aplicativo *Blynk* mostrou-se uma ótima opção para aplicações que envolvam o NodeMCU em projetos de automação, pois, tem um bom tempo de ação e apresenta confiabilidade nos valores coletados dos sensores, tendo em vista que não apresentou distorções ou erros de leitura, além de ser relativamente simples de implementar, configurar, desenvolver e operar aplicações por meio de sua plataforma.

Entretanto, apesar de ser uma ferramenta gratuita, é importante ressaltar que para projetos mais amplos envolvendo o aplicativo *Blynk*, faz-se necessário a aquisição de “energia” adicional, isso porque o *Blynk* fornece uma quantidade limitada desse recurso que é utilizado no aplicativo para a aquisição dos *widgets* (ferramentas), que são os botões, os indicadores e semelhantes utilizados para a criação do projeto *Blynk*.

Na Figura 27 é possível observar a interface da aplicação desenvolvida. Estão apresentados os resultados dos testes envolvendo o aplicativo e o protótipo, com os comandos sendo realizados por meio do aplicativo e seus resultados podendo ser consultados nas imagens (parte inferior de cada item). Também estão exibidos os valores de cada sensor do na tela do *smartphone*.

Figura 27: Teste prático do projeto em funcionamento



Fonte: Elaborado pelo autor

Conforme a Figura 27, também é possível observar no item 1, que o aplicativo está marcando 32°C de temperatura, 43% de umidade relativa do ar, o botão do *LED* está acionado (vermelho), e o *LED* virtual azul está ligado, indicando que o sensor de chuva está molhado. Isso pode ser confirmado com a imagem inferior do item 1, onde mostra o *LED* vermelho ligado e o sensor de chuva que está com uma porção de água em sua superfície.

No item 2, o cenário é parecido, mas agora a temperatura está em 31°C e a umidade em 46%. A porção de água foi retirada, conforme a imagem inferior, também apresentado na tela do aplicativo que está com o *LED* virtual azul desligado e o *LED* vermelho permanece ligado.

No item 3, a temperatura e umidade se repetem, o botão do *LED* vermelho foi desligado passando para a cor branca, e o sensor de chuva foi molhado novamente, conforme pode ser confirmado na tela do aplicativo, com o *LED* virtual azul ligado e na imagem inferior, com o sensor molhado.

No item 4, a temperatura e umidade se mantêm sem alteração, o *LED* vermelho permanece desligado, e o sensor de chuva desativado com sua superfície sem a presença de água, também apresentado na tela do aplicativo com o *LED* virtual azul desligado.

Esses testes foram realizados cinco vezes, e os resultados obtidos foram semelhantes aos apresentados, não foram notadas distorções ou erros que chamassem a atenção ou que comprometessem a estrutura do projeto.

Sobre a plataforma *Voiceflow* foi uma ferramenta muito útil para o desenvolvimento do trabalho, considerando se tratar de um site com um plano gratuito bem amplo que permite a utilização de muitos de seus recursos sem limitações, além de ter uma estrutura de blocos didática que facilita ao usuário a criação de projetos. Entretanto, é importante mencionar uma ressalva que foi percebida durante os testes do projeto: Para que o *Voiceflow* consiga acessar os servidores do *Blynk* e realizar os comandos de maneira correta, a conta do usuário no site da *Amazon* precisa estar com um endereço válido dos Estados Unidos da América, isso porque alguns recursos da Alexa só estão disponíveis nos servidores locais desse país.

A assistente pessoal também operou de maneira adequada, entendendo devidamente os comandos a ela direcionados e respondendo corretamente, apontando os valores dos sensores conforme apresentados pelo aplicativo. No geral, a Alexa atuou bem, com bom tempo de interação e resposta e com uma distância de interação considerável, apenas apresentando poucas situações de desvio de função, devido a comandos mal pronunciados, conseqüentemente mal compreendidos por ela.

A seguir, será descrita uma interação realizada com a assistente pessoal Alexa apresentado os resultados obtidos do diálogo. Observação: A interação é realizada simulando um ambiente residencial, por isso, alguns comandos ou respostas são apresentadas tendo em vista este ambiente.

**(usuário)** - Alexa, inicie meu projeto!

**(Alexa)** Olá, o que você deseja realizar?

**(usuário)** - Me informe a temperatura da sala.

**(Alexa)** A temperatura da sala de sua casa é de 32 graus celsius. Mais algum comando?

**(usuário)** - Me informe a umidade relativa do ar.

**(Alexa)** A umidade relativa do ar da sala de sua casa é de 38 por cento. Mais algum comando?

**(usuário)** - Ligue o *LED* vermelho.

**(Alexa)** Certo, *LED* ligado! Mais algum comando?

**(usuário)** - Está chovendo lá fora?

**(Alexa)** Não, no momento não está chovendo. Mais algum comando?

**(usuário)** - Desligue o *LED*.

**(Alexa)** *LED* desligado. Mais algum comando?

**(usuário)** - No momento não.

**(Alexa)** Tudo bem, qualquer coisa é só me chamar.

A aplicação prática pode ser conferida em vídeo por meio do *link* disponível no APÊNDICE A – ATUAÇÃO DA ALEXA (VÍDEO), onde é apresentado o funcionamento do projeto com a interação do usuário e a assistente Alexa.

Se comparado aos trabalhos correlatos, o atual projeto se destaca pelo fato de unir em sua aplicação tanto o aplicativo de celular, quanto a assistente pessoal para executar ações semelhantes de automação, além de se utilizar da ferramenta *Voiceflow* como um diferencial para realizar essa interação.



## CONCLUSÃO

Tendo em vista a proposta primária do projeto, realizar a integração dos dispositivos *Blynk* e *Echo* de modo que possibilitasse o controle de um microcontrolador ESP8266 NodeMCU por meio de um aplicativo e por comandos de voz utilizando a assistente pessoal Alexa, pode se concluir que para fins práticos o trabalho realizado constitui-se em um caminho inicial para direcionar novos projetos na área de automação envolvendo comandos de voz e aplicativo, isso porque, este projeto atual pode ser expandido ou modificado para atender a diversas aplicações, seja na automação residencial, em projetos de monitoramento remoto que envolvam agricultura, projetos escolares e afins.

O trabalho também demonstrou que é possível criar projetos com um bom grau de personalização, de modo a satisfazer as necessidades do usuário, pois, permite que tanto o aplicativo quanto a assistente pessoal e o ESP8266 possam ser programados e configurados de maneira diversificada. Por meio do *Voiceflow* é possível desenvolver uma vasta possibilidade de funções para interação com o usuário por meio dos comandos de voz, assim como o *Blynk* também possibilita a utilização de muitos recursos e funções interessantes na área de automação.

A partir dessa ideia é possível implementar novos sensores e realizar o acionamento de outras cargas como lâmpadas, televisores, ventiladores, portões eletrônicos, bombas de água, motores, enfim, o projeto possibilita uma infinidade de opções, tudo podendo ser feito de maneira independente.

Sendo assim, para projetos futuros, ficam opções para a exploração das demais funções do *Voiceflow* que não foram utilizadas neste projeto, de modo que proporcione um diálogo mais fluido e natural com a assistente Alexa, atribuindo novos recursos e mais opções de resposta por parte dela. Também a introdução de outros periféricos, como sensores, para aplicações de automação residencial, sensores de luminosidade para possibilitar ao usuário um controle do nível de iluminação do ambiente, por exemplo. Assim como utilizar o sensor de chuva para que ao ser acionado a Alexa possa informar que começou a chover, sem a necessidade de o usuário questionar sobre isso, também aproveitando para acionar automaticamente um toldo para proteger áreas como varandas, garagens ou até mesmo fechar janelas para proteger o ambiente contra a chuva, enfim, as possibilidades são muitas, e podem variar de acordo com os objetivos de cada projeto.

## REFERÊNCIAS

CAMIOTTO, Giovanna. Após polêmica, Amazon cria ‘novo nome’ e voz masculina para Alexa. **Ei nerd**, 2021. Disponível em: < <https://www.einerd.com.br/amazon-alexa-criancas-bullying/>>. Acesso em: 24 de agosto de 2021.

CARDOSO, Pedro. Tudo sobre Amazon Echo Dot: veja se funciona no Brasil e ficha técnica. **Techtudo**, 2018. Disponível em: <<https://www.techtudo.com.br/noticias/2018/05/tudo-sobre-amazon-echo-dot-veja-se-funciona-no-brasil-e-ficha-tecnica.ghhtml>> Acesso em: 26 de agosto de 2021.

HAACK, William et al. Security Analysis of the Amazon Echo, 2017. Disponível em: <<https://courses.csail.mit.edu/6.857/2017/project/8.pdf>>. Acesso em: 17 agosto de 2021.

HOME AUTOMATION AND VIRTUAL ASSISTANTS – WHAT ARE PEOPLE USING NOW? **Homesales**, 2017. Disponível em: <<https://homesales.com.au/news/latest/whats-new-home-automation-virtual-assistance/>>. Acesso em: 22 de junho de 2021.

KOLBAN, Neil. Kolban’s Book on the ESP8266. Texas, 2015. Disponível em: <<http://neilkolban.com/tech/esp8266/>>. Acesso em: 30 de agosto de 2021.

MOURA JÚNIOR, Anderson David de. **Automação residencial de baixo custo**. 2020. Monografia (Graduação em Engenharia da Computação) - Faculdade de Tecnologia e Ciências Sociais Aplicadas, Centro Universitário de Brasília, Brasília, 2020. Disponível em: <<https://repositorio.uniceub.br/jspui/handle/prefix/15113>>. Acesso em: 19 de outubro de 2021.

NETO, Leandro Dallarosa. **Protótipo de automação residencial utilizando uma assistente de voz**. 2018. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Universidade Regional de Blumenau, centro de ciências exatas e naturais. Blumenau-SC. 2018. Disponível em: [http://dsc.inf.furb.br/arquivos/tccs/monografias/2018\\_1\\_leandro-dallarosa-neto\\_monografia.pdf](http://dsc.inf.furb.br/arquivos/tccs/monografias/2018_1_leandro-dallarosa-neto_monografia.pdf). Acesso em: 22 de junho de 2021.

OLIVEIRA, Euler. Conhecendo o Blynk. **Blogmasterwalkershop**, 2017. Disponível em: <<https://blogmasterwalkershop.com.br/blynk/conhecendo-o-blynk>>. Acesso em: 26 de agosto de 2021.

OLIVEIRA, Greici. NodeMCU – Uma plataforma com características singulares para o seu projeto IoT. **Blogmasterwalkershop**, 2016. Disponível em: <<https://blogmasterwalkershop.com.br/embarcados/nodemcu/nodemcu-uma-plataforma-com-caracteristicas-singulares-para-o-seu-projeto-iot>>. Acesso em: 23 de julho de 2021.

PAIXÃO, Priscila Reis Soares et al. Development of an automation system, integrating a virtual assistant and iot devices. **ITEGAM-JETIA**, 6 de setembro de 2019.

SANTOS, Alberto Marianno. Dicas do Voice Flow — Bloco Avançado. **Medium**, 2020. Disponível em: <<https://medium.com/@albertomarianno/dicas-do-voice-flow-bloco-avan%C3%A7ado-9c30d7c0a0b4>>. Acesso em: 29 de agosto de 2021.

SCHWARTZ, ERIC Hal. Voiceflow fecha rodada de financiamento de \$ 20 milhões para estender a plataforma de AI de conversação. Voicebot.ai, 2021. Disponível em: <<https://voicebot.ai/2021/07/29/voiceflow-closes-20m-funding-round-to-extend-conversational-ai-platform/>>. Acesso em: 29 de agosto de 2021.

SERRANO, Tiago Medicci. Introdução ao Blynk App. Embarcados, 2018. Disponível em: <<https://www.embarcados.com.br/introducao-ao-blynk-app>>. Acesso em: 29 julho de 2021.

SGARBI, Julio A., TONIDANDEL, Flavio. Domótica Inteligente: Automação Residencial baseada em Comportamento. In: Workshop de Teses e Dissertações em Inteligência Artificial, Ribeirão Preto, São Paulo, 2006.

VOICEFLOW AGORA PERMITE CRIAR APPS DE VOZ EM UM SÓ LUGAR. **Newvoice**, 2021. Disponível em: <<https://newvoice.ai/2021/02/23/voiceflow-agora-permite-criar-apps-de-voz-em-um-so-lugar/>>. Acesso em: 29 de agosto de 2021.

## APÊNDICE A – ATUAÇÃO DA ALEXA (VÍDEO)

A seguir, está o *link* do vídeo no YouTube na modalidade não listado, onde é possível acompanhar a demonstração prática do projeto em funcionamento.

Link: <https://youtu.be/VQT5sPOKxoY>

Figura 28: Vídeo apresentando o projeto em funcionamento



Fonte: Elaborado pelo autor

## APÊNDICE B – CÓDIGO FONTE DO PROJETO

A seguir está o código com toda a programação do projeto, com comandos da *IDE* Arduino desenvolvido neste trabalho, para auxiliar em trabalhos e projetos futuros que sigam no caminho da automação com aplicativos de celular, assistentes pessoais e semelhantes.

O código também foi publicado no Github, uma plataforma de desenvolvimento que hospeda códigos de projetos de programadores de todo o mundo. Nele, criadores podem interagir e trocar experiências e dicas para auxiliar no desenvolvimento de seus trabalhos.

Link do código no Github: <https://github.com/Rodrigo-Tomaz/TCC-Blynk-e-Alexa/commit/bcf4070e6fe73a38814907e601ebabab5e68823c>

```
// Bibliotecas do Projeto
#define BLYNK_PRINT Serial // Função p/ exibição de status do projeto
#include <ESP8266WiFi.h> // Configurações de conexão com a rede Wi-Fi
#include <BlynkSimpleEsp8266.h> // Configurações de conexão com o servidor Blynk
#include <SimpleTimer.h> // Funções de tempo e milésimos de segundo
#include <DHT.h> // Configurações dos sensores de temperatura e umidade

#define LED_Red 4 // Definindo Pino(D2) como saída p/ Led Vermelho
#define pinSensorD 12 // Definindo Pino(D6) p/ sensor Digital de chuva
#define DHTPIN 14 // Definindo Pino(D5) do ESP8266 p/ o sensor DHT
#define DHTTYPE DHT11 // Definindo tipo de sensor: DHT 11

WidgetLED Led1(V8); // Led Virtual na tela do Blynk (indicador de Chuva)
DHT dht(DHTPIN, DHTTYPE); // Setando configurações junto a biblioteca
BlynkTimer timer; // Atribuindo o tempo ao 'timer'

// Configurações de servidor e rede
char auth[] = "Auth Token do projeto"; // Token do projeto Blynk
char ssid[] = "Rede-Wi-Fi"; // Rede Wi-Fi Local
char pass[] = "Senha"; // Senha da rede

void Chuva(){ // Função para verificação do sensor de Chuva
```

```

if (digitalRead(pinSensorD) == HIGH )
{
  Led1.off(); // LED Virtual no Blynk OFF
}
else
{
  Led1.on(); // LED Virtual no Blynk ON
}
}

void Temp() // Função para monitoramento de Temperatura e Umidade
{
  float h = dht.readHumidity(); // Definindo h p/ receber valor de umidade
  float t = dht.readTemperature(); // Definindo t p/ receber valor de temperatura

  if (isnan(h) || isnan(t)) {
    Serial.println("Falha ao ler o sensor DHT!");
    return; }

  Blynk.virtualWrite(V6, h); // Enviando h p/ o pino virtual V6 do Blynk
  Blynk.virtualWrite(V5, t); // Enviando t p/ o pino virtual V5 do Blynk
}

BLYNK_WRITE(V1)
{
  int pinValue = param.asInt(); // Requisitando valor binário do APP Blynk
  digitalWrite(LED_Red,pinValue);
}

void setup()
{
  Serial.begin(9600); // iniciando comunicação serial
  pinMode(pinSensorD, INPUT); // Entrada para sensor de Chuva

```

```
pinMode(LED_Red,OUTPUT);    // Saída para ligação do Led
digitalWrite(LED_Red,LOW);  // Mantendo Led inicialmente OFF
Blynk.begin(auth, ssid, pass); // Iniciando funções do Blynk
dht.begin();                // Iniciando funções do sensor DHT

timer.setInterval(1000L, Temp); // chamando a função Temp a cada segundo
timer.setInterval(1000L, Chuva); // chamando a função Chuva a cada segundo
}

void loop()
{
  Blynk.run(); // Mantém o código ativo (rodando)
  timer.run(); // Ativa instruções de temporização
}
```

## Documento Digitalizado Restrito

### Trabalho de Conclusão de Curso

**Assunto:** Trabalho de Conclusão de Curso  
**Assinado por:** Rodrigo Tomaz  
**Tipo do Documento:** Anexo  
**Situação:** Finalizado  
**Nível de Acesso:** Restrito  
**Hipótese Legal:** Informação Pessoal (Art. 31 da Lei no 12.527/2011)  
**Tipo do Conferência:** Cópia Simples

Documento assinado eletronicamente por:

- **Rodrigo Tomaz Pedrosa, ALUNO (201712030001) DE TECNOLOGIA EM AUTOMAÇÃO INDUSTRIAL - CAJAZEIRAS**, em 09/11/2021 10:12:47.

Este documento foi armazenado no SUAP em 09/11/2021. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse <https://suap.ifpb.edu.br/verificar-documento-externo/> e forneça os dados abaixo:

**Código Verificador:** 368542

**Código de Autenticação:** bfa89b04e9

