UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

LIVIA DE OLIVEIRA ALMEIDA

Controle de Ambiente por Voz usando Serviços em Nuvem e Microcontrolador de Baixo Custo

Sorocaba
2022

LIVIA DE OLIVEIRA ALMEIDA

Controle de Ambiente por Voz usando Serviços em Nuvem e Microcontrolador de Baixo Custo

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Eduardo Paciencia Godoy

Sorocaba
2022

A447c

Almeida, Livia de Oliveira

Controle de Ambiente por Voz usando Serviços em Nuvem e Microcontrolador de Baixo Custo / Livia de Oliveira Almeida. -- Sorocaba, 2022

52 p. : tabs., fotos

Trabalho de conclusão de curso (-) - Universidade Estadual Paulista (Unesp), Instituto de Ciência e Tecnologia, Sorocaba

Orientador: Eduardo Paciencia Godoy

1. Automação. 2. Automação Residencial. 3. Microcontrolador de Baixo Custo. 4. Serviços em Nuvem. 5. Comando por voz. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Controle de Ambiente por Voz usando Serviços em Nuvem e Microcontrolador de Baixo Custo.

LIVIA DE OLIVEIRA ALMEIDA

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO GRAU DE **BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

PROF. DR. MAURÍCIO BECERRA VARGAS

COORDENADOR

BANCA EXAMINADORA:

Prof. Dr. Eduardo Paciencia Godoy
Orientador/UNESP-Campus de Sorocaba

Prof. Dr. Maurício Becerra Vargas
UNESP- Campus de Sorocaba

Me. Rodrigo Pita Rolle
UNESP – Campus de Sorocaba

Janeiro de 2022

AGRADECIMENTOS

Agradeço a minha família por sempre estar presente me apoiando nas minhas decisões e conquistas. Em especial, agradeço ao meu companheiro Breno, por me incentivar tanto e ser meu porto seguro em todos os momentos. Agradeço aos meus colegas de curso, pelas grandes amizades que construí e que tornaram a trajetória mais leve e divertida. Agradeço aos professores pelos ensinamentos e aos técnicos e funcionários por todo suporte. Agradeço ao Prof. Dr. Eduardo Paciência Godoy pela oportunidade e orientação neste trabalho e nas disciplinas que foram essenciais para sua realização.

ALMEIDA, L. O. CONTROLE DE AMBIENTE POR VOZ USANDO SERVIÇOS EM NUVEM E MICROCONTROLADOR DE BAIXO CUSTO. TRABALHO DE GRADUAÇÃO (ENGENHARIA DE CONTROLE E AUTOMAÇÃO) – INSTITUTO DE CIÊNCIA E TECNOLOGIA DE SOROCABA, UNESP - UNIVERSIDADE ESTADUAL PAULISTA, SOROCABA, 2021.

RESUMO

Seguindo uma forte tendência da criação de tecnologias pensadas com o propósito de garantir o conforto do indivíduo, a automação residencial tem ganhado cada vez mais espaço no mercado. Recentemente, com a evolução dos assistentes virtuais, esse modelo de automação tem investido em comandos por voz. A princípio, os comandos por voz foram utilizados para pequenas pesquisas e leitura de mensagens em celulares. No entanto, esse recurso tem evoluído muito com a possibilidade de interações mais inteligentes. Os recursos dos assistentes virtuais estão sendo repensados para que a comunicação entre humanos e máquinas seja a mais natural possível. Dentro deste contexto, surgiram os dispositivos, tal como a Alexa, dedicados a atender pequenos comandos, tal como ligar uma música e dar informações a respeito do clima. Com a evolução da automação residencial, há a possibilidade de dar comandos para acionamentos de diversos eletrodomésticos, que aos poucos vão se adaptando à nova realidade. No entanto, essas soluções ainda são caras e demoradas. O trabalho em questão busca encontrar uma solução para controle remoto de ambientes. Com apenas um *smartphone*, utilizando um serviço em nuvem e um microcontrolador de baixo custo, o usuário deve conseguir gerenciar entradas e saídas, digitais e analógicas de sensores e outros dispositivos a quilômetros de distância. Uma opção definida neste estudo foi a utilização do aplicativo Google Home. Com essa ferramenta, que integra o Google Assistant, foi possível saber e alterar as condições de um ambiente remoto através de comandos de voz dados através de um celular. Assim, o usuário aciona o Google Assistant por voz, o aplicativo Google Home interpreta o comando e o vincula com os dispositivos já configurados em sua plataforma. Os dispositivos presentes no aplicativo se comunicam com um ambiente no Node-RED por meio do serviço Smart NORA. O ambiente Node-RED é criado em uma máquina virtual dentro da Google Cloud. O microcontrolador se comunica com o ambiente Node-RED usando o protocolo MQTT. Ademais, os comandos podem ser feitos para apenas obter informações recebidas pelo microcontrolador ou para comandar sinais que saem do microcontrolador.

PALAVRAS-CHAVE: GOOGLE HOME, ESP32, MQTT, SMART NORA, NODE-RED.

ALMEIDA, L. O. ENVIRONMENT CONTROL BY VOICE COMMANDS USING CLOUD SERVICES AND LOW-COST MICROCONTROLLER. GRADUATION PROJECT (BACHELOR'S DEGREE IN CONTROL AND AUTOMATION ENGINEERING) – SOROCABA INSTITUTE OF SCIENCE AND TECHNOLOGY, UNESP – SÃO PAULO STATE UNIVERSITY, SOROCABA, 2021.

ABSTRACT

Following a strong trend of creating technologies designed with the purpose of guaranteeing the individual's comfort, home automation has been gaining more and more space in the market. Recently, with the evolution of virtual assistants, this automation model has invested in voice commands. At first, voice commands were used for short searches and reading messages on cell phones. However, this feature has come a long way with the possibility of smarter interactions. The capabilities of virtual assistants are being rethought so that communication between humans and machines is as natural as possible. Within this context, devices such as Alexa emerged, meant to follow small commands, such as turning on music and giving information about the weather. With the evolution of home automation, it is now possible to give commands to various household appliances, which gradually adapt to the new reality. However, these solutions are still expensive and time-consuming. The work in question seeks to find a solution for remote control of environments. With just a smartphone, using a cloud service and a low-cost microcontroller, the user should be able to manage digital and analog inputs and outputs from sensors and other devices miles away. One option defined in this study was to use the Google Home application. With this tool, which integrates Google Assistant, it is possible to know and change the conditions of a remote environment through voice commands using a cell phone. Thus, the user activates the Google Assistant by voice, the Google Home app interprets the command and links it to the devices already configured on their platform. Devices present in the app communicate with an environment on Node-RED through the Smart NORA service. The Node-RED environment is created in a virtual machine inside the Google Cloud. The microcontroller communicates with the Node-RED environment using the MQTT protocol. Moreover, commands can be made to only receive information received by the microcontroller or to command signals that come out of the microcontroller.

KEYWORDS: GOOGLE HOME, ESP32, MQTT, SMART NORA, NODE-RED.

LISTA DE ABREVIAÇÕES

TCP/IP - *Transmission Control Protocol/ Internet Protocol*

BLE - *Bluetooh Low Energy*

IFTTT - *If This Then That*

JVM - *Java Virtual Machine*

NLU - *Natural-Language Understanding*

MQTT - *Message Queuing Telemetry Transport*

WLAN - *Wireless Local Area Network*

MAC - *Media Access Control*

WPS - *Wi-Fi Protect Setup*

SD - *Secure Digital*

SDIO - *Secure Digital Input Output*

MMC - *Microgrid Master Controller*

UART - *Universal Asynchronous Receiver-Transmitter*

SRAM - *Static Random-Access Memory*

ROM - *Ready-only Memory*

FreeRTOS - *Free Real-Time Operating System*

RC - *Resistência Capacitância*

PWM - *Pulse-Width Modulation*

JSON - *JavaScript Object Notation*

UDP - *User Datagram Protocol*

HTTP - *Hypertext Transfer Protocol*

URI - *Uniform Resource Identifier*

TLS - *Transport Layer Securtiy*

LAN - *Local Area Network*

DHCP - *Dynamic Host Configuration Protocol*

BBS - *Basic Service Set*

SoftAP - *Software Enabled Access Point*

RTS - *Request to Send*

CTS - *Clear to Send*

ACK - *Acknowledgement*

WebRTC - *Web Real-Time Communication*

ICE - *Interactive Connectivity Establishment*

SDP - *Session Description Protocol*

RGB - *Red Green and Blue*

HSV - *Hue, Saturation and Value*

API - *Application Programming Interface*

IoT - *Internet Of Things*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

1.1 Justificativa

Não é à toa que muitos se assustam com o espaço que a tecnologia ocupa na vida humana nos dias de hoje. Quando se fala em tecnologia, fala-se em todas as conquistas humanas ao controlar a natureza em favor próprio. Desta forma, pode-se afirmar que a história da tecnologia se inicia há 3,3 milhões de anos atrás, idade das primeiras ferramentas de pedra descobertas. Com o decorrer do tempo, foi possível dominar o fogo, a agricultura, navegar pelos oceanos, produzir motores à combustão, entre outras conquistas. Cerca de 140 anos atrás, Thomas Edison, com a invenção da lâmpada, deu a largada a uma era totalmente revolucionária. Há mais ou menos 120 anos, Gugliemo Marconi inventou o rádio, há 90 anos surgiu a televisão. Um grande passo foi dado há 70 anos com o surgimento do primeiro transistor. Há somente 40 anos surgiu o primeiro computador pessoal e possibilidade de transmissão de dados via TCP/IP (*Transmission Control Protocol/ Internet Protocol*), conhecida como Internet. O ritmo da evolução tecnológica acelera cada vez mais e não é por acaso. A possibilidade de contar com as tecnologias no dia a dia faz com que tenhamos mais tempo para nos aplicarmos a questões mais complexas. Não é necessário nos preocuparmos mais com a caça para não morrermos de fome, ou com a confecção de cópias manuais de livros para guardar o conhecimento, ou mesmo passarmos horas viajando ou nos deslocando para consultar informações necessárias para solucionar problemas que surgem.

À medida que a tecnologia apoiou as indústrias, aquilo que antes era fabricado manualmente e sob encomenda passou a ser fabricado em larga escala com ajuda de máquinas especializadas. Conforme a humanidade foi se desenvolvendo, novas necessidades foram surgindo e novas exigências passaram a ser feitas para o mercado. A primeira revolução industrial no século 18 foi marcada pelo uso de ferro e aço e uso de novas fontes de energia, tal como petróleo e combustão interna. Já a segunda revolução industrial ocorreu entre os séculos 19 e 20. Com ela, passou-se a produzir ferro e aço em larga escala, a fazer uso generalizado das máquinas e a usar a energia elétrica. A terceira revolução industrial ou o surgimento da Indústria 3.0, há 30 anos atrás, teve como principais acontecimentos, a evolução dos semicondutores, a criação dos computadores pessoais e *mainframe*, automatização de processos com supervisão humana e a possibilidade de acesso à Internet. Há pouco tempo, a Indústria 4.0 começou a ser tema das discussões. Marcada pela quarta

revolução industrial, é caracterizada pelo uso de tecnologias como Internet das Coisas, *Big Data*, impressão 3D, computação em nuvem, produção customizada, realidade aumentada, *cyber* segurança, robótica autônoma, integração de sistemas, Inteligência Artificial e manufatura aditiva. Com isso, pode-se observar a crescente necessidade de tornar cada vez mais fácil e lucrativa a produção, mas também a volta da busca por produtos únicos.

Hoje é possível notar uma forte tendência estabelecida. Tendência esta que visa uma maior preocupação com o bem estar individual. As pessoas estão mais preocupadas em potencializar o tempo em que não estão trabalhando ou dormindo. A tecnologia ganhou um novo espaço com essas novas necessidades. As pessoas não querem mais passar horas no trânsito sem que consigam pelo menos aproveitar esse tempo de alguma outra maneira, seja praticando um outro idioma, seja respondendo um *e-mail* que ficou para trás há dias. A rapidez com que desejamos que nossos problemas sejam resolvidos, exige também respostas imediatas da nossa parte. Desta forma é natural que desejemos estar sempre conectados. Algumas pessoas veem este desejo como uma característica prejudicial, mas podemos identificar diversas maneiras de gerenciar nossa vida e permitir que a tecnologia contribua positivamente para ela.

Seguindo esta linha de tecnologias individualizadas visando o bem estar do indivíduo, a automação residencial tem ganhado cada vez mais espaço no mercado. A princípio surgiram os assistentes virtuais que passaram a auxiliar o usuário com pequenas pesquisas e ações por comando de voz. Com a evolução da inteligência artificial novas habilidades foram incorporadas e passaram a surgir pequenos dispositivos dedicados exclusivamente para atender esses comandos, tal como a Amazon Alexa sendo utilizada no Amazon Echo. A assistente Alexa pode ser questionada a respeito do tempo, pode inicializar uma música ou realizar pesquisas na internet. Surgiu também o Google Nest integrado com o Google Home. Independente dos dispositivos dedicados, as funcionalidades foram incorporadas nos *smartphones*. Com isso, eletrodomésticos estão sendo desenvolvidos para serem capazes de se comunicarem com os *smartphones*. Mas como essa evolução ainda é algo recente, muitos dispositivos que oferecem essa integração ainda estão longe de poderem ser consumidos pela maioria das pessoas por conta dos preços elevados.

Nesse contexto, visando fornecer soluções com alto valor tecnológico e preços mais acessíveis, o projeto atual propõe a implementação de um sistema no qual os usuários podem controlar remotamente ambientes, em termos de temperatura, umidade, iluminação, utilizando comandos por voz utilizando basicamente uma placa ESP32 e um *smartphone*.

1.2 Objetivos

O trabalho desenvolvido tem como objetivo geral oferecer uma solução que permita o controle de ambientes via comandos de voz a baixo custo. Com auxílio de um microcontrolador ESP32, uma rede de internet Wi-Fi e sensores, deve ser possível desenvolver um produto que possibilite ao usuário receber informações sobre e determinar variáveis de um ambiente controlado realizando um pedido em voz alta para o assistente de voz da Google. O objetivo secundário do projeto prevê a exploração de conhecimentos de Internet das Coisas e computação em nuvem. Utilizando essas tecnologias, o usuário será capaz controlar um ambiente de forma remota.

Além disso, como objetivo específico, o material produzido neste projeto em questão poderá ser utilizado em experimentos de laboratório pelos alunos da disciplina de Rede Industriais e de Comunicação do curso de Engenharia de Controle e Automação.

1.3 Estrutura do Trabalho

O trabalho pretende, primeiramente, explorar alguns conceitos necessários para o entendimento da solução. Em seguida, é mostrado como o desenvolvimento foi realizado, destacando pontos mais relevantes. Por fim, são discutidos os resultados obtidos e possíveis explorações a partir da solução desenvolvida.

2. REVISÃO BIBLIOGRÁFICA

2.1 Automação residencial no cenário atual

Combinando a aceleração da migração para as nuvens causada pela pandemia e a necessidade de potencializar o tempo fora do trabalho, fica cada vez mais factível que as pessoas se desloquem cada vez menos e passem a realizar todas as tarefas no conforto de suas casas. Com isso, as casas devem estar ainda mais preparadas e equipadas para minimizar atividades necessárias que não promovem prazer ao serem realizadas, tal como lavar a louça, varrer o chão etc.

Existem algumas soluções que podem ser encontradas no mercado. Como por exemplo o robô aspirador ou os interruptores inteligentes, cada vez mais comuns. Até mesmo as máquinas de lavar louças e refrigeradores já estão sendo integrados com os assistentes virtuais. A Samsung está investindo fortemente neste setor, sendo que seus celulares já são configurados com o aplicativo SmartThings, capaz de se comunicar com todos os outros dispositivos inteligentes da marca. Assim como a Amazon e a Google, a Samsumg também oferece um dispositivo dedicado com assistente virtual, o Samsung Hub. Soluções como as propostas viabilizam cada vez mais o acesso das pessoas à tecnologia. No entanto, os altos valores ainda limitam e muito o acesso.

Para obtenção de um dispositivo dedicado a assistentes virtuais tal como como o Echo Dot, o Google Nest e o Samsung Hub, pesquisando pelos preços atualizados, seria necessário um investimento de pelo menos 150 reais, podendo chegar até 1500 reais, a depender das funcionalidades. Isso sem contar os custos com os eletrodomésticos capazes de se comunicar com os dispositivos e as configurações que nem todo mundo consegue realizar sem ajuda profissional.

2.2 Soluções alternativas

Considerando o contexto geral de soluções de automação residencial usando serviços em nuvem, Pujari, Patil, Behadure e Asnodkar (2020) mapeiam aplicações de Internet das Coisas em sistemas inteligentes de automação residencial. No artigo citado, os autores propõem a elaboração de um sistema de monitoramento de variáveis tais como temperatura, umidade e luminosidade, utilizando o microcontrolador ESP32 com controle remoto. Para esta aplicação, seria elaborado um aplicativo de Android conectado a uma base de dados alocada na nuvem da Google, que teria uma interface para monitoramento em tempo real. O

ESP32 faria sua comunicação via Wi-Fi e Bluetooh Low Energy (BLE) com os sensores e com o aplicativo.

Uma aplicação de monitoramento por voz usando o Google Assistant, foi proposta por Hadi, Shidiqui, Zaeni et al (2019, pp. 106-110). A aplicação proposta usa a API DialogFlow, disponibilizada pela Google e que tem o propósito de sustentar uma conversação mais próxima da humana, utilizando para isso ferramentas de inteligência artificial. A entrada da voz seria feita através de um *smartphone* e inserida na API por meio do Google Assistant. A API estaria hospedada na nuvem e o áudio de entrada seria direcionado para o Google Assistant Server e tratado com a interface Actions-on-Google. Neste contexto, o servidor se comunicaria com a API, que consultaria a base de dados para retornar algo ao usuário e gerar um *input* para alguma aplicação. Para exemplificar a aplicação, os autores demonstram como seria um sistema de irrigação. Além disso, os autores mostram os resultados do uso de Machine Learning, obtendo 75% de sucesso de acerto nos testes com a API.

Rajalakshmi e Shahnasser (2018, pp. 1-4) propuseram uma aplicação de Internet das Coisas usando Node-Red e a assistente Alexa. Para isso, os autores utilizam um microcontrolador Raspberry Pi 3, uma placa Intel Edison, um ESP32 e um ESP8266. O Node-Red é instalado no Raspberry Pi e usa o protocolo MQTT(*Message Queuing Telemetry Transport*) e os demais dispositivos se comunicam com os sensores via Wi-Fi. Para armazenamento e tratamento dos dados, utiliza-se os serviços de web da Amazon, tais como, AWS IoT, AWS Lambda, AWS EC2 e a assistente Alexa. Os dados de monitoramento seriam mostrados em um dashboard do Node-Red.

Quanto à comunicação entre o assistente virtual, o banco de dados e possíveis APIs que irão associar ações aos dispositivos conectados, pode-se pensar nos serviços em nuvem: DialogFlow do Google, Watson da IBM, Amazon Lex e Azure Bot Service da Microsoft. Há ainda a possibilidade de utilizar a aplicação IFTTT (*If This Then That*) e a plataforma Adafruit IO, que é uma plataforma de IoT como sugerido em um projeto do site Flip Flop (NASCIMENTO, 2020).

A aplicação IFTTT tem como principal funcionalidade ajudar na comunicação entre aplicativos e dispositivos. Ou em outras palavras, a solução ajuda o usuário a conectar serviços utilizando Applets. Um Applet é um programa desenvolvido em Java que pode ser adicionado a uma página HTML, tal como é adicionada uma imagem. No entanto, quando o navegador suporta aplicações Java, o código do Applet é transferido para seu sistema e executado por um navegador de uma Máquina Virtual Java (JVM). Por meio dos Applets, a

aplicação IFTTT pode ser associada à Alexa da Amazon, ao calendário do Google, ao Twitter, ao Google Assistant, por exemplo. O IFTTT é uma maneira fácil de chamar serviços conectados com os Applets, caso alguma condição prévia seja satisfeita. Para utilizá-lo, basta primeiro selecionar um serviço que receberá as entradas, e quais entradas serão consideradas, depois disso, deve-se selecionar o serviço que executará a ação e qual será ela. Para conexão com os serviços, é possível usar o modo não autenticado e o modo autenticado que exige uma conta e senha na aplicação utilizada. Além disso, o IFTTT oferece alguns modelos prontos para utilização, com os serviços já associados, bastando customizar somente a entrada e a ação. É possível utilizar a versão gratuita do serviço IFTTT, porém há um limite para quantidade de Applets que podem ser utilizados, bem como para quantidade de requisições e lógica condicional, tempo de execução e suporte customizado. Uma limitação que essa solução pode apresentar é a comunicação unilateral. Para que fosse possível construir uma comunicação mais próxima da linguagem natural entre o usuário, o Google Assistant e os dispositivos a serem controlados, seria necessário criar muitas relações do tipo IFTTT independentes, o que seria um problema na versão gratuita, devido às limitações já mencionadas.

A companhia Adafruit foi fundada em 2005 pela Limor Fried, uma engenheira do MIT. Sua plataforma Adafruit IO oferece a possibilidade de conectar projetos à Internet com a promessa de manter os dados do usuário totalmente privados por padrão. Trata-se de um serviço de nuvem focado em desenvolvimento de soluções IoT. Oferece integração com *dashboards* para gerenciamento de dados de sensores, contendo botões para acionamentos de dispositivos remotos. Além de oferecer suporte para comunicação com o serviço IFTTT e Zapier, comercializa os próprios dispositivos, que muitas vezes podem ser difíceis de serem encontrados. Esse serviço em nuvem oferece um plano gratuito, mas com limitação de 30 dados por minuto, possibilidade de armazenamento na nuvem por 30 dias e 5 entradas. O plano pago custa 10 dólares por mês e é limitado a 60 dados por minuto e 60 dias de armazenamento e oferece a opção de *inputs* ilimitados, como informado no site do provedor.

Quando se escolhe utilizar os recursos IFTTT e a plataforma Adafruit, as ações sempre serão limitadas a entradas muito bem definidas e cadastrados previamente. Uma opção mais interessante, dada a evolução da computação e da interação com os usuários, é utilizar os recursos de para processamento de linguagem natural em nuvem, que comportam entradas mais simples, mas também podem ser escalados com soluções mais complexas.

Podemos comparar as opções entres os serviços mais conhecidos que são: o DialogFlow da Google, o Watson Assistant da IBM, a Amazon Lex e Azure Bot Service da Microsoft. Quando se trata de plataformas para *voicebots*, que o é como se espera que a interação ocorra, deve se considerar alguns aspectos como: seus recursos para processamento de linguagem natural, sua capacidade de entender o contexto, manter a coerência e fazer predições, devem ser consideradas questões a respeito da segurança dos dados, a possibilidade de automatizar sequências e sua rapidez e eficiência para responder, o preço para sua manutenção e a sua compatibilidade com os dispositivos e serviços.

O Amazon Lex também pode interpretar tanto texto, quanto voz. É capaz de converter fala em texto e texto em fala, incluindo experiências do usuário nas interações. Pode se comunicar com Facebook Messenger, Kik, Slak e Twilio SMS. O IBM Watson Assistant é um serviço que tenta imitar interações humanas, pode ser usado com serviços em nuvem ou *on-premises*. O assistente pode se comunicar com o Facebook Messenger também, com agentes de voz de celulares e outras APIs. O Azure Bot Service conta com uma ferramenta poderosa de inteligência artificial usando o Azure Cognitive Services. É mais bem integrado com os próprios serviços da Microsoft, tal como Teams e Skype, por exemplo. Mas também pode se comunicar com o Facebook Messenger.

O DialogFlow era conhecido previamente como API.ai e foi adquirido pela Google em 2016. O serviço oferecido pela Google é capaz de interpretar múltiplas entradas, incluindo áudios de um celular ou de um gravador de voz e é capaz de responder ao usuário de várias maneiras com uma interpretação do contexto. Na visão de Fernandes (2020) o DialogFlow é uma ferramenta fácil de se utilizar, mesmo por quem não tem muitas habilidades técnicas e é o modo mais rápido para criar um *bot* conversacional inteligente. Pode ser integrado ao Google Assistant, websites, Slack, Facebook Messenger, Skype, Twitter e muitos outros. Segundo um artigo comparativo, escrito por Jeff Petters (2020), os serviços em nuvem da Google são mais adequados para desenvolvimento de aplicativos em nuvem, além de oferecer soluções ecologicamente conscientes. Oferece dois tipos de agente virtual, a saber, o CX e o ES. O primeiro fornece um suporte mais avançado para agentes grandes ou muito complexos. O ES é adequado para agentes menores ou mais simples.

2.3 Assistentes virtuais

Partindo do ponto de vista da interação com o usuário, é possível listar 4 principais assistentes virtuais acionados por voz, a saber, a Alexa da Amazon, a Siri da Apple, a Cortana da Microsoft e o Google Assistant.

O assistente virtual da Google foi introduzido pela primeira vez ao público em maio de 2016, disponível para dispositivos Android e iOS. Em dezembro de 2016, a Google lançou sua plataforma "Actions on Google", que possibilitou a execução de ações solicitadas pelo usuário, via assistente virtual e em pouco tempo já havia milhares de serviços disponíveis baseados em ações provenientes da interação por voz com o usuário. Segundo Matt Lawson (2017), já no começo de 2017, 20% das pesquisas eram feitas por comando de voz usando o assistente e estimava-se que em 2020 50% das pesquisas fossem realizadas usando voz. Ainda em 2017, o Google Assistant foi lançado em português. No começo de 2018, empresas como Lenovo, Sony, JBL e LG já ofereciam compatibilidade com o serviço. Em maio de 2018, 46% dos *smartphones* possuíam acesso ao assistente virtual e mais de 5 mil dispositivos conectados.

A assistente Siri foi lançada pela Apple em Outubro de 2011, em uma atualização do iPhone 4S. Até 2016, a assistente só estava disponível para produtos Apple. Após esta data, passou a ser disponibilizada em aplicações de terceiros. Segundo Yoffie et al (2018, pp. 1-25), as categorias de aplicação eram restritas a chamadas de voz, pesquisa, mensagens e serviços *rife-hailing*, visando a proteção da privacidade do usuário.

A Alexa surgiu a partir da tecnologia desenvolvida por uma pequena empresa britânica chamada Evi, que divulgou a criação do assistente em 2012. Em novembro de 2014, a Amazon lançou a versão da Alexa via Echo. Em abril de 2019, sua versão em português foi anunciada e já tinha 90 mil funções disponíveis para *download* nos dispositivos. As ferramentas para automação residencial foram lançadas em abril de 2015, com o kit Alexa Skills. Seu serviço de voz oferece reconhecimento automático de voz (ASR) e entendimento de linguagem natural (NLU). No fim de 2017, o serviço já tinha mais de 25 mil usuários.

A Microsoft lançou a Cortana em abril de 2014, com o *update* do sistema operacional de seu dispositivo móvel. Em 2015 e 2016, o serviço passou a ser compatível com iOS e Android. No ano de 2017, foi lançado o kit Cortana Skills e, no começo de 2018, o assistente possuía 235 habilidades aprendidas, o que é muito menos do que os demais assistentes.

Segundo uma pesquisa realizada pela Reviews (2020), entre os três melhores assistentes de 2020, a saber, Alexa, Siri e Google Assistant, o último é o que oferece a melhor

acurácia nas respostas e é o segundo em quesito maior compatibilidade, perdendo para a Alexa. A pesquisa considerou um estudo realizado em 2018, pela Perficient Digital, que testou os assistentes fazendo mais de 4000 perguntas a eles. Um dos pontos contra levantados pelos usuários é que o assistente Alexa não é pré-instalado nos *smartphones* e que para se ter acesso é necessário obter o Amazon Alexa App, diferentemente do Google Assistant ou da Siri que atua independente de um aplicativo específico.

Uma publicação de Berdasco et al. (2019), em uma pesquisa realizada em 2019, pela avaliação dos usuários, os melhores assistentes são o Google Assistant e a Alexa, apresentando a descrição excelente em quesito qualidade e exatidão na resposta dada. Em um artigo, Thamp et al. (2021) mostra que, segundo os usuários consultados, o assistente virtual da Google apresenta maior qualidade de reconhecimento de voz e melhor interação livre com o humano, já que é capaz de entender uma maior variação de voz.

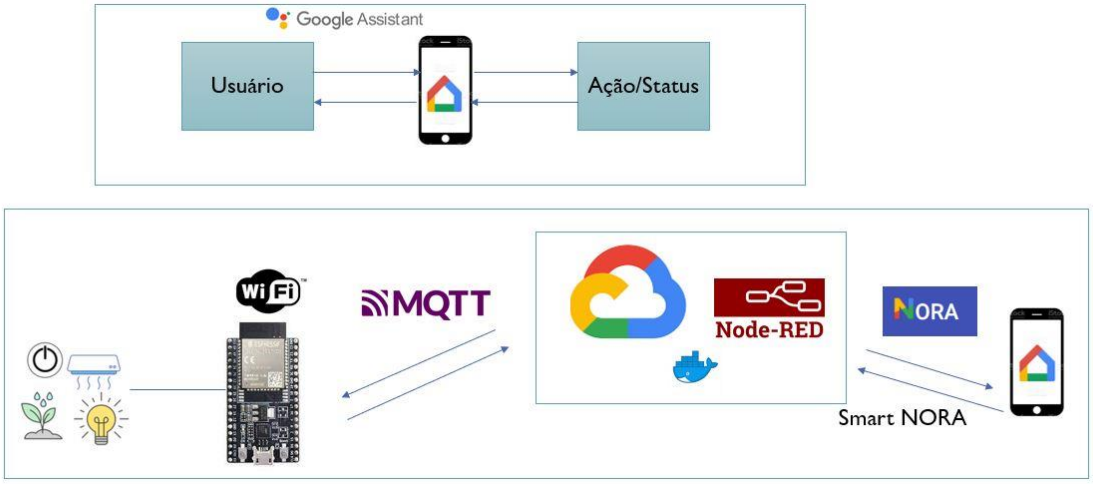
3. CONTROLE DE AMBIENTE POR VOZ

O trabalho em questão fez uso de produtos da Google, tais como do aplicativo Google Home, do Google Assistant e da Google Cloud. Foi utilizado também soluções como o serviço de integração Smart NORA e o serviço Node-RED. Além destas ferramentas, foi trabalhado com a plataforma Arduino e o microcontrolador ESP32, utilizando o protocolo MQTT.

3.1 Proposta do Trabalho

O trabalho realizado visa propor uma opção mais barata para controle de ambiente, integrando recursos já existentes no mercado. Com o aplicativo Google Home, sem a necessidade de um dispositivo como o Google Nest ou Amazon Echo, usando o ESP32, é possível consultar sensores e dar comandos para controlar a iluminação, o ar condicionado e afins. Usando apenas uma máquina virtual em nuvem e o serviço Smart NORA, o controle por voz, utilizando qualquer celular, fica disponível de forma remota. O fluxo pode ser visto na figura 1.

Figura 1 - Arquitetura do Sistema.



Fonte: Autor

Em um primeiro momento, o usuário irá realizar o *download* do aplicativo Google Home fornecido pelo Google, se seu celular já não estiver disponível. O aplicativo já possui integração com o Google Assistant, fazendo com que todos os acionamento possam ser realizados via comando de voz. Os comandos são inicializados em qualquer situação utilizando a frase “Hey Google”. Portanto, quando o usuário desejar se informar sobre ou

alterar qualquer variável do ambiente, ele deve aproximar-se do *smartphone* e dizer a frase já descrita.

O Google Home é configurado via Google Cloud, onde está alocada uma máquina virtual com um ambiente Node-RED. O Node-RED oferece um espaço para configurar todos os fluxos entre a comunicação com o aplicativo e os comandos que serão enviados para a placa ESP32. A comunicação entre o Google Home e o ambiente Node-RED é facilitado por um serviço chamado Smart NORA. Este serviço disponibiliza nós pré configurados que acionam ações no Google Actions aos dispositivos a serem configurados. Este serviço é ativado realizando um cadastro que será vinculado com o cadastro no Google Home e nas configurações dos nós no ambiente Node-RED.

Dentro do Node-RED, alocado na Google Cloud, as ações serão ligadas à ESP32 através de fluxos, onde as saídas e entradas da placa são definidas através de tópicos. Isto porque a comunicação que é estabelecida entre o microcontrolador e o ambiente em nuvem utiliza o protocolo MQTT. Desta forma, as entradas do ESP32 são publicadas em tópicos, lidos por nós no Node-RED, e as saídas são lidas para ESP32, enviadas por publicações no ambiente Node-RED.

Para que o microcontrolador saiba quais tópicos deve publicar, quais ele deve se inscrever e o que deve fazer com cada informação, é utilizada a Arduino IDE. Dentro de um programa escrito em C++, é possível configurar todas as saídas e entradas do ESP32. Para representar os possíveis dispositivos que receberiam ou enviariam informações para o microcontrolador, foram utilizados LEDs e um sensor.

Da forma como o projeto foi realizado, seria possível substituir os LEDs e sensor por quaisquer dispositivos desejados. O projeto considerou saídas e entradas digitais e analógicas.

Utilizando essa solução, seria possível ajustar tanto iluminação, quanto temperatura, umidade, acionar qualquer tomada, sem precisar estar no mesmo ambiente que os dispositivos, sendo necessário apenas ter acesso à rede de internet.

3.2 Materiais e métodos

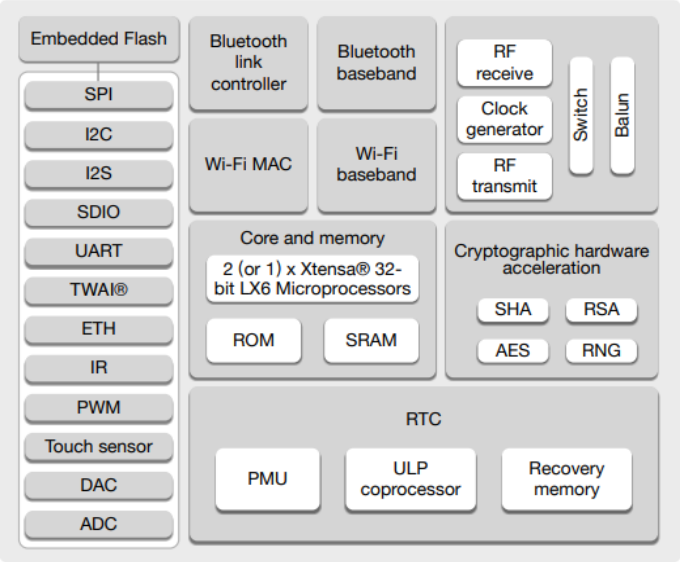
3.2.1 ESP32

O ESP32 é um microcontrolador de baixo custo que suporta comunicação sem fio via Wi-Fi e Bluetooth. O fato de ser pequeno faz com que possa ser incorporado em soluções de

pequeno porte. A placa opera em uma grande faixa de temperatura, que varia de -40°C a 125°C. Apresenta baixo consumo de energia, além de possuir recursos como modos de energia e escalonamento dinâmico de energia. Por esses motivos, é destaque em aplicações de Internet das Coisas.

Este microcontrolador permite comunicação usando TCP/IP, WLAN (*Wireless Local Area Network)* MAC (*Media Access Control*) e Wi-Fi, suportando níveis de segurança Wi-Fi WPA (*Wi-Fi Protect Access*), WPA2, WPA2-Entreprise e WPS (*Wi-Fi Protect Setup*). É possível operar como ponto de acesso com interface para o usuário, usando Ethernet MAC, Controlador hospedeiro SD/SDIO/MMC (*Secure Digital/ Secure Digital Input Output/ Microgrid Master Controller)* e duas interfaces UART (*Universal Asynchronous Receiver-Transmitter*). Tem estrutura Tensilica Xtensa LX6, com armazenamento de 32 *bits*, velocidade entre 160 e 240MHz, memória SRAM (*Static Random-Access Memory*) de 520KB, *flash* de 4MB e ROM (*Ready-only Memory*) de 448KB. Usa o sistema operacional FreeRTOS (*Free Real-Time Operating System*), *open source*, gerenciado pelo MIT. Para programá-lo é possível usar o framework ESP-IDF da Linux, o MSYS2 no Windows ou mesmo a IDE do Arduino. Neste trabalho foi utilizado a interface de desenvolvimento do Arduino. Sua arquitetura pode ser vista na figura 2.

Figura 2 - Diagrama de blocos da arquitetura do microcontrolador ESP32.



Fonte: ESP32 Datasheet Espressif System[1].

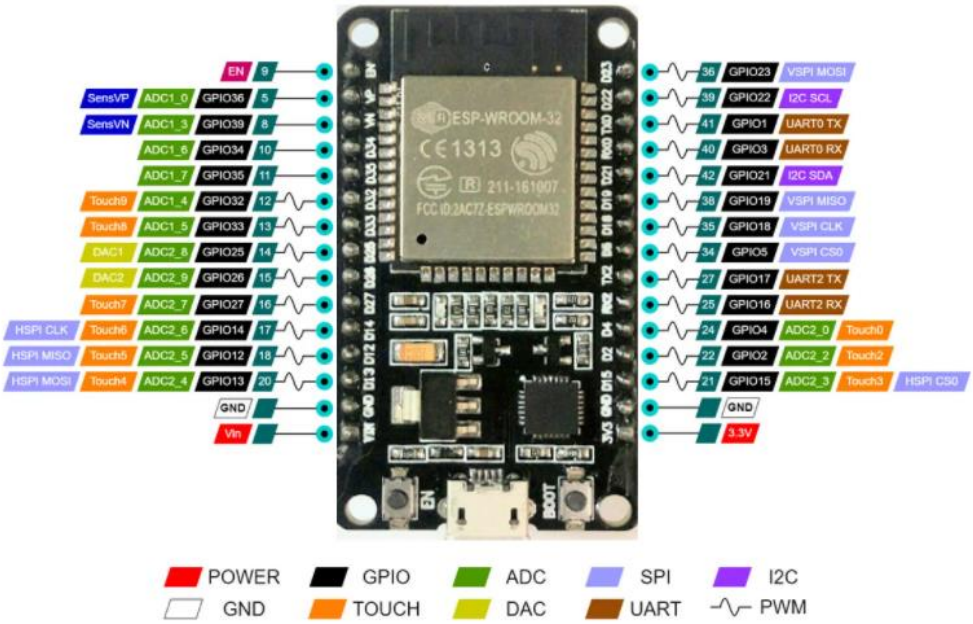
[1] Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

O ESP32 conta com um oscilador interno de 8MHz com calibração e um oscilador Resistência Capacitância (RC) também com calibração. Quanto aos periféricos, conta com interface Ethernet MAC, motor PWM (*Pulse-Width Modulation*), LED PWM e 25 pinos de propósito geral.

É possível também gerenciar o modo de energia do microcontrolador com cinco modos: ativo, modem desativado, parcialmente desativado, totalmente desativado ou em hibernação.

É importante destacar que existem muitas opções de placas ESP32 com mais ou menos recursos. Para este trabalho foi utilizada uma placa ESP32 de 30 pinos tal como mostrado na figura 3.

Figura 3 - Desenho esquemático da placa ESP32 com caracterização dos pinos.



Fontes: Montagem realizada pelo autor.[2]

Como mostra a legenda da figura 3, os pinos com marcação em azul são pinos seriais, os em verde são pinos analógicos, os em amarelo são pinos de controle e os em cinza são pinos que suportam dados digitais. Dentre os pinos analógicos, somente o ADC1_0, ADC1_3, ADC1_6 e ADC1_7 não têm ligação PWM.

[2] Disponível em: https://microcontrollerslab.com e https://deepbluembedded.com.

3.2.2 Arduino IDE

No desenvolvimento da aplicação para ESP32 foi utilizada a plataforma de desenvolvimento do Arduino. O ambiente de desenvolvimento do Arduino é *open source* e possui uma interface muito fácil de manipular. A IDE foi desenvolvida em Java e é multiplataforma. Nela é possível programar em C e C++.

O código desenvolvido na plataforma do Arduino foi utilizado para realizar a comunicação da placa ESP32 com o ambiente Node-RED na máquina virtual na Google Cloud, via MQTT. Foram utilizadas as bibliotecas *ArduinoJson.h, WiFi.h, PubSubClient.h* e a *SimpleDHT.h*.

A biblioteca *ArduinoJson* foi necessária visto que a inscrição em vários tópicos diferentes poderia atrasar ou mesmo confundir os dados recebidos pela placa. Desta forma, foi realizada a inscrição em apenas um tópico e realizada a separação dos dados em subtópicos. Para facilitar e até minimizar o tamanho da mensagem, os dados recebidos foram enviados em uma estrutura JSON(*JavaScript Object Notation*). Esta biblioteca permite o armazenamento de dados do JSON em um doc que utiliza memória da pilha (forma estática) ou aloca memória de forma dinâmica. Para pode utilizar o método estático deve tomar cuidado com o tamanho do arquivo para que não ocorra *overflow* (BLANCHON, 2014-2021).

Já a biblioteca *WiFi.h* foi utilizada para realizar a comunicação da placa com a rede via WiFi. Essa biblioteca permite configurar o usuário e senha da rede e dispõe de comandos para tanto iniciar a conexão quanto checar seu *status*.

Para realizar a comunicação da placa com o Node-RED, como já mencionado, foi utilizado o protocolo MQTT. A biblioteca *PubSubClient.h* dispõe de funções que facilitam a execução desse modo de comunicação. Por exemplo, com a função setServer, é definido qual *broker* e qual porta serão usados. Ou, então, na função setCallback foi possível estipular uma função para atualização de dados dos tópicos.

A biblioteca *SimpleDHT.h* fornece funções para leitura do sensor DHT11. Com ela é possível recuperar um objeto já com as informações de temperatura e umidade, além de permitir identificar quando a leitura foi realizada corretamente e quando não.

3.2.3 Google Cloud

Hoje existem muitos provedores de nuvem no mercado, tais como Oracle, AWS da Amazon, Azure da Microsoft, IBM e a Google. Foi escolhido a Google visando um trabalho mais integrado com o Google Assistant e deixando a possibilidade de exploração de outros recursos oferecidos pelo provedor. Além disso, sabe-se que o número de pessoas utilizando celulares Android é muito maior que o das pessoas que usam outros sistemas operacionais. Em muitos aparelhos Android funções como o Google Assistant já vêm previamente configuradas.

Após habilitado o teste gratuito na Google Cloud Platform, será disponibilizado um crédito de 300 dólares, no período de 90 dias para o usuário. Além disso, se o usuário permanecer dentro dos limites do nível gratuito, ele poderá manter seu serviço sem nenhum custo após o período de teste. Para o serviço utilizado neste trabalho, o limite do nível gratuito permitiria a criação de uma instância da VM e2-micro não preemptiva por mês para as regiões de us-west1, us-central1 e us-east1 com um disco permanente padrão de 30GB por mês (GOOGLE, 2021).

Cabe ressaltar que na solução proposta foi utilizado o mínimo de recursos oferecidos pela Google Cloud. O provedor de nuvem conta com APIs como a IoT Core, Cloud Pub/Sub, ou serviços como Google Actions, Google Functions e o Dataflow que poderiam ser integradas a soluções mais complexas.

3.2.4 Comunicação MQTT

O microcontrolador ESP32 pode se comunicar com outro dispositivo usando o protocolo TCP/IP. Este protocolo não corresponde diretamente a uma camada do modelo OSI, mas é uma referência feita a duas camadas, a saber, a de transporte usando o protocolo TCP e a de *network* usando o protocolo IP.

A camada de *network* é responsável por receber e enviar pacotes de dados. Usando o protocolo IP, um datagrama é criado indicando sua versão, a quantidade de dados sendo enviados, um campo para verificar posteriormente se todos os dados estão corretos, o tempo de vida, informações sobre o protocolo de transporte, o endereço IP da fonte e da origem, além de conter *flags* indicando se o datagrama faz parte de um pacote maior ou não.

A camada de transporte é onde é decidido qual cliente e servidor devem receber os dados. No caso do protocolo TCP a comunicação é feita estabelecendo uma troca de mensagens

entre o cliente e servidor, diferentemente do protocolo UDP (*User Datagram Protocol*) que apenas envia dados sem a necessidade de retorno do cliente. A conexão via protocolo TCP é estabelecida com uma troca de *flags* que permite a sincronização entre cliente e servidor, antes que o pacote de dados principal seja enviado. As informações dessa comunicação e os dados a serem enviados são colocados em um cabeçalho que está contido no datagrama do protocolo IP. Nesse cabeçalho estão presentes informações sobre a porta de origem e destinação, uma sequência de números para rastrear onde o segmento deve se encaixar para quando a mensagem é fragmentada, contém também informações sobre seu tamanho, mecanismos para checagem da mensagem, a mensagem e outras informações.

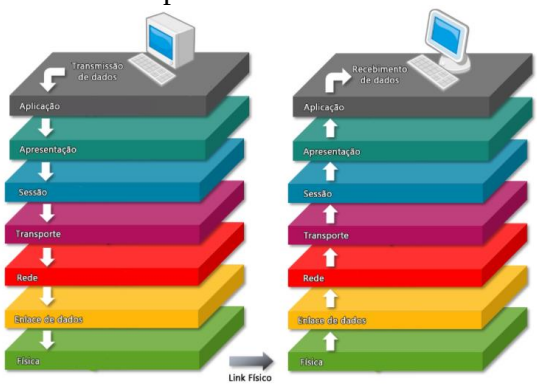
No nível da camada de aplicação pode-se utilizar o protocolo HTTP (*Hypertext Transfer Protocol*). Esse protocolo é baseado em requisições e respostas na forma de uma URI (*Uniform Resource Identifier*), que é um conjunto de caracteres que direcionam o cliente para determinado conteúdo. No entanto, dependendo do uso, esse protocolo é inseguro, já que pode expor informações no direcionamento. No contexto de aplicações para Internet das Coisas, é comum se utilizar o protocolo MQTT. Este protocolo permite que a comunicação seja leve por meio da assinatura e publicação de mensagens. Para que essa comunicação ocorra não são necessários grandes recursos computacionais ou de rede. A comunicação é intermediada por um *broker* que gerencia o fluxo das mensagens. Além disso, pode utilizar o protocolo de segurança TLS (*Transport Layer Securtiy*) que usa encriptação de mensagens e recursos de autenticação.

Em uma visão geral, a camada de aplicação gera uma mensagem que é encapsulada pelo cabeçalho da camada de transporte, que por sua vez é encapsulado pelo datagrama da camada de *network* que é agregado a uma estrutura da camada de enlace, que está contida em outra estrutura da camada física. É possível ter uma visão geral das camadas na figura 4.

O ESP32 também suporta comunicação via WLAN e Wi-Fi. O modo de conexão WLAN cria uma rede local sem fio entre dispositivos. Da mesma forma que a LAN (*Local Area Network*) os dispositivos são adicionados usando o protocolo DHCP (*Dynamic Host Configuration Protocol*), isto é, alocando endereços de IP dinâmicos clientes automaticamente. O Wi-Fi é um tipo de WLAN que segue o padrão IEEE 802.11. Esse padrão tem uma série de especificações feitas pelo Instituto de Engenheiros Elétricos e Eletrônicos (IEEE). Utilizando o Wi-Fi, o ESP32 habilita o protocolo Wi-Fi MAC automaticamente, ativando algumas funções como infraestrutura simultânea para BBS (*Basic Service Set*) Station mode, ou SoftAP (*Software Enabled Access Point*) mode, ou

Promiscuos mode, também apresenta modos de proteção como RTS (*Request to Send*), CTS (*Clear to Send*) ou bloqueio imediato ACK (*Acknowledgement*).

Figura 4 - Esquema de encapsulamento entre as camadas de rede.

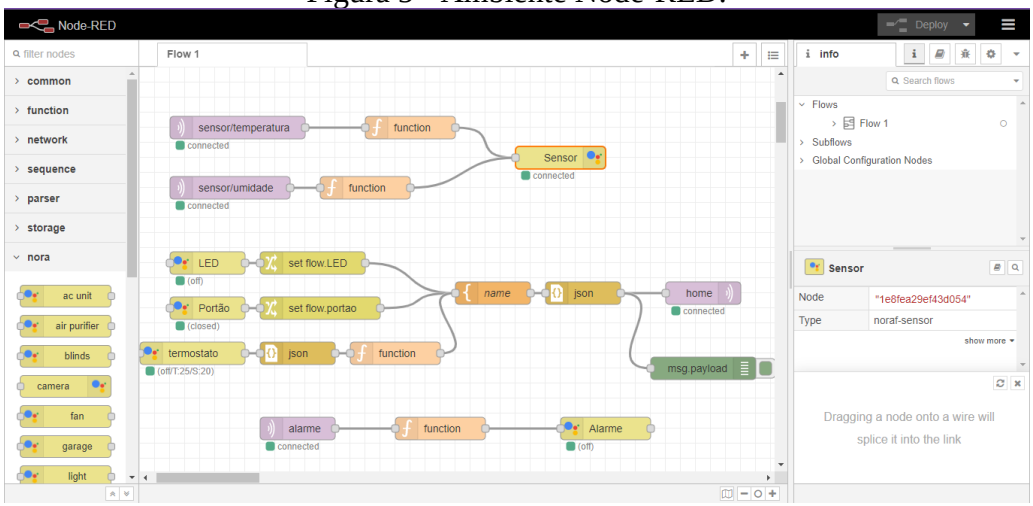


Fonte: tucones.blogspot[3]

3.2.5 Node-RED

Node-RED é uma ferramenta de programação que serve para integrar dispositivos, APIs e outros serviços online. Através de nós configuráveis, é possível comunicar e gerenciar ações e estados. Os nós podem ser desenvolvidos separadamente e instalados no ambiente, assim como os já configurados podem ser exportados e importados posteriormente em outros *flows*. Seu ambiente funciona via *browser*, podendo estar disponível localmente ou em algum outro provedor, disponível via IP.

Figura 5 - Ambiente Node-RED.



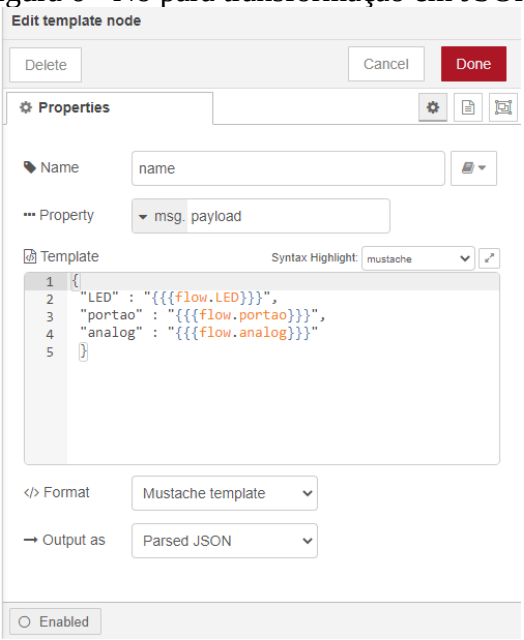
Fonte: Autor

[3] Disponível em: https://tucones.blogspot.com/2010/11/redenetwork-explicacao-resumida-camada.html

Os fluxos dentro do ambiente Node-RED são editáveis e existem alguns nós padrão para facilitar a integração entre as diferentes paletas de nós diferentes. Por exemplo, existe um nó chamado função e nele há a possibilidade de manipular dados usando JavaScript, ou ainda, um nó que coloca os dados no formato JSON. O *runtime* é realizado usando Node.js. No site da aplicação há uma quantidade grande de informações e discussões que facilitam muito o seu uso (NODE-RED, 2021).

Para esse trabalho, foi instalada a biblioteca *node-red-contrib-smartnora* na versão 1.5.1. Um ponto de destaque é o nó name (Fig. 6), este nó foi utilizado para agrupar os dados recebidos do Google Home e estruturá-los em um JSON, antes de enviar para o ESP32.

Figura 6 - Nó para transformação em JSON.



Fonte: Autor

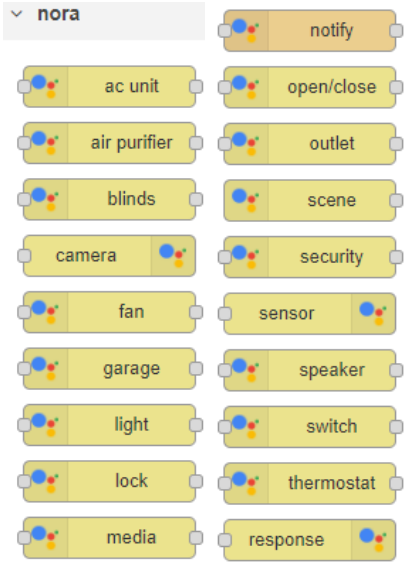
3.2.6 Smart NORA

O Smart NORA é um *package* desenvolvido para integrar o aplicativo Google Home com outros serviços no Node-RED, sendo que a sigla NORA vem de *NOde-Red home Automation*. O serviço Smart NORA foi baseado no serviço NORA que já havia feito muito sucesso. No entanto, a nova versão foi reescrita a fim de possibilitar a comunicação com a Google Cloud, além de adicionar novos nós que possibilitam execução em rede local e envio de notificações diretamente do Node-RED para os dispositivos (TATAR, 2021). É possível acessar todo o código do serviço entrando na página do GitHub do desenvolvedor Andrei Tatar.

O serviço foi desenvolvido usando as linguagens HTML, TypeScript e JavaScript e por ser um código aberto está sempre recebendo atualizações e contribuições de usuários da plataforma GitHub. A aplicação foi desenvolvida usando a plataforma Firebase da Google, o que pode ser observado analisando o código disponível no GitHub do autor.

O serviço Smart NORA oferece 21 nós diferentes para realizar ações. Os nós estão conectados com *actions* da plataforma Google Smart Home que fornece suporte para conexão de devices através do Google Home e do Google Assistant. Por esse motivo, é possível gerenciar os estados dos nós utilizando comandos de voz, via Google Assistant. Os nós disponíveis são mostrados na figura 7. Na plataforma Smart Home, há exemplos de como seriam as requisições para sincronização, *query* e execução de cada *action*.

Figura 7 - Nós Smart NORA no Node-RED.



Fonte: Autor

Antes de tratar de cada nó em especial, é importante ressaltar que existem informações que são comuns a todos eles. Ao configurar qualquer nó, sempre haverá um campo para indicar o nome do dispositivo, que aparecerá no Google Home. Por exemplo, um nome de dispositivo poderia ser: ar-condicionado ou AC ou Ar. O nome dado será o utilizado para acionar o comando por voz posteriormente. Também faz parte das informações comuns aos nós, a opção de enviar todas as mensagens que chegam para a saída, sem qualquer filtro. Além disto, a opção de ignorar todas as mensagens que não sejam de determinado tópico. É possível determinar o ambiente que o dispositivo faz parte, por exemplo, sala ou quarto. A opção *Two Factor* dá a opção de usar uma verificação para execução da ação. Essa

verificação pode não existir, pode ser um pedido de confirmação ou uma senha. Também pode-se configurar um tópico para aquele dispositivo e definir se o comando é realizado de forma assíncrona ou não.

O nó *ac unit* é um nó que representa um dispositivo de ar-condicionado, setado como *action.devices.types.AC_UNIT* na Google Smart Home. É possível configurar neste nó a unidade de temperatura, Celsius ou Fahrenheit, o intervalo de temperatura, os modos de temperatura (*cold, on, off, fan* etc), qual o intervalo mínimo entre o mínimo e o máximo valor no modo *heatcool*, qual a velocidade da ventilação em porcentagem, qual a linguagem para mudar as configurações do dispositivo, quais velocidades deseja pré configurar (nome e valor). A entrada e a saída terão um modelo que conterá o modo utilizado, sendo o *off* o padrão, um valor inicial de temperatura, um valor máximo e mínimo de temperatura caso o modo seja *heatcool*, a temperatura, umidade e velocidade correntes e se o dispositivo está *online*.

O nó *blinds* representa as cortinas de uma casa. Sua *action* pode ser referenciada como *action.devices.types.BLINDS*. Por padrão, 100% na opção *openPercent* representa uma cortina totalmente aberta e 0, a cortina totalmente fechada. No entanto, há uma opção que permite inverter os valores, sendo 100 totalmente fechada e 0 totalmente aberta. Além destas configurações, há um campo que permite verificar se o dispositivo está online ou não. Como este nó trabalha com porcentagem, haveria a possibilidade de reutilizá-lo para outros casos, como por exemplo a abertura de uma válvula ou o aquecimento de uma piscina. Seria o caso de uma saída analógica PWM.

O nó *camera* habilita a câmera existente no *device*, neste caso, no *smartphone*. Em suas configurações, é possível habilitar ou não a necessidade de autenticação para seu uso. Se o usuário habilitar esta opção, deverá adicionar um *token*. Pode-se definir se as opções de streaming via HTTP, com alta resolução ou não, em tempo real ou não, serão habilitadas. Exceto a transmissão em tempo real com protocolo WebRTC (*Web Real-Time Communication*), os demais serão configurados, incluindo uma URL e um ID. A URL será o *endpoints* que receberá a imagem e o ID será usado para processar o fluxo de câmera quando um Chromecast for utilizado. No caso do WebRTC deverá ser configurado uma URL para receber e trocar informações via o protocolo SDPs. Haverá um campo para a oferta do protocolo SDP, outro para especificar o servidor ICE (*Interactive Connectivity Establishment*) e um campo para verificar se a câmera está online.

O nó *fan* representa um ventilador. Nele poderá ser realizado o controle da velocidade, utilizando percentual ou através de modos. No caso do uso de modos, a linguagem deverá ser configurada, bem como o modo (velocidade), que inclui um nome e um valor. O nome do modo será a referência do Google Assistant para configurar determinada velocidade. Como entrada e saída, além da velocidade em porcentagem ou modo, também haverá a informação de que o ventilador se encontra ligado ou desligado. Este é um exemplo em que poderiam reunir saídas digitais e analógicas em apenas um nó.

O nó *garage* representa a abertura e fechamento de uma porta de garagem. Nele é possível gerar um aviso através do Google Assistant de que a porta já abriu ou já fechou, ou enviar uma mensagem informando seu estado. Como entrada e saída haverá a informação sobre o quanto a porta está aberta em porcentagem, sendo que 0 aparecerá quando ela estiver completamente fechada. Também é possível verificar se o nó está *online* ou não. Para este nó, a abertura da porta exige a configuração de uma senha.

O nó *light* se refere a uma iluminação em geral. Neste nó é possível não só configurar se a iluminação estará acesa ou apagada mediante um comando de voz, mas também conforme a intensidade ou cor no ambiente mudem. O usuário também poderá ser notificado caso não haja mudança acionando o comando de voz. Caso habilitado o controle de luminosidade, a escala será dada em porcentagem, sendo 0 sem nenhuma luminosidade. O *status* da iluminação poderá ser enviado ou recebido como conteúdo de entrada e saída. O nó oferece também suporte para 3 modos de cor, a saber, RGB (*Red Green and Blue*), HSV (*Hue, Saturation and Value*) e temperatura.

O nó *lock* foi pensado para o trancamento de uma porta de garagem. Também oferece a opção de alertar o usuário sobre o *status* da ação, caso o comando não seja realizado via comando de voz. E como entrada e saída, tem as informações, trancado ou destrancado, emperrado ou desemperrado e *online* ou *offline*. Neste nó, a inclusão de uma senha é obrigatória.

O nó *media* inclui a configuração de uma lista de possíveis dispositivos, dentre eles, controle remoto, TV, alto-falantes e receptores de áudio e vídeo em geral. A primeira configuração diz respeito ao tipo de dispositivo a ser configurado. A própria Google oferece uma lista de dispositivos possíveis. A configuração *On/Off Trait* verifica se o dispositivo está ou não disponível para configuração. Também é possível configurar o volume, verificar se suporta ou não determinada mídia, direcionar uma ação como *play*, *pause*, próximo. É

possível determinar em qual linguagem os comandos serão recebidos, quais entradas serão aceitas para cada ação, seleção de aplicativo ou canal.

O nó *notitfy* pode ser utilizado para enviar notificações de *web* a respeito dos dispositivos. Para que esse serviço funcione, é preciso habilitar o envio de notificações no *site* do Smart NORA. No entanto, nem todos os dispositivos e navegadores suportam notificações *web*. Na configuração deste nó, é possível atribuir uma etiqueta, um título para a notificação, uma mensagem e um ícone. A realização de uma ação após a notificação é opcional, mas pode ser configurada com botões. Para a configuração dos botões, é indicada uma descrição e um valor para cada um.

O nó *open/close* pode ser utilizado para diversos dispositivos. Dentre eles, pode-se citar, um closet, uma porta, uma janela, uma válvula, um portão, entre outros. Esse nó suporta a configuração de múltiplas direções de fechamento ou abertura, além dos dois estados. É possível notificar o usuário sobre o *status* do dispositivo, é possível habilitar o trancamento ou destrancamento e verificação de emperramento. Para este nó, a configuração de uma senha é também obrigatória.

O nó *outlet* é um nó com propriedades binárias. Representa uma tomada ou uma saída qualquer. Além da entrada e saída ligada ou desligada, é possível verificar se o dispositivo está *online* e notificar o usuário sobre o seu estado.

O nó *scene* representa a configuração única de um conjunto de dispositivos de um dado espaço. Essa ação pode ser configurada como reversível, ativa ou desativada.

O nó *security* representa um sistema de segurança que pode estar ativado ou não. É possível que o sistema inclua múltiplos níveis de segurança e, sendo assim, é possível verificar qual é o nível vigente. Uma outra configuração que pode ser testada é o tempo em segundos até que o nível de segurança comece a funcionar.

O nó *sensor* representa um sensor de temperatura e umidade. A temperatura pode ser dada em Celsius ou em Fahrenheit e umidade é dada em porcentagem. Esse é um exemplo de entrada analógica.

O nó *switch* representa um interruptor. É um nó simples que pode ser utilizado em múltiplas funções. Trata-se de uma saída digital, sendo representada em valores booleanos. Assim como outros nós, é possível verificar se o interruptor está *online* e notificar o usuário de seu estado.

Por fim, o nó *thermostat* é um nó que representa um aquecedor ou um ar-condicionado. Ele pode ser configurado em modos. É possível determinar a temperatura que será setada

com o seu ligamento, o intervalo de temperatura possível, sua unidade (°C ou F), a umidade em porcentagem e se o dispositivo será alterado com comandos ou somente através de requisições do próprio sensor. É possível, por exemplo, configurar um cenário onde o termostato será ligado no modo resfriamento, quando o sensor indicar uma temperatura de 30°C ou no modo de aquecimento, quando o sensor indicar uma temperatura de 4°C.

3.2.7 Google Home App

O aplicativo Google Home foi desenvolvido com o propósito de integrar os serviços Google, utilizando o Google Assistant, com dispositivos inteligentes presentes na residência do usuário. Ele pode ser configurado no Google Nest, em Chromecast ou *smartphones*. Cada funcionalidade conta com um conjunto de *actions* que podem ser setadas pelo usuário utilizando apenas a voz. Com ele é possível controlar a luminosidade, o volume de uma TV, as câmeras da casa, a velocidade e temperatura do ar-condicionado etc.

A interface do aplicativo permite que ambientes sejam configurados e dispositivos sejam vinculados a esses ambientes. Por exemplo, há como configurar uma TV da sala e outra para o quarto, ou mesmo identificar casas diferentes. Também é possível adicionar membros que tenham acesso às configurações da casa.

O fato dessas configurações serem hospedadas na nuvem faz com que o usuário possa acessá-las de qualquer lugar. Além disso, é possível vincular uma conta Google que irá integrar todos os serviços em um só lugar.

Existem algumas limitações que podem ser encontradas nos recursos oferecidos pelo aplicativo. Algumas configurações podem não apresentar uma visualização direta no aplicativo, como por exemplo, a leitura de temperatura e umidade corrente no projeto desenvolvido. Essas informações podem ser só acessadas via Google Assistant. Mas é claro que com todos os recursos existentes na nuvem é possível realizar grandes projetos.

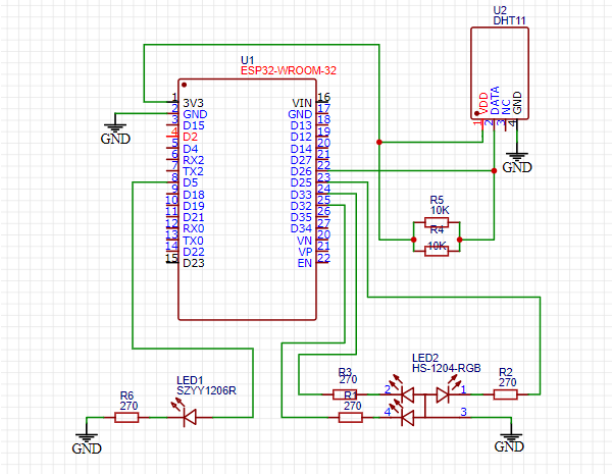
4. DESENVOLVIMENTO

4.1 Hardware

Para explorar as possibilidades de controle de dispositivos residenciais pelo ESP32, foram simuladas entradas e saídas digitais e analógicas. Como entrada digital foi utilizado o acionamento do botão *boot* do microcontrolador. Esse botão simula um evento qualquer que dispara um alarme. Esse alarme só pode ser desativado pelo usuário via Google Home. Outra entrada digital é a iluminação, só que neste caso o acionamento pode ser feito via comando de voz ou apertando o botão do dispositivo no aplicativo. Tanto o botão *boot* como o LED que simula a entrada digital são integrados no microcontrolador. Para acessar o *boot* no código desenvolvido no Arduino IDE, deve-se acionar o pino 1 e o LED no pino 2.

O pino D5 está ligado a um LED comum. Esse LED representa a abertura de um portão, representando uma saída digital. A diferença é que o acionamento de abertura só é possível com *input* de uma senha. Os pinos D25, D32 e D33 foram utilizados para simular a saída de dados analógicos. A saída é obtida mediante o ajuste da temperatura de um termostato. Dependendo da temperatura escolhida, obtém-se uma cor diferente no LED RGB, decorrente de uma configuração combinada nos pinos mencionados. O pino D26 foi utilizado para a entrada de dados analógicos, que variam conforme a leitura de um sensor de temperatura e umidade. O circuito foi representado na figura 8. A configuração utilizada foi escolhida conforme as necessidades do circuito e o material disponível para sua construção.

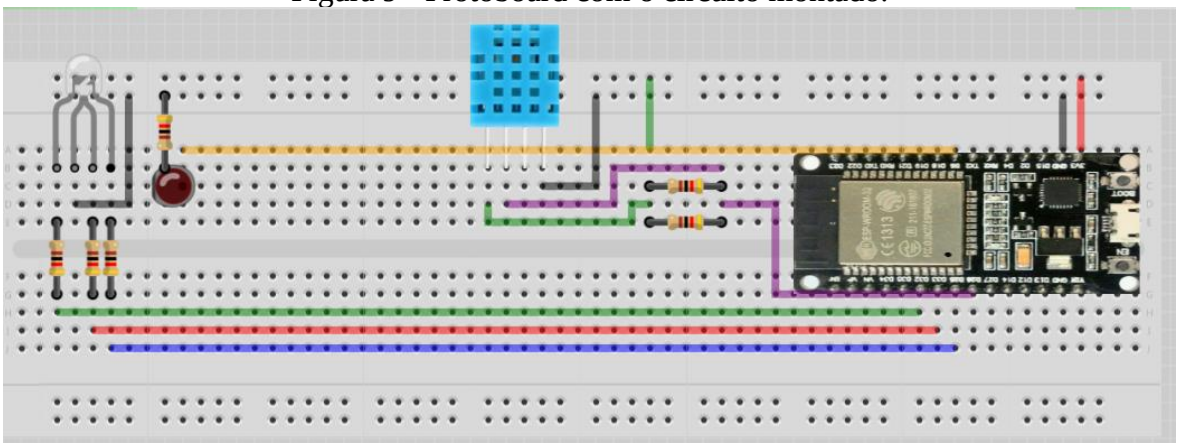
Figura 8 - Circuito montado para construção do sistema.



Fonte: Autor

No *protoboard* o circuito pode ser representado pela figura 9.

Figura 9 - Protoboard com o circuito montado.

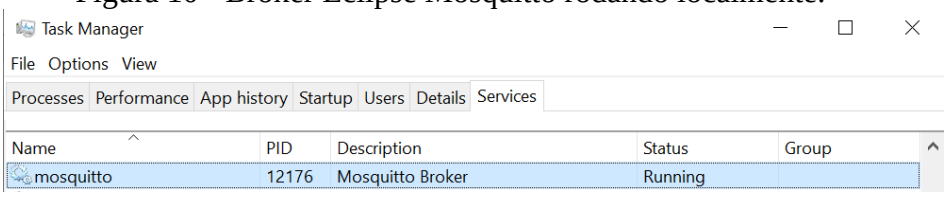


Fonte: Autor

4.2 Comunicação entre ESP32 e a Google Cloud

Tendo configurado o circuito, é preciso realizar a comunicação do microcontrolador com a nuvem. Para realizar essa comunicação foi primeiramente estabelecida uma comunicação MQTT com um servidor rodando localmente. Utilizando as bibliotecas *ArduinoJson.h, WiFi.h, PubSubClient.h* e a *SimpleDHT.h*, já mencionadas acima, foi criado um código com a finalidade de enviar e receber dados utilizando funções que permitem configurar a inscrição (*subscribe*) e publicação (*publish*) em tópicos. Foram utilizados o *broker* Eclipse Mosquitto e a extensão do navegador Chrome MQTT Lens. Através dessas duas ferramentas, foi possível identificar exatamente como e se a comunicação estava acontecendo. Após instalado o *broker* Eclipse Mosquitto, foi verificado no gerenciador de tarefas, na aba serviços, se ele estava rodando.

Figura 10 - Broker Eclipse Mosquitto rodando localmente.

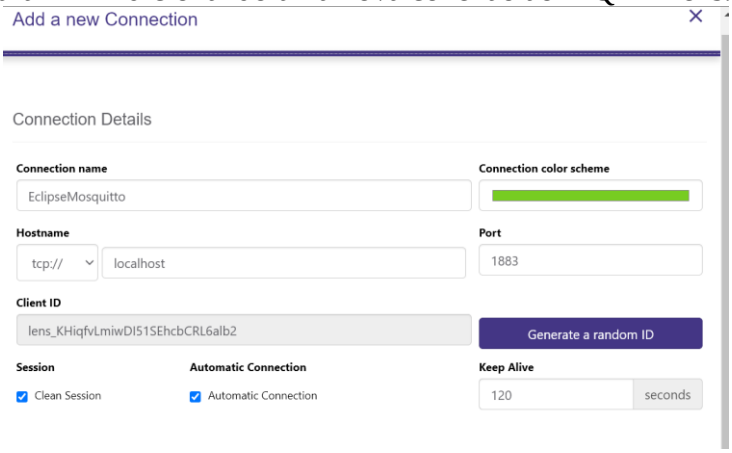


Fonte: Autor

Com o *broker* local rodando (Fig. 10), é possível configurar a extensão MQTT Lens para mostrar todas as mensagens recebidas no tópico assinado e publicar mensagens em algum tópico escolhido. Como o *broker* neste momento de teste está ativo localmente, o

hostname a ser configurado será ‘localhost’ ou ‘127.0.0.1’. A figura 11 mostra a configuração citada sendo realizada. Também é possível restringir o acesso às mensagens configurando um usuário e uma senha.

Figura 11 - Adicionando uma nova conexão ao MQTT Lens.



Fonte: Autor

Após criar uma configuração para o *hostname* local e verificar que ela está rodando, aparecerá uma tela parecida com a mostrada na figura 12. O campo *Subscribe* deverá ser preenchido com os tópicos dos quais é desejado receber mensagens e o campo *Publish* com o tópico no qual é desejado enviar uma mensagem, a ser escrita no campo *Message*.

Figura 12 - Verificando os dados recebidos no tópico sensor/temperatura.

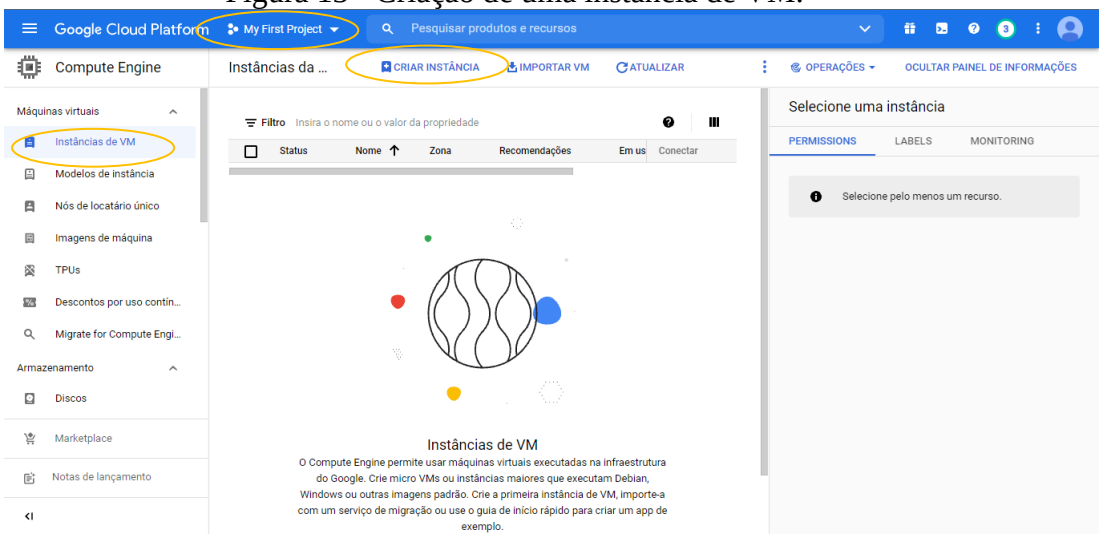
Fonte: Autor

Ao verificar que a comunicação via MQTT está funcionando de forma correta, foi o momento de configurar um ambiente em nuvem para que a comunicação não ficasse restrita à rede local. Foram pensadas diversas formas de viabilizar essa comunicação em nuvem. Por

exemplo, a princípio foi cogitada a possibilidade de utilizar a própria API Google Cloud Pub/Sub em conjunto com o serviço Firebase Realtime Database. No entanto, a solução envolveria o uso de diversas APIs integradas e a necessidade de criação de um aplicativo para integração com o usuário final. Além do custo que esta solução envolveria, foi evidenciado que ela não seria viável dado o tempo de desenvolvimento do presente projeto e o tempo disponível para realização de tal atividade em possíveis laboratórios da disciplina de Redes Industriais e de Comunicações. Aproveitando o conhecimento já abordado com os alunos, optou-se por utilizar o serviço Node-RED. A fim de continuar a utilizar recursos da do provedor, foi criada uma instância de máquina virtual utilizando uma imagem de *container* do Node-RED. Assim, a configuração dos dispositivos está acessível em nuvem em um IP temporário fornecido pelo provedor. Este endereço de IP também poderia ser fixado, caso existisse a necessidade.

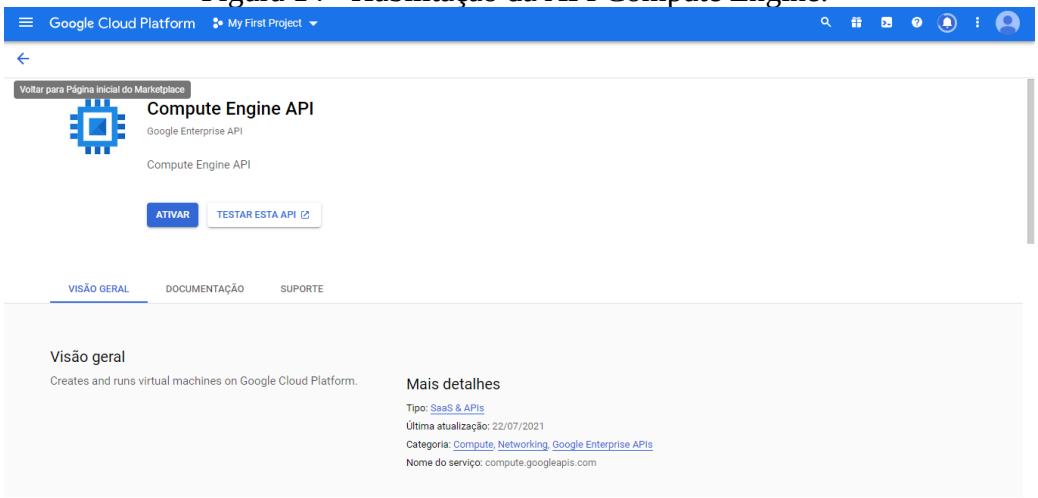
Para rodar remotamente a imagem do *container*, foi preciso habilitar a API Compute Engine (Fig. 14). Com a API Compute Engine habilitada, foi possível a criação de uma nova instância de VM no projeto inicial chamado *My First Project* como mostra na figura 13.

Figura 13 - Criação de uma instância de VM.



Fonte: Autor

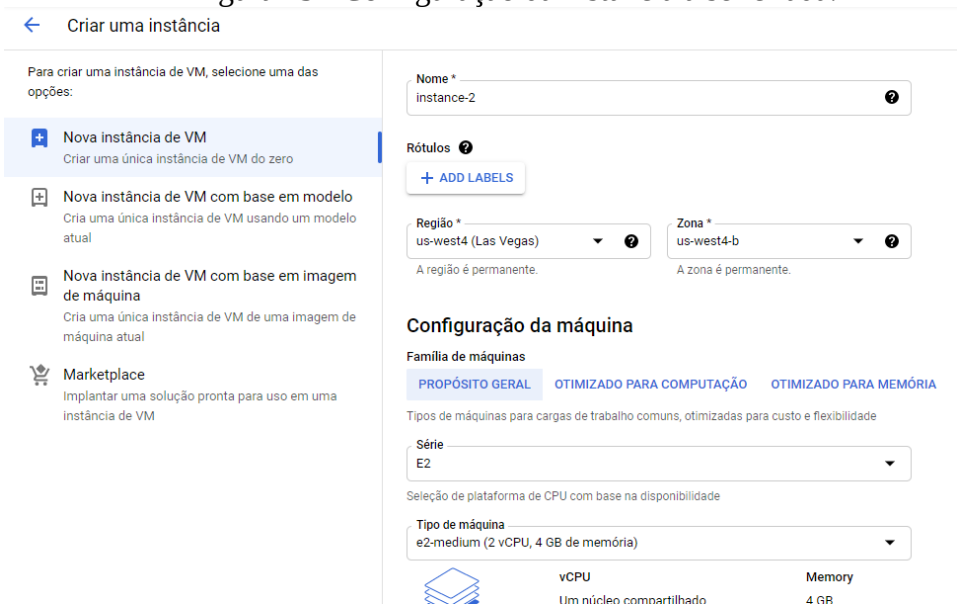
Figura 14 - Habilitação da API Compute Engine.



Fonte: Autor

Na criação da instância é possível selecionar a região e zona onde ela deverá estar hospedada, bem como sua configuração, seu poder de processamento e armazenamento, seu sistema operacional e outros detalhes (Fig. 15). Cabe ressaltar neste momento que todos os detalhes podem alterar o custo do serviço. Para o propósito deste trabalho, as configurações padrão não foram modificadas.

Figura 15 - Configuração da instância a ser criada.

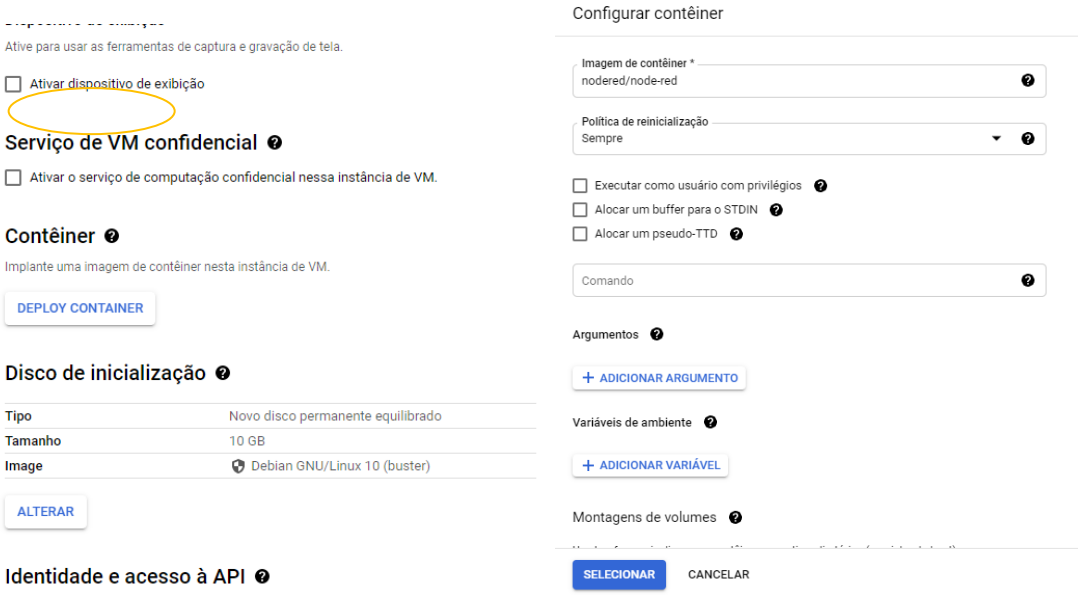


Fonte: Autor

Somente as opções de *container* serão personalizadas nesse caso. O repositório Docker Hub oferece inúmeras imagens de *containers*, incluindo a do Node-RED. Com essa imagem,

é possível criar um ambiente exclusivo e isolado para execução do serviço, o que o torna mais rápido e mais seguro também. A figura 16 mostra a configuração do container na Google Cloud.

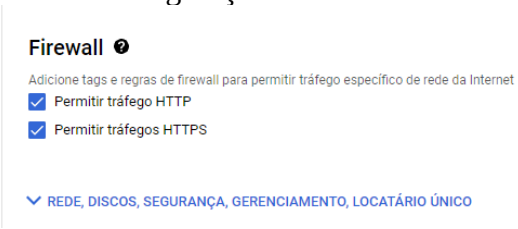
Figura 16 - Configuração da imagem de container.



Fonte: Autor

Na configuração do *container*, é preciso incluir a imagem do *container nodered/node-red*. É importante não esquecer de selecionar as duas opções de Firewall, para permitir o tráfego tanto HTTP, quanto HTTPS.

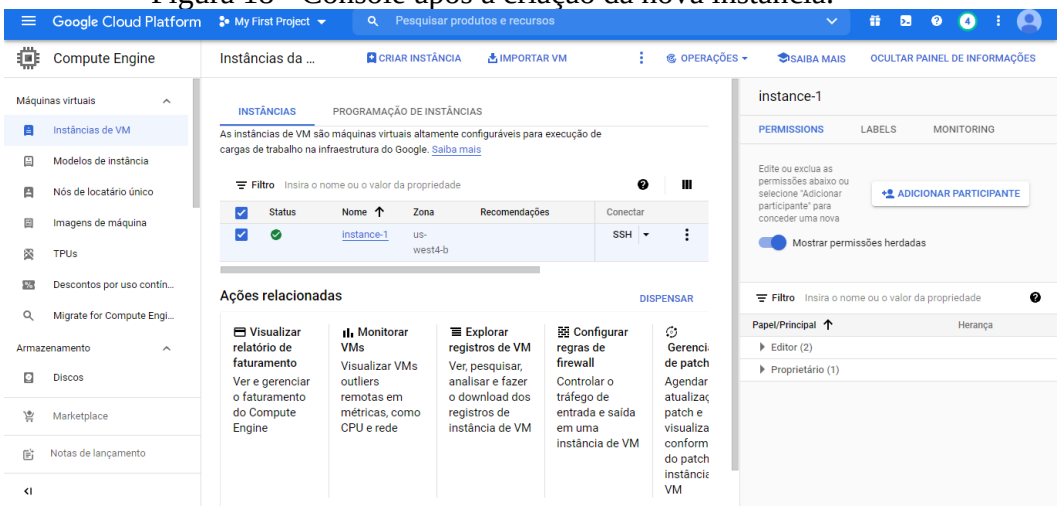
Figura 17 - Configuração do Firewall no container.



Fonte: Autor

Após todas as configurações já mencionadas, será possível visualizar a instância criada na página da API Compute Engine, tal como mostrado na figura 18.

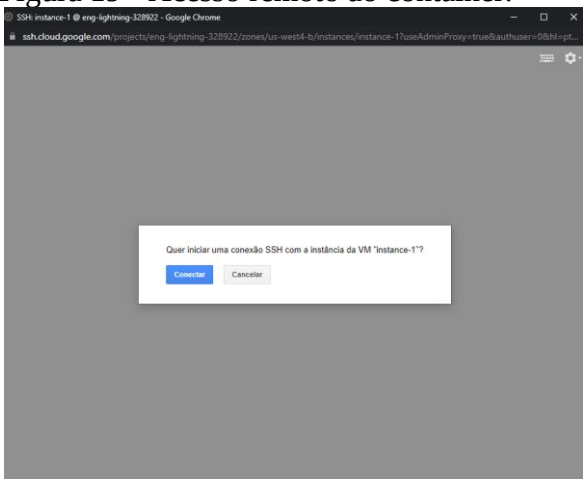
Figura 18 - Console após a criação da nova instância.



Fonte: Autor

Selecionando a instância criada, é possível visualizar todos os seus detalhes, tanto de sistema operacional, quanto de permissões. É possível obter informações a respeito dos IPs tanto interno primário, quanto externo. O IP externo é temporário e pode mudar cada vez que a instância for reiniciada. Para poder configurar a instância, deve-se acessar o ambiente criado. Como esse acesso é remoto, é preciso inicializar uma conexão SSH e escolher a opção *Conectar* (Fig. 19). A sigla SSH é a abreviação de *Secure Shell*, que é um protocolo de rede para estabelecer acesso seguro a um local remoto.

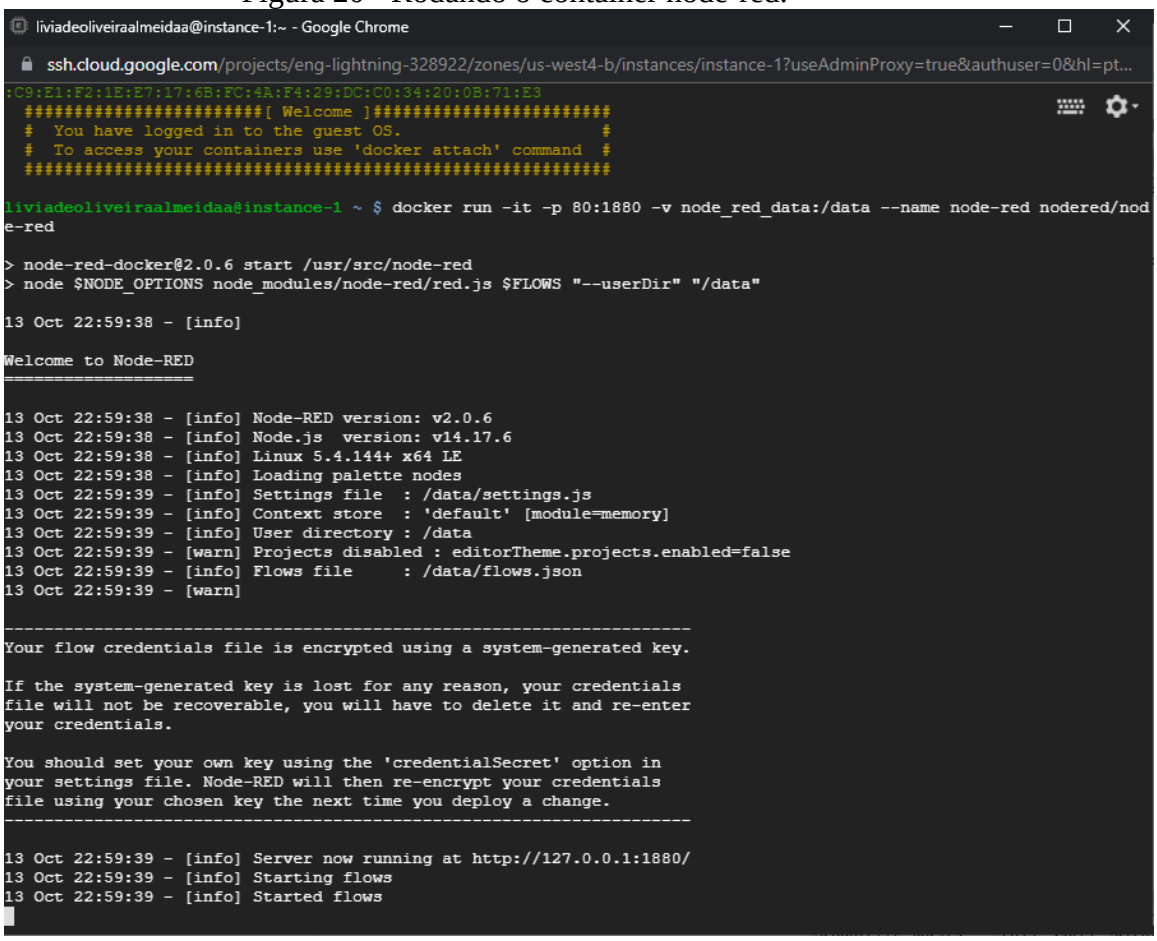
Figura 19 - Acesso remoto ao container.



Fonte: Autor

Já dentro da instância, foi digitado o comando: *docker run -it -p 80:1880 -v node_red_data:/data --name node-red nodered/node-red* (Fig. 20). Este comando inicia o *container* na porta 1880 e armazena seus dados na pasta chamada *node-red*. Quando o *container* precisa ser reiniciado, basta digitar os comandos *docker start <name>* e *docker stop <name>*. Neste caso, foi dado o nome node-red.

Figura 20 - Rodando o container node-red.

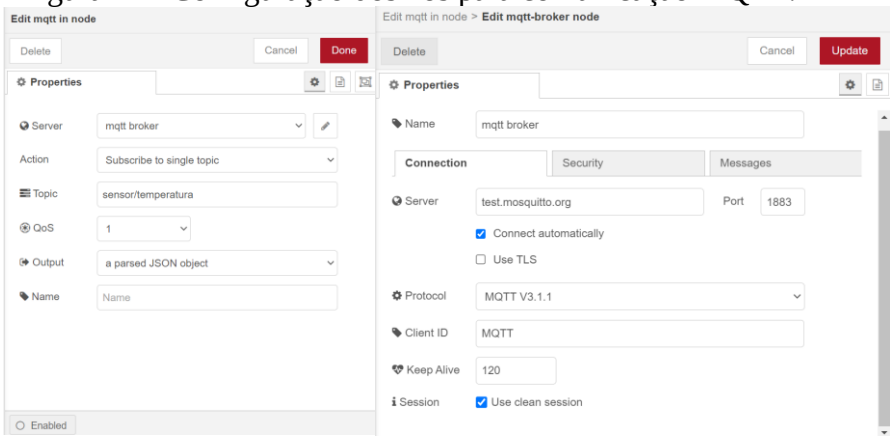


Fonte: Autor

Após a execução do comando, foi possível entrar em um ambiente Node-RED remoto, usando o IP externo. Por exemplo, usando http://34.125.208.3.

Tendo acesso remoto ao serviço Node-RED, foi preciso configurar a comunicação entre o microcontrolador e o ambiente. Para estabelecer esta comunicação, foram utilizados os nós *mqtt in* (Fig. 21) e *mqtt out*. A utilização desses nós é possível ajustando as informações do *broker* selecionado e o protocolo desejado. A aba de segurança é onde as informações de usuário e senha devem ser colocadas, caso seja necessário adotar parâmetros de segurança mais rígidos.

Figura 21 - Configuração dos nós para comunicação MQTT.

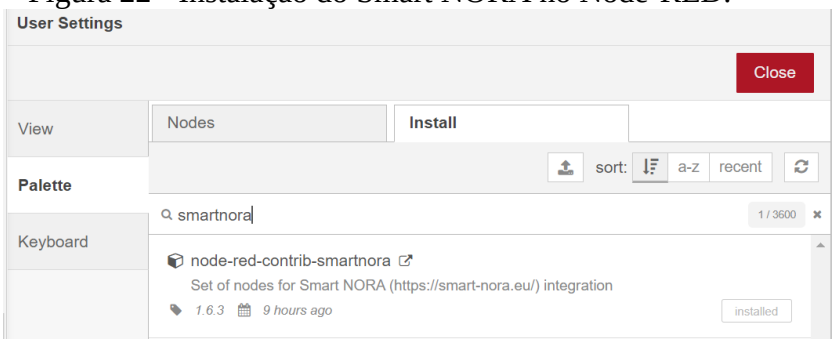


Fonte: Autor

4.3 Comunicação entre o aplicativo Smart Home e o Node-RED

Tendo estabelecido a comunicação entre o microcontrolador ESP32 e o ambiente Node-RED remoto, foi preciso configurar a comunicação entre o ambiente e o usuário final. Para esta comunicação foi utilizado o serviço Smart NORA. Para utilizar o serviço, o usuário deve realizar seu cadastro no site https://smart-nora.eu/ utilizando um e-mail e uma senha e posteriormente realizar a verificação do e-mail. A utilização deste serviço possibilitou a implementação mais rápida da integração entre o ambiente Node-RED, o aplicativo Google Home e o assistente virtual Google Assistant. Após realizado o cadastro do usuário no site do serviço, só foi necessário instalar o pacote *node-red-contrib-smartnora* no Node-RED (Fig. 22).

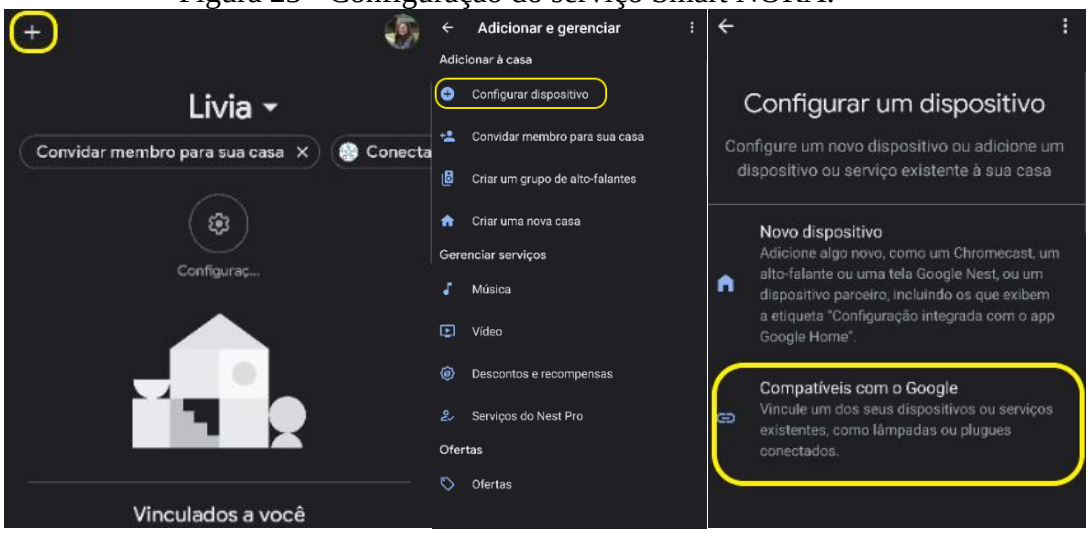
Figura 22 - Instalação do Smart NORA no Node-RED.



Fonte: Autor

Com o pacote do Smart NORA instalado, os nós incluídos no ambiente, são adicionados no aplicativo com o mesmo usuário cadastrado. Para que essa sincronização ocorresse de forma automática, foi preciso acessar a configuração de dispositivo no aplicativo, como mostra a figura 23. Seguindo as telas mostradas na figura mencionada, foi possível encontrar o serviço Smart NORA a ser configurado.

Figura 23 - Configuração do serviço Smart NORA.



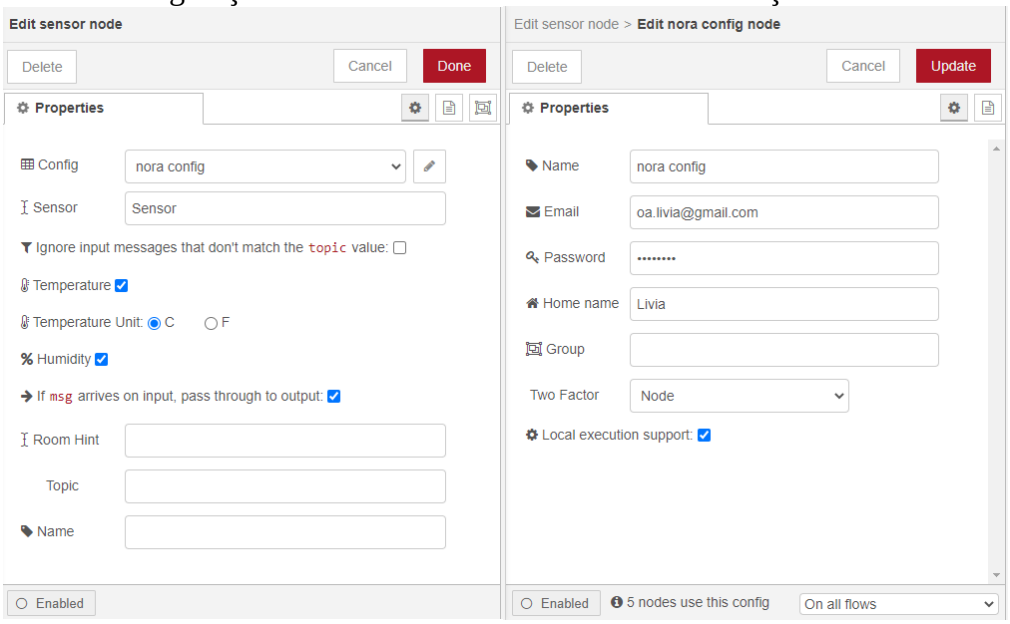
Fonte: Autor

Vale enfatizar que a sincronização entre o ambiente Node-RED e o aplicativo depende da configuração do *login* com o mesmo e-mail e senha, anteriormente, cadastrados pelo site, e aceitação dos termos e condições do serviço.

Para visualização dos dispositivos no aplicativo, é preciso realizar *Deploy* do ambiente. Para este trabalho foram utilizados os nós: sensor, iluminação, garagem e termostato. Nota-se que o nó de iluminação foi utilizado duas vezes para propósitos diferentes, o que é totalmente possível, visto que o nome do nó que aparece no Google Home é editável.

Na figura 24 é possível ver a configuração utilizada no nó do sensor, por exemplo. Visto que o sensor utilizado é capaz de verificar tanto a temperatura, quanto a umidade, as duas opções foram selecionadas na configuração. No campo *Config* o usuário deve entrar com as suas credenciais do serviço Smart NORA, para que as configurações sejam carregadas no aplicativo Google Home. Para a configuração do serviço Smart NORA, basta fornecer o e-mail do cadastro e senha. O nome dado à casa é opcional, bem como a configuração de autenticação de dois fatores.

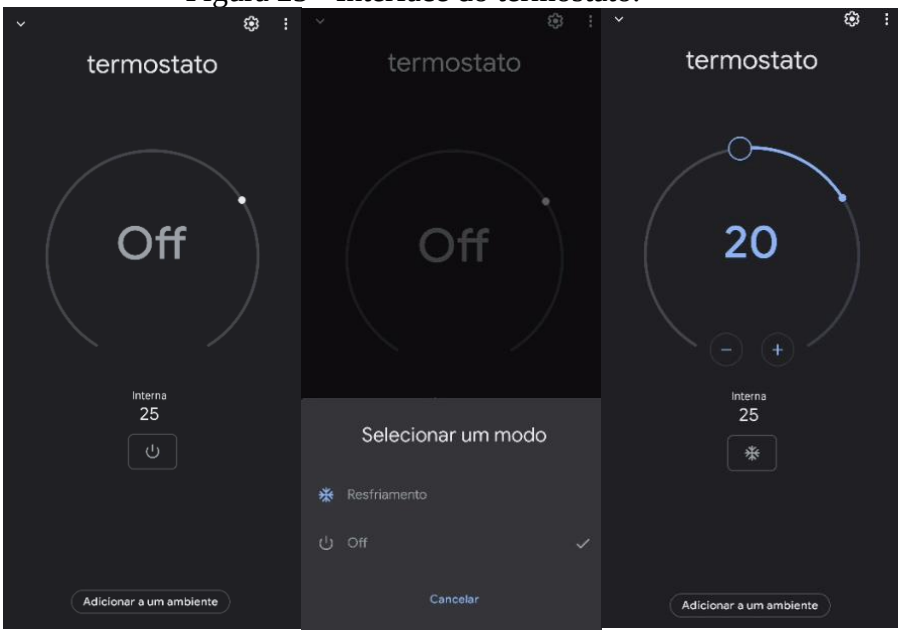
Figura 24 - Configuração do nó do sensor e do cadastro no serviço Smart NORA.



Fonte: Autor

É possível observar que no caso do Sensor e do Portão, aparece uma engrenagem no canto superior direito. Para estes dispositivos, não há uma interface de configuração. No caso do Sensor, as informações de temperatura e umidade podem ser consultadas usando o Google Assistant por comando de voz. O portão abre ou fecha, utilizando comando de voz também. Para os demais dispositivos há uma interface de configuração ao selecioná-los. As imagens 25 e 26 mostram as configurações possíveis para cada dispositivo. Para o termostato, ao selecionar o botão de ligar ou solicitar a ativação do termostato, será requisitado um modo, sendo que há a possibilidade de configurar um dos modos para ser o padrão.

Figura 25 - Interface do termostato.



Fonte: Autor

A interface do Alarme e do LED são muito semelhantes, visto que eles foram configurados com o mesmo tipo de nó.

Figura 26 - Interface dos nós Alarme e LED.

Fonte: Autor

4.4 Fluxo no ambiente Node-RED

Após estabelecer a comunicação do ESP32 com o ambiente Node-RED e a comunicação do aplicativo Smart Home com o mesmo ambiente, foi preciso trabalhar com os dados recebidos e enviados. Neste momento foi importante entender quais tipos de dados deveriam ser recebidos pelos nós direcionados ao aplicativo Smart Home e como eles poderiam ser enviados constantemente e não apenas quando alterados. Para este problema, foram setadas variáveis globais no ambiente Node-RED, para que o estado anterior fosse replicado, caso não houvesse nenhuma mudança. Além disso, foi percebido que a inscrição de tópicos em separado para cada dispositivo causava confusão e atraso na atualização do estado de cada variável e, consequentemente, atraso nos comandos que deveriam ser enviados para os LEDs. A solução encontrada para esse problema foi a transformação dos dados separados em um único arquivo no formato JSON. Além de possibilitar uma melhor sincronização dos dados, o formato JSON é leve e simples. O nó utilizado para esta conversão foi mostrado na figura 6.

4.5 Resultados

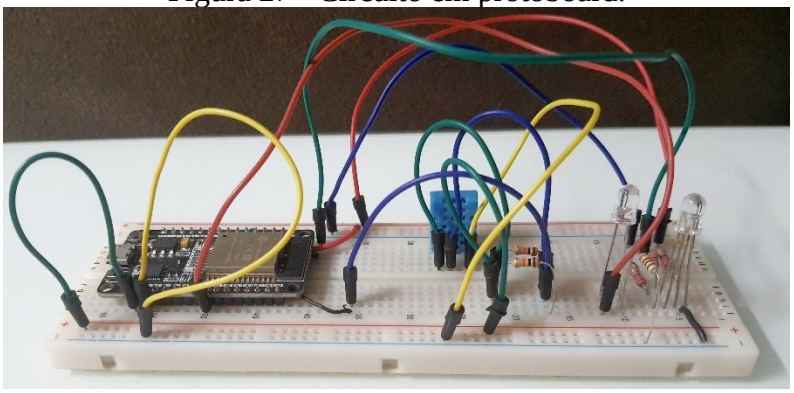
Com o término do projeto, foi possível obter um sistema capaz de controlar recursos conectados com o ESP32, utilizando um *smartphone* com o aplicativo Google Home e fazendo acionamentos via Google Assistant. Foi explorado exemplos com entradas e saídas tanto analógicas, quanto digitais. E, desta forma, garantindo que futuros usos possam ser facilitados. De certa forma o serviço Smart NORA gera alguma limitação, já que foi pensado para implementação de ações em ambientes residenciais. No entanto, é possível ter uma boa noção de como implementar um serviço que possa criar outras funcionalidades.

A partir do desenvolvimento e integração do serviço Smart NORA é possível integrar muitas outras funcionalidades presentes na plataforma Google Home, já que todos os dias os recursos são melhorados.

4.5.1 Validação do sistema

Com o sistema desenvolvido e funcionando, obteve-se o seguinte sistema, conforme mostrado nas figuras 27, 28 e 29. Na figura 27 é possível verificar o circuito montado em uma *protoboard*.

Figura 27 - Circuito em protoboard.



Fonte: Autor

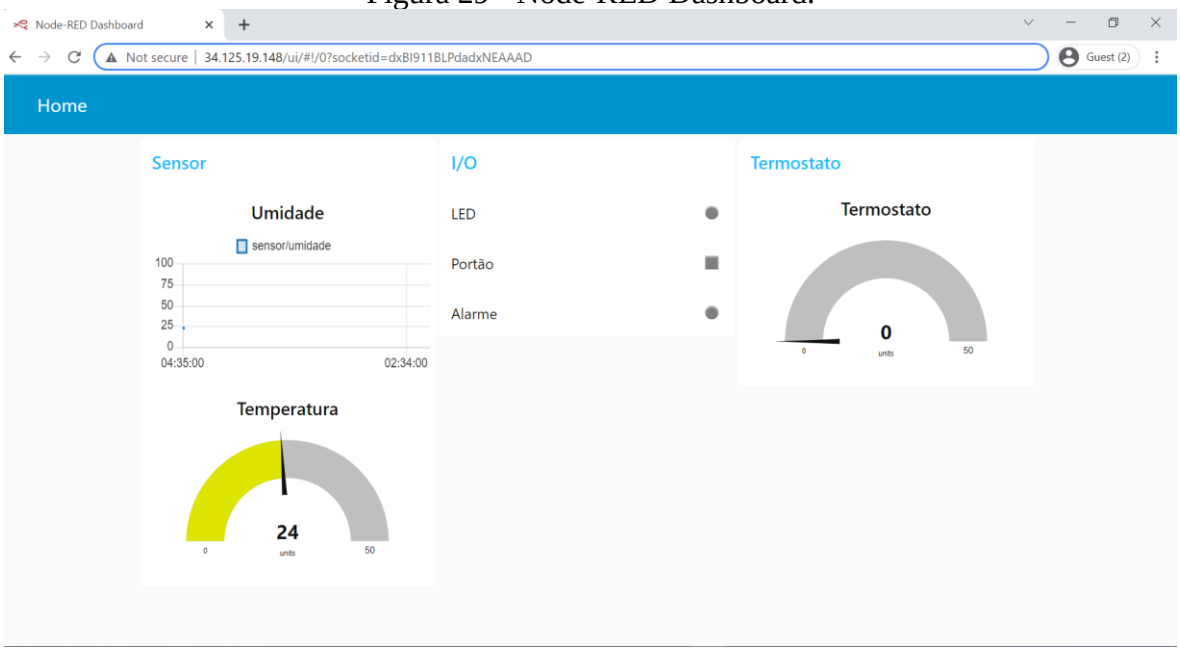
A figura 28 mostra a tela do aplicativo Google Home, após todos os nós estarem configurados.

Figura 28 - Tela no aplicativo Google Home.

Fonte: Autor

A figura 29 mostra um dashboard criado para melhor visualização das informações sendo alteradas tanto na *protoboard*, quando via comando de voz pelo aplicativo.

Figura 29 - Node-RED Dashboard.



Fonte: Autor

Por fim, foi gravado um vídeo onde todo o sistema e funções são mostradas com maiores detalhes. O vídeo pode ser acessado através do link https://www.youtube.com/watch?v=CZYxioutEGI. No vídeo é possível acompanhar todos os comandos sendo realizados e o comportamento da tela do aplicativo e da *protoboard*.

Assim como mostrado no vídeo citado, para o teste de funcionamento do sistema foram executadas as ações e obtidos seus respectivos resultados como mostra a tabela 1.

Tabela 1 - Resultados obtidos com o teste de funcionamento.

Comando	Ação	Retorno	Retorno correto?
"Ok Google, qual é a temperatura do sensor?"	Leitura dos dados publicados no tópico "sensor/temperatura", conforme dados lidos no sensor	"No momento a temperatura do sensor está definida em 25°C"	Sim

“Ok Google, qual é a umidade do sensor?”	Leitura dos dados publicados no tópico “sensor/umidade”, conforme dados lidos no sensor	“O sensor está marcando 24% de umidade”	Sim
1. **“Ok Google, abrir portão”** 2. **“120”** 3. **“Ok Google, fechar o portão”**	Escrita do *status* do portão no tópico “home” e envio dos sinais *high* e *low* ao LED.	1. “Você pode informar o código de segurança?” 2. “Claro, abrindo o portão” 3. “Tudo bem, fechando o portão”	Sim
1. **“Ok Google, ligar LED”** 2. **“Ok Google, desligar LED”**	Atualização do *status* do LED integrado no tópico “home” e envio de sinais *high* e *low* ao LED.	1. “Tudo bem, ligando o LED” 2. “Claro, desligando LED”	Sim
1. **“Ok Google, ligar termostato”** 2. **“Ok Google, alterar temperatura do termostato para 30°C”** 3. **“Ok Google, alterar temperatura para 19°C”**	Atualização do *status* e temperatura do termostato no tópico “home” e envio de dados para o LED RGB, com alteração de cor.	1. “Tudo bem, mudando o termostato para ligado” 2. “Tudo bem, ajustando o termostato para 30°C” 3. “Sem problemas, ajustando o termostato para 19°C”	Sim

4. **“Ok Google, desligar termostato”**		4. “Sem problemas, mudando o termostato para desligado”	
Simulação do disparo com acionamento do botão *boot*	Mudança do *status* do Alarme no tópico “home” e atualização no aplicativo	*Status* atualizado e com possibilidade de desativação.	Sim

Fonte: Autor

Conforme pode ser visualizado na tabela 1, os resultados esperados foram os mesmos que os obtidos nos testes realizados.

5. CONCLUSÃO

O objetivo desse projeto foi apresentar uma solução que oferecesse a possibilidade de controlar um ambiente utilizando serviço em nuvem e um microcontrolador de baixo custo. Foram investigadas diversas soluções já existentes e publicadas em artigos. Muitas das soluções exploradas tiveram que ser descartadas devido ao tempo, custo e complexidade necessários para seu desenvolvimento, levando em conta que a solução apresentada deveria poder ser replicada em uma disciplina de graduação. Por exemplo, havia a possibilidade de utilizar recursos disponibilizados pela Google Cloud para comunicação MQTT entre o microcontrolador e um ambiente em nuvem, utilizando as APIs Pub/Sub, IoT e Firebase. Esses recursos demandam uma integração não tão simples de ser realizada, sendo necessária a criação de um ambiente que pudesse se comunicar com o Google Home. No entanto, a solução adotada foi simples e barata, sendo que o custo seria apenas o de utilização da nuvem para hospedar a máquina virtual com o ambiente Node-RED. Além disso, o serviço Smart NORA garantiu uma rápida e eficiente comunicação entre o ambiente Node-RED e o aplicativo Google Home, com a vantagem de estar vinculado com as *actions*, que são constantemente atualizadas pelo provedor. Assim, conclui-se que é possível adotar uma solução de baixo custo para controle de ambiente remotos utilizando serviço em nuvem.

Entende-se que, a partir desse trabalho, seria possível desenvolver novos nós, com funcionalidades específicas para qualquer controle desejado, incluindo apenas novos elementos ao serviço Smart NORA, visto que o código fonte está inteiramente disponível para consulta no GitHub. Além disso, para futuros trabalhos, seria interessante explorar soluções inteiramente sem custo, hospedando a máquina virtual em um provedor que ofereça essa possibilidade. Seria interessante também, pensar nas possibilidades de comunicação entre os sensores e dispositivos com o ESP32, já que no presente trabalho esta conexão é realizada de forma direta. E, por fim, poderia ser relevante um trabalho com mais detalhes a respeito da interação do Google Assistant e o Google Actions, podendo explorar formas personalizadas de integração entre os serviços.

6. REFERÊNCIAS BIBLIOGRÁFICAS

PUJARIA, U. *et al.* Internet of Things based Integrated Smart Home Automation System. **SSRN Electron**. **J**., 2020, DOI: 10.2139/ssrn.3645458.

HADI, M. S. *et al.* Voice-Based Monitoring and Control System of Electronic Appliance Using Dialog Flow API Via Google Assistant. **ICEEIE 2019 - Int. Conf. Electr. Electron. Inf. Eng. Emerg. Innov. Technol. Sustain. Futur.,** pp. 106–110, 2019, DOI: 10.1109/ICEEIE47180.2019.8981415.

RAJALAKSHMI, A.; SHAHNASSER, H. Internet of things using node-red and alexa, **17th Int. Symp. Commun. Inf. Technol. Isc.,** 2017, vol. 2018-Janua, pp. 1–4, 2017, DOI: 10.1109/ISCIT.2017.8261194.

YOFFIE, D. B. *et al.* Voice War: Hey Google vs. Alexa vs. Siri, **Harvard Business School Case**, 718-519, 2018. (Revised January 2020).

REVIEWS. The Best Voice Assistants of 2020. **Reviews.com**. Disponível em: https://www.reviews.com/home/smart-home/best-voice-assistant/. Acesso em 20 abr. 2021.

BERDASCO, A. *et al.* User Experience Comparison of Intelligent Personal Assistants: Alexa, Google Assistant, Siri and Cortana, **Proceedings**, 2019, DOI: 10.3390/proceedings2019031051.

THAMPI, S. M. *et al.* Advances in Signal Processing and Intelligent Recognition Systems, **Communications in Computer and Information Science 968.**, 2021. Disponível em: http://www.springer.com/series/7899. Acesso em 18 abr. 2021.

NASCIMENTO, F. S. do. ESP32 e Google Assistant: Controlando por comando de voz. **FilipeFlop**, 2020. Disponível em: https://www.filipeflop.com/blog/controlando-o-esp32-por-comandos-de-voz-via-google-assistant/. Acesso em 20 abr. 2021.

ORACLE. Applets. **ORACLE**. Disponível em: https://www.oracle.com/java/technologies/applets.html. Acesso em 02 mai. 2021.

IFTTT. About - IFTTT. **IFTTT**. Disponível em: https://ifttt.com/about. Acesso em 02 mai. 2021.

ADAFRUIT. Welcome to Adafruit IO. **ADAFRUIT**. Disponível em: https://io.adafruit.com/. Acesso em 10 mai. 2021.

FERNANDES, S. R. Dialogflow vs Lex vs Watson vs Azure Bot - Chatbot Platforms Quick Comparison, 2020. **Linkedin**. Disponível em: https://www.linkedin.com/pulse/dialogflow-vs-lex-watson-azure-bot-chatbot-quick-sherwin-fernandes. Acesso em 11 mai. 2021.

PETTERS, J. AWS vs Azure vs Google: Cloud Services Comparison. **Varonis,** 2020. Disponível em: https://www.varonis.com/blog/aws-vs-azure-vs-google/. Acesso em 12 mai. 2021.

BLANCHON, B. StaticJsonDocument. **ArduinoJson 6**, 2014-2021. Disponível em: https://arduinojson.org/v6/api/staticjsondocument/. Acesso em 24 out. 2021.

GOOGLE. Programa gratuito do Google Cloud. **Google Cloud**. Disponível em: https://cloud.google.com/free/docs/gcp-free-tier#free-tier-usage-limits. Acesso em 25 out. 2021.

NODE-RED. Running under Docker : Node-RED. **Node-RED**. Disponível em: https://nodered.org/docs/getting-started/docker. Acesso em 13 out. 2021.

NODE-RED. Node-RED. **Node-RED**. Disponível em: https://nodered.org/. Acesso em 12 out. 2021.

SMARTNORA. NORA. **Smart NORA**. Disponível em: https://smart-nora.eu/. Acesso em 12 out. 2021.

GITHUB. GitHub - andrei-tatar/node-red-contrib-smartnora: Node Red Google Home integration. **GitHub**. Disponível em: https://github.com/andrei-tatar/node-red-contrib-smartnora. Acesso em 12 out. 2021.

GOOGLE. Overview | Actions on Google Smart Home. **Google Developers.** Disponível em: https://developers.google.com/assistant/smarthome/overview. Acesso em 27 out. 2021.

LAWSON, M. “4 things you need to know about the future of marketing.”. **Think With Google**. Disponível em: https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/future-of-marketing-mobile-micro-moments/. Acesso em 27 out. 2021.