Controle de Dispositivos Residenciais utilizando IoT

Ana Clara dos Santos Rosa, Arielli Ajudarte da Conceição,
Eliza Aparecida Crisóstomo Reis, Marcelo de Oliveira Marques, Antonio Alves Ferreira Junior
Instituto Nacional de Telecomunicações - Inatel
ana.ac@ges.inatel.br, arielli.a@get.inatel.br, elizaaparecida@get.inatel.br, marcelo@inatel.br, antonioa@inatel.br

***Abstract*—This project aims to create a fully connected environment, that allows the control of devices in a house, using the concept of Internet of Things. In this, an application is responsible for sending the control information, through the MQTT protocol, so that, according to the programming inserted in a Node MCU ESP8266, it is possible to load simple loads, such as a lamp or a curtain. In addition to a temperature control performed by an air conditioning simulator. The tests demonstrated that the components fit the proposal, concluding that the prototype reaches its initial objective**

***Index Terms*—App, Broker, IoT, MQTT, Node MCU**

***Resumo*—Este trabalho objetiva a criação de um ambiente conectado que possibilite o controle de dispositivos de uma casa, utilizando o conceito de Internet das Coisas. Nele, um aplicativo é o responsável por mandar as informações de controle, através do protocolo *MQTT*, para que, de acordo com a programação inserida em um *Node MCU ESP8266*, seja possível acionar cargas simples, tal como uma lâmpada ou uma cortina. Além de controle da temperatura, realizado por um simulador de ar-condicionado. Os testes demonstraram que os componentes adequam-se à proposta, concluindo que o protótipo atinge seu objetivo inicial.**

Palavras chave*—Aplicativo, *Broker, IoT, MQTT, Node MCU

I. Introdução

De acordo com o [1], Internet das coisas (*IoT*) se refere ao conceito de uma interligação das coisas (físicas e virtuais) baseadas na interoperabilidade das tecnologias de informação e de comunicação existentes e em evolução.

”Dispositivos Residenciais”é um projeto realizado em parceria com o *Smart Campus* do *Inatel*, que faz a prototipação de um ambiente automatizado, de modo que seja possível representar a comunicação entre vários equipamentos de uma casa. Para isso é utilizado o protocolo de aplicação *MQTT*, o qual possibilita a troca de mensagens e comandos entre a aplicação e o *Hardware*.

II. Metodologia

Este trabalho é uma pesquisa-ação que faz uso da Metodologia da Engenharia e, para ser desenvolvido, foi dividido em etapas menores executadas por cada uma das integrantes da equipe, e concluído para que pudesse ser implementado e validado.

Faz uso de Eletrônica Básica, Microcontroladores e Desenvolvimento de *Software*, enquadrando-se nas áreas de Automação Residencial, Engenharia de Telecomunicações, Engenharia de *Software* e Redes de Telecomunicações.

III. Desenvolvimento

O projeto é dividido em *Software*, a parte de programação, e *Hardware*, a parte física.

A. SOFTWARE

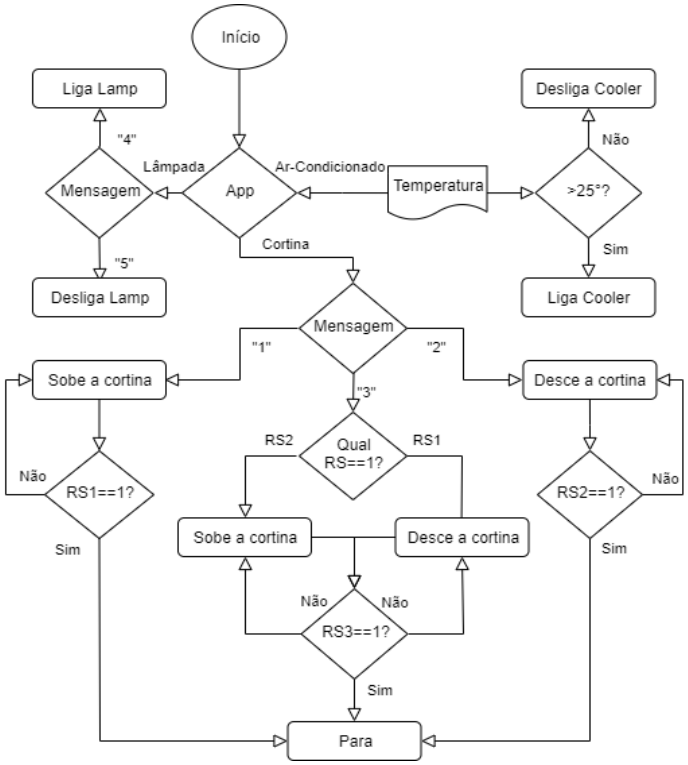


Fig. 1. Fluxograma.

O fluxograma exibido na Figura 1 simplifica o funcionamento geral do projeto composto pelo código do aplicativo e o do microcontrolador.

a.1) Aplicativo

O aplicativo consiste em uma interface com o usuário. Ele é o encarregado de enviar e receber dados atráves do protocolo *MQTT*, desenvolvido através da plataforma *Android Studio*, e foi programado na linguagem *Java*. Essa plataforma se trata de uma *IDE* que permite a criação de aplicativos para dispositivos móveis *Android*.

Ele oferece a opção de enviar comandos através da fala ou por botões convencionais. Caso o usuário decida operar o aplicativo por comando de voz, é necessário estar em um ambiente sem muitos ruídos, pressionar o botão de voz e falar o procedimento pretendido, de forma clara e objetiva. Se o usuário desejar que a cortina suba, por exemplo, deverá dizer ”subir cortina”, e assim por diante. A aplicação do comando de voz foi feita através do método *setOnClickListener*, nativo do *Android*, responsável pela captação de fala e transcrição para texto.

Seguindo a lógica do fluxograma, no aplicativo há três ítens disponíbilizados o ar-condicionado, a cortina e a lâmpada. Estes são os dispositivos que estão atualmente conectados ao projeto e podem ser visualizados na tela inicial, mostrada na Figura 2.

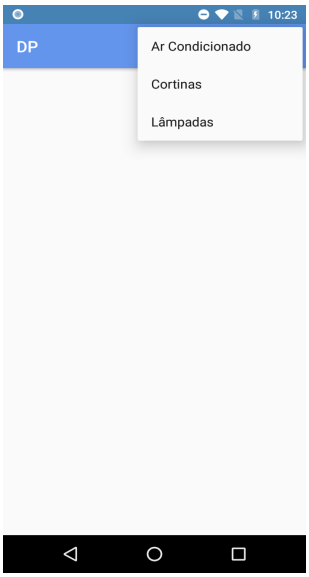


Fig. 2. Ítens.

Após a seleção do ítem da cortina ou o da lâmpada, uma tela do dispositivo em questão é aberta, onde são oferecidos alguns botões com as opções de controle e outro para acionar o comando de voz. Quando um comando é feito, por qualquer um desses meios, é enviado um número de 1 à 5 ao *broker MQTT*, de acordo com o que foi selecionado. Esse número é publicado e recebido pelo *Node MCU*, onde a ação é realizada.

Na Figura 3 é mostrada a tela da cortina com os botões com as opções de controle da movimentação e o de acionamento do comando de voz, o qual exemplifica o que foi citado acima.

Fig. 3. Tela Cortina.

Já escolhendo o ar-condicionado, são exibidas no aplicativo, Figura 4, as informações de temperatura que foram recebidas da comunicação *MQTT*.

Fig. 4. Tela Temperatura.

a.2) Node MCU

O *Node MCU ESP8266* é o ”cérebro”do projeto. É nele que é realizada a ação requerida no aplicativo. Isso ocorre devido a um código *Wiring* baseado em *C/C++*, programado na IDE do *Arduino 1.0.1*, que permite a conexão com a *Internet* e com o *broker MQTT*, o recebimento das informações da comunicação *MQTT* e dos sensores de entrada, o processamento de todos esses dados e a ação de resposta dos dispositivos de saída. Essas informações recebidas do *broker* são os números publicados através dos comandos enviados pelo aplicativo. Se for pedido para acender uma lâmpada, por exemplo, o *app* manda essa mensagem, através do protocolo, o *ESP8266* analiza-a e, de acordo com sua função, faz com que a lâmpada acenda.

Detalhando o funcionamento geral, por exemplo, da cortina, quando o usuário solicita o acionamento de subida ou descida, os números ”1”e ”2”são enviados, respectivamente. Esses números são recebidos pelo microcontrolador que faz com que o motor execute a ação desejada enquanto o *Reed Switch* (abreviado para RS no fluxograma) não responde com um nível lógico alto, mostrando que a cortina alcançou seu limite e deve parar. Além disso, há a oportunidade de fazê-la parar no meio, através de um terceiro RS, que se encontra no meio dos outros dois sensores. Seu funcionamento baseia-se no lugar em que a cortina parou seu movimento, ou seja, em qual *Reed Switch* ela parou ao ter sido fechada ou aberta. Dependendo dessa localidade, a cortina deve subir ou descer até que alcançe esse terceiro sensor. A organização e explicação do funcionamento desses sensores podem ser vistas no tópico de *Hardware*.

Para o ar-condicionado, no entanto, o *ESP8266* publica os dados de temperatura no *broker* para o aplicativo receber e, além disso, gerencia o acionamento da ventoinha/*cooler*.

B. HARDWARE

No diagrama em blocos da Figura 5 são mostrados os principais módulos do projeto, descritos a seguir.

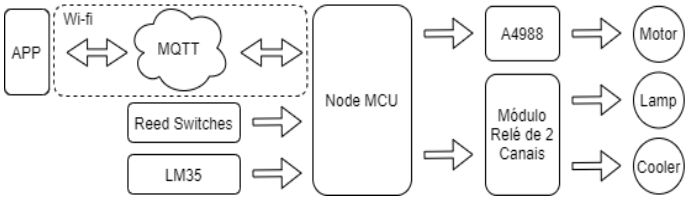


Fig. 5. Diagrama em Blocos.

b.1) ar-condicionado

O sensor de temperatura *LM35* e uma ventoinha de computador de 12V (*cooler*) simulam um ar-condicionado. Quando a temperatura medida pelo sensor ultrapassa 25°C, a ventoinha liga para estabilizá-la, com a ajuda de um relé para acionamento/desacionamento. Após isso, os dados de temperatura são publicados no *broker* e mostrados no aplicativo.

b.2) cortina

De acordo com a opção de controle selecionada no aplicativo, o *Node MCU* faz com que a cortina suba, desça ou pare no meio. Sua movimentação é realizada pelo motor de passo *Nema 17*. O controle de seus passos ocorre por meio de dois pinos do microcontrolador, ligados aos pinos *STEP* (passo) e *DIR* (direção) do módulo *A4988*. Na Figura 6 é apresentada a montagem entre *driver*, o motor, e o *ESP8266*.

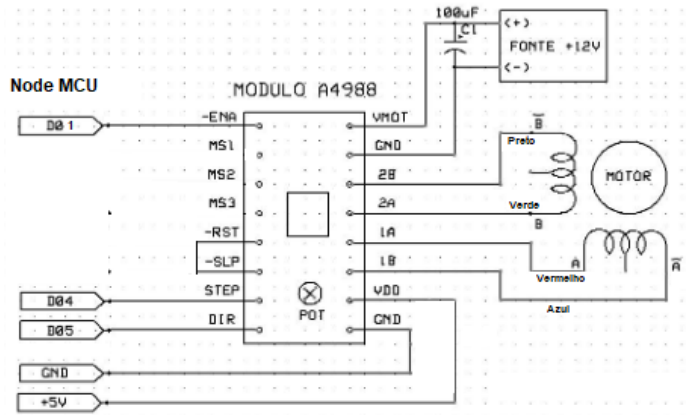


Fig. 6. Diagrama de Montagem.

A cortina é feita de um tecido simples e foi projetada de acordo com o modelo Persiana Rolô, que tem um rolo cujo movimento faz com que ela suba ou desca.

Os sensores magnéticos *Reed Switches* definem quando essa movimentação deve parar. Eles fazem isso devido às suas características de fecharem seus contatos quando são excitados por campos magnéticos de ímãs. Em vista disso, três sensores foram colocados ao lado da janela do protótipo e um ímã foi preso a barra da cortina, de modo que, quando passe pelo *Reed Switch*, ele fecha contato, parando o movimento do motor, segundo a programação do microcontrolador.

Cada comando tem um sensor magnético de parada, portanto, não há como um interferir no outro.

b.3) lâmpada

Uma lâmpada, alimentada pela rede elétrica, também com a ajuda de um relé para acionamento, acende ou apaga a partir dos comandos requeridos no aplicativo.

b.4) Node MCU

Segundo o [2], o *Node MCU* é uma plataforma open source da família *ESP8266*, de baixo custo, suporte integrado a redes *Wi-Fi*, tamanho reduzido e baixo consumo de energia, o que o torna ideal para aplicações em *IoT*.

Sua função no projeto é fazer todo o processamento dos dados recebidos da comunicação *MQTT* e pelos sensores e enviá-los para os dispositivos de saída, para que assim possam realizar seus respectivos comandos.

b.5) protótipo

Durante o desenvolvimento do projeto houve várias versões de protótipos para o projeto. A mais atual pode ser vista na Figura 7.

Fig. 7. Protótipo final.

Ela foi produzida no *FabLab* do *Inatel* através de uma máquina de corte à laser. Os suportes, a cortina, entre outros detalhes, foram ajustados pela marcenaria e também pelas integrantes da equipe.

A alimentação do protótipo implementado consiste em uma fonte de 5V para alimentar o *Node MCU*, o módulo relé e os *Reed Switches*, outra de 12V para a ventoinha, o motor e o driver e a rede elétrica de 127V eficazes para a lâmpada.

C. PROTOCOLO MQTT

Segundo o [3], o *MQTT* é um protocolo simples e leve, que permite a conexão com a *Internet*, o que o torna padrão para tecnologias *IoT*.

No diagrama ele é visualizado como uma nuvem entre o *App* e o microcontrolador, enquanto as setas mostram que ao mesmo tempo que ambos recebem mensagens, eles também enviam, através da rede *Wi-fi*.

Ele funciona como esquema de troca de mensagens que é fundamentado no modelo *Publish-Subscribe*, no qual há um *broker*, um servidor que recebe e encaminha mensagens, e

vários clientes, que são os equipamentos conectados a esse servidor.

Quando um cliente se conecta a um *broker MQTT*, ele assina um ”tópico”qualquer. Ao ser publicada uma mensagem pelo usuário nesse tópico, o *broker* encaminha-a a todos os clientes que o assinarem. Seguindo essa organização, é possível especificar que determinados clientes somente podem interagir com determinadas mensagens. Nesse projeto, por exemplo, as mensagens que são recebidas do aplicativo estão em um tópico (o ”app”) e as que são enviadas à ele estão em outro (”temp”), justamente para que as mensagens não se misturem.

D. TESTES REALIZADOS

A princípio, iniciou-se o desenvolvimento do *Firmware* do microcontrolador, adaptado do [4], que contêm os comandos envolvendo a comunicação *MQTT*. Ele foi programado através da *IDE* do Arduino, ensinado no tutorial em [5]. Após isso, funções foram criadas para cada um dos dispositivos controlados.

Esses componentes foram previamente testados em programas exemplos, encontrados em alguns tutoriais, verificando se funcionavam corretamente e se atendiam ao que era requerido. Eles foram implementados no programa principal do *Node MCU*, e em seguida, testados com os comandos solicitados pela própria Serial do *Arduino*. Posteriormente foi utilizado o aplicativo *MyMQTT*, e a plataforma chamada *CloudMQTT*, encontrada em [6], para testar a comunicação *MQTT*, e só depois de serem incluídos no aplicativo eles foram testados e validados.

A Figura 8, da serial do *Arduino* mostra as conexões com a *Internet* e com o *broker* sendo concluídas, a chegada das mensagens e a realização dos comandos.

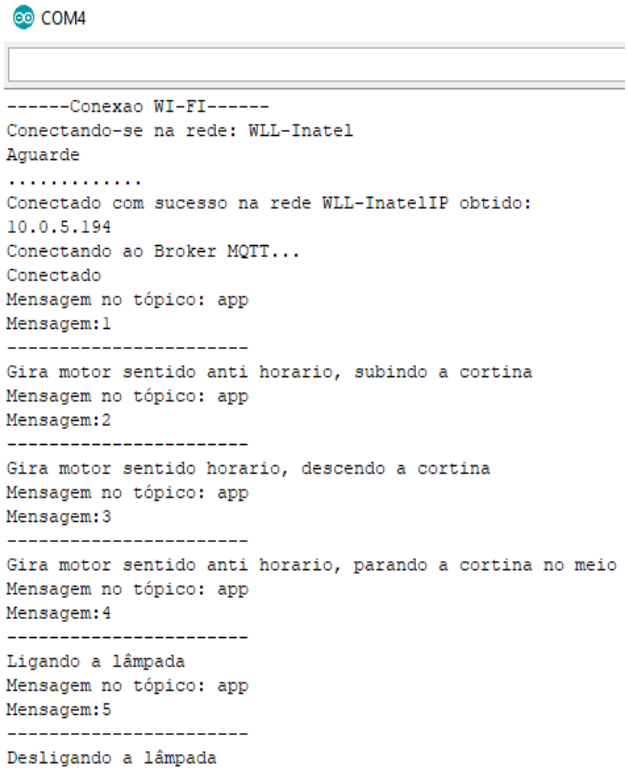


Fig. 8. Serial do Arduino.

O aplicativo foi iniciado com base no tutorial em [7]. Para testá-lo, utilizaram-se emuladores tanto do próprio *Android Studio*, quanto celulares requisitados do laboratório.

O primeiro ítem controlado pelo projeto foi a cortina. O código e a montagem entre o driver e o motor foi baseado no tutorial em [8]. Antes de inserir no protótipo, foi visualizado o giro do motor em funcionamento e testada também a sua força, verificando se o motor conseguiria movimentá-la e não espanar.

Com a inserção das opções de controle da cortina na tela principal do aplicativo, o Node MCU pôde ser testado junto ao *app*. Após os testes foi visto que o motor realizava movimentos que não estavam dentro do proposto. Mais tarde, foi descoberto que esses movimentos eram realizados para que ele voltasse à posição inicial de seu giro. Esse erro foi corrigido com a implementação de um comando de sua biblioteca no código do *Node MCU* (*setCurrentPosition(0)*), conhecido no site [9], que, após um giro, redefinia-se a posição em que foi parado para a nova posição 0, impedindo que o motor tentasse fazê-la sozinho.

Até então, a programação do motor da cortina era girar por um tempo de aproximadamente 7 segundos, o que às vezes fazia com que ela subisse um pouco mais ou menos do que o esperado. Isso foi corrigido com a implementação de 2 *Reed Switches*: um para cada comando de descer e subir a cortina. Para possibilitar mais opções de controle, foi acrescentada, tanto no aplicativo quanto no codigo do *Node MCU*, a opção ”Parar no meio”, o que levou a inserção de um terceiro sensor. A inclusão desses sensores foi baseada no tutorial em [10].

Posteriormente à conclusão da cortina, a lâmpada foi incluída nos programas, mas, na prática, ela não acendia. Para arrumar isso, foi seguido o tutorial em [11], onde foi acrescentado um relé para acionamento.

Por último, foi anexado mais um ítem, o ar-condicionado. Para isso, foi feito um simulador. Após os testes, foi visto que o *cooler*/ventoinha também precisava de um relé para acionamento e foi seguido o tutorial em [12] para tal.

Como eram necessários 2 relés, tanto para a lâmpada quanto para a ventoinha, foi priorizada a utilização de um módulo relé de 2 canais.

Após todos os testes terem sido concluídos com êxito, deu-se prioridade ao aplicativo, que apenas contava com vários botões na tela principal sem nenhuma separação e organização de ítens. Esse problema foi resolvido com auxílio de vídeo-aulas ínstrutivas [13, 14], que ajudaram na organização do *layout*. Novas abas foram criadas para os ítens, além de um menu de escolhas para cada um deles. A tela do ar-condicionado, no entanto, continha o valor de temperatura que estava sendo medido pelo *LM35*. Procurou-se acrescentar um gráfico com esses valores, para que o usúario obtivesse mais informações sobre o dispositivo, mas, após várias falhas, a ideia foi abandonada, dando prioridade para outros recursos importantes, como o controle por voz.

Ele foi implementado utilizando o método nativo do *Android*. Com esse recurso é possível captar o áudio enviado pelo aplicativo e transformá-lo em texto, dando a possibiidade de enviar comandos para os dispositivos sem precisar de muitos movimentos. Inicialmente foi realizado um teste padrão

para entendimento do método, mas foi encontrado um erro de compatibilidade de comandos envolvendo os botões e o comando por voz. Para resolvê-lo,foi utilizada a comparação de *strings*, de modo que somente um comando fosse enviado para o *broker*. Com isso, obteve-se êxito nessa função.

IV. Conclusão

Objetivando criar um ambiente automatizado, o projeto "Dispositivos Residenciais"simula uma casa inteligente. Ao usuário é dado o direito de determinar as ações que ele quer realizar na casa enquanto o projeto faz com que essas ações sejam realizadas, devido ao *ESP8266* e um conjunto de sensores, entre outros componentes.

Compreender ainda mais o *IoT* na prática foi um dos maiores conhecimentos adquiridos, além de conhecer alguns componentes e trabalhar com o protocolo *MQTT*.

Algumas dificuldades surgiram na elaboração do projeto, tais como o desenvolvimento no *Android Studio*, já que não havia prática com a linguagem *Java*, além de vários problemas de instabilidade da plataforma. Ademais, o controle dos dispositivos através do reconhecimento da fala foi fonte de alguns problemas devido a falta de conhecimento desta tecnologia. Vários testes foram realizados a fim de gerar maior entendimento.

Para futuras versões, poderão ser implementados novos equipamentos, melhorar e automatizar ainda mais o comando por voz do aplicativo e inserir mais configurações, de forma que o ambiente seja totalmente controlado para uma ação em específico, como estudar ou dormir. Além de acrescentar novas funções de controle de temperatura.

Referências

[1] ITU-T. *Global Standards for the Internet of Things*. URL: www . itu . int / en / ITU - T / techwatch / Pages / internetofthings (acesso em 10/2019).

[2] Master Walker. *NodeMCU – Uma plataforma com características singulares para o seu projeto ioT*. URL: https : / / blogmasterwalkershop . com . br / embarcados/nodemcu/nodemcu-uma-plataforma-com-caracteristicas - singulares - para - o - seu - projeto - iot/ (acesso em 10/2019).

[3] IBM. *Conhecendo o MQTT*. URL: https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html (acesso em 10/2019).

[4] Filipeflop. *Controle e Monitoramento IoT com NodeMCU e MQTT*. URL: https://www.filipeflop.com/blog/controle-monitoramento-iot-nodemcu-e-mqtt/ (acesso em 09/2019).

[5] RoboCore. *Como programar o NodeMCU com Arduino IDE*. URL: https://www.robocore.net/tutoriais/como-programar - nodemcu - arduino - ide . html (acesso em 09/2019).

[6] CloudMQTT. *Hosted message broker for the Internet of Things*. URL: https://www.cloudmqtt.com (acesso em 09/2019).

[7] Wildans Tech Blog. *MQTT Android Client Tutorial*. URL: https : / / wildanmsyah . wordpress . com / 2017 / 05 / 11/mqtt-android-client-tutorial/ (acesso em 09/2019).

[8] Arduino e Cia. *Como usar o driver A4988 com motor de passo Nema 17*. URL: https://www.arduinoecia.com.br/driver-a4988-com-motor-de-passo-nema-17/ (acesso em 09/2019).

[9] AccelStepper. *AccelStepper Class Reference*. URL: https: / / www . airspayce . com / mikem / arduino / AccelStepper / classAccelStepper.html (acesso em 11/2019).

[10] Engcorper. *Tutorial: Sensor Magnético com Arduino*. URL: http : / / engcomper . blogspot . com / 2011 / 09 / tutorial - sensor - magnetico - com - arduino . html (acesso em 10/2019).

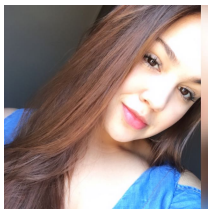
[11] FilipeFlop. *Controlando lâmpadas com Módulo Relé Arduino*. URL: https://www.filipeflop.com/blog/controle-modulo-rele-arduino/ (acesso em 10/2019).

[12] Maker Pro. *How to Make a Temperature-Controlled Fan Using Arduino*. URL: https://maker.pro/arduino/projects/how - to - make - a - temperature - controlled - fan - using - arduino (acesso em 11/2019).

[13] CODEficando. *Como criar Menus no ActionBar da activity usando Android Studio*. URL: https : / / maker . pro / arduino / projects / how - to - make - a - temperature - controlled-fan-using-arduino (acesso em 12/2019).

[14] CODEficando. *Como invocar Activities e passar parâmetros no Android Studio*. URL: https : / / www . youtube . com / watch ? v = NvYPmMgAmcw & feature = youtu.be (acesso em 12/2019).

Autores



Ana Clara dos Santos Rosa é Técnica em Informática (2018) pelo Instituto Federal de Ensino, Ciência e Tecnologia do Sul de Minas - Campus Pouso Alegre. Graduanda em Engenharia de Software pelo Instituto Nacional de Telecomunicações - INATEL. Bolsista de Iniciação Científica no CS&I Lab na área de Internet of Things com o projeto "Dispositivos Residenciais".

Arielli Ajudarte da Conceição é Técnica em Telecomunicações (2018) pela Escola Técnica em Eletrônica - ETE "FMC". Graduanda em Engenharia de Telecomunicações pelo Instituto Nacional de Telecomunicações - INATEL. Monitora de Introdução à Engenharia de Telecomunicações no INATEL. Bolsista de Iniciação Científica no CS&I Lab na área de Internet of Things com o projeto "Dispositivos Residenciais".