Controle e Monitoramento de uma Residência Utilizando Conceito de Internet das Coisas e Armazenamento de Dados em Nuvem

Euler Moreira Barbosa
Discente do Curso de Engenharia Mecatrônica
Universidade Federal de São João del-Rei
eulerbarbosa96@gmail.com

Edgar Campos Furtado
Depto. Engenharias de Telecomunicações e Mecatrônica
Universidade Federal de São João del-Rei
edgar@ufsj.edu.br

Resumo -- Internet das coisas ou IoT (*Internet of Things*) é o modo como objetos do cotidiano estão conectados em rede, se comunicando entre si e com usuários, fornecendo dados através de sensores e realizando ações através de atuadores. Este trabalho tem por objetivo desenvolver um sistema de controle e monitoramento aplicável ao contexto residencial, usando o conceito IoT e o armazenamento de dados em Nuvem, via plataforma Firebase©. No sistema proposto o usuário será capaz de realizar o acionamento elétrico de cargas como, pontos de iluminação, portões eletrônicos, eletrodomésticos, bem como realizar o monitoramento de sinais como, por exemplo, temperatura e umidade. Para isso será desenvolvido um aplicativo Android© e um aplicativo Web que será a interface entre o usuário e o sistema.

Palavras-chave – Internet das Coisas; Firebase©; Sistema Android©; Microcontrolador; Automação Residencial.

I. INTRODUÇÃO

Automação residencial é a integração de diversos sistemas de uma casa, com o objetivo de realizar o monitoramento/controle de uma maneira simples e eficiente. O objetivo é automatizar tarefas do dia-a-dia, propiciando maior conforto e comodidade para os moradores. Desde tarefas simples, como acionar uma lâmpada, até tarefas mais complexas, como o monitoramento da temperatura ou de imagens por câmeras de segurança.

Aliado ao conceito de Internet das Coisas (IoT, *Internet of Thinks*) é possível realizar o controle e o monitoramento da residência remotamente, desde que se tenha acesso a internet.

A internet é um conjunto de redes que se conectam entre si formando uma imensa teia, possibilitando o compartilhamento remoto de dados entre dispositivos [1]. A comunicação entre esses dispositivos é feita por meio do protocolo TCP/IP (*Transmission Control Protocol/ Internet Protocol*), que funciona com o modelo cliente/servidor, no qual um dispositivo envia solicitações a outro dispositivo. O TCP realiza a quebra da mensagem em partes menores, enviando-as pela internet. O dispositivo que recebe essas partes utiliza outra ferramenta do TCP para reunir a mensagem original.

No Brasil o principal meio de acesso à internet é o celular. Em 2017, 49% dos lares brasileiros dependiam de um celular para acessar a rede mundial de computadores. Além disso, computadores e tablets também são dispositivos de acesso à internet bastante comum [2].

O sistema operacional mais utilizado entre dispositivos móveis, como o celular, é o Android©. Esse sistema é baseado em Linux© e pertence à empresa Google©, tendo sido lançado em 2008 [3].

Tendo em vista tais aspectos, este trabalho tem como objetivo apresentar o desenvolvimento de um sistema de controle/monitoramento para uma residência, baseado no conceito *IoT* e plataforma FireBase©, como banco de dados e a autenticação de usuários. Para realizar a interface com o usuário será desenvolvido um aplicativo Android©, e um Web Site, que possibilitará o acesso ao sistema através de um navegador Web em qualquer sistema operacional. As ações propostas pelo sistema podem ser resumidas como:

- realizar o controle e o monitoramento de uma residência;
- ser acessível de qualquer dispositivo que tenha acesso a internet;
- realizar o armazenamento e a disponibilização de dados por meio da computação em nuvem;
- ser de caráter expansível e flexível.

Foram encontrados na literatura alguns trabalhos correlatos. Em [4] o autor apresenta uma proposta de sistema de automação industrial utilizando computação em nuvem, de caráter expansível, que realiza toda a tratativa de dados na nuvem e é acessível de qualquer dispositivo com conexão a internet. Em [5] é apresentado um sistema de automação residencial que busca propiciar a simplicidade de implementação e manutenção, na qual o usuário interage com o sistema por meio de um aplicativo para dispositivos móveis. Em [6] os autores se propõem a desenvolver um projeto de automação residencial, implementando um sistema eletrônico com foco em tornar o ambiente residencial mais acessível para pessoas portadoras de deficiência física.

O texto é organizado como segue. Na Seção II é apresentado o diagrama de blocos do sistema proposto juntamente com um breve conceito de todas as tecnologias envolvidas. Na Seção III é apresentada a metodologia utilizada no projeto, expondo todas as etapas de desenvolvimento. Na seção IV são apresentados resultados, e discussões sobre as funcionalidades do sistema. Na seção V é abordada uma proposta de melhorias no sistema para projetos futuros. E na seção VI é apresentada a conclusão do trabalho.

II. Sistema Proposto

A Figura 1 apresenta o diagrama de blocos do sistema.

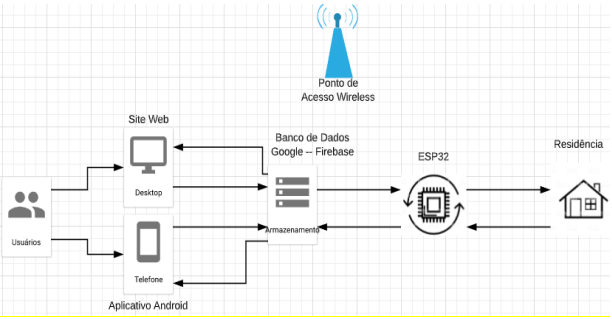

Figura 1: *Diagrama de Bloco do Sistema.*

Neste projeto será utilizado o microcontrolador ESP32, para realizar tradução local dos comandos, o Google – Firebase© como o serviço de computação em Nuvem e será criado um aplicativo Android© e um site Web que servirá de interface entre o usuário e o sistema.

A. *Microcontrolador (µC) ESP32*

O µC ESP32 foi desenvolvido pela empresa chinesa Espressif Systems. Possui conectividade Wi-Fi e Bluetooth integrados, o que facilita o uso de tais µC em *IoT*. Além disso, possui as seguintes especificações:

- CPU: Xtensa© Dual-Core 32-bit LX6;
- Clock máximo: 240 MHz;
- Memória ROM: 448 KBytes;
- Memórias: RAM: 520 Kbytes, e Flash: 4 MB;
- Wifi padrão 802.11 b/g/n, de 2.4 Ghz (máximo: 150 Mbps), com Wi-Fi Direct (P2P), P2P Discovery, P2P Group Owner mode e P2P Power Management;
- Antena embutida na PCI;
- Bluetooth BLE 4.2;
- Conector micro USB (comunicação e alimentação);
- Modos de operação: STA/AP/STA+AP;
- Portas GPIO: 11, com funções de PWM, I2C, SPI;
- Tensão de operação: 4,5V a 9 V;
- 16 conversores analógico digital (ADC).

Os terminais GPIO podem fornecer corrente de até 12 mA. Além disso, a PCI possui dez sensores de toque capacitivos e um sensor de temperatura, que podem ser usados para monitorar a temperatura de operação do µC. A PCI disponibiliza ainda tensões: 5,0V, 3,3V, GND, terminais de comunicação serial TX e RX, entre outros [7]. A figura 2 apresenta o diagrama de blocos dos componentes internos do µC ESP32. Observa-se que, de fato, é um µC versátil para aplicações *IoT*.

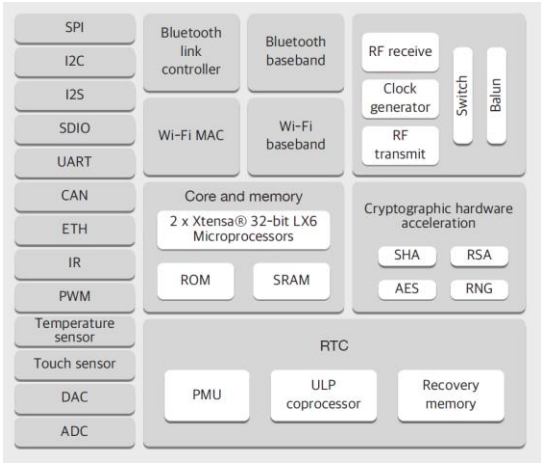

Figura 2: *Diagrama de blocos de componentes internos do ESP32.[20]*

Para a programação do ESP32 utilizou-se o ambiente de desenvolvimento integrado (IDE, *Integrated Development Environment*) gratuito Arduino IDE. Entretanto, após instalação do Arduíno IDE é preciso instalar o drive da placa ESP32, disponível em: *https://github.com/espressif/arduino-esp32.git*.

O Arduino IDE usa linguagem baseada no C/C++, que é bem difundida e de fácil aprendizado.

B. *Computação em Nuvem*

A computação em nuvem é o fornecimento de serviços de computação, incluindo servidores, armazenamento, bancos de dados, rede, software, análise e inteligência, pela Internet, conhecida como: a nuvem.

Em geral, os serviços de computação em nuvem são pagos, existindo opções gratuitas, porém com limitações.

Há três diferentes maneiras de implantar os serviços de computação em nuvem [8], ou seja:

- **Nuvem pública:** pertencem a um provedor de serviço terceirizado, que fornece recursos de computação pela Internet. Nesse tipo de nuvem, todo o hardware, software e outras infraestruturas de suporte são de propriedade e gerenciadas pelo provedor de nuvem. O acesso a esses serviços e a gerência é feito, em geral, por um navegador Web;
- **Nuvem privada:** Se refere aos recursos de computação em nuvem usados exclusivamente por uma única empresa ou organização. Uma nuvem privada pode estar localizada fisicamente no datacenter local da empresa;

- **Nuvem hibrida**: Combinam nuvens públicas e privadas ligadas por uma tecnologia que permite que dados e aplicativos sejam compartilhados entre elas. Permitindo que os dados e os aplicativos se movam entre nuvens privadas e públicas. [8].

Pode-se dividir a computação em nuvem em duas partes: o *front-end* (contato direto do usuário final) e o *back-end* (contato mais próximo à origem do recurso), ambos se conectam por uma rede, usualmente a internet. O *front-end* é o lado do usuário no computador, e o *back-end* é a sessão nuvem do sistema [9].

Banco de Dados: Firebase©

O Google Firebase© é um Baas (*Back-end as a Service*) para aplicações Web e Mobile. Lançado em 2004, o Baas é um serviço em que toda a estrutura do *back-end* como: configuração de servidor, integração com banco de dados, sistema de *push notification* e outros serviços, estão prontos para integração com o aplicativo desenvolvido [10]. Essa plataforma oferece vários serviços de computação em nuvem, que potencializa o desenvolvimento de aplicativos. Para acessar o Google – Firebase© é necessário possuir uma conta Google©.

O serviço possui diferentes planos pagos e um plano gratuito (plano Spark), com algumas limitações, mas que oferece quantidade considerável de recursos para o desenvolvimento de projetos. Os serviços disponibilizados no Firebase© podem ser separados em quatro categorias: *Analytics, Develop, Grow e Earn.*

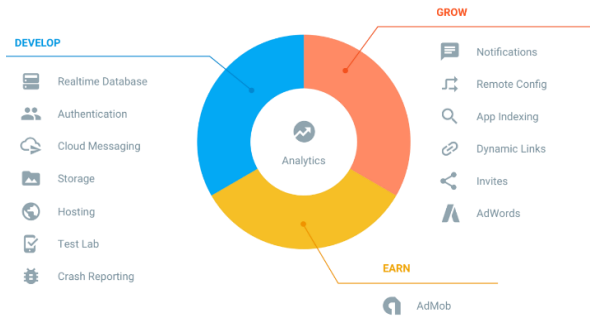

***Figura 3**: Serviços disponibilizados pelo Google -- Firebase.[21]*

Dentro dessas categorias podem ser citados os serviços:

- *Analytics*: é uma solução de análise de apps gratuita que fornece insights sobre o uso do aplicativo e o envolvimento do usuário;
- *Realtime Database*: é um banco de dados hospedado na nuvem NoSQL. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados e permanecem disponíveis quando o app está off-line;
- *Authentication*: fornece serviços de *back-end*, SDKs (*Software Development Kit*) e bibliotecas de IU (interface de usuários) prontas para autenticar usuários no aplicativo. Possui suporte à autenticação por meio de senhas, números de telefone e provedores de identidade federados como Google, Facebook, e Twitter;
- *Cloud Messaging*: é uma solução de mensagens entre plataformas que permite o envio confiável de notificações sem custo. Permite, também, notificar um app cliente de que dados estão disponíveis para sincronização ou enviar mensagens de notificação para promover novas interações e a retenção de usuários;
- *Cloud Storage*: é um serviço de armazenamento de objetos criado para a escala do Google©. Com os SDK do Firebase© para *Cloud Storage*, usa-se a segurança do Google para fazer o upload e o download de arquivos nos aplicativos Firebase [11].

C. Android©

O Android© é um sistema operacional para dispositivos móveis baseado no Linux©. Foi desenvolvido pela *Open Handset Alliance*, uma aliança entre várias empresas, dentre elas a Google©.

Dentre as vantagens sistema Android© pode-se citar: a integração com os outros serviços oferecidos pela Google© e a quantidade de aplicativos disponíveis, sendo muitos deles gratuitos [12].

Dentre as várias linguagens de programação para desenvolvimento de aplicativos Android©, o Google© indica as linguagens Kotlin, Java e C/C++.

O IDE oficial do Android© é o Android Studio, sendo baseado no IntelliJ IDEA. Além do editor de código e das ferramentas de desenvolvedor avançadas do IntelliJ©, o Android Studio oferece recursos para aumentar a produtividade na compilação de apps Android, como:

- sistema de compilação flexível baseado em Gradle;
- emulador rápido com inúmeros recursos;
- ambiente unificado, possibilita o desenvolvimento para todos os dispositivos Android;
- realização de alterações ao aplicativo em execução sem necessidade de reiniciar o aplicativo;
- compatibilidade com C++ e NDK;
- frameworks e ferramentas de teste;
- modelos de código e integração com GitHub, auxiliando na criação de recursos comuns de apps e importação de exemplos de código;
- ferramentas de lint para detectar problemas de desempenho, usabilidade, compatibilidade com versões, entre outros;
- compatibilidade integrada com o *Google Cloud Platform*, facilitando a integração do *Google Cloud Messaging* e do *App Engine* [13].

O Android Studio tem disponíveis as opções de linguagem em Java e Kotlin.

D. Desenvolvimento Web

O desenvolvimento web trata da criação de um app, um portal ou um site, conectado a uma rede, podendo ser a internet ou uma rede interna. Os sistemas web são desenvolvidos para realizar operações em tempo real.

Um site Web pode ser acessado de qualquer lugar do mundo, o que dá maior mobilidade para as pessoas que irão utiliza-lo. Além disso, não é necessário se preocupar com todo o suporte de software e hardware que irá hospedá-lo, já que o mesmo é feito na nuvem (Conceito de Computação em Nuvem citado anteriormente) [14].

Existem várias ferramentas que são importantes para o desenvolvimento Web, dentre as quais se pode destacar: o HTML, o CSS, o JavaScript e o PHP. Nas seções a seguir são apresentados os conceitos de cada uma delas.

HTML

HTML (*Hyper Text Markup Language*) é a linguagem de descrição de documentos usada para a produção de páginas na Web. A linguagem utiliza tags para definir os diferentes elementos, tais como texto, elementos multimídia, formulários, hiperlink, entre outros.

Ao acessar um sítio na internet, os navegadores identificam as *tag* e apresentam a página conforme está especificada. O usuário tem acesso direto com informações HTML, pois o navegador interpreta o documento, e o resultado é mostrado na tela [15].

CSS

Css (*Cascating Style Sheets*) é uma linguagem utilizada para adicionar estilos aos documentos web. Os estilos são usados para se definir a apresentação dos conteúdos e a aparência das páginas. Quando utilizado em conjunto com o HTML pode mudar todo o estilo de apresentação de uma página Web.

O Css pode ficar separado do HTML. Ao mudar um estilo no Css, mudam-se todos os componentes do HTML que estão utilizando aquele estilo [15].

JavaScript

É uma linguagem de programação com alguns recursos de orientação a objetos, sendo usada para deixar a página HTML mais dinâmica. Essa linguagem tem a capacidade de, durante a execução, criar mais tags HTML e exibir no navegador sem que a página tenha que ser atualizada [15].

PHP

O PHP (*Personal Home Page*) é uma linguagem de programação dinâmica para produção de web sites. É processado no servidor, retornando para o usuário do site apenas HTML. O PHP não é uma linguagem de programação compilada, como o C++ ou JAVA, mas sim interpretada pelo servidor. O script PHP pode conter ou não tags HTML [15].

Visual Studio Code

O Visual Studio é uma plataforma da Microsoft© para o desenvolvimento de aplicações em diversas linguagens, inclusive páginas Web. Essa plataforma possui atalhos de API disponíveis, além de fazer preenchimento automático dos comandos para agilizar a construção do código.

É umas das IDE mais relevantes do mercado, devido aos vários recursos de desenvolvimento disponibilizados [16].

III. METODOLOGIA

O projeto a ser desenvolvido tem como objetivo realizar a automação de uma residência, através do controle do acionamento de lâmpadas, portões eletrônicos, eletrodomésticos, entre outros. Além do monitoramento de variáveis do ambiente como temperatura e umidade. O usuário realizará esse controle através de um aplicativo Android©, instalado em um dispositivo móvel (celular ou tablet) ou através de um site Web, que poderá ser acessado por um navegador, instalado em qualquer dispositivo com acesso à internet.

Para armazenar os dados do sistema será utilizado o recurso *Realtime database* do Google – Firebase©. Podem-se dividir as variáveis do sistema em dois tipos:

- Variáveis digitais: São as variáveis que definem o estado dos objetos controlados como, por exemplo, os eletrodomésticos, sendo representadas no banco de dados por 0 (desligado) ou 1 (ligado);
- Variáveis analógicas: Trata-se de variáveis do ambiente, como temperatura e umidade, ou seja, podem assumir um conjunto ilimitado de valores dentro do range de medição.

Tendo em vista tais aspectos foi criado um projeto no Google – Firebase©, usando a versão gratuita (Plano Spark). Criou-se um banco de dados, no formato JSON (*JavaScript Object Notation*), dividido em variáveis digitais e variáveis analógicas.

Para a criação desse banco de dados foi acessado no menu do projeto a opção *Database* e escolhido o *Realtime DataBase*. Foram adicionados ao banco de dados três variáveis digitais, e quatro variáveis analógicas.

Foram criadas as classes *analog* e *digital.* Que contém as seguintes informações respectivamente:

```
public class analog {
    private String id;
    private double temp1Value;
    private double temp2Value;
    private double umid1Value;
    private double umid2Value;
public class digital {
    private String id;
    private int lamp1Port;
    private int lamp2Garagem;
    private int lamp3Varanda;
```

Essas classes serão utilizadas para salvar as variáveis analógicas e digitais, respectivamente, de modo a facilitar a manipulação das mesmas. No aplicativo serão realizadas as buscas, e serão realizados os envios de valores ao banco de dados.

A. Aplicativo Android

Após a criação do banco de dados, foi dado início ao desenvolvimento do aplicativo Android© e integração do mesmo ao projeto criado no Google – Firebase©. Para a programação do aplicativo foi utilizado a IDE Android Studio, na linguagem de programação Java©. Foram criadas quatro *activitys* (telas).

Para realizar a comunicação com o banco de dados foi necessário importar as seguintes bibliotecas:
com.google.firebase.FirebaseApp;
com.google.firebase.database.DataSnapshot;
com.google.firebase.database.DatabaseError;
com.google.firebase.database.DatabaseReference;
com.google.firebase.database.FirebaseDatabase;
com.google.firebase.database.ValueEventListener;

Foi necessário inicializar o Google – Firebase© no aplicativo criando-se uma variável *DatabaseReference.* Logo após, foi inicializado o banco de dados através de uma variável *FirebaseDatabase.*

Para realizar a busca de valores no banco de dados foi criada a função *buscaDados().* Esta função realiza a busca no banco de dados e salva os valores em uma variável para, posteriormente, serem exibidos no aplicativo.

A Figura 4 ilustra o fluxograma do aplicativo desenvolvido.

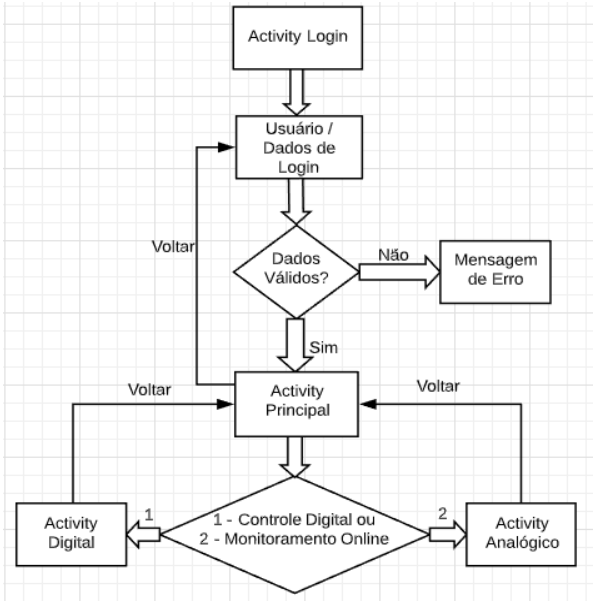

Figura 4: *Fluxograma do Aplicativo*

Activity Login

Nessa *activity* o usuário entrará com os dados de login, e-mail e senha. Os dados serão analisados pelo aplicativo e caso sejam válidos, será inicializada outra *acitivity* (Activity Principal).

Para a implementação desta funcionalidade foi utilizado o Firebase *Authentication*. São disponibilizados vários métodos de login. Foi utilizado o método e-mail/senha. Para isso, no projeto criado, na aba *Authentication* em métodos de login foi ativado o método e-mail/senha. E em usuários foi adicionado um novo usuário, inserindo-se o e-mail e a senha. É possível adicionar vários usuários, porém foi adicionado apenas um.

No aplicativo foram utilizadas as bibliotecas *com.google.firebase.auth.AuthResult* e *com.google.firebase.auth.FirebaseAuth*. Foi criada a função *login(String email, String senha)*, que verifica se os dados de e-mail e senha fornecidos pelo usuário são validos. Caso sejam validos, é aberto uma nova tela. Caso os dados não sejam validos, é exibido uma mensagem de erro ao usuário.

Activity Principal

Nessa *activity* o usuário terá a opção de escolher entre realizar o controle digital da residência (acionamento de eletrodomésticos, lâmpadas, entre outros) e realizar o monitoramento de algumas variáveis do ambiente (temperatura, umidade, entre outras). Para isso, a *activity* possuirá dois botões, que possibilitam ao usuário a opção de escolher qual ação realizar. As duas opções inicializam duas *activitys* distintas.

Activity Digital

Nessa *activity* o usuário irá realizar o controle de lâmpadas, eletrodomésticos, portão eletrônico, entre outros, da residência. Serão adicionados botões que darão ao usuário as opções das ações que o mesmo poderá executar. Dessa forma, sempre que um botão é acionado, há o envio de valores ao banco de dados. Para isso, é inicializada uma variável do tipo *digital*. Utilizando a função *buscaDados()* é feito a busca do valores atuais do banco de dados e salva nesta variável. Quando um botão é acionado, é salvo um novo valor a esta variável e feita a atualização da mesma no banco de dados. Para realizar o envio de dados é utilizado as mesmas bibliotecas que são utilizadas para buscar os dados.

A busca e a atualização de valores no banco de dados acontecem de forma instantânea, não exigindo uma conexão de internet muito veloz. Caso não haja conexão de internet o comando que foi dado no aplicativo fica salvo, e é executado assim que se obtenha o sinal de internet.

Por fim, a função *buscaDados()* também é utilizada para a busca dos valores digitais, para informar o usuário sobre a estado dos dispositivos que o mesmo comanda.

Nesta *activity* o usuário também tem a usuário de realizar o acionamento das cargas do sistema por meio de comando de voz.

Para a implementação desta funcionalidade foi utilizada a biblioteca *android.speech.RecognizerIntent.* Na *activity* Digital foi criado o botão *acionamento por voz.* Quando este botão é acionado, é ativado o microfone do celular e o reconhecimento de voz do Google©. Este comando de voz é capturado e passado como parâmetro para uma função, que converte o comando em uma *string*. Esta *string* com o comando de voz salvo é então utilizado em estruturas de *if*, que determinam as ações do sistema. Na simulação são utilizados três leds que representam o acionamento de um portão eletrônico e duas lâmpadas. Dessa forma foi configurado os seguintes comandos de voz:

Abrir garagem: Realiza a ação de abrir a garagem (acende o led azul);

Fechar garagem: Realiza a ação de fechar a garagem (apaga o led azul);

Acender garagem: Realiza a ação de acender a lâmpada da garagem (acende o led amarelo);

Apagar garagem: Realiza a ação de apagar a lâmpada da garagem (apaga o led amarelo);

Acender varanda: Realiza a ação de acender a lâmpada da varanda (acende o led verde);

Apagar varanda: Realiza a ação de apagar a lâmpada da varanda (apaga o led verde);

Para realizar tais ações, sempre que é dado um comando por voz e o mesmo é reconhecido, há uma atualização nos valores do banco de dados, utilizando os mesmos recursos do acionamento por botões. Quando o comando de voz não é reconhecido, ou é inválido, nada acontece no sistema. O comando é apenas ignorado.

Activity Analógico

Nessa *activity* o usuário visualizará valores de temperatura, umidade, dentre outros, da residência. Dados que serão provenientes de sensores, sendo armazenados no banco de dados. Para tal ação, temos um botão na *activity*, que ao ser acionado realiza a busca dos valores no banco de dados e os exibem na tela para o usuário.

B. Site Web

Para a criação do site Web foi utilizado a IDE Visual Studio Code da Microsoft©. Foi criada apenas uma tela, onde o usuário pode realizar o controle da residência por meio de botões, e visualizar variáveis do ambiente que são atualizadas em tempo real, sempre que o banco de dados sofre alguma alteração.

Da mesma forma que no aplicativo Android, o site Web foi integrado ao projeto criado no Google – Firebase. Na opção *adicionar app* escolheu-se a plataforma Web, e adicionou-se o SDK fornecido ao código do site Web no Visual Studio Code.

No site Web o usuário realiza o controle e o monitoramento do sistema em apenas uma tela. Nessa tela tem-se os botões do controle digital e os elementos de texto que exibem as variáveis do sistema ao usuário.

O controle do sistema é feito através de botões, sempre que se clica em um botão, um valor é atualizado no banco de dados. Para isso, o site Web foi integrado a um projeto no Google –Firebase©, onde está localizado o banco de dados. Criou-se uma variável do tipo *firebase.database()*. Com esta variável é possível acessar o *Realtime Database*, e realizar a atualização dos valores salvos no local.

Para a exibição das variáveis analógicas do sistema criou-se uma lista, do tipo *list.* Utilizando a variável *firebase.database()* criada anteriormente, é percorrido todos os valores analógicos do banco de dados e salvos nesta lista, que é exibida instantaneamente na tela do site. Sempre que ocorre uma mudança nos valores do banco de dados, essa lista é atualizada automaticamente, mantendo sempre os valores atuais disponíveis na visualização.

C. Programação do µC ESP32

Na Figura 5 apresenta-se o fluxograma do algoritmo implementado no µC ESP32.

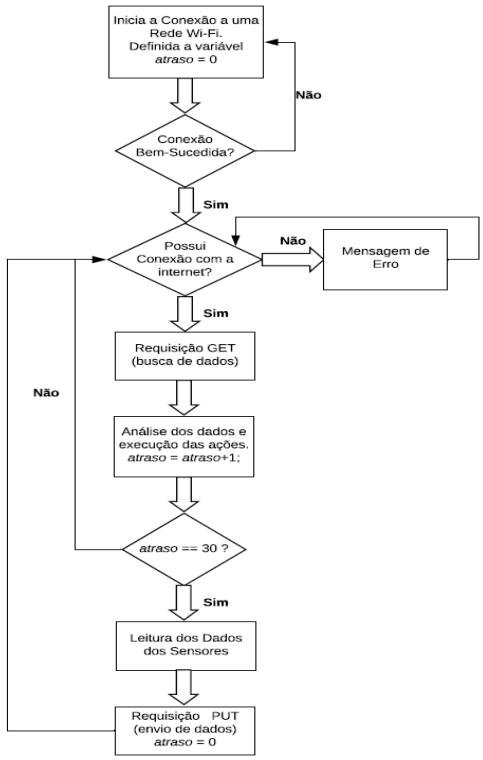

Figura 5: *Fluxograma de Funcionamento do ESP32.*

Pode-se dividir o código em cinco partes: conexão com a internet; busca de dados; tratativa dos dados; leitura dos sensores, e envio de dados, as quais são detalhadas nas seções que seguem.

Conexão com a internet.

Para realizar a conexão do µC com a internet foi utilizada a biblioteca *wifi.h*, e criada uma função em que são passados dois parâmetros: nome e senha da rede Wi-Fi. Em seguida, a função realiza várias tentativas de acesso até que se consiga a conexão com a internet.

Para a verificação de conexão com a internet, na função *void setup()* utilizando a biblioteca *wifi.h*, é passado o comando *WiFi.begin(ssid, password).* Que recebe como parâmetros o nome e a senha da rede Wi-Fi. Logo em seguida, há um laço de repetição *while* que verifica a conexão. O laço de repetição só é encerrado se houver conexão com a internet.

Busca de dados

Realizada a conexão com a internet, se inicia a função *void loop(). ().* Ao iniciar essa função, há a estrutura *if* que verifica novamente se há conexão com a internet, para que se possa executar outras funções.

O µC realiza a busca os valores das variáveis digitais no banco de dados. Para isso utiliza-se a biblioteca *HTTPClient.h*, que é usada para realizar uma requisição GET ao banco de dados. Nessa requisição deve-se informar a URL (*Uniform Resource Locator*) do banco de dados. Após realizar a requisição GET, têm-se como retorno uma *string*, com os dados no formato JSON. Após cada requisição é configurado um delay de 250 milissegundos antes de se iniciar o próximo loop.

Tratativa dos dados

Com os dados no formato JSON não é possível realizar as operações necessárias, sendo necessário interpretar essa *string* em variáveis do tipo *int*, para realizar as comparações que determinaram os valores das saídas digitais do ESP32. Para isso, foi utilizada a biblioteca *ArduinoJson.h*.

A partir das variáveis do tipo *int*, podem ser feitas comparações utilizando a estrutura *if-else*. E de acordo com o resultado dessas comparações as saídas digitais são configuradas em nível alto ou nível baixo.

Leitura dos sensores

Para a simulação do sistema foram utilizados dois sensores DHT11[17], para medição da temperatura e umidade. Para a leitura dos sensores foi utilizado a biblioteca *dht.h*. Esses valores foram armazenados em variáveis do tipo *double*, para serem tratadas posteriormente.

Envio de dados

Os dados são provenientes dos sensores DHT. Para a leitura dos mesmos é utilizada a biblioteca *dht.h*. Após a leitura das variáveis analógicas é necessário enviar ao banco de dados esses valores. Para isso, primeiro utilizou-se a biblioteca *ArduinoJson.h* para criação do objeto do tipo JSON. Em seguida, esse objeto JSON é convertido em uma *string*, que, por fim, é enviada ao banco de dados.

Para o envio de dados foi realizada uma requisição PUT, por meio da biblioteca *HTTPClient.h.*, que recebe como parâmetros: a *string* de dados e a URL do banco de dados.

A requisição PUT é realizada com uma frequência menor do que a requisição GET. A cada 30 requisições GET, é realizada uma requisição PUT. Considerando o tempo das requisições GET e do delay a cada loop, o envio de dados ao banco de dados é realizada em média a cada 1 minuto.

D. Simulação do sistema

Para a verificação do funcionamento do sistema foi construído um protótipo, conforme diagrama de blocos da Figura 6.

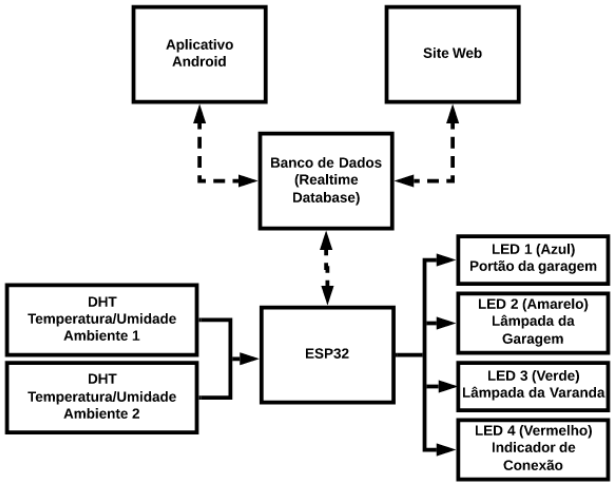

Figura 6: *Simulação do Sistema.*

Foram utilizados quatro diodos emissores de luz (LED, *Light Emissor Diode*) para representar as cargas elétricas ou avisos de conexão, sendo: LED 01 usado para representar o acionamento ou não de um portão eletrônico e, o LED 02 e LED 03 usados para indicar o acionamento de dois pontos distintos de iluminação da residência, por fim LED 04 usado para indicar o estado da conexão do ESP32 com a internet.

Em relação a variáveis analógicas, foram utilizados dois sensores DHT que fornecem valores de temperatura e umidade para dois ambientes distintos.

Além dos elementos citados, foram utilizados também: um celular, com sistema operacional Android©, e um notebook (Windows 10©). Ambos servindo de interface do sistema com usuário por meio do aplicativo Android© e do site Web respectivamente.

IV. RESULTADOS

A. Aplicativo Android

A figura 7 mostra o aplicativo desenvolvido, sendo representadas as telas *activities:* Login, Principal, Digital e Analógico, respectivamente.

A base do aplicativo é a conexão com o projeto criado no Google - Firebase©. Dessa forma, se não há conexão com a internet o aplicativo não se comunica com o Google - Firebase©. Ao tentarmos entrar no aplicativo por meio do login é retornado uma mensagem de erro. Caso o usuário realize o login, e posteriormente em outra tela, a conexão com a internet venha a falhar, o comando dado pelo usuário é memorizado e executado assim que a conexão seja reestabelecida.

A validação do usuário na *activity* Login é realizada de maneira rápida. O mesmo acontece na comunicação com o banco de banco *(Realtime Database).* Porém, apesar da conexão do aplicativo com o Google - Firebase© não exigir uma conexão de internet veloz, variações na velocidade de funcionamento do aplicativo podem acontecer para diferentes redes de internet a qual o mesmo está conectado.

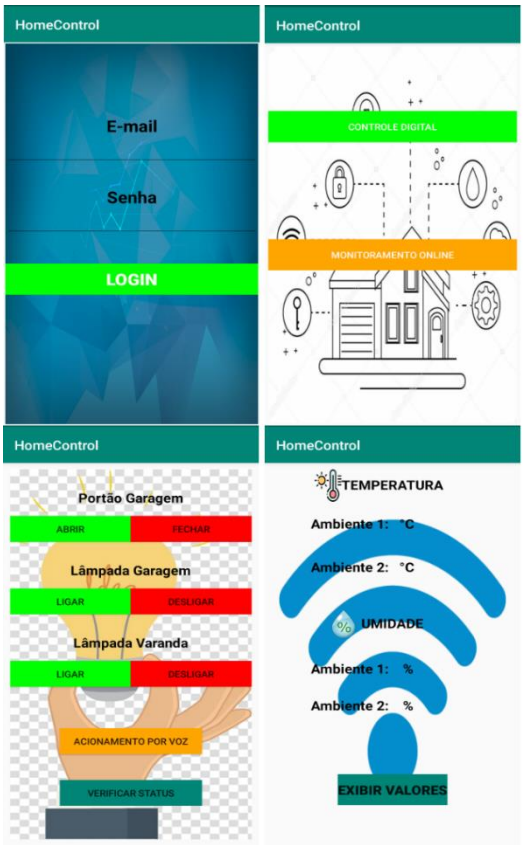

Figura 7: *Exemplo de telas do aplicativo Desenvolvido.*

Outro recurso do aplicativo consistiu na inclusão de comando por voz. Para isso utilizou-se a biblioteca própria do Android/Google. O comando de voz apresentou maior lentidão quando comparado ao acionamento direto por botões. Isso pode ser explicado pelo tempo gasto no reconhecimento da fala e no processamento do comando.

De forma geral, os comandos de voz são facilmente reconhecidos, e foram executados da forma correta. Quando o comando de voz não é reconhecido, o aplicativo não executa nenhuma ação, retornando uma mensagem de erro: *Comando de Voz não Reconhecido*.

B. Site Web

A figura 8 ilustra o layout do Site Web desenvolvido.

Com uma conexão de internet estável, a exemplo do aplicativo Android©, o site Web apresenta funcionamento satisfatório. A comunicação com o banco de dados do Google - Firebase© acontece de forma praticamente instantânea. Na parte de Monitoramento Online, os dados de temperatura e umidade são atualizados automaticamente, sempre que há uma mudança dos valores no banco de dados. O mesmo acontece para as indicações de estados das cargas (ON – OFF).

Figura 8: *Site Web desenvolvido.*

O site Web, assim como o Aplicativo Android©, não necessita de uma conexão de internet veloz, mas tem seu funcionamento influenciado pela mesma.

C. ESP32

Considerando uma rede de internet estável, o µC realiza a conexão com a mesma de maneira rápida. Para realização das requisições GET e PUT, que são usadas na interação com o banco de dados, é exigida uma conexão de internet mais veloz, se comparados ao aplicativo Android© e ao site Web. De fato, quando as requisições são realizadas de maneira lenta, o atraso entre o acionamento de uma carga no aplicativo Android© e/ou site Web e a realização física pode ser muito grande, inviabilizando o sistema de automação proposto.

D. Funcionamento do Sistema

O funcionamento geral do sistema está atrelado à velocidade da rede de internet. Sendo mais afetado pelo funcionamento do ESP32, que é a parte do sistema mais atingida nessa questão, uma vez que o mesmo faz o papel de agente intermediador das ações.

Realizados testes em uma rede de internet com velocidade de aproximadamente 10 Mbps, o atraso entre o acionamento de uma carga no aplicativo Android© e/ou site Web e a realização física foi de cerca de 1 segundo, nos piores casos. Algumas variações no tempo de execução do sistema ocorrem devido a variações no tempo de execução das requisições realizadas pelo µC ESP32. Pois a comunicação entre o banco de dados e aplicativo Android© e/ou site Web se dá de forma instantânea.

V. TRABALHOS FUTUROS

Como citado anteriormente, o funcionamento do sistema é dependente da conexão com a internet. Para trabalhos futuros, sugere-se a implantação de uma rede paralela, como por exemplo, a rede 3g para situações em que a rede Wi-Fi estiver indisponível. Outra opção é configurar o ESP32, para em situações de emergência, funcionar como um servidor, recebendo comandos diretamente do aplicativo Android©.

Com os dados disponíveis no *Realtime Database* é possível gerar um histórico do acionamento de cargas de toda residência, podendo assim, realizar o controle sobre a energia consumida. Dando ao usuário, um relatório completo sobre o consumo de energia de todas as cargas da residência separadamente.

Sugere-se também a inclusão de sensores de verificação de funcionamento, como sensores de luminosidade, para o usuário obter informações sobre o estado das cargas controladas. Além disso a implantação da função de monitoramento por câmeras no sistema, é uma opção bastante vantajosa. Desta forma, o usuário poderá ser capaz de acessar imagens de câmeras, verificando o estado dos acionamentos realizados, e garantindo a segurança de sua residência.

Para o monitoramento por câmeras, é necessária a integração do aplicativo Android e do site Web ao DVR (*Digital Video Recorder*) utilizado na residência. Nesse caso, o fabricante do DVR deve fornecer informações como, por exemplo, o link RTSP e/ou SDK do equipamento para posterior integração no aplicativo e WebSite. Como exemplo, pode-se citar a Intelbras©, que fornece o SDK para auxiliar o desenvolvedor na integração com o DVR [19].

VI. CONCLUSÃO

O trabalho desenvolvido envolveu a integração de diferentes tecnologias, como programação em nuvem, programação de um microcontrolador, desenvolvimento de aplicativo Android©, integração de comandos de voz e de um site Web.

Nesse sistema, todo o armazenamento de dados é feito na nuvem, utilizando o Google – Firebase©. Dessa forma, o aplicativo Android© pode ser instalado em mais de um dispositivo, uma vez que não há a necessidade de armazenamento local. E o site Web também pode ser acessado de qualquer dispositivo. Tornando o sistema acessível de qualquer lugar do mundo, sendo necessário apenas uma conexão estável com a internet.

O microcontrolador utilizado apresenta uma série de tecnologias que facilitam a implementação de projetos de IoT. A principal tecnologia utilizada foi a conexão Wi-Fi, que se mostrou bastante estável durante a simulação do sistema. Além disso, o ESP32 apresentou uma velocidade de execução de tarefas satisfatória para o uso em questão.

Além disso, o sistema é de caráter flexível e expansível, sendo possível realizar a expansão do sistema para controlar e monitorar uma grande quantidade de eventos em uma residência.

Por fim, pode-se destacar o uso do Google – Firebase©. Essa plataforma de desenvolvimento mobile apresenta inúmeros recursos que facilitaram o desenvolvimento deste trabalho. Estão disponíveis várias bibliotecas do Google – Firebase© para a programação Android©, o que torna a integração com um Aplicativo simples e eficiente. Foram utilizados no trabalho os recursos de *Realtime Database* e *Authentication*, que tiveram desempenho satisfatório.

AGRADECIMENTOS

Primeiramente a Deus, por me ter dado sabedoria e perseverança para sempre seguir em frente durante todo esse período.

Aos meus pais Zilmar e Marlene, meu irmão Gilsimar, por todo apoio em todos os momentos da vida, sempre me incentivando a dar meu melhor em todas as coisas da vida.

A todos os amigos que fiz durante este tempo na universidade, em especial aos meus amigos Diego, Mauricio e Matheus meus sinceros agradecimentos pelo bom convívio e aprendizagem.

A todos os professores, que contribuíram de forma grandiosa no meu desenvolvimento técnico e humano, sempre dispostos a compartilhar o conhecimento de forma aberta e sincera.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] EDUCAÇÃO. *O que é internet e as redes de computadores*. Disponível em <https://www.educacao.cc/tecnologica/o-que-e-internet-e-as-redes-de-computadores.html> Acessado em: 26 out. 2019.

[2] VALENTE, Jonas. *Celular se torna principal forma de acesso à internet no Brasil*. AGÊNCIA BRASIL. Disponível em <http://agenciabrasil.ebc.com.br/geral /noticia/2018-07/celular-se-torna-principal-forma-de-acesso-internet-no-brasil> Acessado em: 26 out. 2019.

[3] CIDRAL, Beline. *Afinal, o que é o Android*. Disponível em <https://www.techtudo.com.br/artigos/noticia/2011/01 /afinal-o-que-e-android.html> Acessado em: 26 out. 2019.

[4] JUNIOR, J. A. T, *Controle e Supervisão de Sistema de Automação Utilizando Computação em Nuvem*. Dez 2018. 13p. Trabalho de Conclusão de Curso. Universidade Federal de São João Del Rei (UFSJ), 2018.

[5] LIMA, Davi M. *Um Sistema de Automação Residencial Modular sob Internet das Coisas e Computação Ubíqua*, Universidade Federal do Ceará, 2018, 67p. Disponível em < http://www.repositorio.ufc.br /bitstream/riufc/34506/1/2018_tcc_dmlima.pdf> Acessado em: 12 dez. 2019.

[6] GOMES, A. B. SILVA, G. A. C. GELACKI, R. *Automação Residencial Utilizando Uma Plataforma de Baixo Custo*, Universidade Tecnológica Federal do Paraná, 2016, 44p. Disponível em < http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/8691/1/PG_COAUT_2016_1_04.pdf> Acessado em: 12 dez. 2019.

[7] ATHOSELECTONICS. *ESP32 – Especificações e projetos.* Disponível em <https://athoselectronics.com/esp32/> Acessado em: 26 out. 2019.

[8] MICROSOFT AZURE. *O que é computação em nuvem?*. Disponível em <https://azure.microsoft.com/pt-br/overview/what-is-cloud-computing/> Acessado em: 26 out. 2019.

[9] EVERIT. *Computação em nuvem: como funciona?*. Disponível em <https://www.everit.com.br/guia-da-computacao-em-nuvem/> Acessado em: 26 out. 2019.

[10] ORLANI, Cláudio. *Firebase: serviços, vantagens, quando utilizar e integrações*. Disponível em <https://blog.rocketseat.com.br/firebase/> Acessado em: 26 out. 2019.

[11] FIREBASE. *Google Firebase*. Disponível em <https://firebase.google.com> Acessado em: 27 out. 2019.

[12] SIGNIFICADOS. *O que é Android*. Disponível em <https://www.significados.com.br/android/> Acessado em: 27 out. 2019.

[13] DEVELOPERS. *Conheça o Android Studio*. Disponível em <https://developer.android.com/studio/intro?hl=pt-br> Acessado em: 27 out. 2019.

[14] INTELECTUA. *Você realmente sabe o que é desenvolvimento web?*. Disponível em <https://www.intelectua.com.br/blog/o-que-e-desenvolvimento-web> Acessado em: 27 out. 2019.

[15] SODRE, Eduardo. *Desenvolvimento do Framework Java - Fácil*, Fundação Educacional do Município de Assis – FEMA – Assis, 2011. 44p. Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA. São Paulo, 2011.

[16] IMPACTA. Você sabe o que é Visual Studio?. Disponível em <https://www.impacta.com.br/blog/2017/12/11/voce-sabe-o-que-e-visual-studio/> Acessado em: 26 out. 2019.

[17] DATASHEET. DHT11 Humidity & Temperature Sensor. Disponível em <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf> Acessado em: 26 out. 2019.

[18] GITHUB. espressif/arduino-esp32. Disponível em <https://github.com/espressif/arduino-esp32> Acessado em: 23 set. 2019.

[19] MUNDO API. Você sabe as diferenças entre api e sdk? Disponível em <https://mundoapi.com.br/materias/ voce-sabe-as-diferencas-entre-api-e-sdk/> Acessado em: 12 dez. 2019.

[20] FIGURA 2: LABORATÓRIO DE GARAGEM. Conhecendo o esp32. Disponível em< http://labdegaragem.com/profiles/blogs /tutorial-conhecendo-o-esp32-introdu-o-01> Acessado em: 12 dez. 2019

[21] FIGURA 3: TREINAWEB. Firebase: Descubra o que essa plataforma pode te ajudar. Disponível em < https://www.treinaweb.com.br /blog/firebase-descubra-no-que-esta-plataforma-pode-te-ajudar/> Acessado em: 11 dez. 2019