ESP8266 e Relê de Interrupção: Aplicativo de Monitoramento e Relatório de Consumo Elétrico

Juliano Padoin Vieira[1], Alessandro Mainardi de Oliveira[1]

[1]Sistemas da Informação – Universidade Franciscana (UFN)
Caixa Postal:151 – 97010-032 – Santa Maria – RS – Brasil

julianovieira@ufn.edu.br, alessandroandre@ufn.edu.br

Abstract. *This work proposes a device that will monitor the consumption of electronics connected to it, for this purpose an application for mobile phone with Android system was developed. Monitoring will be done by hour, day, week or month, being possible to see your consumption at any time as long as you have internet connectivity. For assistance and control, the device will have a relay to remotely connect or disconnect any equipment connected to it. For this, an ESP8266 microcontroller with Wi-Fi connection with the MQTT protocol was used. Therefore, the objective of this work permits greater transparency about how much an electronic device consumes. However, as a factor to be considered, a high fluctuation in the electricity consumption reports was seen. This is because it was not possible to accurately measure the data received from the ACS712.*

Resumo. *Este trabalho tem como proposta um dispositivo que irá monitorar o consumo de eletrônicos ligados a ele, para isto foi desenvolvido um aplicativo para celular com sistema Android. O monitoramento será por hora, dia, semana ou mês, sendo possível ver seu consumo a qualquer momento desde que se tenha conectividade com a internet. Para auxílio e controle o dispositivo contará com uma relé para possibilitar remotamente ligar ou desligar qualquer equipamento a ele conectado. Para isto foi utilizado um microcontrolador ESP8266 com conexão Wi-Fi com o protocolo MQTT. Logo, o objetivo desse trabalho permeia uma maior transparência acerca do quanto um eletrônico consome. Porém, como fator a se considerar foi visto uma alta oscilação nos relatórios de consumo elétrico. Isto porque não foi possível medir com precisão os dados recebidos do ACS712.*

1. Introdução

Segundo o site O Globo, cerca de 70% da população brasileira em 2018 tinha internet em suas residências, na zona rural cerca de 49%. Na pesquisa, entre 2017 e 2018, constatou-se que o uso de *Smart* TV's está crescendo enquanto os PC's diminuíram, ganhando em disparado está o uso do celular para acessar a internet [Cetic 2019].

A IoT (*Internet of Things*) em português, Internet das Coisas, foi desenvolvido para uma comunicação assíncrona de dispositivos que possui a finalidade de trocar informações com algum propósito por meio de sensores e atuadores que possam ser controlados remotamente [Lena e Oliveira 2019].

Apesar da extensa gama de utilidades, o princípio da IoT é que para tecnologia ser acessível à população o seu custo não pode ser elevado.

Este trabalho tem a finalidade de desenvolver um *hardware* e um *software* baseando-se em *IoT* para monitorar o consumo de energia elétrica do aparelho que estiver conectado a ele, enviando o relatório para um servidor *online* no qual haverá acesso ao link via aplicativo, mas que poderá ser aberto em qualquer navegador e a opção de ligar ou desligar qualquer equipamento a ele conectado.

1.1 Justificativa

Este projeto tem como base uma maior transparência no quanto eletrônicos ligados no dispositivo estão consumindo em relação a energia elétrica. Para a realização desse monitoramento será utilizado uma relé, um microcontrolador ESP8266 e um medidor de consumo de corrente elétrica.

1.2 Objetivo geral

Este trabalho tem como objetivo criar um dispositivo eletrônico para acionamento remoto de equipamentos com relatório do consumo de energia recebido através de um aplicativo para celulares com sistema operacional Android por um custo acessível.

1.3 Objetivo específico

Este trabalho tem o objetivo específico de:

- Montar o hardware para acionamento remoto e relatório de consumo elétrico.
- Criar um aplicativo para a leitura do consumo de energia.
- Configurar a comunicação Wi-Fi entre hardware e software.
- Utilizar o protocolo MQTT para a comunicação remota.
- Gerar um relatório do consumo de energia gasto.
- Acionar ou desligar o dispositivo remotamente.

2. Referencial Teórico

Para a compreensão do trabalho, serão abordadas características para compreensão da extensão, composição e a forma como foi desenvolvido este projeto.

2.1 Consumo de Energia

A Abesco (Associação Brasileira das Empresas de Serviços de Conservação de Energia) fez uma estimativa que entre 2015 e 2017 foi desperdiçado certa de R$ 61,7 bilhões de Reais em energia elétrica. Entende-se que quanto mais desenvolvido é um País, maior será seu consumo de energia elétrica [Content 2019].

Segundo Souza 2017 apud EIA existe uma previsão de crescimento de 50,6% da demanda de energia entre 2015 e 2024. Para tal crescimento espera-se um avanço na tecnologia e estrutura das redes elétricas no país, nas quais as usinas hidrelétricas que por necessitarem de um grande espaço no leito dos rios causam um grande impacto ambiental e social, e as usinas termelétricas necessitam de um espaço menor, mas tem um preço muito alto para geração de energia além de uma alta poluição, são as maiores fornecedoras atualmente.

2.2 MQTT (*Message Queue Telemetry Transport*)

Criado no final dos anos 90 pela IBM, visando um protocolo de comunicação assíncrona, seu objetivo inicial foi o desenvolvido para comunicação em pipelines de petróleo. Assim como a internet das coisas, o MQTT obteve popularidade pelo fato de não haver necessidade do emissor e receptor estarem ativos ao mesmo tempo para sua comunicação.

Outra característica é o fato de ser um pacote de dados leve, que exige pouca banda de internet, e pode ser configurado com 3 tipos de transmissões, chamados de QoS (*Quality of Service*): o primeiro tipo de transmissão é o QoS0, onde o receptor receberá a mensagem no máximo uma vez; caso esteja *offline* no momento da transmissão perderá a mensagem. O segundo tipo é QoS1, onde a mensagem é recebida por pelo menos uma vez, e por fim, tem-se o QoS2 que receberá a mensagem uma única vez.

Dentre suas outras características tem-se uma lista onde aparecerá todos os receptores *online* no momento, bem como a criptografia TSL (mesma usado na conexão https), utilizando a linguagem JSON ou XML [Yuan 2017].

2.3 Firebase *Realtime Database*

O Firebase tem 2 principais tipos de bancos de dados, o *Cloud Firestore* e o *Realtime Database*, este último será o utilizado para nossa aplicação. O *Realtime Database* é um banco de dados noSQL hospedado na nuvem onde existe a opção de importar e exportar os dados na extensão JSON, quando um aplicativo fica *offline* seus dados continuam a ser salvos localmente (se assim configurado), ao ficar online essas informações são automaticamente mescladas com as informações do bando de dados. Com o recurso *Firebase Authentication* é possível definir que dados cada usuário terá acesso e como irá acessá-los [Database 2020].

Segundo a referência acima, a seguir algumas de suas características:

- Tempo real: Com a sincronização em tempo real quando os dados são alterados, o banco automaticamente atualiza os dispositivos *online* conectados de suas mudanças.
- Acesso por múltiplas plataformas: Pode ser acessado via navegador web, aplicativo móvel Android ou iOS, Unity e microcontroladores sendo possível através de regras de segurança configuradas no Database.
- Banco de dados escalonado: No plano pago é possível criar múltiplos bancos trabalhando em conjunto, cada um com suas regras de acesso, sendo possível um aplicativo acessar todos de uma vez.

2.4 Thingsboard.io

Uma plataforma *free source* que nos permite gerenciar dados armazenados e gerar os relatórios de consumo. É possível instalar em sistemas operacionais como Windows, Ubuntu, CentOS redhat, Raspberry Pi3, sistemas nas nuvens como AWS, DigitalOcean, IBM Cloud, etc...

Foi utilizado a opção “Live demo”, que armazena os dados na nuvem gratuitamente para este trabalho, onde será utilizado o protocolo MQTT e o cliente (no

caso o ESP8266) será do tipo SUBSCRIBE, está informação é necessária na hora de enviar as informações para o servidor, que irá junto com a chave de acesso e o tipo de dado enviado (*string, boolean, double, long* ou no nosso caso *float*).

2.5 ESP8266

Este microcontrolador de 32 bits possui um processador Tensílica L106, utilizando até 20% de sua capacidade total de processamento para *Wi-Fi*. 4mb de memória *flash* com alimentação em 3,3v. Seu consumo é de 20μA em modo *sleep* e 170mA em sua capacidade máxima. E dependendo da variante, pode ter até 17 interfaces GPIO. Interfaces seriais síncronas, SPI, I2C e I2S e assíncronas USART [Oliveira 2017].

Com relação a sua conexão *Wi-Fi* pode funcionar nos modos AP(*Access Point*) , como cliente, *ad hoc*, que dispensa o uso de um roteador em comum para todos os aparelhos, ou seja, vira um roteador para seus vizinhos ou *Wi-Fi Direct* que é a conexão ponto a ponto (*peer-to-peer*).Para além das características acima, este módulo se faz importante, devido ao baixo custo dado seu potencial, e o fato de poder acoplar o próximo módulo a ser citado [Oliveira 2017]. Na Figura 1 tem-se a placa de desenvolvimento NodeMcu com o microcontrolador Esp8266 acoplado.

Figura 1. NodeMCU com ESP8266

2.6 ACS712

O ACS712 é um pequeno sensor de corrente invasivo capaz de medir a corrente alternada, que está sendo utilizada entre 0 e +30A (há outros modelos no mercado que medem até 5A ou 20A) ou contínua entre -30 e +30A. Afim de se ter uma explicação fidedigna, cita-se Oliveira (2017), no qual, explica este módulo “O sensor ACS712 tem como saída uma interface analógica, de 0 a 5 volts, na qual cada 66 mV corresponde a 1 A, a partir de 2,5 volts. Assim, 0 A corresponde a 2,5 volts, 30 A é representado por 4,48 volts e -30 A corresponde a 0,52 volt. Como a entrada analógica do ESP8266 só consegue ler até 3,3 volts, é necessário fazer um divisor de tensão com dois resistores para reduzir a tensão máxima a 3,3 volts” [Oliveira 2017]. Na Figura 2 é visualizado o sensor de corrente.

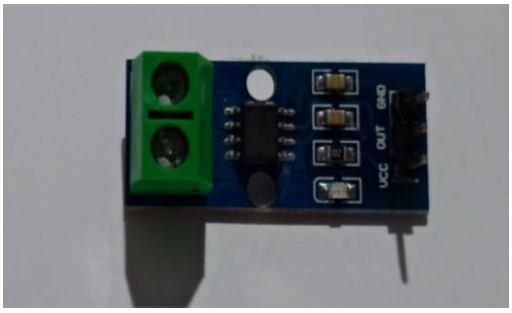


Figura 2. Sensor de corrente invasivo ASC712

A seguir é apresentado o Gráfico 1 onde é exemplificado a amperagem pela tensão.

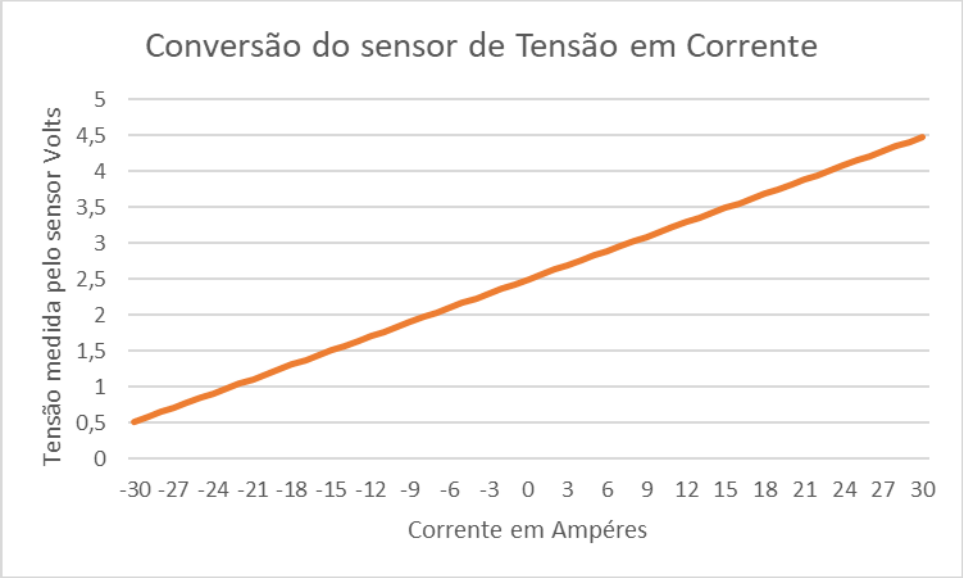


Gráfico 1 – Conversão do valor de Tensão em Corrente

Como a entrada analógica do ESP8266 só consegue ler até 3,3 volts, é necessário fazer um divisor de tensão com dois resistores para reduzir a tensão máxima a 3,3 volts [Oliveira 2017].

Como a tensão máxima é 4,48 volts, é possível usar dois resistores para reduzir esse valor para 3,3 volts. Como sugestão, é possível citar os resistores 5,6 K e 10 K, cuja soma seria 15,6 K. Assim, o valor de 4,48 volts seria reduzido para 3,3 K com um erro de 0,4%, desprezível para esse tipo de sensor [Oliveira 2017]. A Figura 3 ilustra o divisor de tensão.

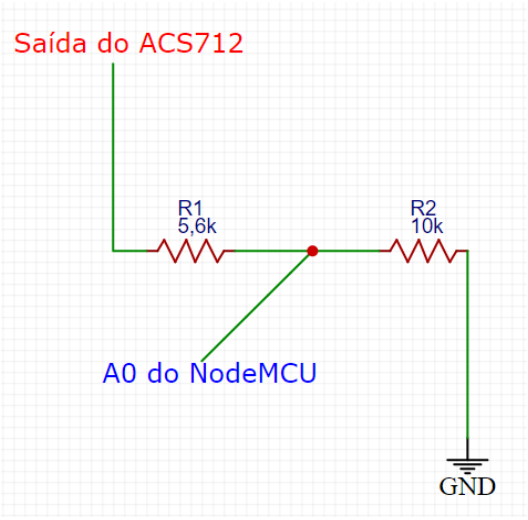


Figura 3. Esquema da ligação da saída do ACS712 e entrada no NodeMCU

2.7 Relé

As GPIO (*General Purpose Input/Output*) do NodeMCU, comumente chamadas de entradas e saídas digitais trabalham com voltagem máximas de 3.3V, o que pode acionar apenas LEDs ou dispositivos de baixíssima corrente, para eletrônicos de maior potência utiliza-se relés que podem ser acionadas na voltagem do NodeMCU de 3.3V, neste dispositivo tem-se as ligações da parte do Arduino: GND(terra), VCC(neutro) e IN(entrada digital), e na outra ponta as entradas: NC (normalmente fechada), COM (comum, entrada da corrente) e NO (normalmente aberta) [Thomsen 2013].

Na Figura 4 é exibido o exemplo de uma relé para acionamento de dispositivos eletrônicos com capacidade de 10 amperes, 250 volts e acionada por uma carga de 3,3V a 5V.

Figura 4. Relé para acionamento de dispositivos eletrônicos.

2.8 Android Studio

Em 2018, 98% das pessoas com acesso à internet conectavam com o celular, tendo isto em vista, para o relatório de consumo de energia e o controle de liga/desliga do dispositivo será utilizado o Android Studio [Cetic 2019].

Oficialmente a ferramenta padrão para quem deseja produzir aplicativos para Android, conta com recursos como emulador com diversas versões do Android, integração com o GtiHub, ferramentas de testes, ferramentas para detectar problemas de

compatibilidade entre versões do Android e compatibilidade com C++ [Developer 2020].

2.9 Feature Driven Development (FDD)

A metodologia escolhida foi o desenvolvimento guiado por funcionalidades, ou FDD, por ser um método simples e focado em funcionalidades como o próprio nome já diz. Ebulon (2014) baseia-se em 2 fases, primeiramente tem-se a concepção e planejamento e a seguir o desenvolvimento e execução, e 5 funcionalidades, explicadas a seguir:

Desenvolver um modelo abrangente: Nessa etapa inicial é quando a equipe conhecerá o produto que será desenvolvido, no caso o aplicativo terá 2 telas, de *login* e a principal.

Construção da lista de funcionalidades: Depois de conhecer o produto, começa a lista de seus requisitos, suas funcionalidades. A seguir é apresentado a lista de requisitos funcionais:

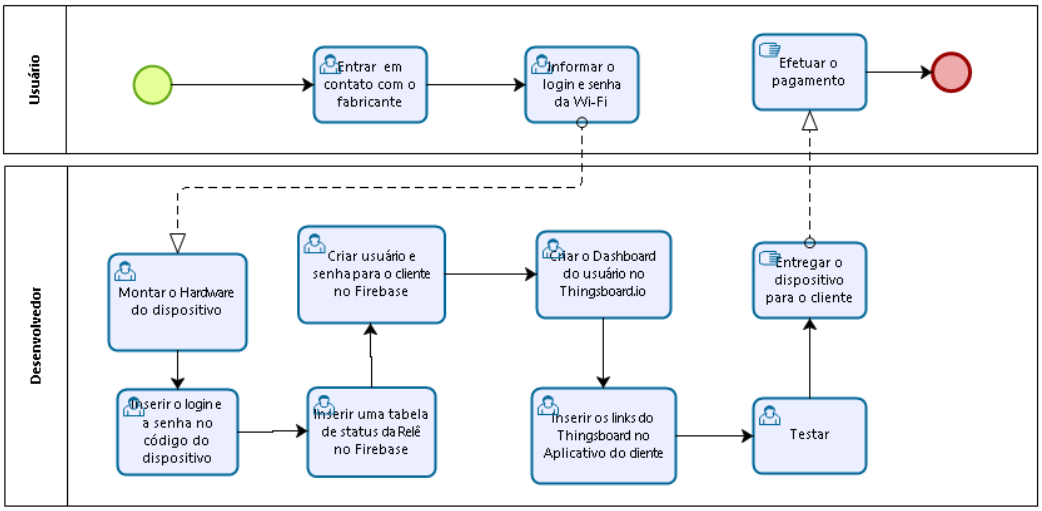
- RF01: Tela de *Login* – A aplicação deve permitir o acesso do usuário as funcionalidades do aplicativo.
- RF02: Visualizar opções: A aplicação apresenta as opções de relatórios de consumo da hora, dia, semana ou mês, assim como terá a opção de acionar ou desligar a relé.
- RF03: Gerar relatório – A aplicação irá redirecionar o usuário a um link com um gráfico do consumo de energia elétrica consumido pelo respectivo tempo escolhido.
- RF04: Gerenciar Relé – A aplicação deve permitir que o usuário ligue ou desligue o eletrônico ligado ao dispositivo.
- RF05: Gerenciar Relé – Comunicação contínua com o *Database Realtime* para atualização do status da relé.

E os requisitos não funcionais:

- RNF01: Android Studio – A aplicação será desenvolvida no Android Studio para versão 4.4 e superior.
- RFN02: Firebase – A aplicação irá conectar no servidor do site Firebase para gerenciar o acionamento da relé.
- RNF03: *Thingsboard* – O dispositivo enviará o consumo elétrico para o respectivo site para poder ser gerado o relatório de consumo.

Planejar por funcionalidade: Nessa etapa no qual já se tem pelo menos uma parte do que seja criado, é hora de começar a planejar e priorizar seus requisitos e prazos.

No fluxograma 1 é demonstrado a modelagem de processos com seu fluxo de execução.



Fluxograma 1. Modelagem de processos

Desenho por funcionalidade: Nesta etapa é determinado quais requisitos tem prioridade sobre os outros, é comum ter nesta etapa diagrama de classes, esboços, o que puder ser útil para a equipe entender o que será desenvolvido.

Construção por funcionalidade: É nesta etapa que o software começa a ser desenvolvido.

3. Metodologia

A seguir, a metodologia usada para o desenvolvimento do *hardware* e *software*.

3.1 *Hardware*

A Figura 5 mostra o esquema eletrônico do sistema implementado, onde pode se observar os componentes utilizados nesta implementação.

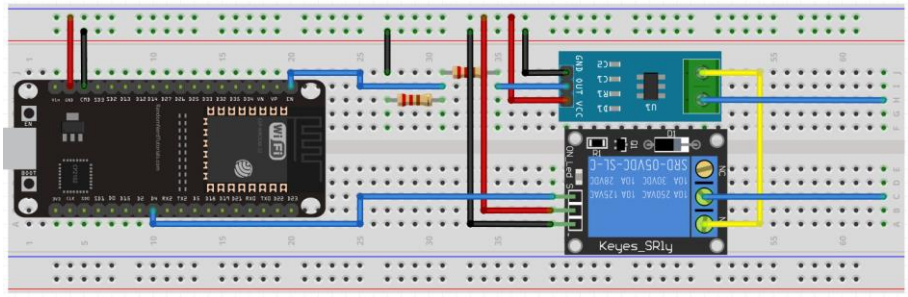


Figura 5. Esquema do circuito

3.2 O processo de funcionamento

O trabalho desenvolvido funciona da seguinte maneira, o cliente irá solicitar o dispositivo, o desenvolvedor irá montar, solicitar o nome da rede e senha *Wi-Fi* do cliente, configurar no código do Arduino, então o vendedor irá cadastrar um usuário e senha para o cliente na base de dados existente no *Realtime Database* como mostra a Figura 6.

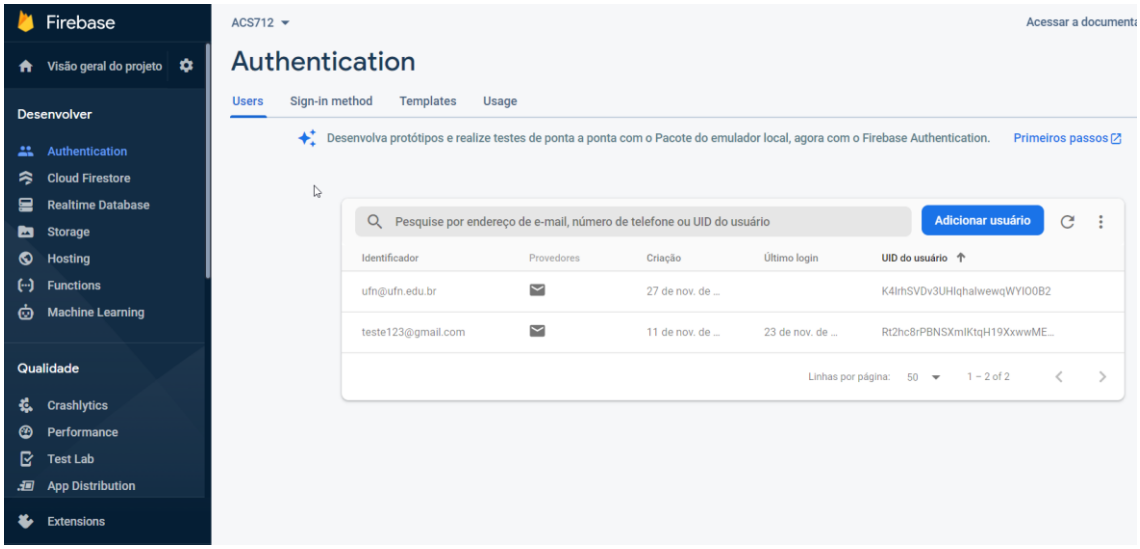


Figura 6. Tela de criação da autenticação para o aplicativo

Então será criado um campo do tipo *string* na base de dados para receber o status do dispositivo, conforme mostrado na Figura 7.

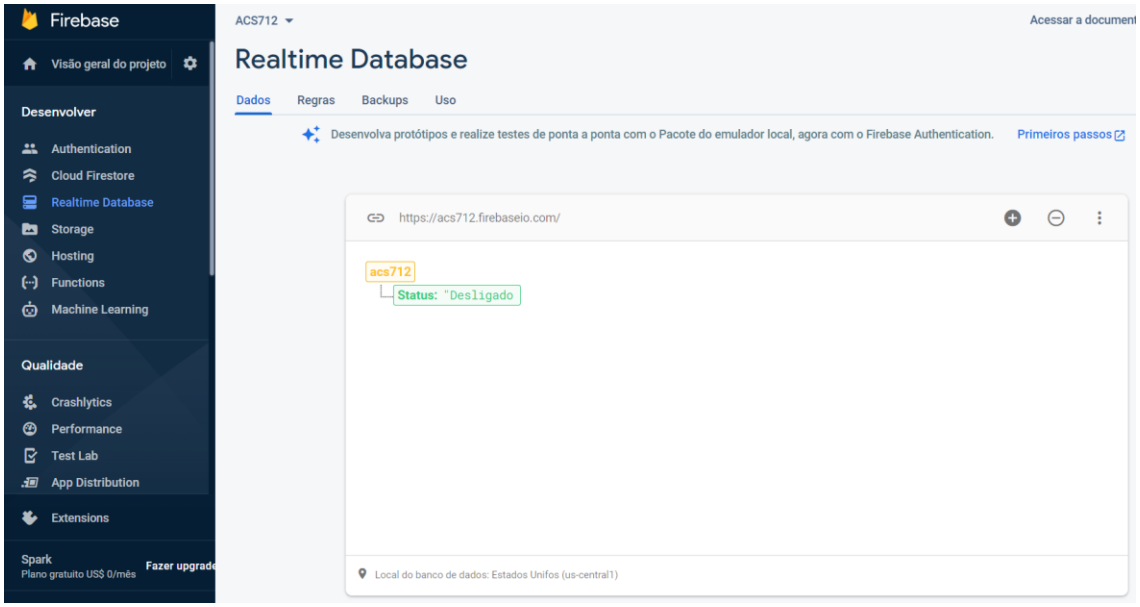


Figura 7. Cadastro do campo de status da relé

Assim como os comandos de ligar e desligar, será criado no *Thingsboard* os 4 *dashboards* com os respectivos tempos (hora, dia, semana e mês) como pode ser observado na Figura 8.

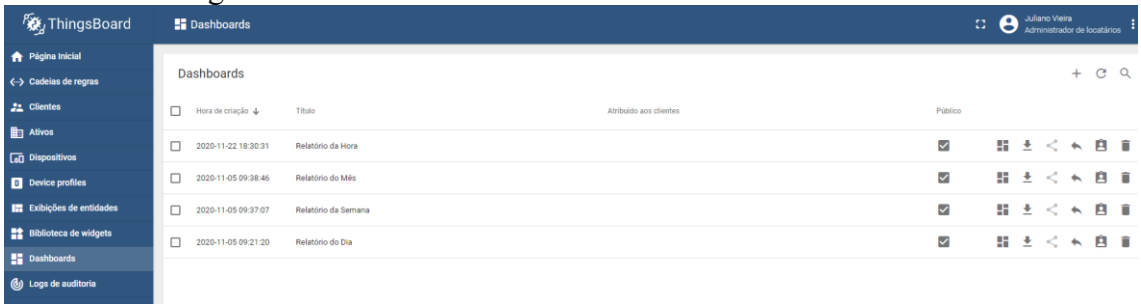


Figura 8. Cadastro dos *dashboards* para os relatórios.

4. Resultados

A Figura 9 mostra a imagem do protótipo, onde na esquerda pode ser observado os componentes soldados na placa e na direita o protótipo na caixa.

Figura 9. Protótipo.

4.1 *Software*

A seguir serão apresentadas as ferramentas para o desenvolvimento do projeto com algumas de suas principais configurações.

4.1.1 IDE Arduino

Para trabalhar com o Arduino é necessária uma ferramenta para a compilação de seu código, para isto foi utilizada a oficial do Arduino chamada IDE Arduino que além de leve e prática, supre as necessidades deste projeto. Para programar na IDE Arduino, as boas práticas nos sugerem começar com as bibliotecas necessárias e logo em seguida as constantes, como a imagem a seguir:

```
1  //Bibliotecas
2  #include "FirebaseESP8266.h"
3  #include "ThingsBoard.h"
4  #include "EmonLib.h"
5  #include <ESP8266WiFi.h>
6  //Variáveis constantes
7  #define FIREBASE_HOST "acs712.firebaseio.com"
8  #define FIREBASE_TOKEN "Chave de acesso do Firebase"
9  #define THINGSBOARD_HOST "demo.thingsboard.io"
10 #define THINGSBOARD_TOKEN "Chave de acesso do Thingsboard"
11 #define WIFI_SSID "Rede Wi-Fi"
12 #define WIFI_PASSWORD "Senha da Wi-Fi"
```

Figura 10. Primeiras linhas do código do Arduíno

Na inclusão das bibliotecas tem-se a do Firebase (desenvolvido por Mobizt versão 3.0.3) e do *ThingsBoard* (desenvolvido por forGGe na versão 0.5) para a comunicação de envio, leitura e/ou atualização de dados com seus respectivos servidores. A EmonLib (desenvolvido por OpenEnergyMonitor versão 1.1.0) faz os cálculos do consumo de energia com base na voltagem (no caso 220V) e na amperagem no eletrônico ligado, e

por último o ESP8266WiFi (desenvolvido por Arduino versão 1.2.7) que é o responsável por conectar o ESP8266 ao roteador.

Depois das bibliotecas pode ser observada algumas variáveis constantes com o endereço de seus servidores e seus respectivos *tokens* para poder enviar e/ou receber informações e por último a configuração da rede Wi-Fi, informando a rede a qual irá se conectar e a respectiva senha, conforme a Figura 10 acima.

4.1.2 Android Studio

Para o desenvolvimento do aplicativo para smartphone foi utilizado o Android Studio pela sua gama de recursos e facilidade de conexão com o *Realtime Database* do Firebase, para isto é necessário incluir algumas dependências como mostra na Figura 11.

```
implementation 'com.google.firebase:firebase-auth:20.0.1'
implementation 'com.google.firebase:firebase-core:18.0.0'
implementation 'com.google.firebase:firebase-database:19.5.1'
```

Figura 11. Dependências necessárias para conexão com o Firebase

O código de verificação de *login* é mostrado na Figura 12. No mAuth é criado uma instância e enviado para o servidor com o e-mail e senha informados nos campos de login e feita uma verificação, em caso positivo, abre a tela de acesso aos relatórios e controle da relé e aparece a mensagem "Autenticado com sucesso.", caso o login ou a senha sejam inválidos ou nulos surge a mensagem "Autenticação falhou." e o usuário tem a opção de continuar nessa tela ou sair clicando no botão "Fechar":

```
private void login(String email, String password) {

    mAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    Log.d(TAG,  msg: "signInWithEmail:success");
                    FirebaseUser user = mAuth.getCurrentUser();
                    Toast.makeText( context: LoginActivity.this,  text: "Autenticado com sucesso."
                                    + user.getEmail(), Toast.LENGTH_SHORT).show();
                    startActivity(new Intent( packageContext: LoginActivity.this,
                            ProfileActivity.class));
                    finish();
                } else {
                    Log.w(TAG,  msg: "signInWithEmail:failure", task.getException());
                    Toast.makeText( context: LoginActivity.this,  text: "Autenticação falhou.",
                            Toast.LENGTH_SHORT).show();
                }
            }
        });
}
```

Figura 12. Código de validação de *Login*

4.2 Tela de *Login*

A Figura 13 mostra a primeira tela, a de *login*, onde o usuário irá inserir seu e-mail e senha pré-cadastrados.

Figura 13. Tela de *Login*

Depois de logado, aparece a tela principal do programa, na Figura 14, é possível observar o controle da relé, ligando ou desligando os equipamentos e o controle aos relatórios do consumo de energia por hora, dia, semana ou mês:

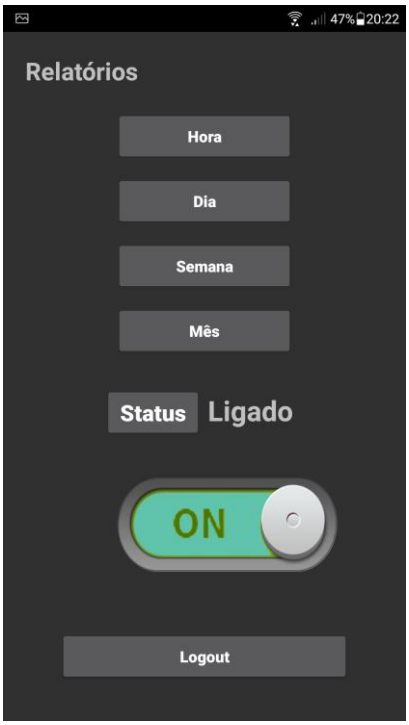


Figura 14. Tela principal

Ao clicar em qualquer botão referente ao relatório será redirecionado ao navegador do *smartphone* que levará ao respectivo gráfico no site demo.thingsboard.io

como mostra a Figura 15. Porém logo depois de ligar o dispositivo, o relatório mostra um pico no valor da corrente, que não é um valor real, que será tratado nas próximas versões.

Figura 15. Tela com o relatório gerado da última hora

4.3 Dados imprecisos

Durante o projeto para comparação foi utilizado um Arduino Uno, foram testados os mesmos sensores, apenas não foram utilizados os resistores já que o Arduino Uno trabalha com voltagem de 5V nas portas analógicas. E nossos testes tiveram uma melhor precisão, já com o ESP8266 não foi possível medir o consumo de energia elétrica com confiabilidade, testou-se várias formulas, as variáveis foram modificadas, se conectou fios de puro cobre (CAT6) e mesmo assim, quando media o consumo perto do esperado em um dispositivo, ao conectar outro aparelho com consumo de energia diferente mostrava um consumo totalmente diferente da realidade, sendo assim teve-se uma acuracidade nos resultados tornando os relatórios insatisfatórios. Talvez com um sensor não invasivo como o 100A SCT-013, se obteria um melhor resultado para o projeto em questão.

5. Conclusão

No desenvolvimento deste trabalho foi apresentado uma solução de baixo custo para automação residencial, com um sensor de consumo de energia elétrica é possível um maior controle sobre nossos gastos, através de relatórios do consumo de energia. Para

isso foi utilizado a leitura corrente com o sensor ASC712, porém seus dados não são precisos.

Além disso é mostrado a implementação de um sistema com relé para o ligar e desligar os utensílios eletrônicos ligado neste dispositivo através da comunicação Wi-Fi, utilizando o protocolo MQTT para a comunicação remota.

Como o projeto tem um foco em baixo custo, foi utilizado servidores online e gratuitos, com o intuito do aprendizado, mas para um melhor controle em implementação comercial é necessário utilizar servidores pagos.

Algumas ideias para trabalhos futuros visando este projeto seriam caso o microcontrolador não consiga acessar a rede Wi-Fi se transforma-se em um *Access Point* para ser possível configurá-lo através de um aplicativo para se conectar em uma nova rede. Implementar um botão de ligar e desligar para o controle manual, caso perca o acesso ao dispositivo. Criação de uma "régua" como se normalmente utiliza para ligar o computador, onde cada tomada seja alimentada por uma relé e conectada a um sensor de corrente elétrica, assim seria possível vários dispositivos, cada um sendo monitorado por um sensor.

6. Referências

Cetic (2019); TIC DOMILICIOS, <http://data.cetic.br/cetic/explore?idPesquisa=TIC_DOM>. Acesso em 22 Maio, 2020.

Content (2019); O ciclo da energia e seus desperdícios, <https://exame.abril.com.br/geral/o-ciclo-da-energia-e-seus-desperdicios>.

Costa, G. R. D (2018); Monitoramento de variáveis ambientais para um data center, Universidade Estadual Paulista "Júlio de Mesquita Filho", Sorocaba - São Paulo.

Database, Firebase (2020); Firebase Realtime Database, <https://firebase.google.com/docs/database>. Acesso em: 23 Nov, 2020

Acesso em: 15 Abril, 2020.

Developers, Android (2020); Conheça o Android Studio, <https://developer.android.com/studio/intro>. Acesso em 17 Mai. 2020.

Ebulon (2014). Agile Software Development using Feature Driven Development (FDD), <http://www.nebulon.com/fdd/index.html>, Acesso em: 27 Abril, 2020.

Lena, Fábio Quoos; Oliveira, Alessandro Mainardi de, (2017) Sistema Distribuído de Automação Residencial. 2019, 15f, Trabalho Final de Graduação (Monografia), Universidade Franciscana, Santa Maria – Rio Grande do Sul.

Oliveira, S. (2017) Internet das coisas com ESP8266, ARDUINO e RASPBERRY PI pi, 1ª. Ed. São Paulo, Novatec.

Oliveira, Bruno A. S; Assis, Servílio S.; Nolli, Carlos R. (2018); Desenvolvimento de um protótipo de sistema de monitoramento de energia elétrica via internet, 2018, 14f, programa de pós-graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte.

Ribas, Gabriel Azzulin; Oliveira, Alessandro Mainardi de. (2017); Uso do Manual de Boas Práticas em uma Aplicação Android para Avaliação de Eventos de Massa, 2017, Trabalho de Conclusão de Curso, Centro Universitário Franciscano - Santa Maria.

Souza, Vinicius de. (2017); Ferramenta computacional de geração de energia elétrica em residências que integram geração fotovoltaica e banco de baterias, 2017, 54f, Trabalho de conclusão de curso, Universidade Tecnológica Federal do Paraná, Cornélio Procópio - Paraná.

Silva, L. L. D. (2018); Sistema de monitoramento de consumo de energia elétrica residencial com a utilização do protocolo MQTT. 2018, 82f, Trabalho de Conclusão de Curso, Universidade Federal de Santa Catarina - Araranguá.

Torres, Andrei B. B; Rocha, Atslands R.; Souza, José Neuman de (2016); Análise de Desempenho de Brokers MQTT em Sistema de Baixo Custo, 2016 12f, Trabalho Final de Graduação (Monografia), Universidade Federal do Ceará - Fortaleza.

Thomsen, A. (2013). Controlando lâmpadas com Módulo Relé Arduino. Disponível em: <https://www.filipeflop.com/blog/controle-modulo-rele-arduino>. Acesso em 08 Dez. 2020.

Yuan, Michael (2017), “Conhecendo o MQTT”, <https://developer.ibm.com/br/articles/iot-mqtt-why-good-for-iot>. Acesso em: 14 abr, 2020.