INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
PRÓ-REITORIA DE PESQUISA, INOVAÇÃO E PÓS-GRADUAÇÃO

Resultado Final - Edital nº 30/2021 - PIBIC/FAPESQ

CLAS.	PROJETO	COORDENADOR DE PROJETO	PROD.	PROJETO	PONTOS	SITUAÇÃO
1º	TRATAMENTO DE FERIDA ABERTA EM CÃES COM O USO DE CURATIVO BIOLOGICO COM PELE DE TILÁPIA	Ana Lucelia de Araujo	34,60	190,00	224,60	Aprovado com Bolsa
2º	Avaliação da presença HPV em tumores de cérvice de mulheres do nordeste do Brasil	Maria Angelica Ramos da Silva	18,60	186,00	204,60	Aprovado com Bolsa
3º	Uma solução baseada em Internet of the Things (IoT) para gerenciamento da iluminação pública	Ramon Leonn Victor Medeiros	18,55	160,00	178,55	Aprovado com Bolsa

João Pessoa, 04 de setembro de 2021.

FRANCISCO DANTAS NOBRE NETO:06459621470

Assinado de forma digital por FRANCISCO DANTAS NOBRE NETO:06459621470
Dados: 2021.09.04 15:29:39 -03'00'

Francisco Dantas Nobre Neto
Diretor de Pesquisa

RELATÓRIO FINAL

IFPB - Campus João Pessoa

Identificação do Programa: PIBIC/FAPESQ – Edital nº 30/2021

Período abrangido pelo relatório (mês/ano): setembro/2021 a agosto/2022

Título do projeto: Uma solução baseada em Internet of the Things (IoT) para gerenciamento da iluminação pública.

Grande Área / Área: *(De acordo com Tabela da CAPES/CNPq) CIÊNCIA DA COMPUTAÇÃO (CIÊNCIAS EXATAS E DA TERRA)*

Endereço: Av. Primeiro de Maio, 720 - Jaguaribe
Telefone: (83) 3612-1200
Bolsista: Luan Gomes de Carvalho
Telefone: (83) 98800-7770
E-mail: luan.gomes@academico.ifpb.edu.br
Voluntário: Matheus Jabes Lyra Duarte de Lima
Telefone: (83) 99661-1632
E-mail: matheus.jabes@academico.ifpb.edu.br
Coordenador do projeto: Ramon Leonn Victor Medeiros
Telefone: (83) 99658-3413
E-mail: ramon.medeiros@ifpb.edu.br

RESUMO

O relatório final apresenta as atividades desenvolvidas, referentes ao projeto Uma solução baseada em *Internet of the Things (IoT)* para gerenciamento da iluminação pública, no período de setembro/2021 a agosto/2022. Dentro deste intervalo foram realizadas pesquisas a respeito de componentes e módulos eletrônicos para integrar o projeto, prototipagem de circuitos e elaboração de modelos para a realização da montagem do protótipo e código que viriam a servir como base para a conclusão do projeto. Enquanto ainda não se tinha decidido o hardware que viria de fato a ser utilizado, foram feitos diversos experimentos em simuladores virtuais a fim de ter uma melhor noção do hardware que melhor se alinharia com o projeto. A visita a SEINFRA em conjunto com as simulações realizadas, serviram como base para se constatar as reais necessidades do gerenciamento da iluminação pública, este contato com os funcionários e equipamentos que de fato eram usados na iluminação pública foi um marco para a consolidação do projeto, pois só assim os pesquisadores saíram do campo teórico e constataram o que seria mais conveniente para o projeto, evitando a implementação de funções desnecessárias que só viriam a aumentar a complexidade do projeto sem agregar benefícios significativos ao projeto e foram capazes de desenvolver algo mais próximo da demanda real necessária. Foram adquiridos os componentes que se cogitava serem os mais próximos do que seria usado na versão final do projeto, onde obtivemos duas ESPs32, módulos relés, fontes HI-LINK de 220v CA - 5v CC, módulos *LDR*, e módulos sensores de tensão. A princípio o microcontrolador escolhido foi o ESP32 porém, inicialmente não se havia o conhecimento necessário

para elaborar códigos que pudessem ser utilizados nele, portanto se seguiram as pesquisas utilizando o Arduino como microcontrolador, todavia com o progresso nos estudos, o ESP não se mostrou mais um problema e ele retornou a ser usado como microcontrolador titular do projeto. Desde o início do projeto foram elaborados diversos protótipos, primeiramente usando apenas programas como o tinkercad, em seguida foram usados material físico, como os microcontroladores, leds de 5mm, cabos jumpers e protoboard, até que finalmente foi possível a elaboração de protótipos mais complexos, onde estes por sua vez, sendo conectados diretamente à rede elétrica residencial, acendendo lâmpadas fluorescentes convencionais. Os estudos a respeito da conexão se iniciaram conectando um ESP32 na rede *Wi-Fi* onde fomos capazes de controlar o ESP de maneira remota acedendo e desligando uma lâmpada via página HTML. A partir daí os esforços se concentraram para realizar conexões através da rede *Mesh*. Após isso se analisou a possibilidade de realizar uma conexão por um meio mais simples, que posteriormente concluiu-se não ser o ideal, porém se mostrou uma ferramenta que pode vir a ser útil posteriormente. A conexão via rede *Mesh* foi um sucesso, porém nem todos os objetivos deste projeto foram alcançados, em virtude da complexidade de programação e da arquitetura dos protocolos. Todavia, ressalta-se que pela importância da base tecnológica e pelo impacto na gerência pública, um novo projeto foi aprovado pelo CNPq para dar continuidade a este. Nosso intuito a partir de agora será implementar o *MQTT*. O projeto SISTEMA DE INFORMAÇÃO PARA MONITORAMENTO DA ILUMINAÇÃO PÚBLICA, que foi aprovado através do Edital n° 22/2022 - Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação – PIBITI/CNPq está dando continuidade a este atual que chegou ao fim.

Palavras Chave: ESP32, *Mesh*, Conexão

1. APRESENTAÇÃO

1.1 INTRODUÇÃO

Ao lado do constante crescimento da infraestrutura das cidades, crescem também os problemas trazidos por falta de controle humano dirigido ao gerenciamento direto de tais recursos. A exemplo disso temos centenas e, até, milhares de postes de iluminação pública nas cidades. Infraestrutura de iluminação pública que, segundo a Resolução Normativa ANEEL nº 414, de 9 de setembro de 2010, em seu artigo 2018, é de propriedade e, consequentemente, está sob responsabilidade das prefeituras de cada município.

Dentre os vários problemas resultantes desta transferência de propriedade pode-se citar a cobrança de taxa extra e contingente limitado de trabalhadores e meios para o suporte à infraestrutura de iluminação pública. Somando a isto, há uma série de problemas envolvendo dispositivos voltados ao

usufruto público com defeitos de funcionamento, causando problemas inerentes ao desperdício de energia e segurança das vias por falta de iluminação.

Desta forma surgem questionamentos como: Seria possível a aplicação de conceitos de Internet da Coisas (*Internet of the Things - IoT*) para o desenvolvimento de um sistema para monitoramento de redes de iluminação pública? Qual seria a arquitetura de rede de sensores sem fio que apresenta um bom custo-benefício de implantação? É possível o desenvolvimento de um sistema de baixo custo?

A necessidade de ter sob controle as intempéries características de sistemas elétricos é uma constante, porém, acaba deixada de lado pelos gestores públicos devido ao alto custo operacional usual de soluções desse tipo. Assim, para uma solução ser efetiva, deve-se aliar o melhor custo-benefício possível, alinhando um orçamento possível e acessível a uma capacidade de resposta de qualidade. Incluindo o fato de a base do sistema ser um dispositivo embarcado, cujo a operabilidade requer pouca ou nenhuma mudança estrutural.

1.2 JUSTIFICATIVA

Visando isso, a aplicação de sistemas de monitoração a distância viabiliza o uso de soluções mais eficientes, a exemplo de uma rede composta por um conjunto de dispositivos de sistema embarcado ligada a uma central operada pelo setor responsável pela manutenção, que recebe os dados de funcionamento de cada poste e é capaz de disparar alertas de mal funcionamento, junto à sua localização para facilitar e agilizar a identificação de problemas.

Dispomos hoje de uma vasta gama de microcontroladores de baixo custo de obtenção, consumo e manutenção, capazes de realizar tais monitoramentos e ainda serem interligados com comunicadores sem fio, facilmente aplicáveis para esse fim se utilizando dos preceitos de *hardware* e *software open-source*. É possível facilitar o acesso remoto para tornar ainda mais eficaz a autonomia operacional do projeto, permitindo o acesso dos profissionais de manutenção de campo aos logs gerados pelo dispositivo em tempo real onde quer que estejam.

A partir do desenvolvimento deste, há possibilidade de desenvolvimento de outros equipamentos, a medição de outros tipos de parâmetros, como velocidade de rotação de motores, fluxo de gás e outros fluidos em tubulações, temperaturas entre outras aplicações para atender os mais diversos nichos de usuários.

1.3 OBJETIVOS

Objetivo Geral

Desenvolver um sistema para gerenciamento remoto em tempo real da iluminação pública aplicando conceitos de *Internet of the Things (IoT)*.

Objetivos Específicos

- Selecionar tecnologias de *hardware* e *software* a serem utilizadas no projeto;
- Montar o *hardware* do sistema embarcado a ser utilizado;
- Definir protocolo de rede de comunicação a ser utilizado;
- Desenvolver o *software* do sistema embarcado;
- Avaliar em ambiente e configuração de teste solução proposta.

2. DESENVOLVIMENTO

2.1 METODOLOGIA

A primeira atividade na execução do projeto consiste em realizar um levantamento sobre *hardwares* utilizados atualmente no desenvolvimento de soluções *IoT*. Serão escolhidos plataforma de prototipagem/microcontrolador, sensores, atuadores, fontes de alimentação, e afins. Durante esta etapa é preciso alinhar o potencial computacional com o custo de cada componente.

O *hardware* escolhido terá impacto direto com a linguagem de programação a ser utilizada e o sistema de gerenciamento *web* a ser escolhido. Variáveis que devem ser consideradas nesse processo.

Como atividade seguinte se apresenta a montagem do sistema embarcado. Neste momento é construído placas de circuito e, assim, a ligação física dos componentes. Bem como a verificação de suas capacidades operacionais.

A definição do protocolo de rede de comunicação a ser utilizado vai influenciar diretamente na topologia de rede aplicada ao sistema. Os estudos sobre este protocolo são paralelos à escolha do *hardware*. Bem como sua aplicação será em paralelo a outras atividades como escolha da plataforma *web* de gerenciamento e o desenvolvimento do *software* embarcado.

Para escolher a plataforma web de gerenciamento dos dispositivos embarcados serão considerados alguns aspectos como *hardware* escolhidos; protocolos de redes escolhidos; facilidade de conexão e reconexão para os nós (*nodes*); níveis de segurança aplicados; políticas de privacidade de dados; entre outros.

A atividade de desenvolvimento do *software* a ser embarcado nos dispositivos apenas poderá ser concluída depois da escolha da plataforma *web* de gerenciamento e do protocolo de comunicação. Durante todo o desenvolvimento do *software*, assim como no desenvolvimento do *hardware*, será dada prioridade a utilização de ferramentas e bibliotecas *open-source*.

Esta etapa de desenvolvimento deverá respeitar padrões de projeto de *software*, documentação e requisitos de segurança.

Por fim, o sistema será avaliado em um ambiente o mais próximo possível da aplicação real. Simulando o tipo de luminária, a distância entre nós (*nodes*) e a corrente elétrica disponível na rede de iluminação pública, por exemplo.

3. RESULTADOS E DISCUSSÃO

Alguns dos resultados do projeto:

- Realizada escolha de componentes eletrônicos e microcontrolador;
- Realizada escrita de programa para monitoramento de ponto de iluminação;
- Realizada conexão do sistema com a Internet;
- Realizado monitoramento via *IFTTT* (*If This Then That*);

Iniciamos o projeto adquirindo e aprimorando o conhecimento necessário para realizá-lo, durante os primeiros meses do projeto realizamos o minicurso de Arduino, que acabou sendo o pilar que estruturou todo o desenrolar do projeto, servindo de base até mesmo para a programação do ESP32, em paralelo com o minicurso, estudados e aprofundamos nosso conhecimento no funcionamento de componentes eletrônicos, assim nos capacitando elaborar e executar o que viesse a ser necessário para conclusão do protótipo.

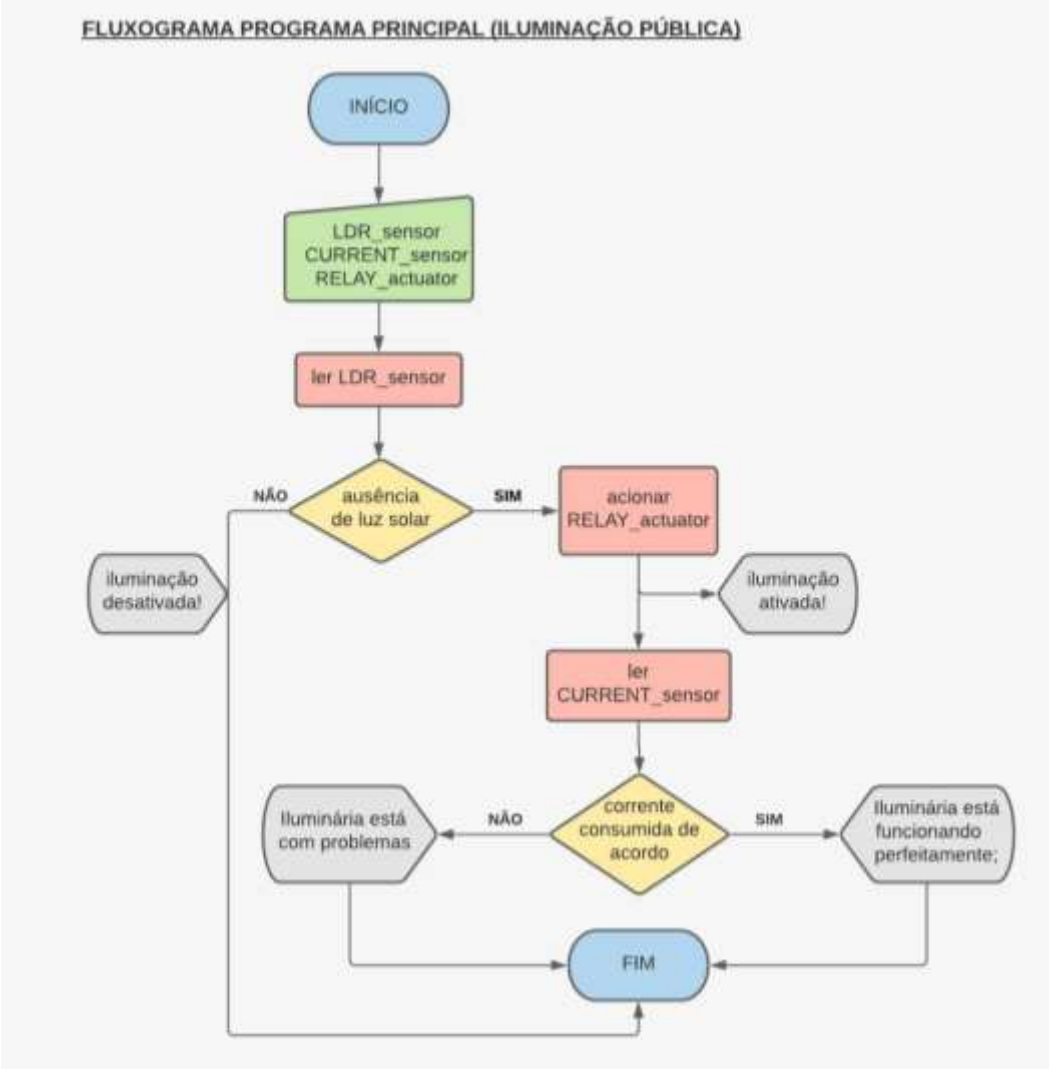


Figura 1: Fluxograma do programa principal

No dia 4 de outubro elaboramos o primeiro protótipo para o projeto via *Thinkercad*, com a adição de novos componentes, o *Thinkercad* não possuía mais as ferramentas necessárias para o avanço do protótipo, se tornando assim obsoleto.

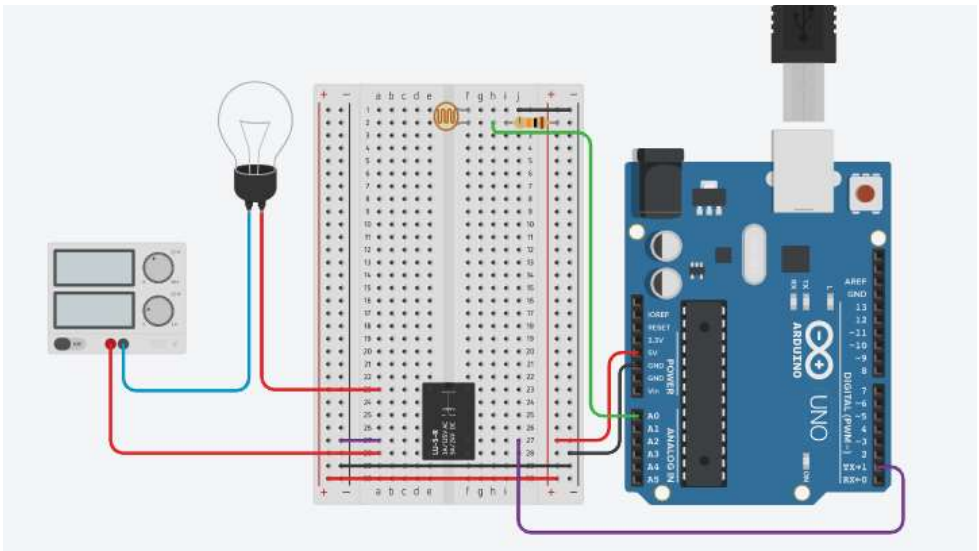


Figura 2: Diagrama no Thinkerkad

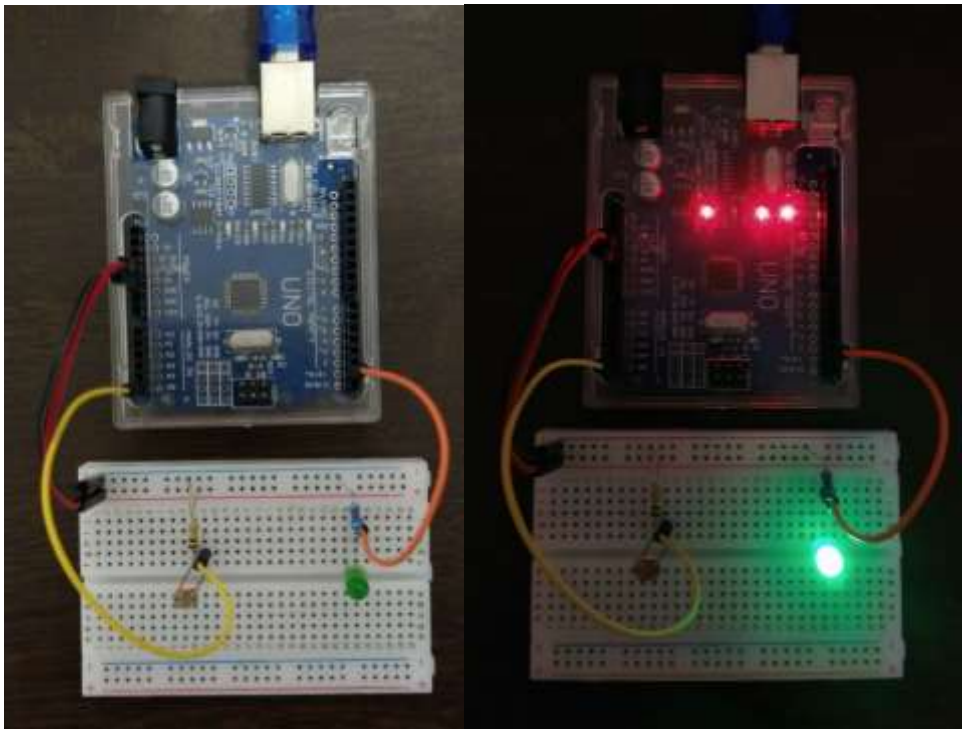


Figura 3: Primeiro protótipo

Visando utilizar novos recursos, migramos do *Thinkercad* para o *Fritzing*.

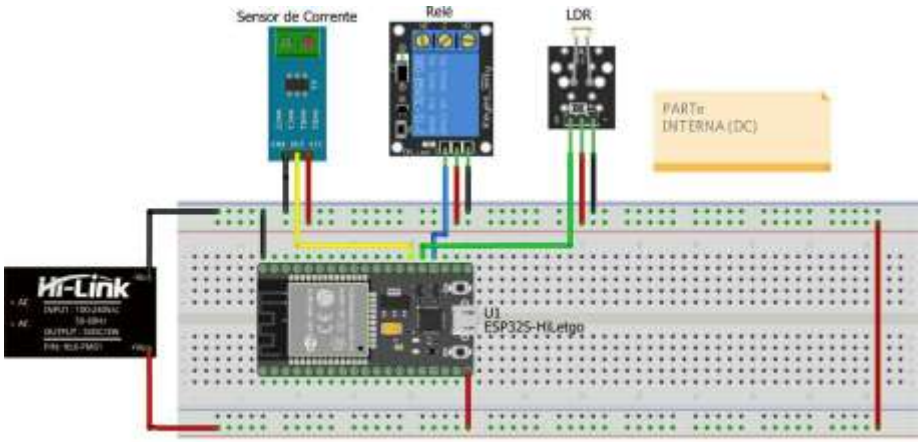


Figura 4: Protótipo no Fritzing

A fim de dar um novo passo decidimos adquirir componentes que viriam a se tornar definitivos ou o mais próximo do que será usado no fim do projeto, recebemos no dia 1 de março de 2022, a elaboração do circuito e programa estava finalizado e operando no dia 8 de abril de 2022.

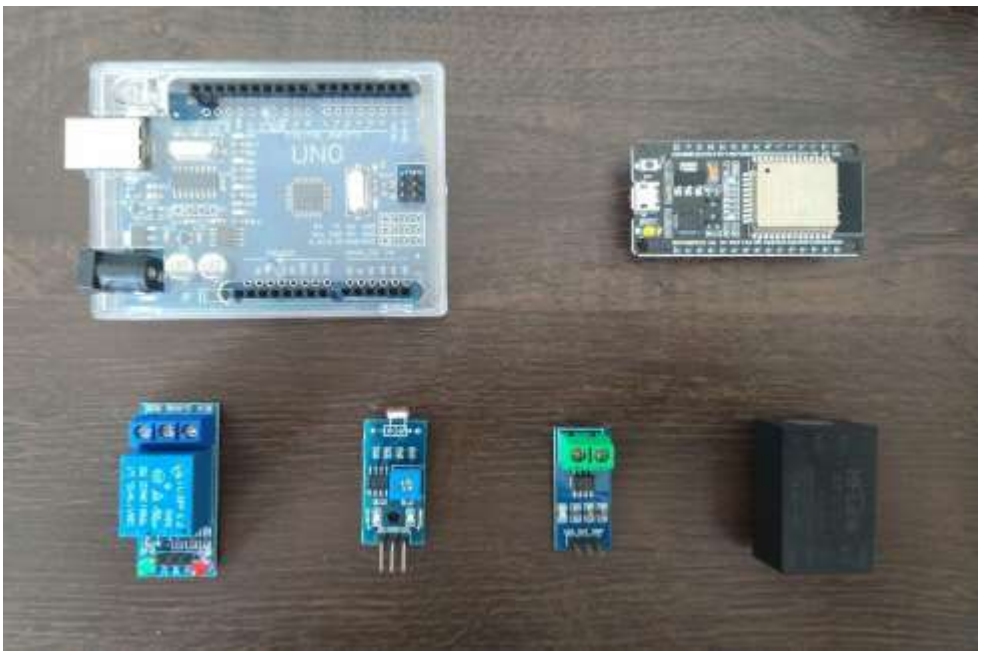
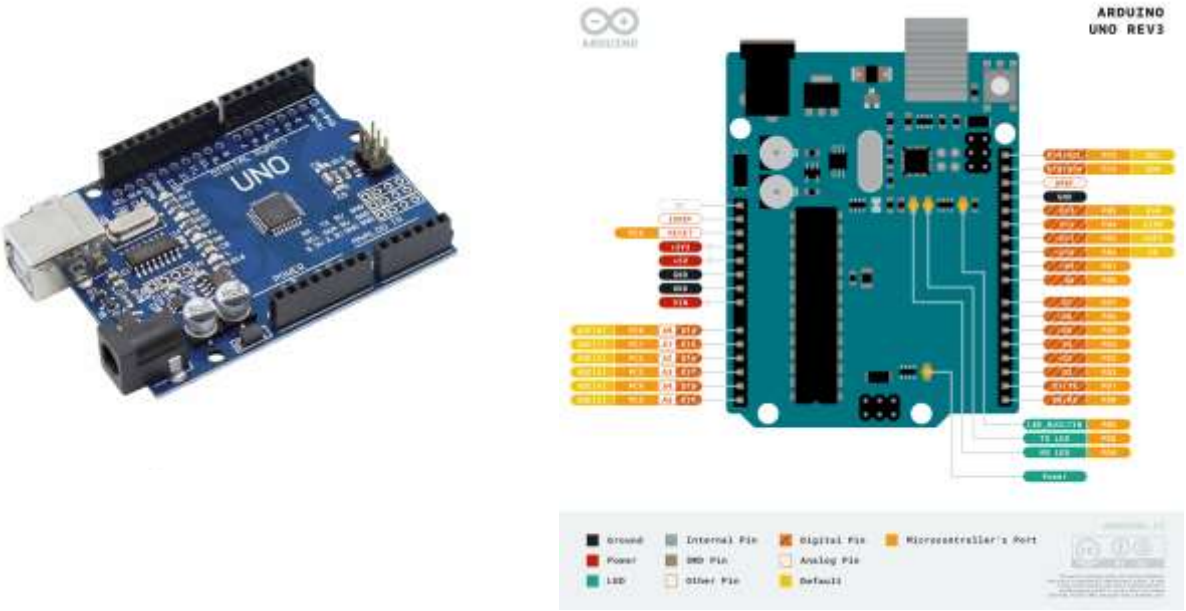


Figura 5: Componentes adquiridos

A princípio o ESP32 seria usado como microcontrolador, porém, diante da carência de informações a respeito da programação utilizando o ESP, optamos por manter os estudos com o Arduino.



(a) *(b)*

Figura 6: (a) Arduino uno R3 / (b) Detalhes internos e pinagem

Em reunião, foram decididos os componentes necessários para prototipação do dispositivo idealizado (o núcleo do sistema de controle e monitoramento da iluminação pública), ressaltando-se o microcontrolador ESP32 sob um módulo acoplado a uma placa de desenvolvimento (porta USB acessível, regulador de tensão, *crystal*).

A escrita nativa de códigos para o ESP32 se dá através da linguagem C ou C++. Somente através delas conseguimos explorar os recursos necessários para o dispositivo idealizado, visto que o Arduino IDE não promove integração com o restante das C *libraries* desenvolvidas pela *Espressif*.

Dada a instalação do driver, bem como a instalação do IDF, partiu-se para a escrita do programa que seria posteriormente carregado no microcontrolador.

Os desafios começaram desde a instalação do IDF no S.O, até o estudo da programação e estruturas necessárias para a escrita de um programa no SoC.

No Windows 10 64-bit, tivemos um problema acerca da configuração da variável PATH do sistema. Houve certa dificuldade de se proceder com comandos no cmd, visto nosso hábito de CLI estar voltado para bash. Foi necessário, frente a essas dificuldades, migrar para um sistema operacional do tipo GNU/Linux. Escolhemos o Ubuntu em sua versão 20.4. TLS. Através do guia de instalação na documentação oficial pela *Espressif*, conseguimos com sucesso fazer a instalação do IDF, bem como configurar precisamente o PATH.

O IDF se utiliza de algumas ferramentas como *Make*, *Ninja* para fazer todo o processo de compilação e *build* dos códigos. Há um arquivo *Makefile* que "seta" algumas configurações (penso que essas configurações estão dadas pelo *menuconfig* dentro do IDF, mas ainda não consegui descobrir).

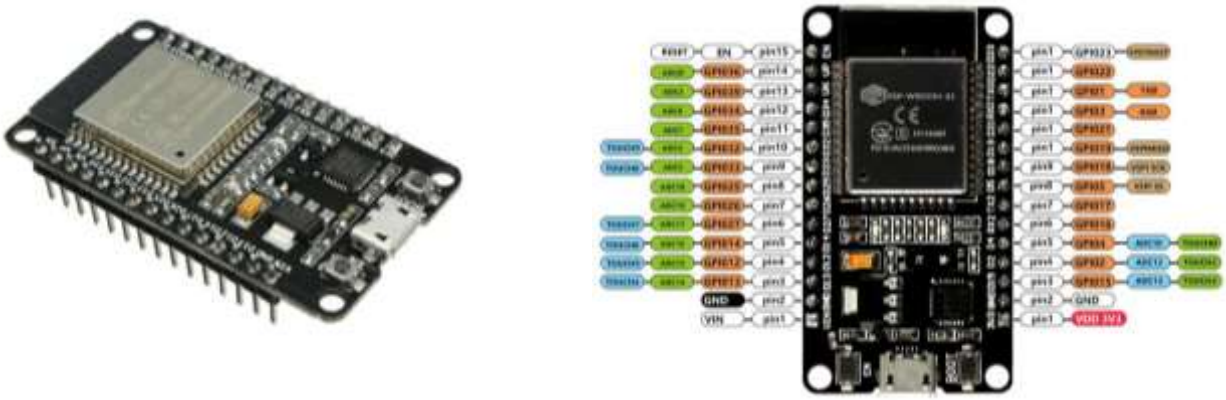
O IDF não conta com um editor de texto integrado, sendo necessário o uso de algum editor por fora, logo, utilizamos o VIM. Conseguimos fazer alguns códigos básicos para testes, mas nada perto do que se pretende para o nosso projeto.

Para fazer um bom código, precisamos entender as funções específicas criadas pelas C *libraries* também específicas. A descrição de várias funções pode ser vista através da documentação oficial, porém, nem todas são claras, gerando dúvidas no momento da implementação.

O desconhecimento acerca de estruturas do IDF, bem como das C *libraries* criadas pela empresa *Espressif* têm sido a maior dificuldade encontrada até agora, nos impossibilitando de minimamente fazer o que seria a nossa primeira meta.

Perante a impossibilidade (até o momento) de se prosseguir com o ESP32, usamos o microcontrolador ATMEGA328. Tal chip atrelado à programação via Arduino IDE promove uma escrita de código trivial com um bom custo-benefício.

Apesar das dificuldades com o ESP32 não o abandonamos do projeto, seguimos estudando-o em paralelo, devido a semelhança com o Arduino, à medida que aprimoramos nosso conhecimento, solucionamos os problemas encontrados no ESP.



(a) (b)

Figura 7: (a) ESP32 / (b) Detalhes internos e pinagem

Com vista no futuro, e de forma a melhorar entendimento do protótipo, fizemos um diagrama esquemático. Este diagrama será usado como base para construção de um protótipo em *Printed Circuit Board (PCB)*.

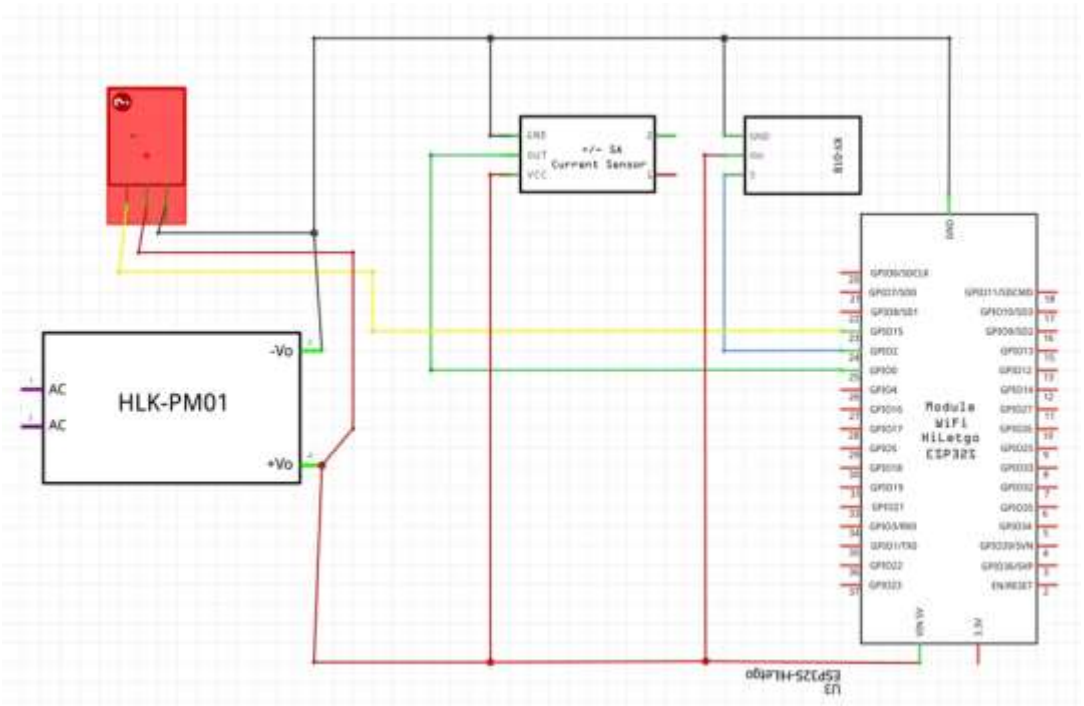


Figura 8: Diagrama esquemático protótipo

Figura 9: Protótipo em funcionamento

No primeiro protótipo conectado diretamente a corrente alternada de 220v utilizamos um arduino uno R3, fonte *Hi-Link*, módulo relé, módulo LDR, e um sensor de corrente elétrica. O sensor de corrente forneceu dados confiáveis, porém a fim de obter dados mais precisos, completos e confiáveis, analisamos a possibilidade de implantar também um sensor de tensão elétrica. Idealmente, pensou-se também na aquisição de um módulo acelerômetro para monitorar se o poste sofreu algum desvio de ângulo com relação ao solo, todavia a fim de dar prosseguimento ao projeto a aquisição não se consolidou, nem do sensor de tensão elétrica e nem do acelerômetro.

Tendo em vista a realização do protótipo, decidiu-se concentrar o foco na construção do código que faria a conexão entre os microcontroladores ESP32.

Após extensa pesquisa e testes diante do limitado Arduino Uno R3, concluímos que a melhor alternativa, de fato, seria retornar ao microcontrolador ESP32. O fracasso ante a programação do ESP32 através do ESP-IDF nos fez buscar por alternativas. Escolhemos trabalhar com o ESP32 através da Arduino IDE.

Realizamos a primeira conexão do ESP32 com o *Wi-Fi*, onde este funcionou como um servidor HTTP dentro da rede local, motivo pelo qual conseguimos controlar o ponto de iluminação via *Web*.

```
#include <WiFi.h>

const char* rede = "YUMI"; //Nome da rede
const char* senha = "28150314"; //Senha da rede
int LED = 2;

WiFiServer server(80);
```

```
void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT);

  Serial.println();
  Serial.print("Conectando-se a ");
  Serial.println(rede);
  WiFi.begin(rede, senha);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi conectada.");
  Serial.println("Endereço de IP: ");
  Serial.println(WiFi.localIP());

  server.begin();
}

void loop() {
  WiFiClient client = server.available();
  if (client) {
    Serial.println("Novo Cliente.");
    String currentLine = "";
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (c == '\n') {
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();
            client.print("Click <a href=\"/H\">aqui</a> para ligar o led.<br>");
            client.print("Click <a href=\"/L\">here</a> para desligar o led.<br>");
            client.println();
            break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
          currentLine += c;
        }
        if (currentLine.endsWith("GET /H")) {
          digitalWrite(LED, HIGH);
        }
        if (currentLine.endsWith("GET /L")) {
          digitalWrite(LED, LOW);
        }
      }
    }
    client.stop();
    Serial.println("Cliente Desconectado.");
  }
}
```

Tabela 1: Código do servidor HTTP (ESP32) sobre Wi-Fi c/ conexão à Internet

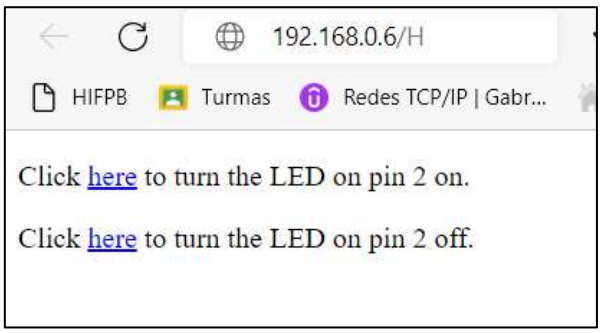


Figura 12: Página web simples, apenas em HTML, gerada pelo servidor HTTP (ESP32)

Após isso, iniciamos estudos a fim de conectar dois microcontroladores ESP32 entre si.

Obtemos êxito em conectar os dois ESP32. Esta conexão se deu via *Wi-Fi* e fez uso da biblioteca não-ofical "*Painlessmesh*" em seu código. Ressalta-se que nessa prática os dois dispositivos funcionaram como *access point*s e *clients* simultaneamente. Essa é uma característica de funcionamento desses dispositivos sobre o paradigma de conexão *Mesh*.

Em termos de comunicação wireless, normalmente, as redes apresentam dois modos distintos: *Access Point mode* (*AP*), *Infrastructure mode* (*Ad-Hoc*). No primeiro modo, os *wireless hosts* somente se comunicam com outros *hosts* através de um *access point*, que se responsabiliza pelo encaminhamento dos pacotes.

Já na segunda forma, os *wireless hosts* dispensam o acesso ao *access point*, comunicando-se diretamente com outro *host*, porém, as redes *ad-hoc* não possuem a capacidade inerente para *multi-hop*, i.e, se um dispositivo A puder alcançar um dispositivo B e o dispositivo B puder alcançar o dispositivo C, mas o A não puder alcançar o C, então A e C não poderão se comunicar, pois o B não retransmitirá nenhum pacote de A para C. Com isso, verifica-se que não é objetivo do *Ad-Hoc* que A e C se comuniquem por meio do B.

Nesse âmbito, o roteamento *Mesh* fornece a facilidade *multi-hop* que o modo *Ad-Hoc* não possui, permitindo que cada *Host* em uma rede, denominado nó, atue como um roteador e transmita pacotes em nome de quaisquer outros *Hosts*.

A conexão *Mesh* ocorre na camada de rede do modelo *OSI* (*Open System Interconnect*). Com isso, *Wireless Mesh Networks (WMNs*) resultam da combinação de arquiteturas sem-fio *Ad-Hoc*, primeira camada, com o roteamento *Mesh*, pertencente à terceira camada do modelo *OSI*

Selecionou-se a tecnologia *Mesh*, pois a microcontrolador ESP32 escolhido no projeto possui suporte a uma implementação nativa de rede *Mesh* pela *Espressif*. Além do mais, pela distância dos pontos de iluminação serem próximos ao alcance pelo *Wi-Fi* 2.4Ghz, de forma que os sinais não sofram colisão. Bem como, pela maior facilidade de gerência por ser de escopo local e pela largura de banda alta para transmissão de dados.

Fomos capazes de realizar a conexão de rede *Mesh* não apenas entre os ESP32, mas também com um ESP8266, o que demonstrou não apenas a possibilidade de utilizar uma conexão entre

microcontroladores da *Espressif* diferentes, mas também a reutilização do mesmo código de programação da rede.

```
#include "painlessMesh.h"
#define   MESH_PREFIX    "xxxxx"; //Nome da rede
#define   MESH_PASSWORD   "xxxxx"; //Senha da rede
#define   MESH_PORT      5555

Scheduler userScheduler;
painlessMesh  mesh;

void sendMessage() ;

Task taskSendMessage( TASK_SECOND * 1 , TASK_FOREVER, &sendMessage );

void sendMessage() {

  String msg = "Olá do nó";
  msg += mesh.getNodeId();
  mesh.sendBroadcast( msg );
  taskSendMessage.setInterval( random( TASK_SECOND * 3, TASK_SECOND * 3 ));
}

void receivedCallback( uint32_t from, String &msg ) {
  Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}

void newConnectionCallback(uint32_t nodeId) {
    Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
}

void changedConnectionCallback() {
  Serial.printf("Changed connections\n");
}

void nodeTimeAdjustedCallback(int32_t offset) {
    Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(),offset);
}

void setup() {
  Serial.begin(115200);

//mesh.setDebugMsgTypes( ERROR | MESH_STATUS | CONNECTION | SYNC | COMMUNICATION | GENERAL | MSG_TYPES |
REMOTE );
  mesh.setDebugMsgTypes( ERROR | STARTUP );

  mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );
  mesh.onReceive(&receivedCallback);
  mesh.onNewConnection(&newConnectionCallback);
  mesh.onChangedConnections(&changedConnectionCallback);
  mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);

  userScheduler.addTask( taskSendMessage );
  taskSendMessage.enable();
}
void loop() {
  mesh.update();
}
```

Tabela 2: Código da rede mesh sobre Wi-Fi (uso da lib. painlessmesh)

Do ponto de vista do monitoramento, utilizou-se o *IFTTT*, abreviação de "*If This Then That*", como recomendado pelo orientador do projeto. A ideia foi fazer um dos protótipos dentro da rede *Mesh*

enviar um *e-mail* com o status de leitura de um dos seus respectivos sensores. Todavia, pelos menos de forma inicial, não foi possível realizar integração com a rede *Mesh*, tornando o teste do *IFTTT* restrito a um dispositivo fora da rede *Mesh*.

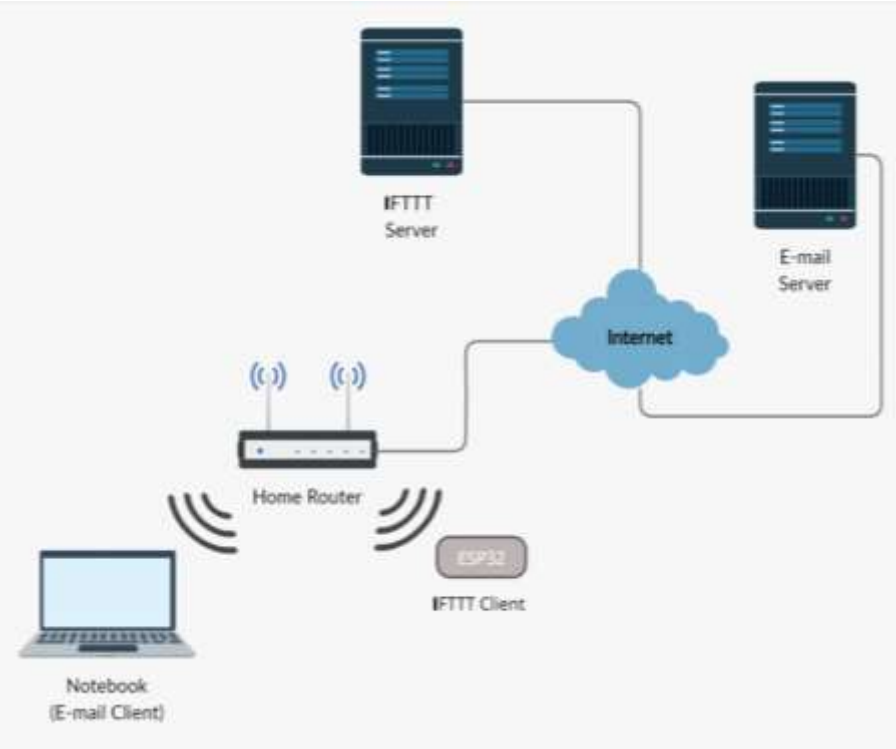


Figura 13: Estrutura de rede do IFTTT

```
#include <WiFi.h>
#include <SPI.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

const char* rede = "xxxxx"; //Nome da rede
const char* senha = "xxxxx"; //Senha da rede
int LED = 2;

WiFiServer server(80);

WiFiClient client;

int   HTTP_PORT   = 80;
String HTTP_METHOD = "GET";
char   HOST_NAME[] = "maker.ifttt.com";
String  PATH_NAME    = "https://maker.ifttt.com/trigger/send-email/with/key/zm1IjwCNuYEV0pYXs34to"; // mudar chave do Webhooks
String queryString = "?value1=27";

void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT);

  Serial.println();
  Serial.print("Conectando-se a ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi conectada.");
  Serial.println("Endereço de IP: ");
  Serial.println(WiFi.localIP());

  server.begin();

  if (client.connect(HOST_NAME, HTTP_PORT)) {
    Serial.println("Conectado ao Server");
    client.println("GET " + PATH_NAME + queryString + " HTTP/1.1");
    client.println("Host: " + String(HOST_NAME));
    client.println("Connection: close");
    client.println(); // end HTTP header

    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.print(c);
      }
    }

    client.stop();
    Serial.println();
    Serial.println("Desconectado");
  }
    Serial.println("Falha na Conexão");
  }
}

void loop() {
  WiFiClient client = server.available();
  if (client) {
    Serial.println("Novo Cliente.");
    String currentLine = "";
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (c == '\n') {
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();
            client.print("Click <a href=\"/H\">aqui</a> para ligar o led.<br>");
            client.print("Click <a href=\"/L\">here</a> para desligar o led.<br>");
            client.println();
            break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
          currentLine += c;
        }
        if (currentLine.endsWith("GET /H")) {
          digitalWrite(LED, HIGH);
        }
        if (currentLine.endsWith("GET /L")) {
          digitalWrite(LED, LOW);
        }
      }
```

```
    }
    client.stop();
    Serial.println("Cliente Desconectado.");
  }
}
```

Tabela 3: Código para uso do IFTTT

Neste caso, ao ser ligado, o ESP32 enviou uma mensagem de status da temperatura do sensor a um *e-mail* configurado. A ideia futura é que o *IFTTT* seja usado de forma a enviar *e-mails* com informações de caráter crítico sobre o dispositivo, visto que servidores de *e-mail* possuem outro objetivo.

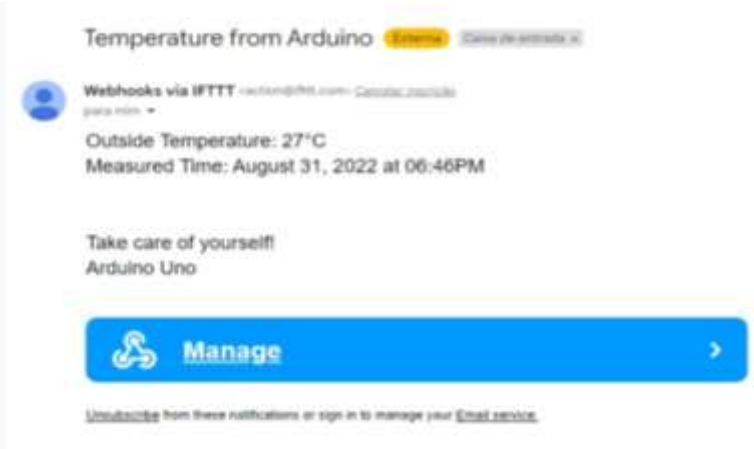


Figura 14: Resultado do monitoramento do dispositivo via IFTTT

Os resultados obtidos no período entre setembro de 2021 a agosto de 2022 foram apenas parciais do que se esperava, pois, muitos problemas/percalços foram encontrados durante o desenvolvimento do sistema. A experiência e conhecimento adquirido foram de grande utilidade. Soluções surgiram, mas não de forma eficiente, motivo pelo qual novas reflexões durante o projeto foram consideradas. Faltam ainda novas ideias a serem colocadas em prática, motivo pelo qual foi submetido ao CNPq continuação para este projeto. O *IFTTT* se mostrou um bom protocolo no que diz respeito ao monitoramento, todavia, será necessário um maior esforço nas camadas inferiores do modelo *OSI* de comunicação.

4. CONCLUSÃO

Nem todos os objetivos deste projeto foram alcançados, em virtude da complexidade de programação e da arquitetura dos protocolos.

Todavia, ressalta-se que pela importância da base tecnológica e pelo impacto na gerência pública, um novo projeto foi aprovado pelo CNPq para dar continuidade a este. Nosso intuito a partir de agora será implementar o *MQTT*. O projeto SISTEMA DE INFORMAÇÃO PARA MONITORAMENTO

DA ILUMINAÇÃO PÚBLICA, que foi aprovado através do Edital n° 22/2022 - Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação – PIBITI/CNPq está dando continuidade a este atual que chegou ao fim.

5. REFERÊNCIAS

ANEEL. Agência Nacional de Energia Elétrica. Iluminação pública. 24 fev. 2016. Disponível em: https://www.aneel.gov.br/destaques-consumidor/-/asset_publisher/kM1X2uTBr6qH/content/iluminacao-publica/655804. Acesso em: 20 ago. 2021

CAMPBELL, Carlene E-A. LOO, K-K. KURDI, Heba A. KHAN S. Comparison of IEEE 802.11 and IEEE 802.15.4 for Future Green Multichannel Multi-radio Wireless Sensor Networks. International Journal of Communication Networks and Information Security (IJCNIS) Vol. 3, No. 1, April 2011

GARCIA, Fernando P. SOUZA, José Neuman de. ANDRADE, Rossana M. C. Sistemas de Monitoramento Passivo para RSSF – Soluções Existentes e uma Nova Proposta Energeticamente Eficiente, 31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2013

LIU, Yu. TONG, Kin-Fai. QIU, Xiangdong. LIU, Ying DING Xuyang. "Wireless Mesh Networks in IoT networks," 2017 International Workshop on Electromagnetics: Applications and Student Innovation Competition, 2017, pp. 183-185, doi: 10.1109/iWEM.2017.7968828.

LOUREIRO, Antonio A.F. NOGUEIRA José Marcos S. RUIZ, Linnyer Beatrys. MINI Raquel Aparecida de Freitas. NAKAMURA, Eduardo Freire. FIGUEIREDO, Carlos Maurício Seródio. Redes de Sensores Sem Fio. Departamento de Ciência da Computacão, Universidade Federal de Minas Gerais. Publicado em 2003.

MQTT For Sensor Networks (MQTT-SN) Protocol Specification Version 1.2 – Disponível em <https://mqtt.org/mqtt-specification/>. Acesso em 11 ago. 2021.

PRAKASH, N. RAJALAKSHMI, M. e NEDUNCHEZHIAN, R. "Analysis of QoS for Conveying Authorisation Based on Internet of Things (IoT) in Wireless Sensor Networks (WSN)," 2020 7th International Conference on Smart Structures and Systems (ICSSS), 2020, pp. 1-9, doi: 10.1109/ICSSS49621.2020.9202338.

PANTONI, Rodrigo. FONSECA, Cleber e BRANDÃO, Dennis. Energy Efficiency – The Innovative Ways for Smart Energy, the Future Towards Modern Utilities. InTech, Chapters published October 17, 2012 under CC BY 3.0 license

SILICON LABS. Wi-Fi in Sensor Applications – Disponível em: <http://www.redpinesignals.com/pdfs/White-Paper-Sensor-Networks.pdf>. Acesso em 10 ago. 2021

SHAFIQUE, K. KHAWAJA, B. A. SABIR, F. QAZI, S. MUSTAQIM, M. "Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios," in IEEE Access, vol. 8, pp. 23022-23040, 2020, doi: 10.1109/ACCESS.2020.2970118. TOWNSEND, Antony. Smart Cities: Big Data, Civic Hackers, and the Quest for a New Utopia. W. W. Norton & Company. Publicado em 2013.

6. PARECER DO ORIENTADOR

Classificação de desempenho do bolsista e voluntário(s):

Excelente [ ] Bom [ ] Regular [ ] Ruim [ ]

7. ASSINATURAS

Coordenador de Projeto: RAMON LEONN VICTOR MEDEIROS

Bolsista: LUAN GOMES DE CARVALHO

Voluntário: MATHEUS JABES LYRA DUARTE DE LIMA