UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

PEDRO CANNAVALE GRAÇA

SISTEMA DE AQUISIÇÃO DE DADOS UTILIZANDO O MÓDULO ESP8266 NodeMCU

Guaratinguetá
2017

Pedro Cannavale Graça

SISTEMA DE AQUISIÇÃO DE DADOS UTILIZANDO O MÓDULO ESP8266 NodeMCU

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estatual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica .

Orientador: Prof^{o} Dr. Teófilo Miguel de Souza

Guaratinguetá
2017

Graça, Pedro Cannavale
G729s Sistema de aquisição de dados utilizando o módulo ESP8266 NodeMCU / Pedro Cannavale Graça – Guaratinguetá, 2017.
42 f : il.
Bibliografia: f. 34-35

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2017.
Orientador: Prof. Dr. Teófilo Miguel de Souza

1. Microcontroladores. 2. Detectores. 3. Aquisição de dados.
I. Título

CDU 621.3.026

Luciana Máximo
Bibliotecária/CRB-8 3595

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

PEDRO CANNAVALE GRAÇA

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE **"GRADUADO EM ENGENHARIA ELÉTRICA "**

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Profº Dr. LEONARDO MESQUITA
Coordenador

BANCA EXAMINADORA:

Profº Dr. Teófilo Miguel de Souza
Orientador/UNESP-FEG

Profº MSc. Evaldo Chagas Gouvêa
UNESP-FEG

Profº MSc. Thaís Santos Castro
UNESP-FEG

Dezembro , 2017

À minha mãe Rita de Cássia Cannavale Graça e ao meu pai Paulo Cesar da Silva Graça, por todo apoio, compreensão e carinho.

AGRADECIMENTOS

Agradeço a minha família por todo apoio durante a minha formação.

Agradeço ao meu orientador, Prof. Dr. Teófilo Miguel de Souza, pela confiança depositada em minha pessoa.

Agradeço ao Prof. Me. Evaldo Chagas Gouvêa, por toda a ajuda durante o desenvolvimento deste trabalho

Agradeço a minha namorada, Carina Lemos, pela paciência e ajuda nos momentos mais difíceis.

Agradeço aos meus amigos da faculdade, pelo companheirismo em especial a William Hideki.

“Decidir o que não fazer é tão importante quanto decidir o que fazer“
(Steve Jobs)

RESUMO

A proposta deste projeto é criar um protótipo de um sistema de aquisição de dados, capaz de medir grandezas como luminosidade, temperatura e umidade, em ambientes de trabalho como laboratórios de pesquisa. O projeto divide-se em três partes principais: o conjunto de sensores, composto pelo sensor DHT22 e o LDR, o microcontrolador NodeMCU e a plataforma IoT. Inicialmente os sensores realizam a medição de suas respectivas grandezas, esses dados coletados são então tratados pelo microcontrolador com suporte a conexão WiFi, que é responsável por realizar a interface entre os dados e a plataforma IoT. Uma vez na plataforma, os dados podem ser visualizados em tempo real pelo usuário ou armazenados em banco de dados para que possam ser consultados depois, isso tudo pode ser realizado pelo usuário através de um acesso remoto utilizando um navegador *web*. Para a validação deste sistema foi realizada a calibração dos sensores e todos os dados foram comparados com dados provenientes de equipamentos vendidos no mercado. Os dados coletados revelam aspectos interessantes sobre o ambiente de trabalho do Laboratório de Diamante CVD no Centro de Energias Renováveis e demonstram a importância deste tipo de sistema.

PALAVRAS-CHAVE: NodeMCU. Plataforma IoT. Aquisição de dados.

ABSTRACT

The purpose of this project is to create a prototype of a data acquisition system capable of measuring such as luminosity, temperature and humidity, in work environments such as research laboratories. The project was divided into three main parts: set of sensors, consisting of the DHT22 sensor and the LDR, the NodeMCU microcontroller and the IoT platform. Initially the sensors perform the measurement of their respective magnitudes, this collected data is then treated by the microcontroller with WiFi connection support, which is responsible for performing the interface between the data and the IoT platform. Once in the platform, the data can be viewed in real time by the user or stored in database so they can be queried later, this can all be accomplished by the user through a remote access using a web browser. For the validation of this system the calibration of the sensors was performed and all data were compared with data from equipment sold in the market. The data collected reveal interesting aspects about the working environment of the CVD Diamond Laboratory in the Renewable Energy Center and demonstrate the importance of this type of system.

KEYWORDS: NodeMCU. IoT platform. Data acquisition.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UNESP	Universidade Estadual Paulista
WIFI	*Wireless Fidelity* (Conexão sem fio)
IoT	*Internet of Things* (Internet das coisas)
MIT	*Massachusetts Institute of Technology* (Instituto de Tecnologia de Massachusetts)
SOC	*System on Chip* (Sistema em um chip)
MCU	*Microcontroller Unit* (Unidade de microcontrolador)
SPI	*Serial Peripheral Interface* (Interface periférica serial)
UART	*Universal asynchronous receiver/transmitter* (Receptor/Transmissor assíncrono universal)
IP	*Internet Protocol* (Protocolo de internet)
GPIO	*General Purpose Input/Output* (Porta de entrada / saída de propósito geral)
GND	*Ground* (Terra)
IDE	*Integrated Development Environment* (Ambiente de desenvolvimento)
LDR	*Light Dependent Resistor* (Resistor dependente da luz)
SSL	*Secure Socket Layer* (Protocolo de Segurança)
API	*Application Programming Interface* (Interface de programação de aplicativos)
HTML	*HyperText Markup Language* (Linguagem de Marcação de Hipertexto)

LISTA DE SÍMBOLOS

R$	Unidade monetária Brasileira (Real)
%	Porcentagem
°C	Grau Celsius
Ω	Ohm

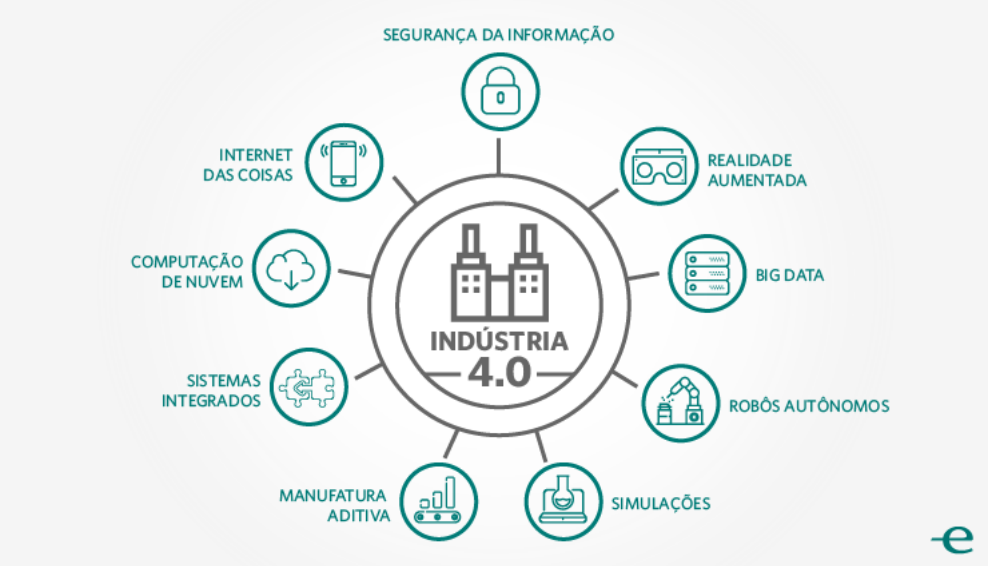
SUMÁRIO

1 INTRODUÇÃO

Apoiada no grande desenvolvimento do sistema WiFi (Wireless Fidelity, Conexão sem fio), o mundo está vivendo a sua quarta revolução industrial, conhecida como Indústria 4.0. Por causa disso surgiram novos conceitos e tecnologias que podem ser observados na Figura 1. Como por exemplo: IoT (Internet of Things, Internet das Coisas), termo criado pelo MIT (Massachusetts Institute of Technology, Instituto de Tecnologia de Massachusetts), o qual representa a comunicação entre diversos dispositivos, criando um mundo onde qualquer processo pode ser monitorado, controlado e analisado; Computação em Nuvem, tecnologia explorada inicialmente por grandes empresas de tecnologia como Google, Oracle e Microsoft, busca explorar a relação entre cliente e servidor; com a computação em nuvem, computadores clientes podem utilizar a capacidade de armazenamento de servidores, para realizar diversas tarefas como: armazenar e acessar dados, executar programas e periféricos, tudo através da conexão com a internet e podendo ser realizado de qualquer lugar do mundo; *Big Data*, refere-se basicamente a uma quantidade enorme e complexa de informação.

Desses conceitos e tecnologias surgem novos dispositivos capazes de se comunicar e armazenar dados, estes são os principais responsáveis pelo aprimoramento e digitalização dos processos físicos. Com isso, torna-se muito comum a presença no mercado de sistemas de monitoramento e aquisição de dados, pois através da tecnologia IoT, os sensores, computadores e outros dispositivos, que seriam as *coisas*, conseguem se comunicar entre si e gerar conjunto de dados e relatórios, que são normalmente utilizados no planejamento estratégico das empresas, na gestão dos principais serviços públicos e até mesmo na administração de cidades (CLINE, 2017).

Figura 1 – Industria 4.0



Fonte: Griletti (2017)

Entretanto, o conceito IoT não serve apenas para o meio empresarial: um sistema de aquisição de dados baseado nesta tecnologia pode ser facilmente implementado em laboratórios de pesquisa. Normalmente durante o desenvolvimento dos experimentos, é comum alunos e professores coletarem

dados como temperatura, luminosidade e umidade; aperfeiçoar e digitalizar este processo, pode ajudar a revelar uma influência das condições do ambiente de trabalho nos experimentos realizados.

Como exemplo o laboratório de Diamante CVD do Centro de Energias Renováveis da UNESP de Guaratinguetá já possui um equipamento responsável por monitorar a temperatura e umidade do ambiente, durante experimentos ele costuma ser consultado por alunos e professores. A introdução de um sistema baseado na tecnologia IoT, traz diversos benefícios, entre eles: permite uma análise mais precisa e de maior qualidade, aumenta a velocidade no processo, além de possibilitar aos responsáveis pela pesquisa uma flexibilidade devido à possibilidade de um acesso remoto aos dados coletados.

Esta monografia é constituída de cinco capítulos. O segundo capítulo trata da Revisão Bibliográfica sobre os principais componentes que constituem este sistema de aquisição de dados. O terceiro capítulo apresenta de forma detalhada todas as etapas para o desenvolvimento deste sistema. O quarto capítulo engloba as amostras dos dados obtidos e suas respectivas análises. O quinto capítulo trata da conclusão deste projeto.

1.1 OBJETIVO

Criar um sistema de aquisição de dados para monitorar as condições de ambientes de trabalho. Os dados coletados serão armazenados para que possam ser posteriormente analisados, buscando verificar a influência dos mesmos nos experimentos.

2 REVISÃO BIBLIOGRÁFICA

2.1 MICROCONTROLADOR

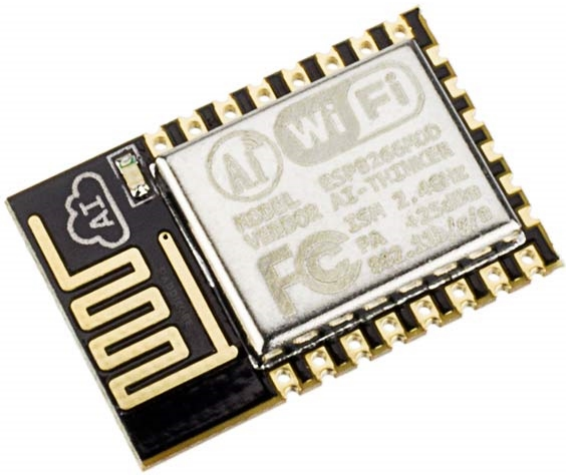
2.1.1 ESP8266

O módulo ESP8266 é um microcontrolador da empresa Espressif (OLIVEIRA, 2017) que trata-se basicamente de um WiFI-SOC, ou seja, ele possui a capacidade de se conectar a uma rede WiFi, pode atuar como uma aplicação stand-alone, onde ele não precisa de nenhum outro componente para funcionar, ou como um servidor escravo MCU (Microcontroller Unit, Unidade de microcontrolador). Na segunda condição ele funciona como um adaptador WiFi para outro microcontrolador, como por exemplo o arduino; importante ressaltar que normalmente neste tipo de condição, é necessário utilizar adaptadores do tipo SPI (Serial Peripheral Interface, Interface periférica serial) ou UART (Universal asynchronous receiver/transmitter, Receptor/Transmissor assíncrono universal).

Em operação o ESP8266 pode funcionar em duas configurações, são elas: *access point* e *client*. Na primeira configuração, ele funciona basicamente como um roteador, criando uma rede WiFi restrita por *login* e senha. Neste modo o ESP8266 cria um servidor com IP (Internet Protocol, Protocolo de internet) aleatório ou predefinido dependendo da programação realizada, este servidor pode conter uma página *web* com informações dos componentes ligados ao ESP8266. Como *client*, ele estabelece uma conexão com a rede WiFi escolhida, uma vez conectado também cria um servidor e todos os dispositivos conectados na mesma rede WiFi que o ESP8266 têm acesso a este servidor pelo endereço de IP. Este servidor também pode conter uma página *web* e seu IP também pode ser aleatório ou predefinido na programação. Ainda é possível realizar uma terceira configuração e fazer o ESP8266 trabalhar simultaneamente como *access point* e *client* (ŠKRABA,A. et al, 2016).

Este sistema de aquisição de dados opera na configuração *client*, para que fosse possível a utilização de uma plataforma IoT para a transmissão dos dados, e além disso utiliza na verdade o módulo ESP8266-12 acoplado ao microcontrolador NodeMCU V3. Este módulo, que pode ser visto na Figura 2, possui um tamanho reduzido e um custo razoável, quando comparado a outros adaptadores de WiFi, possui um protocolo WiFi de comunicação 802.11 b/g/n/e/i, que permite um alcance de até 120 m em local aberto.

Figura 2 – Módulo ESP8266-12



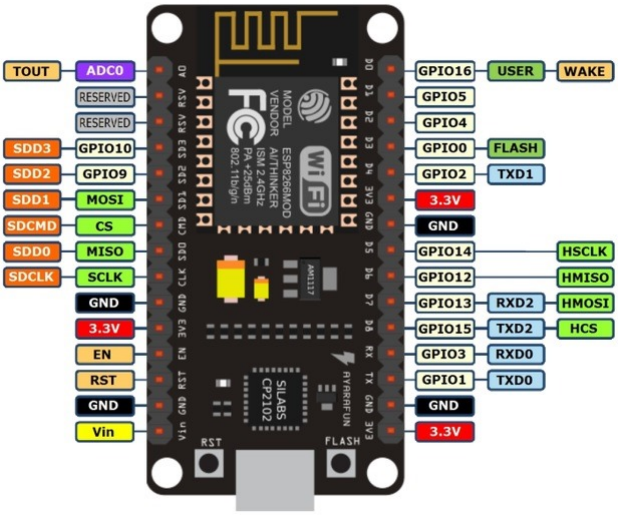
Fonte: Addicore (2017)

2.1.1.1 NodeMCU

O microcontrolador NodeMCU surgiu para facilitar a construção de projetos utilizando o ESP8266, pois apresenta um regulador de tensão de 3,3 V (tensão de operação do ESP8266), e com ele não é mais necessário a utilização de outros componentes como adaptadores SPI e UART para realizar a conexão. Ele surgiu logo após o lançamento do ESP8266, sendo lançado com o intuito de ser uma placa para desenvolvimento de projetos de caráter IoT. Outra grande vantagem deve-se ao fato de apresentar uma interface *USB-serial* acoplada, o que facilita tanto a parte de alimentação do microcontrolador, que pode ser realizada através de um carregador de celular, como a parte de transmissão do programa escrito do computador para a placa. Vale lembrar que o NodeMCU pode ser programado através de *scripts* escritos na linguagem LUA, ou pode ser programado através da linguagem C++ , pela própria plataforma do arduino, como foi realizado neste projeto (OLIVEIRA, 2017).

Dentre suas características físicas, que podem ser observadas na Figura 3, ele possui uma faixa de operação entre 5 e 9 V, isto permite a utilização de por exemplo uma bateria alcalina de 9 V para alimentação do microcontrolador. Além disso possui um total de 11 portas GPIO (General Purpose Input/Output, Porta de entrada / saída de propósito geral), as quais podem ser utilizadas para conectar sensores ou outros dispositivos externos.

Figura 3 – Ilustração NodeMCU junto com Listagem dos Pinos



Fonte: Fallows (2016)

2.2 PROGRAMAÇÃO MICROCONTROLADOR

2.2.1 Linguagem C++

O NodeMCU pode ser programado através de *scripts* escritos na linguagem LUA, ou através da linguagem C++. Este projeto utiliza a linguagem C++ por vários motivos, entre eles: por trata-se de uma linguagem mais difundida, com um número muito grande de adeptos, utilizada em vários programas comerciais e ensinada na maioria das universidades, o que facilita a busca por materiais de consulta, como livros e artigos. Outra vantagem da linguagem C++ é que ela facilita o entendimento da lógica de algoritmos de programação.

Já a linguagem LUA, trata-se de uma *scripting language*, isto implica ente outras coisas, que ela não precisa ser compilada, diferente da linguagem C++. A linguagem LUA é muito fácil de ser aprendida, é uma linguagem simples, e como pode ser visto na Figura 4, o código escrito em LUA fica menor do que o escrito em C++.

Figura 4 – Código em C++ à esquerda, código LUA, à direita

```cpp
#include <iostream>

int fact (int n){
    if (n==0) return 1; else
    return (n*fact(n-1));
}

int main (){
    int input;
    using namespace std;
    cout << "Enter a number: " ;
    cin >> input;
    cout << "factorial: " << fact(input) << endl;
    return 0;
}
```

```lua
function fact (n)
    if n==0 then
        return 1
    else
        return n * (fact(n-1))
    end
end

print ("enter a number")
a = io.read("*number")
print ("Factorial: ",fact(a))
```

Fonte: imagem adaptada do site stackoverflow

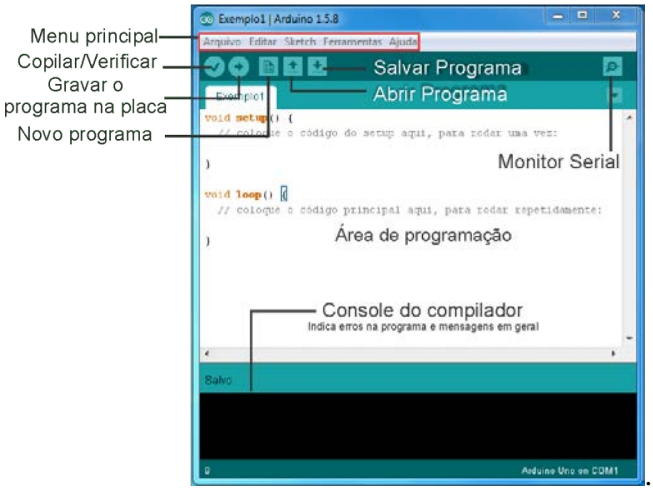
Além dessas características, por ser uma linguagem criada por brasileiros, possui um manual oficial de alta qualidade e fácil entendimento; mesmo tendo menos adeptos do que a linguagem C++, é possível encontrar vários exemplos na internet. Entretanto, ao optar pela linguagem LUA, é necessário utilizar plataformas como o ESPlorer para desenvolver o código, durante o desenvolvimento do projeto esta solução foi testada, mas muitos erros foram encontrados e existe pouco material sobre as plataformas de desenvolvimento da linguagem LUA. Isto foi outro ponto favorável para a escolha da linguagem C++ para o projeto, pois ela permite a utilização da plataforma *Arduino IDE*, para realizar a programação do microcontrolador NodeMCU.

2.2.2 Arduino IDE

Utilizar um ambiente de desenvolvimento integrado, traz várias vantagens para o programador, entre elas: menor tempo e esforço para escrever o código, ferramentas que permitem uma melhor gestão do projeto e normalmente já apresentam um compilador integrado. Este projeto utiliza a IDE do Arduino, porque ela fornece uma grande quantidade de recursos para o desenvolvimento do projeto, como por exemplo: o monitor serial, o qual permite a monitoração da conexão entre o computador e o microcontrolador NodeMCU, e permite ainda que o usuário forneça comandos ao microcontrolador. Outra grande vantagem de se utilizar esta plataforma deve-se à linguagem de desenvolvimento, que é muito semelhante à linguagem C++. Importante ressaltar que qualquer novo arquivo criado na plataforma *Arduino IDE* vem por padrão com duas funções já escritas, a função *void setup*, onde normalmente é realizada a declaração de alguns parâmetros e a função *void loop*, a qual faz com que o programa criado seja executado infinitamente pelo microcontrolador (MCROBERTS, 2011). As funções *setup* e *loop* deste projeto podem ser visualizadas nos Apêndices B e C respectivamente.

Esta plataforma conta ainda com uma interface simples, como pode ser observado na Figura 5, além de um vasto conjunto de bibliotecas referentes ao NodeMCU e ao ESP8266, facilitando a etapa de programação do projeto, pois estas bibliotecas fornecem funções que reduzem o número de linhas de programação, além de fornecer exemplos de projetos que podem ser adaptados conforme a necessidade do usuário.

Figura 5 – Arduino IDE



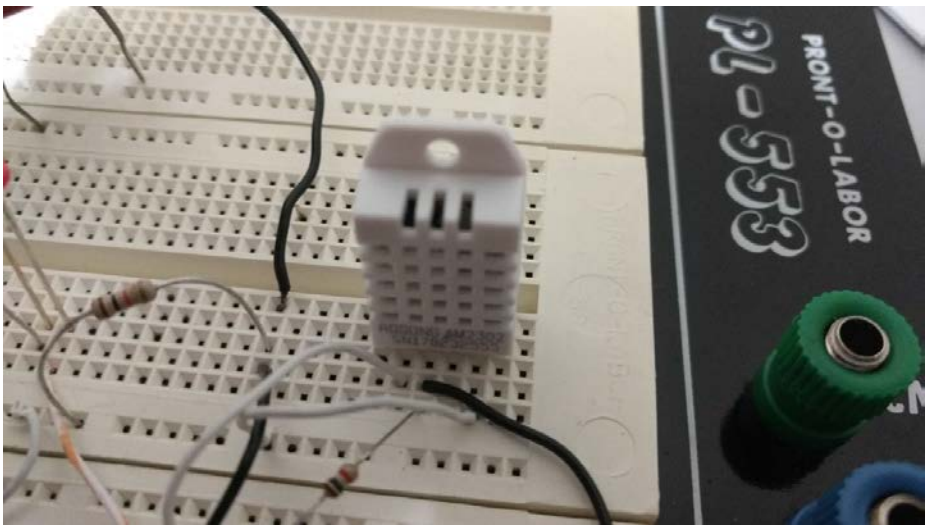
Fonte: Mota (2017)

2.3 SENSORES

2.3.1 Temperatura e Umidade (DHT22)

O sensor DHT22, também conhecido como AM2302, é constituído basicamente por um sensor capacitivo de umidade e dispositivos muito precisos para a medição de temperatura, conectados à um microcontrolador de 8 *bits* de alto desempenho. Este sensor possui apenas 4 pinos, sendo um para a sua alimentação que deve estar entre o intervalo de 3,3 - 5,5 V DC, o outro é para a transmissão dos dados (pino de dados serial e porta bidirecional), possui um pino vazio (NULL) e o último pino restante deve ser ligado no terra (GND). Neste projeto foi utilizado juntamente ao pino de dados, um resistor *pull up* de 1 kΩ para evitar a flutuação do pino de dados, garantindo os níveis lógicos esperados, caso algum dispositivo seja desconectado. Na Figura 6 pode ser observado tanto o sensor DHT22 como o resistor *pull up*.

Figura 6 – Sensor DHT22



Fonte: Produção do próprio autor.

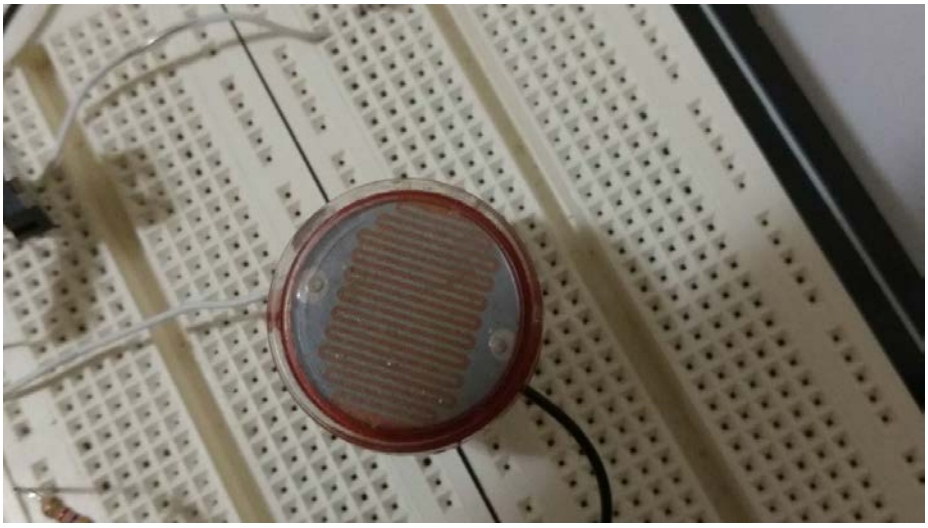
Por apresentar um baixo consumo de corrente enquanto está realizando as medições, em torno de 2,5 mA, além de uma boa precisão e uma boa faixa de operação tanto para a umidade (0 até 100%) como para a temperatura (-40 até 125 °C), este sensor é capaz de atender todas as necessidades do projeto. Vale ressaltar que o mercado apresenta sensores de custo mais reduzidos que o DHT22, como por exemplo o DHT11, seu antecessor, mas que possui uma faixa de operação mais limitada sendo entre 20 até 80% para a umidade e de 0 até 50 °C para a temperatura (AOSONG ELECTRONICS CO, 2017).

2.3.2 Luminosidade (LDR)

O sensor LDR (*Light Dependent Resistor*, Resistor dependente da luz), é composto fisicamente por fotocondutores de sulforeto de cádmio. O LDR trata-se basicamente de um resistor com resistência variável ao nível de intensidade da luz; conforme a luminosidade aumenta, a sua resistência diminui, pois quando a luz incide sobre o sulforeto de cádmiro ocorre a liberação de elétrons para a banda

condutora. Este processo gera um aumento da corrente, e consequentemente ocorre a diminuição da resistência do LDR, em contrapartida com uma baixa incidência de luz, temos uma baixa condutividade e com isso uma alta resistência (KODALI e MANDAL, 2016). As vantagens de se utilizar o sensor LDR neste projeto, o qual pode ser observado na Figura 7, estão relacionadas ao baixo custo do componente e sua fácil utilização, além disso existem vários projetos de circuitos eletrônicos que utilizam LDR, o que facilita a busca por materiais de apoio.

Figura 7 – Sensor LDR

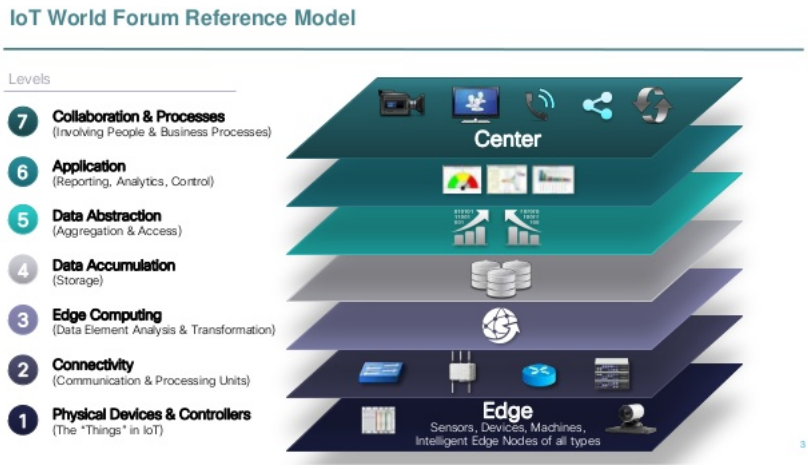


Fonte: Produção do próprio autor.

2.4 PLATAFORMA IOT

A tecnologia IoT pode ser dividida em várias camadas, como pode ser observado na Figura 8, entre essas camadas temos a camada de armazenamento, representada pelas plataformas IoT. Geralmente a tecnologia é dividida em três camadas: a camada dos produtores, camada de armazenamento, e a camada dos consumidores. Na camada dos produtores, que seria a camada de nível mais baixo, estão geralmente os sensores, entretanto pode ser qualquer dispositivo capaz de gerar dados. Na camada de armazenamento, que seria a camada de nível intermediário, estão as plataformas IoT, que têm a função de receber, armazenar e disponibilizar os dados para a última camada, a camada dos consumidores. Nesta camada estão os usuários, aplicações e serviços que utilizam a plataforma para ter acesso aos dados (RODRIGUES, 2016).

Figura 8 – Camadas IoT



Fonte: Green (2014)

Estas plataformas são espaços simples e práticos, independente do tipo de *coisa* que está conectado a ela, utilizam conceitos de computação de nuvem e *big data*, para ajudar o usuário na análise e armazenamento dos dados coletados. Todas as plataformas geralmente apresentam uma grande quantidade de recursos que ajudam os consumidores a visualizarem os dados, cabe ao desenvolvedor pesquisar qual fornece o melhor custo *versus* benefício para o projeto. É preciso verificar pontos importantes relacionados à segurança dos dados, é interessante buscar plataformas que possuam o certificado SSL (*Secure Socket Layer*) de segurança, este protocolo é o principal responsável por criptografar toda a conexão entre o navegador e o servidor.

2.4.1 Thinger.io

Este projeto utiliza a plataforma thinger.io, devido à grande variedade de recursos para a construção do *dashboards*[1], além da possibilidade de armazenar os dados em *data buckets*[2], essas características auxiliam muito na visualização e analise dos dados pelo usuário.

A thinger.io é uma plataforma de código aberto (*open source*) e seu fluxo de trabalho funciona da seguinte maneira:

- Criação do *DashBoard* para um dispositivo e obtenção dos dados
- Análise dos dados pela seleção de vários parâmetros
- Apresentar os dados, selecionando intervalos de tempo como uma facilidade gráfica

Esta plataforma inclui o conjunto completo de API (Application Programming Interface, Interface de programação de aplicativos) para lidar com *HTTP requests*, além disso possui um aplicativo para celular que permite o controle de todos os dispositivos conectados a plataforma (VITHLANI et al, 2017).

[1] Painel que possibilita a visualização dos dados.
[2] Espaço para armazenamento dos dados coletados

Para utilizar a thinger.io, o usuário precisa através do seu navegador web acessar o *site* http://thinger.io, criar um *login* e senha; as informações de acesso serão utilizadas na programação do microcontrolador para realizar a conexão do mesmo com a plataforma. A thinger.io é uma das melhores plataformas do mercado, sua versão gratuita limita o número de dispositivos, de *dashboards* e *data buckets*, mas mesmo assim conseguiu atender todas as necessidades deste projeto.

3 SISTEMA DE AQUISIÇÃO DE DADOS

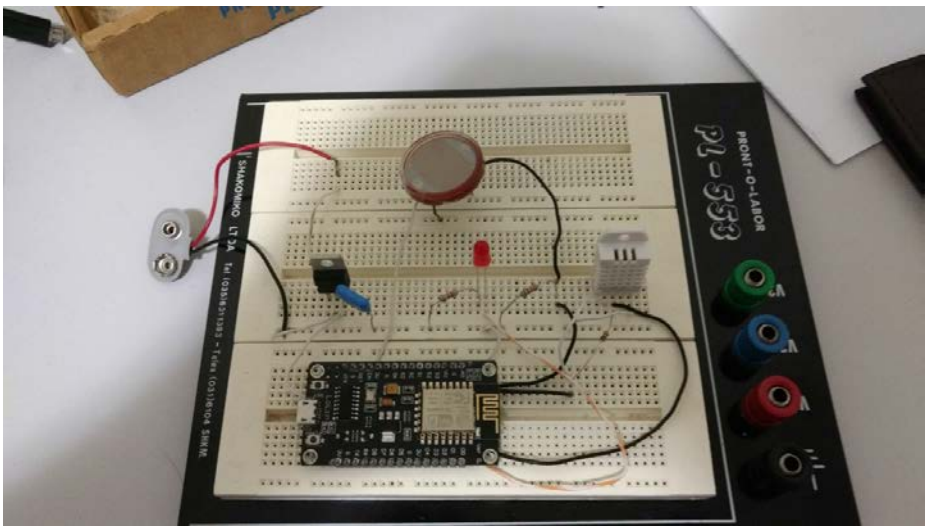
3.1 DESCRIÇÃO DO SISTEMA

O sistema que pode ser observado na Figura 9, foi montado e projetado para ser capaz de medir grandezas como temperatura, umidade e luminosidade e guardar esses dados para que possam ser analisados posteriormente. Pode-se então dividir este sistema de aquisição de dados em três partes:

- Conjunto de sensores
- Microcontrolador NodeMCU
- Plataforma IoT

Inicialmente cada sensor é responsável por realizar a medida da sua grandeza correspondente. No caso do LDR, o valor da sua resistência é convertido por um conversor A/D, e o sinal digital resultante deste conversor é modificado para que a resposta seja fornecida na unidade lux. Já o DHT22 fornece a resposta diretamente na forma digital, a qual é interpretada pela porta GPIO do microcontrolador.

Figura 9 – Foto do Circuito montado no Protoboard



Fonte: Produção do próprio autor.

O microcontrolador NodeMCU recebe toda a programação necessária para realizar a interface entre os dados coletados e a plataforma IoT, ou seja, ele fica responsável por armazenar os dados coletados pelos sensores em variáveis e transmitir os mesmos para a plataforma. Além disso, toda a parte de conexão tanto com a rede WiFi quanto com a plataforma IoT também é realizada pelo microcontrolador. Na plataforma, os dados serão apresentados para o usuário e serão armazenados para análises futuras.

O circuito foi montado com componentes simples e possui duas formas de alimentação: uma alimentação direta onde aproveitamos a porta micro-usb do NodeMCU e conectamos diretamente um carregador de celular, ou um cabo usb/micro-usb e aproveitamos a energia de um computador ou

notebook. Pode ser utilizado também uma alimentação indireta através de um circuito contendo uma bateria alcalina de 9 V e um regulador de tensão de 5 V. A saída do regulador é aplicada no pino *Vin* do NodeMCU.

É possível observar na Figura 9 um led vermelho, que é utilizado para informar falhas na conexão do NodeMCU com a rede WiFi, quando ocorre a falha o led acende, sendo desligado quando a conexão for reestabelecida para que não prejudique a coleta de dados relacionada à luminosidade.

3.2 PROGRAMAÇÃO NODEMCU

3.2.1 Bibliotecas e Variáveis

Para facilitar a etapa de programação do microcontrolador NodeMCU no que diz respeito ao tempo gasto pelo programador para escrever o código e em relação ao número total de linhas, foram adicionadas ao ambiente de desenvolvimento do arduino algumas bibliotecas. Esta parte do projeto pode ser vista no Apêndice A e ao todo foram adicionadas cinco bibliotecas, as quais serão descritas abaixo:

- SPI.h: trata-se da *serial peripheral interface*, esta biblioteca é responsável por fornecer funções capazes de realizar a comunicação entre um microcontrolador e um dispositivo periférico, ou entre dois microcontroladores. Esta comunicação é realizada através de um protocolo serial de dados adequado.

- ESP8266WiFi.h: trata-se de uma evolução da biblioteca padrão de WiFi do arduino. Esta biblioteca permite que o projeto utilize classes como: WiFi, *Client*, *Server* e *IP adress*. Todas essas classes possuem funções que serão utilizadas dentro da função *setup* e da função *loop*, para realizar tanto a parte de conexão com a rede, monitoramento desta conexão e também toda a parte de operação do ESP8266 na configuração *client*.

- ThingerWiFi.h: trata-se da biblioteca responsável por fornecer funções que permitam a conexão do microcontrolador com a plataforma thinger.io , além de fornecer as funções responsáveis pela declaração dos dispositivos e pelo envio de dados à plataforma, todas as funções desta biblioteca começam pela palavra thing.

- DHT.h: trata-se da biblioteca responsável por fornecer funções que facilitam a programação do sensor DHT22, ela contém as funções *dht.readTemperature()* e *dht.readHumidity()*, responsáveis por tratarem o sinal recebido na porta GPIO do microcontrolador.

- math.h: trata-se de uma biblioteca básica do arduino, que permite a realização de operações matemáticas mais complexas.

Além das bibliotecas, é nesta parte inicial do código que são definidos alguns parâmetros e ocorre a criação de todas as variáveis que serão utilizadas no programa. O Apêndice A, mostra que as duas primeiras variáveis criadas (linhas 9 e 10) são do tipo *char*, e recebem as informações da rede WiFi a qual o microcontrolador irá se conectar. Logo em seguida (linhas 12 a 16) são definidos os parâmetros

que possibilitam a conexão com a plataforma IoT e a porta GPIO que será utilizado pelo sensor DHT22. Para evitar erros, todas as variáveis criadas são inicializadas (linhas 20 a 25) com o valor zero, isto impede que algum resquício na memória seja somado ao valor medido.

Antes do programa entrar na função *void setup()*, ainda são chamadas duas funções: a *ThingerWiFi* (linha 27), responsável por realizar a conexão com a plataforma Thinger.io, ela recebe como parâmetros o nome de usuário, a identificação do dispositivo e a senha de acesso, respectivamente nesta ordem. E a função *WiFiServer* (linha 28), originária da biblioteca ESP8266WiFi, responsável por criar um servidor que recebe conexões a partir da porta 80.

3.2.2 Função Setup

A função *setup* (Apêndice B) realiza a conexão do microcontrolador NodeMCU com a rede WiFi escolhida, através da função *WiFi.begin* (linha 13) proveniente da biblioteca ESP8266WiFi.h, a qual recebe como parâmetros o nome e senha de acesso da rede WiFi. Além disso, ainda dentro da função *setup*, também é realizada a conexão dos dispositivos com a plataforma IoT, utilizando a função a *thing.add-wifi* (linha 4), que recebe os mesmos parâmetros que a função *WiFi.begin*.

Na função *setup* também é realizada a declaração do nosso API (linhas 15 a 18), através da função *thing*, que recebe como parâmetros o nome da API. Neste projeto foi utilizado o nome de Sensores, e este API possui três *resources* que são: temperatura, umidade e luminosidade, os quais recebem os valores medidos através dos sensores. Importante mencionar que os nomes aqui escolhidos são importantes, uma vez que as colunas do nosso banco de dados recebe automaticamente o mesmo nome dos *resources*.

Logo em seguida (linhas 21 a 28) existe a função *WiFi.status*, a qual é responsável por verificar a conexão com a rede WiFi, esta função está dentro de um bloco *while*, ou seja, enquanto o microcontrolador não estiver conectado a rede WiFi, o programa irá ficar travado neste ponto. Foi definido dentro do bloco *while* que enquanto não ocorrer a conexão, o led ficará acesso indicando esta falha, isto ocorre através do comando *digitalwrite(5,1)*, onde o número 1 representa o nível lógico alto e o número 5 representa o pino do microcontrolador onde foi realizada a conexão do led. Quando a conexão for realizada o programa irá sair do bloco *while* e o led será apagado, através do mesmo comando, só que alterando o número 1 pelo numero 0 para representar o nível lógico baixo.

Na parte final (linhas 29 a 37) da função *setup*, ocorre a chamada da função *server.begin()* e existem várias linhas de códigos iniciadas pela expressão *serial.*, esta função está diretamente relacionada a possibilidade de se trabalhar com o ESP8266 no modo de operação *access point*, pois como dito anteriormente este servidor iniciado pela *server.begin* pode conter uma página *web* para a transmissão dos dados. As linhas de código iniciadas por *serial.* são responsáveis pela comunicação do microcontrolador com o *monitor serial* da IDE do arduino. Durante o desenvolvimento inicial do projeto, a ferramenta *monitor serial* foi muito utilizada, porque era o meio mais fácil de verificar a comunicação do microcontrolador com os sensores.

Importante ressaltar que o *monitor serial* também é uma ferramenta muito importante para a operação em *access point*, pois é através dele que o usuário consegue descobrir o endereço de IP do servidor criado.

3.2.3 Função Loop

A função *loop*, que pode ser vista no Apêndice C, como o próprio nome exemplifica, uma vez que o programa atinge esta parte do código, esta função fica sendo executada infinitamente até que a alimentação seja desligada. É na função *loop* (linhas 3 a 8) que ocorre toda a parte de coleta dos dados, pois a cada vez que ela é executada ocorre uma atualização dos valores das grandezas.

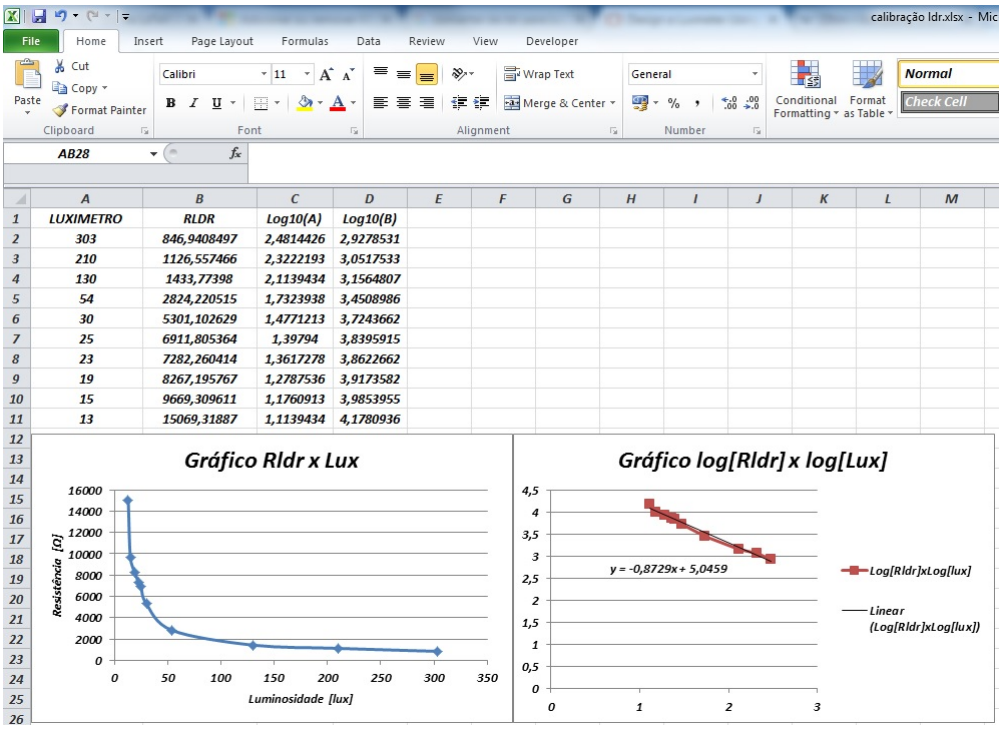
Assim como na função *setup*, na função *loop* também são realizados testes para verificar o funcionamento do projeto. Novamente pela função *WiFi.status* (linha 21) testa-se a conexão com a rede WiFi e quando usa-se o ESP8266 em modo de operação *access point*, testa-se através da função *server.available* (linha 13) o funcionamento do servidor e através da função *client.available* (linha 18) é realizado o teste da conectivade do usuário com o servidor.

Outro ponto importante da função *loop*, é que ao optar por não utilizar uma plataforma IoT para a transmissão dos dados, devemos criar a página web dentro da função, isso é realizado através de uma combinação da linguagem C++ com a linguagem HTML. O intuito deste projeto não foi trabalhar com tal recurso, mas a nível de exemplo a parte final do código presente no Apêndice C, mostra um exemplo de uma página web para disponibilizar o valor de luminosidade para o usuário.

3.3 CALIBRAÇÃO DO LDR

Como a saída do sinal digital proveniente do conversor A/D é um número entre 0 e 1024, sendo este intervalo a representação do intervalo de tensão entre 0 e 3,3V, foi necessário realizar a calibração deste sensor para que a saída fosse expressa em uma grandeza de fácil entendimento. A grandeza escolhida foi o lux, e a calibração foi realizada da seguinte forma: utilizando um luxímetro da marca Homis modelo 204 e com o auxílio do software Microsoft Excel, foi levantada uma curva relacionando o valor da resistência do sensor LDR, com o valor mostrado no luxímetro. Como a curva de resistência por lux trata-se de uma exponencial, foi necessário realizar uma conversão dos eixos para a escala logarítmica e traçar uma reta de tendência utilizando um recurso do próprio Microsoft Excel, para determinar a equação de luminosidade do LDR. O resultado deste procedimento pode ser observado na Figura 10.

Figura 10 – Método para Calibração do LDR



Fonte: Produção do próprio autor.

Vale lembrar que foi necessário calcular o valor da resistência do LDR antes de realizar o processo de calibração, o cálculo foi feito com base nas equações (1) e (2):

$$VoutLdr = 0.00322265625 * ValorLdr \tag{1}$$

$$Rldr = \frac{(9860 * 3.3 - VoutLdr)}{VoutLdr} \tag{2}$$

A equação (1) representa a conversão do valor digital (*ValorLdr*) para o valor analógico (*VoutLdr*). A ligação do LDR com o microcontrolador simboliza um divisor de tensão, então a equação (2) trata-se de um divisor de tensão, onde 9860 é o valor real do resistor, sendo este medido com o auxilio de um multímetro.

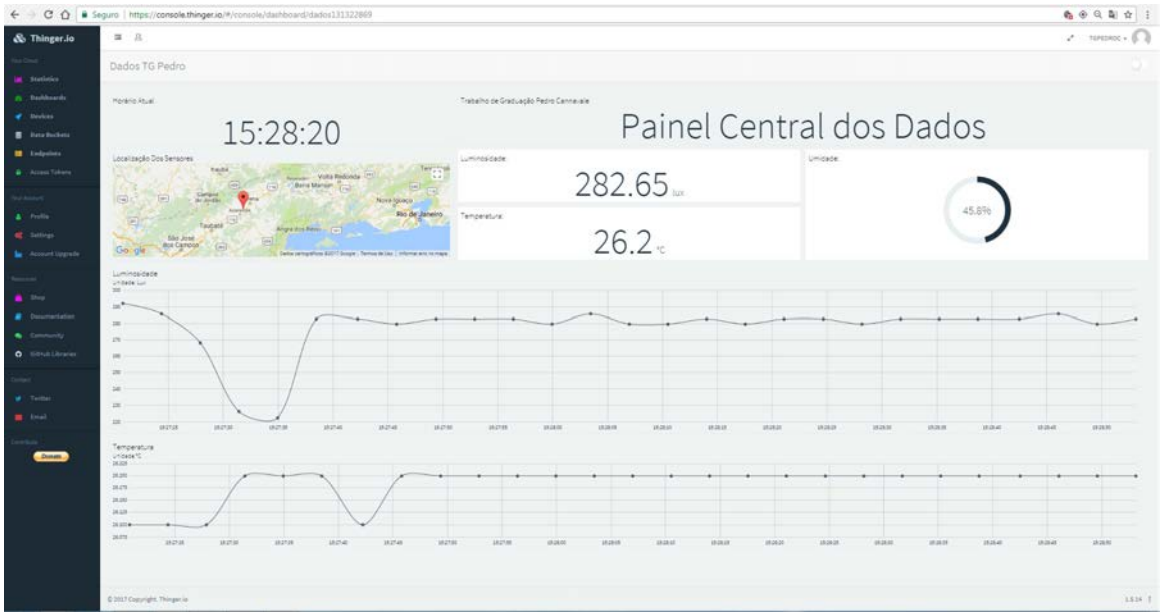
3.4 CONFIGURAÇÃO THINGER.IO

3.4.1 Dashboard

Este projeto utiliza uma plataforma IoT para que os dados possam ser acessados de qualquer lugar. A thinger.io permite que o usuário configure um *dashboard* para facilitar a visualização dos seus dados, o usuário pode optar por componentes do tipo *display*, que são os gráficos, barras de progresso e as caixas de textos, ou do tipo controle de dispositivo onde ele pode definir estados como *ON* ou *OFF*.

Como pode ser observado na Figura 11, o dashboard deste projeto contém vários elementos, entre eles estão: um relógio ajustado no mesmo fuso horário onde está localizado o dispositivo, a localização aproximada do dispositivo, três gráficos para mostrar a variação da temperatura, da luminosidade e da umidade, sendo os dois primeiros do tipo linha e possuem *labels* que facilitam a visualização dos dados, e o último do tipo rosca.

Figura 11 – DashBoard dos Dados



Fonte: Produção do próprio autor.

3.4.2 Data Buckets

A thinger.io possui um sistema de banco de dados próprio, este sistema é conhecido na plataforma como *data buckets*. Os *data buckets* permitem então que os dados medidos possam ser armazenados, nesta plataforma existe uma restrição de que o período mínimo que os dados podem ser guardados, é a cada um minuto. A partir da plataforma é possível enviar todos os dados armazenados para o email pessoal do usuário, os dados são salvos em arquivos de extensão .csv (Arquivo de Valores Separados por Vírgulas), este formato de arquivo pode ser aberto através do Microsoft Excel. Além da opção de exportar todos os dados, a plataforma permite que o usuário exporte especificando um período de dias e horas, como o espaço da plataforma é limitado, o usuário terá que em algum momento usar a função *clear,* onde novamente ele pode optar pela opção *clear all*, onde irá apagar todos os dados do banco, ou especificar um período para a limpeza.

4 RESULTADOS EXPERIMENTAIS E DISCUSSÕES

4.1 BASE DE DADOS

Diversos testes foram realizados com este sistema de aquisição de dados e além disso os dados foram comparados com dados coletados por equipamentos fabricados por grandes empresas do setor de instrumentação e a diferença é irrelevante.

Devido a este projeto utilizar a versão gratuita do thinger.io, o sistema está limitado a armazenar as informações com o tempo mínimo de um minuto. Mesmo com essa limitação, a quantidade de dados armazenada foi grande, sendo assim, será mostrado na Tabela 1 apenas uma amostra do monitoramento do dia 9 Nov. 2017 do Laboratório de Diamante CVD no Centro de Energias Renováveis.

Tabela 1 – Amostra de dados do dia 9 Nov. 2017

Data/Hora	Luminosidade	Temperatura	Umidade
2017-11-09T11:41:45.339	494,674	24,9	57,9
2017-11-09T11:43:33.644	49,817	24,9	57,9
2017-11-09T11:44:44.120	598,666	24,9	58
2017-11-09T11:45:54.631	54,257	24,9	58
2017-11-09T11:47:05.102	54,257	24,9	58
2017-11-09T11:48:15.547	489,486	24,9	58
2017-11-09T11:49:25.988	572,852	24,9	58
2017-11-09T11:50:36.490	49,817	24,9	58
2017-11-09T11:51:47.203	49,817	24,9	58
2017-11-09T11:52:57.648	501,696	24,9	58
2017-11-09T11:54:08.091	50,347	24,9	58,1
2017-11-09T11:55:18.547	49,817	24,9	58,1
2017-11-09T11:56:28.979	552,432	24,9	58,1
2017-11-09T11:57:39.610	521,639	24,9	57,7
2017-11-09T11:58:50.110	544,524	24,9	57,8
2017-11-09T12:01:08.729	282,647	25,2	58,6
2017-11-09T12:02:01.373	288,849	25,4	58,5
2017-11-09T12:04:04.522	288,849	25,6	58,5
2017-11-09T12:05:22.135	295,321	25,9	57,6
2017-11-09T12:06:32.683	292,05	26	57,2
2017-11-09T12:07:43.208	265,504	26,1	55,9
2017-11-09T12:08:53.732	292,05	26,1	54
2017-11-09T12:10:04.258	309,157	26,1	52,3
2017-11-09T12:11:15.777	305,581	25,9	51
2017-11-09T12:12:26.339	309,157	25,7	50,4
2017-11-09T12:13:36.863	305,581	25,6	52,6
2017-11-09T12:16:37.541	650,148	25,6	55,4
2017-11-09T12:17:40.266	589,895	25,7	54,5
2017-11-09T12:18:42.995	585,573	25,6	54,9
2017-11-09T12:19:50.587	592,072	25,6	55,2
2017-11-09T12:20:53.270	600,885	25,6	56,1

Fonte: Produção do Próprio Autor.

Os dados completos podem ser vistos através dos gráficos no Apêndice D; foi selecionada esta amostra dos dados para mostrar alguns aspectos interessantes. Como por exemplo, entre 12:13:36.863 e 12:16:37.541 existe uma diferença superior a um minuto, isto implica em duas causas possiveis, falha na conexão com a rede Wifi ou falha na alimentação do microcontrolador. Observando que em 12:16:37.541 a luminosidade teve seu valor praticamente dobrado, conclui-se que a causa mais provável foi um desligamento na alimentação do projeto para uma alteração da posição do sensor ldr em relação a fonte de luminosidade. Outro aspecto interessante é que quando temos valores muito baixos de luminosidade, conclui-se que o laboratório está com a iluminação desligada, este tipo de monitoramento permite um estudo sobre o consumo de energia elétrica do laboratório em questão. Além disso outro fato também interessante diz respeito a construção do laboratório, já que o mesmo é feito de tijolo ecológico, a temperatura interna do laboratório se matem praticamente constante não importando a temperatura no ambiente externo, que no dia em questão estava em 30°C durante este período de tempo.

A amostra da Tabela 2, serve para demonstrar um erro comum que ocorreu durante os primeiros ensaios. Como pode ser observado as 17:53:13.769 horas do dia 16 Out. 2017 ocorreu uma valor muito baixo de luminosidade, isto indica um problema de mal contato entre o sensor LDR e o microcontrolador NodeMCU, este problema foi resolvido através de uma soldagem adequada.

Tabela 2 – Amostra de dados do dia 16 Out. 2017

Data/Hora	Luminosidade	Temperatura	Umidade
2017-10-16T17:09:22.396	305,581	25,8	54,3
2017-10-16T17:10:25.138	359,549	25,8	52,2
2017-10-16T17:11:27.875	369,575	25,8	52,6
2017-10-16T17:31:21.593	309,157	26,3	49,5
2017-10-16T17:32:32.336	320,399	26,2	48,1
2017-10-16T17:45:16.022	279,642	26,3	53,3
2017-10-16T17:46:18.772	220,492	26,3	54,1
2017-10-16T17:47:22.139	247,886	26,3	52,7
2017-10-16T17:48:21.695	282,647	26,2	51,7
2017-10-16T17:49:24.445	192,296	26,1	49,8
2017-10-16T17:50:27.203	161,228	25,9	49,0
2017-10-16T17:51:29.957	260,229	25.,9	50,2
2017-10-16T17:53:13.769	0,156843	26,2	52,5

Fonte: Produção do Próprio Autor.

4.2 ESTIMATIVA DE CUSTO DO PROJETO

Na Tabela 3 tem-se uma estimativa do custo para montar este projeto. Sendo que os valores dos sensores e resistores foram baseados no comércio local, caso os mesmos sejam adquiridos em lojas *onlines* e em grande quantidades, o preço diminui consideravelmente.

Tabela 3 – Custo do Projeto

Componente	Preço por Unidade	Unidades	Total
Microcontrolador NodeMCU	R$25,00	1	R$25,00
DHT 22	R$17,00	1	R$17,00
LDR	R$1,00	1	R$1,00
Resistores	R$0,20	4	R$0,80
			R$43,80

Fonte: Produção do Próprio Autor.

4.3 DIFICULDADES APRESENTADAS

A principal dificuldade em se trabalhar com o microcontrolador NodeMCU, é que o mesmo não foi projetado para ser encaixado em *protoboards*. Devido a sua largura, ele ocupa todo o espaço da *protoboard* sendo necessário realizar as ligações por baixo dele, isto acaba sendo a principal causa de problemas de mau contato. O ideal é montar o projeto em uma placa perfurada, isto permite uma melhor organização dos componentes além de fornecer uma estética melhor do sistema.

Outra dificuldade é que existem poucos materiais de apoio sobre o NodeMCU, e os melhores materiais estão em outros idiomas, a maioria das pesquisas relacionadas a este microcontrolador estão sendo desenvolvidas na Índia.

4.3.1 Erros mais comuns

Foi muito comum na hora de realizar o *upload* do projeto para o microcontrolador, ocorrer alguns erros, como por exemplo: não identificar a porta usb a qual o microcontrolador esta conectado. Outro erro também muito comum é na hora de ler os dados no monitor serial, por padrão ele está configurado a uma taxa de 9600 *bauds* e para o NodeMCU precisa estar a uma taxa de 115200 *bauds*. Entre outros erros de compilação muita das vezes por falha do programador.

4.4 TRABALHOS FUTUROS

Pode ser apontado como trabalho futuro buscar soluções de alimentação do microcontrolador que deixem o sistema mais remoto. No processo atual é possível utilizar uma bateria alcalina para a alimentação do mesmo, entretando esta possui uma durabilidade de apenas 20 horas de trabalho, seria interessante pensar no conceito de energias renováveis e alimentar o microcontrolador a partir de painéis fotovoltaicos.

Outro detalhe do projeto é que atualmente o LDR não é capaz de medir com precisão a quantidade de lux, em ambientes externos, a explicação mais plausível para tal fato é a saturação do componente, poderia ser pensado em um novo sensor para ser usado no lugar do LDR, ou até algum ajuste para que o componente fosse capaz de realizar tais medições. Vale ressaltar também que as utilidades deste sistema são diversas, como por exemplo, pode-se aproveitar a parte de verificação da conexão com a rede WiFi para gerar um banco de dados de quantas vezes houve queda na conexão ou ainda monitorar por quanto tempo as luzes no laboratório ficam acessas.

5 CONCLUSÃO

O presente trabalhou descreveu todas as etapas de desenvolvimento de um sistema de aquisição de dados para ser utilizado em centros de pesquisa, capaz de verificar a temperatura, umidade e luminosidade, com o objetivo de verificar como esses fatores influenciam nos experimentos realizados. Todos os testes foram realizados no Laboratório de Diamante CVD, do Centro de Energias Renováveis da UNESP – Campus de Guaratinguetá.

O sistema final de medição é baseado em um circuito composto por sensores, um microcontrolador e uma plataforma IoT, entretanto a proposta inicial contava com a utilização de um arduino uno e o microcontrolador ESP8266, durante o desenvolvimento deste projeto este conjunto arduino e ESP8266, foi substituído pelo NodeMCU, que além de possuir as mesmas características deste conjunto, possui mais praticidade na parte de montagem do circuito, além de tornar o projeto mais barato por utilizar menos componentes.

Devido à limitação do sensor LDR, este projeto funciona apenas para ambientes internos, sendo necessário a troca do sensor para que o mesmo possa ser utilizado em ambientes externos.

Outro ponto que pode ser mencionado, é que neste projeto houve uma preocupação muito grande com a maneira como o usuário tem acesso aos dados coletados pelos sensores, devido a essa preocupação que foi escolhido a plataforma IoT thinger.io, em vez de utilizar uma página web criada pelo NodeMCU.

De maneira geral o objetivo principal do projeto foi alcançado com êxito. Todos os dados coletados foram enviados pela plataforma IoT e podem ser visualizados e analisados pelo usuário a qualquer momento e de qualquer lugar.

REFERÊNCIAS

ADDICORE. Ilustração módulo ESP8266. Disponível em: <https://www.addicore.com/ESP8266-ESP-12-p/ad247.htm>. Acesso em: 25 nov. 2017.

AOSONG ELECTRONICS CO. **Temperature and humidity module:** AM2302 product manual. Disponivel em: <http://akizukidenshi.com/download/ds/aosong/AM2302.pdf>. Acesso em: 9 nov. 2017.

CANTO, M. A. R. D. **Projeto e desenvolvimento de um sistema automatizado para monitoramento de fontes alternativas de energia**. 2007. 71 f. Dissertação (Mestrado em Engenharia Mecânica - Transmissão e Conversão de Energia) - Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2007. Disponível em: <http://hdl.handle.net/11449/99291>. Acesso em: 18 nov. 2017.

CLINE, G. **Industry 4.0 and Industrial IoT in Manufacturing:** a sneak peek. Disponível em: <http://www.aberdeenessentials.com/ opspro-essentials/ industry-4-0-industrial-iot-manufacturing-sneak-peek/>. Acesso em: 25 nov. 2017.

FALLOWS, J. **Understanding NodeMCU ESP8266-12E limitations**. Disponível em: <http:// play.fallows.ca/wp/projects/electronics-projects/ understanding-nodemcu-esp8266-12e-limitations/>. Acesso em: 25 nov. 2017.

GREEN, Jim. Building the Internet of Things: an IoT reference model. In: INTERNET OF THINGS WORLD FORUM, 2014, Chicago. **Anais Eletrônicos...** , 2014. p. 1-19. Disponível em: <https://www.slideshare.net/Cisco/building-the-internet-of-things-an-iot-reference-model>. Acesso em: 26 nov. 2017.

GRILETTI, L. **Indústria 4.0:** as oportunidades de negócio de uma revolução que está em curso. Disponível em: <https://endeavor.org.br/industria-4-0-oportunidades-de-negocio-de-uma-revolucao-que-esta-em-curso>. Acesso em: 25 nov. 2017.

KODALI,R. K; MANDAL,S. IoT based weather station. In: INTERNATIONAL CONFERENCE ON CONTROL, INSTRUMENTATION, COMMUNICATION AND COMPUTATIONAL TECHNOLOGIES, 1., 2016, Kanyakumari District. **Anais Eletrônicos...** Kanyakumari District: IEEE, 2016. Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=798 8038>. Acesso em: 21 ago. 2017.

KODALI,R. K; MAHESH,K. S. Low cost ambient monitoring using ESP8266. In: INTERNATIONAL CONFERENCE ON CONTEMPORARY COMPUTING AND INFORMATICS, 2., 2016, Greater

Noida. **Anais Eletrônicos...** Greater Noida: IEEE, 2016. Disponível em: <http://ieeexplore.ieee.org /stamp/stamp.jsp?arnumber=7918788>. Acesso em: 21 ago. 2017.

MCROBERTS M. **Arduino Básico**. São Paulo: Novatec, 2011. 456p.

MOTA, A. **O que é arduino e como funciona**. Disponível em: <https://portal. vidadesilicio.com.br/o-que-e-arduino-uno/>. Acesso em: 25 nov. 2017.

OLIVEIRA, R.R. **Uso do microcontrolador ESP8266 para automação residencial**. 2017. 55 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) – Universidade Federal do Rio de Janeiro, Escola Politécnica, 2017. Disponível em: <http://monografias.poli.ufrj.br/ monografias/monopoli10019583.pdf>. Acesso em: 18 nov. 2017.

RODRIGUES, J. **Mecanismos de segurança de dados para plataformas IOT**. 2016. 169 f. Dissertação (Mestrado em Engenharia de Computadores e Telemática) - Universidade de Aveiro, 2016. Disponível em: <https://ria.ua.pt/bitstrea m/10773 /17273/1/Diss-JoaoRodrigues.pdf>. Acesso em: 20 nov. 2017.

ROVERE, R.L.D. **Protótipo de um sistema inteligente de monitoramento do consumo de energia elétrica de uma residência**. 2016. 63 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica – Energia e Automação) – Universidade de São Paulo, Escola de Engenharia de São Carlos, 2016. Disponível em: <http://www.tcc.sc.usp.br/tce/disponiveis/18/180500/tce-31012017-121929/?& lang=br>. Acesso em: 18 nov. 2017.

ŠKRABA,A. et al. Prototype of group heart rate monitoring with NODEMCU ESP8266. In: MEDITERRANEAN CONFERENCE ON EMBEDDED COMPUTING, 6., 2017, Bar. **Anais Eletrônicos...** Bar: IEEE, 2017. Disponível em:<http://ieeexplore.ieee.org/stamp/stamp.jsp?ar number=7977151>. Acesso em: 21 ago. 2017.

ŠKRABA,A. et al. Streaming pulse data to the cloud with bluetooth LE or NODEMCU ESP8266. In: MEDITERRANEAN CONFERENCE ON EMBEDDED COMPUTING, 5., 2016, Bar. **Anais Eletrônicos...** Bar: IEEE, 2016. Disponível em:<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber= 7525798>. Acesso em: 21 ago. 2017.

STROUSTRUP, B. **The C++ programming language**. 3. ed. Murray Hill, New Jersey: AT& T Labs, 1997. 912 p.

VITHLANI, R. et al. An open source real time IoT based environmental sensor monitoring system. In: INTERNATIONAL CONFERENCE ON RESEARCH AND INNOVATIONS IN SCIENCE, ENGINEERING & TECHNOLOGY , 2., 2017, Vallabh Vidyanagar. **Anais Eletrônicos...** Vallabh Vidyanagar: Kalpa Publications in Computing , 2017. Disponível em: <https://easychair.org/publications/ open/pMKf>. Acesso em: 21 nov. 2017.

APÊNDICE A – DECLARAÇÃO DAS BIBLIOTECAS E VARIÁVEIS

```
1  // Trabalho de Graduacao Pedro Cannavale Graca
2  // Orientador Professor Doutor Teofilo Miguel de
     Souza
3  #include <SPI.h>
4  #include <ESP8266WiFi.h>
5  #include <ThingerWifi.h>
6  #include "DHT.h"
7  #include <math.h>
8
9  const char* nome_redewifi = "CentroDeEnergias";
10 const char* senha_redewifi = "ABCDEFGH";
11
12 #define USERNAME "TGPEDROC"
13 #define DEVICE_ID "tg131322869"
14 #define DEVICE_CREDENTIAL "ABCDEFGH"
15 #define DHTTYPE DHT22
16 #define DHTPIN 4
17
18 DHT dht(DHTPIN, DHTTYPE);
19
20 int valorldr = 0;
21 float voutldr = 0;
22 float rldr = 0;
23 float ldrluxc = 0;
24 float temperatura = 0;
25 float umidade = 0;
26
27 ThingerWifi thing(USERNAME, DEVICE_ID,
     DEVICE_CREDENTIAL);
28 WiFiServer server(80);
```

APÊNDICE B – FUNÇÃO SETUP

```
1  void setup() {
2
3  dht.begin();
4  thing.add_wifi(nome_redewifi, senha_redewifi);
5  Serial.begin(115200);
6  delay(10);
7
8  Serial.println("");
9  Serial.println("");
10 Serial.print("Conectando a ");
11 Serial.print(nome_redewifi);
12
13 WiFi.begin(nome_redewifi, senha_redewifi);
14
15   thing["Sensores"] >> [](pson& out){
16     out["Temperatura"] = temperatura;
17     out["Umidade"] = umidade;
18     out["Luminosidade"] = ldrluxc;
19     };
20
21 while (WiFi.status() != WL_CONNECTED) {
22 delay(500);
23 Serial.print(".");
24 pinMode(5, OUTPUT);
25 digitalWrite(5, 1);
26 }
27 pinMode(5, OUTPUT);
28 digitalWrite(5, 0);
29 Serial.println("");
30 Serial.print("Conectado a rede sem fio ");
```

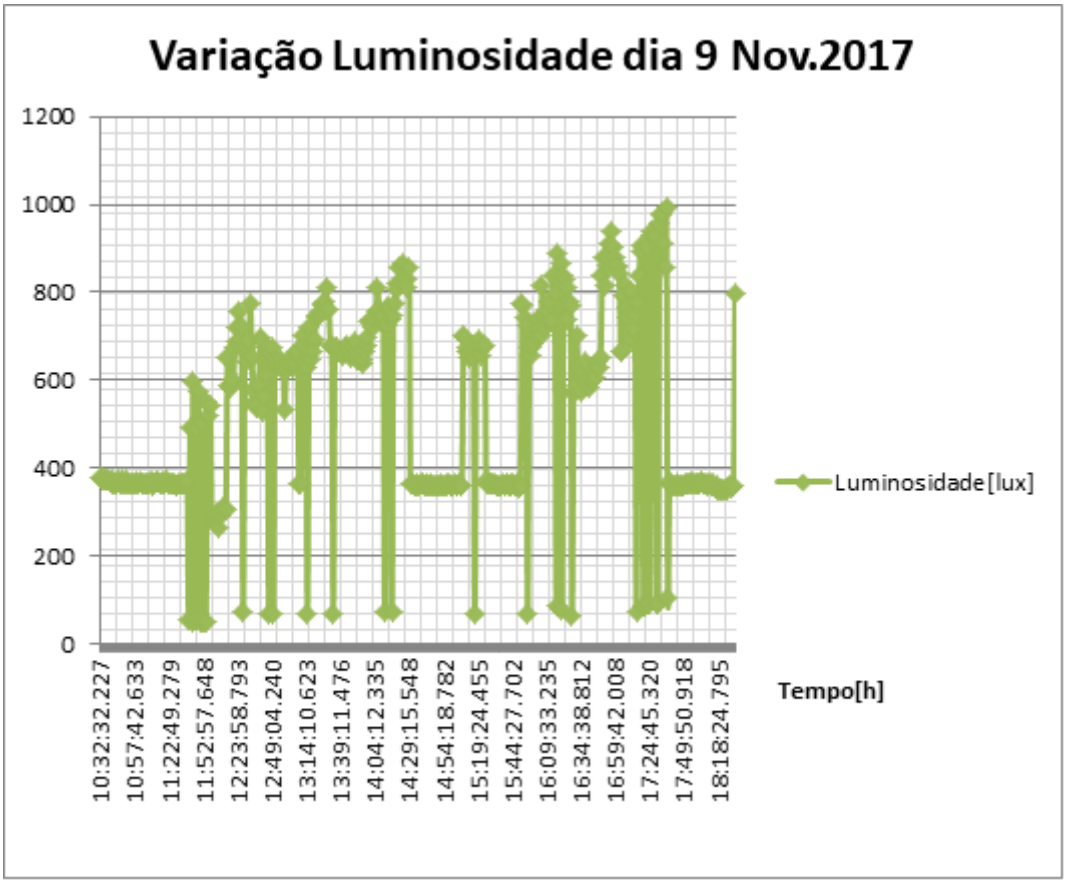
```
31 Serial.println(nome_redewifi);
32 server.begin();
33 Serial.println("Servidor iniciado");
34
35 Serial.print("IP para realizar a conexao ao servidor
     da placa NodeMCU: ");
36 Serial.print("http://");
37 Serial.println(WiFi.localIP());
38 }
```

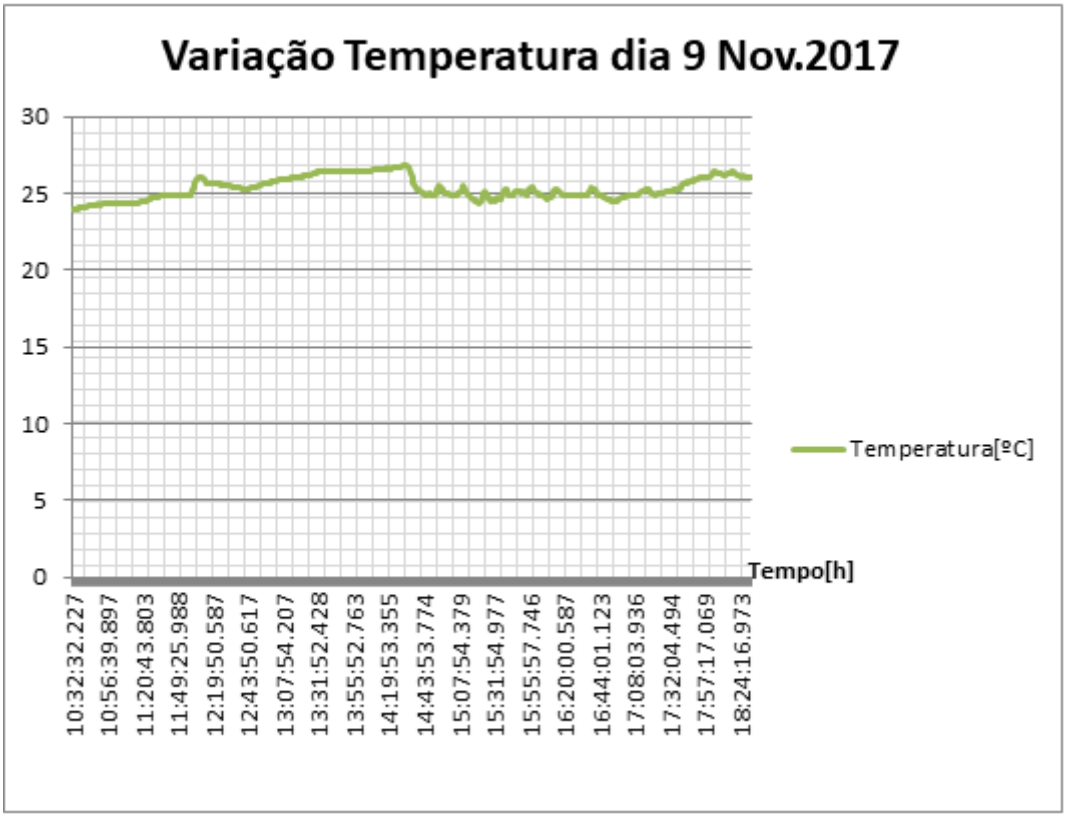
APÊNDICE C – FUNÇÃO LOOP

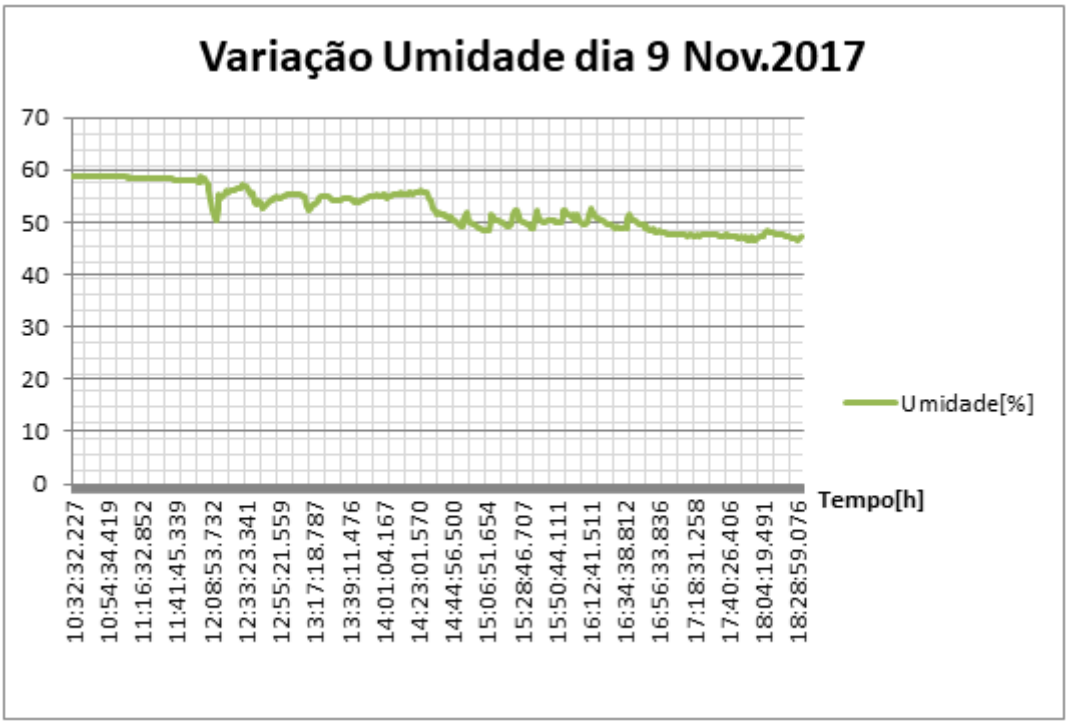
```
1  void loop() {
2
3  valorldr = analogRead(A0);
4  voutldr = 0.00322265625 * valorldr;
5  rldr = (9860 * (3.3 - voutldr))/ voutldr;
6  ldrluxc = pow(10,5.0459) * (pow(rldr,-0.8729));
7  temperatura = dht.readTemperature();
8  umidade = dht.readHumidity();
9
10 delay(3000);
11
12 thing.handle();
13 WiFiClient client = server.available();
14 if (!client) {
15 return;
16 }
17 Serial.println("Novo cliente se conectou!");
18 while(!client.available()){
19 delay(1);
20 }
21 while (WiFi.status() != WL_CONNECTED) {
22 delay(500);
23 Serial.print(".");
24 pinMode(5, OUTPUT);
25 digitalWrite(5, 1);
26 }
27 pinMode(5, OUTPUT);
28 digitalWrite(5, 0);
29 String request = client.readStringUntil('\r');
30 Serial.println(request);
```

```
31 client.flush();
32 client.println("HTTP/1.1 200 OK");
33 client.println("Content-Type: text/html");
34 client.println("");
35 client.println("<!DOCTYPE HTML>");
36 client.println("<html>");
37 client.println("<head>\n<meta charset='UTF-8'>");
38 client.println("<META http-equiv= refresh CONTENT=3
     URL=>");
39 client.println("<title>TG Pedro Cannavale</title>");
40 client.println("</head>");
41 client.println("<h1><center>Trabalho de Graduacao
     Pedro Cannavale Graca</center></h1>");
42 client.println("<h2><center>RA:131322869</center></h1
     >");
43 client.println("<center><font size='5'>Orientador:
     Professor Doutor Teofilo Miguel de Souza</center>"
     );
44 client.print("<center><font size='3'>Luminosidade: ")
     ;
45 client.print(ldrluxc);
46 client.println("lux");
47 client.println("</center></font>");
48 client.println("</html>");
49 delay(1);
50 Serial.println("Cliente desconectado");
51 Serial.println("");
52 }
53  }
```

APÊNDICE D – DADOS COLETADOS DIA 9 NOV. 2017






Variação Umidade dia 9 Nov.2017
70
60
50
40
30
20
10
0
10:32:32.227
10:54:34.419
11:16:32.852
11:41:45.339
12:08:53.732
12:33:23.341
12:55:21.559
13:17:18.787
13:39:11.476
14:01:04.167
14:23:01.570
14:44:56.500
15:06:51.654
15:28:46.707
15:50:44.111
16:12:41.511
16:34:38.812
16:56:33.836
17:18:31.258
17:40:26.406
18:04:19.491
18:28:59.076
Tempo[h]
Umidade[%]